SECURE COMMUNICATION USING AUTHENTICATED CHANNELS

THÈSE N° 4452 (2009)

PRÉSENTÉE À LA FACULTÉ INFORMATIQUE ET COMMUNICATIONS

Institut de systèmes de communications

SECTION DES SYSTÈMES DE COMMUNICATION

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

POUR L'OBTENTION DU GRADE DE DOCTEUR ÈS SCIENCES

PAR

Sylvain PASINI

ingénieur en systèmes de communications diplomé EPF
de nationalité suisse et originaire de Lancy (GE)

acceptée sur proposition du jury:

Prof. B. Faltings, président du jury
Prof. S. Vaudenay, directeur de thèse
Prof. A. Lenstra, rapporteur
Prof. K. Nyberg, rapporteur
Prof. D. Pointcheval, rapporteur

Lausanne, EPFL
2009

To my son, Matteo, and to my wife, Nadia.

Abstract

Our main motivation is to design more user-friendly security protocols. Indeed, if the use of the protocol is tedious, most users will not behave correctly and, consequently, security issues occur. An example is the actual behavior of a user in front of an SSH certificate validation: while this task is of utmost importance, about 99% of SSH users accept the received certificate without checking it. Designing more user-friendly protocols may be difficult since the security should not decrease at the same time. Interestingly, insecure channels coexist with channels ensuring authentication. In practice, these latters may be used for a string comparison or a string copy, e.g., by voice over IP spelling. The shorter the authenticated string is, the less human interaction the protocol requires, and the more user-friendly the protocol is. This leads to the notion of SAS-based cryptography, where SAS stands for Short Authenticated String.

In the first part of this thesis, we analyze and propose optimal SAS-based message authentication protocols. By using these protocols, we show how to construct optimal SAS-based authenticated key agreements. Such a protocol enables any group of users to agree on a shared secret key. SAS-based cryptography requires no pre-shared key, no trusted third party, and no public-key infrastructure. However, it requires the user to exchange a short SAS, e.g., five decimal digits. By using the just agreed secret key, the group can now achieve a secure communication based on symmetric cryptography.

SAS-based authentication protocols are often used to authenticate the protocol messages of a key agreement. Hence, each new secure communication requires the interaction of the users to agree on the SAS. A solution to reduce the user interaction is to use digital signature schemes. Indeed, in a setup phase, the users can use a SAS-based authentication protocol to exchange long-term verification keys. Then, using digital signatures, users are able to run several key agreements and the authentication of protocol messages is done by digital signatures. In the case where no authenticated channel is available, but a public-key infrastructure is in place, the SAS-based setup phase is avoided since verification keys are already authenticated by the infrastructure.

In the second part of this thesis, we also study two problems related to digital signatures:

(1) the insecurity of digital signature schemes which use weak hash functions and (2) the privacy issues from signed documents.

Digital signatures are often proven to be secure in the random oracle model. The role of random oracles is to model ideal hash functions. However, real hash functions deviate more and more from this idealization. Indeed, weaknesses on hash functions have already been discovered and we are expecting new ones. A question is how to fix the existing signature constructions based on these weak hash functions. In this thesis, we first try to find a better way to model weak hash function. Then, we propose a (randomized) pre-processing to the input message which transforms any weak signature implementation into a strong signature scheme. There remains one drawback due to the randomization. Indeed, the random coins must be sent and thus the signature enlarges. We also propose a method to avoid the increase in signature length by reusing signing coins.

Digital signatures may also lead to privacy issues. Indeed, given a message and its signature, anyone can publish the pair which will confirm the authenticity of the message. In certain applications, like in electronic passports (e-passports), publishing the authenticated data leads to serious privacy issues. In this thesis, we define the required security properties in order to protect the data privacy, especially in the case of e-passport verification. The main idea consists for the e-passport to keep the signature secret. The e-passport should only prove that it knows a valid signature instead of revealing it. We propose a new primitive, called Offline Non-Transferable Authentication Protocol (ONTAP), as well as efficient implementations that are compatible with the e-passport standard signature schemes.

Keywords: cryptography, message authentication protocol, short authenticated string, SAS, key agreement protocol, digital signature, voice over IP, hash-and-sign paradigm, weak hash function, offline non-transferable authentication protocol, ONTAP, electronic passport.

Résumé

La motivation principale de ce travail est de concevoir des protocoles de sécurité restant simples d'utilisation. Si l'application demande des tâches trop importantes, les utilisateurs ne se comporteront pas correctement et cela engendrera des problèmes de sécurité. Un exemple est le comportement actuel d'un utilisateur lorsqu'il établit une connexion SSH. Il est sensé authentifier la clé publique qu'il reçoit. Cette opération nécessite l'obtention d'une empreinte de la clé distante ce qui est, en général, trop imposant. Au final, 99% des utilisateurs acceptent simplement la clé publique du serveur sans même la contrôler. La conception d'un protocole plus simple d'utilisation n'est pas une chose facile étant donné le niveau de sécurité escompté. Une chose intéressante est la coexistence des canaux de communication non sécurisés avec des canaux qui permettent d'authentifier des données. En pratique, ces derniers peuvent être une simple comparaison de deux chaînes, la copie d'un nombre d'un appareil à un autre, ou encore la diction par téléphone. Clairement, plus la chaîne à authentifier est courte, moins l'utilisateur aura de travail et plus le protocole sera simple d'usage. Ceci nous amène à la notion de cryptographie basée sur les SAS (en anglais, *SAS-based cryptography*). Le terme SAS provient de *Short Authenticated String* qui signifie chaîne authentifiée courte.

La première partie de cette thèse est consacrée à l'analyse des protocoles d'authentification de messages basés sur les SAS. En particulier, nous analyserons leur sécurité de manière globale puis nous proposerons plusieurs protocoles optimaux. En utilisant ces protocoles, nous montrerons comment construire des protocoles d'échanges de clés authentifiés également basés sur les SAS. Ce type de protocoles permet à un groupe d'utilisateurs de se mettre d'accord sur une clé secrète. L'avantage de la cryptographie basée sur les SAS est qu'elle ne nécessite pas d'information préétablie, ni de faire confiance à une tierce personne, ni d'infrastructure à clés publiques. Par contre, elle demande à l'utilisateur d'échanger de manière authentifiée une courte chaîne, par exemple un SAS de cinq chiffres. En utilisant cette clé secrète, le groupe pourra communiquer de manière sécurisée en utilisant un algorithme de chiffrement conventionnel.

Les protocoles d'authentification de messages basés sur les SAS sont souvent utilisés pour

authentifier les messages d'un protocole d'échange de clé. Par conséquent, chaque nouvelle communication nécessite l'intervention des utilisateurs pour échanger le SAS. Une solution permettant de réduire la charge des utilisateurs est l'utilisation de signatures digitales. En effet, dans une phase d'initialisation, il est possible d'utiliser le protocole d'authentification de messages pour échanger des clés publiques permanentes permettant la vérification de signatures digitales. Après cela, les utilisateurs peuvent exécutés plusieurs protocoles d'échange de clés et l'authentification sera faite grâce aux signatures digitales. Dans le cas où il n'y aurait pas de canal authentifié disponible, mais une infrastructure à clés publiques, la phase d'initialisation peut être évitée puisque les clés de vérification sont déjà authentifiées par l'infrastructure.

La seconde partie de cette thèse est consacrée à deux problèmes liés aux signatures digitales : (1) l'insécurité des signatures digitales qui utilisent des fonctions de hachage faibles et (2) la violation de la sphère privée résultant de documents signés.

Les signatures digitales sont souvent prouvées sûres dans le modèle de l'oracle aléatoire. Le rôle de l'oracle aléatoire est de modéliser des fonctions de hachage. Malheureusement, les fonctions de hachage réelles dévient de plus en plus de cette idéalisation. En effet, des faiblesses ont déjà été découvertes et ce ne sont certainement pas les dernières. La réparation des signatures implémentées avec ces faibles fonctions de hachage est donc un problème. Dans cette thèse, nous rechercherons un meilleur modèle pour les fonctions de hachage. Après cela, nous proposerons un pré-traitement pour le message qui permettra de transformer une implémentation de signature faible en une résistante. Il restera un inconvénient: la longueur de la signature accroît dû à l'aléa ajouté dans le pré-traitement. Nous proposerons également une méthode pour éviter cette allongement de signature en recyclant l'aléa de l'algorithme de signature.

Les signatures digitales peuvent également engendrer des problèmes de sphère privée. En effet, n'importe qui possédant un message ainsi que sa signature peut les publier, la signature confirme l'authenticité du message. Dans certains cas, comme pour les passeports électroniques, la publication des données authentifiées (et personnelles) engendre de sérieux problèmes de sphère privée. Dans cette thèse, nous définirons les propriétés nécessaires pour protéger la sphère privée, spécialement pour le cas des passeports électroniques. L'idée principale pour le passeport est de prouver qu'il possède la signature des données tout en la conservant secrète. Nous proposerons une nouvelle primitive nommée *Offline Non-Transferable Authentication Protocol (ONTAP)*. Nous proposerons également des implémentations efficaces qui sont compatibles avec les standards de signatures utilisés par les passeports électroniques.

Mots clés : cryptographie, protocole d'authentification de messages, short authenticated string, SAS, protocole d'échange de clés, voix sur IP, signature digitale, le paradigme hash-and-sign, fonction de hachage faible, protocole d'authentification non transférable, ONTAP, passeport électronique.

Contents

Acknowledgments

My attraction for security, particularly for cryptography, was developed during my studies. To understand how I got here today, it is necessary to quickly overview my route: At the end of the normal school, I was directly admitted at the Engineering School of Geneva (EIG). I had the chance to explore different areas of interest, such as computer science, engineering or physics, and I found that all seemed "logical" for me. The most mysterious part remained the cluster of electronic components in a device. Indeed, when we open one, we immediately think "how is it possible to understand what is happening there?". An engineer is not only able to understand, but more, he is able to design it. So, when I entered in the HES cycle, I chose the electronic field. At the end of the EIG, I did my diploma work in image processing with Prof. Michel Kocher. Michel changed my academic career by convincing me to continue my studies at the EPFL. During my studies at the EIG, I learned a lot about the functioning of a computer, in both hardware and software levels. The remaining mysterious point for me was the functioning of the Internet: how the Web works, how an email finds its way, and so on. So, at the EPFL, I chose to study communication systems. I discovered that the area was not so complicated and only one question remained: "how to ensure security?". Through the courses Network security, given by Dr. Philippe Oechslin, and Cryptography, given by Prof. Serge Vaudenay, I discovered a fascinating field. During my semester project and my master thesis, my interest for the field of security increased and I decided to pursue my career as teaching and research assistant in the security and cryptography laboratory while doing my PhD thesis under the supervision of Prof. Serge Vaudenay.

Here, I would like to express here my gratitude to those who have marked my career, and those who supported me during these years.

I start with my PhD supervisor, Professor Serge Vaudenay, who offered me the chance to carry out my PhD thesis in the Security and Cryptography laboratory (LASEC). Everything started in March 2004 with the first course "Security and Cryptography". I would like to thank Serge for having introduced me to the fascinating field of cryptography. Cryptography is a mysterious world and I always appreciated how math and computer tools can be combined to build (or break) real systems. A cryptographer has a special way of thinking: every time, everywhere, he tries to find "holes in every wall". One example is the famous "replay attack" in a all you can eat restaurant. Indeed, it is possible to feed the entire table by paying only once, the first plate is used as a kind of ticket. Serge offered me the opportunity to discover the world of research. Thanks to his experience, his vivacity, his vast knowledge, his rigor, and his assistance, I was able to learn a huge amount of things, in cryptography of course, but also in other areas. Thank you Serge!

I thank Prof. Boi Faltings, the president of the jury, as well as Prof. Arjen Lenstra, Prof. Kaisa Nyberg, and Dr. David Pointcheval for having accepted to take time for reviewing this work. I also express my thanks to the the Swiss National Science Foundation (SNSF) which supported this research project (grant 200021-113329).

During these years, I had the chance to meet and to work with many colleagues. A PhD thesis is time limited. There was a kind of rotation among the PhD students in the lab: when a PhD student arrives, he is the newest, and as time goes, the old ones leave and the new ones take their place. In our laboratory, the LASEC, there was always a positive and dynamic atmosphere that was suitable for work, but also for fun. I want to thank all my former colleagues for their welcome and all the moments we have shared together, in order of arrival: Serge Vaudenay, Martine Corval, Pascal Junod, Philippe Oechslin, Gildas Avoine, Yi Lu, Jean Monnerat, Thomas Baignères, Claude Barral, Julien Brouchier, Matthieu Finiasz, and Martin Vuagnoux. I also thank all my colleagues who came after I did with whom I have shared great moments, in order of arrival: Raphael Phan, Khaled Ouafi, Raphaël Overbeck, Rafik Chaabouni, Jorge Nakahara Jr., Pouyan Sepehrdad, and Atefeh Mashatan. I believe that many of them deserve special thanks, in alphabetical order:

As in all reference sections, Gildas Avoine is the first one. I remember very good moments spent with my big friend "Gigi". I also remember our trip to New York filled with many anecdotes. I thank him for his friendship and for all his advice, especially when I arrived and I had difficulties with LaTeX. Between motorcyclists, the current is always on...

The next person who deserves special thanks is Thomas Baignères. Thomas became also one of my very good friends. Thomas was my supervisor during my semester project, he learned me the basics, and advised me to carry out my master thesis as well as my PhD thesis at the LASEC. Without him, I would not be part of the crypto world today. Thanks Thomas!

I also thank Rafik Chaabouni who recently arrived at the lab, for trying to make everyday

of our life more joyful. I think about the jokes he made for the April fool's day or birthdays, and also the ones he makes to people forgetting to lock their machines, and so on.

Our secretary Matine Corval deserves of course to be thanked. She was always friendly and smiling and it was a pleasure to share these years with her. She was like the "academic mother" of the lab. Her assistance was essential for all of us, especially for administrative tasks (to which I really don't understand anything).

I would especially like to thank Matthieu Finiasz for all his help. Thanks to his very good availability, his friendliness, his patience, and his knowledge, I had many opportunities to exchange ideas and to expose problems with him and I often came out of his office with a solution. I get along very well with him and we built a strong friendship.

Jean Monnerat, always of good mood and ready to defend Switzerland and his native canton (the beautiful Jura), gave openings to many animated debates. I thank him for his help, especially in mathematics and LaTeX. Jean also became for me a very good friend and I keep very good memories of moments spent with him.

I would thank Philippe Oechslin with whom I shared my office for over two years. It was an honor for me to share my office with Philippe. Philippe is also the person who introduced me to the world of security through its exciting course "network security". His course encouraged me to follow "Security and Cryptography" mentioned above.

The last person of the lab who deserves my special thanks is Martin Vuagnoux. I would like to thank him for being an excellent co-author, for being a very pleasant office-mate, and for the funny stories which made us laugh so much especially at the coffee breaks. Martin also became a very good friend and I guess that both of us will never forget the Vietnam story (it's a long story but Martin will understand). Thanks for all "Martino"!

I finally would like to thank a few virtual persons from the LASEC and theirs authors would certainly recognize them, in alphabetical order: Diablotin, ElGringo, Kevin_Mitnick, kpts44, and Mojean_is_back.

I would like to thank France Faille, the secretary of the neighbor laboratory. During the years, we had the chance to get to know each other better and I would say that France is one of the most kindly and the most friendly person that I know. I especially thank her for her good mood and her support, but also for its assistance during various administrative matters.

I would like to thank Sven Laur, also known as Swen, with whom I wrote two papers. He was a very pleasant and very competent co-author. By coincidence we share the same hobby: model airplanes.

Dr. Michel Kocher, my supervisor during my first diploma at the Engineering School of Geneva, who motivated, and encouraged me also deserves my thanks. I would like to thank

him for his collaboration during my months of diploma work and for advising me to continue my studies at the EPFL. Without him, I would never have known the EPFL.

I thank those who helped me to improve the readability of this thesis, namely: Martine Corval, France Faille, Atefeh Mashatan, Jorge Nakahara Jr., Serge Vaudenay and Martin Vuagnoux. Special thanks to Jorge who did a great job.

My parents should obviously not be forgotten. Thanks to them, I had the chance (amongst many other things!) to get an excellent education and I greatly thank them. I also would like to thank my wife, Nadia. No matter when and why, she always kept encouraging me and took it upon herself to leave me enough spare time to conclude this thesis (even if it sometimes required a few negotiations!). My final thanks go to my son, Matteo. I share moments of intense happiness with him and this gives me everyday a huge motivation. I finally thank all four of them for their Love which is essential for me to live.

Thanks to everyone!

Sylvain

After this pleasant period spent within the LASEC, I started a new adventure: working as cryptographer at Nagravision (a Kudelski group company). I thank all my new collegues for their warm welcome, especially Olivier Brique, Nicolas Fischer, Pascal Junod, Alexander Karlov, and Karl Osen.

Remerciements

Mon attirance pour la sécurité informatique, et plus particulièrement pour la cryptographie s'est développée au fil de mes études. Afin de comprendre comment je me suis retrouvé ici aujourd'hui, il est nécessaire de survoler rapidement mon parcours : À la fin de ma scolarité obligatoire, j'ai directement débuté des études à l'Ecole d'Ingénieurs de Genève (EIG). Au début, nous avons la chance de pouvoir explorer différents domaines intéressants, comme l'informatique, la mécanique, ou encore la physique. J'ai pu constater que tous me semblaient "logiques". La partie la plus mystérieuse pour moi restait l'amas de composants électroniques contenu dans un appareil. En effet, lorsqu'on en ouvre un, on pense immédiatement "comment est-ce possible de comprendre ce qui se passe là-dedans ?". Un ingénieur est non seulement capable de le comprendre, mais plus encore, il est capable de l'inventer. Lors de mon entrée dans le cycle HES, j'ai donc choisi l'électronique afin d'éclaircir ce domaine obscur. À la fin de mon cycle, j'ai réalisé mon travail de diplôme en traitement d'images avec le Prof. Michel Kocher. Michel a en quelque sorte bouleversé mon parcours académique en me poussant à poursuivre mes études à l'EPFL. La réussite des examens d'admission de l'EPFL m'a permis d'entrer en 3ème année ainsi que de changer de section. Lors de mes années passées à l'EIG, j'ai pu apprendre beaucoup sur le fonctionnement d'un ordinateur, autant au niveau matériel qu'au niveau logiciel. Le point restant mystérieux pour moi était le fonctionnement d'Internet : comment le Web fonctionne, comment un email trouve son chemin, etc. A l'EPFL, j'ai donc choisi d'étudier les systèmes de communication. J'ai découvert que le domaine n'était pas aussi compliqué qu'il en avait l'air à l'origine. Seule une interrogation persistait : "comment garantir la sécurité ?". Grâce au cours de sécurité des réseaux, donné par le Dr. Philippe Oechslin, puis au cours de cryptographie, donné par le Prof. Serge Vaudenay, j'ai découvert un domaine fascinant. Lors de mon projet de semestre et de mon travail de master, mon intérêt pour le domaine de la sécurité s'est accentué encore et j'ai décidé de poursuivre ma carrière académique en tant qu'assistant-doctorant dans le laboratoire de sécurité et de cryptographie (LASEC) du Prof. Serge Vaudenay.

Je tiens ici à exprimer ma reconnaissance aux personnes qui ont marquées mon parcours, ainsi qu'à celles qui mon soutenues pendant ces années.

Pour commencer, je tiens à exprimer ma gratitude à mon superviseur de thèse, le Professeur Serge Vaudenay, qui m'a donnée la chance de réaliser ma thèse dans le laboratoire de Sécurité et Cryptographie (LASEC). Mon premier contact avec Serge a eu lieu en mars 2004 durant le cours "Sécurité et Cryptographie". Je voudrais d'abord le remercier de m'avoir fait découvrir ce domaine fascinant qu'est la cryptographie. La cryptographie est un monde mystérieux et j'ai toujours apprécié la façon dont les outils mathématiques et informatiques peuvent être associés afin de construire (ou de casser) des systèmes réels. Un cryptographe a une façon bien à lui de penser : toujours et partout, il essaiera de trouver "un trou dans chaque mur". Un exemple est la célèbre "attaque par rejeu" dans un restaurant permettant de manger à volonté. En effet, il est possible de nourrir l'ensemble de la table en payant qu'une seule fois, l'assiette servant en quelque sorte de ticket. Serge m'a offert l'opportunité de découvrir le monde de la recherche. Grâce à son expérience, sa vivacité, ses connaissances, sa rigueur et son aide, j'ai pu apprendre énormément de choses, en cryptographie bien sur, mais aussi dans divers autres domaines. Merci Serge !

Je remercie le Prof. Boi Faltings, président du jury, ainsi que le Prof. Arjen Lenstra, la Prof. Kaisa Nyberg et le Dr. David Pointcheval d'avoir accepté de consacrer une partie de leur temps pour relire ce travail. Je remercie également le Fond national suisse (FNS) de la recherche scientifique qui a soutenu financièrement ce projet (subvention 200021-113329).

Durant ces années passées au LASEC, j'ai eu la chance de rencontrer et de travailler avec beaucoup de collègues. Les thèses ayant une durée limitée, il y a eu une sorte de tournus parmi les doctorants. Lorsqu'un thésard arrive, il est le nouveau, et au long des années, les anciens partent pour laisser la place aux nouveaux. Dans notre laboratoire, il y a toujours régné une ambiance et une dynamique qui étaient propices au travail, mais aussi à la détente. Je tiens à remercier tous mes anciens collègues pour leur accueil ainsi que pour tous les moments que nous avons partagés ensembles, par ordre d'arrivée : Serge Vaudenay, Martine Corval, Pascal Junod, Philippe Oechslin, Gildas Avoine, Yi Lu, Jean Monnerat, Thomas Baignères, Claude Barral, Julien Brouchier, Matthieu Finiasz, and Martin Vuagnoux. Je remercie également tous mes collègues arrivés après moi avec qui j'ai également partagés d'excellents moments, par ordre d'arrivée : Raphael Phan, Khaled Ouafi, Raphaël Overbeck, Rafik Chaabouni, Jorge Nakahara Jr., Pouyan Sepehrdad, and Atefeh Mashatan. Je pense que certains méritent des remerciements particuliers, par ordre alphabétique :

Comme dans tous ses papiers, Gildas Avoine est le premier. Je me souviens de beaucoup de bons moment avec mon ami "Gigi", en particulier notre séjour à New York remplit d'anecdotes. Je le remercie pour son amitié et pour ses conseils, spécialement lorsque je suis arrivé et que je n'étais pas encore très familier avec LaTeX. Après tout, le courant passe toujours entre motards...

Thomas Baignères mérite également quelques remerciements. Thomas, ou Tomtom pour les intimes, m'a supervisé pendant mon projet de semestre, il m'a en quelque sorte appris les bases. Il m'a ensuite conseillé de faire mon travail de master, puis, plus tard, ma thèse au

LASEC. Sans lui, je ne ferais pas partie aujourd'hui du monde de la crypto. Merci Thomas !

J'aimerais aussi remercier Rafik Chaabouni, récemment arrivé dans le labo. Rafik essayait chaque jour de rendre notre vie plus joyeuse. Je pense en particulier à ses farces lors du premier avril ou lors d'anniversaires, à ses farces adressées aux personnes qui par mégarde auraient oublié de verrouiller leur machine, et j'en passe.

Bien évidemment, je ne peux oublier de remercier Martine Corval, notre secrétaire. Martine a toujours été aimable et souriante et ça a été pour moi un plaisir de partager ces années avec elle. On pourrait dire qu'elle a été la "maman académique" du laboratoire, nous encourageant ou nous remontant le moral lorsque nous en avions besoin. Son aide a été essentielle pour nous tous, en particulier pour les tâches administratives (pour lesquelles je suis vraiment perdu).

J'aimerais remercier très spécialement Matthieu Finiasz. Grâce à sa disponibilité, son amabilité, sa patience, et ses compétences, j'ai eu beaucoup d'opportunités pour apprendre, pour exposer mes problèmes, pour échanger des idées, et très souvent je ressortais de son bureau avec une solution. Je garde également de bons souvenirs lors de nos séjours à Barcelone et à Paris. En dehors de la vie académique, Matthieu et moi avons construit une Amitié qui à mes yeux est l'une des plus importantes.

Jean Monnerat, toujours de bonne humeur, était toujours prêt à défendre sa nation, la Suisse, ou son canton, le magnifique Jura, ce qui a occasionné de nombreux débats animés. Je remercie Jean pour son aide, particulièrement en mathématiques et en LaTeX. Jean est également devenu pour moi un grand Ami et je garde de très bons souvenirs de moments passés avec lui.

J'aimerais remercier Philippe Oechslin qui a été mon collègue de bureau pendant plus de deux ans. C'était pour moi un honneur de partager un bureau avec lui. Philippe est également la personne qui m'a fait découvrir le monde de la sécurité informatique au travers de son cours "sécurité des réseaux". Son cours était tellement passionnant qu'il m'a beaucoup encouragé à suivre le cours "Sécurité et Cryptographie" du Prof. Serge Vaudenay comme expliqué plus haut. Merci Philippe pour ton cours si passionnant !

La dernière personne du laboratoire que je souhaite remercier ici est Martin Vuagnoux. Je le remercie pour avoir été un excellent co-author, d'avoir été un agréable collègue de bureau, et de nous avoir raconté autant d'histoires marrantes qui nous ont fait tant rigoler, surtout aux pauses cafés. Martin est aussi devenu un très bon Ami et je parie qu'aucun de nous deux oubliera la fameuse histoire du Vietnam (c'est une longue histoire, mais Martin comprendra). Merci pour tout "Martino" !

Je remercie égalements quelques personnages virtuels pour les bons moments partagés ensembles. Leurs auteurs les reconnaîtront très certainement, par ordre alphabétique : Diablotin, ElGringo, Kevin_Mitnick, kpts44, and Mojean_is_back.

J'aimerais remercier France Faille, la secrétaire du laboratoire avec qui nous partagions l'étage. Au long de ces années, nous avons eu la chance de faire plus ample connaissance et je dirais que France est l'une de personne les plus amicales et les plus serviables que je connaisse. Je la remercie spécialement pour les agréables moments que nous avons partagés, pour son soutien ainsi que pour son aide lors de diverses tâches administratives.

Je voudrais remercier Sven Laur, ou plutôt Swen, avec qui j'ai écris plusieurs articles. Il a été un agréable co-auteur et nous avons pu échanger beaucoup d'idées. De plus, par coïncidence, nous partageons la même passion: l'aéromodélisme !

Dr. Michel Kocher, mon superviseur durant mon travail de diplôme à l'Ecole d'Ingénieurs de Genève, mérite un grand Merci. Il m'a encouragé et motivé à poursuivre mes étude à l'EPFL. Sans lui, je n'aurais jamais connu l'EPFL...

Je remercie également les personnes qui m'ont aidées à améliorer la rédaction de cette thèse, soit : Martine Corval, France Faille, Atefeh Mashatan, Jorge Nakahara Jr., Serge Vaudenay ainsi que Martin Vuagnoux. Un merci tout particulier à Jorge qui a fait un travail formidable.

Mes parents ne doivent évidemment pas être oubliés. Grâce à eux, j'ai eu la chance (parmi bien d'autres choses!) de pouvoir réaliser une excellente formation et je les remercie énormément. Je tiens également à remercier mon épouse, Nadia. Peu importe quand et pourquoi, elle m'a toujours encouragé et a pris sur elle afin de me laisser assez de temps libre pour conclure cette thèse (même si parfois cela a demandé quelques négociations). Mes derniers remerciements vont à mon fils, Matteo. Je partage des moments de bonheur intense avec lui et cela me donne chaque jours une grande motivation. Je tiens finalement à les remercier tous les quatre pour leur Amour qui est essentiel pour moi.

Merci à tous !

Sylvain

Après cette agréable période passée au sein du LASEC, j'ai débuté une nouvelle aventure: travailler comme cryptographe chez Nagravision (une compagnie du Groupe Kudelski). J'en profite pour remercier mes nouveaux collègues pour leur accueil chaleureux, en particulier Olivier Brique, Nicolas Fischer, Pascal Junod, Alexander Karlov et Karl Osen.

Chapter

ONE

Introduction

One of the most important goals in cryptography is to establish a secure communication channel between two or more parties. Parties may be human beings involved in confidential voice over IP calls, mobile phones exchanging sensitive data, a user connecting to a bank web site, or a border patrol checking an electronic passport (e-passport). Parties are usually connected through insecure channels, for instance, a wireless link, or the Internet. Thus, the "raw data" are clearly insecure. The goal of a cryptographer is to design a system which establishes this secure communication over any insecure channel. The term *secure* means that the communication should be confidential, i.e., nobody except the parties involved have access to the information. The communication should also ensure authenticity and integrity, i.e., the recipient is ensured that the information was sent as-is and is ensured about who sent it.

Fortunately, symmetric cryptography, also known as conventional or secret-key cryptography, exists. It enables to establish a secure communication in cases where a secret key is shared between the parties. Let all parties involved in a secure communication know a shared key sk. Any secure cipher with sk will ensure a secure communication. An essential condition is to keep sk secret, i.e., nobody except the involved parties know sk. Therefore, to establish a secure channel, it is enough for the parties to exchange (or to agree) on a shared secret key.

In short, setting up a secure communication over an insecure channel can be summarized by two sub-goals as depicted on Figure 1.1. Firstly, a key should be exchanged (or agreed)

between all parties in a secret way. This first phase is often called *key agreement*. Secondly, in order to keep the key secret, the parties involved in the key agreement should be authenticated (otherwise, the key could be leaked to an adversary and becomes no longer secret).

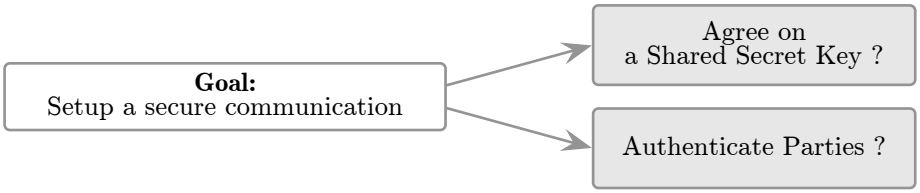


Figure 1.1. *Setting up a Secure Communication Split in Two Goals.*

With no assumption, parties must exchange a secret key. To achieve that goal, they need to use a secure extra channel that ensures confidentiality, authenticity, and integrity at least once before setting up the communication. Such a channel is very expensive in practice because participants must be physically close while the key is exchanged. Indeed, suppose two parties located far away, say Alice in Athens and Bob in Buenos Aires, wanting to setup a secure call. At that time, they do not share any secret and the only way to exchange a secret key is to encounter. Phone, mail, and email, all are not secure. The key establishment is clearly impossible in that case (or at least expensive or slow). Therefore, the authentication of parties, as well as the key agreement, are impossible since everything is insecure. To achieve our goal, we must make some additional assumptions. Different assumptions lead to different models and they may be incomparable due to their very different settings. As depicted in Figure 1.2, we will briefly present models considering a pre-shared key, a public-key infrastructure, and an extra authenticated channel.

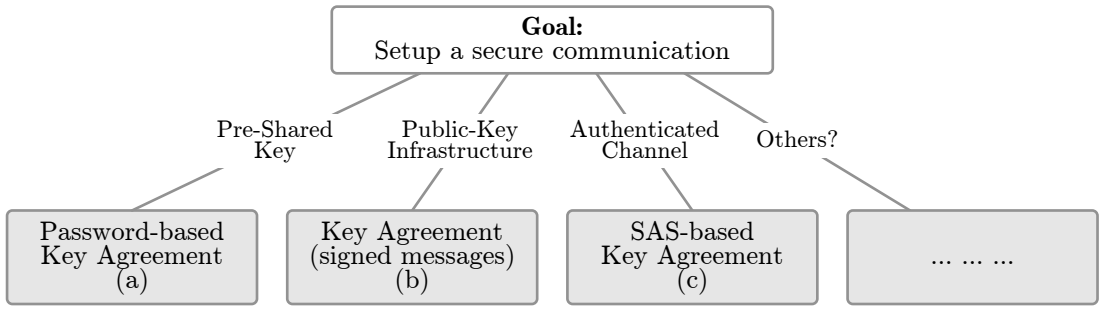


Figure 1.2. *Setting up a Secure Communication According to the Assumptions.*

One can assume that all parties involved in the protocol already share a secret key, often called a password. This is illustrated as case (a) in Figure 1.2. Passwords are often of low entropy since human beings should remember them. Using a password directly to encrypt

the communication is not a good idea. Indeed, the low entropy allows brute force attacks and the password will eventually leak. The password is used to run a key agreement in order to get a session key between involved parties. One drawback with this setting is the password assumption. It means that at some time in the past, all parties exchanged a password secretly. The only gain compared to the previous solution is that the secure channel is used to exchange a smaller amount of data, e.g., a low entropy password instead of a high entropy key.

The cryptographic world was revolutionized in 1976 with the discovery of public-key cryptography. Indeed, Diffie and Hellman [DH76] proposed a secure way to establish a key between two parties under the standard complexity-theoretic assumptions. However, the Diffie-Hellman protocol is insecure against man-in-the-middle attacks. Their key establishment protocol requires authenticated links (the confidential assumption is relaxed). We emphasize that transcripts of common key agreement protocols are usually several thousands bits long and, thus, message authentication is a non-trivial task. Of course, we can use message authentication codes but this requires a shared secret key that we are only trying to establish. Alternatively, we can use digital signatures, but they also require authentic transfer of public keys. Finally, the authentication of the protocol transcript may be done by using an extra authenticated channel. Another example to establish a secret key is the use of any secure public-key cryptosystem such as RSA [RSA78] or ElGamal [ElG85]. For the encryption, anyone knowing the public key can encrypt a plaintext, but only the owner of the corresponding private key will be able to decrypt it. In this case, we must insure that the public key is transferred to the participants in an authenticated way. In a nutshell, the use of public-key primitives can relax the confidential assumption on the extra channel and, thus, setting up a secure communication can be reduced to the problem of message authentication. Indeed, thanks to message authentication, parties are able to establish a shared secret key while its confidentiality, its authenticity, and its integrity is ensured among all parties.

Claim 1.1.

As long as parties are able to authenticate data, they can establish a shared secret key. As a consequence, they can also protect communication over insecure channels.

A first solution to authenticate messages is to assume the existence of a public-key infrastructure. This is illustrated as case (b) in Figure 1.2. Indeed, once a public key is authenticated, it is possible to authenticate the signed messages by checking signatures. An infrastructure is composed of certification authorities. Each authority posses a key pair: a private key and a public key. The authority outputs certificates for public keys ensuring that a given public key is bound to a given identity (as well as validity date and others). An end-user knowing the authority's public key is able to check the validity of any certificate produced by that authority. Usually, the public key of many authorities are setup before, typically they are embedded in web browsers. Thanks to the public key authentication, the end-user can use it to verify message signatures (i.e., message authentication) or to encrypt

information that only the recipient will be able to decrypt. This solution requires a huge infrastructure which is expensive and forces the end-user to trust the certificate authorities (who can certify wrong public keys) and to trust the ring of public keys (typically the web browser).

Another solution to authenticate messages is to use a *message authentication protocol* relying on an extra authenticated channel (case (c) in Figure 1.2). Hence, many practical communication protocols such as SSH, PGP, Bluetooth, and WUSB use extra channels which achieve at least authentication. Indeed, in SSH and PGP, the public keys are authenticated with the help of the user who will check the fingerprints. When a user receives a PGP public key, he or she computes its fingerprint and then calls the claimed owner of the public key. If both fingerprints match, then the public key is authenticated, otherwise, the key has been altered.

This thesis is organized as follows. It begins with **Chapter 2** by giving a big picture on the cryptographic techniques. **Chapter 3** is devoted to give standard cryptographic definitions such as hash functions, random oracle, common reference string model, and commitment schemes. **Part I (Chapters 4-10)** and **Part II (Chapters 11-13)** respectively focus on message authentication protocols which use an extra authenticated channel and on signature schemes. More details on each part are given below. Finally, this thesis concludes with **Chapter 14**.

1.1 SAS-Based Cryptography (Part I)

In this part, we assume that parties can exchange information through an extra authenticated channel in addition to the insecure channel. Normally, the authenticated channel is established by a human operator. For instance, a user can establish it by doing relatively simple tasks, such as copying a string from one device to another or spelling a string on the phone. Indeed, such tasks create authenticated channels in practice, since no adversary controlling the network can forge these (out-of-band) messages. On the other hand, these protocols require the help of the user and, thus, they should request only small tasks in order to stay user-friendly. As a consequence, protocol designers should use the minimum amount of out-of-band data as possible to achieve the desired security level.

As explained above, two-party (authenticated) key agreement protocols may be built by using the Diffie-Hellman [DH76] key agreement. However, the messages should be authenticated to avoid man-in-the-middle attacks. For that reason, the whole protocol transcript is authenticated. Since the protocol transcript is typically of several thousands bits, the transcript itself cannot be sent over the authenticated channel. A message authentication protocol sends the message over the insecure channel and then uses an authenticated string as short as possible in order to validate the authenticity of the received messages. The same idea may be used to build key agreement for groups by using a group key agreement instead

of the Diffie-Hellman protocol.

Rivest and Shamir [RS84] were the first to propose human participation in an authentication protocol. The parties should be able to recognize their respective voices. Since then, the model has evolved. We assume that parties can communicate over two different types of channels. First, parties can communicate over an *insecure channel*. This channel has a high-bandwidth and is cheap. However, the adversary has full control on that channel. Indeed, we assume that he can eavesdrop, drop, modify, insert, forge, delay, and replay any messages. Second, the parties can communicate over an *authenticated channel*. This channel achieves authenticity and, thus, ensures the recipient of a message about who sent it. Integrity is implicitly provided. It has a low-bandwidth and is expensive. As a result, communication on that channel should be as light as possible. Remember that more communication leads to more user participation. The adversary can still eavesdrop, drop, delay, and replay any messages, but he cannot modify, insert, or forge them. This model was formally introduced by Vaudenay [Vau05b] and called SAS-based cryptography where SAS stands for Short Authenticated Strings. However, it is also known on different names, see Figure 1.3.

Manual channel in [GMN04, GN04, LN06a, LN06b, NSS06, RWSN07],

SAS-based cryptography, where SAS stands for Short Authenticated String, in [Vau05b, Pas05, PV06a, PV06b, LP08] and in this thesis,

Two-channel cryptography in [MS07, MS08, Mas08],

User-aided cryptography in [PV05, LP09].

Figure 1.3. *The Different Designations of SAS-based Cryptography.*

There are many applications of key agreement protocols and it is hard to enumerate them. For instance, SAS-based key agreement may be used to pair two Bluetooth devices, to securely print on a selected printer, to quickly setup keys in case of disaster or compromised infrastructure, to establish a secure Voice over IP call, etc. Note that it is becoming usual to add security in Voice over IP.

Balfanz *et al.* [BSSW02] were the first to formalize the fingerprint-based protocol. However, first non-trivial results were obtained by Gehrmann, Mitchell, and Nyberg [GMN04, GN04]. They showed how to construct SAS-based message authentication protocols that preserve reasonable security levels even for short authenticated messages consisting of 4–6 decimal digits. The latter made SAS-based data authentication practical for securing short-range wireless communication such as Bluetooth and wi-fi networks. The original SAS protocol proposed by Vaudenay [Vau05b] was the second important discovery. This protocol was the first to achieve an *optimal* security level with respect to the SAS length. Vaudenay

also introduced the concept of SAS-based cryptography to a wider cryptographic audience.

In **Part I**, we start with the formal security model in **Chapter 4**. In particular, we define the network, the devices, the channels, and the adversarial capabilities. In **Chapter 5** we analyze generic SAS-based message authentication protocols. We propose generic attacks against *any* SAS-based message authentication protocol and we deduce some bounds on their security according to the SAS length. As a result, we will define the notion of *optimality*. **Chapter 6** presents two security models: security in the stand-alone model and security in more complex settings. The stand-alone model only considers one protocol execution and one adversary. It allows us to prove the security of protocols in a formalized way. The latter is an important methodical advance, since one can pose many complex design requirements on message authentication protocols. As an important theoretical result, we prove that stand-alone security guarantees are preserved in complex settings as long as a simple set of usage restrictions are satisfied. The latter significantly simplifies the security analysis of SAS-based message authentication protocols. Indeed, since they do not rely on common secrets, they preserve stand-alone security guarantees even in the Bellare-Rogaway model. More precisely, we show that all SAS-based message authentication protocols are universally composable as soon as they are secure in the stand-alone model. **Chapters 7 and 8** focus on SAS-based message authentication protocols, unilateral and bilateral respectively. Both chapters start with a state-of-the-art, propose new protocol(s), study the works done between the publication of the protocols and the redaction of this thesis, and finally present some applications. In short, Non-Interactive Message Authentication Protocol (NIMAP) were not well studied. Indeed, there were a remaining gap between the used protocol and an optimal protocol. In this thesis, we propose an optimal NIMAP, called PV-NIMAP. Note that it allows to use 100 authenticated bits only while currently 160 are required. Regarding Interactive Message Authentication Protocols (IMAP), an optimal protocol was already been proposed by Vaudenay [Vau05b]. We concluded that there is no need for further work. However, as small additional contribution, Vaudenay [Vau05b] proposed a bilateral construction based on its unilateral one. Clearly, this construction is not optimal with respect to the move complexity and no formal security proof was given. In this thesis, we propose two optimal bilateral protocols, a message mutual-authentication protocol, called PV-SAS-MMA, and a message cross-authentication protocol, called PV-SAS-MCA. The next natural step is to extend the above two-party protocols to any number of participants. **Chapter 9** has the same structure as the two previous chapters and propose an optimal SAS-based group message authentication protocol, called LP-SAS-GMA. Finally, **Chapter 10** presents solutions to really build SAS-based Authenticated Key Agreement (AKA) by using the previous SAS-based message authentication protocols. In particular, this chapter gives two ready-to-use SAS-based key agreements: PV-SAS-AKA for two-party settings and LP-SAS-GKA for group settings.

1.2 Signatures Schemes (Part II)

In Part I, we have seen how to authenticate messages, e.g., public keys. In **Part II**, we assume that parties already have authenticated public keys and want to authenticate messages.

We focus on two major issues of digital signature schemes and all related material is presented in **Part II**. **Chapter 11** gives an overview of the goals of different signature schemes. Then, it defines the necessary background specific to signature schemes, such as a classical signature scheme and its adversarial model, proof of knowledge, the notion of zero-knowledge, etc.

The first sub-part is focused on privacy issues. Indeed, signed documents may lead to privacy issues since a signature is a proof of authenticity. For instance, consider the case of electronic passport (e-passport). A national authority creates an e-passport, stores all required information in the chip, computes a digital signature on these information, and stores it in the e-passport. Clearly, the signature is essential to prove that the stored information are genuine and produced by the national authority. When a border patrol checks the e-passport, he asks to obtain all information. The e-passport gives the stored information as well as the digital signature. As expected, the border patrol is able to check the genuineness of the data. However, as (un)expected, the border patrol may publish or sell this information. The main problem is not the publication of the information, as the person might still claim that information are false, but it is the publication of the signature. Indeed, this latter confirms that the data are correct and the person only has to abdicate. **Chapter 12** carefully analyzes the situation. In particular, a new authentication mechanism is proposed. The main idea is to prove the possession of a valid signature instead of giving it. For instance, in the e-passport case, the signature will be kept secret in the chip. This mechanism is called Offline Non-Transferable Authentication Protocol (ONTAP). Here, the concept of offline non-transferability is important and means that the border patrol is not able to prove the validity of the signature after the ONTAP is done.

The second sub-part is focused on weaknesses of signature schemes following the hash-and-sign paradigm. Due to algebraic group operations, textbook signatures are often defined for input messages of restricted length, typically hundreds of bits. Signing long messages should be possible, so, before signing a (long) message, it will be pre-processed. Usually, the pre-processing is a simple hash function. Hence, this way of signing is called *hash-and-sign*. However, digital signatures are often proven to be secure in the random oracle model while hash functions deviate more and more from this idealization. In **Chapter 13**, we start by analyzing the different variants of the hash-and-sign. Many solutions were proposed to avoid collision-resistance on hash functions. For instance, Bellare and Rogaway [BR97] proposed to use target collision resistant (TCR) randomized pre-hashing. Later, Halevi and Krawczyk [HK06a] suggested to use enhanced TCR (eTCR) hashing. Since the signatures are randomized, the signature length increases. To avoid the increase in signature length

in the TCR construction, Mironov [Mir06] suggested recycling some signing coins in the message pre-processing. Then, we will try to find a better model than the random oracle model to formalize weak hash functions. Based on the work of Liskov [Lis07], we develop the preimage-tractable random oracle model and use it. As a first result, we present a generic pre-processing allowing to build *strongly* secure signature schemes even when the hashing is weak and the internal (textbook) signature is weakly secure. As a second result, we present a method to reuse some random coins from the signing algorithm to the message pre-processing. Our method is applicable to any signature with randomized precomputation (SRP) scheme (defined in this thesis) and, thus, is more generic than the one from Mironov.

1.3 Keyboard Compromising Electromagnetic Emanations

This study is not a part of this thesis. However, the work is interesting and it is a contribution from the author of this thesis. The aim of this research is to give evidence that computer keyboards may radiate compromising electromagnetic emanations [VP09].

We started this work by analyzing the password transition route. When a user types a password, a huge amount of things occur: First, the user recalls it and types it on the keyboard. Second, the keyboard detects the keystrokes. Third, it transmits them to the computer. Fourth, the computer gets the keystrokes. Finally, the computer uses them, for instance, to log in to the computer. If it is a web site or a distant server login, the computer does some computations, for instance, encryption, and sends some data through the network. In order to recover the password, there are many possible attacks and they may target any step. One may use a video camera, a microphone, or an antenna to eavesdrop the typing procedure. One may use a key-logger to eavesdrop the data between the keyboard and the computer. One may use a Trojan horse or exploit a vulnerability in the operating system to recover the data from the computer. Another one may spy the network in order to recover the data passively, if they are encrypted, then he should try to decrypt them. Finally, one may simply attack the server.

Nowadays, most practical attacks against computers exploit vulnerabilities on operating systems or software applications. Plenty of new weaknesses are disclosed everyday. Fortunately, the corresponding patches are commonly delivered within a few days. When the vulnerability concerns hardware, generally no software update can avoid the exposure, the device must be changed.

We noticed that keyboards or keypads are the first sensitive element of the chain (assuming that we do not want to torture the password owner) while they are often used to transmit sensitive information such as passwords, e.g., to log in to computers, to do e-banking money transfer, to enter a credit card PIN number, etc. A weakness on these hardware devices will definitely jeopardize the security of any computer or cash dispenser (ATM).

Existing attacks on keyboards. There already exists some hardware-based attacks on keyboards. One of them consists in putting a small token, called key-logger, between the keyboard and the computer. This device eavesdrops all typed keystrokes and stores them into a memory.

A simple video camera may be used to capture pressed keys [BCV08]. If no direct vision is possible, one may use optical reflections [BDU08]. A blinking keyboard LED can also be used as a covert channel [LU02].

Several works focused on the use of microphones. Each key emits a unique acoustic pattern when it is pressed or released [AA04, ZZT05, BWY06].

Passive timing analysis may also be used to recover keystrokes. For instance, older SSH implementations may be used to recover encrypted passwords [SWT01].

Electromagnetic emanations. Compromising electromagnetic emanation problems appeared already at the end of the 19^{th} century. Wire networks became extremely dense due to the extensive use of telephone. As a consequence, people could sometimes hear other conversations on their phone line (crosstalk) due to undesired coupling between parallel wires. These crosstalks may be easily canceled by twisting the cables.

Academic research on compromising electromagnetic emanations started in the mid 1980's and a significant progress has been done recently [QS01, AARR03]. The threat related to compromising emanations has been constantly confirmed by practical attacks such as Cathode Ray Tubes (CRT) displays image recovery [EL85], RS-232 communications recovery [Smu90], Liquid Crystal Display (LCD) image recovery [KA98], secret key disclosure [GMO01], video displays risks [Kuh05, Tan07], and radiations from FPGAs [MÖPV07].

Our contribution. Since keyboards contain electronic components, they eventually emit electromagnetic waves. These electromagnetic radiation may reveal sensitive information such as keystrokes. Although Anderson and Kuhn [KA98, AK99, Kuh03] already tagged keyboards as risky. They also proposed countermeasures (see US patent [AK04]). However, we did not find any experiment or evidence proving or refuting the practical feasibility to remotely eavesdrop keystrokes, especially on modern keyboards.

To detect compromising emanations, we generally use a receiver tuned on a specific frequency. However, this method may not be optimal since a significant amount of information is lost during the acquisition of the signal. Our approach is to acquire raw signal directly from the antenna and to process the entire captured electromagnetic spectrum.

Thanks to our method, we uncovered four different ways to fully or partially recover keystrokes from wired and wireless keyboards. We implemented a *practical attack* based on these weaknesses. It recovers 95% of the keystrokes at a distance up to 20 meters, even through walls.

We tested 12 different keyboard models bought between 2001 and 2008 (PS/2, USB,

wireless and laptop). They are all vulnerable to at least one of our four attacks. We conclude that most modern computer keyboards generate compromising emanations (mainly because of the manufacturer cost pressures in the design). Hence, they are not safe to transmit highly confidential information.

Relation with this thesis. There is no direct link between this research and this thesis. However, one can note that the channel user-keyboard-computer is assumed to provide confidentiality, authenticity and integrity. The present study gives evidence that there exists many solutions to eavesdrop keystrokes at a distance. Hence, the keyboard does not seem to achieve confidentiality, but it provides authenticity. Application designers should keep this in mind. They should try to avoid the use of PIN codes (confidential) and instead prefer the use of SAS (authenticated).

Academic Contributions from the Author

[**PV06a**] An Optimal Non-interactive Message Authentication Protocol.
Sylvain Pasini and Serge Vaudenay.
In the proceedings of the *Cryptographers' Track at the RSA Conference – CT-RSA '06*.
Contribution of this paper can be found on Chapters 4, 5, and 7.

[**PV06b**] SAS-based Authenticated Key Agreement.
Sylvain Pasini and Serge Vaudenay.
In the proceedings of *Public Key Cryptography – PKC '06*.
Contribution of this paper can be found on Chapters 4, 8 and 10.

[**PV07**] Hash-and-sign with Weak Hashing Made Secure.
Sylvain Pasini and Serge Vaudenay.
In the the proceedings of the *Australasian Conference on Information Security and Privacy – ACISP '07*.
Contribution of this paper can be found on Chapter 13.

[**LP08**] SAS-Based Group Authentication and Key Agreement Protocols.
Sven Laur and Sylvain Pasini.
In the proceedings of *Public Key Cryptography – PKC '08*.
Contribution of this paper can be found in Chapters 4, 6, 9, and 10.

[**LP09**] User-Aided Data Authentication.
Sven Laur and Sylvain Pasini.
In the *International Journal of Security and Networks*, 2009.
Contribution of this paper can be mainly found on Chapters 4 and 6.

[**MPV09**] Efficient Deniable Authentication for Standard Signatures.
Jean Monnerat, Sylvain Pasini, and Serge Vaudenay.
In the the proceedings of the *International Conference on Applied Cryptography and Network Security* – ACNS '09.
Contribution of this paper can be found on Chapters 11 and 12.

[**VP09**] Compromising Electromagnetic Emanations of Wired and Wireless Keyboards.
Martin Vuagnoux and Sylvain Pasini.
In the proceedings of *USENIX Security '09*.
Not a part of this thesis.

Chapter

TWO

The Authentication Problem

One of the main issues in cryptography is the establishment of a secure peer-to-peer (or group) communication over an insecure channel. With no assumption, such as availability of an extra secure channel, this task is impossible. However, given some assumption(s), there exists many ways to setup a secure communication. The application designer chooses the most suitable solution depending mainly on the assumptions, the requirements, the efficiency, and of course the required security.

Section 2.1 surveys some cryptographic primitives. Of course, most of the readers can skip this folklore section which is here for completeness and to make the reader familiar with the terminology. Section 2.2 surveys some different communication ways that human beings are able to use to communicate. Section 2.3 points out that setting up a secure communication can be practically done simply by authenticating some data. Section 2.4 recalls that authenticated messages are in general long and thus their authentication may be tedious, e.g., by phone. This section shows how protocols are able to reduce the amount of authenticated data. Section 2.5 presents techniques for message authentication that are different of usual ones. Finally, Section 2.6 summarizes the chapter and motivates the separation of this thesis in two parts.

2.1 Basics in Cryptography

We consider a message source and a message destination. The source wants to send a message m to the destination with some security properties. We denote by $\widehat{m}$ the received message since it may be different from m. The main cryptographic properties are the following:

- *Confidentiality* aims at preserving the secrecy of the message. Nobody except the destination can deduce information on the transmitted message.
- *Authenticity* aims at guaranteeing that the message was sent by the source.
- *Integrity* aims at guaranteeing that the message $\widehat{m}$ received by the destination is the same as the message m sent by the source.

In the following, the different solutions are classified depending on the assumptions done on the extra channel. In the figures, we use the notation C, A, and/or I for channels achieving respectively confidentiality, authenticity, and/or integrity. If nothing is specified, the channel is insecure.

2.1.1 Symmetric Cryptography

Using *symmetric cryptography*, it is possible to establish a secure peer-to-peer channel only assuming that we can agree on a private and authenticated key. This model was presented by Shannon [Sha49]. Often we refer to this model by *symmetric* cryptography because the source and the destination use the *same* key.

Confidentiality. Confidentiality over an insecure channel can be achieved using *symmetric encryption*. As depicted in Figure 2.1, it is possible to send confidential messages over an insecure channel from a source to a destination. This model [Sha49] is only applicable in the situations where parties are able to use an extra channel achieving confidentiality, authenticity, and integrity.

Symmetric encryption can be done using either stream ciphers, like E0 [Blu03], or block ciphers, like DES [DES77, DES99], AES [AES01], or FOX [JV03].

Authenticity and integrity. Authenticity and integrity on the insecure channel can also be achieved with symmetric cryptography by using *message authentication codes* (MAC). As depicted in Figure 2.2, it is possible to achieve authenticity and integrity on the messages transmitted over an insecure channel from a source to a destination. As before, this is only possible if parties are able to use an extra channel achieving confidentiality, authenticity, and integrity.

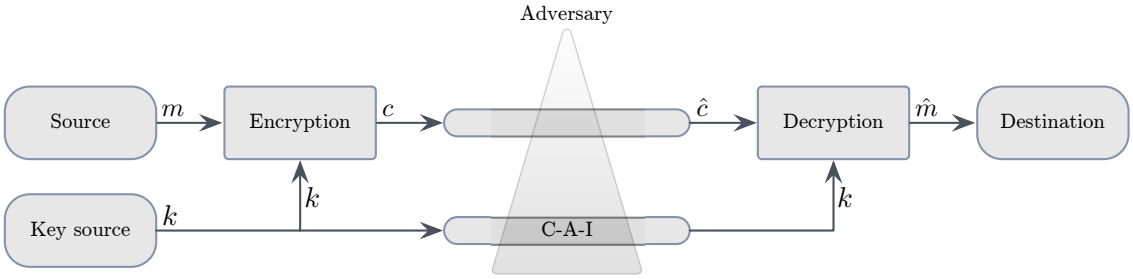


Figure 2.1. *The Shannon Model.*

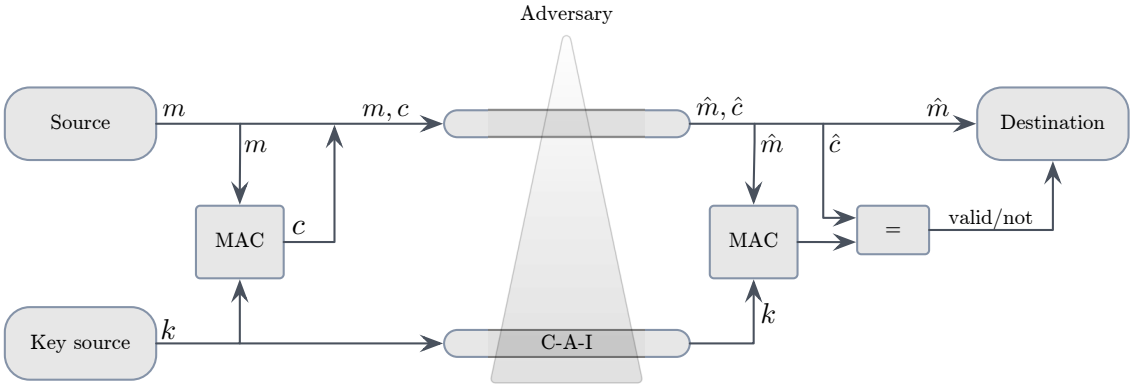


Figure 2.2. *Authentication with Symmetric Cryptography.*

Note that MACs can be built using hash functions or ciphers. HMAC by Bellare-Canetti-Krawczyk [BCK96] is an example of a construction based on hash functions while the One-Key CBC MAC (OMAC) by Iwata and Kurosawa [IK03] is an example based on block ciphers.

2.1.2 Agreeing on a Secret Key without Confidential Channel

In the previous model, confidentiality on the extra channel is mandatory to achieve confidentiality on the insecure channel. There is no gain except the extra channel bandwidth. Merkle [Mer78] and Diffie-Hellman [DH76] discovered at the same time that the confidentiality on the extra channel can be relaxed. Indeed, their model reduce the confidential extra channel to an authenticated extra channel. As depicted in Figure 2.3, the extra channel is still used to agree on a private key but the difference is that the key itself is not sent directly. Indeed, the secret key is the result of a protocol execution, also called a *key agreement protocol*.

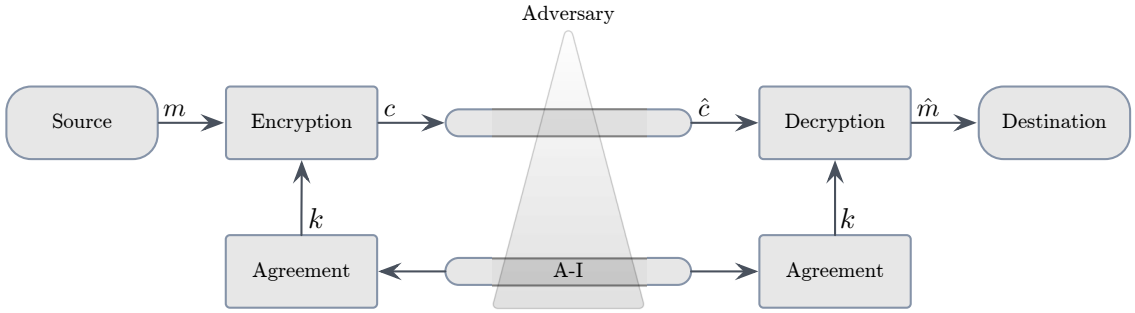


Figure 2.3. *The Merkle-Diffie-Hellman (MDH) Model.*

One of the most well-known key agreement protocol is due to Diffie and Hellman [DH76] and is usually called *the DH protocol*. As depicted in Figure 2.4, it requires only two moves. In order to avoid man-in-the-middle attacks, these two moves should be authenticated.

Alice		Bob
pick $x \in_{\mathsf{u}} \lvert\mathcal{G}\rvert$		pick $y \in_{\mathsf{u}} \lvert\mathcal{G}\rvert$
$X \leftarrow g^x$	$\xrightarrow{X}$	
	$\xleftarrow{Y}$	$Y \leftarrow g^y$
$K \leftarrow Y^x = g^{xy}$		$K \leftarrow X^y = g^{xy}$

Figure 2.4. *The Diffie-Hellman (DH) Key Agreement Protocol.*

Both parties know the public parameter g which spans a group $\mathcal{G}$. Alice picks a random

number x and computes $X \leftarrow g^x$ while Bob picks a random y and computes $Y \leftarrow g^y$. Then, Alice sends X to Bob and Bob sends Y to Alice. Alice, resp. Bob, computes Y^x, resp. X^y, and both result in g^{xy}. Now, both of them share a secret key $K = g^{xy}$. Note that g is chosen such that for any adversary who knows g, X, and Y it is hard to retrieve x and y (Discrete Logarithm Problem). So, anyone seeing (or eavesdropping) X and/or Y is not able to deduce x or y since the discrete logarithm is hard. Consequently, it is hard to find the key K. On the other hand, with no authentication an adversary can run a man-in-the-middle attack between the two participants, so the authentication of the two messages is mandatory for this protocol.

2.1.3 Public-Key Cryptography

Public-key cryptography stands for any scheme for which the knowledge of a public-key does not compromise security. The problem of the MDH model is that it is only adapted to key agreement protocols and cannot achieve confidentiality directly. This becomes possible if we combine the key agreement protocol with some symmetric cryptographic primitives.

In this section, we concentrate on public-key primitives. Indeed, by using public-key cryptography, it is also possible to achieve confidentiality and authenticity on the insecure channel. Of course, with public-key cryptography, we still relax the confidentiality hypothesis on the extra channel.

Confidentiality. Confidentiality over the insecure channel can be achieved using *public-key encryption*. As depicted in Figure 2.5, the destination should first execute a key generation algorithm which creates a key pair, e.g., a private key and a public key. Then, anyone knowing the public key of the destination is able to send confidential messages to him. Only the owner of the corresponding private key, in this case the destination, is able to decrypt the received messages. The use of this model is only possible if the source is ensured to use the correct public key, otherwise this model is subject to man-in-the-middle attacks.

Well-known examples of public-key cryptosystems are RSA [RSA78] and ElGamal [ElG85].

Authenticity and integrity. Authenticity and integrity on the insecure channel can be achieved with public-key cryptography, too. It is done by using *digital signature schemes*. As depicted in Figure 2.6, unlike the encryption, the key generation is done by the source. Then, the source signs the message and sends the message-signature pair to the destination. At the destination, the received message-signature pair is verified by using the public key. As for the encryption, the use of this model is only possible if the destination is ensured to use the correct public key, otherwise this model is subject to man-in-the-middle attacks.

Typical examples of textbook digital signature schemes are RSA [RSA78] and ElGa-

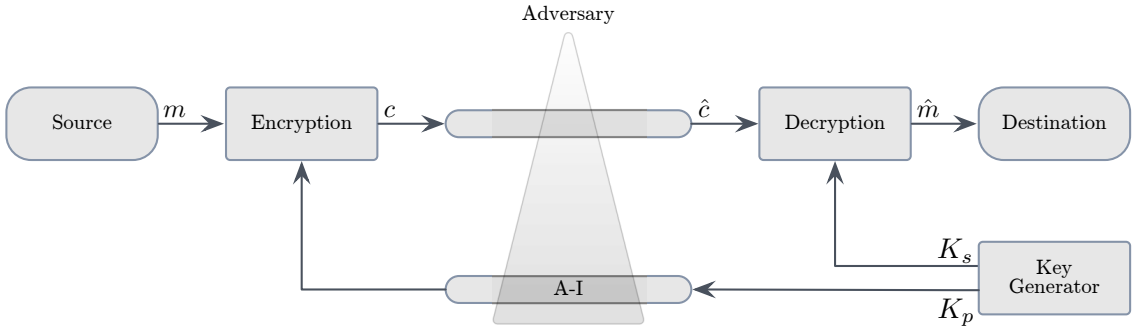


Figure 2.5. *The Public-Key Encryption Model.*

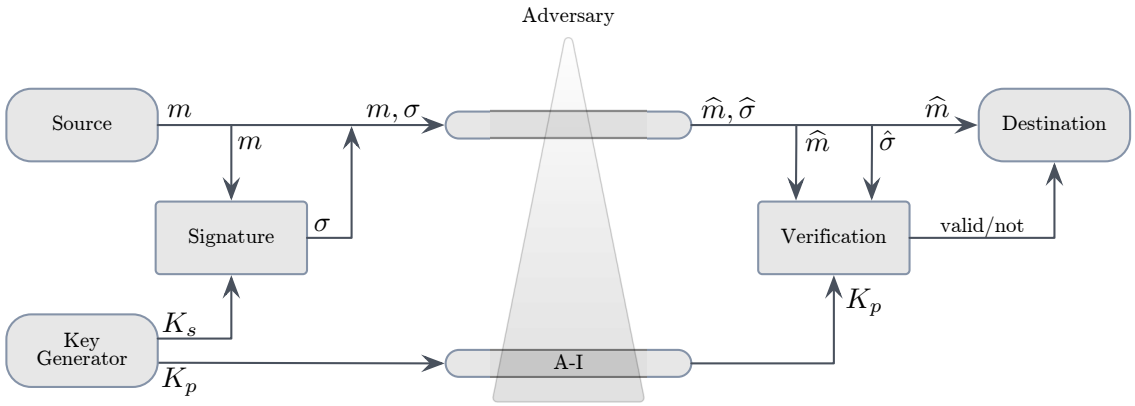


Figure 2.6. *The Public-Key Authentication Model.*

mal [ElG85]. In practice one may for instance use the Digital Signature Standard (DSS) [DSS94, DSS00] (based on ElGamal).

Public Key Infrastructure. One major problem with public-key cryptography is the transmission of the public key. Indeed, any public key can only be used if we are sure that the owner of the corresponding private key is really the correct person. For instance, if Alice wants to send an encrypted message m to Bob, she will encrypt it with the public key of Bob K_p^{Bob}. So, before that, Bob should send the public key to Alice in some way. If he just send it with no precaution, an adversary may replace this key by its own one and then be able to decrypt all messages as depicted in Figure 2.7.

Alice		$\mathcal{A}$		Bob
input: m				**input:** $K_p^{\mathsf{Bob}}, K_s^{\mathsf{Bob}}$
	$\xleftarrow{\widehat{K}_p}$	$(\widehat{K}_p, \widehat{K}_s) \leftarrow \mathsf{keygen}()$	$\xleftarrow{K_p^{\mathsf{Bob}}}$	
$\widehat{c} \leftarrow \mathsf{enc}(\widehat{K}_p, m)$	$\xrightarrow{\widehat{c}}$	$m \leftarrow \mathsf{dec}(\widehat{K}_s, \widehat{c})$		
		$c \leftarrow \mathsf{enc}(K_p^{\mathsf{Bob}}, m)$	$\xrightarrow{c}$	$m \leftarrow \mathsf{dec}(K_s^{\mathsf{Bob}}, c)$

Figure 2.7. *The Folklore Man-in-the-Middle Attack During a Public-Key Transfer.*

Here, the adversary $\mathcal{A}$ relays the message m from Alice to Bob, so the only benefit for $\mathcal{A}$ is the eavesdropping of the communication. However, we can imagine scenarios where $\mathcal{A}$ modifies the messages from Alice before sending it to Bob.

To protect against such an attack, we need to *authenticate* the public key. In the following, we describe the functioning of a *Public Key Infrastructure* (PKI). We will see later other methods allowing to authenticate public keys with no infrastructure and no trusted parties.

We first need to introduce the notion of a *certificate*. Basically, a certificate contains an identity, a public key, an expiration date, and a certificate signature. It is used to prove that the public key is binded to the identity (until the expiration date). For that, a third party signed the data in order to produce the certificate signature. So, anyone knowing the public key of the third party is able to verify the validity of the whole certificate. In Figure 2.8, we use the same example as before, but with certificates. Alice wants to send an encrypted message m to Bob. So, Bob should give his public key K_p^{Bob} to Alice. For that, he will ask Trent to obtain a certificate, i.e., $\mathsf{cert}^{\mathsf{Bob}}$. The public key of Trent, i.e., K_p^{Trent}, is known (and authenticated) by everyone. In short, Alice is able to authenticate the public key of Bob given the public key of Trent. There is some kind of recursion. From the point of view of Alice, she should obtain the public key of Trent (in any way, e.g., by using a certificate) and in addition she should *trust* Trent.

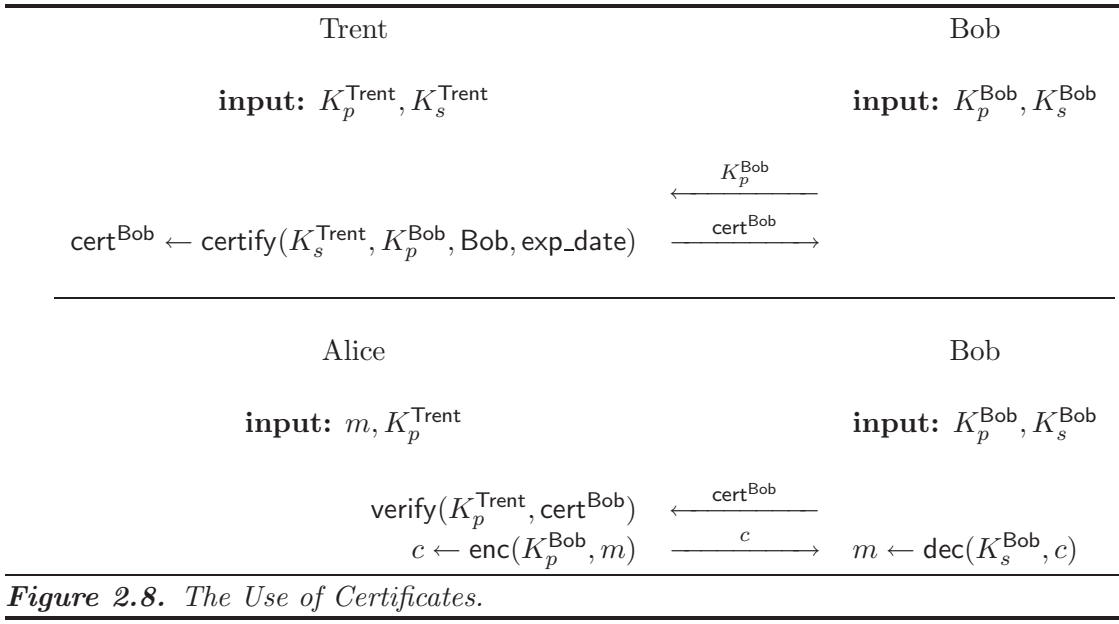


Figure 2.8. *The Use of Certificates.*

A *Public Key Infrastructure* (PKI) is used to handle the creation, the management, and the distribution of certificates. A PKI consists of Certificate Authorities (CA), Registration Authorities (RA), and certificate directories. In short, a CA creates and signs certificates for entities. For that a CA possesses a key pair and a *root certificate* for its public key. The root certificate may be certified by another CA. This leads to a CA hierarchy. One CA receiving a certificate should verify it by checking each certificate from the trusted CA (the one he knows the public key from) to the final certificate. This is also known as the chain of trust. A CA may delegate the registration of entities to a RA. The RA is only responsible to identify the entity, but cannot output certificates without the CA. In fact, only the CA knows the private signing key for yielding certificates to entities.

One of the main PKIs in use today is the one implemented for the World Wide Web. When a user opens a secure Web page (HTTPS, or HTTP with SSL), the server sends a certificate to the browser. The browser should then verify the validity of the certificate. Clearly, a browser does not know the certificate of all web sites in the world. In general, Web browsers only integrate certificates of popular CAs and trust them (by default). In order to be verifiable, the Web server should obtained a certificate for its public key from a popular CA. So, the Web browser is able to check the validity of the received certificate (through the CA) and then to start a secure session with the Web server.

2.2 Communication Channels

In this section, we analyze the different channels at disposal for any human being. Human beings can communicate with different communication channels and they choose one of them depending on some requirements. For example, if a human being needs to reach another human being for urgent matters, he must choose a channel with high availability and low latency such as a telephone link. But, if he wants to transfer money, he must establish a more reliable link for instance by going to the desk to encounter the banker. (Nowadays, he can use the Internet with prior established security association.)

The security of communication channels can be characterized by some *security attributes* which are defined below.

Definition 2.1 (Security Attributes).
Suppose a communication channel between a sender and a receiver. A message m is sent and a message $\widehat{m}$ is received. We define the following security properties*:*

Confidentiality *assumes that only the legitimate receiver can read the message $\widehat{m}$.*

Integrity *assumes that the received message $\widehat{m}$ is the same as the sent message m, i.e., $\widehat{m} = m$.*

Authenticity *assumes that only the legitimate sender can input a message m into the channel. This is often combined with integrity, i.e., $m = \widehat{m}$.*

Freshness *assumes that the received message $\widehat{m}$ was not received before.*

Liveliness *assumes that a message m which has been sent by the sender will eventually be delivered to the receiver.*

Timeliness *assumes that a message m which has been sent by the sender will be delivered to the receiver in real time (transmission time in negligible).*

In addition, to compare the different human communication channels, it is necessary to define other properties which characterize the *usability* of these channels. These *communication properties* are defined below.

Definition 2.2 (Communication Properties).
Suppose a communication channel between a sender and a receiver. We define the following communication properties*:*

The cost *represents the required amount of money spent to establish the communication channel and to transmit a message.*

The availability *expresses the fact that the channel can easily be established at any time.*

The speed rate *represents the amount of data that can be transferred through the channel for a fixed time duration.*

The latency *represents the amount of time between the moment when the message is sent and the moment when it is received.*

Using Definition 2.1 and Definition 2.2, it is possible to compare the usual human communication channels in a cryptographic sense.

Face to face (voice) conversation allows perfect authentication, perfect integrity and in certain cases, confidentiality. In addition, freshness, liveliness, and timeliness are trivially ensured. However, this channel can have a very high cost if, for example, the two persons are far from each other. For the same reasons, the availability is also bad. Note that the communication has (almost) no latency but a low speed rate. In conclusion, this human channel achieves high security but low throughput.

Telephone is less secure than a face to face conversation. It allows a third party to spy the communication and thus does not guarantee confidentiality. On the other hand, it has a much lower cost and a higher availability. In short, it still preserves authentication assuming that both users can recognize each others' voice. Integrity and freshness is also guaranteed, indeed it is pretty hard to modify a message in real time as it is integrated in an interactive conversation.

Mail, like a postcard or a parcel, is not confidential either. It can be easily lost and thus this channel does not guarantee liveliness. We can consider that a handwritten mail achieves authentication by assuming that the recipient can identify the writing. As for telephone, this channel guarantees availability but has a long latency.

Voice mail is in a security sense close to mail except that it can be easily replayed and so messages may not be fresh. Note that voice mail (as well as fax) may have a certain amount of latency since we do not know when the recipient will read the message.

Electronic mail is the worst communication channel in terms of security, it protects nothing by itself. However, it is the most usable communication channel and its costs is very small (too small if we consider the spam phenomenon), the availability and the speed rate are very high.

A short overview of the security and communication properties for each human communication channel is described in Figure 2.9. The pictogram ☺ indicates that the property on the channel is a good feature. Note that the choice is in fact a trade-off between the security and the human usability. In other words, the most secure channel is a face to face conversation but it is the least usable. In the other hand, the most usable is the email but it is the least secure. For a specific use, the choice of the human communication channel is a trade-off between the required security and the available cost.

	Encounter	Telephone	Fax	Mail	Voice mail	E-mail
Authenticity	☺	☺		☺	☺	
Integrity	☺	☺		☺	☺	
Confidentiality	☺					
Freshness	☺	☺		☺		
Liveliness	☺	☺				
Timeliness	☺	☺				
Interactivity	☺	☺				
Low cost		☺	☺	☺	☺	☺
Availability			☺	☺	☺	☺
Speed rate			☺			☺
Low latency	☺	☺				

Figure 2.9. *The Common Human Communications Channels*

Remark 2.3.

With no prior security association, confidentiality is achieved only using a face to face conversation which can be very expensive in certain cases. However, a phone call, which is very easy to setup worldwide, achieves authentication assuming that the two speakers can identify each other by recognizing his voice/behavior.

In the next section we study the problem of the encapsulation of a secure communication channel on insecure (but cheap) channels.

2.3 Towards Usable Solutions to Setup Secure Communications

As described in Section 2.1, assuming a shared secret key it becomes trivial to setup a secure communication by using symmetric cryptography.

Remark 2.4.

Setting up a secure communication can be reduced to the problem of agreeing on a shared secret key.

The problem is now the exchange (or the agreement) of the shared secret key. To relax the confidentiality assumption on the extra channel, one can propose to protect the communication by using public-key cryptography. In this case, each participant only need to authenticate its public key to the other ones. However, public-key encryption is slow and costly comparatively to symmetric encryption. We want to limit its use as much as possible in order to build efficient systems (for lightweight devices). So, in order to obtain efficient systems, data communication should be protected by using symmetric cryptography.

Remark 2.5.
We will try to restrict the use of public-key cryptography, i.e., use it only to exchange the shared secret key.

Key Agreement Protocols. As depicted in Figure 2.3, following the MDH model, it is possible to relax the confidential assumption on the extra channel by running key agreement protocols. For instance, running a DH protocol allows the two participants to agree on a shared secret key while any adversary spying the protocol is not able to deduce the secret key. However, such protocols are subject to man-in-the-middle attacks and, as a consequence, such communications require authentication (and integrity).

Semi-authenticated Key Transfer. Thanks to public-key cryptography, it is possible to send confidential messages to a recipient by encrypting them using the recipient's public key. Then, only the recipient (the owner of the corresponding private key) is able to decrypt the messages. In the case where the message is a symmetric key, we obtain a semi-authenticated key transfer, for example using RSA [RSA78], as depicted in Figure 2.10.

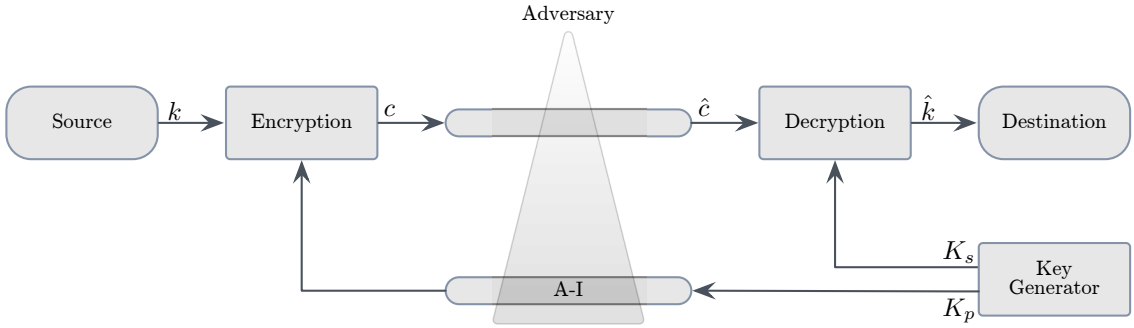


Figure 2.10. *The Semi-Authenticated Key Transfer.*

At the beginning, the extra channel provided confidentiality. Thanks to key agreement protocols or to semi-authenticated key transfer, we only need an extra authenticated channel.

Remark 2.6.
A shared secret key can be established by exchanging messages and only ensuring their authenticity and integrity (no need of confidentiality at all).

Remark 2.3 says that an extra channel which achieves only authenticity can be established simply by telephone. Thus, with Remark 2.6, we can setup a secure communication over an insecure channel without requiring to encounter, but simply using telephone to exchange, in an authenticated way, the DH messages or a public key.

2.4 Message Authentication

Thanks to the previous section, we are now able to setup a secure communication only by exchanging some authenticated messages, e.g., by telephone. So, we no longer need any confidential assumption on the extra channel at all. However, there still remains a problem: exchanging DH messages or a public key by telephone is tedious since they are long. Indeed, an authenticated channel could be seen as an expensive channel. So, we should try to optimize the amount of communication over this channel. Message authentication can be done in different ways:

First, there is *user-aided message authentication protocols*. In practice, the (long) messages that should be authenticated will be sent over the insecure channel and some additional (short) messages will be sent over the extra channel. An example is the folklore protocol used in SSH and PGP as depicted in Figure 2.11. The protocol sends the message to be authenticated, e.g., the public key K_p, over the insecure channel, then authenticates by a human channel its "fingerprint", denoted by $f(K_p)$ in Figure 2.11, and compares the locally computed fingerprint with the one authenticated.

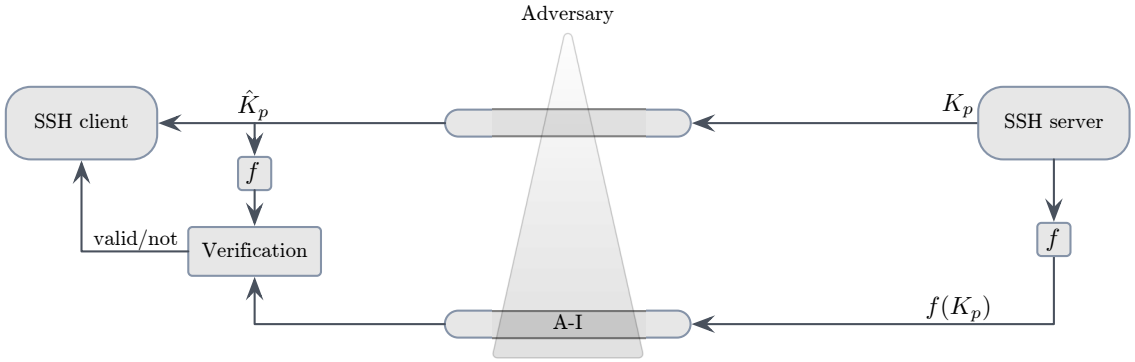


Figure 2.11. *The SSH Public Key Authentication Model.*

Another way to authenticate data is to use *message authentication codes* (MACs) as seen in Figure 2.2. This solution seems an overkill. Indeed, we are trying to agree on a shared secret key to setup a secure communication while we also need one to authenticate messages.

Finally, we can use the MACs equivalent in public-key cryptography: *signature schemes*. This latter solution requires the authentication of the public key of the signer. Note that the authentication of this public key can be done by using a message authentication protocol (as explained just above), or with the help of a trusted third party, e.g., a public-key infrastructure. In both cases, we relax the confidentiality assumption on the extra channel.

Remark 2.7.

In conclusion, to setup a secure communication, we either need a user-aided message

authentication protocol or a signature scheme. In the first case, we need to rely on an extra authenticated channel (telephone, string copy between devices, etc.) while in the second case, we need to rely on the authentication of a public key. This latter authentication can be also done with a user-aided message authentication protocol or with the help of a trusted third party (like a PKI).

2.5 Other Ways for Message Authentication

Here, we briefly present some protocols ensuring message authentication and requiring no pre-shared key and no trusted third party.

2.5.1 Protocols Using Time Bounding

A method to setup secure communications based on vocal biometric signals was proposed by Wu, Bao, and Deng [WBD05]. The protocol is depicted in Figure 2.12. It allows Alice to authenticate Bob and to setup a secret session key. Note that it uses DH values to establish the secret session key. These values are authenticated using voice signals and agreed upon with the help of a timer which allows to detect man-in-the-middle attacks.

In the proposed protocol, Alice, resp. Bob, chooses a value x, resp. y, and computes the DH value $X \leftarrow g^x$, resp. $Y \leftarrow g^y$. Alice, resp. Bob, computes their local key by hashing their DH value, i.e., $K_A \leftarrow \mathsf{H}(X)$, resp. $K_B \leftarrow \mathsf{H}(Y)$. Then, Alice records a "voice challenge" C_A which is authenticated by her voice. Then, she sends an encrypted value E_{C_A} to Bob. The encryption is done using her local key K_A. The "voice challenge" can be for instance the sentence "what did you wrote in your last mail?" or anything which leads to a fresh answer and which can easily be answered by Bob but not so easily by others. Bob makes the same operations and sends E_{C_B} to Alice. Each recorded "voice challenge" must be at least of duration T. We stress that at this time nobody can listen to the challenges since they are encrypted and the keys are kept secret. Alice starts a timer and reveals her DH value X. Then, Bob computes $\widehat{K}_A$, i.e., $\widehat{K}_A \leftarrow \mathsf{H}(\widehat{X})$, decrypts the challenge of Alice, listens to it, and tries to recognize her voice. If the voice is not the one of Alice, Bob stops the protocol, otherwise he records a response R_B to the challenge C_A. Bob can now compute the common key K_{BA} using the DH protocol, i.e., $K_{BA} \leftarrow \widehat{X}^y$. He encrypts the response using the common key K_{BA}, sends the encrypted response and reveals his DH value Y. When Alice receives the values, she stops the timer. She can now deduce the common key K_{AB}, i.e., $K_{AB} \leftarrow \widehat{Y}^x$, and the key of Bob, i.e., $K_B \leftarrow \mathsf{H}(\widehat{Y})$. Now, she can listen to the challenge of Bob $\widehat{C}_B$ and to the response of Bob $\widehat{R}_B$ to her challenge C_A. Finally, she checks the time elapsed, the identity of the two voice messages from Bob, and whether the response is consistent with the challenge or not.

To understand the role of the timer, a man-in-the-middle attack is presented in Figure 2.13.

Alice		Bob
pick $x \in_{\mathsf{u}} \{0,1\}^k$		pick $y \in_{\mathsf{u}} \{0,1\}^k$
$X \leftarrow g^x$		$Y \leftarrow g^y$
$K_A \leftarrow \mathsf{H}(X)$		$K_B \leftarrow \mathsf{H}(Y)$
$C_A \leftarrow \mathsf{record}()$		
$E_{C_A} \leftarrow \mathsf{enc}_{K_A}(C_A)$	$\xrightarrow{E_{C_A}}$	
		$C_B \leftarrow \mathsf{record}()$
	$\xleftarrow{E_{C_B}}$	$E_{C_B} \leftarrow \mathsf{enc}_{K_B}(C_B)$
$\mathsf{start}(timer)$	$\xrightarrow{X}$	$\widehat{K}_A \leftarrow \mathsf{H}(\widehat{X})$
		$K_{BA} \leftarrow \mathsf{H}(\widehat{X}^y)$
		$\widehat{C}_A \leftarrow \mathsf{dec}_{\widehat{K}_A}(\widehat{E}_{C_A})$
		check $\mathsf{identity}(\widehat{C}_A)$
		$R_B \leftarrow \mathsf{record}()$
$t_a \leftarrow \mathsf{stop}(timer)$	$\xleftarrow{E_{R_B} \| Y}$	$E_{R_B} \leftarrow \mathsf{enc}_{K_{BA}}(R_B)$
$\widehat{K}_B \leftarrow \mathsf{H}(\widehat{Y})$		
$K_{AB} \leftarrow \mathsf{H}(\widehat{Y}^x)$		
$\widehat{C}_B \leftarrow \mathsf{dec}_{\widehat{K}_B}(\widehat{E}_{C_B})$		
$\widehat{R}_B \leftarrow \mathsf{dec}_{K_{AB}}(\widehat{E}_{R_B})$		
check $t_a, \mathsf{identity}(\widehat{C}_B), \mathsf{identity}(\widehat{R}_B)$		
check $\mathsf{consistence}(C_A, \widehat{R}_B)$		
output: Bob, K_{AB}		**output**: K_{BA}

Figure 2.12. *Semi-Authenticated Key Agreement Using Voice Records.*

An adversary $\mathcal{A}$ can obtain a challenge from Alice C_A simply by answering a challenge C'_B since it is encrypted and thus cannot be verified by Alice at this moment. Alice responds with her DH value X and gives C_A to any adversary. Note that the adversary $\mathcal{A}$ cannot mimic Bob, but he can encrypt an old challenge of Bob (replay), i.e., C_B^{old}. The adversary $\mathcal{A}$ show now respond to the challenge of Alice. It cannot mimic Bob and consequently he has to request Bob to respond to the challenge C_A from Alice. He proceeds simply by impersonating Alice to Bob, submitting the challenge C_A and obtaining the response R_B. Finally, the adversary answers to Alice with the response R_B (encrypted) from Bob.

Alice		$\mathcal{A}$		Bob
K_A		K_Z		K_B
	$\xrightarrow{E_{C_A}}$			
	$\xleftarrow{\widehat{E}_{C_B}}$	$\widehat{E}_{C_B} \leftarrow \mathsf{enc}_{K_Z}(C_B^{old})$		
$\mathsf{start}(timer)$	$\xrightarrow{X}$	$C_A \leftarrow \mathsf{dec}_{K_A}(E_{C_A})$		
		$\widehat{E}_{C_A} \leftarrow \mathsf{enc}_{K_Z}(C_A)$	$\xrightarrow{\widehat{E}_{C_A}}$	
			$\xleftarrow{E_{C_B}}$	$C_B \leftarrow \mathsf{record}()$
			$\xrightarrow{Z}$	$\mathsf{listen}(C_A)$
		$R_B \leftarrow \mathsf{dec}_{K_B}(E_{R_B})$	$\xleftarrow{E_{R_B}\|Y}$	$R_B \leftarrow \mathsf{record}()$
$t_a \leftarrow \mathsf{stop}(timer)$	$\xleftarrow{\widehat{E}_{R_B}\|Z}$	$\widehat{E}_{R_B} \leftarrow \mathsf{enc}_{K_Z}(R_B)$		
output: Bob, K_{AZ}		**output**: K_{ZA}		**output**: K_{BZ}

Figure 2.13. *Man-In-The-Middle Attack (Simplified Version).*

Note that the time elapsed t_a corresponds to the time for Bob to record the challenge C_B, to listen to the challenge C_A from Alice, to record the response R_B, and some duration δ for computations and transmissions, i.e., $t_a = |C_B| + |C_A| + |R_B| + \delta \geq 3T + \delta$. With no attack, Bob does not have to record his challenge during this time interval since it has been recorded before, i.e., $t_a = |C_A| + |R_B| + \delta \geq 2T + \delta$. Using the timer, Alice can detect man-in-the-middle attacks. This attack shows the important role of the timer in this protocol.

We note that this protocol is very sensitive to the time in which Bob responds. Suppose Bob has made a mistake in his record sample and records another one. He has listened to the challenge of Alice and recorded two samples. Thus, the time elapsed seems to be a man-in-the-middle attack. In conclusion, this protocol is quite hard to implement in a *user-friendly* way since the timing is constrained for Bob.

2.5.2 Protocols Using Distance Bounding

Brands and Chaum [BC93] proposed a practical method to upper bound the physical distance between two devices. For instance, during an authentication phase between an employee and an access control, the system would like to be ensured that the employee is nearby, i.e., within a few meters. The proposed principle is quite simple. It consists of a challenge-response using only one-bit messages. The verifier **V** sends a bit (challenge) and the prover **P** replies immediately with a bit (response). They assume that electronic devices which play the role of the prover can have very short timings between the reception of a challenge and the sending of the corresponding response. The verifier **V** has simply to measure the time elapsed between the sending of the challenge and the reception of the response. Knowing the time elapsed, it can easily deduce the maximal distance between them. Two attacks are described in [BC93]: the mafia fraud which is a man-in-the-middle attack where the adversary is a fraudulent verifier at the same time as a fraudulent prover and an attack in which the prover **P** sends bits out too soon. Solutions for preventing both attacks are proposed in [BC93] and the final protocol proposed is depicted in Figure 2.14.

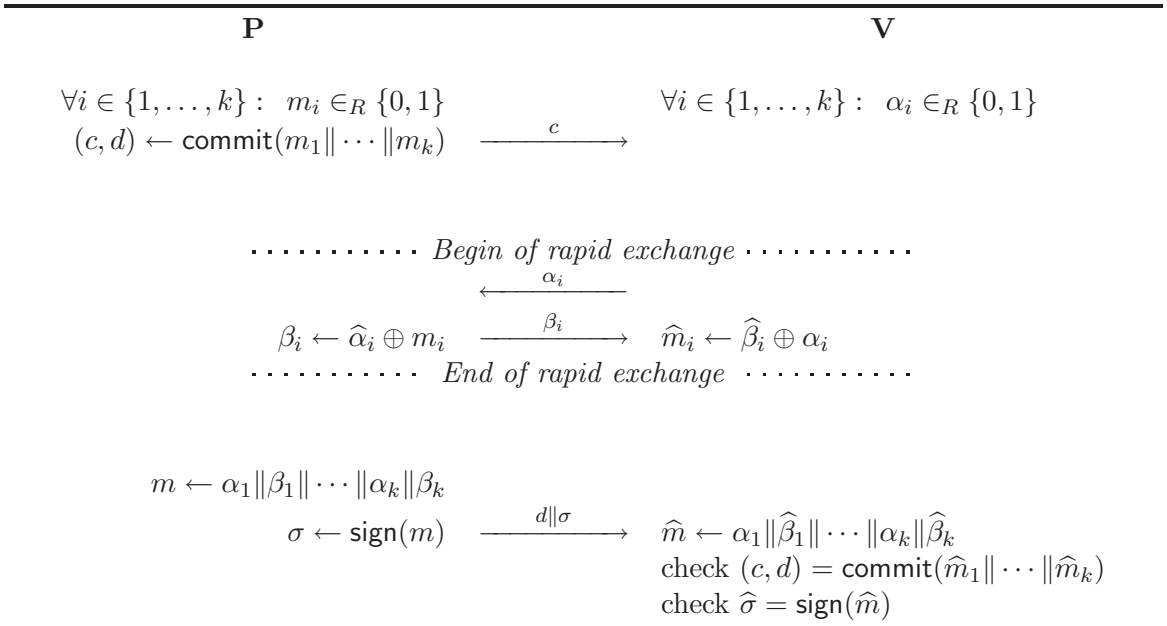


Figure 2.14. *Distance Bounding Protocol.*

Based on this distance upper bounding, Cagalj, Capkun, and Hubaux [CCH05] proposed a key agreement protocol for wireless networks, in particular for peer-to-peer communication. It allows the parties to authenticate DH values given in input. In the previous protocol, it is not clear how the prover should sign the message. Cagalj, Capkun, and Hubaux [CCH05] propose a method which only needs authentication. The protocol is depicted in Figure 2.15.

Alice		Bob
input: id_A, g^{X_A}		**input:** id_B, g^{X_B}
Pick $N_A, R_A \in_{\mathsf{u}} \{0,1\}^k$		Pick $N_B, R_B \in_{\mathsf{u}} \{0,1\}^k$
$m_A \leftarrow 0 \Vert \mathsf{id}_A \Vert g^{X_A} \Vert N_A$		$m_B \leftarrow 1 \Vert \mathsf{id}_B \Vert g^{X_B} \Vert N_B$
$(c_A, d_A) \leftarrow \mathsf{commit}(m_A)$		$(c_B, d_B) \leftarrow \mathsf{commit}(m_B)$
$(c'_A, d'_A) \leftarrow \mathsf{commit}(0 \Vert R_A)$	$\xrightarrow{c_A \Vert c'_A}$	$(c'_B, d'_B) \leftarrow \mathsf{commit}(1 \Vert R_B)$
	$\xleftarrow{c_B \Vert c'_B}$	
	$\xrightarrow{d_A}$	$\widehat{m}_A \leftarrow \mathsf{open}(\widehat{c}_A, \widehat{d}_A)$
		Verify 0 in $\widehat{m}_A$
$\widehat{m}_B \leftarrow \mathsf{open}(\widehat{c}_B, \widehat{d}_B)$	$\xleftarrow{d_B}$	$i_B \leftarrow N_B \oplus \widehat{N}_A$
Verify 1 in $\widehat{m}_B$		
$i_A \leftarrow N_A \oplus \widehat{N}_B$		
Begin of Distance Bounding Phase		
$\beta_0 = 0$		
$\alpha_i \leftarrow R_{A,\text{bit } i} \oplus i_{A,\text{bit } i} \oplus \widehat{\beta}_{i-1}$	$\xrightarrow{\alpha_i}$	
	$\xleftarrow{\beta_i}$	$\beta_i \leftarrow R_{B,\text{bit } i} \oplus i_{B,\text{bit } i} \oplus \widehat{\alpha}_i$
End of Distance Bounding Phase		
	$\xrightarrow{d'_A}$	$0 \Vert \widehat{R}_A \leftarrow \mathsf{open}(\widehat{c}'_A, \widehat{d}'_A)$
$1 \Vert \widehat{R}_B \leftarrow \mathsf{open}(\widehat{c}'_B, \widehat{d}'_B)$	$\xleftarrow{d'_B}$	
$\widehat{i}_{B,\text{bit } i} \leftarrow \alpha_i \oplus \widehat{\beta}_i \oplus \widehat{R}_{B,\text{bit } i}$		$\widehat{i}_{A,\text{bit } 1} \leftarrow \widehat{\alpha}_1 \oplus \widehat{R}_{A,\text{bit } 1}$
		$\widehat{i}_{A,\text{bit } i} \leftarrow \widehat{\alpha}_i \oplus \beta_{i-1} \oplus \widehat{R}_{A,\text{bit } i}$
check $i_A = \widehat{i}_B$		check $i_B = \widehat{i}_A$
check integrity region		check integrity region
output: $\widehat{\mathsf{id}}_B$, i_A		**output:** $\widehat{\mathsf{id}}_A$, i_B

Figure 2.15. *Key Agreement Protocol Using Distance Bounding.*

Brands and Chaum [BC93] proposed a method that prevents frauds where an adversary runs a man-in-the-middle attack between a legitimate prover and a legitimate verifier. Frauds where a malicious prover and an adversary collaborate to cheat a verifier have been left opened. Bussard and Bagga [BB05] propose a solution for preventing both types of attacks.

Distance-bounding authentication cannot be used through the Internet since users are in general far apart and some other users are closer to the participants (with very high probability). In general, this is not very useful for wired links.

In conclusion, this method is not useful for authentication in large networks, but is well adapted for local wireless networks such as mobile phones, headset or PDAs which are very close. We do not go into further details in distance bounding protocols since we concentrate on general methods to establish worldwide authentication.

2.6 Setting up a Secure Communication in a Nutshell

This section summarizes the whole chapter focusing on solutions that can operate around the world.

Considering an extra channel achieving authentication. If we assume that an extra authenticated channel is available, then it is quite trivial to authenticate messages. Thanks to authenticated messages, users are able to agree on a common shared secret key by using a key agreement protocol following the MDH model as depicted in Figure 2.3. Users are also able to agree on a shared secret key by using the semi-authenticated key transfer as depicted in Figure 2.10.

Since the authentication of long messages is tedious, users will prefer to use a *user-aided message authentication protocol*. Such protocols aim to optimizing the amount of authenticated communication. Indeed, the (long) message to be authenticated is sent over the insecure channel and additional (short) authentic information is sent over the extra authenticated channel. In fact, it is a trade-off between the amount of authentic communication and the probability of success of a malicious party. All details concerning message authentication protocols are given in Part I.

Considering a trusted third party. If the users are not in an *ad-hoc* network and a trusted third party, e.g., a PKI, is accessible, then the authentication of messages can be trivially done by using a *signature scheme*. To agree on a shared secret key, as before they can use the MDH model of Figure 2.3 but the difference is that message authentication is done by digital signatures instead of an extra authenticated communication. Details related to signature schemes are treated in Part II.

Considering a trusted setup phase. One may combine the advantages of both setups: the first setup requires no trusted third party and no pre-shared information while the second setup requires no user interaction. So, one may use the first solution, e.g., use a user-aided message authentication protocol, to exchange long-term signature verification keys. Then, these public keys may be used to authenticate messages, e.g., of a DH protocol. This combination requires no trusted third party and no pre-shared key. At the same time, it requires user interaction only once for several sessions (while before it required user interaction for each new session).

Following Remark 2.7 and the above summary, this thesis is split in two parts: the first part focuses on user-aided message authentication protocols while the second part focuses on digital signatures schemes.

Chapter

THREE

Preliminaries

This chapter introduces the necessary notions and definitions used throughout this thesis. In particular, it presents hash functions which is one of the most used cryptographic primitives. It also defines several models of commitment schemes such as tag-based or not, keyed or not, and different types of trapdoors. This chapter also recalls the random oracle and the common reference string models.

3.1 Notations

Let S be a finite set. We write $s \in_{\mathsf{u}} S$ to say that s is uniformly distributed in the set S and we write $s \leftarrow_{\mathrm{u}} S$ to say that s was picked uniformly from the set S.

Throughout this thesis the term "algorithm" stands for a probabilistic polynomial-time (PPT) Turing machine modeled by deterministic functions in terms of an input and random coins. When dealing with protocols, we use $\mathbf{P}$ and $\mathbf{V}$ to denote the prover and verifier respectively. If nothing else is stated, they are both considered to be PPT algorithms.

We denote by $\mathsf{prot}_{\mathbf{P}(\alpha),\mathbf{V}(\beta)}(\gamma)$ an instance of the protocol "prot" between $\mathbf{P}$ and $\mathbf{V}$. The element γ denotes the common input of all participants, e.g., public keys, while α (resp. β) describes the private input of $\mathbf{P}$ (resp. $\mathbf{V}$). Note that when the protocol is not known or is implicitly known, the interaction between the two parties can be denoted by $\langle \mathbf{P}(\alpha), \mathbf{V}(\beta) \rangle (\gamma)$.

In some cases, we need to describe only the view of a party, say for instance $\mathbf{V}$. We denote

it by $\mathsf{View}_{\mathbf{V}}(\mathsf{prot}_{\mathbf{P}(\cdot),\mathbf{V}(\cdot)}(\cdot))$. We call "the view of $\mathbf{V}$" all inputs known by $\mathbf{V}$ (including the random tape and the messages received by $\mathbf{V}$). All other messages can be computed from the view and the algorithm of $\mathbf{V}$.

Results are stated in the framework of exact security, also know as concrete security [BR96, BDJR97]. Exact security quantifies the adversary resources, e.g., the running time, the number of oracle queries, etc. Then, it measures the success probability of the adversary as a function of these resources. We denote any adversary bounded by a time complexity T as a T-time adversary. Our main goal is to construct protocols that are secure against all T-time adversaries. In particular, all security properties are formally specified by a game or a game pair between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$. For a single game $\mathcal{G}$, the advantage is defined by $\mathrm{Adv}(\mathcal{A}) = \Pr[\mathcal{G}^{\mathcal{A}} = 1]$ where $\mathcal{G} = 1$ indicates a successful attack. For a game pair $\mathcal{G}_0, \mathcal{G}_1$, the advantage is defined by $\mathrm{Adv}(\mathcal{A}) = |\Pr[\mathcal{G}_0^{\mathcal{A}} = 1] - \Pr[\mathcal{G}_1^{\mathcal{A}} = 1]|$. Typically, one requires that for all T-time adversaries $\mathcal{A}$ the advantage $\mathrm{Adv}(\mathcal{A})$ is upper bounded by ε.

3.2 Hash functions

Hash function is one of the most important primitives in cryptography. A hash function H is usually used as *compression function* or as *random function*. It takes a message m of arbitrary finite length as input and outputs a fixed-length digest $h = \mathsf{H}(m)$. We denote by $\mathcal{M}$ the space of input messages and by $\mathcal{H}$ the space of output digests. In short, we have

$$\begin{aligned} \mathsf{H}: \quad \mathcal{M} &\to \mathcal{H} \\ m &\mapsto h = \mathsf{H}(m) \ . \end{aligned}$$

3.2.1 Collision Resistant Hash Functions

Collision Resistant Hash Functions (CRHF) are hash functions in which it is hard to construct two different inputs which collide[1]. More precisely, given a CRHF H it is hard to find two different input messages m_1 and m_2, $m_1 \neq m_2$, such that $\mathsf{H}(m_1) = \mathsf{H}(m_2)$. Clearly, an adversary $\mathcal{A}$ against H is playing the CR game with a challenger $\mathcal{C}$ as depicted in Figure 3.1.

Definition 3.1 (Collision Resistance).
We say that H *is* (T, ε)*-CR if any* T*-time adversary* $\mathcal{A}$ *wins the CR game of Figure 3.1 with probability at most* ε*.*

This definition is nonsense mathematically. For simplicity, we will keep it as an *informal* definition as we will not be using it so much. See Rogaway [Rog06] for more discussion.

[1] This definition is not so formal as discussed in Rogaway [Rog06].

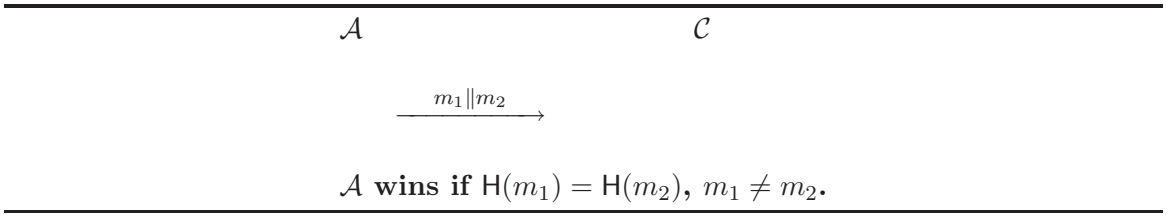


Figure 3.1. *The CR Game.*

3.2.2 Weakly Collision Resistant Hash Functions

Weakly Collision Resistant Hash Functions (WCRHF) are hash functions in which, given a message m, it is infeasible to produce a different message $\widehat{m}$, with $\widehat{m} \neq m$, such that $\mathsf{H}(\widehat{m}) = \mathsf{H}(m)$ where H is a WCRHF. Weak collision resistance is also known as *second preimage resistance*. Clearly, an adversary $\mathcal{A}$ against H is playing the WCR game with a challenger $\mathcal{C}$ as depicted in Figure 3.2.

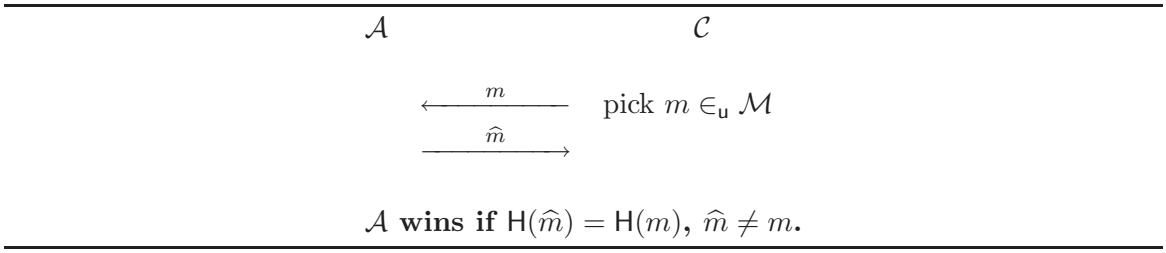


Figure 3.2. *The WCR Game.*

Definition 3.2 (Weakly Collision Resistance).
We say that H *is* (T, ε)*-WCR if any* T*-time adversary* $\mathcal{A}$ *wins the WCR game of Figure 3.2 with probability at most* ε*.*

3.2.3 Keyed and Multi-Keyed Hash Functions

A keyed hash function $\mathsf{H} : \mathcal{M} \times \mathcal{K} \to \mathcal{H}$ takes two arguments: a message of arbitrary finite length $m \in \mathcal{M}$ and a key $k \in \mathcal{K}$, and outputs a fixed length digest $h \in \mathcal{H}$:

$$\begin{aligned} \mathsf{H}: \quad \mathcal{M} \times \mathcal{K} &\to \mathcal{H} \\ (m, k) &\mapsto h = \mathsf{H}_k(m) \end{aligned}$$

Keyed hash functions are commonly used as building blocks in protocols. For example, two participants who share a secret key $k \in_{\mathsf{u}} \mathcal{K}$ can add a digest h to each exchanged message to protect it from tampering. A potential adversary can carry out two types of attacks. First, the adversary might try to impersonate a key holder by creating a valid (message, tag) pair

$(\widehat{m}, \widehat{h})$ with no additional information. Secondly, the adversary might try to substitute a message m by altering the corresponding pair (m, h). Security against impersonation and substitution attacks depends on the *regularity* and the *universality* of the hash function as defined below.

A keyed hash function is often referred to as a hash function family. The key is used to select one function among the family. Hash function families were first introduced by Carter and Wegman [CW79] under the name *universal hash function family*. Wegman and Carter [WC81] defined *strongly universal* hash families. They also introduced *almost strongly universal* hash families informally. Stinson [Sti91, Sti94] defined ε-*almost universal* hash families and ε-*almost strongly universal* hash families. In the first case, ε denotes a collision probability, while in the second case ε denotes a deception probability in the corresponding unconditionally secure MAC. Krawczyk [Kra94] defined ε-otp-secure hash families. They were renamed by Rogaway [Rog95] and they are today known as ε-*almost xor universal* hash families.

The notion of keyed hash functions can be extended to hash functions with many sub-keys, i.e., for

$$\mathsf{H} : \mathcal{M} \times \mathcal{K}_1 \times \cdots \times \mathcal{K}_n \to \mathcal{H} \ .$$

In fact, the many sub-keys may be interpreted as a big key split in n parts.

Definition 3.3 (Almost Regular Hash Function).
A hash function H *is said* ε_{ar}-almost regular *if for any* $m \in \mathcal{M}$ *and any* $h \in \mathcal{H}$*, we can write:*

$$\Pr\left[k \in_{\mathsf{u}} \mathcal{K} : \mathsf{H}(m, k) = h\right] \leq \varepsilon_{\text{ar}} \ .$$

A multi-keyed hash function H *is said* ε_{ar}-almost regular with respect to the sub-key k_i *if for any* $m \in \mathcal{M}$*, any* $\widehat{k}_1, \ldots, \widehat{k}_{i-1}, \widehat{k}_{i+1}, \ldots, \widehat{k}_n \in \mathcal{K}_1 \times \cdots \times \mathcal{K}_{i-1}, \mathcal{K}_{i+1} \times \cdots \times \mathcal{K}_n$*, and any* $h \in \mathcal{H}$*, we can write:*

$$\Pr\left[k_i \in_{\mathsf{u}} \mathcal{K}_i : \mathsf{H}(m, \widehat{\vec{k}}) = h\right] \leq \varepsilon_{\text{ar}} \ ,$$

where $\widehat{\vec{k}}$ *denotes the vector* $(\widehat{k}_1, \ldots, \widehat{k}_{i-1}, k_i, \widehat{k}_{i+1}, \ldots, \widehat{k}_n)$.

Definition 3.4 (Regular Hash Function).
A keyed or multi-keyed hash function H *is* regular *if it is* $\frac{1}{|\mathcal{H}|}$*-almost regular.*

Definition 3.5 (Almost Universal Hash Function).
A hash function H *is said* ε_{au}-almost universal*, if for any two distinct inputs* $m_0, m_1 \in \mathcal{M}$ *we can write:*

$$\Pr\left[k \in_{\mathsf{u}} \mathcal{K} : \mathsf{H}(m_0, k) = \mathsf{H}(m_1, k)\right] \leq \varepsilon_{\text{au}} \ .$$

Assume $\mathcal{K}_1 = \mathcal{K}_2 = \ldots = \mathcal{K}_n = \mathcal{K}$. *A multi-keyed hash function* H *is* ε_{au}-almost universal with respect to the sub-key pairs, *if for any two distinct inputs* $m_0, m_1 \in \mathcal{M}$, *any indexes* i, j, *and any* $k_1, \ldots, k_n, \widehat{k}_1, \ldots, \widehat{k}_n \in \mathcal{K}$, *we can write:*

$$\Pr\left[k_* \in_{\mathsf{u}} \mathcal{K} : \mathsf{H}(m_0, \vec{k}) = \mathsf{H}(m_1, \widehat{\vec{k}})\right] \leq \varepsilon_{\text{au}} \ ,$$

where $\vec{k} = (k_1, \ldots, k_{i-1}, k_*, k_{i+1}, \ldots, k_n)$ *and* $\widehat{\vec{k}} = (\widehat{k}_1, \ldots, \widehat{k}_{j-1}, k_*, \widehat{k}_{j+1}, \ldots, \widehat{k}_n)$ *and the equality* $i = j$ *is allowed.*

Definition 3.6 (Universal Hash Function).
A keyed or multi-keyed hash function H *is* universal *if it is* $\frac{1}{|\mathcal{H}|}$*-almost universal.*

Definition 3.7 (Almost Strongly Universal Hash Function).
A hash function H *is said* ε_{asu}-almost strongly universal, *if for any* a, *any* b, *and any two distinct inputs* $m_0, m_1 \in \mathcal{M}$, *we can write:*

$$\Pr\left[k \in_{\mathsf{u}} \mathcal{K} : \mathsf{H}(m_0, k) = a, \mathsf{H}(m_1, k) = b\right] \leq \frac{\varepsilon_{\text{asu}}}{|\mathcal{H}|} \ .$$

Assume $\mathcal{K}_1 = \mathcal{K}_2 = \ldots = \mathcal{K}_n = \mathcal{K}$. *A multi-keyed hash function* H *is* ε_{asu}-almost strongly universal with respect to the sub-key pairs, *if for any* a, *any* b, *any two distinct inputs* $m_0, m_1 \in \mathcal{M}$, *any indexes* i, j, *and any* $k_1, \ldots, k_n, \widehat{k}_1, \ldots, \widehat{k}_n \in \mathcal{K}$, *we can write:*

$$\Pr\left[k_* \in_{\mathsf{u}} \mathcal{K} : \mathsf{H}(m_0, \vec{k}) = a, \mathsf{H}(m_1, \widehat{\vec{k}}) = b\right] \leq \frac{\varepsilon_{\text{asu}}}{|\mathcal{H}|} \ ,$$

where $\vec{k} = (k_1, \ldots, k_{i-1}, k_*, k_{i+1}, \ldots, k_n)$ *and* $\widehat{\vec{k}} = (\widehat{k}_1, \ldots, \widehat{k}_{j-1}, k_*, \widehat{k}_{j+1}, \ldots, \widehat{k}_n)$ *and the equality* $i = j$ *is allowed.*

Definition 3.8 (Strongly Universal Hash Function).
A keyed or multi-keyed hash function H *is* strongly universal *if it is* $\frac{1}{|\mathcal{H}|}$*-almost strongly universal.*

Definition 3.9 (Almost XOR-Universal Hash Function).
A hash function H *is said* ε_{axu}-almost XOR-universal, *if for any two distinct inputs* $m_0, m_1 \in \mathcal{M}$ *and any difference* $\Delta h \in \mathcal{H}$, *we can write:*

$$\Pr\left[k \in_{\mathsf{u}} \mathcal{K} : \mathsf{H}(m_0, k) \oplus \mathsf{H}(m_1, k) = \Delta h\right] \leq \varepsilon_{\text{axu}} \ .$$

Assume $\mathcal{K}_1 = \mathcal{K}_2 = \ldots = \mathcal{K}_n = \mathcal{K}$. *A multi-keyed hash function* H *is* ε_{axu}-almost XOR-universal with respect to the sub-key pairs, *if for any two distinct inputs* $m_0, m_1 \in \mathcal{M}$, *any difference* $\Delta h \in \mathcal{H}$, *any indexes* i, j, *and any* $k_1, \ldots, k_n, \widehat{k}_1, \ldots, \widehat{k}_n \in \mathcal{K}$, *we can write:*

$$\Pr\left[k_* \in_{\mathsf{u}} \mathcal{K} : \mathsf{H}(x_0, \vec{k}) \oplus \mathsf{H}(x_1, \widehat{\vec{k}}) = \Delta h\right] \leq \varepsilon_{\text{axu}} \ ,$$

where $\vec{k} = (k_1, \ldots, k_{i-1}, k_*, k_{i+1}, \ldots, k_n)$ *and* $\widehat{\vec{k}} = (\widehat{k}_1, \ldots, \widehat{k}_{j-1}, k_*, \widehat{k}_{j+1}, \ldots, \widehat{k}_n)$ *and the equality* $i = j$ *is allowed.*

Definition 3.10 (XOR-Universal Hash Function).
A keyed or multi-keyed hash function H *is* XOR-universal *if it is* $\frac{1}{|\mathcal{H}|}$*-almost XOR-universal.*

3.2.4 Target Collision Resistant Hash Functions

Target Collision Resistant (TCR) hash functions were introduced by Naor and Yung [NY89]. They were renamed in [BR97] as Universal One-Way Hash Functions (UOWHF). In fact, TCR hash functions are simply keyed hash functions with the universal property as in Definition 3.6.

Here, we only redefine the security by a game as it will be used in a later section. In short, by first choosing a message m, then given a key k, it is infeasible to find a different message $\widehat{m}$ such that $\mathsf{H}(\widehat{m}, k) = \mathsf{H}(m, k)$. Clearly, an adversary $\mathcal{A}$ against H is playing the TCR game with a challenger $\mathcal{C}$ as depicted in Figure 3.3.

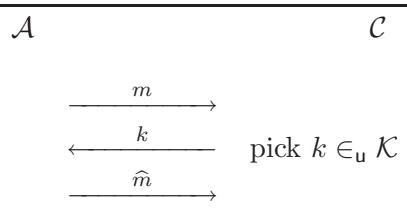


Figure 3.3. *The TCR Game.*

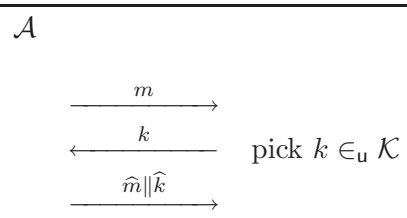
Definition 3.11 (Target Collision Resistance).
We say that H *is* (T, ε)*-TCR if any* T*-time adversary* $\mathcal{A}$ *wins the TCR game of Figure 3.3 with probability at most* ε*.*

3.2.5 Enhanced Target Collision Resistant Hash Function

Enhanced Target Collision Resistant (eTCR) hash functions were introduced by Halevi and Krawczyk [HK06a]. eTCR resistance is a stronger notion than TCR resistance. The main difference in the security games is the ability of the adversary to modify the key in the eTCR game.

As TCR functions, an eTCR function is a collection of functions $\mathsf{H}(\cdot, k)$ from a message space $\mathcal{M}$ to a finite set $\mathcal{H} = \{0,1\}^\ell$ which depends on a random parameter k picked from

a set $\mathcal{K}$. In short, by first choosing a message m, then given a key k, it is infeasible to find a different message-key pair $(\widehat{m}, \widehat{k})$, i.e., $(\widehat{m}, \widehat{k}) \neq (m, k)$, such that $\mathsf{H}(\widehat{m}, \widehat{k}) = \mathsf{H}(m, k)$. Clearly, an adversary $\mathcal{A}$ against H is playing the eTCR game with a challenger $\mathcal{C}$ as depicted in Figure 3.4.



$\mathcal{A}$ **wins if** $\mathsf{H}(\widehat{m}, \widehat{k}) = \mathsf{H}(m, k)$**,** $(\widehat{m}, \widehat{k}) \neq (m, k)$**.**

Figure 3.4. *The eTCR Game.*

Definition 3.12 (Enhanced Target Collision Resistance).
We say that H *is* (T, ε)*-eTCR if any* T*-time adversary* $\mathcal{A}$ *wins the eTCR game of Figure 3.4 with probability at most* ε*.*

We say that H is eTCR if any polynomially bounded adversary wins with negligible probability.

A OW-eTCR hash function is an eTCR hash function for which $(m, k) \mapsto \mathsf{H}(m, k)$ is also one-way (OW).

3.3 Random Oracle Model

The Random Oracle Model (ROM) consists of replacing hash functions by random oracles. While random oracles are essential in many proofs, they do not perfectly model hash functions. In Section 13.2, we will present a weaker model.

3.3.1 Random Oracle

A *Random Oracle* $\mathsf{R} : \{0,1\}^* \mapsto \{0,1\}^n$ often represents a uniformly distributed random function [BR93b].

It is simulated by an oracle managing a table $\mathcal{T}$ as follows:

- The table $\mathcal{T}$ is initially empty.
- When R receives a query with input m,

– if there is an entry (m, r) in the table $\mathcal{T}$, then it simply returns r,

– otherwise, it picks a random value $r \in_{\mathsf{u}} \{0,1\}^n$, inserts the entry (m, r) in the table $\mathcal{T}$, and returns r.

3.3.2 Pseudo-random Generator

A *Pseudo-Random Generator* (PRG) is an important primitive in cryptography. Given a short sequence of (truly) random bits, a PRG allows to generate a longer sequence of (pseudo)random bits in an efficient way. The short sequence of input random bits is often called the *seed*.

Definition 3.13 (Pseudo-random Generator).
A pseudo-random generator is a deterministic algorithm

$$\begin{aligned} G: \quad \{0,1\}^k &\to \{0,1\}^n \\ \mathsf{seed} &\mapsto R = G(\mathsf{seed}) \end{aligned}$$

satisfying the following two conditions:

Efficiency: *G is computable in polynomial-time.*

Pseudo-randomness: *The output R is computationally indistinguishable from an n-bit uniformly distributed random variable.*

The generator G is said (T, ℓ, ε)-PRG resistant if any T-time adversary $\mathcal{A}$ wins the distinguishing game of Figure 3.5 with probability at most $1/2 + \varepsilon$ (or the advantage of $\mathcal{A}$ is at most ε).

$\mathcal{A}$		$\mathcal{C}$
		pick $b \in_{\mathsf{u}} \{0,1\}$
for $i : 1..\ell$:		
select s_i	$\xrightarrow{s_i}$	if $b = 0$: $r_i \leftarrow G(s_i)$
	$\xleftarrow{r_i}$	if $b = 1$: $r_i \leftarrow_{\mathrm{u}} \{0,1\}^n$
$\widehat{b} = \mathsf{guess}(b)$		

$\mathcal{A}$ wins if $\widehat{b} = b$.

Figure 3.5. *The Distinguishing Game.*

A concrete example of such a construction is the Blum-Blum-Shub (BBS) PRG [BBS86]. This generator is very secure, however it is not very efficient. We can also refer to QUAD

recently proposed by Berbain, Gilbert, and Patarin [BGP06]. One advantage of QUAD compared to BBS is its efficiency.

3.4 Common Reference String Model

Cryptographic schemes, like commitment schemes, are often defined with key pairs. While these definitions are essential for proving the security of the overall protocol, they imply some kind of "secure" transmission of the public-key to all participants.

In the Common Reference String (CRS) model, we assume all implementations to use the same trusted public key known as the *common reference string* and denoted by crs. We also believe that no corresponding secret key is kept by anyone. Note that the use of the common public key can be "hard-coded" or can be an oracle access.

The CRS model is not so restrictive as it seems at first glance. All communication standards provide system wide public parameters such as specifications of hash functions, a group generator, or the bit length of public keys. Therefore, one should make a trade-off between computational efficiency and re-usability and size of system-wide parameters. Moreover, there are theoretic constructions that allow generation of a crs in the standard model.

3.5 Commitment Schemes

As depicted in Figure 3.6, a commitment scheme can be seen as a "locked combination safe":

> When Alice wants to commit on a message m to Bob, she places m into the "safe" and closes it (step 1). The safe is also the commitment object, denoted c, and can be given to another party, i.e., to Bob (step 2). Obviously, the message m cannot be known by other parties prior its opening, i.e., the "locked safe" is "hiding" the message (step 3). In addition, the message cannot be modified by Alice, i.e., the "locked safe" is "binding" (step 4). The message is revealed only when the decommitment object, denoted by d, is revealed. Here, d is the combination (step 5).

3.5.1 (Tag-less) Commitment Model

We can formalize a commitment scheme by two algorithms commit and open. For any message m we have $(c, d) \leftarrow \mathsf{commit}(m)$. The c value is called the *commit* value and the d value the *decommit* value. Knowing both c and d, the message can be recovered using the open algorithm, i.e., $m \leftarrow \mathsf{open}(c, d)$. As a "locked safe", a commitment scheme should be

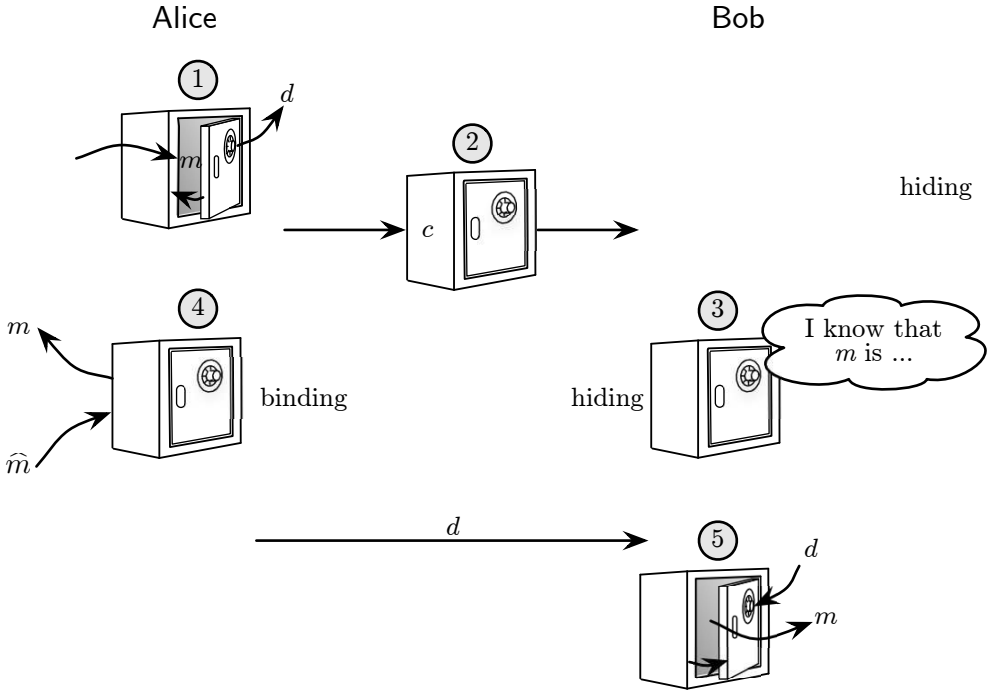


Figure 3.6. *A Combination Safe as Commitment Scheme.*

hiding, meaning that for any c, it is hard to deduce any information about the corresponding message m, and *binding*, meaning that one cannot find c,d,d' such that (c, d) and (c, d') open to two different messages.

We also introduce *keyed commitment schemes*. They have a setup algorithm to generate a key pair, i.e., $(K_p, K_s) \leftarrow \mathsf{setup}(1^\lambda)$. The public key K_p is used in commit and open oracles. Note that K_p may be empty.

3.5.2 Tag-based Commitment Model

Tag-based commitment schemes are particular commitment schemes in which the input message m is composed of two parts: a known part m_t, called tag, and a hidden part m_h. Let k be the bit length of the hidden value m_h. The setup algorithm is the same as for tag-less commitments, but the two algorithms commit and open are redefined. Finally, the three algorithms are defined as follows:

Setup. It yields a pair of keys $(K_p, K_s) \leftarrow \mathsf{setup}(1^\lambda)$.

Commit. For any key pair (K_p, K_s), any tag m_t, and any value m_h, we have $(c, d) \leftarrow \mathsf{commit}(K_p, m_t, m_h)$.

Open. Given K_p, c, d and the tag m_t, the hidden value can be recovered with $m_h \leftarrow \mathsf{open}(K_p, m_t, c, d)$.

Tag-less commitment schemes, in which the whole input message m is kept hidden, can be constructed using a tag-based commitment scheme with an empty tag, i.e., $m_t = \perp$, and a hidden value equal to the conventional message, i.e., $m_h = m$. We finally have $m = m_t \| m_h = m_h$.

3.5.3 Completeness, Hiding, and Binding Properties

We define here three of the most important properties for (tag-based) commitment schemes, i.e., the completeness, the hiding and the binding properties.

Any commitment scheme must clearly satisfy the *completeness property*, i.e., a commit value c and a decommit value d by the commit algorithm with input (m_t, m_h) should open to m_h. More formally, the completeness property is defined as follows.

Definition 3.14 (Completeness Property).
For any key pair $(K_p, K_s) \leftarrow \mathsf{setup}(1^\lambda)$, any tag m_t, any value m_h, and any $(c, d) \leftarrow \mathsf{commit}(K_p, m_t, m_h)$, we have $m_h = \mathsf{open}(K_p, m_t, c, d)$

In addition, commitment schemes should be *hiding*, i.e., the commit value c should reveal no information about the hidden value m_h. More formally, the hiding property is defined as follows.

Definition 3.15 (Hiding Property).
For any key pair $(K_p, K_s) \leftarrow \mathsf{setup}(1^\lambda)$, any tag m_t, any values $m_{h,0}, m_{h,1}$, any random bit b, and any $(c, d) \leftarrow \mathsf{commit}(K_p, m_t, m_{h,b})$, c gives no information on b.

We defeat the hiding property with the semantic hiding (SH) game which is described in Figure 3.7: the adversary $\mathcal{A}$ selects a tag m_t and two hidden values $m_{h,0}$ and $m_{h,1}$ and sends them to the challenger $\mathcal{C}$. The challenger flips an unbiased coin b and commits to $(m_t, m_{h,b})$. Given the commit value c, the adversary $\mathcal{A}$ guesses b and succeeds if the guess is correct.

Definition 3.16 (Semantic Hiding Commitment Scheme).
A scheme is said (T, ε_h)-semantically hiding if any T-time adversary $\mathcal{A}$ wins the SH game of Figure 3.7 with probability at most $\frac{1}{2} + \varepsilon_h$.

We also can defeat the hiding property with the full hiding (FH) game which is described in Figure 3.8. The adversary picks a tag m_t and sends it to the challenger $\mathcal{C}$. $\mathcal{C}$ picks a hidden value $m_h \in \{0,1\}^k$ and commits on (m_t, m_h). $\mathcal{C}$ reveals the value c to $\mathcal{A}$. Finally, $\mathcal{A}$ guesses m_h and wins the attack if the guess is correct.

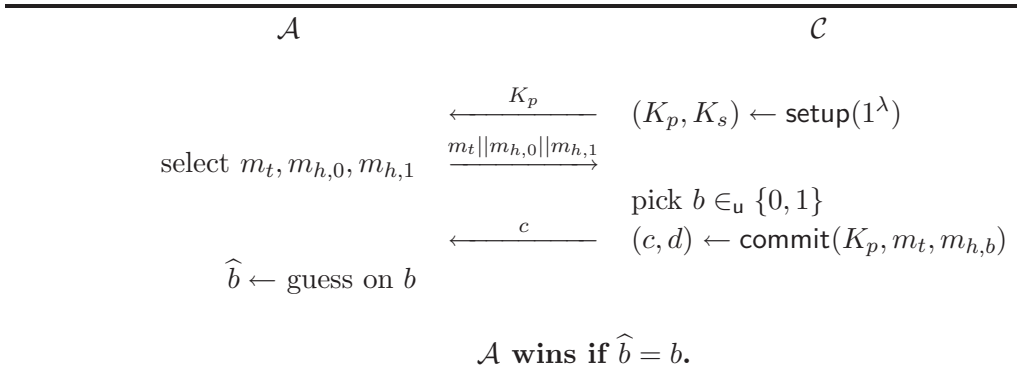


Figure 3.7. *The Semantic Hiding (SH) Game.*

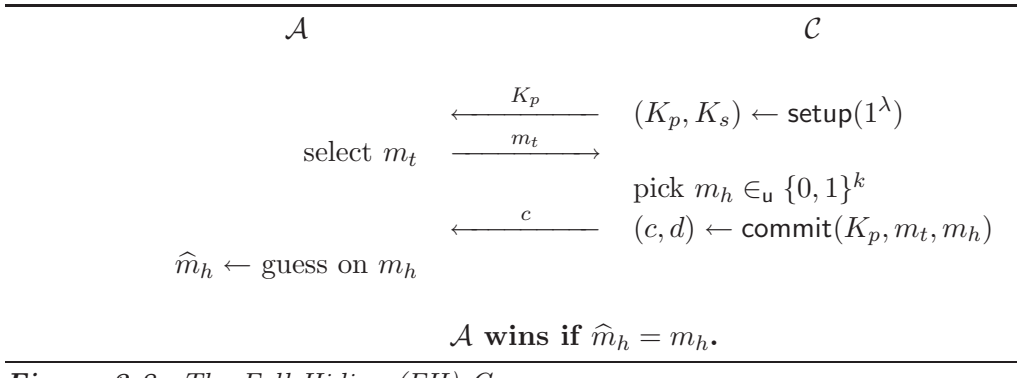


Figure 3.8. *The Full Hiding (FH) Game.*

Definition 3.17 (Full Hiding Commitment Scheme).
Let k be the bit-length of the hidden value. A scheme is said (T, ε_h)-fully hiding if any T-time adversary $\mathcal{A}$ wins the FH game of Figure 3.8 with probability at most $2^{-k} + \varepsilon_h$.

A commitment scheme is said *perfectly* (semantic or full) hiding if it is $(\infty, 0)$-(semantic or full)-hiding.

Here is a useful lemma taken from Vaudenay [Vau05b].

Lemma 3.18 (Semantic versus Full Hiding Properties).
There exists a (small) constant ν such that for any T and any ε_h, a $(T+\nu, \varepsilon_h)$-semantically hiding scheme is a $(T, 2\varepsilon_h)$-fully hiding commitment scheme.

Obviously, a $(T + \nu, \varepsilon_h)$-fully hiding commitment scheme is also (T, ε_h)-semantically hiding. Hence, the two notions of hiding commitment schemes are essentially equivalent.

In addition, commitments should be *binding*, i.e., an adversary which has committed to a message $m_t \| m_h$ by sending the commit value c cannot open to two different hidden values $m_{h,0}$ and $m_{h,1}$. More formally, the binding property is defined as follows.

Definition 3.19 (Binding property.).
For any key pair $(K_p, K_s) \leftarrow \mathsf{setup}(1^\lambda)$, it is hard to find (m_t, c, d_0, d_1) such that $m_{h,b} \leftarrow \mathsf{open}(K_p, m_t, c, d_B) \neq \perp$ for $b = 0, 1$ and $m_{h,0} \neq m_{h,1}$.

Clearly, the semantic binding (SB) game of Figure 3.9 must be hard.

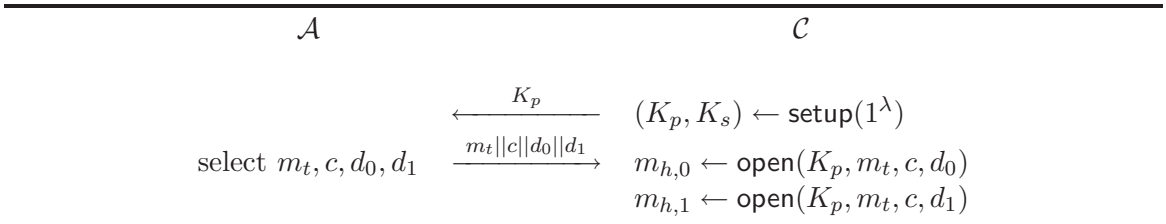


Figure 3.9. *The Semantic Binding (SB) Game.*

Definition 3.20 (Semantic Binding Commitment Scheme).
A scheme is said (T, ε_b)-semantically binding if any T-time adversary $\mathcal{A}$ wins the SB game of Figure 3.9 with probability at most ε_b.

We can also define the full binding (FB) game. As described in Figure 3.10, it works as follows: the adversary $\mathcal{A}$ selects a tag m_t and a commit value c. Then, he sends both to the challenger $\mathcal{C}$. $\mathcal{C}$ picks a random value m_h and sends it back to $\mathcal{A}$. $\mathcal{A}$ proposes a decommit value d and succeeds if it opens to m_h, i.e., if $m_h \stackrel{?}{=} \mathsf{open}(K_p, m_t, c, d)$.

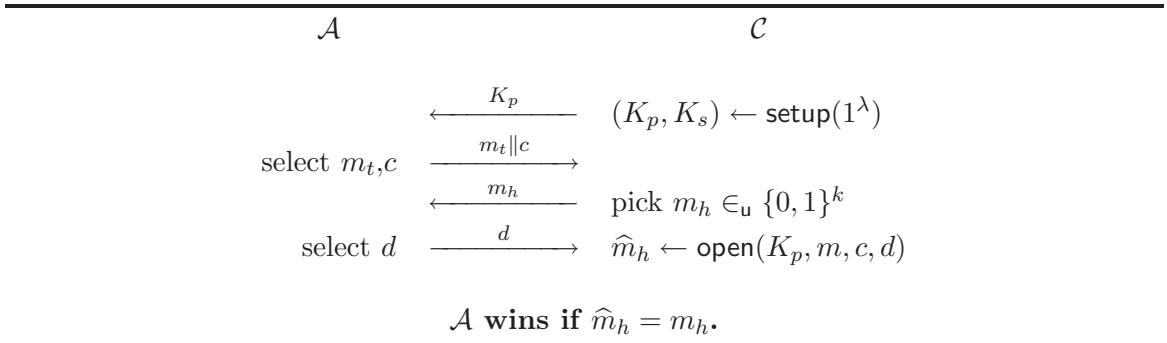


Figure 3.10. *The Full Binding (FB) Game.*

Definition 3.21 (Full Binding Commitment Scheme).
A scheme is said (T, ε_b)-fully binding if any T-time adversary $\mathcal{A}$ wins the FB game of Figure 3.10 with probability at most $2^{-k} + \varepsilon_b$.

A commitment scheme is said *perfectly* (semantic or full) binding if it is $(\infty, 0)$-(semantic or full)-binding.

3.5.4 Non-Malleability

Non-malleability is the strongest property for commitment schemes. Indeed, the binding and hiding properties directly follow from non-malleability but not vice versa. Many notions of non-malleable commitments have been proposed in the cryptographic literature [DDN91, DCIO98, FF00, DG03, LN06a]. All these definitions try to capture requirements that are necessary to defeat man-in-the-middle attacks. Here, we adopt the modernized version of non-malleability with respect to opening. The corresponding definition [LN06a] mimics the framework of non-malleable encryption [BS99] and leads to more natural security proofs compared to the simulation-based definitions [DCIO98, DG03].

For ciphers, non-malleability and security against chosen ciphertext attacks (CCA) are known to be tightly coupled. In fact, these notions coincide if the adversary is allowed to make decryption queries throughout the entire attack [BDPR97] and thus usage of decryption oracles can simplify many proofs without significantly increasing the security requirements. Unfortunately, a similar technique is not applicable to commitment schemes as there can be several different valid decommitment values d_i for a single commitment c. Thus, one must use explicit definitions of hiding, binding, and non-malleability properties in proofs.

The non-malleability property is defined by elaborated games. Thus we focus on tag-less schemes and we use an illustrative pictorial style to specify these games, see Figure 3.11 and Figure 3.12. Intuitively, the goal is: given a valid commitment c, it is infeasible to generate related commitments $\widehat{c}_1, \ldots, \widehat{c}_n$ that can be successfully opened after seeing a decommitment

value d. More formally, the adversary $\mathcal{A}$ consists of two parts: $\mathcal{A}_1$ corresponds to the *active* part of the adversary that tries to create and afterwards open commitments related to c; $\mathcal{A}_2$ captures a desired *target relation*. Note that $\mathcal{A}_1$ is a stateful algorithm and can pass information from one stage to the other but no information can be passed from $\mathcal{A}_1$ to $\mathcal{A}_2$ except some state σ. By convention, a game is ended with the output $\perp$ if any operation leads to $\perp$.

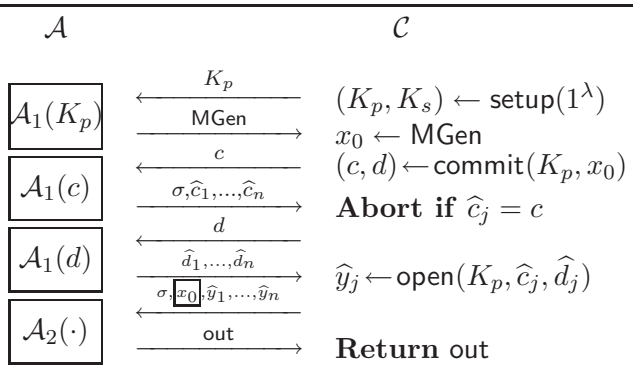


Figure 3.11. *Non-malleability Game* $\mathcal{G}_0^{\mathsf{nm}}$.

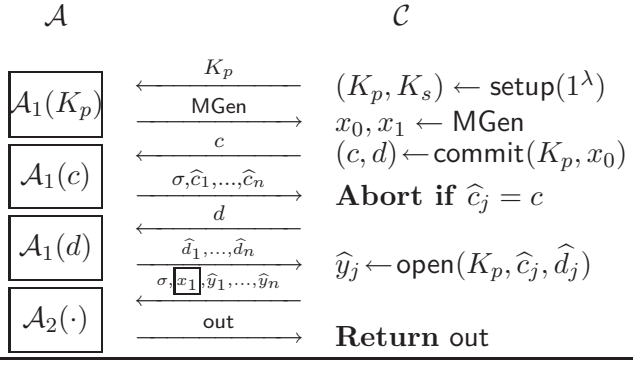


Figure 3.12. *Non-malleability Game* $\mathcal{G}_1^{\mathsf{nm}}$.

Figure 3.11 and Figure 3.12 can be interpreted as follows. In the game $\mathcal{G}_0^{\mathsf{nm}}$, a challenger $\mathcal{C}$ first generates the public parameter K_p. Given K_p, $\mathcal{A}_1$ outputs a message generator MGen. Next, $\mathcal{C}$ selects $x_0 \leftarrow \mathsf{MGen}$ and computes (c, d). Given c, $\mathcal{A}_1$ outputs some commitment values $\widehat{c}_i$ and an advice σ for $\mathcal{A}_2$ and then, given d he generates some decommitment values $\widehat{d}_i$. Finally, $\mathcal{C}$ opens all commitments $\widehat{y}_i \leftarrow \mathsf{open}(K_p, \widehat{c}_i, \widehat{d}_i)$ and tests whether $\mathcal{A}_1$ won or not by computing $\mathcal{A}_2(\sigma, x_0, \widehat{y}_1, \ldots, \widehat{y}_n)$. The condition $\widehat{c}_j \neq c$ eliminates trivial attacks. The game $\mathcal{G}_1^{\mathsf{nm}}$ is almost the same, except the challenger tests a relation $\mathcal{A}_2(\sigma, x_1, \widehat{y}_1, \ldots, \widehat{y}_n)$ instead, where $x_1 \leftarrow \mathsf{MGen}$ is chosen independently from the rest of the game. A commitment scheme is $(T, \varepsilon_{\mathrm{nm}})$-*non-malleable* with respect to to opening if the advantage of any T-time

adversary $\mathcal{A}$ is at most

$$\mathrm{Adv}^{\mathsf{nm}}_{\mathcal{C}om}(\mathcal{A}) = |\Pr[\mathcal{G}^{\mathsf{nm}}_0 = 1] - \Pr[\mathcal{G}^{\mathsf{nm}}_1 = 1]| \leq \varepsilon_{\mathrm{nm}} \ .$$

Note that $\mathcal{A}_2$ can be any computable relation that is completely fixed after seeing c. For instance, we can define $\mathcal{A}_2(\sigma, x, y)$ outputs 1 if $x = y$ and 0 otherwise. Hence, it must be infeasible to construct a commitment $\widehat{c}$ that can be opened later to the same value as the challenge commitment c.

Non-malleable commitment schemes can be built using a CCA2 secure public-key encryption scheme. However, this method is too inefficient for lightweight devices. Efficient non-malleable commitment schemes may be designed by using a hash function as detailed by Laur and Nyberg [LN06a].

3.5.5 Ideal Commitment Model

The notion of *ideal commitment model* describes a scheme which is perfectly hiding and perfectly binding.

For instance, an ideal commitment scheme can be implemented using a trusted third party (TTP) as follows:
Commitment step. The $\mathsf{commit}(m)$ algorithm consists of sending the message m securely to the TTP. The TTP binds m to a unique commit value c, inserts (c, m) in a database $\mathcal{T}$ with a protection flag, and returns c to the owner. Note that there is no decommit value. The protection flag avoids future access from anyone except the owner.
Opening step. The $\mathsf{open}(c)$ algorithm is a simple call to the TTP. The TTP clears the protection flag of (c, m) which becomes available for anyone.

To commit on m, Alice first sends m to the TTP, gets back c, and then forwards c to Bob as depicted in Figure 3.13. As depicted in Figure 3.14, to reveal the message Alice asks the

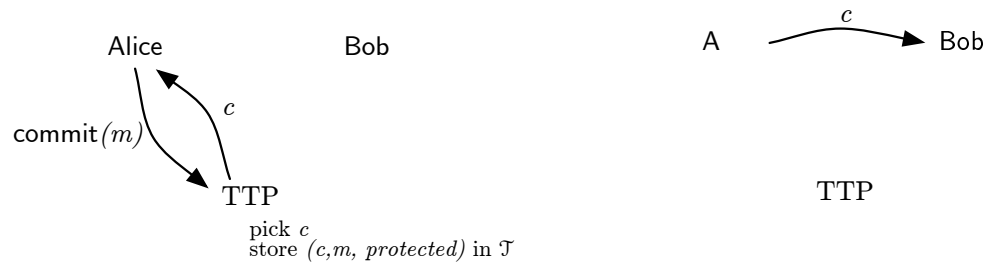


Figure 3.13. *Ideal Commitment: Commit Algorithm and Commitment Phase.*

TTP to clear the protection flag. Now, Bob can open the commitment by giving c to the TTP who sends back the message m (if the protection flag was cleared).

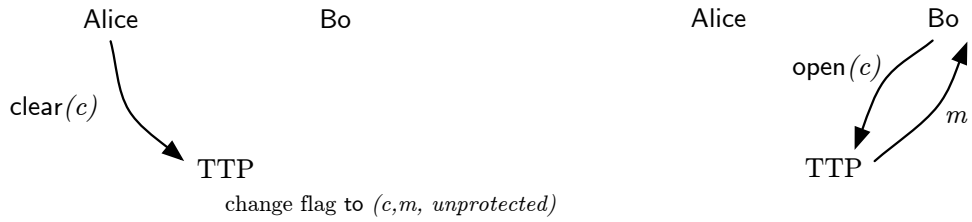


Figure 3.14. *Ideal Commitment, Decommit Algorithm and Decommitment Phase.*

Ideal commitment can also be implemented by using the notion of universal composable commitment schemes as proposed by Damgård and Nielsen [DN02]. Canetti and Fischlin [CF01] proposed a commitment scheme based on the CRS model which behaves like an "ideal commitment service".

3.5.6 Trapdoor Extractable Commitment Schemes

We define an *extractable commitment scheme* as an extension of a general tag-based commitment scheme in which there is an additional deterministic algorithm: extract.

Extract. Given m_t and c, the $m_h \leftarrow \mathsf{extract}(K_s, m_t, c)$ algorithm yields a m_h value whenever there exists d which opens to m_h, that is, $m_h \leftarrow \mathsf{open}(K_p, m_t, c, d)$.

Extractable commitments are perfectly binding since for a given m_t and a given c, only one m_h can be yielded by the extract algorithm. Thus, there exists only one d and any adversary has no advantage in the SB or FB games.

Adversaries playing the SH, FH, SB, or FB games may or may not have access to oracles. They may query an $\mathsf{extract}(K_s, \cdot, \cdot)$ oracle, except on the target tag m_t. (Note that they have no access to K_s, but the oracle has.)

3.5.7 Trapdoor Equivocable Commitment Schemes

We define an *equivocable commitment scheme* as an extension of a tag-based commitment scheme in which there are two additional oracles: simcommit and equivocate.

Simcommit. With input tag m_t, the $(c, \xi) \leftarrow \mathsf{simcommit}(K_s, m_t)$ algorithm yields a fake commit value c and an additional information value ξ using the secret key K_s.

For any $(K_p, K_s) \leftarrow \mathsf{setup}(1^\lambda)$, any m_t, and any m_h, the distribution of c should be the same as the distribution of c generated by any $\mathsf{commit}(K_p, m_t, m_h)$ for any m_h.

Equivocate. Using K_s, and with inputs (m_t, m_h, c, ξ) where c and ξ are outputs from the simcommit algorithm, the equivocate algorithm yields a decommit value $d \leftarrow \mathsf{equivocate}(K_s, m_t, m_h, c, \xi)$ such that (K_p, m_t, c, d) opens to m_h.

Access to simcommit and equivocate oracles is restricted depending on the application. The normal usage of the commitment scheme should be limited to commit and open but we stress that our security model assumes that the adversary may cheat on *some* commitments by having access to simcommit and equivocate oracles. Indeed, our notion of equivocable commitment relates to the notion of *simulation-sound* commitment [MY04].

Equivocable commitment schemes are perfectly hiding since a c can be generated before and independently of the value m_h. In the hiding game, the adversary has access to the simcommit and equivocate but cannot query simcommit with the selected tag m_t. Since the commitment is perfectly hiding, no adversary can win the hiding game with a probability larger than 2^{-k} where k is the length of the message m_h. In the binding game, the adversary has access to the simcommit and equivocate oracles but cannot query simcommit with the selected tag m_t. We say that the equivocable commitment with k-bit values m_h is (T, ε)-secure if any T-time adversary wins in the FB game with probability at most $2^{-k} + \varepsilon$.

Secure equivocable commitment schemes can be easily constructed based on simulation-sound trapdoor commitments by MacKenzie-Yang [MY04] as detailed in [Vau05b]. Constructions can be in the CRS model, e.g., based on the security of DSA signatures [DSS00] or Cramer-Shoup signatures [CS02]. We can also build an efficient equivocable commitment scheme in the ROM.

3.5.8 Trapdoor Commitment Model

Trapdoor commitment schemes were introduced by Brassard, Chaum, and Crépeau [BCC88]. We define (tag-less) (T, ε)-*trapdoor commitment schemes* by four algorithms setup, commit, open, and equivocate. The first three algorithms work as before. The equivocate algorithm defeats the binding property by using the secret key K_s.

There is no fake commitment algorithm. Indeed, the algorithm equivocate needs no additional information except K_s. This primitive is a particular case of *strongly equivocable commitment* as described by Damgård and Groth [DG03].

More precisely, for any $(K_p, K_s) \leftarrow \mathsf{setup}(1^\lambda)$, a trapdoor commitment scheme is such that

Commitment properties. The algorithms setup, commit and open form a tag-less commitment scheme which satisfies the completeness property, which is perfectly hiding, and which is (T, ε)-binding.

Trapdoor property. For any message m, the two distributions

$$(c, d) \leftarrow \mathsf{commit}(K_p, m) \qquad \text{and} \qquad (c \in_{\mathsf{u}} \mathcal{C},\ d \leftarrow \mathsf{equivocate}(K_s, m, c))$$

are indistinguishable.

For instance, a trapdoor commitment based on the discrete logarithm problem was proposed by Boyar and Kurtz [BKK90]. Another trapdoor commitment scheme was proposed by Catalano *et al.* [CGHGN01] which is detailed in the next section.

Trapdoor commitment schemes are perfectly hiding and computationally binding commitment schemes. Note that it is impossible to distinguish whether a committed value and its corresponding decommit value, i.e., (c, d), have been yield by using the standard commit algorithm or by choosing c uniformly and by using the equivocate algorithm.

In the Damgård and Groth construction, adversaries can query $\mathsf{equivocate}(K_s, \cdot, \cdot, \cdot)$ oracle (except on the selected c). In the Boyar and Kurtz and in the Catalano *et al.* constructions, they cannot.

3.5.9 Examples

3.5.9.1 Extractable Random Oracle Commitment

Extractable commitment schemes can be designed from a random oracle R. Such schemes were defined in Pass [Pas03]. Given any input d, R yields a value $c \in \{0,1\}^{\ell_c}$, i.e., $c \leftarrow \mathsf{R}(d)$.

Setup. There is no setup.

Commit. The $\mathsf{commit}(m_t, m_h)$ algorithm with $m_h \in \{0,1\}^{\ell_h}$ picks a random value $e \in_{\mathsf{u}} \{0,1\}^{\ell_e}$, builds $d \leftarrow m_h \| e$, and calls the random oracle $c \leftarrow \mathsf{R}(m_t \| d)$.

Open. The $\mathsf{open}(m_t, c, d)$ algorithm checks that $c \stackrel{?}{=} \mathsf{R}(m_t \| d)$ and extracts m_h from d.

Extract. An $\mathsf{extract}(m_t, c)$ algorithm can be added. Indeed, when there was no collision on the pairs (m_t, c), the m_h can be trivially extracted.

By allowing q queries to R, the scheme is $(\infty, q \cdot 2^{-\ell_e - \ell_h})$-binding with probability at least $1 - q^2 \cdot 2^{-\ell_c - 1}$ [Vau05b]. In practice this scheme is pretty safe with $\ell_c = 2\ell_e$ and $\ell_e = 80$ (see [Vau05b]) and the oracle R can typically be a hash function, e.g., SHA-1 [SHA93, SHA95].

3.5.9.2 Equivocable Random Oracle Commitment

A random oracle *ROR* can also be used to build equivocable commitment schemes. In that case, queries are of the form $c \leftarrow \mathsf{R}(m_t, m_h, e)$ queried with a tag m_t, an ℓ_h-bit string m_h, and an ℓ_e-bit string e. In short, it looks whether an entry (m_t, m_h, e, c) exists in the table $\mathcal{R}$. If not, the oracle creates the entry with a random ℓ_c-bit string c. In any case, the oracle answers c.

Let ℓ_c, ℓ_e, and ℓ_h be three integers. The *equivocable commitment* works as follows:

Setup. There is no setup.

Commit. The algorithm $(c, d) \leftarrow \mathsf{commit}(m_t, m_h)$ simply picks e at random, queries $c \leftarrow \mathsf{R}(m_t, m_h, e)$, sets $d = (m_h, e)$, and outputs (c, d) .

Open. The algorithm $m_h \leftarrow \mathsf{open}(m_t, c, d)$ simply checks that $c \stackrel{?}{=} \mathsf{R}(m_t, d)$ and extracts m_h from d.

Simcommit. $(c, i) \leftarrow \mathsf{simcommit}(m_t)$ simply picks a random ℓ_c-bit string c and a nonce i and stores (i, c, m_t) in a table $\mathcal{T}$.

Equivocate. $d \leftarrow \mathsf{equivocate}(i, m_h)$ gets (i, c, m_t) and removes it form the table $\mathcal{T}$. The oracle then picks a random ℓ_e-bit string e. If $(m_t, m_h, e, \cdot)$ exists in the table $\mathcal{R}$, the oracle fails. Otherwise, it inserts (m_t, m_h, e, c) in $\mathcal{H}$.

Clearly, if the number of oracle accesses to R and $\mathsf{simcommit}$ is limited by q, the probability that the oracle fails at least once is less than $q^2/2 \cdot 2^{-\ell_e}$.

Unless $\mathsf{equivocate}$ fails, this scheme is clearly an equivocable commitment scheme as previously defined. Since all commit values c are generated in an independent way, there are no collisions with probability at least $1 - q^2/2 \cdot 2^{-\ell_c}$. Clearly, being able to decommit any c to two values would lead R to a collision. Hence, the scheme is $\left(\infty, 2^{-\ell_h} + q^2 \cdot 2^{-\ell_e - 1} + q^2 \cdot 2^{-\ell_c - 1}\right)$-secure.

In practice, $\mathsf{simcommit}$ and $\mathsf{equivocate}$ are unused. So, we can just instantiate R by a standard hash function, provided that an instantiation of that kind for a random oracle makes sense [CGH98].

3.5.9.3 Paillier-based Trapdoor Commitment Scheme

A trapdoor commitment scheme was proposed by Catalano *et al.* [CGHGN01] based on the Paillier's trapdoor permutation [Pai99]. The proposed scheme is tag-less. It uses an RSA modulus $N = pq$ and a value $h \in \mathbb{Z}_{N^2}$ such that its order is a multiple of N. The public key is $K_p \leftarrow (N, h)$ and the private key is $K_s \leftarrow (p, q, h)$.

The $\mathsf{setup}(1^\lambda)$ algorithm outputs the two keys, K_p and K_s, as described previously.

The $\mathsf{commit}(K_p, m)$ algorithm of a message $m \in \mathbb{Z}_N$ picks uniformly two random values r, s and outputs $c \leftarrow (1 + mN)r^N h^s \bmod N^2$ and $d \leftarrow (r, s)$. Note that the commit value c is uniformly distributed for any m since r and s are uniformly distributed and $(r, s) \mapsto r^N h^s \bmod N^2$ is the Paillier trapdoor permutation (see [Pai99]). We denote by $\mathcal{F}_h(r, s)$ this permutation.

The $\mathsf{open}(K_p, c, d)$ algorithm yields m by solving $c = (1 + mN)r^N h^s \bmod N^2$ from $d = (r, s)$ and $K_p = (N, h)$.

The $\mathsf{equivocate}(K_s, m, \widehat{c})$ algorithm uses the collision-finding function, i.e., given a commit $\widehat{c}$ and a message m, one can find $\widehat{d} \leftarrow (\widehat{r}, \widehat{s})$ such that $\widehat{c} = (1+mN)\mathcal{F}_h(\widehat{r}, \widehat{s}) \bmod N^2$ by using the trapdoor on the Paillier permutation and knowing p, q, i.e., $(\widehat{r}, \widehat{s}) \leftarrow \mathcal{F}_h^{-1}(\widehat{c}(1+mN)^{-1})$. Thus, given a $\widehat{c}$, an adversary can find $\widehat{d}$ for any message m and thus defeat the binding property.

3.5.9.4 Pedersen Commitment Scheme

An interesting and simple tag-less commitment scheme was proposed by Pedersen in [Ped91].

The setup phase consists in choosing two prime numbers p and q such that q divides $p - 1$. Let G_q be the unique subgroup of order q of $\mathbb{Z}_p^*$ and let g be a generator of G_q. In addition, the algorithm picks a random $y \in G_q$ such that nobody knows $\log_g(y)$. The public parameters are (p, q, g, y). Note that the setup phase can be done by a trusted third party (TTP).

The $\mathsf{commit}(K_p, m)$ algorithm with input message $m \in \mathbb{Z}_q$ starts by picking a random $\ell \in ZZ_q$ and then computes $c \leftarrow g^m y^\ell$. The decommit value is simply $d \leftarrow (m, \ell)$.

The $\mathsf{open}(K_p, c, d)$ algorithm extracts m and ℓ from d. Then it checks that $c \stackrel{?}{=} g^m y^\ell$. If the equality holds then it returns m otherwise it returns $\perp$.

This scheme is perfectly hiding since y^ℓ is a random value and thus c reveals no information about m. On the other hand, this scheme is computationally binding. Indeed, an adversary able to win against the tag-less SB game proceeds as follows: he discloses some c, d, d' such that $d = (m, \ell)$, $d' = (m', \ell')$, $m \neq m'$, and $c = g^m y^\ell = g^{m'} y^{\ell'}$. The secret parameter leaks by $\log_g(h) = \frac{m-m'}{\ell'-\ell}$

Another vulnerability is its malleability. Indeed, an adversary who gets the commit value c can commit to a relative message $\widehat{m}$ by committing on $\widehat{c} \leftarrow c \cdot g^\delta$. When the adversary obtains the decommit value d, i.e., (m, ℓ), he can compute its own decommit value $\widehat{d} \leftarrow (m + \delta, \ell)$. The verifier checks $\widehat{c} = g^{m+\delta} h^\ell$. Thus, any adversary can commit on a related value to m, e.g., $m + \delta$.

A trapdoor can easily be added to this commitment scheme. It simply consists in picking a $x \in_{\mathsf{u}} \mathbb{Z}_q^*$ and then letting $y \leftarrow g^x$ instead of choosing y randomly. Now, there is a public key, i.e., $K_p \leftarrow (p, q, g, y)$, and a private key, i.e., $K_s \leftarrow x$.

We have first to define the fake commitment algorithm $\overline{\mathsf{commit}}$. It consists in yielding $(c, \xi) \leftarrow \mathsf{commit}(m)$ where $\xi \leftarrow (m, \ell)$ for random m and ℓ and $c \leftarrow g^m y^\ell$.

The $\mathsf{equivocate}$ algorithm with input $\widehat{m}$ yields $\widehat{d}$ which opens to $\widehat{m}$ using the fixed commit value c. In fact, it has to find a $\widehat{\ell}$ such that $g^m y^\ell = g^{\widehat{m}} y^{\widehat{\ell}}$. This latter condition can be written as $\widehat{m} - m = (\ell - \widehat{\ell}) \log_g(y)$. Knowing the trapdoor, i.e., $x = \log_g(h)$, an adversary can find the corresponding $\widehat{\ell}$, i.e., $\widehat{\ell} \leftarrow \ell - \frac{\widehat{m}-m}{x}$.

3.5.9.5 A Tag-based Equivocable Commitment Scheme

Here, we present a tag-based trapdoor commitment scheme introduced by MacKenzie and Yang [MY04]. Their scheme is based on the Pedersen commitment scheme [Ped91] and on the DSA signature scheme [DSS94, DSS00].

The $\mathsf{setup}(1^\lambda)$ algorithm generates a DSA pair of keys (K_p, K_s), i.e.,

$$K_p = (g, p, q, y), \qquad K_s = x \ .$$

The $\mathsf{commit}(K_p, m_t, m_h)$ algorithm with $m_h \in \mathbb{Z}_q$ picks a random k and then computes g' and h as follows:

$$k \in_{\mathsf{u}} \mathbb{Z}_q \ , \qquad g' \leftarrow g^k \bmod p \ , \qquad h \leftarrow g^{H(m_t)} y^{g'} \bmod p \ .$$

Note that (r_{DSA}, s_{DSA}) is a DSA signature of m where $r_{DSA} = g' \bmod q$ and $s_{DSA} = \log_{g'}(h)$, i.e., $s_{DSA} = \frac{H(m_t) + x r_{DSA}}{k}$.

Then, it uses the Pedersen commitment scheme to commit on m_h using (g', h) as public parameters, i.e., it picks ℓ and computes c' as follows:

$$\ell \in \mathbb{Z}_q \ , \qquad c' \leftarrow \left(g'\right)^\ell h^{m_h} \bmod p \ .$$

Finally, the commit value c and the decommit value d are the followings:

$$c \leftarrow (g', c') \ , \qquad d \leftarrow (m_h, \ell) \ .$$

The $\mathsf{open}(K_p, m_t, c, d)$ algorithm computes h as follows:

$$h \leftarrow g^{H(m_t)} y^{g'} \bmod p \ .$$

Then, it checks that

$$c' \stackrel{?}{=} \left(g'\right)^{\ell} h^{m_h} \pmod{p}$$

and if the condition is verified it outputs m_h, otherwise it outputs $\perp$.

These three algorithms form a commitment scheme. In addition, there is a $\overline{\mathsf{commit}}(K_s, m_t)$ algorithm which uses the secret key K_s to yield a fake commit value and additional information which are required by the $\mathsf{equivocate}$ algorithm.

The $\overline{\mathsf{commit}}(K_s, m_t)$ algorithm computes a DSA signature $\sigma_{DSA} \leftarrow (r_{DSA}, s_{DSA})$ on m_t by using the secret key K_s, i.e.,

$$k \in \mathbb{Z}_q^* \ , \qquad r_{DSA} \leftarrow g^k \bmod p \bmod q \ , \qquad s_{DSA} \leftarrow \frac{H(m_t) + x r_{DSA}}{k} \bmod q \ .$$

Then, it computes g' and h as

$$h \leftarrow g^{H(m_t)} y^{r_{DSA}} \bmod p \ , \qquad g' \leftarrow h^{s_{DSA}^{-1}} \bmod p$$

and picks ℓ and computes c' as

$$\ell \in \mathbb{Z}_q \ , \qquad c' \leftarrow h^{\ell} \bmod p \ .$$

Finally, the fake commit value $\widehat{c}$ and the auxiliary information ξ are

$$\widehat{c} \leftarrow (g', c') \ , \qquad \xi \leftarrow (\ell, s_{DSA}) \ .$$

Note that g' is in fact equal to $g^k \bmod p$.

The $\mathsf{equivocate}(K_s, m_t, \widehat{m}_h, \widehat{c}, \xi)$ algorithm outputs a fake decommit value $\widehat{d}$ such that $\mathsf{open}(K_p, m_t, \widehat{c}, \widehat{d})$ yields $\widehat{m}_h$. Thus, it must find a $\widehat{d}$, i.e., a $\widehat{\ell}$ (since $\widehat{m}_h$ is given), such that

$$c' = (g')^{\widehat{\ell}} h^{\widehat{m}_h} \pmod{p}$$

which can be written as

$$c' = (g')^{\widehat{\ell}} g^{\widehat{m}_h H(m_t)} y^{\widehat{m}_h g'} \pmod{p} \ .$$

But, we know that $y = g^x \bmod p$ and $g' = g^k \bmod p$ and we obtain

$$c' = g^{k\widehat{\ell} + \widehat{m}_h H(m_t) + x\widehat{m}_h g'} \pmod{p} \ .$$

In reality, the c' value yielded by the $\overline{\mathsf{commit}}$ algorithm is

$$c' = h^{\ell} = g^{H(m_t)\ell + x r_{DSA} \ell} \pmod{p} \ .$$

Recall from DSA that g is of order q. Thus, we can deduce that

$$k\widehat{\ell} + \widehat{m}_h H(m_t) + x\widehat{m}_h g' = H(m_t)\ell + x r_{DSA} \ell \pmod{q} \ .$$

Note that $r_{DSA} = g' \bmod q$. We have

$$\widehat{\ell} = (\ell - \widehat{m}_h)\frac{H(m_t) + xr_{DSA}}{k} \pmod q \ .$$

Consequently, we obtain

$$\widehat{\ell} = (\ell - \widehat{m}_h)s_{DSA} \pmod q$$

and this is the reason why ℓ and s_{DSA} are in the extra information ξ.

Note that this scheme has a stronger binding property called *simulation sound binding property* which guarantees that a commitment made by an adversary with tag m_t is binding even if he saw many *simulated* commit values but never a commitment with m_t. It is showed in [MY04] that if an adversary can break this property then it can also break DSA.

3.6 Entropies

We provide the necessary quantitative definitions of the entropy of a random variable.

Definition 3.22 (Min-Entropy).
Let X a random variable in a set $\mathcal{X}$ with distribution $\mathcal{D}$. We define the min-entropy *of X by*

$$H_\infty(\mathcal{D}) = -\log \max_{x \in_{\mathcal{D}} \mathcal{X}} \Pr[X = x] \ .$$

Definition 3.23 (Renyi Entropy).
Let X a random variable in a set $\mathcal{X}$ with distribution $\mathcal{D}$. We define the Renyi entropy *(of order 2) of X by*

$$H_2(\mathcal{D}) = -\log \sum_{x \in_{\mathcal{D}} \mathcal{X}} \Pr[X = x]^2 \ .$$

3.7 Collisions on the Outputs of a Random Oracle

Mironov [Mir06] computed the probability of collision on the outputs of a random oracle R.

Lemma 3.24 (Collisions on R outputs).
Let $\mathcal{R}$ denotes a set of possible r_j values with cardinality q. We consider ℓ i.i.d. trials r_i with distribution $\mathcal{D}$. Let ε_c be the probability that at least one of the trials is in $\mathcal{R}$ or at least two of the trials are equal. We have

$$\varepsilon_c \leq 2^{-2 \cdot H_\infty(\mathcal{D})} \cdot \ell^2 \cdot q + 2^{-H_\infty(\mathcal{D})} \cdot \ell^2 \ . \tag{3.1}$$

Note that we can use another bound for ε_c in terms of Renyi entropy as described in Lemma 3.25 or as pseudo-randomness as described in Lemma 3.26.

Lemma 3.25 (Collisions on R outputs with respect to Renyi entropy).
Let $\mathcal{R}$ denotes a set of possible r_j values with cardinality q. We consider ℓ i.i.d. trials r_i with distribution $\mathcal{D}$. Let ε_c be the probability that at least one of the trials is in $\mathcal{R}$ or at least two of the trials are equal. We have

$$\varepsilon_c \leq \frac{\ell^2}{2} \cdot 2^{-H_2(\mathcal{D})} + \ell \cdot \sqrt{q} \cdot 2^{-\frac{H_2(\mathcal{D})}{2}} \ . \tag{3.2}$$

Proof.
Let $p_x = \Pr[r = x]$. We have

$$\begin{aligned} \varepsilon_c &= \Pr[\exists i, j : i \neq j, r_i = r_j \text{ or } r_i \in \mathcal{R}] \\ &\leq \frac{\ell^2}{2} \sum_x p_x^2 + \ell \sum_{x \in \mathcal{R}} p_x \leq \frac{\ell^2}{2} \sum_x p_x^2 + \ell\sqrt{q} \sqrt{\sum_x p_x^2} \ . \end{aligned}$$

■

Lemma 3.26 (Collisions on R outputs with respect to a PRG).
Let $\mathcal{R}$ denotes a set of possible r_j values with cardinality q. We consider ℓ i.i.d. trials r_i with distribution $\mathcal{D}$. Let ε_c be the probability that at least one of the trials is in $\mathcal{R}$ or at least two of the trials are equal. Assuming that $\mathcal{D}$ is (ℓ, ε)-PR in $\{0,1\}^\rho$, we have

$$\varepsilon_c \leq q \cdot 2^{-\rho} + \frac{\ell^2}{2} \cdot 2^{-\rho} + \varepsilon \ . \tag{3.3}$$

Part I

SAS-based Message Authentication and Key Agreement Protocols

Chapter

FOUR

Security Model

This chapter presents the security model for (message authentication) protocols relying on an extra authenticated channel. We first define the network setting (nodes, identities, and protocol instances) and the communication model (the available channels). Then, we define the adversary capabilities on the network, on the insecure channel, and finally on the extra authenticated channel. Finally, we present the concept of short authenticated string (SAS).

4.1 Network Model

We define a model for a communication network made up of devices and different communication channels between them. Here, the term "device" is a generic name to describe any communication entity. For instance, a device may be a personal computer, a mobile phone, a satellite, or a television.

Nodes and Unique Identities. We consider a network composed of N communication devices. Each device is located on a node n and each node is given a unique identity id_n. For instance, identities may be interpreted as network addresses. The communication device located on node n, of identity id_n, is denoted by $\mathcal{P}_{\mathsf{id}_n}$.

Key Database. Each node n locally maintains a database of $(\mathsf{sk}_j, \mathsf{id}_j)$ pairs. A pair means that it can use the symmetric key sk_j to communicate securely with the node of identity id_j.

A Protocol Run. A protocol π specifies a sequence of steps which consist of receiving a message and sending a response. The N participants of the network are potentially involved in the protocol execution. An internal short-term state σ keeps track on previously completed steps. Once the protocol is completed, σ is removed. A protocol starts with some specified inputs and an initial state (in terms of database content). It ends with some specified outputs (or an error message) and a final state. The difference between the inputs and outputs with respect to the initial and final states is that the adversary has control over the first ones but not on the states, except if the node was corrupted or some information leaked.

Concurrent Protocol Instances. A node n can run concurrent instances of the same or different protocols. Each instance of a protocol π is formally denoted by a unique instance tag $\pi_n^{(i)}$. Note that the (internal) state of a protocol related to a given tag changes with time as new steps of the protocol are made.

Group Descriptions. Let π be a protocol specifying the interaction between the participants $\mathcal{P}_{\mathsf{id}_1}, \ldots, \mathcal{P}_{\mathsf{id}_n}$. The participants form a group which may be small, e.g., two parties ($n = 2$), or much larger. We denote by $\mathcal{G} = \{\mathsf{id}_1, \ldots, \mathsf{id}_n\}$ the group of participants involved in the protocol. We always assume that the group $\mathcal{G}$ is ordered with respect to the sender identities $\mathsf{id}_1 < \mathsf{id}_2 < \cdots < \mathsf{id}_n$. We may use $\mathcal{H}$ to refer to the group of honest (non-corrupted) participants. Clearly, we always have $\mathcal{H} \subseteq \mathcal{G}$.

4.2 Communication Model

It is often prohibitively expensive to establish physical infrastructure that guarantees integrity of received messages. Authenticity concerns are particularly justified in case of wireless communication, since anybody with the proper equipment can eavesdrop, inject messages and cause communication failures. Thus, we have to assume that participants exchange messages over a communication network that is controlled by a malicious adversary.

However, the latter does not exclude the possibility of truly authentic message transmission, since participants may use alternative ways to communicate. For instance, in many small-range wireless networks a human operator can authentically transfer short messages from one device to another. If entities are further apart, we can transfer such messages over the phone provided that the participants can recognize each other by voice and behavior.

As usual, we consider a model where communication is asynchronous. Nodes can use *in-band* and *out-of-band* communication channels:

In-band communication. The in-band communication channel is totally insecure. It is routed via an active adversary $\mathcal{A}$ who can eavesdrop, delay, modify, drop, and insert messages. More details about the adversary capabilities are given in Section 4.3.

Out-of-band communication. Additionally, nodes are able to send *out-of-band* messages through an authenticated extra channel. More details on that channel are given in Section 4.4.

Figure 4.1 describes a network with three nodes, where each node can send insecure messages as well as authenticated messages to each other.

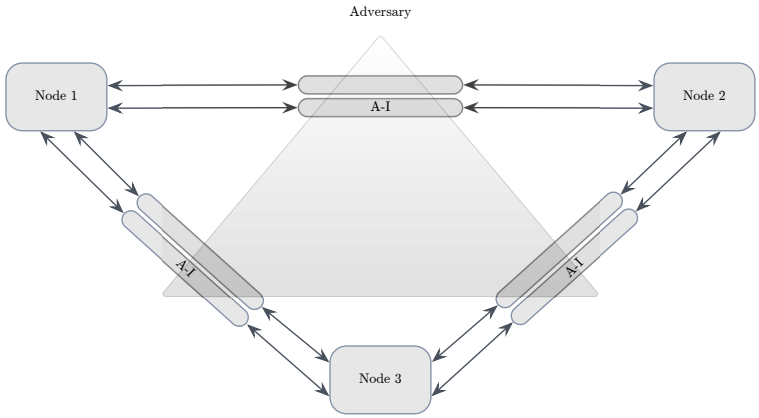


Figure 4.1. *Communication Channels (Example with Three Participants).*

We emphasize that there are no true broadcast channels in our model. Although several wireless networks such as WLAN in *ad-hoc* mode offer physical broadcast channels, there are no guarantees that the signal actually reaches all nodes. If we can guarantee this by physical means, then the authentication task becomes almost trivial. As different recipients can receive different broadcast messages, there is no difference between broadcasting and standard messaging except for efficiency. Similarly, broadcasting authenticated messages does not change the security analysis, although in practice, broadcasting can significantly reduce the necessary human interaction and make the protocol more user-friendly. For instance, a user entering the same PIN on each mobile device in a Bluetooth piconet is certainly less demanding than using different PIN values. The same is true if we consider secure VoIP-based conference calls: a participant giving the same value to all others has much less work than a participant giving a different value to each other.

Note that in some cases, a participant, say Alice, may only send messages without really interacting with any other party. As well, the other parties only receive messages from Alice. In that case, both communication channels are only used in one-way. Such a protocol using one-way channels is said *non-interactive*.

Definition 4.1 (Non-interactive Protocol).
A protocol is said to be non-interactive *if all protocol messages are sent from one node only, e.g., from Alice to the others.*

4.3 Adversarial Model

We adapt different adversarial models from Laur, Pasini and Vaudenay [Vau05b, Vau05a, PV06a, PV06b, LP08, LP09]. All these adversarial models are based on the one from Bellare-Rogaway [BR93a] which places the adversary at the center of the network. The adversary can make queries to any instances on any nodes. By default, the adversary is assumed to have full control over

- the insecure channel,
- the protocol inputs,
- which node launches a new protocol instance,
- which instance makes a new protocol step.

The adversary is also able

- to access the protocol outputs,
- to influence the delivery of messages (without modifying them) over the authenticated channels.

Occasionally, the adversary may

- violate the privacy of the internal state of a given node,
- corrupt a node so that its behavior with respect to future protocol runs is no longer guaranteed.

We assume that the actions of the participants, including potential adversaries, only depend on the received messages and their relative ordering. This assumption is often justified even if a practical instantiation of a protocol depends on explicit timings. In fact, it is straightforward to prove that security guarantees obtained in this simplified model are valid for all practical settings, where exact timings do not depend on the states of private variables.

More formally, the adversary has access to the following oracles:

Launch. The $\pi_n^{(i)} \leftarrow \mathsf{launch}(n, \mathsf{r}, x)$ oracle launches a new protocol instance on node n playing the role r with input x. The role r describes a character, i.e., a role to play in the protocol. It can be for instance Alice or Bob. This launch oracle returns a unique instance tag $\pi_n^{(i)}$. Since a node can run concurrent protocols, there may be several instances related to the same node n. Note that the instance inherits the current node state as input state.

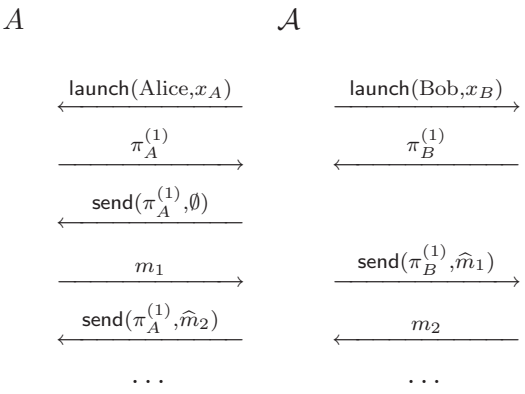
Execute. The $\xi \leftarrow \mathsf{execute}(\{j \in [1,n] : \pi_j^{(i_j)}\})$ oracle runs the full protocol with the given n protocol instances $\pi_1^{(i_1)}, \pi_2^{(i_2)}, \ldots \pi_n^{(i_n)}$ on nodes $1, 2, \ldots, n$ and returns the full transcript of protocol messages and the protocol outputs. This oracle models passive attacks.

Send. The $m' \leftarrow \mathsf{send}(\pi_n^{(i)}, m)$ oracle sends an incoming message m to the instance $\pi_n^{(i)}$. It returns either an outgoing message m' which is meant to be sent to another participant, or the the final output of the protocol if it completed. This models active attacks.

For example, assume a protocol π with two characters, Alice and Bob executed in presence of an adversary $\mathcal{A}$. Let the character Alice be played on node A with input x_A and the character Bob be played on node B with input x_B. A possible protocol execution is depicted in Figure 4.2 in two different ways: an adversary query list and a schematic message representation.

1. $\pi_A^{(1)} \leftarrow \mathsf{launch}(A, \text{Alice}, x_A)$
2. $\pi_B^{(1)} \leftarrow \mathsf{launch}(B, \text{Bob}, x_B)$
3. $m_1 \leftarrow \mathsf{send}(\pi_A^{(1)}, \emptyset)$
4. $m_2 \leftarrow \mathsf{send}(\pi_B^{(1)}, \widehat{m}_1)$
5. $m_3 \leftarrow \mathsf{send}(\pi_A^{(1)}, \widehat{m}_2)$
6. ...

$\equiv$



until a message is a termination message.

Figure 4.2. *Example of Oracle Queries for a Protocol Execution.*

Notation. By convention, we describe protocols by putting a *hat* ($\widehat{\cdot}$) on the notation for messages received by a node (i.e., inputs of the send oracle) which are not authenticated since they can differ from messages which were sent (i.e., outputs of the send oracle) in the case of an active attack. For instance, it is the case for m_1 and $\widehat{m}_1$ as well as for m_2 and $\widehat{m}_2$ in Figure 4.2.

We may give to the adversary also access to the oracle $m \leftarrow \mathsf{receive}(\pi_n^{(i)})$. However, this can be trivially emulated by a $m \leftarrow \mathsf{send}(\pi_n^{(i)}, \emptyset)$. The existence of the receive oracle makes sense only when we are using a non-interactive protocol. Indeed, a non-interactive protocol only uses one-way channels. So, it may seem strange to allow the adversary to ask for receiving a message with the $m \leftarrow \mathsf{send}(\pi_n^{(i)}, \emptyset)$ oracle, i.e., sending an empty message, because this may mean that the channels may be bidirectional.

Note that the Bellare-Rogaway [BR93a] model considers additional oracles specific to protocols using long-term secrets, like key agreements for instance:

Remove. The $\mathsf{remove}(n, \mathsf{id})$ oracle removes any $(\mathsf{sk}, \mathsf{id})$ entry in the database of node n. In practice, this oracle may be implemented by an adversary making denial-of-service attacks in the communication link between n and id so that n decides not to trust this connection anymore and to remove it.

Reveal. The $\mathsf{sk} \leftarrow \mathsf{reveal}(\pi_n^{(i)})$ oracle reveals the session key sk to the adversary $\mathcal{A}$ if the instance $\pi_n^{(i)}$ have accepted them before. This query models the loss of the session key and can be used to show the consequences on the other instances.

Corrupt. The $\mathsf{corrupt}(n)$ oracle corrupts the collection of instances related to the node n. So the behavior of any protocol instance at node n is no longer guaranteed. This query models the corruption of a node (all instances), for example a user-password pair has been stolen or a malicious code has been installed on the device on node n, e.g., with a "Trojan horse".

Test. The $b \leftarrow \mathsf{test}(n, \mathsf{sk}, \mathsf{id})$ tells whether $(\mathsf{sk}, \mathsf{id})$ is an entry of the database of node n ($b = 1$) or not ($b = 0$). In practice, this oracle may be implemented by an active adversary trying to impersonate node n to communicate with id. If the attempt succeeds, it means that sk was correct.

Definition 4.2 (Attack Cost.).
The attack cost *is measured by*

- *q, the number of launched instances of the different roles, i.e., the* online complexity*,*
- *T, the additional time complexity, i.e., the* offline complexity*,*
- *p, the probability of success.*

4.4 Authenticated Channel Models

When referring to "channel", we refer by default to an insecure broadband channel with no additional assumption. As mentioned before, the devices of the network can use extra authenticated channels.

An *authenticated channel* is related to a node identity id. Formally, an authenticated channel from a node n has an identifier id_n. It allows the recipient of a message to know the identity of the node from which the message has been sent. Note that an adversary cannot modify it (i.e., integrity is implicitly protected), but she can delay it, remove it, replay it, and of course read it. In particular, an authenticated channel does not provide confidentiality.

By convention, we note $\mathsf{auth}_{\mathsf{id}_n}(m)$ a message m which has been sent from node n through the authenticated channel.

The send oracle maintains unordered sets of authenticated messages in every channel id_n from node n. Only send oracles with a $\pi_n^{(i)}$ instance can insert a new message in this set. When a send oracle is queried with any instance and any message $\mathsf{auth}_{\mathsf{id}_n}(m)$, it is accepted by the oracle only if m is in the set related to channel id_n. Note that concurrent or successive instances related to the same node write in the same channel, i.e., in the same set. Thus, when an instance $\pi_n^{(i)}$ sends a message, the recipient of this message can only authenticate the node from which it has been sent, i.e., n, but not the connection to the right instance, i.e., i.

For simplicity, we assume that the input or output to the send oracle are either authenticated or non-authenticated messages, but not both. Namely, protocols do not concatenate authenticated and non-authenticated messages.

4.4.1 Weak Authenticated Channels

By default, authenticated channels with no other assumption than authentication and integrity are called *weak*. This means that an adversary can delay, remove, or replay a message. In particular, the sender of the message has no assurance on the message delivery.

4.4.2 Stronger Authenticated Channels

In some cases we need special assumptions on the authenticated channel. Thus, we can consider *stronger authenticated channels*, namely channels in which additional properties are achieved. In the following, we propose some possible properties that can be assumed on a stronger authentication channel.

Stall-free transmission assumes that when a message is released by a send oracle, either it is used as input in the just following send oracle query (either authenticated or not) or it is never used.

Transmission with acknowledgment assumes that messages are released with a destination node identifier and the sender can check whether an instance at the destination node has received the message or not.

Listener-ready transmission assumes that the sender can check if an instance at the destination node is currently ready to listen to the authenticated channel.

Transmission with immediate delivery assumes that an input message of a send oracle is immediately delivered to the recipient.

4.4.3 Examples

As mentionned in Section 2.2, human beings can use different channels to communicate. However, not all achieve authentication.

A **face-to-face conversation** (encounter) and a **telephone call** ensure authenticity. In addition to this, these channels achieve some of the above stronger properties. Suppose two persons want to start communicating. When the first person starts talking, he knows that the second one is listening (listener-ready). When one talks to the other one, he knows that the message is not a replay of a previous conversation since interactivity implies coherent conversations (stall-free). Humans can also sense if the other one has listened to the message (acknowledgment). Finally, in face to face conversation, spoken words will be immediately heard by the other (immediate delivery). However, by telephone, this is not the case. Indeed, there may be delays, crosstalks, and concurrent talks (collisions of voices).

A (postal) **mail** can be stalled and released in a different order. The sender has no confirmation in general that the mail has been received (except using a registered mail). Finally, the recipient may not be ready to receive it. Thus, a mail achieves none of the strong properties. A registered mail (reg. mail) only adds an acknowledgment.

A **voice mail** (or voice record) achieves none of the stronger properties since the message could be a recorded one, the recorder has no confirmation that the destination heard it, and the recipient is in general not ready to listen.

An **electronic mail** (email) is the worst channel in term of security since it has none of these properties. In particular,note that an email with no cryptographic appendix, such as a GPG signature for instance, is in fact *not* an authenticated channel since it can easily be forged.

It is clear that mail, electronic mail and voice record are not delivered immediately.

	Interactive		Non-interactive			
	Encounter	Telephone	Reg. mail	Mail	Voice mail	Email
Authentication	☺	☺	☺	☺	☺	
Stall-free	☺	☺				
Acknowledgment	☺	☺	☺			
Listener-ready	☺	☺				
Immediate delivery	☺					
Strong / Weak	Strong	Strong	Strong	Weak	Weak	-

Figure 4.3. *Stronger Properties on the Extra Channels used by Human Beings.*

There also exist other channels that enable the transfer of a string from one device to another in an authenticated way. The user still plays an important role. He has in possession

the devices, no adversary can access it physically, and so authentication is ensured.

Indeed, a user can **copy a string** from one device to the other. No adversary is able to alter the transferred string, so authentication and integrity is ensured. This channel is also stall-free since no adversary is able to avoid the delivery of the message and no adversary is able to replay an old message. The user is invited to enter the message in the second device, so this kind of channel is listener-ready. It does not achieve immediate delivery since it is not possible to force the user to type the message. Also, by default, the source has no acknowledgement, but if required we can force the user to acknowledge.

The user load can be limited in the case that both devices display the message to be authenticated. In that case, the user only needs to compare them and accept, resp. reject, if they are equal, resp. different. We call such a channel **string comparison** and it ensures basically the same properties as the string copy channel.

	String copy	String comparison
Authentication	☺	☺
Stall-free	☺	☺
Acknowledgment		
Listener-ready	☺	☺
Immediate delivery		
Strong / Weak	Strong	Strong

Figure 4.4. *Stronger Properties on User-aided Extra Channels.*

4.4.4 SAS-based Cryptography

Vaudenay [Vau05b] introduced the notion of *Short Authenticated String* (SAS). Usually, there is a trade-off between the security and the amount of authenticated communication. For many practical applications, the SAS message consists of 6 decimal digits and thus an adversary can succeed a trivial deception attack with probability 10^{-6}. On the other hand, 10^{-6} is also the probability of not noticing an active attack. The latter is small enough to demotivate most of the possible attackers. Of course, the cryptographic security levels can be achieved only with sufficiently long SAS messages. Therefore, it is important to minimize (or optimize) the amount of manually authenticated communication. So, as mentioned before, the extra authenticated channel is modeled as an expensive channel and consequently the authenticated messages should be as short as possible.

Chapter

FIVE

On the Optimal Entropy of Authenticated Communication

In this chapter, we would like to upper bound the security of an arbitrary SAS-based message authentication protocol π given the overall length of authenticated strings it uses. In other words, if we fix the number of authenticated bits to k, then the question is what is the strongest achievable security? To answer that question, in this chapter we propose *generic* attacks against such kinds of protocols. Note that we focus in two-party protocols. Since there exists no protocol resistant to these generic attacks we conclude that it is the strongest achievable security. Any protocol reaching this security level would be *optimal*.

Generic Unilateral Message Authentication Protocol. Assume that the protocol is used to authenticate a message m from Alice to Bob. For that reason, we assume that authenticated messages are only sent by Alice. We consider the more general case by supposing that the protocol can use any sequence of authenticated messages in a given set $\mathcal{S}$ during the protocol. We call it a *transcript*. Note that authenticated strings may be interleaved with regular messages which are not represented in the transcript since they may be easily forged by an adversary. For any input message m, the authenticated transcript used during a protocol run is denoted by SAS_m and it is picked with a distribution $\mathcal{D}_m$ in the set $\mathcal{S}$ of all possible transcripts.

One-shot Adversaries. First, we analyze the security against adversaries which can only use one instance of Alice and one instance of Bob. We call them *one-shot* adversaries.

Multi-shot Adversaries. Second, we consider adversaries which can launch many instances of Alice and Bob. We call them *multi-shot* adversaries. Over a weak authenticated channel, adversaries can delay or replay authenticated messages. With protocols using a k-bit SAS, we may have the following attacks.

Delay attack. An attacker starts a protocol with Alice to recover one authenticated string. Then, the attacker launches several (online) protocols with Bob until the expected authenticated string by Bob matches the one recovered before from Alice. The adversary delivers the authenticated string of Alice to this instance of Bob.

Catalog attack. Similarly, an adversary launches several instances of Alice, recovers many authenticated strings, and builds a catalog of authenticated strings. Then, the adversary starts a protocol with Bob and (if possible) uses one of the recovered authenticated string from Alice. This attack works when the SAS catalog is close to the complete one.

Trade-off attack. We can further trade the number of Bob's instances against the number of Alice's instances and have a birthday paradox effect.

Note that the first two attacks work within a number of trials around 2^k but the third one needs only around $2^{k/2}$ trials.

5.1 Probability of Collision Between Random Variables

In any attack, i.e., one-shot, delay, catalog, or trade-off attack, we need to study the probability that two authenticated strings collide.

We will see later that the one-shot attack is successful if the two authenticated transcripts $\mathsf{SAS}_{\mathsf{Alice}}$ and $\mathsf{SAS}_{\mathsf{Bob}}$ match. By considering the SAS values as independent and identically distributed (i.i.d.) random variables, it is interesting to know the probability of collision between them. In order to build secure protocols, we also need to know when this probability of collision is minimal.

Lemma 5.1 (Collision Between Two Independent Random Variables). *Let X and Y be two independent and identically distributed random variables with distribution $\mathcal{D}$ over a support set $\mathcal{S}$ of n elements, i.e., $|\mathcal{S}| = n$. We have*

$$\Pr[X = Y] = 2^{H_2(\mathcal{D})} \geq \frac{1}{n} \qquad \textit{and} \qquad \Pr[X = Y | \mathcal{D} \textit{ is uniform}] = \frac{1}{n} \ . \tag{5.1}$$

Thanks to Lemma 5.1, we know the probability of collision between two random variables and we know that this probability is minimal when the distribution is uniform.

Proof.
Let n be the size of the set $\mathcal{S}$. Since X and Y are i.i.d. we have

$$\Pr[X = Y] = \sum_{s_i \in \mathcal{S}} \Pr[X = s_i] \cdot \Pr[Y = s_i] = \sum_{s_i \in \mathcal{S}} p_i^2$$

where p_i denotes $\Pr[X = s_i]$. Note that the above line leads to $\Pr[X = Y] = 2^{H_2(\mathcal{D})}$ where $H_2(\cdot)$ denotes the Renyi entropy as in Definition 3.23.
Now, let us write $p_i = \frac{1}{n} + \rho_i$, we obtain

$$\Pr[X = Y] = \sum_{s_i \in \mathcal{S}} p_i^2 = \sum_{s_i \in \mathcal{S}} \left(\frac{1}{n} + \rho_i\right)^2 = \sum_{s_i \in \mathcal{S}} \left(\frac{1}{n}\right)^2 + 2 \sum_{s_i \in \mathcal{S}} \frac{1}{n} \rho_i + \sum_{s_i \in \mathcal{S}} \rho_i^2 \ .$$

Note that $\sum_{s_i \in \mathcal{S}} p_i = \sum_{s_i \in \mathcal{S}} \frac{1}{n} + \sum_{s_i \in \mathcal{S}} \rho_i = 1$ and so we deduce that $\sum_{s_i \in \mathcal{S}} \rho_i = 0$. We finally obtain

$$\Pr[X = Y] = \frac{1}{n} + \sum_{s_i \in \mathcal{S}} \rho_i^2 \ .$$

Clearly, $\sum_{s_i \in \mathcal{S}} \rho_i^2$ is positive. $\sum_{s_i \in \mathcal{S}} \rho_i^2 = 0$ when all ρ_i are null, i.e., when the distribution $\mathcal{D}$ is uniform, and $\sum_{s_i \in \mathcal{S}} \rho_i^2 > 0$ when the distribution $\mathcal{D}$ is non-uniform.

■

Later, we will also see that a multi-shot attack tries to find a collision between two sets of SAS values. In order to build secure protocols, we also need to know when this probability of collision between these sets is minimal.

Lemma 5.2 (SAS Values Should Belong to the Uniform Distribution).
We consider two sets of independent random values $\{X_i\}$, resp. $\{Y_j\}$, of size p, resp. q, where the elements are picked in a set $\mathcal{S}$ of size n with distribution $\mathcal{D}$.

The probability of collision between the two sets is minimal when the distribution $\mathcal{D}$ is uniform.

Assuming that the attacks really look for collision between random SAS values, any authentication protocol must use uniformly distributed SAS values in order to minimize the attacks. Indeed, any other distribution will allow the adversary to find collisions with a higher probability or a smaller complexity. So, the protocol will be less secure.

Proof.
Let $C_{\mathcal{D}}^{p,q}$ be the probability that there exists a X_i which corresponds to a Y_j given a distribution $\mathcal{D}$ along the set $\mathcal{S}$ of n elements, i.e., a collision occurred between the two sets:

$$C_{\mathcal{D}}^{p,q} = \Pr\left[\{X_1, \cdots, X_p\} \cap \{Y_1, \cdots, Y_q\} \neq \emptyset\right] \ .$$

Formally, we have to prove that

$$C_{\mathcal{D}}^{p,q} \geq C_{U_n}^{p,q} \tag{5.2}$$

where U_n is the uniform distribution among a set of n possible elements.

We will proceed by first proving that Equation (5.2) is true for a support $\mathcal{S}$ of one element, i.e., when $n = 1$. Then, we will prove by recurrence that Equation (5.2) is true for any n, i.e., by proving that it is true for n elements assuming that it is true for $n - 1$ elements.

For a set $\mathcal{S}$ of one element, i.e., $n = 1$, the result is straightforward since the only possible distribution is the uniform distribution and we have $C_{\mathcal{D}}^{p,q} = C_{U_1}^{p,q}$.

For a set $\mathcal{S}$ of any size n, we will prove that it is also true by assuming that Equation (5.2) is true for $n - 1$ or less elements.

Let a be an elements of $\mathcal{S}$ and p_a its probability, i.e., $p_a = \Pr_{\mathcal{D}}[X = a]$. We define a new distribution $\mathcal{D}'$ by considering $\mathcal{D}$ in which a never occurs, i.e.,

$$\Pr_{\mathcal{D}'}[X = x] = \begin{cases} 0 & \text{if } x = a \\ \Pr_{\mathcal{D}}[X = x | x \neq a] & \text{if } x \neq a \end{cases} .$$

Considering the particular element a, we apply the total probability theorem, and we can write

$$\begin{aligned} C_{\mathcal{D}}^{p,q} \;=\; & \Pr\left[\{X_i\} \cap \{Y_j\} \neq \emptyset \middle| a \notin \{X_i\}, a \notin \{Y_j\}\right] \\ & \cdot \Pr[a \notin \{X_i\}, a \notin \{Y_j\}] \\ & + \Pr\left[\{X_i\} \cap \{Y_j\} \neq \emptyset \middle| a \in \{X_i\}, a \in \{Y_j\}\right] \\ & \cdot \Pr[a \in \{X_i\}, a \in \{Y_j\}] \\ & + \Pr\left[\{X_i\} \cap \{Y_j\} \neq \emptyset \middle| a \in \{X_i\}, a \notin \{Y_j\}\right] \\ & \cdot \Pr[a \in \{X_i\}, a \notin \{Y_j\}] \\ & + \Pr\left[\{Y_i\} \cap \{Y_j\} \neq \emptyset \middle| a \notin \{X_i\}, a \in \{Y_j\}\right] \\ & \cdot \Pr[a \notin \{X_i\}, a \in \{Y_j\}] \end{aligned}$$

which can be written as

$$\begin{aligned} C_{\mathcal{D}}^{p,q} \;=\; & C_{\mathcal{D}'}^{p,q} \cdot (1 - p_a)^p (1 - p_a)^q \\ & + 1 \cdot [1 - (1 - p_a)^p][1 - (1 - p_a)^q] \\ & + \sum_{i=1}^{p} \left\{ C_{\mathcal{D}'}^{p-i,q} \binom{p}{i} p_a^i (1 - p_a)^{p-i} \right\} \\ & \cdot [1 - (1 - p_a)^p](1 - p_a)^q \\ & + \sum_{i=1}^{q} \left\{ C_{\mathcal{D}'}^{p,q-i} \binom{q}{i} p_a^i (1 - p_a)^{q-i} \right\} \\ & \cdot (1 - p_a)^p [1 - (1 - p_a)^q] \ . \end{aligned}$$

By noting that $\mathcal{D}'$ is distributed over $n-1$ elements only (since a never occurs) and assuming that Equation 5.2 is true for $n-1$ elements, we can write

$$C^{p,q}_{\mathcal{D}'} \geq C^{p,q}_{U_{n-1}} \ .$$

So, we obtain

$$\begin{aligned} C^{p,q}_{\mathcal{D}} &\geq C^{p,q}_{U_{n-1}} \cdot (1-p_a)^p(1-p_a)^q \\ &\quad +1 \cdot [1-(1-p_a)^p][1-(1-p_a)^q] \\ &\quad + \sum_{i=1}^{p} \left\{ C^{p-i,q}_{U_{n-1}} \binom{p}{i} p_a^i (1-p_a)^{p-i} \right\} \\ &\qquad \cdot [1-(1-p_a)^p](1-p_a)^q \\ &\quad + \sum_{i=1}^{q} \left\{ C^{p,q-i}_{U_{n-1}} \binom{q}{i} p_a^i (1-p_a)^{q-i} \right\} \\ &\qquad \cdot (1-p_a)^p[1-(1-p_a)^q] \\ &\triangleq C^{p,q}_{\mathcal{D}_0} \ . \end{aligned}$$

The right hand side of this inequality corresponds to the probability of collisions $C^{p,q}_{\mathcal{D}_0}$ where the distribution $\mathcal{D}_0$ is defined as

$$\Pr_{\mathcal{D}_0}[X=x] = \begin{cases} p_a & \text{if } x = a \\ \frac{1}{n-1}(1-p_a) & \text{if } x \neq a \end{cases} \ .$$

Consequently, we obtain $C^{p,q}_{\mathcal{D}} \geq C^{p,q}_{\mathcal{D}_0}$.

We repeat this step using the same reasoning but using another element. Let $b \neq a$ be an element of the set $\mathcal{S}$ and let p_b its probability over $\mathcal{D}_0$, i.e., $p_b = \Pr_{\mathcal{D}_0}[X=b] = \frac{1}{n-1}(1-p_a)$. Proceeding as before, we obtain $C^{p,q}_{\mathcal{D}_0} \geq C^{p,q}_{\mathcal{D}_1}$ where $\mathcal{D}_1$ is defined as

$$\Pr_{\mathcal{D}_1}[X=x] = \begin{cases} p_b & \text{if } x = b \\ \frac{1}{n-1}(1-p_b) & \text{if } x \neq b \end{cases} \ .$$

Finally, we obtain the following recurrence

$$C^{p,q}_{\mathcal{D}} \geq C^{p,q}_{\mathcal{D}_0} \geq C^{p,q}_{\mathcal{D}_1} \geq \cdots \geq C^{p,q}_{\mathcal{D}_i} \geq \cdots$$

where the distributions are defined as

$$\Pr_{\mathcal{D}_i}[X=x] = \begin{cases} p_i & \text{if } x = a_i \\ \frac{1}{n-1}(1-p_i) & \text{if } x \neq a_i \end{cases}$$

with

$$a_i = \begin{cases} a & \text{if } i \text{ odd} \\ b & \text{if } i \text{ even} \end{cases} \ , \ p_0 = \Pr_{\mathcal{D}}[X=a_i], \ \forall i \geq 1 : p_i = \Pr_{\mathcal{D}_{i-1}}[X=a_i] \ .$$

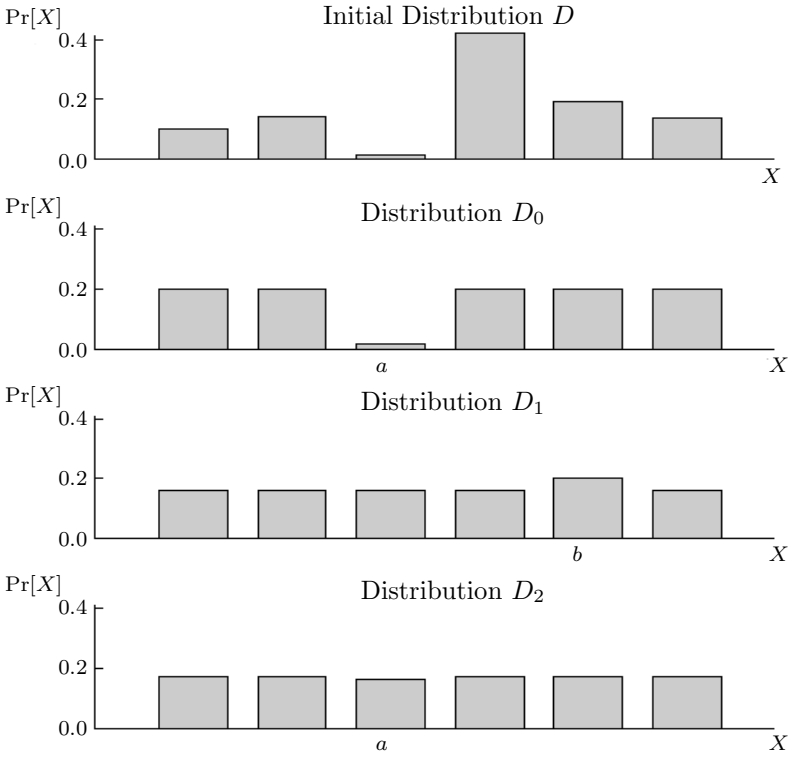


Figure 5.1. *Three First Steps of the Recurrence ($\mathcal{D}$ and $\mathcal{D}_0, \mathcal{D}_1, \mathcal{D}_2$).*

Figure 5.1 represents on its top the original distribution $\mathcal{D}$ and below the three first steps of the recurrence, i.e., $\mathcal{D}_0, \mathcal{D}_1, \mathcal{D}_2$.

The recurrence converges rapidly towards the uniform distribution. Indeed, we have $p_{i+1} = \frac{1}{n-1}(1 - p_i)$ and thus

$$p_i = \frac{1}{n} + (-\frac{1}{n-1})^i \cdot (p_0 - \frac{1}{n})$$

which converges to $\frac{1}{n}$ when $n \to \infty$. We can finally write

$$C_{\mathcal{D}}^{p,q} \geq C_{\mathcal{D}_0}^{p,q} \geq C_{\mathcal{D}_1}^{p,q} \geq \cdots \geq C_{\mathcal{D}_t}^{p,q} \underset{t\to\infty}{\longrightarrow} C_{U_n}^{p,q}.$$

We conclude that any probability of collision $C_{\mathcal{D}}^{p,q}$, where $\mathcal{D}$ is any distribution along $\mathcal{S}$, is bigger or equal to the probability of collision $C_{U_n}^{p,q}$ which uses uniformly distributed random variables.

■

5.2 A Generic One-Shot Attack

Theorem 5.3 (Generic One-Shot Attack).
We consider an arbitrary two-party SAS-based message authentication protocol π using an authenticated channel. π is run between Alice and Bob. Let t be the maximal bit-length of the input messages. Let $\mathcal{S}$ be the set of all possible SAS transcripts for any input message and let n be its cardinality.

There exists a generic polynomial-time attack using only one protocol instance with Alice and another one with Bob with probability of success at least $\frac{1}{n} - 2^{-t}$. The attack running time is $\mu_A + \mu_B + \mathcal{O}(1)$ where μ_A, resp. μ_B, is the complexity of a simulator of Alice, resp. Bob.

The probability of success may reach the bound if and only if the SAS distribution is uniform among the set $\mathcal{S}$.

Theorem 5.3 says that there exists a one-shot attack against any SAS-based message authentication protocol which succeeds with probability at least essentially $\frac{1}{n}$ where n is the size of $\mathcal{S}$. Thus, any SAS-based message authentication protocol which has no better one-shot attack than the proposed one is essentially *optimal*. Indeed, any other protocol can be attacked using this generic attack and consequently cannot have a better security.

In addition, Theorem 5.3 says that a protocol is more resistant to one-shot attacks when its SAS distribution is uniform.

Proof.
We consider a general man-in-the-middle attack in which the adversary first picks $m, \widehat{m} \in_{\mathsf{u}} \{0,1\}^t$ and launches Alice with input m. The attack runs synchronized protocols between Alice and a simulator for Bob, and a simulator for Alice with input $\widehat{m}$ and Bob as depicted in Figure 5.2. Following the attack, every authenticated message which must be sent by

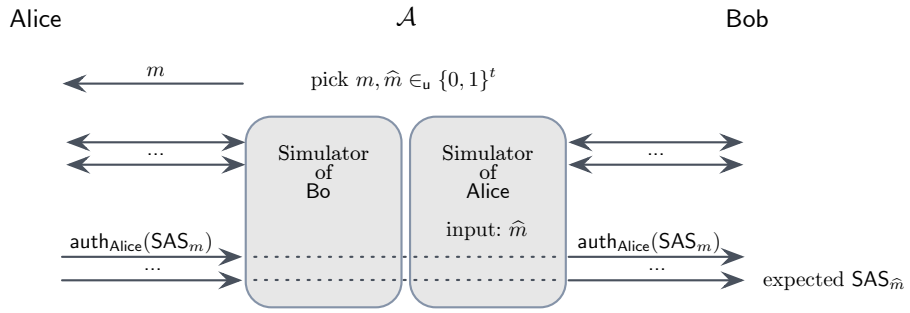


Figure 5.2. *Generic One-Shot Attack.*

the simulator is replaced by an authenticated message which has just been received by the simulator.

Let SAS_m be the (random) sequence of all authenticated strings (the transcript) which would be exchanged in the protocol between Alice and the simulator for Bob if the simulator were honest. Let $\mathsf{SAS}_{\widehat{m}}$ be the similar sequence between the simulator for Alice and Bob. Clearly, if $\mathsf{SAS}_{\widehat{m}} = \mathsf{SAS}_m$, the attack succeeds. Note that an attack makes sense only if $\widehat{m}$ is different from m. Clearly, the probability of success is

$$\Pr[\text{success}] = \Pr[\mathsf{SAS}_m = \mathsf{SAS}_{\widehat{m}} \text{ and } m \neq \widehat{m}] \ .$$

Note that $\Pr[\mathsf{SAS}_m = \mathsf{SAS}_{\widehat{m}}] = \Pr[\mathsf{SAS}_m = \mathsf{SAS}_{\widehat{m}} \text{ and } m \neq \widehat{m}] + \Pr[\mathsf{SAS}_m = \mathsf{SAS}_{\widehat{m}} \text{ and } m = \widehat{m}]$ and so,

$$\Pr[\text{success}] = \Pr[\mathsf{SAS}_m = \mathsf{SAS}_{\widehat{m}}] - \Pr[\mathsf{SAS}_m = \mathsf{SAS}_{\widehat{m}} \text{ and } m = \widehat{m}] \ .$$

From the simple rule $\Pr[A \text{ and } B] \leq \Pr[B]$ we can write

$$\Pr[\text{success}] \geq \Pr[\mathsf{SAS}_m = \mathsf{SAS}_{\widehat{m}}] - \Pr[m = \widehat{m}] \ .$$

SAS_m and $\mathsf{SAS}_{\widehat{m}}$ are two identically distributed independent random variables whose support is included in $\mathcal{S}$. Due to Lemma 5.1 we can write

$$\Pr[\mathsf{SAS}_m = \mathsf{SAS}_{\widehat{m}} | m \neq \widehat{m}] \geq \frac{1}{n}$$

where the equality occurs if and only if the SAS distribution is uniform (also in Lemma 5.1). Since m and $\widehat{m}$ are uniformly distributed in $\{0,1\}^t$, we have $\Pr[m = \widehat{m}] = 2^{-t}$.

So, we finally obtain

$$\Pr[\text{success}] \geq \frac{1}{n} - 2^{-t}$$

where the equality holds if and only if the SAS distribution is uniform among the set $\mathcal{S}$.

■

5.3 A Generic Multi-Shot Attack

In this section, we sketch a generic attack bounded by q_A, resp. q_B, instances of Alice, resp. Bob, which works against any SAS-based message authentication protocol.

Theorem 5.4 (Generic Multi-Shot Attack).
We consider an arbitrary two-party SAS-based message authentication protocol π using a weak authenticated channel. π is run between Alice and Bob. Let t be the maximal bit-length of the input messages. Let $\mathcal{S}$ be the set of all possible SAS transcripts for any input message and let n be its cardinality.

There exists a generic polynomial-time attack which uses $q_A \leq n$ instances of Alice and $q_B \leq n$ instances of Bob with probability of success at least $1 - e^{-\frac{q_A q_B}{n}} - q_A q_b 2^{-t}$. The attack running time is $q_A \mu_B + q_B \mu_A + \mathcal{O}(1)$ where μ_A, resp. μ_B, is the complexity of a simulator of Alice, resp. Bob.

Proof.
We consider that the adversary can use q_A instances of Alice and q_B instances of Bob. As before, we consider a general man-in-the-middle attack. The adversary first picks $m_i \in_{\mathsf{u}} \{0,1\}^t$ with $i = 1, .., q_A$ and $\widehat{m}_j \in_{\mathsf{u}} \{0,1\}^t$ with $j = 1, .., q_B$ and launches several instances of Alice with input m_i. For each instance of Alice, the attack runs a synchronized protocol with a simulator for Bob. Then, it launches the instances of Bob and for each one it runs a simulator for Alice with input $\widehat{m}_j$. Every authenticated message which must be sent by a simulator of Alice is replaced by an authenticated message which has just been received by a simulator of Bob.

Let SAS_{m_i} be the (random) sequence of all authenticated strings (the transcript) which would be exchanged in the i^{th} instance of the protocol between Alice and the simulator of Bob if the simulator were honest, and $\mathsf{SAS}_{\widehat{m}_j}$ be the similar sequence between the j^{th} instance of the simulator of Alice and Bob. Let $\mathcal{S}$ the set of all possible protocol authenticated transcripts and n its cardinality. Note that all SAS_{m_i} and all $\mathsf{SAS}_{\widehat{m}_j}$ are independent and identically distributed among the set $\mathcal{S}$ and let $\mathcal{D}$ be their distribution.

Clearly, if there exists a couple (i, j) such that $\mathsf{SAS}_{\widehat{m}_j} = \mathsf{SAS}_{m_i}$, the attack succeeds. Note that an attack makes sense only if $\widehat{m}_j$ is different from m_i. Consequently, the probability of

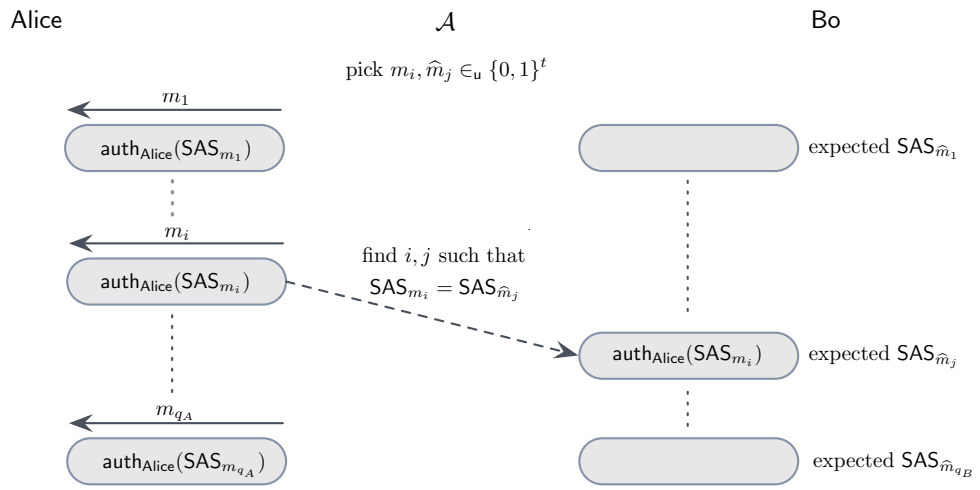


Figure 5.3. *Generic Multi-Shot Attack.*

success can be written as

$$\Pr[\text{success}] \;=\; \Pr[\exists\, i,j \text{ s.t. } (\mathsf{SAS}_{m_i} = \mathsf{SAS}_{\widehat{m}_j} \text{ and } m_i \neq \widehat{m}_j)] \;.$$

As in the previous proof, note that

$$\begin{aligned}\Pr[\exists\, i,j \text{ s.t. } \mathsf{SAS}_{m_i} = \mathsf{SAS}_{\widehat{m}_j}] = \\ \Pr[\exists\, i,j \text{ s.t. } \mathsf{SAS}_{m_i} = \mathsf{SAS}_{\widehat{m}_j} \text{ and } m_i \neq \widehat{m}_j] \\ + \Pr[\exists\, i,j \text{ s.t. } \mathsf{SAS}_{m_i} = \mathsf{SAS}_{\widehat{m}_j} \text{ and } \forall k,\ell \text{ s.t. } \mathsf{SAS}_{m_k} = \mathsf{SAS}_{\widehat{m}_\ell} : m_k = \widehat{m}_\ell]\end{aligned}$$

In the last equation, the first term of the right hand side is the probability of success. So,

$$\begin{aligned}\Pr[\text{success}] = &\Pr[\exists\, i,j \text{ s.t. } \mathsf{SAS}_{m_i} = \mathsf{SAS}_{\widehat{m}_j}] \\ &- \Pr[\exists\, i,j \text{ s.t. } \mathsf{SAS}_{m_i} = \mathsf{SAS}_{\widehat{m}_j} \text{ and } \forall k,\ell \text{ s.t. } \mathsf{SAS}_{m_k} = \mathsf{SAS}_{\widehat{m}_\ell} : m_k = \widehat{m}_\ell]\end{aligned}$$

Due to the rule $\Pr[A \text{ and } B] \leq \Pr[B]$, we obtain

$$\begin{aligned}\Pr[\text{success}] \;\geq\; &\Pr[\exists\, i,j \text{ s.t. } \mathsf{SAS}_{m_i} = \mathsf{SAS}_{\widehat{m}_j}] \\ &- \Pr[\forall k,\ell \text{ s.t. } \mathsf{SAS}_{m_k} = \mathsf{SAS}_{\widehat{m}_\ell} : m_k = \widehat{m}_\ell] \qquad (5.3)\end{aligned}$$

First, we should lower bound the first term of Equation (5.3), i.e., $\Pr[\exists\, i,j \text{ s.t. } \mathsf{SAS}_{m_i} = \mathsf{SAS}_{\widehat{m}_j}]$. Due to Lemma 5.2, we can bound this probability by the one using a uniform distribution, i.e.,

$$\Pr[\exists\, i,j \text{ s.t. } \mathsf{SAS}_{m_i} = \mathsf{SAS}_{\widehat{m}_j}] \geq \Pr[\exists\, i,j \text{ s.t. } \mathsf{SAS}_{m_i} = \mathsf{SAS}_{\widehat{m}_j} | \mathcal{D} \text{ is uniform}].$$

Consequently, we can bound the probability of collision by using the birthday paradox. Indeed, we are looking for a collision between a set of q_A elements and another set of q_B elements where each element was picked uniformly in a set of size n. A few details on the birthday paradox are given in Appendix A. So, we obtain

$$\Pr[\exists\, i, j \text{ s.t. } \mathsf{SAS}_{m_i} = \mathsf{SAS}_{\widehat{m}_j}] \geq 1 - e^{-\frac{q_A q_B}{n}} \ .$$

We should now upper bound the second term of Equation (5.3), i.e., $\Pr[\forall k, \ell \text{ s.t. } \mathsf{SAS}_{m_k} = \mathsf{SAS}_{\widehat{m}_\ell} : m_k = \widehat{m}_\ell]$. This is the probability that for each SAS collision, there is also a collision on the input messages. Clearly, this probability is smaller than the probability that there is only one collision on the input messages. So, we can write

$$\begin{aligned} \Pr[\forall k, \ell \text{ s.t. } \mathsf{SAS}_{m_k} = \mathsf{SAS}_{\widehat{m}_\ell} : m_k = \widehat{m}_\ell] &\leq \Pr[\exists k, \ell \text{ s.t. } m_k = \widehat{m}_\ell] \\ &\leq q_A q_B 2^{-t} \ . \end{aligned}$$

■

5.4 A Generic Multi-Shot Attack Against Non-Interactive Protocols

We can also provide a generic attack against *any* SAS-based non-interactive message authentication protocol (NIMAP). Indeed, with non-interactive protocols an adversary can run the catalog attack. Indeed, he may launch several instances of Alice and recover many authenticated SAS. Then, he simulates many instances of the protocol and finally only launches one instance of Bob and uses a SAS from the catalog.

Theorem 5.5 (Generic Multi-Shot Attack Against Non-Interactive Protocols). *We consider an arbitrary two-party SAS-based non-interactive message authentication protocol (NIMAP) π using a weak authenticated channel. π is run between Alice and Bob. Let t be the maximal bit-length of the input messages. Let $\mathcal{S}$ be the set of all possible SAS transcripts for any input message and let n be its cardinality.*

There exists a generic polynomial-time attack which uses q_A instances of Alice, one instance of Bob, and $\mathcal{O}(T)$ protocol simulations, with probability of success at least $1 - e^{-\frac{q_A \cdot T}{n}} - q_A T 2^{-t}$. The attack running time is $q_A \mu_B + T(\mu_A + \mu_B) + \mathcal{O}(1)$ where μ_A, resp. μ_B, is the complexity of a simulator of Alice, resp. Bob.

Proof.
This attack is similar to the one of Theorem 5.4 except that the instances of Bob can be simulated. Indeed, Bob only receives messages and then takes a deterministic decision, accept or reject. Since Bob does not provide any contribution in the messages, its behavior can be totally simulated.

We consider the generic attack in which the adversary starts by simulating T Alice instances launched with random inputs $\widehat{m}_j \in_{\mathsf{u}} \{0,1\}^t$ and obtains a list of possible SAS, i.e., $\mathsf{SAS}_{\widehat{m}_j}$. Then, he launches q_A real instances of Alice with random inputs $m_i \in_{\mathsf{u}} \{0,1\}^t$ and consequently obtains q_A authenticated SAS, i.e., SAS_{m_i}. The attack succeeds when at least one authenticated SAS released by Alice corresponds to a computed one, i.e., there exists i, j such that $\mathsf{SAS}_{m_i} = \mathsf{SAS}_{\widehat{m}_j}$. The adversary can launch a single Bob with input $\widehat{m}_j$ by simulating Alice and can use SAS_{m_i} for the authentication when needed.

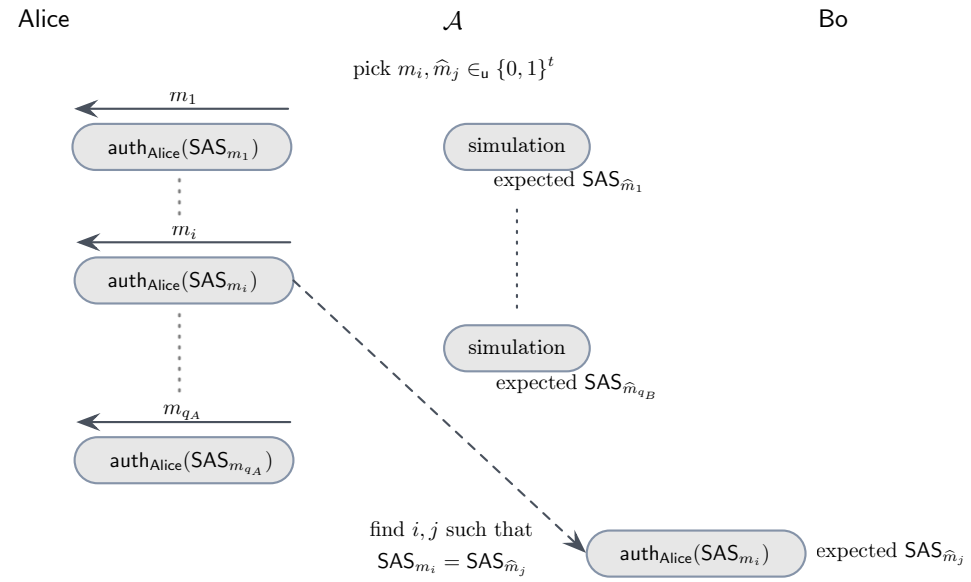


Figure 5.4. *Generic Multi-Shot Attack against Non-Interactive Protocols.*

From the proof of Theorem 5.4 we deduce that the probability of success is at least $1 - e^{-\frac{q_A \cdot T}{n}} - q_A T 2^{-t}$.

■

5.5 A Short Overview on Generic Attacks Against Unilateral Protocols

In conclusion, we have three theorems which prove the best expected security of *any* SAS-based message authentication protocol. The theorems and lemmata are also proving that a protocol achieves the highest security level when the distribution of the authenticated transcripts is *uniform*. In other words, given a set of possible authenticated transcripts $\mathcal{S}$, the protocol will achieve the strongest security level when the entropy of the authenticated

string is maximum, i.e., when the distribution of the authenticated strings is uniform among $\mathcal{S}$.

To summarize the previous results, let $\mathcal{S}$ be the set of all authenticated transcripts and n be its cardinality, i.e., $n = |\mathcal{S}|$. Then,

Lemma 5.2 says that a SAS-based message authentication protocol achieves the highest security when the distribution of the authenticated transcripts is uniform. A similar result was presented by Laur and Nyberg [LN06b, Appendix B].

Theorem 5.3 says that there exists a one-shot attack against *any* SAS-based message authentication protocol, either interactive or non-interactive, which succeeds with probability essentially $\frac{1}{n}$. A similar result was presented by Laur and Nyberg [LN06b, Theorem 4, Appendix B].

Theorem 5.4 says that there exists a generic attack bounded by q_A instances of Alice and q_B instances of Bob against *any* SAS-based message authentication protocol, either interactive or non-interactive, using a weak authenticated channel which succeeds with probability essentially $1 - e^{-\frac{q_A q_B}{n}}$. Hence, they cannot be secure unless $q_A \cdot q_B$ is negligible compared to n.

Theorem 5.5 says that there exists a generic attack bounded by q_A instances of Alice and by a complexity T against *any* SAS-based non-interactive message authentication protocol using a weak authenticated channel which succeeds with probability essentially $1 - e^{-\frac{q_A \cdot T}{n}}$. Hence, they cannot be secure unless $q_A \cdot T$ is negligible compared to n.

One may ask about the relation between the one-shot security and the multi-shot security of a protocol. Here is a useful lemma given by Vaudenay [Vau05b].

Lemma 5.6 (One-Shot Attacks versus Multi-Shot Attacks).
We consider a SAS-based message authentication protocol with claimant Alice and verifier Bob in which a single SAS is sent. We denote by μ_A (resp. μ_B) the complexity of Alice's (resp. Bob's) part.

We consider adversaries such that the number of instances of Alice (resp. Bob) is at most q_A (resp. q_B). We further denote T_0 and p_0 their time complexity and probability of success, respectively.

For any q_A, q_B, and any multi-shot adversary $\mathcal{A}$, there is a generic transformation which transforms $\mathcal{A}$ into a one-shot adversary with complexity $T \leq T_0 + \mu_A q_A + \mu_B q_B$ and probability of success $p \geq p_0/q_A q_B$.

5.6 Extension to Two-Party Bilateral Protocols

Generic Bilateral Message Authentication Protocol. Now, assume that the protocol is used to authenticate a message m_A from Alice to Bob and to authenticate a message m_B from Bob to Alice. Clearly, there will be an authenticated transcript in both directions. We cannot put these messages in the same set since the authentication is done by different identities. So, we have to manage one authenticated transcript for each participant.

An adversary against a message cross-authentication is successful if at least one instance ends with an output message $\widehat{m}$ and an identity $\widehat{\mathsf{id}}$ and no instance was launched on the node of identity $\widehat{\mathsf{id}}$ with input message $\widehat{m}$.

Clearly, attacking Alice only may be reduced to an attack against a unilateral protocol. So, depending on the number of instances (one-shot or multi-shot), the adversary may proceed as in Theorem 5.3 or as in Theorem 5.4. The same remark holds for Bob.

So, the overall attack, targeting Alice and/or Bob, will succeed with probability specified by Theorem 5.3 (for one-shot) or by Theorem 5.4 (for multi-shot) at least.

5.7 Optimality of a Protocol

In this chapter, we presented generic attacks which can be applied to *any* SAS-based message authentication protocol. So, *all* SAS-based message authentication protocols are vulnerable to these attacks. This means that the most secure protocol will be vulnerable to these attacks as well.

Definition 5.7 (Optimality in Term of Deception).
A SAS-based message authentication protocol π is said to be optimal *if the best attack is one of the above generic attacks.*

A similar definition was presented by Laur and Nyberg [LN06b, Definition 1, Appendix B]. Note that they also stated that there is no optimal two-move protocol [LN06b, Corollary 1, Appendix B]. This means that any optimal protocol has at least three moves over the insecure channel (and one authenticated move).

Definition 5.8 (Optimality in Term of Moves).
A SAS-based message authentication protocol π is said to be round-optimal *if it requires three moves over the insecure channel.*

5.8 Unconditional Security

Naor, Segev, and Smith [NSS06] analyzed the generic security of (unilateral) message authentication protocols. In particular, they proposed an unconditionally secure k-round interactive protocol π, with $k \geq 3$, allowing to authenticate an n-bit message such that any (unbounded) adversary $\mathcal{A}$ succeeds in a deception attack with probability at most ε, i.e., $\mathrm{Adv}_{\pi}^{\mathsf{forge}}(\mathcal{A}) \leq \varepsilon$. This protocol requires a SAS length of $k = 2\log(1/\varepsilon) + 2\log^{k-1}(n) + \mathcal{O}(1)$ bits. Clearly, for any integer $n > 0$, any $0 < \varepsilon < 1$, choosing a big enough number of rounds, i.e., $k = \log(n)$, leads to SAS values of $2\log(1/\varepsilon) + \mathcal{O}(1)$ bits.

Naor, Segev, and Smith [NSS06] also proved that to authenticate an n-bit message where $n \geq 2\log(1/\varepsilon) + 4$ and $0 < \varepsilon < 1$, there exists no authentication protocol such that $k \leq 2\log(1/\varepsilon) - 6$. It is a lower bound on the SAS length. To break this bound, Naor, Segev and Smith [NSS06] proved that the protocol should uses a one-way function.

Remember that our above results (with respect to computational security) say that we should at least use a SAS length of $1/\varepsilon$ bits to ensure that any adversary succeeds in a deception attack with probability at most ε. Naor, Segev, and Smith [NSS06] conclude that the advantage of assuming computational security (instead of unconditional security) is to reduce the amount of authenticated communication from $2\log(1/\varepsilon)$ to $\log(1/\varepsilon)$. Also, at the same time, the number of moves is reduced.

Another interesting point is the particular case of NIMAP with respect to unconditional security. Indeed, Wang and Safavi-Naini [WSN08] showed that there is no unconditionally secure NIMAP except a trivial one which sends the message itself over the authenticated channel. Note that the proof is valid when no secret is shared, and with no additional property on the authenticated channel. This proof was also presented by Mashatan [Mas08].

Chapter

SIX

Stand-Alone Security versus Complex Settings Security

Usually, during the analysis of a protocol π, we first consider security in the stand-alone model. In that model, the adversary can only run a single instance of the protocol and thus it is much easier to write a formal security proof. However, the stand-alone security model covers only the case where no other protocols are executed together with the protocol π. In particular, it is not clear whether a concurrent execution of several different protocol instances remains secure. So, in a second time, we analyze the security of the protocol in a more complex setting, where the adversary may use several instances of the protocol as well as other concurrent protocols. We will show that concurrent compositions of SAS-based message authentication protocol(s) remain secure if some natural assumptions are satisfied. As mentioned before, we consider security against *chosen input* attacks and we assume there may be some public key in the common reference string (CRS) model.

6.1 Stand-Alone Security

Stand-alone security consider a model where there is only one adversary and one protocol instance (with all the required parties). In order to describe an attack, we describe a game between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$ who simulates the whole protocol execution, i.e., $\mathcal{C}$ simulates the behavior of all involved participants.

As example, consider any group of participants $\mathcal{G}$, involving $\mathcal{P}_1, \ldots, \mathcal{P}_n$, running a protocol π in the presence of an adversary $\mathcal{A}$. The corresponding security game is depicted in Figure 6.1. Precisely, the challenger $\mathcal{C}$ first generates system wide public parameters $\mathsf{crs} \leftarrow \mathsf{setup}$ and sends them to the adversary $\mathcal{A}$. Then, $\mathcal{A}$ can adaptively specify inputs for all parties $\mathcal{P}_1, \ldots, \mathcal{P}_n$ who can join the protocol π. More precisely, a party $\mathcal{P}_i$ remains inactive until $\mathcal{A}$ specifies its input x_i. If $x_i = \perp$ then the party $\mathcal{P}_i$ refuses to join the protocol π. Otherwise, the party $\mathcal{P}_i$ joins the protocol π with the input x_i. $\mathcal{A}$ can also adaptively corrupt protocol participants. At the end of the execution, the challenger collects all outputs $\vec{y} = (y_1, \ldots, y_n, y_a)$ and determines whether $\mathcal{A}$ succeeded in deception or not. Note that in Figure 6.1, we used a predicate $b \leftarrow \mathsf{dec}(\vec{y})$ which indicates if there where a deception ($b = 1$) or not ($b = 0$). In the following we will use the notation $\vec{y} \leftarrow \pi^{\mathcal{A}}(\mathcal{G}, \vec{x})$ to assign to $\vec{y}$ the participant outputs after a protocol execution of π in presence of an adversary $\mathcal{A}$ who chose the group $\mathcal{G}$ and the respective inputs $\vec{x}$.

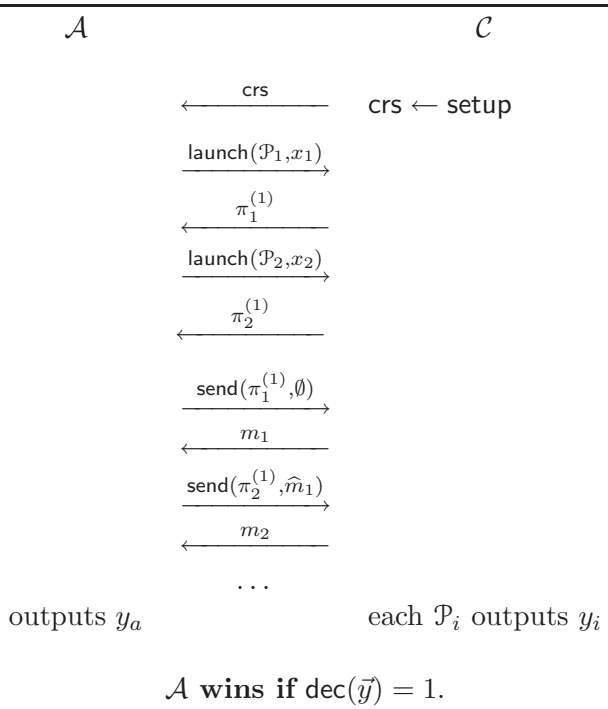


Figure 6.1. *Generic Security Game in the Stand Alone Model.*

For simple protocols, it is easy to define deception by listing all invalid end configurations, i.e., to define when $\mathsf{dec}(\vec{y})$ is equal to 0 or to 1. However, such an approach quickly becomes tedious and technical for complex protocols. Hence, we should use a generic approach.

Idealized Implementation

Many protocols are designed to meet complex security objectives. So, before starting the design, we first need to state clearly the objectives that the protocol should achieve. For that we define an idealized implementation, denoted by π°, which catches all unavoidable attacks. For instance consider a two-party unilateral message authentication protocol. In short, a party $\mathcal{P}_{\mathsf{id}_1}$ wants to send a message m to a party $\mathcal{P}_{\mathsf{id}_2}$ in an authenticated way. Obviously, we cannot guarantee that the message m reaches the destination in the real world. Indeed, the adversary can always drop all in-band messages. However, since m is authenticated (and thus integrity is guaranteed), $\mathcal{P}_{\mathsf{id}_2}$ should receive either the message m or the abort signal $\perp$. The fact that $\mathcal{P}_{\mathsf{id}_2}$ may receive $\perp$ is a good example of unavoidable attack. Note that the adversary may decide whether to corrupt the sender depending on the first in-band message which often reveals the input m. In other words, the adversarial corruption pattern can always depend on the message m.

Thanks to the ideal implementation, we can then define the security of a real protocol π through a specific game that directly quantifies how much the real execution π diverges from the idealized implementation π°.

All the security goals of the protocol can be formalized by choosing an appropriate ideal world model where a trusted third party $\mathcal{T}$ does all computations on behalf of the participants. More formally, the ideal world model is defined as follows:

Definition 6.1 (Ideal World Model).
In the ideal world model, we consider an ideal protocol π° executed in presence of participants $\mathcal{P}_1, \ldots, \mathcal{P}_n$, a trusted third party $\mathcal{T}$, and an ideal world adversary $\mathcal{A}^\circ$.

$\mathcal{T}$ can securely *exchange messages with $\mathcal{P}_1, \ldots, \mathcal{P}_n$, and with $\mathcal{A}^\circ$.*

π° can be formalized by specifying the behavior of $\mathcal{P}_1, \ldots, \mathcal{P}_n$ and $\mathcal{T}$.

All computations are done by $\mathcal{T}$. That is, $\mathcal{P}_1, \ldots, \mathcal{P}_n$ are forced to submit their inputs x_i to $\mathcal{T}$. Then, $\mathcal{T}$ computes the outputs y_i for all participants and sends them back securely to each one.

If a participant $\mathcal{P}_k$ does not submit its input x_k, $\mathcal{T}$ sends $\perp$ to all participants to abort the protocol. So, either all participants receive their outputs y_i or either all participants receive the abort output $\perp$.

As concrete example, we present an idealized implementation for a *group message authentication protocol* (GMAP), where the set of participants is determined dynamically and the final outcome is combined from all inputs. More precisely, a GMAP for a group $\mathcal{G} = \{\mathsf{id}_1, \ldots, \mathsf{id}_n\}$ works as follows:
Each participant $\mathcal{P}_{\mathsf{id}}$, with $\mathsf{id} \in \mathcal{G}$, starts with input m_{id} and ends with output $(\mathcal{G}, \vec{m}$, where $\vec{m} = (m_{\mathsf{id}_1}, \ldots, m_{\mathsf{id}_n})$. As a result, given $\mathcal{G}$ and $\vec{m}$ it is trivial to restore who participated

in the protocol and what was the corresponding inputs. The corresponding idealized implementation, as depicted in Figure 6.2, models all unavoidable attacks. In particular, note that a real world adversary can always control who joins the group by selectively blocking in-band messages. As seen before, $\mathcal{A}$ may also drop all in-band messages and thus all $\mathcal{P}_{\mathsf{id}_i}$ may receive $\perp$.

1. An initiator node $\mathcal{P}_{\mathsf{id}_*}$ sends $(\mathsf{id}_*, m_{\mathsf{id}_*})$ to $\mathcal{A}^\circ$ and then to $\mathcal{T}$.
2. The adversary adaptively determines the set of participants $\mathcal{G}$:
   (a) $\mathcal{A}^\circ$ sends an identity id to $\mathcal{T}$ who sends an invitation message to $\mathcal{P}_{\mathsf{id}}$.
   (b) $\mathcal{P}_{\mathsf{id}}$ chooses to join the protocol or not:
   Join. He joins by sending $(\mathsf{id}, m_{\mathsf{id}})$ to $\mathcal{A}^\circ$ and then to $\mathcal{T}$. ($\mathsf{id} \in \mathcal{G}$)
   Refuse. He does not join by sending $\perp$. ($\mathsf{id} \notin \mathcal{G}$)
   (c) Steps (a)–(b) are repeated until $\mathcal{A}^\circ$ stops the group formation phase.

$\star$. The group is now fixed and represented by $\mathcal{G}$. The inputs are $\vec{m} = \{\mathsf{id} \in \mathcal{G} : m_{\mathsf{id}}\}$.

3. The adversary $\mathcal{A}^\circ$ can now either halt the execution or not:
   Halt. To halt, $\mathcal{A}^\circ$ sends 0 to $\mathcal{T}$. Then, all $\mathsf{id} \in \mathcal{G}$ will receive $\perp$ from $\mathcal{T}$.
   Continue. To continue, $\mathcal{A}^\circ$ sends 1 to $\mathcal{T}$. Then, all $\mathsf{id} \in \mathcal{G}$ will receive $(\mathcal{G}, \vec{m})$ from $\mathcal{T}$.

Figure 6.2. *Idealized Implementation of a Dynamic GMAP.*

Also, note that we can obtain descriptions of different GMAP by modifying the ideal implementation. For example, if we drop the first step then we obtain a group authentication protocol with no initiator.

Back to the Real World

By using the idealized implementation π°, we are now able to determine if a (real world) adversary $\mathcal{A}$ against the (real world) protocol π succeeded in deception or not. In short, $\mathcal{A}$ *succeeds in deception* if it is impossible to achieve the same end configuration for honest participants in the ideal world.

Definition 6.2 (Success in Deception).
Let $\mathcal{G} = \{\mathsf{id}_1, \ldots, \mathsf{id}_n\}$ be the set of participants, let $\mathcal{H} \subseteq \{\mathsf{id}_1, \ldots, \mathsf{id}_n\}$ be the set of honest (non-corrupted) participants and let $(x_i)_{i \in \mathcal{H}}$ be the corresponding inputs.

Then the adversary $\mathcal{A}$ fails in deception *if one can choose inputs $(\widehat{x}_i)_{i \notin \mathcal{H}}$ such that $\mathcal{A}^\circ$ can achieve the same end configuration $(y_i)_{i \in \mathcal{H}}$ for the honest parties.*

We denote by $\mathsf{dec}(\pi^{\mathcal{A}}(\mathcal{H}, \vec{x}))$ *the predicate which indicates a deception in the real world.*

In other words, Definition 6.2 says that

- if the real attack is reproducible in the ideal world, it is not considered as a deception and the adversary fails in deception,
- if the real attack is not reproducible in the ideal world, then the adversary succeeds in deception.

Definition 6.3 (Stand-Alone Security in Term of Deception).
A protocol π *is* (T, ε)-secure in the stand-alone model *if for any* T*-time real world adversary* $\mathcal{A}$ *the deception probability* $\mathrm{Adv}^{\mathsf{forge}}_{\pi}(\mathcal{A})$ *defined by*

$$\mathrm{Adv}^{\mathsf{forge}}_{\pi}(\mathcal{A}) \quad = \quad \Pr[\mathsf{crs} \leftarrow \mathsf{setup}, (\mathcal{G}, \vec{x}) \leftarrow \mathcal{A}(\mathsf{crs}) : \ \mathsf{dec}(\pi^{\mathcal{A}}(\mathcal{G}, \vec{x}) = 1)]$$

is bounded by ε*.*

Real versus Ideal Worlds

Let us now consider *compatible* real and ideal world adversaries. A compatible pair of adversaries implies that both specify the same inputs x_i and corrupt the same set of participants in the same order. To be punctual, we assume that the setup procedure is executed also in the ideal world[1] and the corresponding input distributions and corrupted parties coincide for any value of public parameters crs.

Let $\vec{\psi} = (\psi_i, \ldots, \psi_n, \psi_a)$ and $\vec{\psi}^{\circ} = (\psi_i^{\circ}, \ldots, \psi_n^{\circ}, \psi_a^{\circ})$ denote the corresponding output distributions respectively in the real and ideal worlds. For any pair of compatible adversaries $(\mathcal{A}, \mathcal{A}^{\circ})$, if $\mathcal{A}$ succeeds in deception, then the two vectors $\vec{\psi}$ and $\vec{\psi}^{\circ}$ should be different. So, the statistical difference between the distributions $\vec{\psi}$ and $\vec{\psi}^{\circ}$ is at least $\mathrm{Adv}^{\mathsf{forge}}_{\pi}(\mathcal{A})$. Theorem 6.4 shows for the case of message authentication protocols that it is possible to reach this bound when all inputs are extractable from the outputs of honest participants. Namely, for any real adversary $\mathcal{A}$ there exists a canonical ideal world adversary $\mathcal{A}^{\circ}$ with comparable running time such that the corresponding output distributions are $\mathrm{Adv}^{\mathsf{forge}}_{\pi}(\mathcal{A})$-close.

Theorem 6.4 (Stand-Alone Security of a GMAP).
Let π *be a* (T, ε)*-secure GMAP in the stand-alone model.*

For any T*-time real world adversary* $\mathcal{A}$ *there exists a compatible* $T + \mathcal{O}(1)$*-time ideal world adversary* $\mathcal{A}^{\circ}$ *such that the corresponding output distributions in the real and ideal worlds, i.e.,* $\vec{\psi}$ *and* $\vec{\psi}^{\circ}$ *respectively, are* ε*-close.*

[1] Strictly speaking, such an assumption is unnecessary but it guarantees re-usability of a common reference string.

Proof.
Although we give the proof only for group message authentication protocols, it holds for all message authentication protocols considered in this thesis.

For the proof, we construct an universal interface $\mathcal{I}$ between the real world adversary $\mathcal{A}$ and the ideal world. As depicted in Figure 6.3, the interface $\mathcal{I}$ acts as a translation unit. From the point of view of $\mathcal{A}$, the interface $\mathcal{I}$ simulates the real world execution of π and thus acts as a challenger. $\mathcal{I}$ carries out the ideal world attacks corresponding to the real world ones.

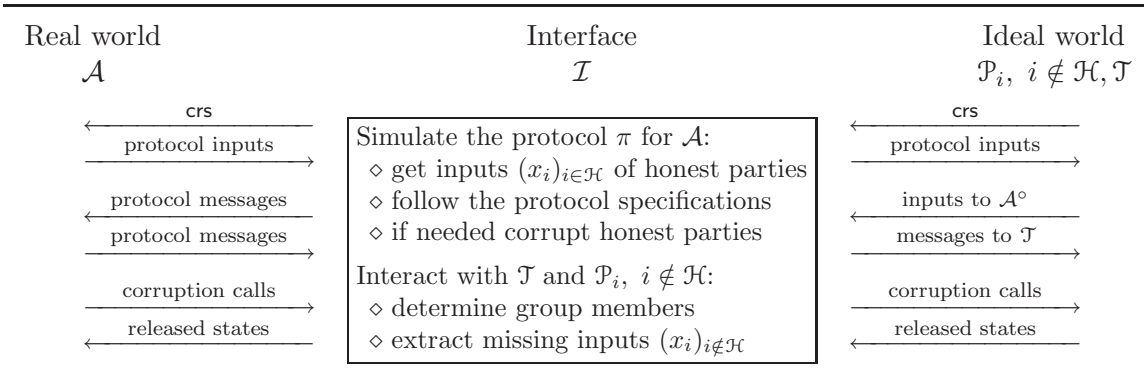


Figure 6.3. *The Canonical Interface for the Real World Adversary.*

Public parameters and protocol inputs. Note that in the security game the public parameter crs is fixed by the challenger and given to the real world adversary $\mathcal{A}$. So it may come either from the ideal world or either from the interface. Following our model, $\mathcal{A}$ chooses the protocol inputs and then gives them to the challenger, in this case to $\mathcal{I}$. Note that the protocol inputs may depend on the public parameters crs. Remember that we are using two compatible adversaries, i.e., both should specify the same protocol inputs and the same corrupted participants.

Honest participants. The simulation of honest participants is straightforward. Indeed, the public parameters crs are fixed and all honest participants send their inputs x_i to $\mathcal{A}^\circ$, in this case to $\mathcal{I}$. Hence, $\mathcal{I}$ can do all missing computations on behalf of the honest participants.

Corrupted participants. When $\mathcal{A}$ makes a corruption call, $\mathcal{I}$ just forwards the call to the ideal world and then adds all variables that are used in simulation to the released state.

Group formation. The simulation of the group formation is also possible since the actions of $\mathcal{A}$ uniquely determine when and which parties are included into the group.

Simulation termination. At the end of the simulation, $\mathcal{I}$ internally obtains all outputs of the honest parties (since they were simulated). $\mathcal{A}$ submits also its remaining outputs. As the simulation is perfect, the corresponding output vector $\vec{y}$ coincides with the outputs obtained in the real execution.

Now $\mathcal{I}$ can extract the missing inputs $(\widehat{x}_i)_{i \notin \mathcal{H}}$ from the outputs of honest parties. $\mathcal{I}$ can submit them to $\mathcal{T}$ as the inputs of corrupted participants. There are three possibilities:

1. All honest parties halt with $\perp$, then $\mathcal{I}$ must send 0 to $\mathcal{T}$.
2. All outputs of honest participants provide the same missing inputs $(\widehat{x}_i)_{i \notin \mathcal{H}}$. Then $\mathcal{I}$ must send 1 to $\mathcal{T}$.
3. The outputs of honest participants lead to different inputs $(\widehat{x}_i)_{i \notin \mathcal{H}}$. Then, $\mathcal{I}$ fails.

As assumed in the theorem statement, the failure probability (case 3) must be less than ε, otherwise the protocol cannot be (T, ε)-secure. Therefore, the compound adversary $\mathcal{I}$-$\mathcal{A}$ consisting of $\mathcal{I}$ and $\mathcal{A}$ has indeed the desired properties.

■

6.2 Security in Complex Settings

The stand-alone security model of Section 6.1 is adequate only if a protocol is executed in *isolation*. This assumption is rarely fulfilled in practice. Indeed, protocols are often executed concurrently to complete more elaborate tasks. In such settings, stand-alone security guarantees are commonly insufficient since the adversary can utilize external information that leaks from the others protocols. For example, the adversary may repeat or swap messages when several instances of the same protocol are executed at the same time. As seen in Chapter 4, Bellare and Rogaway were the first to define a formal attack model [BR93a, BR95] that considers attacks against several instances of the same authentication protocol. However, the Bellare-Rogaway model does not cover the cases when the authentication protocol is executed together with other kind of protocols. In the following, we prove that all stand-alone secure SAS-based message authentication protocols are universally composable. In other words, a SAS-based message authentication protocol π always preserves security in a *computational context* $\varrho\langle\cdot\rangle$ that uses the protocol π in a black-box way, i.e., the context $\varrho\langle\cdot\rangle$ provides only the inputs and uses only the outputs of π.

We emphasize that universal composability does not automatically guarantee security in the Bellare-Rogaway model. The main cause follows from the fact that all classical authentication protocols rely on a trusted setup procedure π_{ts} that generates long-term secrets. As a result, the universal composability guarantees security only if all protocol instances use independently generated long-term secrets. The Bellare-Rogaway model considers a setting where all protocol instances share the same long-term secrets and thus universal composability might be insufficient. Obviously, these security notions coincide if we can guarantee security in the stand-alone model with no trusted setup. The latter makes SAS-based message authentication protocols (SAS-MAP) special, since they are universally composable and secure in the Bellare-Rogaway model at the same time.

6.2.1 Reminder on Universal Composability

As emphasized above, protocols are seldomly executed in isolation. Indeed, a protocol π is often only a small part of the entire computational procedure also known as computational context. Now if a context $\varrho\langle\cdot\rangle$ has only black-box access to π, we can freely use different protocols as long as they implement the same functionality. In particular, we can compare the behavior of the real and ideal implementations π and π°. To be precise, we must compare the corresponding compound protocols $\varrho\langle\pi\rangle$ and $\varrho\langle\pi^\circ\rangle$.

Definition 6.5 (Universal Composability).
Let $\vec{x} = (x_1, \ldots, x_n, x_a)$ denote the inputs of the participants $\mathcal{P}_1, \ldots, \mathcal{P}_n$ and the input of the adversary $\mathcal{A}$ at the beginning of the context $\varrho\langle\cdot\rangle$. Similarly, let the vectors $\vec{y} = (y_1, \ldots, y_n, y_a)$ and $\vec{y}^\circ = (y_1^\circ, \ldots, y_n^\circ, y_a^\circ)$ denote the outputs of the compound protocols $\varrho\langle\pi\rangle$ and $\varrho\langle\pi^\circ\rangle$.

A protocol π is said $(T, T^\circ, T_\varrho, \varepsilon)$-universally composable *if for any input distribution $\vec{x} \leftarrow \mathfrak{D}$, for any T_ϱ-time computational context $\varrho\langle\cdot\rangle$, and for any T-time adversary $\mathcal{A}$ against the protocol $\varrho\langle\pi\rangle$, there exists a T°-time adversary $\mathcal{A}^\circ$ against $\varrho\langle\pi^\circ\rangle$ such that the statistical difference between the output distributions of $\vec{y}$ and $\vec{y}^\circ$ is at most ε.*

Definition 6.5 remains ambiguous unless we completely specify the execution and communication model. In the following, we consider the classical setting, where the adversary has full control over the protocols scheduling and message delivery. Namely, the execution of a protocol is divided into fine-grained micro-rounds. All parties are initially inactive except the adversary. The adversary can activate other participants so that only one of them is active in each micro-round. During a micro-round, the active participant can either read one incoming message or compose a single outgoing message. After that the party is suspended and the control goes back to the adversary who can choose next party for activation. The execution ends when all participants have halted. See the manuscript of Canetti [Can00] for detailed discussion and for further references. We remark that the approach outlined above corresponds to the most intuitive formalization given by Lindell [Lin03] of universal composability but there are several more popular alternatives [Can01, PW01].

Protocols with Shared Setup

Many protocols rely on pre-shared information like long-term secret keys or certificate chains. Such protocols can be divided into two phases:

- In the first phase, a trusted dealer creates and securely distributes the necessary pre-shared data.
- The second phase corresponds to the actual execution of the protocol.

Hence, the protocol π itself is a pair of sub-protocols $(\pi_{\text{ts}}, \pi_{\text{ex}})$ where π_{ts} corresponds to the *trusted setup* and π_{ex} corresponds to the actual execution. Normally, we want to reuse pre-shared data and thus different protocols may share the same setup phase π_{ts}. As a result, messages from different protocols become correlated and this creates new attack opportunities. The security model of Chapter 4 proposed by Bellare and Rogaway formalizes the corresponding threats for message authentication protocols, see the articles [BR93a, BR95].

Differently from the stand-alone model, the adversary $\mathcal{A}$ can simultaneously attack many protocol instances $\pi_{\text{ex}}^{(1)}, \ldots, \pi_{\text{ex}}^{(q)}$ that share the same setup phase π_{ts}. More formally, $\mathcal{A}$ can adaptively launch new protocol instances $\pi_{\text{ex}}^{(i)}$ by specifying the set of participants $\mathcal{G}^{(i)}$ and the corresponding inputs $\vec{x}^{(i)}$. The adversary $\mathcal{A}$ *succeeds in deception* if at least one protocol instance ends with successful deception.

Note that a shared setup phase may weaken protocol instances even if we reuse only public parameters. Hence, we must prove that security in the CRS model guarantees security in the Bellare-Rogaway model of Chapter 4.

6.2.2 Composability Guarantees of a SAS-based Message Authentication Protocol

Regardless of the desired idealized implementation it is possible to prove that all SAS-MAP are universally composable if they are secure in the stand-alone model. For brevity, we prove the corresponding result only for group authentication protocols. We then discuss on the limitations of this proof technique below.

Theorem 6.6 (Universal Composability of a SAS-MAP).
Let π be a (T_π, ε)-secure SAS-MAP in the stand-alone model.

There are constants c_1, c_2 such that the protocol π is $(T, T^\circ, T_\varrho, \varepsilon)$-universally composable whenever $T^\circ \geq c_1 \cdot T$ and $T + T_\varrho \leq T_\pi - c_2$.

Note that in the proof, we will do some implicit assumptions like no shared setup or possibility of message identification. We will then discuss on the protocol requirements in order to satisfy these assumptions.

Proof.
Let $\varrho\langle\cdot\rangle$ be a T_ϱ-time computational context and let $\mathcal{A}$ be a T-time (real world) adversary against the compound protocol $\varrho\langle\pi\rangle$. For the proof we construct an efficient interface $\mathcal{I}^*$ between $\mathcal{A}$ and the ideal world protocol $\varrho\langle\pi^\circ\rangle$. Now note that $\mathcal{I}$ depicted in Figure 6.3 and described in the proof of Theorem 6.4 is sufficient for this purpose if we can separate protocol and non-protocol messages. The corresponding construction is depicted in Figure 6.4. That is, we direct non-protocol messages past the interface $\mathcal{I}$. To be precise, we must guarantee

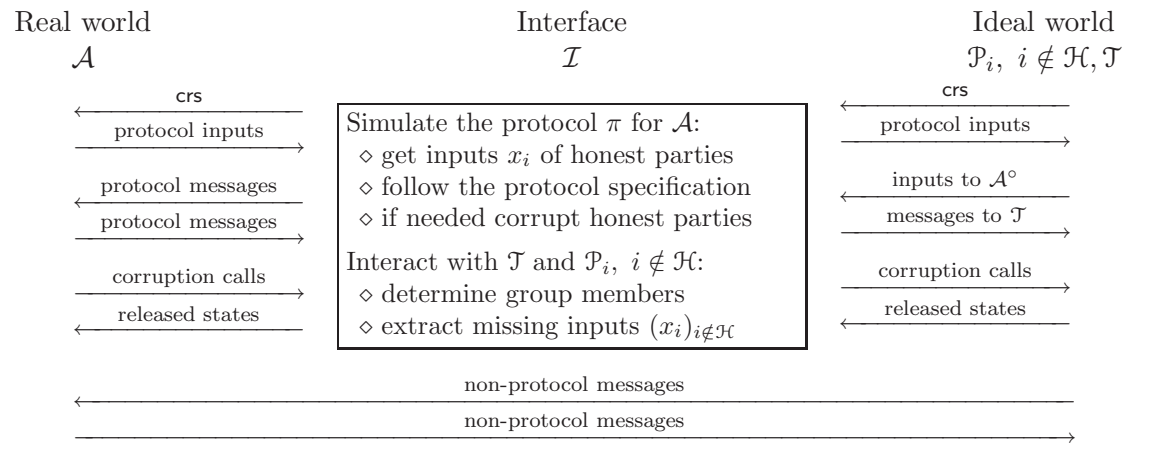


Figure 6.4. *The Canonical Interface for Complex Settings.*

that the simulation is perfect, i.e., the adversary sees the messages in the same order as in the real execution. Hence, we must additionally assume that there is a general (possibly dynamic) scheduling policy that uniquely determines in which order an honest participant $\mathcal{P}_i$ outputs protocol and non-protocol messages. Secondly, the corruption calls are handled by $\mathcal{I}$ that corrupts the honest participant and adds the variables used in simulation to the state of released participant.

Now it is possible to verify that the simulation of the protocol is perfect and that there can be a discrepancy between the real and ideal world outputs $\vec{y}$ and $\vec{y}^\circ$ only if $\mathcal{A}$ succeeds in deception. The corresponding deception probability must be less than ε or otherwise $\mathcal{A}$ together with the context $\varrho\langle\cdot\rangle$ form a new stand-alone adversary $\mathcal{A}^*$ that achieves $\mathrm{Adv}^{\mathsf{forge}}_{\pi}(\mathcal{A}^*) > \varepsilon$. The latter leads to a contradiction, since the running time of $\mathcal{A}^*$ is $T + T_\varrho + \mathcal{O}(1)$. Now the claim follows, as the overhead in the simulation is constant.

■

Shared Setup in the CRS Model. As a first limitation of Theorem 6.6, note that the interface $\mathcal{I}$ may completely fail if protocols share the same trusted setup π_{ts}. For obvious reasons, such failures are caused by protocols $\pi^{(1)}_{\mathrm{ex}}, \ldots, \pi^{(q)}_{\mathrm{ex}}$ that share the long-term secrets. If the trusted setup π_{ts} is run independently from $\mathcal{I}$, then it does not know the corresponding long-term secrets and cannot simulate the execution of honest parties. Alternatively, if the setup π_{ts} is a part of $\mathcal{I}$, then we can replace only a single protocol instance $\pi^{(i)}_{\mathrm{ex}}$ with the ideal implementation. After that the corresponding compound adversary $\mathcal{A}$-$\mathcal{I}$ knows all long-term secrets and all other protocol instances $\pi^{(1)}_{\mathrm{ex}}, \ldots, \pi^{(q)}_{\mathrm{ex}}$ become insecure. Hence, one needs more elaborate security proofs for all message authentication protocols that are based on long-term secrets. The latter is expected result, as some of these protocols are known to

be secure in the stand-alone model but insecure in the Bellare-Rogaway model.

However, if the trusted setup phase π_{ts} generates *only public values*, then these problems disappear since the knowledge of public parameters is sufficient to simulate the behavior of honest parties. In particular, the adversary $\mathcal{A}^*$ is still a valid stand-alone adversary and the proof of Theorem 6.6 still holds.

Corollary 6.7 (Universal Composability of SAS-MAP in the CRS Model). *Let π be a (T_π, ε)-secure SAS-MAP in the stand-alone model. Assume that π is instantiated in the CRS model and π may be split in a trusted setup phase and an execution phase, i.e., $\pi = (\pi_{\text{ts}}, \pi_{\text{ex}})$.*

There are constants c_1, c_2 such that the protocol is $(T, T^\circ, T_\varrho, \varepsilon)$-universally composable whenever $T^\circ \geq c_1 \cdot T$ and $T + T_\varrho \leq T_\pi - c_2$ even if the protocol shares the setup phase π_{ts} with other protocols.

This result represents the main technical difference between classical and SAS-based message authentication. SAS-based message authentication protocols are (in general) not based on long-term secrets and thus they remain secure in any computational context. Classical message authentication protocols are also universally composable as long as secret keys are used only once. If we want to reuse secret keys, we must consider $\pi_{\text{ts}}, \pi_{\text{ex}}^{(1)}, \ldots, \pi_{\text{ex}}^{(q)}$ as single multi-round authentication protocol and prove its security in the Bellare-Rogaway model of Chapter 4.

Message Identification. As a second restriction, note that the proof of Theorem 6.6 outlined above is valid only if the interface $\mathcal{I}^*$ can correctly separate protocol messages from non-protocol messages. Otherwise, messages may become switched between different protocols and the corresponding *synchronization errors* can cause arbitrary failures. To avoid such subtle issues, theoretical treatments often assume that each message contains a specific tag that uniquely determines the corresponding protocol instance [Can00]. Indeed, we can assume that each protocol has an initiator $\mathcal{P}_i$ who first broadcasts or sends directly to group members a unique tag, denoted by tag. This unique tag is appended as identifier to each protocol message. We emphasize that an adversary can alter tag. In other words, we can always avoid synchronization errors for in-band communication without excessive performance penalties. However, the latter is not true for authenticated messages, since we do not want to increase the amount of communication over the (expensive) extra channel. Finally, any SAS-MAP remains universally composable if the restriction rules of Figure 6.5 hold.

Restrictions $\mathcal{R}_1$–$\mathcal{R}_3$ are natural requirements. We can force the restriction $\mathcal{R}_4$, if we guarantee that no more than one protocol instance for the same group is run at the same time. The latter is a relatively mild limitation, since two or more parallel instances of an authentication protocol in the same group can be replaced with a single protocol instance.

$\mathcal{R}_1$: Randomness used in the protocol instance is freshly generated.

$\mathcal{R}_2$: The outputs are never used before all parties reach accepting state.

$\mathcal{R}_3$: All group members have different identities, i.e., $\mathcal{G}$ is indeed a set.

$\mathcal{R}_4$: The authenticated messages determine a unique protocol instance.

Figure 6.5. *Restriction Rules.*

Several articles [LN06a, LP08] have just postulated the usage restrictions $\mathcal{R}_1$–$\mathcal{R}_4$ and have not studied the maximal damage caused by synchronization errors. For two-party protocols, the corresponding extra advantage has been estimated by Pasini and Vaudenay [Vau05b, PV06a, PV06b]. Chapter 5 gives details on these estimations. In particular, Lemma 5.6 can be re-written as follows.

Lemma 6.8 (Security of Two-Party Message Cross-authentication Protocols). *Let π be a (T, ε)-secure message cross-authentication protocol between $\mathcal{P}_1$ and $\mathcal{P}_2$ in the stand-alone model.*

If $\mathcal{P}_1$ launches up to q_1 and $\mathcal{P}_2$ up to q_2 concurrent instances of the protocol π, then the deception probability can increase up to $q_1 q_2 \cdot \varepsilon$.

Proof (Sketch).
We consider an adversary $\mathcal{A}$ against the protocol π who can launch up to q_1 instances of π with $\mathcal{P}_1$ and up to q_2 with $\mathcal{P}_2$. A successful deception pairs an instance launched by $\mathcal{P}_1$ and an instance launched by $\mathcal{P}_2$ with the same idea than depicted in Figure 5.3. In the following we build a stand-alone adversary $\mathcal{A}^*$ against only one protocol execution.

Let ε_{ij} denote the probability that the first deception event happens for the i^{th} instance launched by $\mathcal{P}_1$ and for the j^{th} instance launched by $\mathcal{P}_2$. Then the overall deception probability $\mathrm{Adv}^{\mathsf{forge}}_{\pi,\ldots,\pi}(\mathcal{A})$ is just $\sum_{i,j} \varepsilon_{ij}$. Hence, one can create a stand-alone adversary $\mathcal{A}^*$ by simulating all protocol instances except for a random instance pair that is substituted with the challenge instance. Clearly, $\mathcal{A}^*$ succeeds in deception at least with the probability that the pair was correctly chosen, i.e.,

$$\mathrm{Adv}^{\mathsf{forge}}_{\pi}(\mathcal{A}^*) \geq \frac{1}{q_1 q_2} \cdot \mathrm{Adv}^{\mathsf{forge}}_{\pi,\ldots,\pi}(\mathcal{A}) \ .$$

Remember that $\mathrm{Adv}^{\mathsf{forge}}_{\pi}(\mathcal{A}^*) \leq \varepsilon$, the claim directly follows.

■

6.3 SAS-based Protocol Security in a Nutshell

In Section 6.1, we defined the security of a SAS-based message authentication protocol (SAS-MAP) executed in isolation. In Section 6.2, we proved that a SAS-MAP secure in the stand-alone model remains secure in complex settings in which other protocols are executed provided that $\mathcal{R}_1$–$\mathcal{R}_4$ are satisfied.

Remember that an adversary $\mathcal{A}$ succeeds in deception if the end state of at least one protocol instance $\pi^{(i)}$ is invalid, i.e., honest parties accept different outputs. Since a single instance of a SAS-MAP has non-negligible deception probability we must bound the number of protocol instances that can be launched. We say that a SAS-MAP π is (T, q, ε)-*strongly self-composable* if any T-time adversary $\mathcal{A}$ that can launch up to q protocol instances succeeds in deception with probability at most ε.

Theorem 6.9 (Security of a SAS-MAP).
Let π be a (T, ε)-secure SAS-MAP in the stand-alone model.

If restrictions $\mathcal{R}_1$–$\mathcal{R}_4$ are satisfied, then the protocol instances are also $(\tau,\ q,\ q\varepsilon)$-self-composable for $\tau = T - \mathcal{O}(1)$.

Proof.
Let $\mathcal{B}$ be such a τ-time adversary that contradicts the claim, i.e., $\mathrm{Adv}^{\mathsf{forge}}_{\pi^{(1)},\ldots,\pi^{(q)}}(\mathcal{B}) > q\varepsilon$. Without loss of generality, we can assume that an adversary launches the protocol instances in the following way. First, $\mathcal{B}$ chooses the initiator $\mathcal{P}_i$ and then the set of participants that get the introduction message tag from $\mathcal{P}_i$ and decide to reply. Second $\mathcal{B}$ provides the corresponding inputs to the participants. For simplicity, assume that tag is in $\{1, \ldots, q\}$ and let $\varepsilon^{(\mathsf{tag})}$ denote the probability that $\mathcal{B}$ succeeds in deception with respect to the instance $\pi^{(\mathsf{tag})}$. Clearly, $\mathrm{Adv}^{\mathsf{forge}}_{\pi^{(1)},\ldots,\pi^{(q)}}(\mathcal{B}) = \sum_{\mathsf{tag}=1}^{q} \varepsilon^{(\mathsf{tag})}$.

By the assumption we have $\varepsilon^{(1)} + \cdots + \varepsilon^{(q)} > q\varepsilon$. Hence, we have the following simple reduction strategy $\mathcal{A}$. Given K_p from $\mathcal{C}$:

1. Choose a target protocol instance $k \in_{\mathsf{u}} \{1, \ldots, q\}$.
2. Simulate the Bellare-Rogaway model until $\mathcal{B}$ specifies $\mathcal{G}^{(k)}$ and $\widehat{\vec{m}}^{(k)}$.
3. Send $\mathcal{G}^{(k)}$ and $\widehat{\vec{m}}^{(k)}$ to the challenger $\mathcal{C}$ in the stand-alone model.
4. Continue the simulation by generating all messages tagged by $\mathsf{tag} \neq k$.
5. Obtain other messages with $\mathsf{tag} = k$ from the stand-alone environment.
6. If required by $\mathcal{B}$, corrupt the true nodes in the stand-alone environment.

Clearly, $\mathcal{A}$ provides a perfect simulation of the Bellare-Rogaway model for $\mathcal{B}$. The probability that $\mathcal{A}$ chose the right target instance k is $\frac{1}{q}$ and thus

$$\mathrm{Adv}^{\mathsf{forge}}_{\pi^{(k)}}(\mathcal{A}) = \frac{1}{q}\,\mathrm{Adv}^{\mathsf{forge}}_{\pi^{(1)},\ldots,\pi^{(q)}}(\mathcal{B}) = \frac{\varepsilon_1 + \cdots + \varepsilon_q}{q} > \varepsilon$$

and we have a desired contradiction.

■

Chapter

SEVEN

Two-Party Unilateral Message Authentication

We start this chapter by defining the notion of two-party unilateral message authentication. Then, we first focus on non-interactive protocols and we will come back to interactive ones later. Prior proposals of non-interactive protocols such as the folklore protocol from Balfanz *et al.* [BSSW02] or the MANA family from Gehrmann-Mitchell-Nyberg [GN04, GMN04] are not optimal and the latter requires stronger properties on the authenticated channel. In Section 7.3, we present the first optimal non-interactive protocol, called PV-NIMAP, that we published in [PV06a]. It ensures the same security level as the one from Balfanz *et al.* by using much less authenticated communication. It is based on a trapdoor commitment scheme either in the CRS or in the RO model. In Section 7.4, we discuss works following the publication of our protocol. In Section 7.5, we focus on interactive protocols. We will see that there is no need for further work in interactive unilateral protocols. Indeed, the original SAS-based protocol from Vaudenay [Vau05b] is already optimal. Finally, in Section 7.6, we discuss possible applications and particularly the choice between an interactive or a non-interactive protocol.

7.1 Unilateral Message Authentication Primitive

Unilateral Message Authentication Protocols (UMAP) are typically used to exchange public keys so that *secure* communications can be set up. There exists Non-Interactive Message Authentication Protocol (NIMAP) and Interactive Message Authentication Protocol (IMAP). For a better usability, a NIMAP is preferred in many applications. For more discussion, see Section 7.6. Remember that we are using the notion of Short Authenticated String (SAS) introduced by Vaudenay [Vau05b], see Section 4.4.4. We consider that the extra authenticated channel is limited to k-bit strings for each protocol instance. In the following we consider two parties: a claimant Alice located on node A of identity id_A and a verifier Bob located on node B of identity id_B.

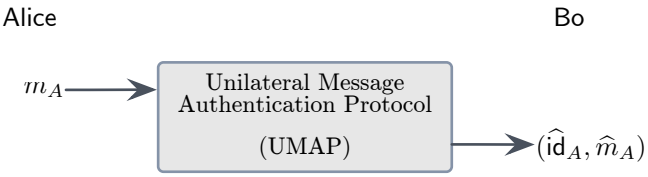


Figure 7.1. *Unilateral Message Authentication Protocol (UMAP).*

Definition 7.1 (Unilateral Message Authentication).
As depicted in Figure 7.1, a unilateral message authentication *protocol has input m_A on the side of the claimant, Alice of identity id_A, and has output $(\widehat{\mathsf{id}}_A, \widehat{m}_A)$ on the side of the verifier, Bob.*

The run is honest if the output is $\widehat{\mathsf{id}}_A = \mathsf{id}_A$ and $\widehat{m}_A = m_A$. An attack is successful when Bob gets as output $(\widehat{\mathsf{id}}_A, \widehat{m}_A)$ and no instance on the node with identity $\widehat{\mathsf{id}}_A$ was launched with input $\widehat{m}_A$.

7.2 Prior Work on Non-Interactive Protocols

7.2.1 A CRHF-based NIMAP

SSH, PGP, and GPG all use the same simple protocol in order to exchange public keys in an authenticated way. This protocol was formalized by Balfanz *et al.* [BSSW02]. It is non-interactive and based on a CRHF H. As depicted on Figure 7.2, the protocol consists in sending the message (or the public-key) over the insecure channel and then in validating their integrity with the authenticated communication. Here, the authenticated string is simply the k-bit hashed value of the input message m.

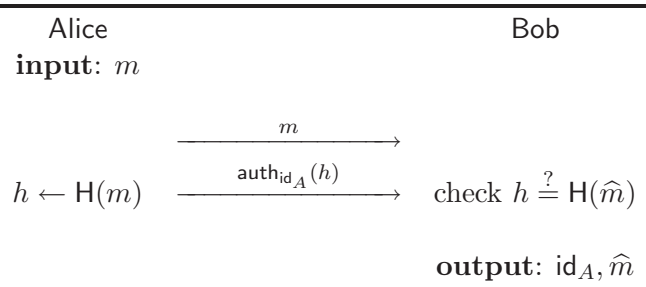


Figure 7.2. *A CRHF-based NIMAP from Balfanz et al.*

Note that the authenticated string is constant for all instances of the protocol which use the same input m, i.e., the authenticated string is $\mathsf{H}(m)$. This characteristic allows adversaries to run offline attacks. An attacker has "simply" to find a collision on the hash function between two messages m_1 and m_2 and then succeeds with probability 1.

Here is the corresponding security result from [Vau05b].

Theorem 7.2 (Security of the CRHF-based NIMAP).
Let μ be the overall time complexity of the message authentication protocol in Figure 7.2 using weak authentication. We denote by T, q, and p the time complexity, number of oracle queries launch*, and probability of success of adversaries, respectively.*

There is a generic transformation which transforms any adversary into a collision finder on H *whose complexity is $T + \mu q$ and probability of success is p.*

In short, the best known offline attack against this protocol is the collision attack. An adversary has a probability of success of $1 - e^{-\frac{1}{2}T^2 2^{-k}}$ by using T hash computations. This attack clearly succeeds for $T = \mathcal{O}(2^{k/2})$. Collision resistance requires the number of authenticated bits to be at least 160 and cannot be reduced considering offline attacks and using only weak authentication. So, the fingerprint is not user-friendly despite the trials to transform the tedious hexadecimal representation into nice word-based representations.

Moreover, note that CRHF are threatened species these days as evidenced by these attacks [BCJ+05, WLF+05, WYY05b, WYY05a, WY05]. For instance, it is possible to forge two different RSA keys with the same MD5 hash as shown in [LWdW05, LdW05].

7.2.2 A NIMAP with Strong Authentication: Mana

Another type of protocols is the MANA family proposed by Gehrmann, Mitchell, and Nyberg [GN04, GMN04]. These protocols are more resistant against offline attacks due to the randomization of the authenticated string. Indeed, the protocols are based on an universal one-way hash function family (UOWHF) H. The authenticated value is composed of two

parts: a fresh random key K and the hashed value $\mu = \mathsf{H}(m, K)$ of the input message m. So, two executions of the same protocol with the same input message lead to different authenticated strings.

The MANA family considers devices able to input and output data. They distinguish devices with limited input capability, like one button to accept or reject, and devices with elaborated input capability, like a keypad. With the same idea, they distinguish devices with limited output capability, like one LED to accept or reject, and devices with elaborated output capability, like a display.

MANA I considers a scenario in which the device on node A has an elaborated output and a limited input, while the device on node B has a limited output and an elaborated input. The authenticated channel, or manual channel, lets the user read the displayed message on device A and then type it on device B. Then, device B outputs accept/reject and the user copies it back to device A.

MANA II considers a scenario in which both the devices have an elaborated output and a limited input. The authenticated channel, or manual channel, lets the user read the two displayed messages, compare them, and if they are equals type accept on both devices.

A formalization of MANA I and MANA II is depicted in Figure 7.3.

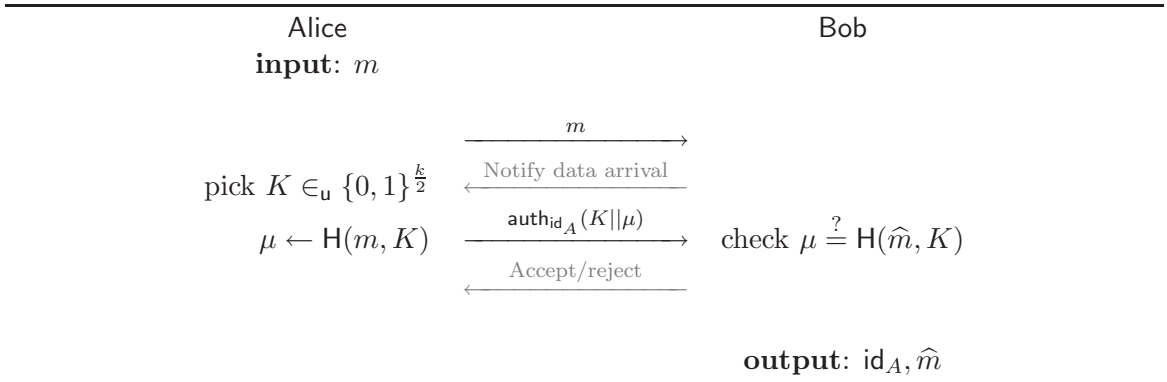


Figure 7.3. *The Formalization of MANA II.*

MANA II requires to send the authenticated message on a *stall-free* channel. Indeed, using weak authentication, an adversary who gets $\mathsf{auth}_{\mathsf{Alice}}(K||\mu)$ has enough time to find a message $\widehat{m}$ such that $\mu = \mathsf{H}(\widehat{m}, K)$ and to substitute m with $\widehat{m}$. We can only achieve security with a stronger authenticated channel which achieves stall-free transmissions. However, this requirement renders the protocol "less non-interactive" by imposing a strong assumption on the communication model.

Theorem 7.3 (Security of MANA II).
Let π be the protocol of Figure 7.3 using stall-free authentication. Let H be an ε-universal one-way hash function (UOWHF) family.

The protocol π is $(T, q_A + q_B, q_A q_B \varepsilon)$-secure where q_A (resp. q_B) denotes the number of instances of Alice (resp. Bob).

Remember that k is the size of the authenticated string and the key K and the hash μ have the same length, i.e., $k/2$. Hence, even if an adversary has an unbounded computational power, his probability of success is at most $2^{-\frac{k}{2}}$.

Proof.
Consider an adversary $\mathcal{A}$ in the stand-alone model. The adversary $\mathcal{A}$ has no advantage to send $\widehat{m}$ before it has received m. In addition, $\mathcal{A}$ is not able to send $\widehat{m}$ after receiving $K||\mu$ due to the stall-free assumption. Thus, the attacker must select m and $\widehat{m}$ at the same time and hope that the equality $\mathsf{H}(\widehat{m}, K) \stackrel{?}{=} \mathsf{H}(m, K)$ occurs (where K is unknown yet). Clearly, the assumption on H limits the probability of success to ε.

Now, consider the security in complex settings. Using Theorem 6.8, we can deduce that the probability of success of an adversary is at most $q_A q_B \varepsilon$.

■

7.3 An Optimal NIMAP: PV-NIMAP

In this section, we propose a new NIMAP, called PV-NIMAP. Our protocol has the same security as the one presented by Balfanz *et al.* [BSSW02] but using less authenticated bits and without requiring the hash function to be collision-resistant. PV-NIMAP is depicted on Figure 7.4 and was published in [PV06a]. It is based on a trapdoor commitment scheme in the Common Reference String (CRS) model or in the Random Oracle model.

In this protocol the input message m is transmitted by sending $(c, d) \leftarrow \mathsf{commit}(K_p, m)$. The message can be recovered by anyone using the open function. To authenticate this message, the hashed value of c is sent over an authenticated channel. We prove that this protocol is secure with authenticated strings which can be shorter than in the protocol of Figure 7.2. A non-deterministic commitment scheme is the heart of the protocol since an attacker cannot predict the c value and thus cannot predict the $\mathsf{H}(c)$ value which is the authenticated one. This renders collision attacks infeasible. Indeed, the best strategy now is a second preimage attack. As seen before, collision resistant hash functions are threatened species these days [BCJ+05, WLF+05, WYY05b, WYY05a, WY05], however we hope that they still resist against second preimage attacks.

Lemma 7.4 (Stand-Alone Security of PV-NIMAP).
Consider the message authentication protocol π depicted in Figure 7.4. We assume that the function H is $(T + \mu, \varepsilon_h)$-WCR and the commitment scheme is a $(T + \mu, \varepsilon_c)$-trapdoor commitment scheme in the CRS model (resp. the equivocable random oracle commitment scheme).

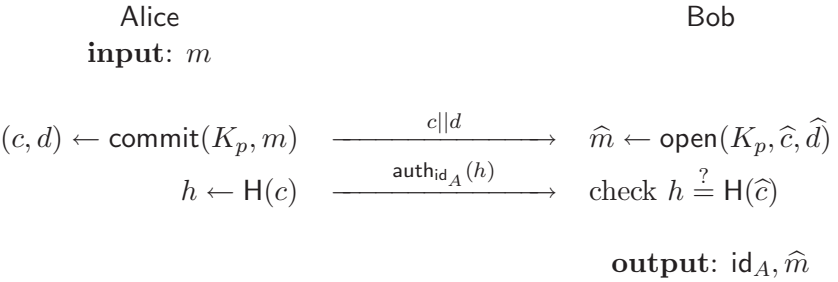


Figure 7.4. *The New (WCR-based) NIMAP: PV-NIMAP.*

There exists a (small) constant μ such that for any T the protocol π is $(T, \varepsilon_h + \varepsilon_c)$-secure in the stand-alone model.

Recall that the c value is sent through the insecure broadband channel and thus has not to be minimized. Thus, we can use an ε_c as small as desired since we can use any commitment scheme as secure as desired. So, we essentially have a (T, ε_h)-secure protocol in the stand-alone model.

Assuming that H is optimally WCR, the best WCR attack using T hash computations has a probability of success $\varepsilon_h \approx 1 - e^{-T2^{-k}}$. So, the adversary needs $T = \mathcal{O}(2^k)$ to succeed with a one-shot attack. Thus, using the same amount of authenticated bits as the protocol of Figure 7.2, our protocol has a better resistance against offline attacks. Equivalently, we can achieve the same security as the protocol of Figure 7.2, but using only half authenticated bits, e.g., 80 bits.

An example of possible implementation of our protocol is given in Figure 7.5. We implemented the commitment scheme in the random oracle model and instantiated it with a hash function.

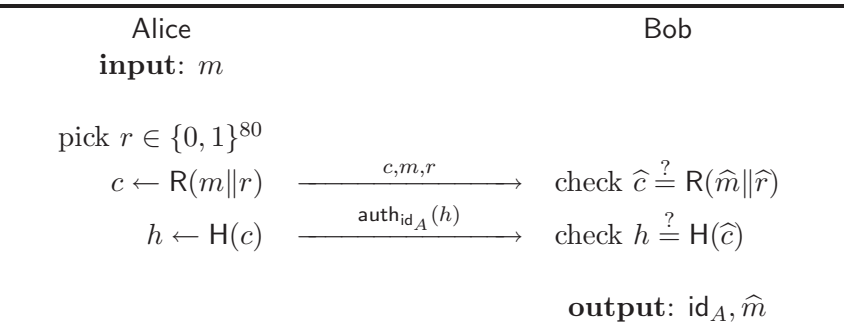
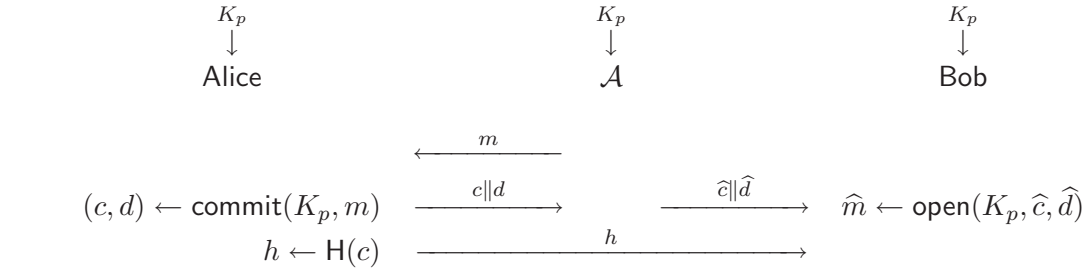


Figure 7.5. *An Example of Instantiation of PV-NIMAP.*

Proof.
Consider a T-time adversary $\mathcal{A}$ in the stand-alone model against the protocol depicted in

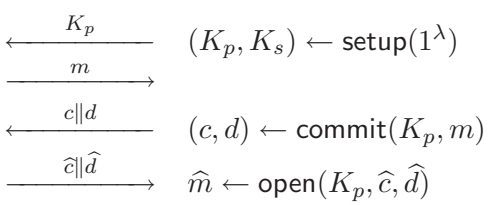
Figure 7.4. $\mathcal{A}$ follows the game as depicted in Figure 7.6 in which it runs a man-in-the middle attack.



Winning condition: $\mathsf{H}(\widehat{c}) = h$ and $\widehat{m} \neq m$.

Figure 7.6. *Game Against PV-NIMAP.*

Clearly, the game of Figure 7.6 can be reduced to an adversary who plays a game with a challenger $\mathcal{C}$ as described in Figure 7.7.



Winning condition: $\mathsf{H}(\widehat{c}) = \mathsf{H}(c)$ and $m \neq \widehat{m}$.

Figure 7.7. *Reduced Game Against PV-NIMAP.*

Given c, $\mathcal{A}$ has to find a $\widehat{c}$ such that $\mathsf{H}(\widehat{c}) = \mathsf{H}(c)$. In addition, it must find a $\widehat{d}$ which opens to $\widehat{m}$ (using $\widehat{c}$) which is different from the input m. He can of course choose a $\widehat{c}$ either equal or either different to c. We study the two cases.

Case 1 ($\widehat{c} = c$). $\mathcal{A}$ chooses $\widehat{c}$ equal to c and obviously fulfills the condition $\mathsf{H}(\widehat{c}) = \mathsf{H}(c)$. $\mathcal{A}$ should find a $\widehat{d}$ that will open to a message $\widehat{m}$ different from m. As depicted in Figure 7.8, we can reduce $\mathcal{A}$ to an adversary against the binding game of Figure 7.4. We use an algorithm $\mathcal{B}$ bounded by complexity μ which plays the binding game with a challenger $\mathcal{D}$ on one side and simulates a challenger for $\mathcal{A}$ on the other side at the same time. Using $\mathcal{A}$ and $\mathcal{B}$, we construct an adversary $\mathcal{AB}$ which plays the binding game. Note that $\mathcal{AB}$ has a complexity bounded by $T + \mu$.

First, $\mathcal{D}$ generates the pair of keys (K_p, K_S) and sends K_p to $\mathcal{B}$. $\mathcal{B}$ sends it to $\mathcal{A}$ and receives a message m from $\mathcal{A}$. He computes (c, d) using the commit function with K_p and sends $c\|d$ to $\mathcal{A}$. As assumed, $\mathcal{A}$ chooses a $\widehat{c}$ equal to c and also sends $\widehat{c}\|\widehat{d}$ to $\mathcal{B}$. $\mathcal{B}$ can now deduce $\widehat{m}$ using the open function with inputs c and $\widehat{d}$. Finally, $\mathcal{B}$ sends all required values to the challenger $\mathcal{D}$.

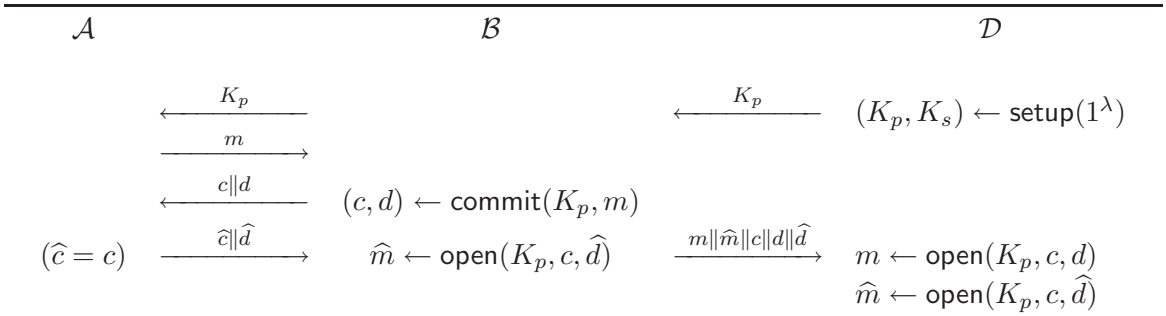


Figure 7.8. *Reduction to the SB Game ($\widehat{c} = c$).*

Note that $\mathcal{B}$ simulates perfectly a challenger for $\mathcal{A}$. Hence, $\mathcal{A}$ and $\mathcal{AB}$ win their respective game at the same time. Consequently, both win with the same probability of success. Recall that the probability of success of an adversary bounded by a complexity $T + \mu$ against the binding game of Figure 7.4 is smaller than ε_c when the commitment scheme is a $(T + \mu, \varepsilon_c)$-trapdoor commitment scheme (see Section 3.5.8). Hence, the probability that $\mathcal{A}$ succeeds and $c = \widehat{c}$ is at most ε_c. Note that this case equally applies to equivocable random oracle commitment schemes (see Section 3.5.9.2).

Case 2 ($\widehat{c} \neq c$). $\mathcal{A}$ searches a $\widehat{c}$ different from c. As depicted on Figure 7.9, we can reduce $\mathcal{A}$ to an adversary against a second preimage search game. We use an algorithm $\mathcal{B}$ bounded by a complexity μ with the help of one query to the equivocate oracle. $\mathcal{B}$ plays the second preimage game with a challenger $\mathcal{D}$ on one side and simulate a challenger for $\mathcal{A}$ on the other side at the same time. Using $\mathcal{A}$ and $\mathcal{B}$, we construct an adversary $\mathcal{AB}$ which plays the second preimage game with $\mathcal{D}$. Note that $\mathcal{AB}$ has a complexity bounded by $T + \mu$.

First, $\mathcal{B}$ generates the keys and sends K_p to $\mathcal{A}$. $\mathcal{B}$ receives a message m from $\mathcal{A}$ and receives a challenge c from $\mathcal{C}$. $\mathcal{B}$ can deduce the decommit value d by calling the oracle $\mathsf{equivocate}(m, c)$. Note that c has been picked uniformly and consequently the distribution of (c, d) is the same as if they have been yield by the commit algorithm. Then, $\mathcal{B}$ can send $c\|d$ to $\mathcal{A}$. $\mathcal{A}$ sends a $\widehat{c}\|\widehat{d}$ to $\mathcal{B}$. Finally, $\mathcal{B}$ sends it to the challenger $\mathcal{D}$.

Note that $\mathcal{B}$ simulates perfectly a challenger for $\mathcal{A}$. Hence, $\mathcal{A}$ and $\mathcal{AB}$ win their respective game at the same time and consequently with the same probability of success. Recall that the probability of success of an adversary against a second preimage game

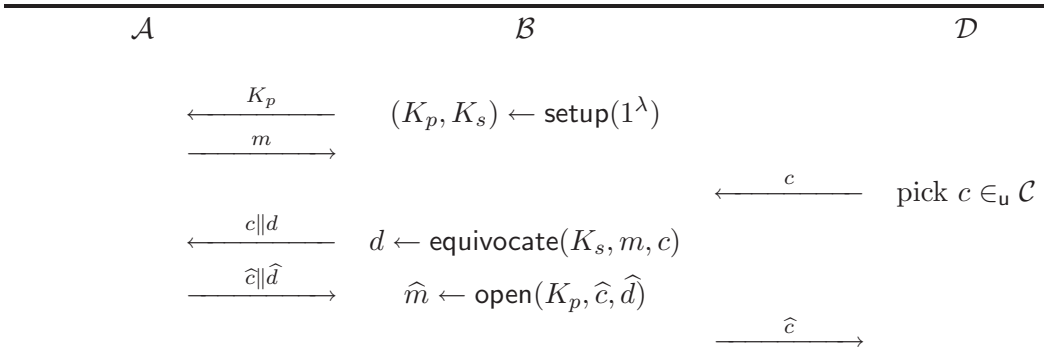


Figure 7.9. *Reduction to the WCR Game ($\widehat{c} \neq c$).*

bounded by a complexity $T + \mu$ is smaller than ε_h when H is a $(T + \mu, \varepsilon_h)$-WCR hash function (see Section 3.2.2). Hence, the probability that $\mathcal{A}$ succeeds and $c \neq \widehat{c}$ is at most ε_h. Note that the proof equally applies to equivocable oracle commitment schemes since it is unlikely that the challenge c was output by a commit oracle.

We conclude that any T-time adversary in the stand-alone model against the protocol of Figure 7.4 has a probability of success smaller than $\varepsilon_c + \varepsilon_h$ when the protocol uses a $(T + \mu, \varepsilon_h)$-WCR hash function H and a $(T + \mu, \varepsilon_c)$-trapdoor commitment scheme.

■

We consider now more powerful adversaries and we consider the security in complex settings.

Theorem 7.5 (Security of PV-NIMAP).
Consider the message authentication protocol π depicted in Figure 7.4. We assume that the function H is $(T + \mu, \varepsilon_h)$-WCR and the commitment scheme is a $(T + \mu, \varepsilon_c)$-trapdoor commitment scheme in the CRS model (resp. the equivocable random oracle commitment scheme).

There exists a (small) constant μ such that for any T the protocol π is $(T, q_A+1, q_A(\varepsilon_h+\varepsilon_c))$-secure where q_A denotes the number of instances of Alice.

Assuming that WCR hash functions and trapdoor commitments such that $\varepsilon_c \ll \varepsilon_h$ and $\varepsilon_h = \mathcal{O}(T2^{-k})$ exist, we have $p = \mathcal{O}(q_A T 2^{-k})$. As an example, assuming that an adversary is limited to $q_A \leq 2^{10}$, $T \leq 2^{70}$, and that the security level requires $p \leq 2^{-20}$, the protocol of Figure 7.2 requires $k \geq 160$ while our protocol PV-NIMAP only requires $k \geq 100$.

Even though collisions on MD5 have been found [WY05], using MD5 [Riv92], our protocol PV-NIMAP still achieves a quite luxurious security since we only need resistance to second

preimages only. However, the protocol of Figure 7.2 is insecure.

Remember from Chapter 5 that any NIMAP cannot be secure unless $T \cdot q_A$ is negligible against n. Here, n is equal to 2^k. Thus, our proposed protocol is *optimal* provided that WCR hash functions and trapdoor commitment schemes such that $\varepsilon_c \ll \varepsilon_h = \mathcal{O}(T2^{-k})$ exist. By comparison with our protocol, we can note that the protocol of Figure 7.2 is not optimal.

Proof.
Recall from Lemma 7.4 that any adversary bounded by a complexity τ in the stand-alone model has a probability of success at most $\varepsilon_h + \varepsilon_c$.

Consider any adversary who launches q_A instances of Alice and q_B instances of Bob. Clearly, we can simulate all instances of Bob, pick one who will make the attack succeeds, and launch only this one. Hence, we reduce to $q_B = 1$. We have $T = \tau + \mathcal{O}(q_B)$.

Using Theorem 6.8, we conclude that any adversary has a probability of success at most $q_A(\varepsilon_h + \varepsilon_c)$.

■

7.4 Following Works

After the publication of the above contribution, i.e., PV-NIMAP, some work was done on this area. So, we briefly present them.

An HCR-based NIMAP. Mashatan and Stinson [MS07, Mas08] studied generic NIMAP security. They concluded that in a chosen-message adversarial model, the security of a NIMAP relies on some binding property between the data sent over the insecure channel and the authenticated string. Clearly, an adversary trying to attack a NIMAP protocol is equivalent to an adversary trying to attack a Hybrid-Collision Resistant (HCR) hash function. So, they defined HCR hash functions as follows: a function H is (T, ε)-HCR if any adversary $\mathcal{A}$ bounded by complexity T wins the HCR game with probability at most ε. The HCR game lets the adversary to choose a message $m \in \{0,1\}^{\ell_1}$, then a challenger chooses a random $K \in \{0,1\}^{\ell_2}$, and finally the adversary gives a $L \in \{0,1\}^{\ell_1+\ell_2}$ and wins if $L \neq m\|K$ and $H(m\|K) = H(L)$. The final protocol [MS07, Mas08] based on a HCR hash function is depicted in Figure 7.10.

Here is the corresponding security result from [Mas08].

Theorem 7.6 (Security of the HCR-based NIMAP).
Let H be a (T, ε)-HCR hash function. The protocol depicted in Figure 7.10 is $(T,\ q{+}1,\ q\varepsilon)$-secure.

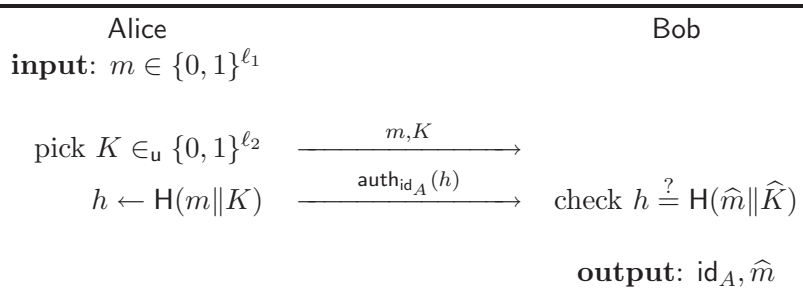


Figure 7.10. *An HCR-based NIMAP.*

For the same security level, this protocol requires the same amount of authenticated data, i.e., $k \geq 100$, by using a large enough ℓ_2, i.e., $\ell_2 \geq 80$. As advantage, this protocol does not use any commitment scheme and requires no public parameters. One may note that the HCR definition is very close to the eTCR one, see Section 3.2.5.

An eTCR-based NIMAP. Reyhanitabar *et al.* [RWSN07] do a similar work as Mashatan and Stinson [MS07, Mas08], and quite at the same time.

As depicted in Figure 7.11, they propose a protocol based on an eTCR hash function, see Section 3.2.5.

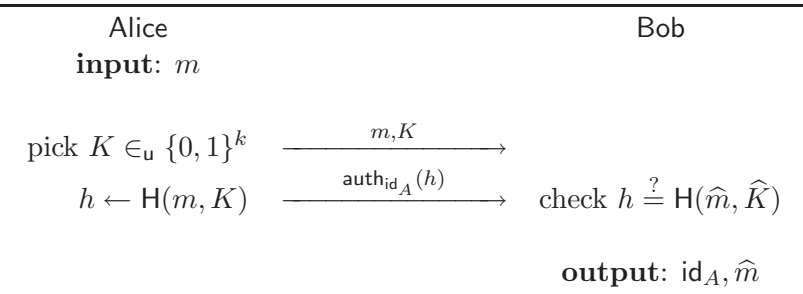


Figure 7.11. *An eTCR-based NIMAP.*

Here is the corresponding security result from [RWSN07].

Theorem 7.7 (Security of the eTCR-based NIMAP).
Let H be a (T, ε)-eTCR hash function. The protocol depicted in Figure 7.11 is $(T - \mu q - \sigma,\ q+1,\ q\varepsilon)$-secure where μ is the maximal complexity of Alice and σ the complexity for a H computation.

Note that Reyhanitabar *et al.* also propose a practical implementation of the eTCR hash function by using the Merkle-Damgård hash construction [Dam90, Mer90] and by using SHA1 [SHA95] in a randomized hashing mode as compression function. For the same security

level as PV-NIMAP, this protocol requires a SAS length of $k = 100 + \log_2(L+2)$ where L is the block length in their eTCR construction (blocks are 512-bit long by using SHA1).

7.5 On Interactive Protocols

The Original SAS-based IMAP: Vau-SAS-IMAP. Another interesting unilateral protocol is the one from Vaudenay [Vau05b], here called Vau-SAS-IMAP. Due to interactivity, this protocol avoids offline attacks. Indeed, after the exchange of the three messages through the insecure channel, Alice has to authenticate a short string $\mathsf{SAS} = R_A \oplus R_B$ where R_A and R_B were randomly selected at the beginning by Alice and Bob respectively. Note that the SAS is independent of the message m, but only depends on the fresh R_A and R_B, and thus no offline attack is possible. The protocol is depicted in Figure 7.12.

Alice		Bob
input: m		
pick $R_A \in_{\mathsf{u}} \{0,1\}^k$		pick $R_B \in_{\mathsf{u}} \{0,1\}^k$
$(c,d) \leftarrow \mathsf{commit}(m, R_A)$	$\xrightarrow{m,c}$	
	$\xleftarrow{R_B}$	
	$\xrightarrow{d}$	$\widehat{R}_A \leftarrow \mathsf{open}(\widehat{m}, \widehat{c}, \widehat{d})$
$\mathsf{SAS} \leftarrow R_A \oplus \widehat{R}_B$	$\xrightarrow{\mathsf{auth}_{\mathsf{id}_A}(\mathsf{SAS})}$	check $\mathsf{SAS} \stackrel{?}{=} \widehat{R}_A \oplus R_B$
		output: $\mathsf{id}_A, \widehat{m}$

Figure 7.12. *The Original SAS-based IMAP: Vau-SAS-IMAP.*

Vaudenay [Vau05b] proves that an adversary in the stand-alone model has a probability of success at most $2^{-k} + \varepsilon$ where $\varepsilon \geq 0$ represents a small advantage for attackers due to a non-perfect commitment scheme (non perfectly hiding).

Here is the corresponding security result from [Vau05b]

Theorem 7.8 (Stand-Alone Security of Vau-SAS-IMAP).
Assume a commitment scheme which is either (T, ε)-extractable or (T, ε)-equivocable.

There exists a (small) constant μ such that the protocol depicted in Figure 7.12 is $(T + \mu, 2^{-k} + \varepsilon)$-secure in the stand-alone model.

This protocol also resists against general attacks run by any adversary bounded by a number of protocol runs, i.e., the online complexity. As before, by using Theorem 6.8. we deduce that attacks which use q_A instances of Alice and q_B instances of Bob have a probability of success against this protocol at most $q_A q_B (2^{-k} + \varepsilon)$.

Theorem 5.3 tells us that the security of the protocol of Figure 7.12 is optimal among all comparable protocols since it reaches the maximal security for a message authentication protocol and there cannot exists a better protocol using the same amount of authenticated bits.

Using the previous example, assuming that an adversary is limited to $q_A \leq 2^{10}$, $T \leq 2^{70}$, and that the security level requires $p \leq 2^{-20}$, the optimal IMAP protocol of Figure 7.12 requires $k \geq 50$ while our optimal PV-NIMAP requires $k \geq 100$. We clearly see the gap between IMAP and NIMAP efficiency with respect to the amount of authenticated data.

An ICR-based IMAP. Recently, Mashatan and Stinson [MS08, Mas08] proposed a new interactive protocol. However, this protocol is not in the same adversarial model than we use in this thesis. Indeed, they consider *offline complexity* as the computational complexity before the target instance is launched and the *online (computational) complexity* as the computational complexity while the target instance is running. In other words, they limit the online computational complexity which leads to a different adversarial model.

We briefly present their protocol in Figure 7.13 with no additional detail. The protocol is based on an Interactive-Collision Resistant (ICR) hash function, see [MS08, Mas08] for more details.

Alice		Bob
input: m		
pick $K \in_{\mathsf{u}} \{0,1\}^{\ell_2}$	$\xrightarrow{m,K}$	
	$\xleftarrow{R}$	pick $R \in_{\mathsf{u}} \{0,1\}^{\ell_3}$
$\mathsf{SAS} \leftarrow H(m\|K\|\widehat{R})$	$\xrightarrow{\mathsf{auth}_{\mathsf{id}_A}(\mathsf{SAS})}$	check $\mathsf{SAS} \stackrel{?}{=} H(\widehat{m}\|\widehat{K}\|R)$
		output: $\mathsf{id}_A, \widehat{m}$

Figure 7.13. *An ICR-based IMAP.*

By letting the same security parameters as before, this protocol requires $k \geq 30$, resp. $k \geq 50$, authenticated bits when the online computational complexity is $t_{on} = 2^{10}$, resp. $t_{on} = 2^{30}$.

Note that this protocol only uses two moves over the insecure channel (in a special adversarial model) while Laur and Nyberg [LN06b, Corollary 1, Appendix B] showed that the minimal number of rounds is three (in the usual adversarial model), see Section 5.7.

7.6 Applications

Unilateral SAS-based message authentication protocols can typically be used where public-key cryptography is needed, and consequently where authentication of a public key is required. For instance,

- distant host authentication, e.g., SSH,
- e-mail authentication, e.g., GPG signature,
- secure e-mail, e.g., GPG encryption,
- secure voice over IP, e.g., PGPfone.

Another possible application can be authentication of legal documents. For instance, if two persons would exchange a document with no complex appendix, such as GPG signature, they can simply use one of the above protocols.

One may balance the advantages of a non-interactive unilateral message authentication protocol with respect to an interactive one, and vice-versa:

Usability. An interactive protocol allows shorter authenticated strings and thus is more user-friendly. However, an interactive protocol forces the participants to be synchronized and in some applications may be less user-friendly.

Cost. In terms of amount of authenticated bits, interactive protocols have a lower cost since they do not have to resist against offline attacks and thus can use shorter SAS. An exception is MANA. Indeed, it is a non-interactive protocol using only few authenticated bits. On the other hand, it requires a stronger authenticated channel which is more expensive than a weak authenticated one.

Security. We saw that both non-interactive and interactive protocols are vulnerable against active attacks. However, only the non-interactive protocols are vulnerable against offline ones.

Fortunately, each type has advantages and inconvenients. The selection should be done by the designer with respect to the target application.

SSH. Users connect to SSH servers. Usually, the authentication of the server is done by authenticating its public key while the authentication of the client is done by giving a username-password (after the secure tunnel is established).

So, the authentication of the public key is of utmost importance. Most of the time, the authentication of the distant public key is made by comparing the fingerprint computed

locally with the distant fingerprint which is obtained from the SSH server in an authenticated way. One advantage of this approach is the non-interactivity. Indeed, we cannot suppose that there is someone that will give to the users their personal SAS in order to authenticate the server public-key. Clearly, an interactive protocol is not a good choice.

PGP, GPG. These two applications actually use a message authentication protocol to authenticate public keys. The used protocol is non-interactive. Consequently, it allows two distant users to run a non-synchronized authentication. On the other hand, they must authenticate a long string, e.g., 160 bits (often represented with nice words).

An interactive protocol may be used for peer-to-peer key authentication as we will see in Section 7.6. Here, we assume that two distant users can run a synchronized protocol. The interactivity has the advantage of allowing shorter SASs.

PGPfone. PGPfone is a software package allowing transform a computer into a secure telephone. Suppose two persons want to communicate securely, i.e., Bob calls Alice. A possible scenario can be to start with a non-secure communication. If Bob knows Alice, the non-secure communication is an authenticated channel. Thus, Alice can send its public key to Bob, e.g., using another TCP port or by email, and then she can authenticate their public key by spelling its fingerprint. After this authentication step, the two users can setup a secure communication by using the public key which has just been exchanged. Clearly, both users are eventually synchronized, thus an interactive protocol can be run simultaneously on another TCP port with no additional constraint. This interactive method would allow shorter SASs.

Bluetooth Devices Pairing and Wireless USB. An alternative method to Bluetooth pairing can be to exchange a public key and then authenticate it. Note that a pairing requires all devices available at the same time. Consequently, an interactive protocol requires no additional constraint. Thus, short SAS can be used to authenticate public keys. For small devices, this can be restrictive since costly commitment schemes can be hard to implement. Recall that protocol of Figure 7.12 requires a commitment scheme.

Wireless USB, or any similar standard, will appear soon. Some USB devices, such as printers, will become wireless. As for Bluetooth, a WUSB device requires a pairing step otherwise attacks may be run. Let us consider some applications. We do not focus on the implementation of the authentication protocol, but rather on the user interface.

A headset has no keyboard and no display. A solution to exchange the SAS would be that the ear-phone pronounces the SAS and then the user types it on the telephone keyboard.

To establish a security association with a computer keyboard, the users could simply type on his keyboard a SAS displayed on the screen of the computer. Here, we consider that

display–user–keyboard is an authenticated channel. Note that our protocol does not need confidentiality on the channel to transmit the SAS, unlike a PIN code.

When a user on the road would like to print a confidential document from his laptop computer to a wireless printer, authentication of the printer is necessary. In the case where the printer has a display, it is not a problem: the printer just displays the SAS which should be then typed on the laptop. Otherwise, the printer can still print the SAS on a page. The protocol can be interactive in this case to allow short SAS.

A hard disk has in general no input and no output with the user. Here are some alternate ways for "deaf-mute" devices:

1. Start with a wired connection to setup a security association.
2. "Hard-code" a key on the device and print them on its box. The user has to type it on its computer during the authentication step.
3. Use *resurrecting duckling* paradigm from Stajano-Anderson [SA99]. In short, when a device is booted the first time, it searches another device in its vicinity and then always trusts the first one found.
4. Use a distance bounding authentication (see Section 2.5.2) which is well adapted in this case since we are doing pairing of near by devices.

A Peer-To-Peer File Authentication

It seems reasonable to assume that two users can be available during the exchange of a message, e.g., a public key. In this section, we propose a peer-to-peer application that helps to authenticate files.

The application communicates through the Internet network (the insecure channel). In addition, they need a human communication channel, typically a telephone (the authenticated channel). The application is based on the client-server model and works as follow: Suppose Bob wants to receive an authenticated file from Alice. Bob launches the SAS File Exchange application and waits for a connection (i.e., he is the server). Then, he contacts Alice and asks her to authenticate one of her files. Alice launches the application too, selects the file and the destination address, and starts the protocol (i.e., she is the client). At the end both should check by telephone that the two SASs they have matches.

The Vaudenay SAS-based authentication protocol, called Vau-SAS-IMAP and depicted in Figure 7.12, has been adapted and the implemented version is depicted in Figure 7.14.

Note that if Alice and Bob have already exchanged the file, e.g., by email, only authentication is made. Otherwise, i.e., $m_B = \perp$, the file is exchanged at the beginning of the protocol.

Alice		Bob
input: m		**input**: m_B
		Wait for connection...
	$\xrightarrow{\text{open connection}}$	
	$\xleftarrow{\mathsf{file\_request}}$	$\mathsf{file\_request} \leftarrow \begin{cases} 1 & \text{if } m_B = \perp \\ 0 & \text{otherwise} \end{cases}$
if $\mathsf{file\_request} = 1$ then ask Alice		
$\mathsf{answer} \leftarrow \begin{cases} 1 & \text{if she accepts} \\ 0 & \text{otherwise} \end{cases}$	$\xrightarrow{\mathsf{answer}}$	
if $\mathsf{answer} = 0$ then abort		if $\mathsf{answer} = 0$ then abort
	$\xrightarrow{m \text{ (if file\_request=1)}}$	$\widehat{m}_B \leftarrow \begin{cases} \widehat{m} & \text{if } m_B = \perp \\ m_B & \text{otherwise} \end{cases}$
pick $R_A \in_{\mathsf{u}} \{0,1\}^k$		
pick $Rdm \in_{\mathsf{u}} \{0,1\}^{80}$		
$c \leftarrow \mathsf{SHA1}(m \Vert R_A \Vert Rdm)$	$\xrightarrow{c}$	
	$\xleftarrow{R_B}$	pick $R_B \in_{\mathsf{u}} \{0,1\}^k$
	$\xrightarrow{R_A}$	
	$\xrightarrow{Rdm}$	
$\mathsf{SAS} \leftarrow R_A \oplus \widehat{R}_B$	$\xrightarrow{\mathsf{auth}_{\mathsf{id}_A}(\mathsf{SAS})}$	check $\widehat{c} \stackrel{?}{=} \mathsf{SHA1}(\widehat{m}_B \Vert \widehat{R}_a \Vert \widehat{Rdm})$
		check $\mathsf{SAS} \stackrel{?}{=} \widehat{R}_A \oplus R_B$
		output: Alice, $\widehat{m}_B$

Figure 7.14. *Implementation of Vau-SAS-IMAP with the Random Oracle Commitment.*

The final application is composed of three windows: The *main window* is the only one visible at launch time and allows the user to open one of the two others. The *receive window* is typically opened by Bob to start a server, while the *send window* is opened to send the file to the destination. Figure 7.15 shows all three windows. In particular, it shows the receive and send windows in a verbose mode.

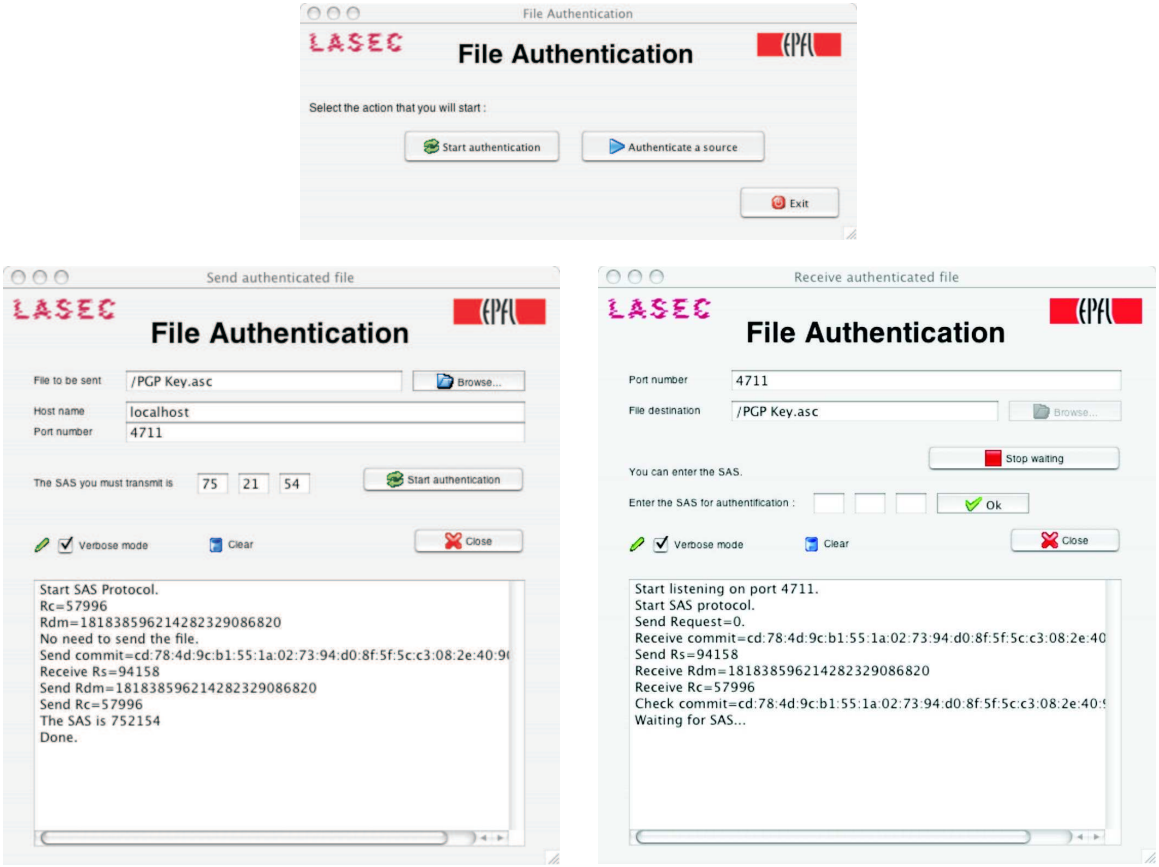


Figure 7.15. *SAS File Exchange.*

At the end of the protocol, Alice should authenticate its displayed SAS by reading it while Bob types it. We used a six decimal digits SAS, but only five are really used and the last is used to check the redundancy. This particularity was added in order to allow users to make mistake(s) without reducing the security. If simply two trials were allowed, any attacker would have had two trials to attack the protocol. Consequently, the probability of success would have been twice the probability of success of the original protocol and this is not good. In our solution when Bob enters a SAS, first the redundancy is checked. If the redundancy is bad, Bob has entered a non-valid SAS and can try another one (only one). Note that an attacker has no chance trying to use a SAS with bad redundancy since it would be rejected

with probability 1. If the redundancy is correct, either the SAS entered is correct and also the file is authenticated, or it is false and the file authentication is aborted since an attack may have occurred.

The proposed application allows users to exchange files in an authenticated way. It can be used to exchange PGP or GPG public keys by authenticating only six decimal digits. Note that the current method uses the protocol of Figure 7.2 (using fingerprints) and requires 160 authenticated bits, e.g., 32 hexadecimal digits. Fortunately, the proposed protocol requires the exchange of only 15 authenticated bits which is much smaller than the current method.

Chapter

EIGHT

Two-party Bilateral Message Authentication

We start this chapter by defining the notion of two-party bilateral message authentication. Then, we study prior proposals and in particular the original SAS-based bilateral protocol from Vaudenay [Vau05b]. This protocol enables message cross-authentication by interleaving two unilateral protocols, one in each direction. However, this protocol is not optimal and there is no formal security proof. In Section 8.3 we propose an *optimal* protocol, called PV-SAS-MMA, achieving message mutual-authentication (MMA) while in Section 8.4 we propose an *optimal* protocol, called PV-SAS-MCA, achieving message cross-authentication (MCA). We published both protocols in [PV06b]. Both are provably secure and optimal with respect to the SAS length and the number of moves. In Section 8.5, we discuss works following the publication of our protocols. Finally, in Section 8.6, we discuss possible applications.

8.1 Bilateral Message Authentication Primitives

Any unilateral message authentication protocol can be turned into a bilateral message cross-authentication protocol. Namely, each party runs a unilateral message authentication protocol. However, such kind of construction is in general not optimal with respect to the number of moves. As well, such a construction may lead to two different SASs while in many applications we prefer two equal SASs for usability reasons and because some authentication channels may provide symmetric authentication at no extra cost.

Bilateral message authentication is typically used to agree or to exchange public keys. Such a protocol can be for instance combined with a key agreement protocol to build an authenticated key agreement as we will see in Chapter 10. We distinguish two types of bilateral message authentication: mutual-authentication is used to authenticate the same message in the two directions while cross-authentication is used to exchange two messages in an authenticated way. As in the previous chapter, consider two parties: Alice located on node A of identity id_A and Bob located on node B of identity id_B.

8.1.1 Message Mutual-Authentication

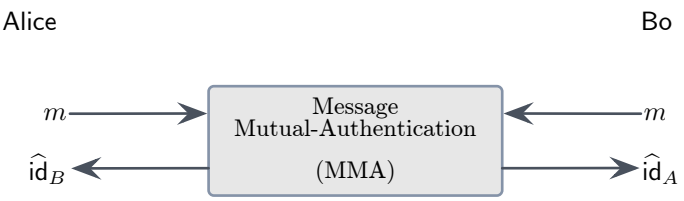


Figure 8.1. *Message Mutual-Authentication (MMA).*

Definition 8.1 (Message Mutual-Authentication).
A Message Mutual-Authentication *(MMA) protocol between Alice and Bob starts with inputs m_A and m_B and ends with outputs $\widehat{id}_B$ and $\widehat{id}_A$, respectively.*

An honest run should lead to $m_A = m_B$, $\widehat{id}_B = id_B$, and $\widehat{id}_A = id_A$.

An adversary is successful if some instance of an uncorrupted node started with any input m and ended with any output $\widehat{\mathsf{id}}$ such that no instance of the node of identity $\widehat{\mathsf{id}}$ was launched with input m.

8.1.2 Message Cross-Authentication

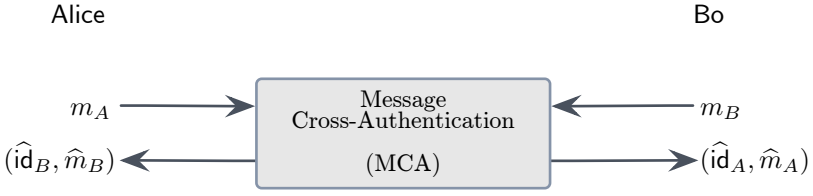


Figure 8.2. *Message Cross-Authentication (MCA).*

Definition 8.2 (Message Cross-Authentication).
A Message Cross-Authentication *(MCA) protocol between Alice and Bob starts with inputs* m_A *and* m_B *and ends with outputs* $(\widehat{\mathsf{id}}_B, \widehat{m}_B)$ *and* $(\widehat{\mathsf{id}}_A, \widehat{m}_A)$*, respectively.*

An honest run should lead to $(\widehat{\mathsf{id}}_A, \widehat{m}_A) = (\mathsf{id}_A, m_A)$ *and* $(\widehat{\mathsf{id}}_B, \widehat{m}_B) = (\mathsf{id}_B, m_B)$*.*

An adversary is successful if some instance ended of an uncorrupted node with a pair (m, id) *but no instance of the node of identity* id *was launched with input* m*.*

8.1.3 MCA versus MMA Protocols

Obviously, we can transform an MCA protocol into an MMA protocol by just checking on both sides that the output message is equal to the input one.

We can also transform an MMA protocol into an MCA protocol at the price of at most two extra moves: Alice first sends her input message m_A to Bob and then Bob sends his input message m_B to Alice. Alice and Bob initiate an MMA protocol with input $m_A \| \widehat{m}_B$ and $\widehat{m}_A \| m_B$ respectively. If no attack occurred, the final outputs of Alice and Bob are id_B and id_A respectively. Note that if the MMA protocol has moves over the insecure channel, we may combine the two additional moves with them and thus optimize the total number of moves.

8.2 Prior Work

To compare protocols we focus on the number of messages over the insecure channel and on the length of authenticated messages. Furthermore, a protocol with *two equal* SAS to be sent in both directions (called symmetric SAS) will be considered as better than a protocol with two not necessarily equal SAS to be exchanged. Indeed, some authentication channels may provide symmetric authentication at no extra cost.

8.2.1 A Trivial MMA

A trivial MMA protocol consists in authenticating in the two directions the digest of the input message by using a collision-resistant hash function (CRHF) as depicted in Figure 8.3.

As for the construction in Section 8.1.3, this protocol can be transformed into an MCA protocol by using two additional moves over the insecure channel to exchange m_A and m_B. By authenticating Diffie-Hellman keys, we obtain a 2-move Authenticated Key Agreement (AKA) protocol with symmetric SAS, but the length of the SAS is quite long, typically 160 bits. More details about AKA are given in Chapter 10.

Alice		Bob
input: m_A		**input**: m_B
$\mathsf{SAS}_A \leftarrow \mathsf{H}(m_A)$		$\mathsf{SAS}_B \leftarrow \mathsf{H}(m_B)$
	$\xrightarrow{\mathsf{auth}_{\mathsf{id}_A}(\mathsf{SAS}_A)}$	check $\mathsf{SAS}_A \stackrel{?}{=} \mathsf{SAS}_B$
check $\mathsf{SAS}_B \stackrel{?}{=} \mathsf{SAS}_A$	$\xleftarrow{\mathsf{auth}_{\mathsf{id}_B}(\mathsf{SAS}_B)}$	
output: id_B		**output**: id_A

Figure 8.3. *A Trivial MMA Protocol.*

Another MMA protocol was proposed by Gehrmann-Mitchell-Nyberg [GMN04]. It is used in cases where both devices have a keypad and a simple output. In this protocol, the user is asked to make some computations, as picking a random number or aborting or not the protocol.

8.2.2 The Original SAS-based MCA Protocol: Vau-SAS-MCA

A SAS-based message cross-authentication protocol was proposed by Vaudenay [Vau05b], here called Vau-SAS-MCA. It consists in interleaving two SAS-based unilateral message authentication protocols (depicted in Figure 7.12), one in each direction. It results in a 4-move MCA protocol with symmetric SAS. This may be transformed into a 4-move AKA protocol with symmetric SAS based on the Diffie-Hellman key agreement (see Chapter 10).

Alice		Alice
input: m_A		**input**: m_B
pick $R_A \in_{\mathsf{u}} \{0,1\}^k$		pick $R_B \in_{\mathsf{u}} \{0,1\}^k$
$(c_A, d_A) \leftarrow \mathsf{commit}(0 \Vert m_A, R_A)$	$\xrightarrow{m_A, c_A}$	
	$\xleftarrow{m_B, c_B}$	$(c_B, d_B) \leftarrow \mathsf{commit}(1 \Vert m_B, R_B)$
	$\xrightarrow{d_A}$	$\widehat{R}_A \leftarrow \mathsf{open}(0 \Vert \widehat{m}_A, \widehat{c}_A, \widehat{d}_A)$
$\widehat{R}_B \leftarrow \mathsf{open}(1 \Vert \widehat{m}_B, \widehat{c}_B, \widehat{d}_B)$	$\xleftarrow{d_B}$	
$\mathsf{SAS} \leftarrow R_A \oplus \widehat{R}_B$	$\xrightarrow{\mathsf{auth}_{\mathsf{id}_A}(\mathsf{SAS})}$	$\mathsf{SAS} \stackrel{?}{=} \widehat{R}_A \oplus R_B$
check SAS are the same	$\xleftarrow{\mathsf{auth}_{\mathsf{id}_B}(\mathsf{SAS})}$	
output: $\mathsf{id}_B, \widehat{m}_B$		**output**: $\mathsf{id}_A, \widehat{m}_A$

Figure 8.4. *The Original SAS-based MCA Protocol: Vau-SAS-MCA.*

8.3 An Optimal MMA Protocol: PV-SAS-MMA

The goal is to propose a new MCA protocol improving the number of exchanged messages compared to the protocol of Figure 8.4. Before presenting the final MCA protocol, we start with an MMA protocol. Then, we will extend our MMA protocol to build an optimized MCA protocol. We have published this MMA protocol in [PV06b].

As depicted in Figure 8.5, and without any attack, Alice and Bob start the MMA protocol with the same message, i.e., $m_A = m_B$. Each participant chooses a k-bit random value R_A and R_B, respectively. Alice starts by committing on her random value R_A by sending c, keeping R_A hidden. Bob sends the random value R_B. Then, Alice opens her random R_A by sending the decommit value d. Finally, both authenticate the SAS which has been computed using a simple XOR function.

Alice **input**: m_A		Bob **input**: m_B
pick $R_A \in_{\mathsf{u}} \{0,1\}^k$		pick $R_B \in_{\mathsf{u}} \{0,1\}^k$
$(c,d) \leftarrow \mathsf{commit}(m_A, R_A)$	$\xrightarrow{c}$	
	$\xleftarrow{R_B}$	
	$\xrightarrow{d}$	$\widehat{R}_A \leftarrow \mathsf{open}(m_B, \widehat{c}, \widehat{d})$
$\mathsf{SAS} \leftarrow R_A \oplus \widehat{R}_B$	$\xrightarrow{\mathsf{auth}_{\mathsf{id}_A}(\mathsf{SAS})}$	$\mathsf{SAS} \stackrel{?}{=} \widehat{R}_A \oplus R_B$
check SAS is the same	$\xleftarrow{\mathsf{auth}_{\mathsf{id}_B}(\mathsf{SAS})}$	
output: id_B		**output**: id_A

Figure 8.5. *The New SAS-based MMA Protocol: PV-SAS-MMA.*

Theorem 8.3 (Stand-Alone Security of PV-SAS-MMA).
Let π be the protocol depicted in Figure 8.5. Assume that we have a (T, ε)-secure equivocable commitment scheme in the CRS model.

There exists a (small) constant μ such that the protocol π is $(T - \mu, 2^{-k} + \varepsilon)$-secure in the stand-alone model.

Proof.
Obtaining authenticated SAS. Any adversary which would attack an instance of either Alice or Bob needs one SAS to send her/him so that she/he can complete. This required SAS can easily be obtained from any instance of Alice since she does not need any prior authenticated message. It can also be obtained from any instance of Bob, but Bob should receive another SAS before and in addition, if Bob replies, then the output SAS is equal to the one sent before.

We consider that we have an adversary $\mathcal{A}$ with an instance of Alice, denoted by Alice^* and a target instance, either of Alice or Bob. We assume that the adversary complexity is bounded by $T-\mu$ for some constant overhead μ to be determined by the following reductions. As depicted in Figure 8.6, we consider two cases: attacks targeting an instance of Bob and attacks targeting an instance of Alice. In the case both are targets we arbitrarily designate one as *the* target. Let p_A, resp. p_B, be the probability that the target is an instance of Alice, resp. an instance of Bob. We have $p_A + p_B = 1$. Let q_A, resp. q_B, be the success probability conditioned to both cases, respectively. The overall success probability of $\mathcal{A}$ is $p = q_A p_A + q_B p_B$.

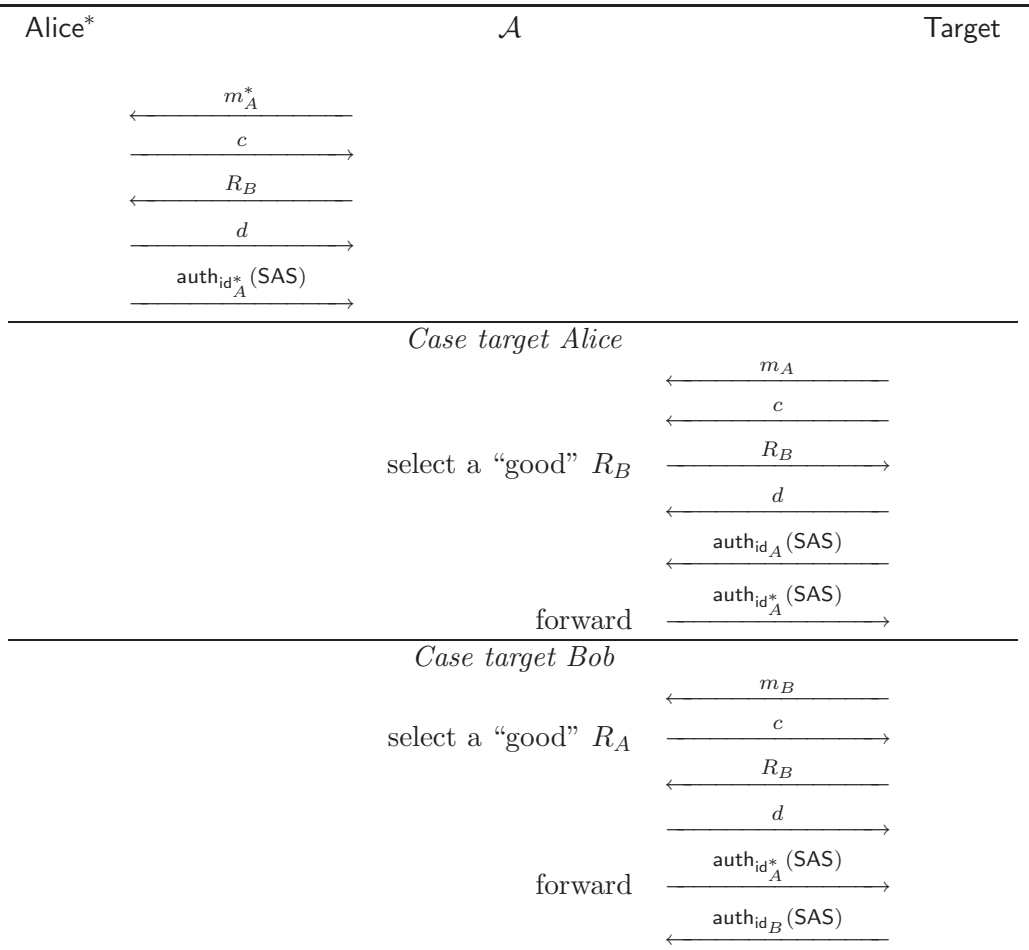


Figure 8.6. Alice^* *and the Target Instance (Alice or Bob).*

Simulation of Alice^***.** In short, we have an adversary $\mathcal{A}$ playing with two instances: the instance of Alice which will give the authenticated SAS denoted by Alice^* and the target instance which may be either an instance of Alice or an instance of Bob. In both cases,

we define a simulator $\mathcal{B}$ that simulates the two instances. $\mathcal{B}$ first needs to simulate the interaction with Alice^*. The simulation works as follows. $\mathcal{B}$ first picks a random k-bit SAS. When Alice^* is launched by the adversary $\mathcal{A}$, we simulate a commitment c by using $\mathsf{simcommit}$. Then, the corresponding $\widehat{R}_B$ is sent to Alice^*, the commit value is equivocated so that it opens to the key $\mathsf{SAS} \oplus \widehat{R}_B$. This simulation of Alice is perfect and has the property to determine the final SAS at the beginning. Note that this instance Alice^* was simulated for some input message m_A^*.

Case of a target Alice. Given the SAS from Alice^*, the adversary $\mathcal{A}$ tries to pass the authentication with the target Alice for some message $m_A \neq m_A^*$. In that case, the adversary $\mathcal{A}$ can be transformed in an adversary playing the hiding game against the commitment scheme. So, algorithm $\mathcal{B}$ plays the role of interface. As depicted in Figure 8.7, $\mathcal{B}$ simulates the target Alice for $\mathcal{A}$ and at the same time $\mathcal{B}$ simulates an adversary playing the hiding game with a challenger $\mathcal{C}$.

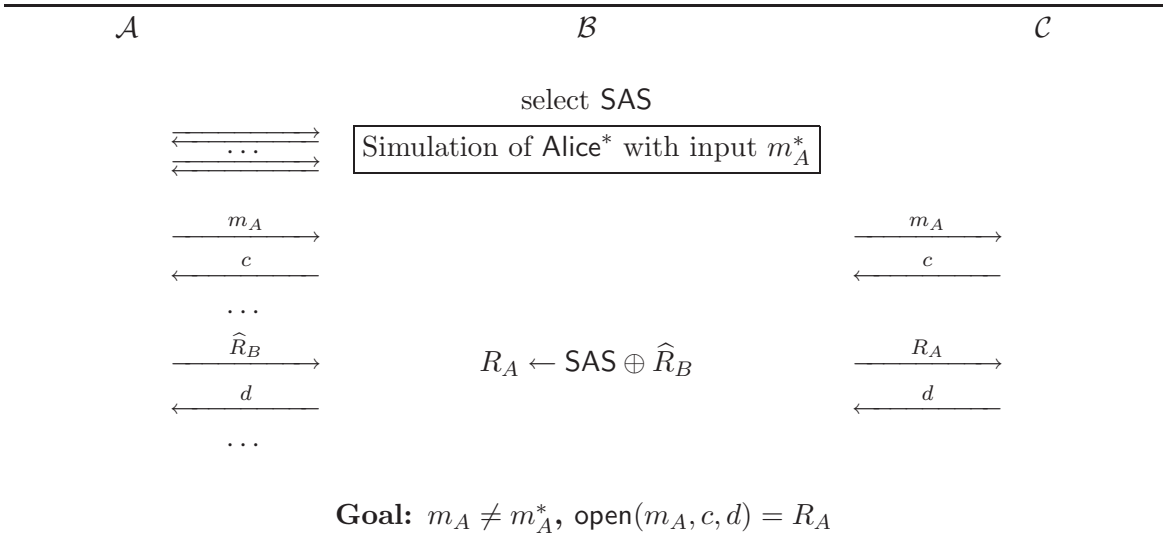


Figure 8.7. *Simulator Playing the Hiding Game (case when the target is Alice).*

In short, the challenger $\mathcal{C}$ will choose a random value R_C and commit on it by sending c to $\mathcal{B}$. The algorithm $\mathcal{B}$ forwards the commit value c to the adversary $\mathcal{A}$. The adversary $\mathcal{A}$ tries to find a random $\widehat{R}_B$ such that $\bar{R}_A \oplus \widehat{R}_B = \mathsf{SAS}$. Note that $\bar{R}_A$ should be the random value from the target Alice, in this case simulated by $\mathcal{B}$, and thus $\bar{R}_A$ is the guessed value by $\mathcal{A}$ given the commit value c on R_C. So, $\mathcal{A}$ will returns $\widehat{R}_B$. $\mathcal{B}$ only needs to remove the SAS from $\widehat{R}_B$ and forwards it to $\mathcal{C}$. So, when $\mathcal{A}$ wins his game, $\mathcal{B}$ wins the hiding game. Since the equivocable commitment scheme is always perfectly hiding, we deduce $q_A = 2^{-k}$.

Case of a target Bob. Given the SAS from Alice^*, the adversary $\mathcal{A}$ tries to pass the authentication with the target Bob for some message $m_A \neq m_A^*$. In that case, the adversary $\mathcal{A}$ can be transformed in an adversary playing the binding game against the commitment

scheme. So, algorithm $\mathcal{B}$ plays the role of interface. As depicted in Figure 8.8, $\mathcal{B}$ simulates the target Bob for $\mathcal{A}$ and at the same time $\mathcal{B}$ simulates an adversary playing the binding game with a challenger $\mathcal{C}$.

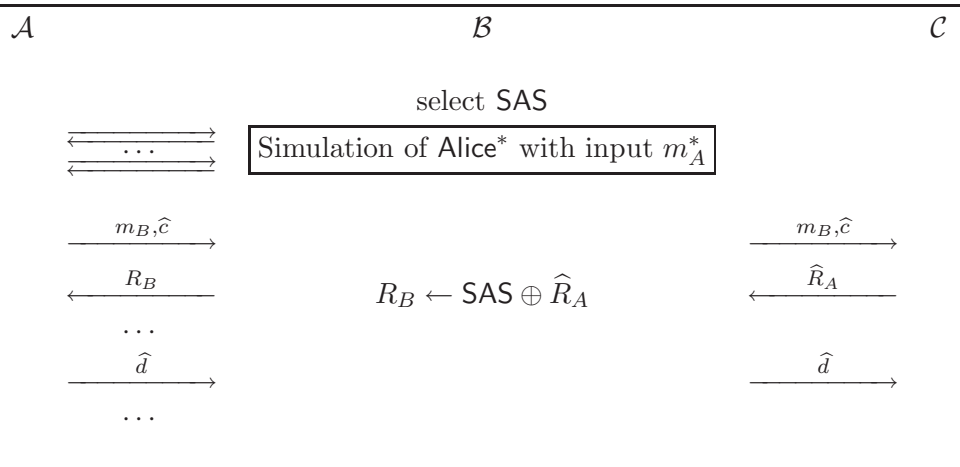


Figure 8.8. *Simulator Playing the Binding Game (case when the target is Bob).*

The idea is the same as in the case with a target Alice. In short, the adversary succeeds if $\widehat{d}$ decommits to a key which leads Bob to the right SAS, thus to the key $\widehat{R}_A$. In that case, we win the binding game with probability $q_B = \varepsilon$.

To summarize, we made an adversary playing either the hiding or either the binding game with overall probability of success $p \leq 2^{-k} + \varepsilon$.

■

Lemma 8.4 (Security of PV-SAS-MMA).
Let π be the protocol depicted in Figure 8.5. Assume that we have a (T, ε)-secure equivocable commitment scheme in the CRS model.

There exists a (small) constant μ such that the protocol π is $\big(T - \mu,\ q_A + q_B,\ q_A(q_A + q_B)(2^{-k} + \varepsilon)\big)$-secure where q_A and q_B denotes the number of instances of Alice and Bob respectively.

Proof.
A successful adversary interaction defines the first attacked instance and a prior sequence initiated by one instance of Alice followed by a chain (possibly empty) of instances of Bob and ended by the attacked instance. Every (non-attacked) instance of Bob in this sequence is sending a SAS identical to the received one to the next instance. Every intermediate instance of Bob terminates with an output message which must be equal to the input message of the previous instance in the sequence (otherwise, they would be successfully attacked). However,

the final instance in the sequence outputs a message which is different from the input of the previous instance. Hence, every instance in the sequence but the final one has the same input message and all instances yield the same SAS. Clearly, sending the output SAS from the leading Alice to the tailing instance produces a successful attack with no intermediate instance of Bob.

Reduction from several instances to one instance of Alice and one target instance. Let $\mathcal{A}_0$ be an adversary who launches at most q_A instances of Alice and q_B instances of Bob. We transform it into an adversary $\mathcal{A}$ who launches an instance of Alice and a single target instance (of either Alice or Bob) as follows:

1. $\mathcal{A}$ first picks two random numbers I, J such that $1 \leq I \leq q_A$ and $1 \leq J < q_A + q_B$.
2. We initialize counters i and j to 0 and run $\mathcal{A}_0$ step by step.
   - Every time $\mathcal{A}_0$ would like to make a launch query to launch an instance of Alice, we increment i. If $i = I$, we really launch it and call the instance π. Otherwise, we increment j and if $j = J$, we really launch it and call the target instance π'. Otherwise, we simulate the oracle call.
   - Every time $\mathcal{A}_0$ would like to make a launch query to launch an instance of Bob, we increment j. If $j = J$, we really launch it and call the target instance π'. Otherwise, we simulate the oracle call.
   - If we have to send a SAS to π, we just simulate the oracle call.
   - If we have to send a SAS to π' and we already got a SAS from π which is equal to the expected one, we just send it. Otherwise, the attack fails.

Due to the previous discussion, if $\mathcal{A}_0$ succeeds, if π' is the first attacked instance for $\mathcal{A}_0$, and if π is the leading instance of Alice in the sequence, then $\mathcal{A}$ succeeds. Hence, the probability of success of $\mathcal{A}$ is at least $\frac{1}{q_A(q_A+q_B-1)}$ times the probability of success p of $\mathcal{A}_0$, i.e.,

$$\mathrm{Adv}_{\pi}^{\mathsf{forge}}(\mathcal{A}) \geq \frac{1}{q_A(q_A + q_B - 1)} \mathrm{Adv}_{\pi}^{\mathsf{forge}}(\mathcal{A}_0)$$

■

8.4 An Optimal MCA Protocol: PV-SAS-MCA

In this section, we propose a new MCA protocol which is more optimized than the one depicted in Figure 8.4 proposed by Vaudenay. Indeed, our proposed MCA protocol, called PV-SAS-MCA improves the number of exchanged messages through the broadband insecure

channel. In addition, we give a formal security proof for it. The proposed MCA protocol is based on the MMA protocol of the previous section and we also published it in [PV06b].

Our protocol uses an almost strongly universal hash function family h (see Definition 3.7). In practice, one can use $h(x, K) = \mathsf{trunc}(\mathsf{hash}(K\|x))$ where hash is a collision-resistant hash function (CRHF) and trunc truncates to the leading ρ bits.

As depicted in Figure 8.9, Alice and Bob start the protocol with input m_A and m_B respectively. Both Alice and Bob pick a random key denoted by K and R respectively. Then Alice uses a commitment scheme to commit on her key K and to temporarily keep it hidden. She sends m_A and c to Bob, who replies with m_B and R. Now, all random variables are fixed and Alice can reveal her hidden K by sending d. Finally, they authenticate a string $\mathsf{SAS} = R \oplus h(m_B, K)$.

Note that m_B is directly authenticated through the SAS while m_A is indirectly authenticated. Indeed, m_A is (indirectly) authenticated thanks to the tagged commitment scheme which binds m_A to K and thanks to the direct authentication of K.

Note that in the previous MMA protocol the key K was denoted by R_A. Both R_A in the MMA and K in the MCA are picked randomly in a binary set. We replaced R_A by K because now in the MCA protocol it is used as key in the hash function. Unlike the previous MMA protocol, the committed key K can now be much larger since its length may be now different from the SAS length.

Note that we added an identity test on Alice's side to avoid trivial reflection attacks.

Alice **input**: m_A		Bob **input**: m_B
pick $K \in_{\mathsf{u}} \{0,1\}^{\kappa}$		pick $R \in_{\mathsf{u}} \{0,1\}^{\rho}$
$(c, d) \leftarrow \mathsf{commit}(m_A, K)$	$\xrightarrow{m_A, c}$	
	$\xleftarrow{m_B, R}$	
	$\xrightarrow{d}$	$\widehat{K} \leftarrow \mathsf{open}(\widehat{m}_A, \widehat{c}, \widehat{d})$
$\mathsf{SAS} \leftarrow \widehat{R} \oplus h(\widehat{m}_B, K)$	$\xrightarrow{\mathsf{auth}_{\mathsf{id}_A}(\mathsf{SAS})}$	$\mathsf{SAS} \stackrel{?}{=} R \oplus h(m_B, \widehat{K})$
check SAS is the same check $\mathsf{id}_A \neq \mathsf{id}_B$	$\xleftarrow{\mathsf{auth}_{\mathsf{id}_B}(\mathsf{SAS})}$	
output: $\mathsf{id}_B, \widehat{m}_B$		**output**: $\mathsf{id}_A, \widehat{m}_A$

Figure 8.9. *The New SAS-based MCA Protocol: PV-SAS-MCA.*

For a moment, assume that m_A is fixed, i.e., we have a unilateral message authentication protocol for m_B. Then the usage of one-time pad $h(\widehat{m}_B, K) \oplus \widehat{R}$ assures that the adversary

cannot succeed unless he transfers the commitment $\widehat{c}$ before the decommitment d is released. But if the adversary just forwards c, then we arrive at the standard XOR-universality game, where the adversary must find $m_B \neq \widehat{m}_B$ such that $h(m_B, K) \oplus h(\widehat{m}_B, K) = R \oplus \widehat{R}$ for an unknown key $K \in_{\mathsf{u}} \{0,1\}^\kappa$. Alternatively, the adversary can try to alter the non-malleable commitment but this is guaranteed to fail. So, for the same reason, the message m_A is also guaranteed to reach Bob without modifications.

If we do require the two SAS to be independent, we can replace K and R by two vectors $K = (K_A, K_B)$ and $R = (R_A, R_B)$, leading to two independent strings $\mathsf{SAS}_A = \widehat{R}_A \oplus h_{K_A}(\widehat{m}_B)$ and $\mathsf{SAS}_B = \widehat{R}_B \oplus h_{K_B}(\widehat{m}_B)$ to be exchanged and checked. We can easily check that the result below also applies to this variant.

Theorem 8.5 (Stand-Alone Security of PV-SAS-MCA).
Consider the protocol π depicted in Figure 8.9. Let h be an ε_{asu}-almost strongly universal hash function. Let ℓ_e, ℓ_c be the parameters of the equivocable commitment scheme of Section 3.5.9.2 based on a random oracle H *bounded by t queries. Let $\varepsilon = t^2 2^{-\ell_e} + t^2 2^{-\ell_c}$.*

There exists a (small) constant μ such that π is $(T - \mu,\ 2^{-\rho} + \varepsilon + \varepsilon_{\text{asu}})$-secure in the stand-alone model.

Hence, this bound is essentially tight and so our protocol is essentially *optimal*.

Proof.
Let h be an ε_{asu}-almost strongly universal hash function family[1] with ρ-bit digests, i.e., $\forall a, b, \alpha, \beta : \Pr[k \in_{\mathsf{u}} \mathcal{K} : h(a,k) = \alpha, h(b,k) = \beta] \leq 2^{-\rho}\varepsilon_{\text{asu}} + 2^{-2\rho}$. We can replace this condition by the following two properties:

- h is ε_{ar}-almost regular with $\varepsilon_{\text{ar}} = 2^{-\rho} + \varepsilon_{\text{asu}}$, see Definition 3.3.
- h is ε_{axu}-almost XOR-universal with $\varepsilon_{\text{axu}} = 2^{-\rho} + \varepsilon_{\text{asu}}$, see Definition 3.9.

We define a new character, the *flipped Bob*, who proceeds as Bob but first issues a SAS equal to $R \oplus h(m_B, \widehat{K})$ then receives a SAS for verification. In a new protocol, Alice and the flipped Bob can interact with two crossing SAS exchanges.

The interaction of the adversary with an instance of Alice consists of two steps, i.e., A_1 and A_2, and the interaction with an instance of Bob as well, i.e., B_1 and B_2:

A_1 sending to Alice her message m_A (for the launch query) and getting her commit value c (for the first send query).

A_2 giving her Bob's alleged message $\widehat{m}_B$ and random value $\widehat{R}$ and getting her decommit value d.

[1] Note that this definition of almost strongly universal hashing is slightly different from [Sti91, Sti94] in the sense that perfect regularity is not required.

B_1 sending him his message m_B (for the launch query) and Alice's alleged message $\widehat{m}_A$ and commit value $\widehat{c}$ and getting his random value R (for the first send query).

B_2 giving him Alice's alleged decommit value $\widehat{d}$.

The second step must be performed after the first one, i.e., for Alice first A_1 and then A_2 and for Bob first B_1 and then B_2. Alice's SAS will be equal to $\widehat{R} \oplus h(\widehat{m}_B, K)$ where K is the result of $\mathsf{open}(m_A, c, d)$ while the Bob's SAS will be equal to $R \oplus h(m_B, \widehat{K})$ where $\widehat{K}$ is the result of $\mathsf{open}(\widehat{m}_A, \widehat{c}, \widehat{d})$.

The adversary is *successful* if the two instances complete and compute the same SAS and if the input message of at least one instance is different from the output message of the other instance.

Note that the sending instance and the target instance must be different. Indeed, no instance of Bob can send a SAS to himself otherwise it would have to be received before being sent. Similarly, no instance of Alice can accept a SAS coming from herself.

In what follows we show that all cases can be simulated so that we can win a hard game, proving that the probability of success is at most $2^{-\rho} + \varepsilon + \varepsilon_{\text{asu}}$).

On Bob's incoming $\widehat{c}$ (Step B_1). In the random oracle equivocable commitment model, we only consider the event where no collision occurred. Hence, a commit value $\widehat{c}$ issued by the adversary for an instance of Bob is either a real output by H and can only be opened in a single way, or no output from H. In the latter case, we can consider $(\perp, \perp, \perp, \widehat{c})$ as a new entry in the H list and count it as an extra oracle call. This way, $\widehat{c}$ can never be opened. Hence, with probability at least $1-(t+1)(t+2)2^{-\ell_e-1}-(t+1)(t+2)2^{-\ell_c-1}$, which is larger than $1-\varepsilon$, the commit value(s) $\widehat{c}$ from the adversary are either openable in a single fixed way or not openable. If they are not openable, the adversary fails. If openable $\widehat{c}$ are issued by an oracle call to H by the adversary, we can thus virtually replace the adversary release of $\widehat{c}$ by an adversary release of $\widehat{K}$ and step B_2 can be ignored. If openable $\widehat{c}$ are issued by other oracle calls to H, it can only be by a simulation of Alice, leading us to $c = \widehat{c}$, thus $\widehat{K} = K$ and $m_A = \widehat{m}_A$.

Cases Alice-Alice. We denote #2 the instance of Alice whose A_2 step is the last. Since the commitment is perfectly hiding, Alice leaks no information about $K^{(2)}$ (variable K for Alice #2) until this very last step. Hence, $K^{(2)}$ is independent from the rest and $\widehat{R}^{(1)} \oplus \widehat{R}^{(2)} \oplus h(\widehat{m}_B^{(1)}, K^{(1)}) = h(\widehat{m}_B^{(2)}, K^{(2)})$ holds with probability at most ε_{ar}.

Cases Bob-Bob. We denote #2 the instance of Bob whose B_1 step is the last one. Those cases produce no oracle calls to H by Alice, so $\widehat{K}^{(1)}$ and $\widehat{K}^{(2)}$ are selected by the adversary

before the $B_1^{(2)}$ step. Note that $R^{(1)}$ is already released. The attack succeeds if $R^{(2)} = R^{(1)} \oplus h(m_B^{(1)}, \widehat{K}^{(1)}) \oplus h(m_B^{(2)}, \widehat{K}^{(2)})$ where $R^{(2)}$ is independent of the right hand term and selected at random by the second Bob. Clearly, this succeeds with probability $2^{-\rho}$.

Cases Alice-Bob. Without loss of generality, we can assume that B_2 is the last step.

In the case $A_1A_2B_1B_2$, R is selected in step B_1 so the adversary succeeds with probability $2^{-\rho}$.

In the case $A_1B_1A_2B_2$ with $c \neq \widehat{c}$ or in the case $B_1A_1A_2B_2$ (necessarily with $c \neq \widehat{c}$), the adversary has no information about K until step A_2 and succeeds when $\widehat{R} \oplus R \oplus h(m_B, \widehat{K}) = h(\widehat{m}_B, K)$. Hence the adversary succeeds with probability at most $\varepsilon_{\mathrm{ar}}$.

In the case $A_1B_1A_2B_2$ with $c = \widehat{c}$, we must have $m_A = \widehat{m}_A$. This can only be an attack for $m_B \neq \widehat{m}_B$. The adversary has no information about K until step A_2 and succeeds when $\widehat{R} \oplus R = h(m_B, K) \oplus h(\widehat{m}_B, K)$, hence with probability at most $\varepsilon_{\mathrm{axu}}$.

■

With the same proof idea, we can give a proof in the standard model (or CRS model) instead of in the random oracle model. For that, we do not longer consider an instantiation of the commitment scheme, but we consider a commitment scheme with generic parameters. This leads to the following theorem.

Theorem 8.6 (Stand-Alone Security of Generic PV-SAS-MCA).
Consider the protocol π depicted in Figure 8.9. Let h be an $\varepsilon_{\mathrm{ar}}$-almost regular and $\varepsilon_{\mathrm{axu}}$-almost XOR-universal hash function. Let the commitment scheme be $(T, \varepsilon_{\mathrm{h}})$-hiding, $(2T, \varepsilon_{\mathrm{b}})$-binding and $(T, \varepsilon_{\mathrm{nm}})$-non-malleable

There exists a (small) constant μ such that the protocol π is $(T - \mu,\ \max\{\varepsilon_{\mathrm{ar}}, \varepsilon_{\mathrm{axu}}\} + \varepsilon_{\mathrm{h}} + \varepsilon_{\mathrm{b}} + \sqrt{\varepsilon_{\mathrm{b}}} + \varepsilon_{\mathrm{nm}})$-secure in the stand-alone model.

See the article [LP09] for the formal proof.

In Theorem 8.6, we used an equivocable commitment scheme. Such a commitment is perfectly hiding and $\varepsilon_{\mathrm{b}} = 2^{-\rho}$. It remains to quantify $\varepsilon_{\mathrm{b}} + \sqrt{\varepsilon_{\mathrm{b}}} + \varepsilon_{\mathrm{nm}}$ which was essentially hidden in the ε. In conclusion, if we compare Theorem 8.5 with Theorem 8.6, then we see that they claim the same security level.

Lemma 8.7 (Security of PV-SAS-MCA).
Consider the protocol π depicted in Figure 8.9. Let h be an $\varepsilon_{\mathrm{asu}}$-almost strongly universal hash function. Let ℓ_e, ℓ_c be the parameters of the equivocable commitment scheme of Section 3.5.9.2 based on a random oracle H bounded by t queries. Let $\varepsilon = t^2 2^{-\ell_e} + t^2 2^{-\ell_c}$.

There exists a (small) constant μ such that the protocol π is $(T - \mu,\ q_A + q_B,\ q_A(q_A -$

$1+q_B)(2^{-\rho}+\varepsilon+\varepsilon_{\text{asu}})$)*-secure where* q_A *and* q_B *denotes the number of instances of Alice and Bob respectively.*

By launching q instances of either Alice or Bob with pairwise different input messages and by picking independent uniformly distributed $\widehat{R}$, all SAS are independent and uniformly distributed so we have one matching with probability $1-2^{-q\rho}\cdot 2^{\rho}!/(2^{\rho}-q)!$ which is roughly $\frac{q(q-1)}{2}2^{-\rho}$ when $q \ll 2^{\frac{\rho}{2}}$.

Clearly, launching q protocol instances and targeting a success probability p, we should choose $\rho \approx \log_2 \frac{q^2}{2p}$. With the same analysis as in [Vau05b], in a network of N participants, each limited to R runs of the protocol, and a maximal attack probability *at most* p, we should use $\rho \approx \log_2 \frac{N^2R^2}{2p}$. When p is the probability to attack a target node, we should use $\rho \approx \log_2 \frac{NR^2}{2p}$. Considering a real scenario on a huge network, i.e., with $N \approx 2^{20}$, $q \approx 2^{10}$, and $p \approx 2^{-10}$, we obtain $\rho \approx 49$ bits. In an ATM-like environment, we can take $N = 2$, $R = 3$, and $p = 3 \cdot 10^{-4}$, leading us to $\rho \approx 15$ bits. In between, we believe that $\rho \approx 20$ bits provides enough security in a small community of human users.

Proof.
We consider an adversary successfully running his attack with many instances of the original MCA protocol.

We say that a given instance is attacked if it completed the protocol during which a SAS was received, with an output which is not consistent with the input of the instance who issued the received SAS. An attacked instance, i.e., the target (either Alice or Bob) must receive one SAS from a sending instance. Clearly, both instances must agree on the SAS to complete. Hence, if the SAS sent by the target instance is forwarded to the sending instance then both instances fully interact.

As in proof of Lemma 8.4, we can reduce an adversary $\mathcal{A}$ launching q_A instances of Alice and q_B instances of Bob into an adversary $\mathcal{A}_0$ launching one instance of each. Due to the identity check, we avoid reflection attacks and thus there is only $q_A(q_A - 1 + q_B)$ possible matching pairs.

We can guess the pair of instances with probability $\frac{1}{q_A(q_A-1+q_B)}$. Hence, we can simulate all instances except the two guessed ones. Since the SAS verification phase is the last step on both instances, there is no trouble to make the two instances exchange their SAS. We thus transform the initial adversary $\mathcal{A}$ with success probability p into an adversary $\mathcal{A}_0$ with success probability at least $\frac{p}{q_A(q_A-1+q_B)}$.

■

8.5 Following Works

Almost simultaneously to the publication of our proposed MCA protocol depicted in Figure 8.9, Laur and Nyberg [LN06a] proposed another one, called MANA IV and depicted in Figure 8.10, with many similar aspects:

- Both protocols send a message digest $h(m, K)$ over the insecure channel instead of the hash key K. Since the hash key can be arbitrarily long, one can bypass Simmons's bounds and achieve the optimal deception probability.
- Nevertheless, we still have to guarantee that the adversary has no access to the key K before both parties have acquired the common output. Such structural restrictions are enforced by clever use of a commitment scheme. So, both protocols of Figures 8.9 and 8.10 uses commitments to temporarily hide hash key(s).

But differently from the SAS protocol family, in MANA IV the message pair (m_A, m_B) is directly authenticated with the hash function.

Alice **input**: m_A		Bob **input**: m_B
$r_A \in_{\mathsf{u}} \{0,1\}^\kappa$		$r_B \in_{\mathsf{u}} \{0,1\}^\rho$
$(c, d) \leftarrow \mathsf{commit}(\mathsf{crs}, r_A)$	$\xrightarrow{m_A, c}$	
	$\xleftarrow{m_B, r_B}$	
	$\xrightarrow{d}$	$\widehat{r}_A \leftarrow \mathsf{open}(\mathsf{crs}, \widehat{c}, \widehat{d})$
$\mathsf{SAS}_A \leftarrow h(m_A \Vert \widehat{m}_B, r_A, \widehat{r}_B)$	$\xrightarrow{\mathsf{auth}_{\mathsf{id}_A}(\mathsf{SAS}_A)}$	
	$\xleftarrow{\mathsf{auth}_{\mathsf{id}_B}(\mathsf{SAS}_B)}$	$\mathsf{SAS}_B \leftarrow h(\widehat{m}_A \Vert m_B, \widehat{r}_A, r_B)$
output: $\mathsf{id}_B, \widehat{m}_B$		**output**: $\mathsf{id}_A, \widehat{m}_A$

Figure 8.10. *The MANA IV Protocol.*

Again, the protocol structure guarantees that the adversary cannot succeed if messages are transferred abnormally, i.e., $\widehat{m}_B, \widehat{r}_B$ arrive before $\widehat{m}_A, \widehat{c}$ or $\widehat{d}$ is received before $\widehat{m}_B, \widehat{r}_B$. Now for the normal runs, the adversary has to fix messages $\widehat{m}_A, \widehat{m}_B$ before both sub-keys r_A and r_B become public. As a result, information theoretical properties of the hash function are sufficient to guarantee authenticity.

Theorem 8.8 (Stand-Alone Security of MANA IV).
Consider the protocol π depicted in Figure 8.10. Let h be an hash function which is $\varepsilon_{\mathrm{ar}}$-almost regular w.r.t. sub-keys and $\varepsilon_{\mathrm{au}}$-almost universal w.r.t. the sub-key r_A. Let the commitment scheme be $(T, \varepsilon_{\mathrm{h}})$-hiding, $(2T, \varepsilon_{\mathrm{b}})$-binding, and $(T, \varepsilon_{\mathrm{nm}})$-non-malleable.

There exists a (small) constant μ such that the protocol π with ℓ-bit SAS is $(T - \mu,\ \max\{\varepsilon_{\mathrm{ar}}, \varepsilon_{\mathrm{au}}\} + 2\varepsilon_{\mathrm{h}} + \varepsilon_{\mathrm{b}} + \sqrt{\varepsilon_{\mathrm{b}}} + \varepsilon_{\mathrm{nm}})$-secure in the stand-alone model.

See the article [LN06a] for the formal proof.

Finally, observe that protocols with an optimal deception bound utilize similar techniques. First, all of them use one-time pad encryption to assure that the adversary preserves the temporal order between protocol messages. Secondly, the commitment scheme is used as an additional measure against substitution attacks. Thirdly, it seems that there are no other designs patterns that could overcome the shortcomings of the MANA I protocol.

8.6 Applications

Applications of such protocols can be traditional key agreement, but run in an *ad-hoc* way. For instance, it can be used to exchange PGP public keys to be authenticated by a human-to-human telephone conversation. It can also be used to secure peer-to-peer Voice over IP communications as detailed in Section 10.7. Other straightforward applications can be the Bluetooth-like establishment of symmetric keys between associated wireless devices, e.g., for wireless USB.

Chapter

NINE

Group Message Authentication

We start this chapter by defining the notion of group message authentication. Then, we study the only prior SAS-based proposal, i.e., Group-MANA IV proposed by Valkonen, Asokan, and Nyberg [VAN06]. This protocol is not optimal with respect to the number of moves and in addition it requires the election of a group leader. In Section 9.3 we propose an *optimal* group message authentication protocol, called LP-SAS-GMA and published in [LP08]. We emphasize that there is no need of group leader in our protocol. Finally, in Section 9.4, we discuss possible applications.

9.1 Group Message Authentication Primitive

The methodology presented in the previous chapters can be naturally extended to a group setting. However, there are some important differences. First, it is much more difficult to assure proper temporal order for send and receive events, since there are more events to be synchronized. Secondly, the set of participants might be determined *dynamically* during the protocol execution based on participation. Hence, we must authenticate also the group description.

We consider n parties involved in the protocol: each party $\mathcal{P}_i$ is located in node i and has identity id_i. The n parties form a group $\mathcal{G} = \{\mathsf{id}_1, \ldots, \mathsf{id}_n\}$.

There are several important aspects to note:

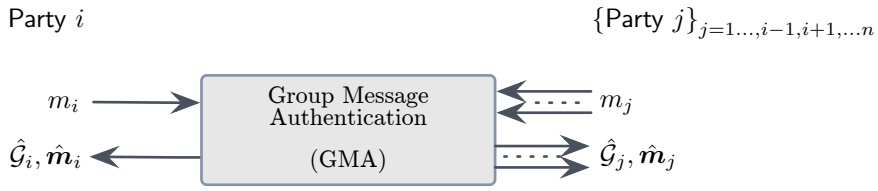


Figure 9.1. *Group Message Authentication (GMA).*

First, as said before, a group may be dynamically formed based on the participation in a group message authentication (GMA) protocol, for example fast setup of *ad-hoc* military networks. But then an adversary can always split the group into several subgroups and block the traffic between the subgroups. As a result, each subgroup agrees on a different output. Such attacks cannot be defeated unless parties know the description of $\mathcal{G}$ in advance, i.e., there is some authenticated way to broadcast $\mathcal{G}$.
Second, an adversary may set up several dummy network nodes in order to corrupt communication or secretly shuffle different groups. Thus, we consider a scenario where a subset $\mathcal{G}$ of all network nodes wants to establish a common message $\vec{m}$. At the end of the protocol, either all participants halt or each $\mathcal{P}_{\mathsf{id}}$, $\mathsf{id} \in \mathcal{G}$ obtains values $\widehat{\mathcal{G}}_{\mathsf{id}}$ and $\widehat{\vec{m}}_{\mathsf{id}}$. We allow adaptive malicious corruption[1] of group participants, i.e., at any time during the protocol execution $\mathcal{A}$ can take total control over any node $\mathcal{P}_{\mathsf{id}}$.

Definition 9.1 (Group Message Authentication).
A Group Message Authentication *(GMA) protocol with an n-party group $\mathcal{G} = \{\mathsf{id}_1, \ldots, \mathsf{id}_n\}$ works as follows: each participant $\mathcal{P}_{\mathsf{id}_i}$ starts with input m_i and ends with output $(\widehat{\mathcal{G}}_i, \widehat{\vec{m}}_i)$ where $\widehat{\vec{m}}_i = (\widehat{m}_{i,1}, \ldots, \widehat{m}_{i,n})$. Let $\mathcal{H} \subseteq \mathcal{G}$ denote the group of uncorrupted participants at the end of the protocol.*

An honest run should lead to the same group representation and the same output messages for all (uncorrupted) participants, i.e., $\forall \mathsf{id}_i, \mathsf{id}_j \in \mathcal{H} : (\widehat{\mathcal{G}}_i, \widehat{\vec{m}}_i) = (\widehat{\mathcal{G}}_j, \widehat{\vec{m}}_j)$.

An adversary is successful if there exists a pair of uncorrupted participants $\mathcal{P}_{\mathsf{id}_i}, \mathcal{P}_{\mathsf{id}_j}$ such that $(\widehat{\mathcal{G}}_i, \widehat{\vec{m}}_i) \neq (\widehat{\mathcal{G}}_j, \widehat{\vec{m}}_j)$.

9.2 Prior Work

Note that an *authenticated broadcast* primitive is sufficient for group message authentication. Namely, participant $\mathcal{P}_{\mathsf{id}_i}$ can first send all messages m_i to a leader $\mathcal{P}_{\mathsf{id}_*}$ who then uses the authentic broadcast primitive to transfer the gathered input $\vec{m}_* = (m_i)_{i \in \mathcal{G}}$ and the group

[1] In many cases, adaptive corruption is impossible, but with our new protocol being secure against adaptive corruption, it makes no sense to consider weaker models.

description $\mathcal{G}$ to all participants. Next, all other participants $\mathcal{P}_{\mathsf{id}_j}$, $\mathsf{id}_j \in \mathcal{G}$ verify that the received message $\widehat{\vec{m}}_*$ is consistent with their input m_j. The protocol is halted if a complaint is raised over the insecure channels.

The authenticated broadcast primitive itself can be achieved running several unilateral authentication protocols in parallel. However, the latter automatically increases the number of different messages over the insecure channels.

Alternatively, we can design specific protocols, where the same insecure message is transferred to all group members. Although this does not formally decrease the amount of insecure communication, it makes the protocol more user friendly.

9.2.1 Group-MANA IV

The first specific SAS-based group message authentication protocol was sketched by Valkonen, Asokan, and Nyberg [VAN06]. The corresponding protocol, called Group-MANA IV protocol, is a simple extension of the MANA IV protocol (see Figure 8.10).

Let $\vec{m}$ be the data negotiated between the group of participants $\mathcal{G}$. To ensure that each participant $\mathcal{P}_{\mathsf{id}_i}$ shares the same data $\widehat{\vec{m}}_i$, they should authenticate it. Note that Group-MANA IV is in reality a group message mutual-authentication protocol (and not a group message cross-authentication protocol). However, it can be transformed with a similar technique as presented in Section 8.1.3.

To start the protocol, one participant, say $\mathcal{P}_{\mathsf{id}_1}$, is elected as protocol leader. By consequent, the protocol steps are different for the leader than for the other participants. So, Figure 9.2 represents the execution of the leader $\mathcal{P}_{\mathsf{id}_1}$ and an arbitrary $\mathcal{P}_{\mathsf{id}_i}$ with $i \neq 1$.

The corresponding security proof is rather technical and gives no additional insight. For more details, see [VAN06].

Our SAS-GMA protocol, presented in the following section and depicted in Figure 9.3, is more round efficient. Remember that in order to transform the protocol of Figure 9.2 into a group MCA protocol, we should add at least one round over the insecure channel. So, the gap efficiency between the protocols of Figures 9.3 and 9.2 is even greater than in a first look.

9.3 An Optimal GMA Protocol: LP-SAS-GMA

This new group message authentication protocol, called LP-SAS-GMA, borrows the ideas from the Vau-SAS-MCA, depicted in Figure 8.4, and MANA IV, depicted in Figure 8.10. Both aforementioned protocols use commitment schemes to temporarily hide certain keys.

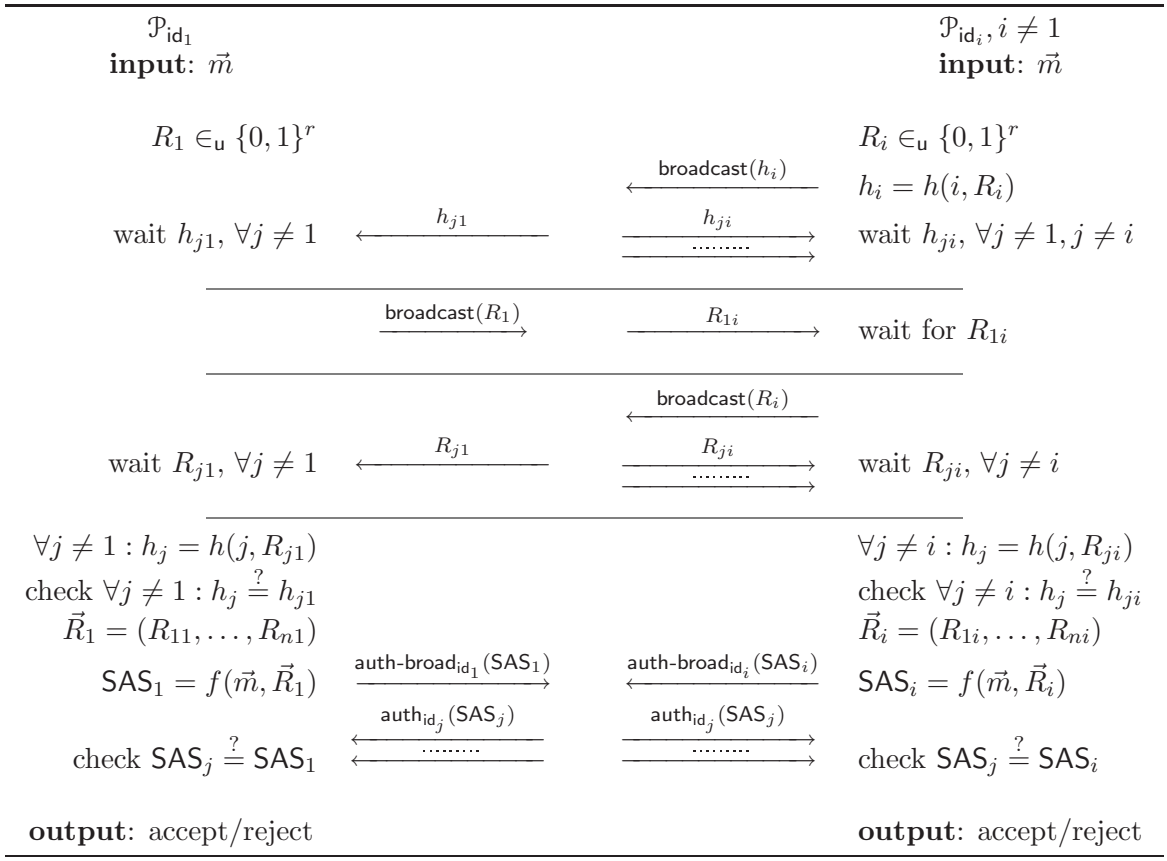


Figure 9.2. *The Group-MANA IV Protocol.*

Similarly to Vau-SAS-MCA, all sub-keys in our protocol are released after the adversary has delivered all messages. And similarly to MANA IV, messages m_i are sent in clear and authenticated test values are ℓ-bit hash codes.

Since LP-SAS-GMA is symmetric, Figure 9.3 only specifies the behavior of a single party $\mathcal{P}_{\mathsf{id}_i}$ who wants to participate in the protocol. Here $\widehat{\mathcal{G}}_i$ denotes the group of participants who joined $\mathcal{P}_{\mathsf{id}_i}$ during the first round before the timeout. Of course, if the group $\widehat{\mathcal{G}}_i$ is known beforehand then $\mathcal{P}_{\mathsf{id}_i}$ can wait until all other group members have sent their first messages.

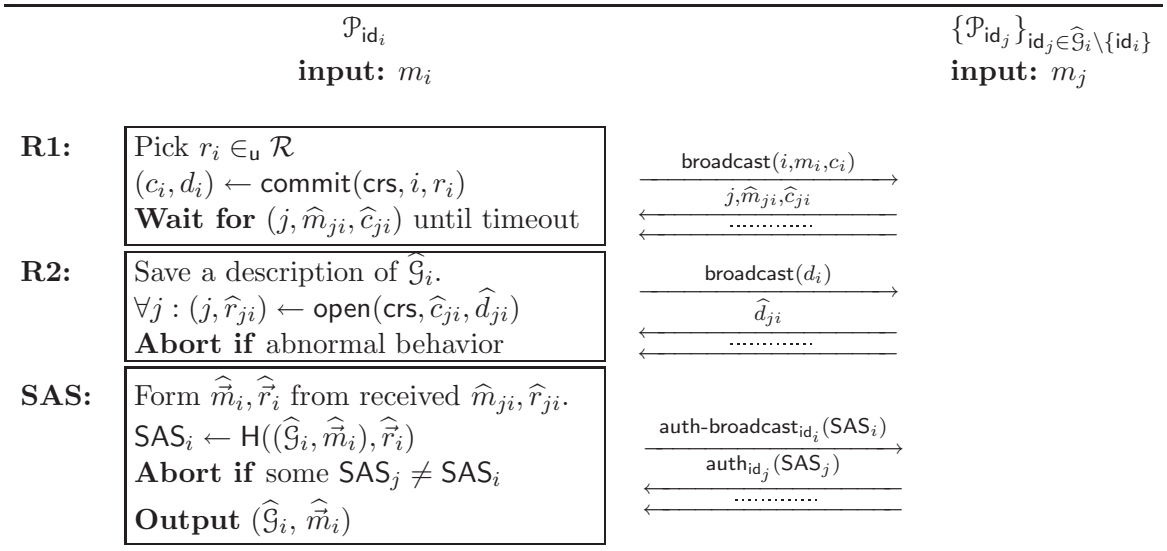


Figure 9.3. *The New SAS-based GMA Protocol: LP-SAS-GMA.*

For clarity, variables $\widehat{m}_{ji}, \widehat{c}_{ji}, \widehat{d}_{ji}$ denote the values from $\mathcal{P}_{\mathsf{id}_j}$ that are received by $\mathcal{P}_{\mathsf{id}_i}$. The hats still indicate a possible modification by an adversary. As for the group descriptions, the output vector $\vec{\widehat{m}}_i = \{\forall j : \widehat{m}_{ji}\}$ and the sub-key vector $\vec{\widehat{r}}_i = \{\forall j : \widehat{r}_{ji}\}$ are ordered with respect to sender identities. To be exact, $\widehat{m}_{ii} = m_i$, $\widehat{r}_{ii} = r_i$ and j ranges over $\widehat{\mathcal{G}}_i$. Also note that (i, r_i) and $(\widehat{\mathcal{G}}_i, \vec{\widehat{m}}_i)$ are shorthands for binary strings that uniquely encode the corresponding elements.

Implementation Details

The cryptographic requirements for the hash function H and the commitment scheme $\mathcal{C}om$ are formally specified by Theorem 9.2, but there are many other minor details that are not covered by Figure 9.3.

Assume that the final output $(\widehat{\mathcal{G}}_i, \vec{\widehat{m}}_i)$ can always be encoded as an s-bit string. Then, the hash function $\mathsf{H} : \{0,1\}^s \times \mathcal{R}^* \to \mathcal{T}$ must support variable number of sub-keys r_j, since

the size of the group is variable. For example, we can use a single keyed hash function H_1 and some sort of secure combiner to derive a new master key from sub-keys. The restriction $(\widehat{\mathcal{G}}_i, \widehat{\vec{m}}_i) \in \{0,1\}^s$ is not limiting in practice. For instance, we can use a collision resistant hash function (CRHF) like SHA-256 to compress an encoding of any length to a 256-bit string.

We will need a hash function that is ε_{au}-almost universal and ε_{ar}-almost regular and could handle variable number of sub-keys at the same time. *A priori*, it is not clear that such hash functions exist. Therefore, we give one possible explicit construction. Let all sub-keys be from $\{0,1\}^{2s}$ and messages from $\{0,1\}^s$ for a certain integer s which bound the message space. To hash a message x, we first compute an intermediate key $a \leftarrow r_1 \oplus \cdots \oplus r_n$; split a into two halves a_1, a_2; interpret x, a_1, a_2 as elements of the Galois field $\mathsf{GF}(2^s)$ and define $h(x, r_1, \ldots, r_n) = a_1 x + a_2$ over $\mathsf{GF}(2^s)$. Considering two different inputs x_0 and x_1, it is straightforward to verify that a pair $h(x_0, \vec{r}), h(x_1, \widehat{\vec{r}})$ is uniformly distributed over $\{0,1\}^{2s}$ in the universality experiment. To get shorter hash values, we can output ℓ lowest bits. Then the hash function has optimal bounds $\varepsilon_{\text{ar}} = \varepsilon_{\text{au}} = 2^{-\ell}$.

Secondly, we assume that the description of H and the public parameters of *Com* are fixed and distributed by a trusted authority.

Thirdly, we assume that a participant $\mathcal{P}_{\mathsf{id}_i}$ halts if there is any hint of an attack:

1. some group member halts,
2. there are duplicates $(j, \widehat{m}_{ji}, \widehat{c}_{ji}) \neq (j, \widehat{m}'_{ji}, \widehat{c}'_{ji})$,
3. a sub-key has an invalid form $(j, \cdot) \neq \mathsf{open}(K_p, \widehat{c}_{ji}, \widehat{d}_{ji})$,
4. some SAS messages do not match.

Another important aspect is secure comparison of SAS messages. In principle, it is sufficient to deliver a minimal amount of messages so that the participants can detect $\mathsf{SAS}_\alpha \neq \mathsf{SAS}_\beta$ for $\alpha, \beta \in \mathcal{G}$, where $\mathcal{G}$ is the set of all active participants of the protocol. If it is possible to detect all these active nodes, then a single node can broadcast the SAS message so that the remaining nodes can compare it to their SAS message. For many applications such as securing conference calls over VoIP, forming Bluetooth piconets and other wireless device networks, the group is known in advance. Hence, broadcast of a single SAS message is a viable option. Also note that the group formation can be combined with node detection in the Bluetooth networks and thus the timeout effect is marginal.

Stand-Alone Security

The security proof for LP-SAS-GMA is straightforward but quite technical. Hence, we present the proof of Theorem 9.2 in smaller chunks to make it more comprehensible. Note

that the security level depends linearly on $|\mathcal{G}|$ but the constant term $\max\{\varepsilon_{\mathrm{au}}, \varepsilon_{\mathrm{ar}}\} \approx 1/|\mathcal{T}| \approx 2^{-\ell}$ dominates over the term $n \cdot \varepsilon_{\mathrm{nm}} + \varepsilon_{\mathrm{b}}$. Therefore, the deception probability asymptotically approaches the theoretical lower bound $2^{-\ell}$.

Theorem 9.2 (Stand-Alone Security of LP-SAS-GMA).
Consider the protocol π depicted on Figure 9.3. Let n be the maximal size of the group $\mathcal{G}$. Let H be a multi-keyed hash function which is $\varepsilon_{\mathrm{au}}$-almost universal with respect to each sub-key pair and $\varepsilon_{\mathrm{ar}}$-almost regular with respect to each sub-key. Let the commitment scheme be $(T + \mu, \varepsilon_{\mathrm{b}})$-binding and $(T + \mu, \varepsilon_{\mathrm{nm}})$-non-malleable

There exists a (small) constant μ such that π is $(T, n \cdot \varepsilon_{\mathrm{nm}} + \varepsilon_{\mathrm{b}} + \max\{\varepsilon_{\mathrm{au}}, \varepsilon_{\mathrm{ar}}\})$-secure in the stand-alone model.

Proof.
For the sake of contradiction, we assume a T-time adversary $\mathcal{B}$ that violates the bound on the deception probability, i.e.,

$$\mathrm{Adv}_{\pi}^{\mathsf{forge}}(\mathcal{B}) > n \cdot \varepsilon_{\mathrm{nm}} + \varepsilon_{\mathrm{b}} + \max\{\varepsilon_{\mathrm{au}}, \varepsilon_{\mathrm{ar}}\} \quad .$$

Then, we transform $\mathcal{B}$ into an adversary $\mathcal{A}$ against the non-malleability games $\mathcal{G}_0^{\mathsf{nm}}$, $\mathcal{G}_1^{\mathsf{nm}}$ depicted in Figures 3.11 and 3.12. Clearly, we have

$$\begin{aligned} \mathrm{Adv}^{\mathsf{nm}}(\mathcal{A}) &= |\Pr[\mathcal{G}_0^{\mathsf{nm}} = 1] - \Pr[\mathcal{G}_1^{\mathsf{nm}} = 1]| \\ &= |\Pr[\mathcal{G}_0^{\mathsf{nm}} = 1 | \mathcal{A}_1 \neq \perp] - \Pr[\mathcal{G}_1^{\mathsf{nm}} = 1 | \mathcal{A}_1 \neq \perp]| \cdot \Pr[\mathcal{A}_1 \neq \perp] \quad . \end{aligned}$$

The exact reduction is depicted in Figure 9.4. It will be explained further in Lemma 9.3 and in Lemma 9.4, but here, we just note that $\mathcal{A}_1$ simulates an instance π of the SAS-GMA protocol for $\mathcal{B}$ so that $\mathcal{A}_2$ can compute the predicate "$\mathcal{B}$ succeeds in deception" in the non-malleability game.

More precisely, $\mathcal{A}_1$ replaces the commitment c_k of $\mathcal{P}_k$ by the challenge commitment $c \leftarrow \mathsf{commit}(\mathsf{crs}, k, r)$ for $r \in_{\mathsf{u}} \mathcal{R}$. As $\mathcal{A}_1$ can pass information to $\mathcal{A}_2$ only via the commitment vector $\widehat{\vec{c}}$ and the advice σ, then the predicate "$\mathcal{B}$ succeeds in deception" must be computable from σ, $\widehat{\vec{c}}$ and the corresponding decommitment vector $\widehat{\vec{d}}$. The latter is only possible if $\mathcal{P}_k$ is the last honest party to release his decommitment value d_k, see Lemma 9.4. $\mathcal{A}_1$ must choose k randomly from the group $\mathcal{G}$ provided by $\mathcal{B}$ after seeing crs and guess that it is the right one. Lemma 9.3–9.5 establish

$$\begin{aligned} \mathrm{Adv}_{\mathcal{C}om}^{\mathsf{nm}}(\mathcal{A}) &= |\Pr[\mathcal{G}_0^{\mathsf{nm}} = 1 | \mathcal{A}_1 \neq \perp] - \Pr[\mathcal{G}_1^{\mathsf{nm}} = 1 | \mathcal{A}_1 \neq \perp]| \cdot \Pr[\mathcal{A}_1 \neq \perp] \\ &\geq (1 - \max\{\varepsilon_{\mathrm{au}}, \varepsilon_{\mathrm{ar}}\}) \cdot \frac{1}{n}(\mathrm{Adv}_{\pi}^{\mathsf{forge}}(\mathcal{B}) - \varepsilon_{\mathrm{b}}) \\ &\geq \frac{1}{n}(\mathrm{Adv}_{\pi}^{\mathsf{forge}}(\mathcal{B}) - \varepsilon_{\mathrm{b}}) - \frac{1}{n} \cdot \max\{\varepsilon_{\mathrm{au}}, \varepsilon_{\mathrm{ar}}\} \\ &> \varepsilon_{\mathrm{nm}} \quad . \end{aligned}$$

As the working time of $(\mathcal{A}_1, \mathcal{A}_2)$ is $T + 2\tau + \mathcal{O}(n) = T + \mathcal{O}(1)$ where τ is the working time of the honest parties, we have reached a desired contradiction.

■

Lemma 9.3 (Stand-Alone Security of LP-SAS-GMA (Part 1)).
The sub-adversary $\mathcal{A}_1$ described below satisfies $\Pr[\mathcal{A}_1 \neq \perp] \geq \frac{1}{n} \cdot (\mathrm{Adv}_{\pi}^{\mathsf{forge}}(\mathcal{A}) - \varepsilon_{\mathrm{b}})$ *and the challenger $\mathcal{C}$ never halts unless $\mathcal{A}_1 = \perp$.*

Proof.
The sub-adversary $\mathcal{A}_1$ sketched in Figure 9.4 first forwards crs to $\mathcal{B}$ who replies with $\mathcal{G}$ and $\vec{m}$. Hence, $\mathcal{A}_1(\mathsf{crs})$ can choose $k \in_{\mathsf{u}} \mathcal{G}$ and send to $\mathcal{C}$ a description of the uniform distribution over $\{k\} \times \mathcal{R}$ as MGen.

Clearly, $\mathcal{A}_1$ simulates the execution of the protocol π. It starts by choosing the protocol inputs $\vec{m}$ and then follows the protocol specifications. Note that $\mathcal{A}_1$ simulates the transmission of messages over the insecure channel. Indeed, these messages are routed via the active adversary $\mathcal{B}$ and may be modified. So, $\mathcal{A}_1$ sends the protocol messages to $\mathcal{B}$ and receives (possibly modified) messages from $\mathcal{B}$. For instance, these active attacks are illustrated in Figure 9.4 by a message (i, m_i, c_i) sent by the participant $\mathcal{P}_i$ (simulated by $\mathcal{A}$) to all other participants in the group $\mathcal{G}$. Then, the parties $\mathcal{P}_j$ (simulated by $\mathcal{A}$ too) will receive $(i, \widehat{m}_{ij}, \widehat{c}_{ij})$ indicating that it was sent by $\mathcal{P}_i$, the hat indicates that it transited through the adversary $\mathcal{B}$, and finally was received by $\mathcal{P}_j$. If $\mathcal{B}$ corrupts $\mathcal{P}_i$, then $\mathcal{A}_1$ gives the control over $\mathcal{P}_i$ to $\mathcal{B}$ as in the real execution of π (if $\mathcal{P}_k$ is corrupted then d_k must be released).

To be more precise, the simulation of π follows the specification of the SAS-GMA except for computing c_k, d_k. Indeed, given c from the challenger $\mathcal{C}$, the sub-adversary $\mathcal{A}_1$ will use it in the simulation of π so that $c_k \leftarrow c$ and collect all messages received by all nodes in $\mathcal{G}$. The simulation continues until $\mathcal{P}_k$ must release d_k. To proceed, $\mathcal{A}_1$ passes all variables that are needed to compute the predicate "$\mathcal{B}$ succeeds in deception" to $\mathcal{C}$:

1. Compute sets $\mathcal{I} = \{(j,i) : \widehat{c}_{ji} \neq c\}$ and $\mathcal{J} = \{(j,i) : \widehat{c}_{ji} = c\}$.
2. Send σ to $\mathcal{C}$ where σ contains the sets $\mathcal{I}, \mathcal{J}, \mathcal{G}$, all observed $\widehat{m}_{ji}$, and the current $\mathcal{H}$.
3. Send all plausible commitments $\widehat{\vec{c}} = (\widehat{c}_{ji})$ for $(j,i) \in \mathcal{I}$ to $\mathcal{C}$.

Then the challenger $\mathcal{C}$ releases d, and $\mathcal{A}_1$ continues the simulation of π with $d_k \leftarrow d$ until the end or it halts if one of the following failure events occur:

F_1: The adversary $\mathcal{B}$ fails in deception.

F_2: A double opening is revealed: $\perp\mathsf{open}(\mathsf{crs}, c, d) \neq \mathsf{open}(\mathsf{crs}, c, \widehat{d}_{ji}) \neq \perp$.

F_3: The node $\mathcal{P}_k$ is not the last honest node to reveal the decommitment.

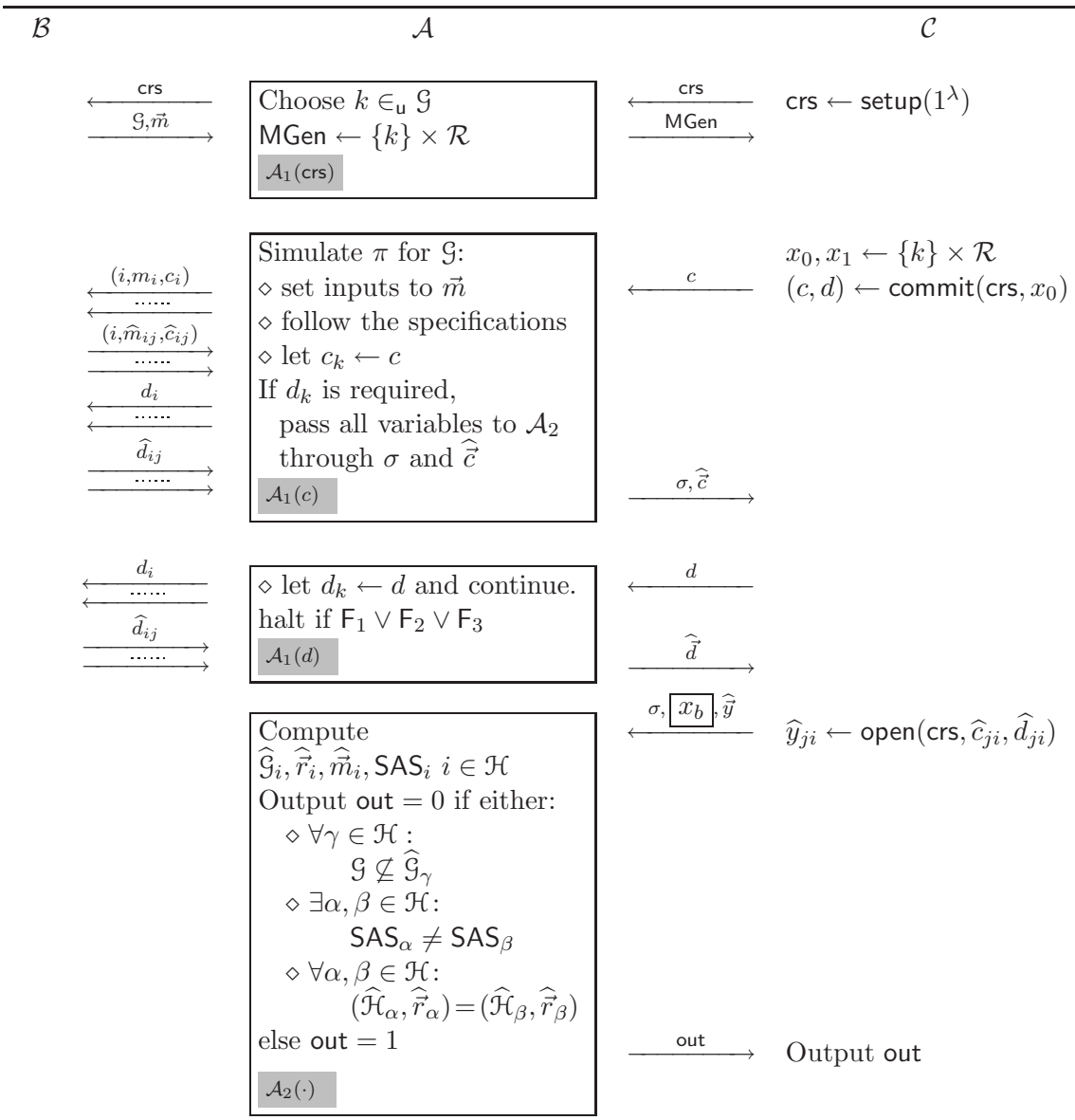


Figure 9.4. *Reduction to the NM Game $\mathcal{G}_b^{\mathsf{nm}}$ for $b \in \{0, 1\}$.*

By this construction, we have $\Pr[\neg\mathsf{F}_1] = \mathrm{Adv}^{\mathsf{forge}}_{\pi}(\mathcal{B})$. As well, we have $\Pr[\mathsf{F}_2] \leq \varepsilon_{\mathrm{b}}$ or otherwise $\mathcal{A}_1$ can be used to defeat the binding property of the commitment scheme.

Note that $\mathcal{B}$ cannot succeed if it corrupts all nodes. So, without loss of generality, we assume that $\mathcal{H} \neq \emptyset$. If $\mathcal{P}_k$ is the last honest node that releases d_k, then the simulation is perfect. Clearly, we have $\Pr[\neg\mathsf{F}_3] = \frac{1}{|\mathcal{G}|}$. The latter is true even if $\neg\mathsf{F}_1$ and $\neg\mathsf{F}_2$. We finally obtain

$$\Pr[\mathcal{A}_1 \neq \perp] = \quad \Pr[\neg\mathsf{F}_3 | \neg\mathsf{F}_1 \wedge \neg\mathsf{F}_2] \cdot \Pr[\neg\mathsf{F}_1 \wedge \neg\mathsf{F}_2] \geq \quad \frac{1}{n} \cdot (\mathrm{Adv}^{\mathsf{forge}}_{\pi}(\mathcal{B}) - \varepsilon_{\mathrm{b}}) \ .$$

Finally, note that $\mathcal{C}$ halts only if some $\widehat{d}_{ji}$ is an invalid decommitment value but then $\mathcal{B}$ fails also in the simulation of π and $\mathcal{A}_1 = \perp$.

■

Lemma 9.4 (Stand-Alone Security of LP-SAS-GMA (Part 2)).
If $\mathcal{A}_1 \neq \perp$ in the game $\mathcal{G}^{\mathsf{nm}}_0$, then $\mathcal{A}_2$ described below correctly recovers the end state of the simulation and thus $\Pr[\mathcal{G}^{\mathsf{nm}}_0 = 1 | \mathcal{A}_1 \neq \perp] = 1$.

Proof.
Assuming that $\mathcal{A}_1 \neq \perp$, then the simulation conducted by $\mathcal{A}_1$ ended with a successful deception.

$\mathcal{A}_2$ is in charge to determine if the predicate "$\mathcal{B}$ succeeds in deception" holds or not.

As $\mathcal{P}_k$ was indeed the last honest node to release d_k, then $\widehat{m}_{ji}, \widehat{r}_{ji}$ for indices $(j,i) \in \mathcal{I} \cup \mathcal{J}$ are sufficient to recover all SAS messages computed by $\mathcal{H}$. All observed $\widehat{m}_{ji}$ are trivially recoverable from σ. By the construction, $\widehat{y}_{ji} = \mathsf{open}(\mathsf{crs}, \widehat{c}_{ji}, \widehat{d}_{ji}) = (j, \widehat{r}_{ji})$ for $(j,i) \in \mathcal{I}$ so the $\widehat{r}_{ji}$ are available from the $\widehat{y}_{ji}$. Since F_2 cannot happen, $\mathsf{open}(\mathsf{crs}, \widehat{c}_{ji}, \widehat{d}_{ji})$ will open to the same message for all $(j,i) \in \mathcal{J}$, i.e., $\mathsf{open}(\mathsf{crs}, c, d) = x_0$. As a result, $\mathcal{A}_2$ can compute all $\widehat{\vec{m}}_i$ and $\widehat{\vec{r}}_i$ for $i \in \mathcal{H}$ by setting $(k, r_{kk}) \leftarrow x_0$ and replacing $\mathsf{open}(\mathsf{crs}, \widehat{c}_{ji}, \widehat{d}_{ji})$ calls with appropriate values specified above. Then it remains to restore $\mathsf{SAS}_i \leftarrow h((\widehat{\mathcal{H}}_i, \widehat{\vec{m}}_i), \widehat{\vec{r}}_i)$ for $i \in \mathcal{H}$, to test $\mathsf{SAS}_\alpha = \mathsf{SAS}_\beta$ for all $\alpha, \beta \in \mathcal{H}$, and to output 1 in case of deception. Recall that deception happens only if the test values SAS_α match but some $(\widehat{\mathcal{G}}_\alpha, \widehat{\vec{m}}_\alpha) \neq (\widehat{\mathcal{G}}_\beta, \widehat{\vec{m}}_\beta)$ and $\mathcal{G} \subseteq \widehat{\mathcal{G}}_\alpha$.

As $\mathcal{A}_2$ computes the predicate "$\mathcal{B}$ succeeds in deception" and since $\mathcal{A}_1 \neq \perp$ implies $\neg\mathsf{F}_1$, we have $\Pr[\mathcal{G}^{\mathsf{nm}}_0 = 1 | \mathcal{A}_1 \neq \perp] = 1$.

■

Lemma 9.5 (Stand-Alone Security of LP-SAS-GMA (Part 3)).
Let $\mathcal{A}_2$ be as described in Lemma 9.4. Then we can bound the conditional probability $\Pr[\mathcal{G}^{\mathsf{nm}}_1 = 1 | \mathcal{A}_1 \neq \perp] \leq \max\{\varepsilon_{\mathrm{au}}, \varepsilon_{\mathrm{ar}}\}$.

Proof.
Assuming that $\mathcal{A}_1 \neq \perp$, then the simulation conducted by $\mathcal{A}_1$ ended with a successful deception. Consequently, $c = \mathsf{commit}(\mathsf{crs}, k, r_k) = \mathsf{commit}(\mathsf{crs}, x_1)$ could have been broadcast only as $\widehat{c}_{ki}$, otherwise $\mathcal{B}$ would have failed in deception. Therefore, $\mathcal{I} \subseteq \{k\} \times \mathcal{H}$, $\widehat{\mathcal{G}}_i$, $\widehat{\vec{m}}_i$, and all components of $\widehat{\vec{r}}_i$ except $\widehat{r}_{ki}$ for $i \in \mathcal{H}$ are fixed when $\mathcal{A}_2$ starts. Next, we bound the probability $\mathsf{SAS}_\alpha = \mathsf{SAS}_\beta$ for all $\alpha, \beta \in \mathcal{H}$.

Consider the authentic broadcast of c_k first, i.e., the case $\mathcal{I} = \{k\} \times \mathcal{H}$. The condition $\neg\mathsf{F}_1$ implies $(\widehat{\mathcal{G}}_\alpha, \widehat{\vec{m}}_\alpha) \neq (\widehat{\mathcal{G}}_\beta, \widehat{\vec{m}}_\beta)$ for some $\alpha, \beta \in \mathcal{H}$. As $x_1 \in_{\mathsf{u}} \{k\} \times \mathcal{R}$, the universality of H with respect to all sub-key pairs[2] yields

$$\Pr\left[\widehat{r}_k \in_{\mathsf{u}} \mathcal{R} : \mathsf{H}((\widehat{\mathcal{G}}_\alpha, \widehat{\vec{m}}_\alpha), \ldots, \widehat{r}_k, \ldots) = \mathsf{H}((\widehat{\mathcal{G}}_\beta, \widehat{\vec{m}}_\beta), \ldots, \widehat{r}_k, \ldots)\right] \leq \varepsilon_{\mathrm{au}}$$

where "..." denote the fixed components of $\widehat{\vec{r}}_\alpha$ and $\widehat{\vec{r}}_\beta$. So, we have obtained

$$\Pr\left[\mathcal{A}_2 = 1 | \mathcal{I} = \{k\} \times \mathcal{H}\right] \leq \Pr\left[\mathsf{SAS}_\alpha = \mathsf{SAS}_\beta | \mathcal{I} = \{k\} \times \mathcal{H}\right] \leq \varepsilon_{\mathrm{au}} \ .$$

In the remaining case, let $\mathcal{H}_0$ be the set of honest nodes that receive c_k, i.e., $\mathcal{I} = \{k\} \times \mathcal{H}_0$. Since there is a compulsory node γ such $\mathcal{H} \subseteq \mathcal{G} \subseteq \widehat{\mathcal{G}}_\gamma$ there are nodes $\alpha \in \mathcal{H}_0$ and $\beta \in \mathcal{H} \setminus \mathcal{H}_0$ such that $\mathcal{A}_2$ compares SAS_α and SAS_β. Moreover, α, β, and SAS_β are fixed before x_1 is revealed and almost regularity with respect to all sub-keys provides

$$\Pr\left[\widehat{r}_k \in_{\mathsf{u}} \mathcal{R} : \mathsf{H}((\widehat{\mathcal{G}}_\alpha, \widehat{\vec{m}}_\alpha), \ldots, \widehat{r}_k, \ldots) = \mathsf{SAS}_\beta\right] \leq \varepsilon_{\mathrm{ar}}$$

where "..." denote the fixed components of $\widehat{\vec{r}}_\alpha$. Therefore, we have proved the desired claim, i.e., $\Pr\left[\mathcal{G}_1^{\mathrm{nm}} = 1 | \mathcal{A}_1 \neq \perp\right] \leq \max\{\varepsilon_{\mathrm{au}}, \varepsilon_{\mathrm{ar}}\}$.

■

Security in Complex Settings

As an informal result, we give the overall security of our LP-SAS-GMA depicted in Figure 9.3.

Theorem 9.6 (Security of LP-SAS-GMA).
Consider the SAS-GMA protocol π depicted in Figure 9.3. Let n be the maximum size of the group $\mathcal{G}$. Let H be a multi-keyed hash function which is $\varepsilon_{\mathrm{au}}$-almost universal with respect to each sub-key pair and $\varepsilon_{\mathrm{ar}}$-almost regular with respect to each sub-key. Let the commitment scheme be $(T + \mu, \varepsilon_{\mathrm{b}})$-binding and $(T + \mu, \varepsilon_{\mathrm{nm}})$-non-malleable

There exists a (small) constant μ such that the protocol π is $(T,\ q,\ q \cdot (n\varepsilon_{\mathrm{nm}} + \varepsilon_{\mathrm{b}} + \max\{\varepsilon_{\mathrm{au}}, \varepsilon_{\mathrm{ar}}\}))$-secure.

[2]Note that the varying components $\widehat{r}_{k\alpha} = \widehat{r}_{k\beta}$ can be in different locations of $\widehat{\vec{r}}_\alpha$, $\widehat{\vec{r}}_\beta$.

Proof.
The proof directly follows from Theorem 9.2, proving the security of LP-SAS-GMA in the stand-alone model, and from Theorem 6.9 proving self-composability of any $SGMA$ protocol.

■

9.4 Applications

Applications of SAS-based group message authentication protocols (SAS-GMA) are similar to applications in two-party settings. However, group protocols are more general than two-party protocol. Therefore, applications may be any of the two-party applications, see Section 8.6, but adding the group feature. For instance, a secure Voice over IP telephone call can be now extended to a secure Voice over IP *conference* call.

Chapter

TEN

From Message Authentication to Key Agreement

In this chapter, we show how to combine message authentication with any key agreement protocol. The resulting key agreement protocol is secure against active attacks. More precisely, we start by defining authenticated key agreements. Then, in Section 10.2, we study two well established key agreement protocols: the Diffie-Hellman [DH76] and the Burmester-Desmedt [DB94]. The first one is actually limited to two-party settings while the second one was designed for group settings. Both protocols are resistant to passive attacks only. Therefore, protocol messages require authentication. In Section 10.3, we study prior work on specific authenticated key agreements. In Section 10.4, we present a generic construction for authenticated key agreement based on two primitives: message authentication and key agreement. We published this construction in [PV06b]. Then, we present two optimal SAS-based authenticated key agreement protocols: one for two-party settings in Section 10.5, called PV-SAS-AKA and published in [PV06b], and another one for group settings in Section 10.6, called LP-SAS-GKA and published in [LP08]. Finally, we present possible applications and, in particular, we study the design of a secure voice over IP system.

10.1 Authenticated Key Agreement Primitive

In general, the definition for a two-party setting is easier to understand than the one for a group setting. Therefore, we start by giving the definition for two-party and we will then extend it to groups.

Definition 10.1 (Two-Party Authenticated Key Agreement).
An Authenticated Key Agreement *(AKA) protocol between Alice and Bob starts with no input, is independent from the current state, and ends with no output but a final state on each side specifying a pair* $(\mathsf{sk}, \mathsf{id})$.

An honest run of an AKA protocol should end with state $(\mathsf{sk}, \mathsf{id}_B)$ *on the side of Alice and with state* $(\mathsf{sk}, \mathsf{id}_A)$ *on the side of Bob.*

An attack is successful if a $\mathsf{test}(n, \mathsf{sk}, \mathsf{id})$ *query is positively answered where* n *and* id *corresponds to nodes on which no* reveal *nor* corrupt *query was made. For simplicity, we do not consider attacks making Alice and Bob end on some inconsistent states. Namely,* mutual authentication *is assumed to be (implicitly or explicitly) made by further communication.*

Definition 10.2 (Group Key Agreement).
A Group Key Agreement *(GKA) protocol* π *between a group* $\mathcal{G}$ *composed of* ℓ *participants* $\mathcal{G} = \{\mathsf{id}_1, \ldots, \mathsf{id}_\ell\}$ *starts with no input, is independent from the current state, and outputs a pair* $(\mathsf{sk}, \mathcal{G}')$.

An honest run of a group AKA should end with the same output $(\mathsf{sk}, \mathcal{G}')$ *for all participants.*

An attack is successful if a $\mathsf{test}(n, \mathsf{sk}, \mathsf{id})$ *query is positively answered where* n *and* id *corresponds to nodes on which no* reveal *nor* corrupt *query was made. An attack is also successful if two non-corrupted participants outputs two different pairs* $(\mathsf{sk}, \mathcal{G}')$.

Obviously, any GKA protocol that is (T, ε_a)-immune against active attacks and (T, ε_p)-secure against passive attacks is also $(T, \varepsilon_a + \varepsilon_p)$-secure, as long as both definitions are given in the same attack model. A formal proof is given later. In particular, we can construct also universally composable SAS-based key agreement protocols as long as the underlying key agreement protocol is universally composable against passive attacks. However, stand-alone security is sufficient for many practical settings, since the key agreement protocols are often executed in isolation to set up the communication network.

Note that it is important to minimize the amount of manually authenticated communication in scenarios where nodes can form many subgroups. In particular, it should be easy to exclude corrupted nodes from the group without transferring any additional SAS message.

10.2 (Non-Authenticated) Key Agreement

10.2.1 The Diffie-Hellman Key Agreement Protocol

The Diffie-Hellman (DH) two-party key agreement protocol [DH76] allows two parties, say Alice and Bob, to agree on a shared secret key. It is provably secure against passive attacks only. Clearly, this means that protocol messages should be authenticated to avoid active attacks.

A classical authenticated DH protocol over a multiplicative group spanned by a generator g consists for Alice (resp. Bob) in picking a random (secret) integer x_A (resp. x_B), sending the Diffie-Hellman public keys, $y_A = g^{x_A}$ (resp. $y_B = g^{x_B}$) over the authenticated channel, computing $\mathsf{sk}_A = y_B^{x_A}$ (resp. $\mathsf{sk}_B = y_A^{x_B}$) and ending with state $(\mathsf{sk}_A, \mathsf{id}_B)$ (resp. $(\mathsf{sk}_B, \mathsf{id}_A)$). If the run is honest, we should obtain $\mathsf{sk}_A = \mathsf{sk}_B = y_A^{x_A x_B}$. In this case, authenticated messages are pretty long, but authentication is necessary to thwart man-in-the-middle attacks. For more details, see Section 2.1.2.

10.2.2 The Burmester-Desmedt Group Key Agreement Protocol

The Burmester-Desmedt (BD) group key agreement protocol [DB94] is provably secure against passive attacks [BD05] and thus is a perfect starting point for a SAS-based GKA. The BD protocol is a generalization of the DH key agreement protocol. Note that other two-party key agreement protocols can also be generalized to groups, for instance, see the compiler of Just and Vaudenay [JV96].

For simplicity, consider a group $\mathcal{G}$ of n participants[1] $\mathcal{P}_{\mathsf{id}_1}, \ldots, \mathcal{P}_{\mathsf{id}_n}$ arranged in a ring. As depicted on Figure 10.1, the protocol only requires two rounds over an authenticated channel, while most of the schemes requires $\mathcal{O}(n)$ rounds. Let g be a generator of a q-element secure Diffie-Hellman Decision Group $\mathbb{G}$. At the end of the protocol, each participant $\mathcal{P}_i$ obtains $\mathsf{sk}_i = g^{k_1k_2+k_2k_3+\ldots+k_nk_1}$.

[1] The protocol can be trivially generalized to any n-element group $\mathcal{G}$.

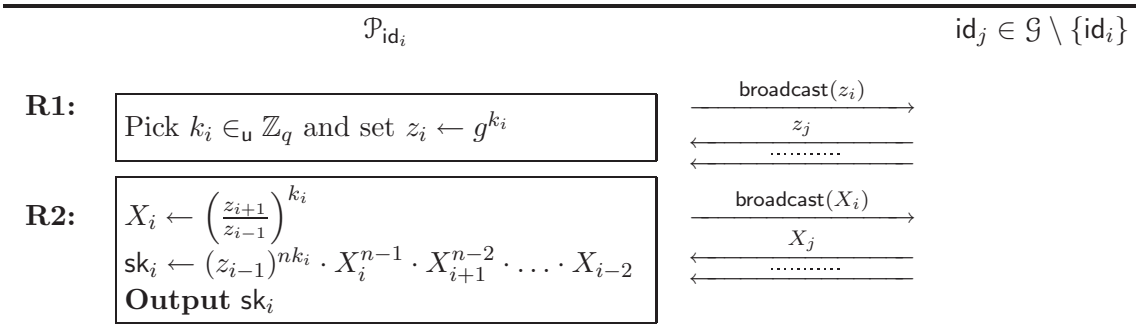


Figure 10.1. *The BD Group Key Agreement Protocol.*

The key derivation for an arbitrary party $\mathcal{P}_{\mathsf{id}_i}$ with $\mathsf{id}_i \in \mathcal{G}$ works as follows:

$$
\begin{aligned}
\mathsf{sk}_i &= (z_{i-1})^{nk_i} \cdot X_i^{n-1} \cdot X_{i+1}^{n-2} \cdot \ldots \cdot X_{i-2} \\
&= \left[z_{i-1}^{k_i}\right] \cdot \left[z_{i-1}^{k_i} \cdot X_i\right] \cdot \left[z_{i-1}^{k_i} \cdot X_i \cdot X_{i+1}\right] \cdot \ldots \cdot \left[z_{i-1}^{k_i} \cdot X_i \cdot X_{i+1} \cdot \cdots \cdot X_{i-2}\right] \\
&= \left[z_{i-1}^{k_i}\right] \cdot \left[z_{i-1}^{k_i} \cdot \left(\frac{z_{i+1}}{z_{i-1}}\right)^{k_i}\right] \cdot \left[z_{i-1}^{k_i} \cdot \left(\frac{z_{i+1}}{z_{i-1}}\right)^{k_i} \cdot \left(\frac{z_{i+2}}{z_i}\right)^{k_{i+1}}\right] \cdot \ldots \\
&\qquad \ldots \cdot \left[z_{i-1}^{k_i} \cdot \left(\frac{z_{i+1}}{z_{i-1}}\right)^{k_i} \cdot \left(\frac{z_{i+2}}{z_i}\right)^{k_{i+1}} \cdot \ldots \cdot \left(\frac{z_{i-1}}{z_{i-3}}\right)^{k_{i-2}}\right] \\
&= \left[g^{k_{i-1}k_i}\right] \cdot \left[g^{k_{i-1}k_i} \cdot \frac{g^{k_ik_{i+1}}}{g^{k_{i-1}k_i}}\right] \cdot \left[g^{k_{i-1}k_i} \cdot \frac{g^{k_ik_{i+1}}}{g^{k_{i-1}k_i}} \cdot \frac{g^{k_{i+1}k_{i+2}}}{g^{k_ik_{i+1}}}\right] \cdot \ldots \\
&\qquad \ldots \cdot \left[g^{k_{i-1}k_i} \cdot \frac{g^{k_ik_{i+1}}}{g^{k_{i-1}k_i}} \cdot \frac{g^{k_{i+1}k_{i+2}}}{g^{k_ik_{i+1}}} \cdots \frac{g^{k_{i-1}k_{i-2}}}{g^{k_{i-3}k_{i-2}}}\right] \\
&= \left[g^{k_{i-1}k_i}\right] \cdot \left[g^{k_ik_{i+1}}\right] \cdot \left[g^{k_{i+1}k_{i+2}}\right] \cdot \ldots \cdot \left[g^{k_{i-2}k_{i-1}}\right] \\
&= g^{k_1k_2+k_2k_3+\ldots+k_nk_1}
\end{aligned}
$$

So, if the run is honest, i.e., all the sent messages are reaching the other parties with no modification, then all parties end with the same secret key sk.

10.3 Prior Authenticated Key Agreements

In this section, we present two two-party AKA protocols: the Hoepman and the PGPfone protocols. For both, the security is not formally proven ([Hoe04] only provides a sketch of argument for the security).

In what follows, we will study a generic construction reducing the amount of authenticated bits in AKA protocols. Using this construction with the DH protocol and the Vau-SAS-MCA

protocol, we obtain the DH-SC protocol of Čagalj, Čapkun, and Hubaux [CCH05]. By using an *optimized* SAS-MCA protocol we can save one protocol move. More details about this construction are presented in Section 10.5.

So far, there is no AKA specially designated for groups. One can, for instance, use the BD presented in Section 10.2.2 and add some authentication to the exchanged messages. We study this construction in Section 10.6.

10.3.1 The Hoepman AKA Protocol

We first informally present an AKA protocol from Hoepman [Hoe04]. It is based on the DH protocol and it uses an authenticated channel for the authentication of each DH value. This protocol runs in three steps: commitment, authentication, and opening. Note that the original protocol has a fourth step: the key validation.

As depicted in Figure 10.2, instead of revealing its DH public key, each party first commits on it, keeping it hidden. In the next step, each participant authenticates a piece of its DH public key and opens their commitment just after. Finally, both open the received commitments and check the authenticated string before completing the regular DH protocol.

Alice		Bob
pick x_A, $y_A \leftarrow g^{x_A}$		pick x_B, $y_B \leftarrow g^{x_B}$
$(c_A, d_A) \leftarrow \mathsf{commit}(y_A)$	$\xrightarrow{c_A}$	
	$\xleftarrow{c_B}$	$(c_B, d_B) \leftarrow \mathsf{commit}(y_B)$
$\mathsf{SAS}_A \leftarrow \mathsf{pieceof}(y_A)$	$\xrightarrow{\mathsf{auth}_{\mathsf{id}_A}(\mathsf{SAS}_A)}$	
	$\xleftarrow{\mathsf{auth}_{\mathsf{id}_B}(\mathsf{SAS}_B)}$	$\mathsf{SAS}_B \leftarrow \mathsf{pieceof}(y_B)$
	$\xrightarrow{d_A}$	$\widehat{y}_A \leftarrow \mathsf{open}(\widehat{c}_A, \widehat{d}_A)$
$\widehat{y}_B \leftarrow \mathsf{open}(\widehat{c}_B, \widehat{d}_B)$	$\xleftarrow{d_B}$	
check $\mathsf{SAS}_B \stackrel{?}{=} \mathsf{pieceof}(\widehat{y}_B)$		check $\mathsf{SAS}_A \stackrel{?}{=} \mathsf{pieceof}(\widehat{y}_A)$
$\mathsf{sk}_A \leftarrow (\widehat{y}_B)^{x_A}$		$\mathsf{sk}_B \leftarrow (\widehat{y}_A)^{x_B}$
final state: $\mathsf{sk}_A, \mathsf{id}_B$		**final state**: $\mathsf{sk}_B, \mathsf{id}_A$

Figure 10.2. *The Hoepman AKA Protocol.*

10.3.2 PGPfone

An interesting AKA protocol was used by Zimmermann for the PGPfone in 1995[2]. The protocol is depicted in Figure 10.3. Compared to the previous one, the advantages are first to reduce the number of moves over the insecure channel and second to make both authenticated strings equal. Note that this protocol was given with no security proof.

This protocol also consists in exchanging DH public keys and then authenticating them. For that reason, the first participant, i.e., Alice, commits to its public key, then the second participant reveal its public-key, and Alice opens their commitment. Finally, the authenticated string SAS is a piece of the digest (denoted truncH on Figure 10.3) of the DH protocol transcript.

Alice		Bob
pick x_A, $y_A \leftarrow g^{x_A}$		pick x_B, $y_B \leftarrow g^{x_B}$
$(c, d) \leftarrow \mathsf{commit}(y_A)$	$\xrightarrow{c}$	
	$\xleftarrow{y_B}$	
	$\xrightarrow{d}$	$\widehat{y}_A \leftarrow \mathsf{open}(\widehat{c}, \widehat{d})$
$\mathsf{sk}_A \leftarrow (\widehat{y}_B)^{x_A}$		$\mathsf{sk}_B \leftarrow (\widehat{y}_A)^{x_B}$
$\mathsf{SAS}_A \leftarrow \mathsf{truncH}(y_A \Vert \widehat{y}_B)$		$\mathsf{SAS}_B \leftarrow \mathsf{truncH}(\widehat{y}_A \Vert y_B)$
	$\xrightarrow{\mathsf{auth}_{\mathsf{id}_A}(\mathsf{SAS}_A)}$	check $\mathsf{SAS}_A \stackrel{?}{=} \mathsf{SAS}_B$
check $\mathsf{SAS}_A \stackrel{?}{=} \mathsf{SAS}_B$	$\xleftarrow{\mathsf{auth}_{\mathsf{id}_B}(\mathsf{SAS}_B)}$	
final state: $\mathsf{sk}_A, \mathsf{id}_B$		**final state**: $\mathsf{sk}_B, \mathsf{id}_A$

Figure 10.3. *The PGPfone AKA Protocol.*

10.4 KA+MA = AKA

The most straightforward way to achieve (T, ε)-immunity against active attacks is to authenticate the entire protocol transcript. Namely, participants must first execute the group key agreement protocol and then use a group message authentication protocol to verify that all transferred message were unaltered. A naive implementation, where protocols are executed sequentially, adds three extra rounds to the key agreement protocol. However, since message authentication protocols are universally composable, we can execute them in parallel and save two messages for two-party and one complete round for group protocols.

[2] personal communication.

It is also possible to merge both protocols more tightly and thus obtain a more efficient protocols, see [LN06a].

For two-party protocols, it is reasonable to combine the DH key agreement protocol with one of the MCA protocols discussed in Chapter 8. A schema of the construction is depicted in Figure 10.4.

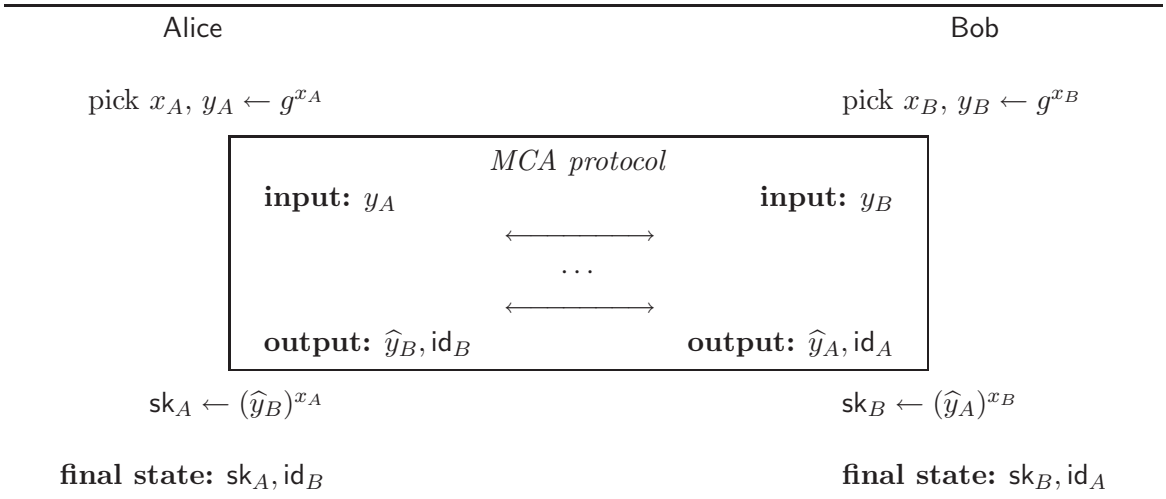


Figure 10.4. *The DH Protocol Over a MCA Protocol.*

We propose a generic SAS-based construction for an AKA protocol that we call the constructed AKA protocol or simply *the* AKA protocol. For this, we use an initial AKA protocol (with longer strings to be authenticated), that we call the AKA_0 protocol, and an MCA protocol with short SAS. Consider that the AKA_0 protocol requires $n_k \geq 2$ moves with the $(n_k - 1)^{\text{th}}$ being from Alice to Bob. Consider also that the MCA protocol requires $n_a \geq 2$ moves over the insecure channel, the first one being from Alice to Bob. In the AKA protocol, the $(n_k - 2)$ first moves of the AKA_0 are performed over the insecure channel. Then, both participants assembles his view of the protocol transcript τ by concatenating all protocol messages (sent and received ones). Then, an MCA protocol starts. Alice wishes to authenticate τ concatenated with her $(n_k - 1)^{\text{th}}$ message α in the AKA_0 protocol. Bob wishes to authenticate the same $\tau \| \alpha$ concatenated with his last message β in the AKA_0 protocol. (Note that Bob selects the message to be authenticated after receiving Alice's first message in the MCA protocol.) At the end, both participants use the authenticated messages to complete the AKA_0 protocol and end with final states as specified in the AKA_0 protocol. We have $n_k + n_a - 2$ moves in total.

Note that MCA can have $n_a < 2$. For instance the trivial MCA sending input messages MCA over the authenticated channel. In that case, we augment the MCA protocol by virtual moves and we obtain n_k moves in total. However, MCA protocols with $n_a < 2$ must have pretty large SAS to exchange the messages.

We can make a similar construction based on an n'_a-move MMA protocol instead of an MCA protocol. In that case, if the $n'_a \geq 1$ then we can only encapsulate the last move β of the AKA_0 protocol in the MMA protocol, leading us to $\max(n_k, n_k + n'_a - 1)$ moves in total.

Theorem 10.3 (Security of the AKA Construction).
Let us consider a (T, ε_k)-secure AKA protocol with n_k moves (the AKA_0 protocol) and a (T, ε_a)-secure MCA protocol with n_a moves. The generic construction π is essentially an AKA protocol with $\max(n_k,\ n_k + n_a - 2)$ moves in which the structure of authenticated messages is similar as in the MCA protocol.

There exists a constant μ such that for any T the protocol π is $(\mu \cdot T,\ \varepsilon_k + \varepsilon_a)$-secure.

Using the DH protocol and an n_a-move MCA protocol leads us to a $\max(2, n_a)$-move AKA protocol in which the structure of authenticated messages is similar as in the MCA protocol.

With the construction based on an n'_a-move MMA protocol, we obtain $\max(2, n'_a + 1)$ moves.

In the case where we want to achieve small SAS, we must have n_A and n'_A at least equal to 2, leading us to n_a moves using MCA protocols and $n'_a + 1$ moves using MMA protocols.

Since (n'_a+1)-move MCA protocols can be made from n'_a-move protocols, we may decrease the total number of moves in AKA protocols by starting from MCA protocols directly.

Proof.
While we give the proof for a two-party setting, it can be extended to group settings by using the same idea.

Let the constructed AKA_0 protocol view from Alice, i.e., sent and received messages, be the sequence of messages $m_{A,1}, m_{A,2}, \ldots, m_{A,k}$ where $m_{A,k-1}$ is from Alice to Bob and $m_{A,k}$ from Bob to Alice. We denote by τ_A the sequence $m_{A,1}, m_{A,2}, \ldots, m_{A,k-2}$, by α_A the message m_{k-1}, and by $\widehat{\beta}$ the message m_k. We further let $\widehat{\tau}_B \| \widehat{\alpha}_B \| \widehat{\beta}$ be the accepted message from Bob at the end of the MCA protocol.

Similarly, for each instance of Bob, we let τ_B be the constructed transcript of the $n_k - 2$ first messages in the AKA protocol, $\widehat{\tau}_A \| \widehat{\alpha}_A$ be the accepted message at the end of the MCA protocol, and β be his last message in the AKA_0 protocol assuming that Alice's last one is $\widehat{\alpha}$. We let $\alpha = \widehat{\alpha}$. Bob's message to be authenticated is $\tau_B \| \alpha_B \| \beta$.

In the following, we consider an adversary $\mathcal{A}$ against the AKA protocol. By using a simulator $\mathcal{B}$ interacting with $\mathcal{A}$, we will reduce $\mathcal{A}$ into an adversary against the MCA protocol. By using another simulator $\mathcal{C}$ interacting with $\mathcal{A}$, we will reduce $\mathcal{A}$ into an adversary against the AKA_0 protocol.

Clearly, running $\mathcal{A}$ in parallel with $\mathcal{B}$ and $\mathcal{C}$ with the same random source, we derive that whenever $\mathcal{A}$ succeeds, either $\mathcal{B}$ or $\mathcal{C}$ succeeds.

The simulator $\mathcal{B}$ simply simulates instances running the AKA_0 protocol and launches the MCA protocol instances when appropriate. test, remove, reveal and corruct queries can easily be simulated. Clearly, the attack against the MCA protocol succeeds with probability at most ε_a. So, it does not succeed with probability at least $1 - \varepsilon_a$. In those cases, we have $\tau_B = \widehat{\tau}_B$, $\tau_A = \widehat{\tau}_A$, $\alpha_A = \widehat{\alpha}_A = \alpha_B = \widehat{\alpha}_B$, and $\beta = \widehat{\beta}$, just as if the instance of Alice and Bob had the AKA_0 protocol run over an authenticated channel.

The simulator $\mathcal{C}$ simply simulates the MCA protocol and replaces inputs to the send oracle by authenticated ones when possible, or fails. Clearly, the attack against the AKA_0 protocol succeeds with probability at most ε_k.

■

10.5 An Optimal AKA Protocol: PV-SAS-AKA

Given the previous section, it is now straightforward to build an AKA based on a KA and a MCA protocol. In this section, we give a SAS-based Authenticated Key Agreement, called PV-SAS-AKA as example. The construction uses our PV-SAS-MCA protocol and the DH key agreement protocol. The resulting protocol is *optimal*.

Alice		Bob
pick x_A, $y_A \leftarrow g^{x_A}$		pick x_B, $y_B \leftarrow g^{x_B}$
pick $K \in_{\mathsf{u}} \{0,1\}^\kappa$		pick $R \in_{\mathsf{u}} \{0,1\}^\rho$
$(c,d) \leftarrow \mathsf{commit}(y_A, K)$	$\xrightarrow{y_A, c}$	
	$\xleftarrow{y_B, R}$	
	$\xrightarrow{d}$	$\widehat{K} \leftarrow \mathsf{open}(\widehat{y}_A, \widehat{c}, \widehat{d})$
$\mathsf{SAS} \leftarrow \widehat{R} \oplus h(\widehat{y}_B, K)$	$\xrightarrow{\mathsf{auth}_{\mathsf{id}_A}(\mathsf{SAS})}$	$\mathsf{SAS} \stackrel{?}{=} R \oplus h(y_B, \widehat{K})$
check SAS is the same	$\xleftarrow{\mathsf{auth}_{\mathsf{id}_B}(\mathsf{SAS})}$	
check $\mathsf{id}_A \neq \mathsf{id}_B$		
$\mathsf{sk}_A \leftarrow (\widehat{y}_B)^{x_A}$		$\mathsf{sk}_B \leftarrow (\widehat{y}_A)^{x_B}$
final state: $\mathsf{sk}_A, \mathsf{id}_B$		**final state:** $\mathsf{sk}_B, \mathsf{id}_A$

Figure 10.5. *An Optimal SAS-based AKA Protocol: PV-SAS-AKA.*

10.6 An Optimal GKA Protocol: LP-SAS-GKA

In many practical settings, we must be able to exclude group members that behave maliciously. Ideally, this operation should not use additional authenticated messages. Consequently, a textbook key agreement protocol is not suitable for our needs. Indeed, a shared key sk gets compromised as soon as a group member gets corrupted and in that case the whole protocol should be run again with the new restricted group. To avoid this problem, we need a key agreement protocol that also fixes long-term pairwise authentication keys so that we can re-run key agreement protocols with no additional authenticated communication. The corresponding key agreement protocol is depicted in Figure 10.6. We published it in [LP08].

In short, the group members run a SAS-based GKA only once. The first run allows the group members to obtain a common group secret key sk as well as long-term pairwise authentication keys $\mathsf{sk}_{\mathsf{id}_i,\mathsf{id}_j}$ for all pairs $\mathsf{id}_i, \mathsf{id}_j \in \mathcal{G}$. These latters keys provide possibility to re-run ordinary GKA protocols with no additional SAS message. The group secret key sk is generated by the BD GKA while the long-term pairwise authentication keys $\mathsf{sk}_{\mathsf{id}_i,\mathsf{id}_j}$ are formed based on Diffie-Hellman key exchange.

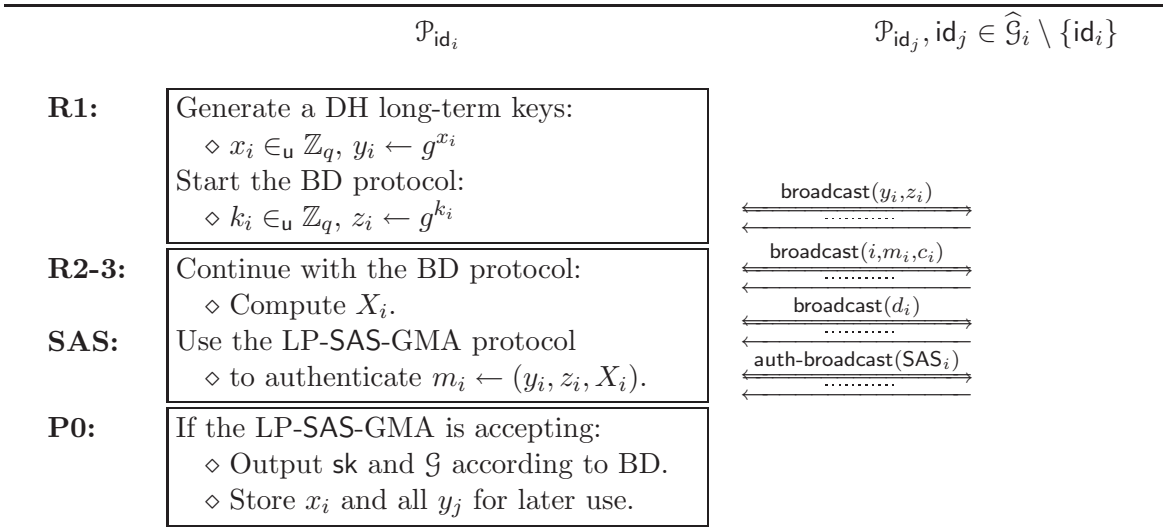


Figure 10.6. *An Optimal SAS-based GKA Protocol: LP-SAS-GKA.*

As the transcript of the BD protocol is authenticated with our LP-SAS-GMA, the BD protocol is immune against active attacks with the same guarantees as Theorem 6.9 and Theorem 9.2 specify. Moreover, any two parties $\mathsf{id}_\alpha, \mathsf{id}_\beta \in \mathcal{H}$ can establish a pairwise secret key $\mathsf{sk}_{\alpha,\beta} = f(g^{x_\alpha x_\beta})$, as they both know the corresponding long-term public keys $y_i = g^{x_i}$ for all group members $i \in \mathcal{G}$. Hence, they can use any classical authentication protocol to protect new instances of the GKA protocol against active attacks. In particular, we can

merge small groups $\mathcal{G}_1, \mathcal{G}_2$, if there is an honest party $\mathcal{P}_i \in \mathcal{G}_1 \cap \mathcal{G}_2$, by sending all intergroup communication through $\mathcal{P}_i$.

Of course, if the formed group is known to have a static nature, then one can skip the setup of long-term DH keys $\mathsf{sk}_{\alpha,\beta}$.

10.7 Applications

Key agreements have many applications. Here, we restricts to applications where an authenticated channel is available, like telephone, or like user channel such as string comparison or string transmission. For instance, SAS-based key agreements may be used to establish a secret key between cell phones, or between devices in an *ad-hoc* network. Note that a similar protocol to our LP-SAS-GKA of Figure 10.5 is used in Bluetooth version 2.1.

Secure Voice over IP

Voice over IP (VoIP) is an emerging application these days. The telephone is widely developed and people are familiar with its use. However, it is still expensive. VoIP emerges principally because it is very cheap, and sometimes free. In addition, voice over IP applications may offer others features as video stream, chat, file transfer, etc.

The user choice today is Skype. Users chose a free and easy to use application. In particular, installation, account creation, and use should be straightforward for any user with no technical skill. The calls are encrypted. However, it is hard to say how the encryption is done and with which key. It seems that Skype is able to decrypt all data. Skype acts as a trusted third party.

One interesting point with voice over IP is the presence of an authenticated channel. Indeed, starting a simple voice over IP call, the channel is clearly insecure and anyone can eavesdrop the conversation. While this channel is vulnerable to passive attacks, it seems that it resists to active attacks. Indeed, the ability of the user to recognize the voice and the behavior of the interlocutor helps a lot. In addition, users are using an interactive channel and it is even more complicated to inject forged packet or to replay recorded sequences. Indeed, it seems hard to forge the vocal channel in order to produce the same voice than the source, to mimic the same behavior than the source, to inject the forged or replayed sequence correctly in the flow of the conversation, and everything in real time. For that reasons, we assume that the vocal channel ensure authenticity and integrity (but not confidentiality) and so users are able to exchange SAS.

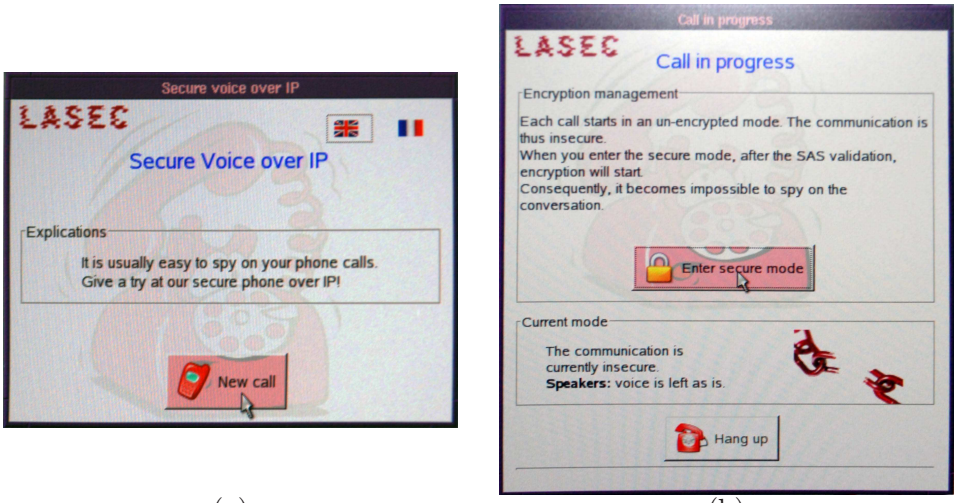
In 2006, we implemented a VoIP system with two main goals: firstly, to present our PV-SAS-AKA protocol and, secondly, to familiarize people with cryptography. This latter

argument required the development of an eavesdropper. Indeed, the functional demonstration is composed of two complete systems (PC and phone) and an additional speaker which simulates an eavesdropper as depicted in Figure 10.7. When the communication is in clear, the speaker plays the eavesdropped voice signals, and when the communication is encrypted, the speaker seems to play anything.

Figure 10.7. *Installation Overview of the Secure Voice over IP System.*

We implemented the PV-SAS-AKA (see Figure 10.5). For the demo purpose, we adapted the protocol a little. For instance, starting the application opens the main window, depicted in Figure 10.8(a) and launches a TCP server. When a user starts a call, he selects an IP address (the TCP port number is fixed as parameter in the application), and then starts a TCP client which will connect to the TCP server. Note that there is no integration of SIP identities and any other similar protocol. For the demo application, we hard-coded the destination IP addresses in order to facilitate the work of the users. Otherwise, there may try to call wrong IP's. So, after the distant user accepted th call, the application directly opens the call in progress window, depicted in Figure 10.8(b). The communication is currently insecure and both users can communicate. When they decide they can switch to the secure mode. One of the user simply clicks on the "Enter secure mode" button. This will launch the key agreement protocol.

The implemented protocol is depicted in Figure 10.9. We chosen a SAS length of 15 bits, i.e., $\rho = 15$, and clearly the adversary will have a probability of success close to 2^{-15}. Note that it is hard for an adversary to launch multiple instances of the protocol since the users will detect the attack. We fixed the length of K to 30 bits, i.e., $\kappa = 30$, in order to avoid attack on K, clearly the adversary has more chance to targets his attack to the SAS itself. We implemented the commitment with a hash function, here MD5. For that, we need some



(a) (b)

Figure 10.8. *The Main (a) and Call in Progress Windows (b).*

additional randomness in order to hide the value K (since p, g, and y_A are public). We chosen the length of the seed r to be 80-bit long in order to avoid second preimage attacks. At the end, we obtain a DH shared secret key.

It remains for the users to agree on the exchanged DH keys by checking they have the same SAS. At the end of the protocol, the application shows the local SAS, as depicted in Figure 10.10, and each user should accept or reject it. Users should check orally that both have the same SAS, typically, one spells its SAS and the other one agree or disagree.

The protocol continues only if both users accept the SAS. If one rejects its SAS, the application stays in the insecure mode. A user can check if the application encrypt the data or not by watching to the state in the call in progress window, see Figure 10.8(b). The final derived DH key is confidential thanks to the DH protocol and authentic thanks to the PV-SAS-MCA protocol. However, we should obtain an AES secret key. We simply hash the DH key with MD5 and we obtain the desired key. Since now, the call is encrypted with AES and the derived key.

Users leave the insecure mode as soon as one decide to do that.

In conclusion, we have a fully functional voice over IP application. The security association is done end-to-end and users are ensured that nobody spies the conversation in the middle, unlike Skype. Further work may be done in this application to make it compliant with SIP identities. Another approach would be to implement the cryptography part in an existing (insecure) application.

Alice		Bob
	Public: g, p	
Wait for connection		Wait for connection
	$\xrightarrow{\text{open connection}}$	
pick x_A, $y_A \leftarrow g^{x_A}$		pick x_B, $y_B \leftarrow g^{x_B}$
pick $K \in_{\mathsf{u}} \{0,1\}^{30}$		pick $R \in_{\mathsf{u}} \{0,1\}^{15}$
$r \in_{\mathsf{u}} \{0,1\}^{80}$		
$c \leftarrow \mathsf{MD5}(r \Vert p \Vert g \Vert y_A \Vert K)$	$\xrightarrow{y_A, c}$	
	$\xleftarrow{y_B, R}$	
$d \leftarrow (K \Vert r)$	$\xrightarrow{d}$	check $\widehat{c} \stackrel{?}{=} \mathsf{MD5}(\widehat{r} \Vert p \Vert g \Vert \widehat{y}_A \Vert \widehat{K})$
$\mathsf{SAS} \leftarrow \widehat{R} \oplus \mathsf{trunc}_{15}(\mathsf{MD5}(K \Vert \widehat{y}_B))$	$\xrightarrow{\mathsf{auth}_{\mathsf{id}_A}(\mathsf{SAS})}$	$\mathsf{SAS} \stackrel{?}{=} R \oplus \mathsf{trunc}_{15}(\mathsf{MD5}(\widehat{K} \Vert y_B))$
check SAS are the same	$\xleftarrow{\mathsf{auth}_{\mathsf{id}_B}(\mathsf{SAS})}$	
$\mathsf{sk}_A^{\mathsf{AES}} \leftarrow \mathsf{MD5}\left((\widehat{y}_B)^{x_A}\right)$		$\mathsf{sk}_B^{\mathsf{AES}} \leftarrow \mathsf{MD5}\left((\widehat{y}_A)^{x_B}\right)$
final state: $\mathsf{sk}_A^{\mathsf{AES}}, \mathsf{id}_B$		**final state:** $\mathsf{sk}_B^{\mathsf{AES}}, \mathsf{id}_A$

Figure 10.9. *The Implemented PV-SAS-AKA for Secure VoIP.*

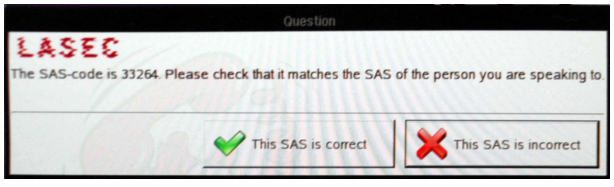


Figure 10.10. *SAS Confirmation (Done on Both Sides).*

Part II

Signatures Schemes

Chapter

ELEVEN

Definitions of Digital Signatures and Interactive Proofs

We start with some generalities on digital signature schemes. In particular, we present different types of signatures that may be interesting in future sections. We also formally define digital signatures, i.e., we present the usual security properties and adversarial models. We finally define the necessary background related to interactive proofs. For instance, we define the notions of proof of knowledge, zero-knowledge proof, and deniability.

11.1 Overview of Digital Signatures

When we talk about cryptography, everyone has in mind data encryption. In other words, the cryptography suggests very often the problem of confidentiality. However, *digital signatures undoubtedly constitute one of the most fundamental tools of public-key cryptography* [Mon06]. A signature scheme allows to bind some information with an identity associated to a public key. Consequently, thanks to the knowledge of a public key and its related identity, a signature scheme allows to authenticate some information. In addition to authentication, thanks to public-key encryption, signature schemes allow to setup a secure communication as seen in Chapter 2 by authenticating public keys.

A Few Examples of Digital Signatures Primitives. The world of digital signatures is huge. Depending on the application requirements, cryptographers developed several models. First, there is the basic signature with a signer and one or several verifiers. The signature should be unforgeable. Then, still in two party settings, there are schemes for which the signature is invisible, undeniable, non-transferable, or any other assumption. There are also schemes dedicated to more than one signer, for instance group signatures or threshold signatures while there are schemes dedicated to allow a third party to prove the correctness of the signature. As said by Cao and Liu [CL08] there exists more than 60 digital signatures models and even cryptography experts have difficulties to understand them due to the various properties. This paragraph presents (informally) some useful digital signature schemes.

Basic digital signature scheme. As explained in Section 2.1.3 and as depicted in Figure 2.6, we have a signer and a verifier. The signer generates a signing key and a verification key. He keeps the signing key private and reveals (and authenticates) the verification key to the verifier. The signer is now able to produce a signature for any message by using his signing key. Then, he gives the message and its signature to the verifier who is able to verify the validity of the pair by using the verification key.

The general security properties are completeness and unforgeability. In short, completeness says that if the signature was produced with the input message and the signing key, then the verifier should accept the pair if it uses the corresponding verification key. Unforgeability says that it is infeasible to produce a pair without the signing key which will be accepted by the verifier.

Multi-signature. In addition to the basic properties, a multi-signature scheme [IN83] allows several people to sign jointly and efficiently the same message. For instance, all committee members of an association sign together a document.

Group signature. In addition to the basic properties, a group signature scheme [CvH91] allows any member of a group to sign a message. The verifier can verify the validity of the signature, i.e., can know that a member of the group signed the message, but the verifier can not know which member signed the message (anonymity). In case of dispute, the scheme has a revocation mechanism which allows a designated group authority to discover the signer's identity.

Threshold signature. In addition to the basic properties, a t-threshold signature [DF90] allows t or more group members to cooperate and to produce a signature on behalf of the group.

Ring signature. As group signatures, ring signatures protect the signer identity behind the group (anonymity). Unlike group signatures, ring signatures [RST01] have no group managers, no setup procedures, no revocation procedures, and no coordination.

Proxy signature. A proxy signature scheme [MUO96] enables a proxy signer to sign messages on behalf of a genuine signer. For instance, a president on vacation can delegate the signature to his vice-president.

Blind signature. A blind signature scheme [Cha84] allows a message owner to get a signature on this message by a signer without revealing any information about the message. The resulting blind signature can be universally verified with the unblinded message in the same manner than a regular digital signature.

One-time signature. A one-time signature scheme only allows the signature of a single message using a given private key.

Fail-stop signature. A fail-stop signature scheme [Pfi91, vHP92] allows the signer to revoke its key. Indeed, if the signer is confronted with an alleged signature that he has not produced, then he can prove that the alleged signature is in fact forged. Afterwards, he can revoke his verifying key, thus the name fail-stop signature scheme.

Invisible (or undeniable) signature. Undeniable signatures [CvA90] are not universally verifiable. Indeed, a signature can only be verified with the help of the signer, i.e., through a verification protocol. This raises a new problem, a dishonest signer may refuse to authenticate a message. For that reason, undeniable signatures have a denial protocol in addition to the verification protocol. The signer cannot falsely deny a valid signature.

Designated-confirmer signature. Here, there are three entities: a signer, a verifier, and a confirmer. A designated-confirmer signature scheme [Cha94] works like an undeniable signature scheme with the main difference that the verification is shifted from the signer to the confirmer. With undeniable signatures, the signer cooperation is essential: if he does not cooperate, the recipient cannot use the signature. Designated-confirmer signatures solve this weakness by adding a third party, the confirmer, able to help the recipient to verify the signature.

Designated-verifier signature (DVS). DVS was motivated by privacy issues associated with disseminations of signatures. A DVS scheme [JSI96] allows a prover to convince a designated-verifier that he has signed a message. The designated-verifier is unable to convince anyone else of this fact, i.e., he cannot transfer the signature to a third party. We say that the signature was *designated* to one verifier.

Universal designated-verified signature (UDVS). An UDVS [SBWP03] can function as a standard publicly-verifiable digital signature but has an additional functionality which allows any holder of a signature to designate it to any desired designated-verifier. As for DVS, the designated-verifier can verify that the message was signed by the signer, but is unable to convince anyone else of this fact. The term *universal* means that *anyone* who posses the signature is able to designate it.

Universal designated-verified signature proof (UDVSP). A drawback with UDVS is that they require the verifier to create a key pair with the same parameters than the signer. In some applications, this is a big restriction. UDVSP [BSNS05] employs an interactive protocol between the signature holder and the verifier allowing to prove the signature validity. The verifier never obtains the signature, he is only convinced of its validity and cannot transfer the proof. The signature holder's privacy is protected and so no verifier public key is necessary.

Many others signature schemes exists. Many of them can be derived from the above examples by combining some properties. For instance, there exists threshold proxy multi-signature [TYH04] which gathers the properties of the above threshold signature, proxy signature, and multi-signature.

In order to survive to these many emerging models, Cao and Liu [CL08] propose a classification method. In short, they classify a signature scheme by a string like $A_aB_bC_cD_dE_e$. The $a \in [0, 26]$ describes a classification based on the signing party, for instance somebody, a group, somebody on behalf of the original signer, anonymous, or many others. The $b \in [0, 28]$, $c \in [0, 8]$, $d \in [0, 1]$, $e \in [0, 1]$ describe a classification based respectively on the verifying party, the lucidity of the content of the message, the method for producing the public key, and the consequence of updating the private key. Thanks to this methodology they are able to classify most of the existing signatures schemes.

11.2 Digital Signature Schemes Formally

We call the *domain* or *message space* the set of all possible input messages and we denote it by $\mathbb{M}$. We call the *signature space* the set of all possible signatures and we denote it by $\mathbb{S}$. We formalize a *digital signature scheme* S by three algorithms:

- The setup algorithm $(K_p, K_s) \leftarrow \mathsf{setup}(1^\lambda)$ generates a public-private key pair depending on a security parameter λ.
- The sign algorithm $\sigma \leftarrow \mathsf{sign}(K_s, m)$ outputs a valid signature $\sigma \in \mathbb{S}$ of a message $m \in \mathbb{M}$ by using the private key K_s.
- The verify algorithm $b \leftarrow \mathsf{verify}(K_p, m, \sigma)$ tells whether the pair (m, σ) is valid or not by using the public key K_p. It returns $b = 1$ if and only if the pair is valid with respect to K_p and $b = 0$ otherwise.

The scheme is said *complete* if for any $(K_p, K_s) \leftarrow \mathsf{setup}(1^\lambda)$, any message $m \in \mathbb{M}$, and any $\sigma \leftarrow \mathsf{sign}(K_s, m)$, then $\mathsf{verify}(K_p, m, \sigma)$ outputs 1.

11.2.1 FML-DS versus AML-DS

We define *fixed message-length digital signature* (FML-DS) schemes any signature scheme which applies only to a restricted domain, e.g., $\mathbb{M} = \{0,1\}^{r(\lambda)}$. As opposite, we define *arbitrary message-length digital signature* (AML-DS) schemes the ones which have an infinite domain, e.g., $\mathbb{M} = \{0,1\}^*$.

An FML-DS can be transformed into AML-DS following the hash-and-sign paradigm. Here, hashing is used as a *domain extender*. For instance, DSA [DSS94, DSS00] is based on SHA-1 [SHA95] while RSA [RSA78] uses MD5 [Riv92] in the standard PKCS #1 v1.5 that is used in X.509 certificates [HFPS99].

11.2.2 Adversarial Model

Here, we adopt the model from Goldwasser, Micali and Rivest [GMR84, GMR88]. We consider an adversary $\mathcal{A}$ playing a game with a challenger $\mathcal{C}$. At the beginning of the game, the challenger $\mathcal{C}$ generates a key pair. Thus, $\mathcal{C}$ knows the private key K_s and is able to produce valid signatures for any messages. The goal for $\mathcal{A}$ is to *break* the signature scheme S by yielding a valid pair $(\widehat{m}, \widehat{\sigma})$ which was not produced by $\mathcal{C}$.

We first note that there exist different degree of freedom associated to adversaries.

Known Message Attacks. A *known message attack* (KMA) only allows the adversary $\mathcal{A}$ to obtain from the challenger $\mathcal{C}$ random messages together with their respective valid signature.

Chosen Message Attacks. A *chosen message attack* (CMA) allows the adversary $\mathcal{A}$ to adaptively choose input messages and then to ask the challenger $\mathcal{C}$ to sign it.

No Message Attacks. A *no message attack* ($\emptyset$MA) does not allow the adversary $\mathcal{A}$ to see a sample pair message-signature.

As well, we distinguish two different goals:

Universal Forgery. An *universal forgery* (UF) implies that the adversary $\mathcal{A}$ found an algorithm which outputs valid forged signatures $\widehat{\sigma}$ for any input message $\widehat{m}$.

Existential Forgery. An *existential forgery* (EF) implies that the adversary $\mathcal{A}$ has output a valid forged signature $\widehat{\sigma}$ for a specific message $\widehat{m}$.

We say that an attack is *successful* if an adversary $\mathcal{A}$ yields a valid signature $\widehat{\sigma}$ for a message $\widehat{m}$ either imposed (UF) or chosen (EF) and $\mathcal{C}$ never signed the message $\widehat{m}$ but only a set of messages $m_1, \ldots, m_\ell$ which are either known (KMA) or chosen (CMA) by the adversary.

Textbook signature schemes such as ElGamal [ElG85] or plain RSA [RSA78] signatures are often existentially forgeable.

In the following we will consider only the *weak* security model, i.e., UF-KMA, and the *strong* security model, i.e., EF-CMA.

Weak Unforgeability: UF-KMA Game. Let $\mathcal{A}$ be any T-time adversary playing the UF-KMA game with a challenger $\mathcal{C}$ as depicted in Figure 11.1. At the beginning of the game, $\mathcal{C}$ generates a key pair and gives the public key K_p to $\mathcal{A}$. Then, $\mathcal{C}$ gives a challenge message $\widehat{m}$ to $\mathcal{A}$. $\mathcal{C}$ also produces samples of signed messages, i.e., some valid signature to *known* messages, and gives them to $\mathcal{A}$. At the end, $\mathcal{A}$ should output a *valid signature* $\widehat{\sigma}$ for the challenge message $\widehat{m}$.

Definition 11.1 (UF-KMA Security).
The signature scheme is said (T, ℓ, ε)-UF-KMA-secure if any T-time adversary $\mathcal{A}$ with ℓ valid signatures on known *messages wins the game of Figure 11.1 with probability at most ε.*

The scheme is said UF-KMA-secure if for any $T = \mathsf{poly}$ and $\ell = \mathsf{poly}$ there exists $\varepsilon = \mathsf{negl}$ such that the scheme is (T, ℓ, ε)-UF-KMA-secure.

$\mathcal{A}$		$\mathcal{C}$
		$(K_p, K_s) \leftarrow \mathsf{setup}(1^\lambda)$
	$\xleftarrow{K_p}$	
		pick $\widehat{m} \in_{\mathsf{u}} \mathbb{M}$
	$\xleftarrow{\widehat{m}}$	
		$\forall i \in [1, \ell]$:
		pick $m_i \in_{\mathsf{u}} \mathbb{M}$
	$\xleftarrow{m_i \Vert \sigma_i}$	$\sigma_i \leftarrow \mathsf{sign}(K_s, m_i)$
select $\widehat{\sigma}$	$\xrightarrow{\widehat{\sigma}}$	$b \leftarrow \mathsf{verify}(K_p, \widehat{m}, \widehat{\sigma})$

$\mathcal{A}$ wins if $b = 1$ and $\widehat{m} \notin \{m_1, \ldots, m_\ell\}$.

Figure 11.1. *The UF-KMA Game.*

Strong Unforgeability: EF-CMA Game. Let $\mathcal{A}$ be any T-time adversary playing the EF-CMA game with a challenger $\mathcal{C}$ as depicted in Figure 11.2. At the beginning of the game, $\mathcal{C}$ generates a key pair and gives the public key K_p to $\mathcal{A}$. Then, $\mathcal{A}$ is allowed to query (adaptively) $\mathcal{C}$ with *chosen* messages in order to obtain their (genuine) signature. At the end, $\mathcal{A}$ should outputs a *forged valid pair* message-signature $(\widehat{m}, \widehat{\sigma})$.

Definition 11.2 (EF-CMA Security).
The signature scheme is said (T, ℓ, ε)-EF-CMA-secure if any T-time adversary $\mathcal{A}$ bounded

by ℓ signature requests on chosen messages wins the game of Figure 11.2 with probability at most ε.

The scheme is said EF-CMA-secure if for any $T = \mathsf{poly}$ and $\ell = \mathsf{poly}$ there exists $\varepsilon = \mathsf{negl}$ such that the scheme is (T, ℓ, ε)-EF-CMA-secure.

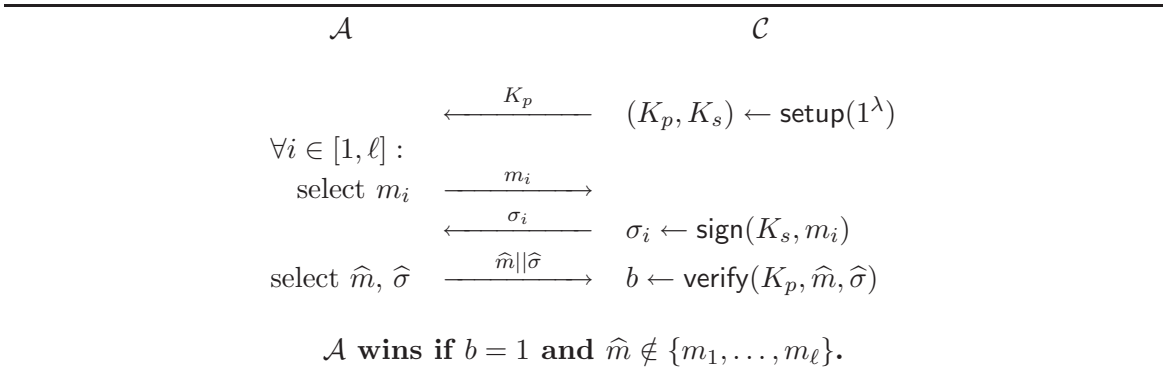


Figure 11.2. *The EF-CMA Game.*

11.3 Interactive Proofs (in the Standard Model)

In this section, we recall some basic material required to understand the concept of zero-knowledge proof of knowledge. To manage simple definitions, we restrict to the standard model and we will extend the definitions later. Most of the material in this section was taken from the book of Oded Goldreich [Gol01].

11.3.1 Binary Relation and Binary Language

Hard problems in cryptography may be described by a binary relation. As example, we take the discrete logarithm problem: Let p be a prime number, $\mathcal{G} \subseteq \mathbb{Z}_p$ be a group, and g be a generator of $\mathcal{G}$ of order q. Given a random $w \in [0, \ldots, q-1]$, we compute the pair (x, w) with $x = (p, q, g, W)$ and $W = g^w \in \mathcal{G}$. Clearly, given w it is easy to compute W but not reversely. We denote by $\mathsf{DL}_g(W)$ the discrete logarithm of W in $\mathcal{G}$ with respect to the generator g, in this case w. For fixed group $\mathcal{G}$ and generator g, we build a relation R as follows :

$$R_{DL} = \{(x, w) \ : \ x = (p, q, g, W), \mathsf{ord}(g) = q, W = g^w \pmod p\} \ .$$

Formally, let $R \subseteq \{0,1\}^* \times \{0,1\}^*$ be a binary relation with a polynomial-size witness, i.e., for any $(x, w) \in R$ we have $|w| \leq \mathsf{poly}(|x|)$. For some $(x, w) \in R$, we can see x as an

instance of some computational problem and w as the solution to that instance. w is called a *witness* for x.

Let L_R be a language related to the binary relation R. L_R is the set of all x such that there exists a witness w and (x, w) is in R, i.e.,

$$L_R = \{x \ : \ \exists w \text{ such that } (x, w) \in R\} \ .$$

Let $R(x)$ denote the set of all witnesses for $x \in L_R$, i.e.,

$$R(x) = \{w \ : \ (x, w) \in L_R\} \ .$$

11.3.2 Interactive Turing Machines

An interactive Turing machine (ITM) is roughly a classical Turing machine with additional tapes for communications. More precisely, an ITM is a deterministic machine with

- a read-only input tape,
- a read-only random tape,
- a read-only input communication tape,
- a read-write work tape,
- a write-only output communication tape,
- and a write-only output tape.

In general we do not consider an interactive Turing machine alone but a pair of machines. Indeed, they are linked together to form an *interactive system* as depicted in Figure 11.3.

Let A and B be two interactive Turing machines. They are able to communicate through their communication tapes: the input communication tapes of each machine coincides with the output communication tape of the other one. When machine A sends a message, it writes the message on its output communication tape, this is automatically written on the input communication tape of the machine B who finally reads it and thus B receives the message.

11.3.3 Interactive Proof Systems

Let $(\mathbf{P}, \mathbf{V})$ be a pair of interactive Turing machines (ITMs). We want that the prover $\mathbf{P}$ is able to convince the verifier $\mathbf{V}$ of the validity of a true statement. For a given x, the prover $\mathbf{P}$ wants to prove to the verifier $\mathbf{V}$ that x belongs to the language L, i.e., $x \in L$. For that,

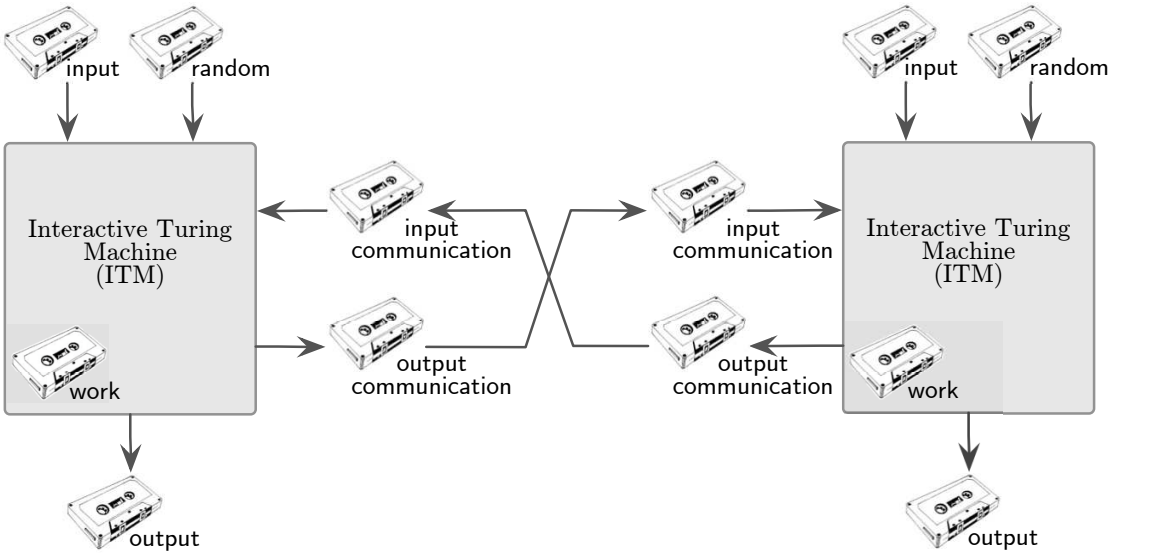


Figure 11.3. *Two Connected Interactive Turing Machines: an Interactive System.*

P and **V** will use an *interactive proof* with common input x denoted by $\mathsf{proof}_{\mathbf{P},\mathbf{V}}(x)$. At the end, **V** outputs accept or reject. We want that **P** was able to convince **V** that $x \in L$. However, we want that nobody can fool **V** into believing a false statement, i.e., that $x \in L$ while it is false.

Definition 11.3 (Interactive Proof System).
Let $c, s : \mathbb{N} \to [0,1]$ *be two functions such that* $c(n) > s(n)+1/\mathsf{poly}(n)$ *for some polynomial* $\mathsf{poly}(\cdot)$. *Let* $(\mathbf{P},\mathbf{V})$ *be a pair of interactive Turing machines (ITMs).* $\mathsf{proof}_{\mathbf{P},\mathbf{V}}(\cdot)$ *is an* interactive proof system *for the language* L *with completeness bound* $c(n)$ *and soundness bound* $s(n)$ *if the following conditions hold:*

- Efficiency: **P** *is unbounded,* **V** *runs in polynomial-time.*
- Completeness: *For any* $x \in L$, **V** *accepts in* $\mathsf{proof}_{\mathbf{P},\mathbf{V}}(x)$ *with probability at least* $c(|x|)$.
- Soundness: *For any input* $x \notin L$ *and any interactive machine* $\mathbf{P}^*$, **V** *accepts in* $\mathsf{proof}_{\mathbf{P}^*,\mathbf{V}}(x)$ *with probability at most* $s(|x|)$.

11.3.4 Proof of Knowledge

Here we recall the definition of a proof of knowledge originally formalized in the plain model by Bellare and Goldreich [BG93].

In order to formalize the notion of proof of knowledge we first need to introduce the

concept of *knowledge extractor* Ext. The Ext algorithm gets input x and authorization to access the prover, while he attempts to compute w such that $(x, w) \in R$.

Definition 11.4 (Proof of Knowledge).
Let $(\mathbf{P}, \mathbf{V})$ *be a pair of interactive Turing machines (ITMs).* $\mathsf{proof}_{\mathbf{P}(w),\mathbf{V}}(x)$ *is a* proof of knowledge *for the relation* R *with soundness error* $\kappa(x)$ *if the following conditions hold:*

- Efficiency: $\mathsf{proof}_{\mathbf{P},\mathbf{V}}(\cdot)$ *is polynomially bounded, and* $\mathbf{P}$ *and* $\mathbf{V}$ *are polynomial-time.*
- Completeness: *On common input* x*, if there exists a witness* w *such that* $(x, w) \in R$*, then* $\mathbf{V}$ *in* $\mathsf{proof}_{\mathbf{P}(w),\mathbf{V}}(x)$ *always* accept.
- Soundness: *Given a* PPT *machine* $\mathbf{P}^*$*, let* $\varepsilon(x)$ *be the probability that* $\mathbf{V}$ *accepts in* $\mathsf{proof}_{\mathbf{P}^*,\mathbf{V}}(x)$*. There exists an extractor* Ext *and a constant* k *such that for any* $\mathbf{P}^*$*, and any* x*, if* $\varepsilon(x) > \kappa(x)$*, then* $\mathsf{Ext}^{\mathbf{P}^*}(x)$ *outputs a witness* w *such that* $(x, w) \in R$ *within expected time*
$$\mathcal{O}\left(\frac{|x|^k}{\varepsilon(x) - \kappa(x)}\right)$$
where an access to $\mathbf{P}^*$ *only counts as one step.*

$\kappa(x)$ represents the probability that a malicious prover is able to convince an honest verifier (without knowing the correct witness w corresponding to x) .

11.3.5 Zero-Knowledge

Consider any proof of knowledge between a prover $\mathbf{P}$ and a verifier $\mathbf{V}$. The notion of *zero-knowledge* underlines the fact that no information leaks to the verifier except the validity of the statement. The main idea behind zero-knowledge is that any verifier should be able to run the simulator $\mathsf{Sim}_{\mathsf{zk}}$ by himself (instead of interacting with a prover). The first consequence it that anybody may have generated the proof by himself. The second consequence is that the transcript of the proof is by the way not a proof of an interaction with the prover. This concept was formalized by Goldwasser, Micali, and Rackoff [GMR85, GMR89]. Here, we recall the definitions from Barak, Lindell, and Vadhan [BLV03, BLV04] and we distinct two cases: in the first one, the protocol is zero-knowledge until the verifier remains honest, while, in the second one, the protocol is zero-knowledge for any (honest or malicious) verifier.

Definition 11.5 (Honest-Verifier Zero-Knowledge).
A proof of knowledge $\mathsf{proof}_{\mathbf{P}(w),\mathbf{V}(z)}(x)$ *for a relation* R *is* honest-verifier zero-knowledge *(HVZK) if there exists a* PPT *simulator* $\mathsf{Sim}_{\mathsf{zk}}$ *such that*
$$\left\{\mathsf{View}_{\mathbf{V}}\left(\mathsf{proof}_{\mathbf{P}(w),\mathbf{V}(z)}(x)\right)\right\}_{z\in\{0,1\}^*, x\in L_R} \text{ for arbitrary } w \in R(x)$$

and

$$\left\{\mathsf{Sim}_{\mathsf{zk}}(x,z)\right\}_{z\in\{0,1\}^*,x\in L_R}$$

are computationally indistinguishable.

Definition 11.6 (Zero-Knowledge).
A proof of knowledge $\mathsf{proof}_{\mathbf{P}(w),\mathbf{V}(z)}(x)$ *for a relation* R *is* zero-knowledge *(ZK) if for any* PPT $\mathbf{V}^*$*, there exists a* PPT *simulator* $\mathsf{Sim}_{\mathsf{zk}}$ *such that*

$$\left\{\mathsf{View}_{\mathbf{V}^*}\left(\mathsf{proof}_{\mathbf{P}(w),\mathbf{V}^*(z)}(x)\right)\right\}_{z\in\{0,1\}^*,x\in L_R} \text{ for arbitrary } w\in R(x)$$

and

$$\left\{\mathsf{Sim}_{\mathsf{zk}}(x,z)\right\}_{z\in\{0,1\}^*,x\in L_R}$$

are computationally indistinguishable.

11.4 Interactive Proofs in the Common Reference String Model

In order to prove the security of a new design, we often need keyed cryptographic primitives. The consequence is that parties involved in the execution of a protocol rely on a public key. In practice we want to avoid the generation of public/private keys as well as their transmission. The *Common Reference String* (CRS) model, see Section 3.4, assumes that all implementations use the same trusted public key, denoted by crs, and the corresponding private key is unknown by everyone. In the following, we recall some definitions from Pass [Pas03, Pas04]. Note that in these definitions crs is assumed to be uniformly distributed.

First, we consider any proof of knowledge between a prover $\mathbf{P}$ and a verifier $\mathbf{V}$ but now in the CRS model.

Definition 11.7 (Proof of Knowledge in the CRS model).
Let $(\mathbf{P},\mathbf{V})$ *be a pair of interactive Turing machines (ITMs),* crs *be any uniformly distributed CRS, and* $\kappa(x)$ *be a real valued function.* $\mathsf{proof}_{\mathbf{P}(w),\mathbf{V}}(x,\mathsf{crs})$ *is a* proof of knowledge in the CRS model *for the relation* R *with soundness error* $\kappa(x)$ *if the following conditions hold:*

- Efficiency: $\mathsf{proof}_{\mathbf{P},\mathbf{V}}(\cdot)$ *is polynomially bounded, and* $\mathbf{P}$ *and* $\mathbf{V}$ *are polynomial-time.*
- Completeness: *On common input* (x,crs)*, if there exists a witness* w *such that* $(x,w)\in R$*, then* $\mathbf{V}$ *in* $\mathsf{proof}_{\mathbf{P}(w),\mathbf{V}}(x,\mathsf{crs})$ *always* accept*.*
- Soundness: *Given a* PPT *machine* $\mathbf{P}^*$*, let* $\varepsilon(x)$ *be the probability that* $\mathbf{V}$ *accepts in* $\mathsf{proof}_{\mathbf{P}^*,\mathbf{V}}(x,\mathsf{crs})$*. There exists an extractor* Ext *and a constant* k *such that for*

any $\mathbf{P}^*$*, and any* x*, if* $\varepsilon(x) > \kappa(x)$*, then* $\mathsf{Ext}^{\mathbf{P}^*}(x)$ *outputs a witness* w *such that* $(x, w) \in R$ *within expected time*

$$\mathcal{O}\left(\frac{|x|^k}{\varepsilon(x) - \kappa(x)}\right)$$

where an access to $\mathbf{P}^*$ *only counts as one step.*

Here we extend the notion of zero-knowledge from the standard model to the CRS model. Clearly, now, all participants have access to the (public) string crs.

Definition 11.8 (Zero-Knowledge in the CRS Model).
Let crs *be any uniformly distributed common reference string (CRS). A proof of knowledge* $\mathsf{proof}_{\mathbf{P},\mathbf{V}}(\cdot)$ *for a relation* R *is* zero-knowledge in the CRS model *if for any* PPT $\mathbf{V}^*$*, there exists a* PPT *simulator* $\mathsf{Sim}_{\mathsf{zk}}$ *such that*

$$\left\{\mathsf{View}_{\mathbf{V}^*}\left(\mathsf{proof}_{\mathbf{P}(w),\mathbf{V}^*(z)}(x, \mathsf{crs})\right)\right\}_{z\in\{0,1\}^*, x\in L_R} \text{ for arbitrary } w \in R(x)$$

and

$$\left\{\mathsf{Sim}_{\mathsf{zk}}(x, z)\right\}_{z\in\{0,1\}^*, x\in L_R}$$

are computationally indistinguishable.

11.5 Deniability in Zero-Knowledge Proofs

As said before, the main idea behind zero-knowledge is that any verifier should be able to run the simulator by himself (instead of interacting with a prover). However in the CRS model, the simulator $\mathsf{Sim}_{\mathsf{zk}}$ is able to choose the crs public key while no verifier is able to do that in reality. We conclude that the above standard definition for zero-knowledge in the CRS model is not exactly what we are looking for.

Here we recall the concept of *deniability*. Informally, deniability means that any transcript of protocol should not yield an evidence of interaction with $\mathbf{P}$. In the CRS model, deniable zero-knowledge is different from the (standard) zero-knowledge in the sense that in the deniable case the simulator $\mathsf{Sim}_{\mathsf{zk}}$ is not able to choose crs. This difference solves the above issue.

Definition 11.9 (Deniable Zero-Knowledge in the CRS Model).
Let crs *be any uniformly distributed common reference string (CRS). A proof of knowledge* $\mathsf{proof}_{\mathbf{P},\mathbf{V}}(\cdot)$ *for a relation* R *is* deniable zero-knowledge in the CRS model *if for any* PPT $\mathbf{V}^*$*, there exists a* PPT *simulator* $\mathsf{Sim}_{\mathsf{zk}}$ *such that*

$$\left\{\mathsf{crs}, \mathsf{View}_{\mathbf{V}^*}\left(\mathsf{proof}_{\mathbf{P}(w),\mathbf{V}^*(z)}(x, \mathsf{crs})\right)\right\}_{z\in\{0,1\}^*, x\in L_R} \text{ for arbitrary } w \in R(x)$$

and

$$\left\{\mathsf{crs}, \mathsf{Sim}_{\mathsf{zk}}(x, z, \mathsf{crs})\right\}_{z\in\{0,1\}^*, x\in L_R}$$

are computationally indistinguishable.

While the above definition applies for the CRS model, we note that it is the same case in the random oracle (RO) model. Indeed, in the definition of zero-knowledge in the random oracle model, the simulator $\mathsf{Sim}_{\mathsf{zk}}$ is able to choose the random oracle H. As in the CRS model, in reality it is not the case.

Here, we give a generic definition for deniable zero-knowledge which is effective in *all three models*: the standard model, the common reference string model, and the random oracle model.

Definition 11.10 (Deniable Zero-Knowledge Proof of Knowledge).
Let crs *be any common reference string (CRS). Let* H *be a random oracle. Let* $\kappa(x)$ *be a real valued function.* $\mathsf{proof}_{\mathbf{P}^{\mathsf{H}}(w),\mathbf{V}^{\mathsf{H}}}(x, \mathsf{crs})$ *is a* proof of knowledge *for the relation* R *with soundness error* $\kappa(x)$ *if the following holds :*

- Efficiency: $\mathsf{proof}_{\mathbf{P},\mathbf{V}}(\cdot)$ *is polynomially bounded, and* $\mathbf{P}$ *and* $\mathbf{V}$ *are polynomial-time.*
- Completeness: *On common input* (x, crs)*, if there exists a witness* w *such that* $(x, w) \in R$*, then* $\mathbf{V}$ *in* $\mathsf{proof}_{\mathbf{P}^{\mathsf{H}}(w),\mathbf{V}^{\mathsf{H}}}(x, \mathsf{crs})$ *always* accept*.*
- Soundness: *Given a* PPT *machine* $\mathbf{P}^{*\mathsf{H}}$*,* crs*, and* H*, let* $\varepsilon(x)$ *be the probability that* $\mathbf{V}$ *accepts in* $\mathsf{proof}_{\mathbf{P}^{\mathsf{H}}(w),\mathbf{V}^{\mathsf{H}}}(x, \mathsf{crs})$*. There exists an extractor* Ext *and a constant* k *such that for any* crs*, any* H*, any* $\mathbf{P}^*$*, and any* x*, if* $\varepsilon(x) > \kappa(x)$*, then* $\mathsf{Ext}^{\mathbf{P}^*}(x)$ *outputs a witness* w *such that* $(x, w) \in R$ *within expected time*

$$\mathcal{O}\left(\frac{|x|^k}{\varepsilon(x) - \kappa(x)}\right)$$

where an access to $\mathbf{P}^*$ *only counts as one step.*

A proof of knowledge $\mathsf{proof}_{\mathbf{P}^{\mathsf{H}},\mathbf{V}^{\mathsf{H}}}(\cdot)$ *for a relation* R *is* deniable zero-knowledge *if for any* PPT $\mathbf{V}^*$*, there exists a* PPT *simulator* $\mathsf{Sim}_{\mathsf{zk}}^{\mathsf{H}}$ *(with access to* H*) such that*

$$\left\{\mathsf{crs}, \mathsf{H}, \mathsf{View}_{\mathbf{V}^*}\left(\mathsf{proof}_{\mathbf{P}^{\mathsf{H}}(w),\mathbf{V}^{*\mathsf{H}}(z)}(x, \mathsf{crs})\right)\right\}_{z\in\{0,1\}^*, x\in L_R} \text{ for arbitrary } w \in R(x)$$

and

$$\left\{\mathsf{crs}, \mathsf{H}, \mathsf{Sim}_{\mathsf{zk}}^{\mathsf{H}}(x, z, \mathsf{crs})\right\}_{z\in\{0,1\}^*, x\in L_R}$$

are computationally indistinguishable.

Clearly, when crs and H are constant and H is polynomially computable, we obtain the definition in the standard model. When only crs is constant, we obtain the random oracle model. When only H is constant and polynomially computable, we obtain the CRS model. When $\mathbf{V}^*$ is restricted by $\mathbf{V}$, i.e., $\mathbf{V}^* = \mathbf{V}$, we obtain the honest verifier zero-knowledge (HVZK) definition.

It seems that the need for deniability makes the CRS model collapse down to the plain model (see Pass [Pas04]). Indeed, there exists an efficient generic transformation of deniable zero-knowledge protocols from the CRS model into the plain model. However this transformation adds some more rounds which increases the round complexity.

So, deniable ZK protocols in the CRS model may still be attractive in practice.

Chapter

TWELVE

Preserving the Privacy of Signed Documents

Consider Alice has a pair of keys, i.e., a private signing key and a public verification key. Alice would like to prove to Bob the authenticity of one of her documents d. She signs d using her signing key, obtains the signature σ, and sends both d and σ to Bob. If the signature is universally verifiable, Bob can now check the validity of the document d by verifying that the signature σ is correct (with respect to the verification key from Alice).

The above scenario is occurring many times every day, but we often neglect that Bob can publish the document d and its corresponding signature σ. Proceeding in that way, all the world may know the existence of d, but worse the world is also convinced that the document d was treated as is by Alice. At the beginning, Alice just wanted to prove the authenticity of her document d to Bob and not to the rest of the world! One of the issues of digital signatures is that the signature may prove the authenticity of private document and in some situations this transferability leads to privacy issues.

Monnerat, Vaudenay, and Vuagnoux [MVV07, Vau07, VV07] studied the e-passport standards and identified the privacy issue from leaking of signatures on private data. See Section 12.7 for more details on e-passports. For instance, data such as official name, true date of birth, citizenship and a facial image together with a digital signature could easily be released on the Internet or sold by a malicious verifier, allowing to convince anybody of their authenticity. None of these data is really confidential. For instance, the *true* date of birth of some person can fairly be estimated or propagated by gossiping. The person can still claim that the gossiped date of birth is incorrect and keep it private. What is more sensitive is

a *proof* that a date of birth is true because the person can no longer deny evidence that the proof is correct. For this reason, the authors suggested to use non-transferable proof of signature knowledge. So, the passport can convince the border patrol of the authenticity of the data without revealing the signature and, therefore, strengthening privacy.

In our scenario, we have a signer (the national authority), a prover (the e-passport), and a verifier (the border patrol). Clearly, the passport should not know the signing key which is kept secret by the national authority.

Full non-transferability requires a public-key infrastructure (PKI) for verifiers. Certificate verification should be secured to avoid transfer attacks using rogue keys. The EAC standard [MRT04b] proposes the use of a PKI for border patrols, however this is not enough for the privacy issues we have in mind. One problem thought is that this PKI is meant for access control of more sensitive data or services. Although, basic access shall be widely available. In addition, the key of verifiers in EAC will only be checked for verifiers in a country with agreements with the home country. That is, non-transferability would only be enforced in friendly countries but not in others. This seems pretty weird.

For these reasons, we want to avoid PKI for verifiers and we will focus on a weaker form of non-transferability which holds after the protocol is complete. That is, we enforce offline non-transferability which is equivalent to deniability (sometimes called self-simulatability). Zero-knowledge proofs are inherently deniable in the plain model while zero-knowledge protocols in the common reference string (CRS) model or the random oracle model (ROM) are not necessarily deniable. However, protocols in the CRS or ROM are more attractive for efficiency reasons. So, we have to consider the notion of deniable zero-knowledge [Pas03, Pas04].

The above reasons motivated the study of non-transferable proof of signature knowledge with a strong focus on the efficiency. The goal is to find a protocol which can be implemented on e-passports (see Section 12.7) for proving the knowledge of a valid signature to a border patrol in a setting where there is no PKI for border patrols and these may be dishonest. Protocols for proof of knowledge are already well studied, see [CDM00]. Here, we try to increase the efficiency and inter-operability to accommodate popular standards of digital signature. The international e-passport standard [MRT04a] proposes the use of RSA- and ElGamal-based signature schemes. The EAC [MRT04b] extension suggests that e-passports are able to run ECDH protocols. So, there is a little place for public-key cryptography.

In this chapter, we propose a solution to protect the privacy. Our solution consists for Alice in proving to Bob that the signature is valid instead of revealing it. So, in Section 12.3, we introduce the definition of an offline non-transferable authentication protocol (ONTAP). We propose a generic transform of a signature scheme into an ONTAP by using a deniable ZK proof of knowledge. In order to build secure ONTAP, we study strong constructions of proof of knowledge in Section 12.4. In particular, we study a generic transform of Σ-protocols into this type of interactive proof. Unsurprisingly, our generic transform just adds a commit message at the beginning and allows to avoid vulnerability with respect to malicious verifiers.

In Section 12.5, we apply ONTAP to standard signature like RSA- and ElGamal-based schemes. We propose ONTAP protocols which can be efficiently executed in constrained environments such as e-passports. In particular, we give an efficient example based on the Guillou-Quisquater protocol for RSA-based signatures and another example based on the Schnorr protocol for ElGamal-based signatures (being compatible with PKCS#1v1.5, ISO/IEC 9796, RSA-PSS, Schnorr, DSA, ECDSA, and others). Finally, to motivate our constructions, we give a short overview on e-passports in Section 12.7.

We published the ONTAP primitive as well as the efficient RSA- and ElGamal-based implementations in [MPV09].

12.1 Related Work

Non-transitive signatures [Des88, OO91] and deniable message authentication [DDN03] also deal with transferability issues but do not immediately apply for a three-party settings where an intermediate player (e.g., the e-passport) shows to another one that some data was authenticated by an authority. Remember that the e-passport should be able to prove the validity of a signature without knowing the private signing key (which will be the national authority private key).

Undeniable signatures [CvA90, GKR00, MV04] only consider a two-party setting. In order to confirm or deny a signature, the prover must know the secret key. Clearly, a passport cannot know the authority secret key.

With undeniable signatures, the signer cooperation is essential. This motivated the introduction of designated confirmer signatures but mostly to protect the verifier from signers who would refuse to participate in verification protocols. This is not our case and the security definitions are very different since they protect against different threats.

Asokan, Shoup, and Waidner [ASW98, ASW00] proposed a solution to cross-exchange signatures between two parties in a fair way (using a trusted third party in a fair way). For that, they propose a way to transform a signature scheme into a verifiable escrow scheme. A verifiable escrow scheme is based on a homomorphism and allows to produce an escrow signature from a signature. Then, given the escrow signature, one can verify that it is really a signature without obtaining the signature. Finally, someone can recover the signature from the escrow signature by using the secret key. One drawback for our application is that the escrow signature is verifiable, thus it is some kind of signature and not deniable.

Non-transferability was studied by Jakobsson *et al.* [JSI96, CM00] and they introduced designated-verifier proofs. The idea is to designate the signature to a verifier (using its public-key) and then only the designated-verifier can be convinced on the validity of the signature. One drawback is that the verifier must be known at the signature time. Later, Steinfeld *et al.* [SBWP03] introduced Universal Designated-Verifier Signature (UDVS). This

scheme applies to a three-party setting: signer, designator, and verifier. *Universal* refers to that any designator who obtained a universally verifiable signature from the signer is able to designate it to a verifier. This relies on a PKI for verifiers. Compared to designated-verified proofs, an UDVS allows to produce signatures without knowing the verifier public-key and the signature will be designated to one verifier later (which is not possible with designated-verifier proofs). However, the method for having every verifier attached to a public key is an overkill. This motivated Baek *et al.* [BSNS05] to define a weaker notion of non-transferability and they published the Universal Designated Verifier Signature Proof (UDVSP). The primitive is similar to the concept of UDVS except that no signature is given to the verifier. The designator does not need to know the verifier, only a signature proof is given to the verifier. This primitive assumes that verifiers are honest. Clearly, our application scenario does not meet this assumption and this construction does not remain secure when the verifiers may be malicious [LW06, SSNB07]. In addition, the proposed UDVS or UDVSP constructions rely on bilinear mappings which seems not very easy to implement in the case of e-passports.

Recently, Shahandashti, Safavi-Naini, and Baek [SSNB07] worked on Credential Ownership Proofs (COP). There have some similar features as non-transferable signature. However, COP allow users to copy/share the credits which is clearly not desirable in the case of e-passports. They also protect against "double spending" which is not necessary in our case.

12.2 On Non-Transferability

Consider a binary relation R with elements of the form (x, w) and any prover $\mathbf{P}$ able to prove his knowledge of some witness w. *Non-transferability* aims to prevent a malicious verifier $\mathbf{V}^*$ to convince another party, say $\widetilde{\mathbf{V}}$, that $\mathbf{P}$ knows the witness w.

A zero-knowledge protocol is necessary but not sufficient to avoid the transfer of a proof. Namely, a straightforward way to transfer a proof may be achieved by $\mathbf{V}^*$ in forwarding messages from $\mathbf{P}$ to $\widetilde{\mathbf{V}}$ and vice versa. $\mathbf{P}$ does not notice that he is convincing $\widetilde{\mathbf{V}}$ instead of $\mathbf{V}^*$. Such an attack is often called a *relay attack*, also known as a *mafia fraud attack*.

We propose two flavors of non-transferability: *online* versus *offline* non-transferability. Both avoid the transfer of the proof after the proof terminated but only *online* non-transferability forbids the transfer of the proof during the execution of the protocol. We may also consider a *semi-offline* case. In this situation, $\mathbf{V}^*$ and $\widetilde{\mathbf{V}}$ are allowed to communicate before but *not* during the protocol execution between $\mathbf{V}^*$ and $\mathbf{P}$. Since $\mathbf{V}^*$ and $\widetilde{\mathbf{V}}$ have no information before the protocol execution, talking before is equivalent to talking after the protocol. So, the semi-offline case is equivalent to the offline one.

Definition 12.1 (Non-Transferability).
Consider an honest prover $\mathbf{P}$ *with respect to an interactive (2-party) protocol* proof *for a*

binary relation R.

We say that proof *is* online non-transferable *if for any interactive* PPT $\mathbf{V}^*$ *there exists a* PPT Sim*, called the* simulator*, such that for any interactive* PPT $\widetilde{\mathbf{V}}$ *the two following protocol views*

$$\mathsf{View}_{\widetilde{\mathbf{V}}}\left(\mathbf{P}(x, K_p^{\mathbf{V}^*}, w) \leftrightarrow \mathbf{V}^*(x, K_p^{\mathbf{V}^*}, K_s^{\mathbf{V}^*}) \leftrightarrow \widetilde{\mathbf{V}}(x, K_p^{\mathbf{V}^*})\right)$$

and

$$\mathsf{View}_{\widetilde{\mathbf{V}}}\left(\mathsf{Sim}(x, K_p^{\mathbf{V}^*}, K_s^{\mathbf{V}^*}) \leftrightarrow \widetilde{\mathbf{V}}(x, K_p^{\mathbf{V}^*})\right).$$

indexed by all pairs $(x, w) \in R$ are indistinguishable with respect to $|x|$[1], where $K_p^{\mathbf{V}^*}$ and $K_s^{\mathbf{V}^*}$ represent the public and private keys of $\mathbf{V}^*$.

We say that proof *is* offline non-transferable *if it satisfies the same property under the additional rule that $\mathbf{V}^*$ and $\widetilde{\mathbf{V}}$ are* only *allowed to communicate after the interaction between $\mathbf{V}^*$ and $\mathbf{P}$ is complete.*

In Definition 12.1 we consider non-transferability only for the pairs in the relation. In our context, we do not care about a pair $(x, w) \notin R$, since it corresponds to an invalid signature. Namely, transferring the fact that a prover possesses an invalid signature is without interest.

Before, we saw that the zero-knowledge property on the interactive protocol is a necessary but not sufficient condition to achieve non-transferability. It is true for online non-transferability, but in reality a zero-knowledge protocol is enough to ensure *offline* non-transferability.

12.3 Offline Non-Transferable Authentication Protocol (ONTAP)

As said before, one of the issues of digital signatures is that the signature may prove the authenticity of a private document. By consequent, digital signatures may lead to privacy issues. In this section, we propose a solution to protect the privacy which consists in proving the knowledge of the signature instead of revealing the signature itself.

Our definition simplifies and strengthens the definition of Baek *et al.* [BSNS05] to address offline non-transferability with a malicious verifier.

Definition 12.2 (ONTAP).
We define an offline non-transferable authentication protocol *(ONTAP) by the two following algorithms and the interactive verification protocol:*

[1]For instance, the statistical distance or the advantage of the best distinguisher should be negligible in $|x|$.

- *The* $(K_p, K_s) \leftarrow \mathsf{setup}(1^\lambda)$ *algorithm generates a key pair given a security parameter* λ*.*
- *The* $\sigma = (\sigma_p, \sigma_s) \leftarrow \mathsf{sign}(K_s, m)$ *algorithm outputs a signature* $\sigma \in \mathbb{S}$ *of a message* $m \in \mathbb{M}$*.* σ *is split in two parts: a public part* σ_p *and a private part* σ_s*.*
- *The* $\mathsf{iProof}_{\mathbf{P}(\sigma_s),\mathbf{V}}(K_p, m, \sigma_p)$ *protocol allows a prover* $\mathbf{P}$ *to convince a verifier* $\mathbf{V}$ *that he knows a* σ_s *to complete* σ_p *in a valid signature for* m*. At the end,* $\mathbf{V}$ *accepts or rejects.*

The scheme is complete *if for any* $(K_p, K_s) \leftarrow \mathsf{setup}(1^\lambda)$*, any message* $m \in \mathbb{M}$*, and any* $(\sigma_p, \sigma_s) \leftarrow \mathsf{sign}(K_s, m)$*, then* $\mathbf{V}$ *always accepts in the* $\mathsf{iProof}_{\mathbf{P}(\sigma_s),\mathbf{V}}(K_p, m, \sigma_p)$ *protocol.*

The UDVSP [BSNS05] uses a KeyGen algorithm which is equivalent to our setup algorithm. The Sign algorithm outputs a classical signature universally verifiable by using the Verify algorithm, there is a Transform algorithm which generates a modified signature (with a public and secret part) from the universally verifiable one. Our sign algorithm may be built with the Sign and Transform algorithms from the UDVSP and conversely. We removed the Verify algorithm since it is useless with our definition. Finally, there is an interactive proof IVerify as our iProof. So, the two definitions are conceptually equivalent. The main difference comes from the security requirements.

The ONTAP is secure if it satisfies the next two definitions.

Definition 12.3 (Offline Non-Transferability of ONTAP).
Consider an adversary $\mathcal{A}$ *against the ONTAP.* $\mathcal{A}$ *plays a game with a challenger* $\mathcal{C}$*. The goal of* $\mathcal{A}$ *is to get evidence that some message* $\widehat{m}$ *was signed. During a training phase,* $\mathcal{A}$ *is allowed to query a sign oracle denoted* Sign*. After the training phase,* $\mathcal{A}$ *selects some* $\widehat{m}$*,* $\mathcal{C}$ *signs it and reveals* $\widehat{\sigma}_p$*. Then,* $\mathcal{A}$ *runs a session of* $\mathsf{iProof}_{\mathbf{P}(\widehat{\sigma}_s),\mathcal{A}}(K_p, \widehat{m}, \widehat{\sigma}_p)$ *protocol. At the end of the game,* $\mathcal{A}$ *outputs all inputs queried to* Sign *and its state* λ*.*

We introduce Sim *which plays the same game but selects no* $\widehat{m}$ *and runs no* iProof *protocol.*

The ONTAP scheme is said offline non-transferable *if for any adversary* $\mathcal{A}$ *there exists a simulator* Sim *such that their output in the game of Figure 12.1 are computationally indistinguishable.*

Definition 12.4 (Unforgeability of ONTAP).
Consider an adversary $\mathcal{A}$ *against the ONTAP.* $\mathcal{A}$ *plays a game with a challenger* $\mathcal{C}$*. The goal of* $\mathcal{A}$ *is to convince* $\mathcal{C}$ *by running the* iProof *protocol that he knows* $\widehat{\sigma}_s$ *to complete* $\widehat{\sigma}_p$ *in a valid signature for* $\widehat{m}$*.*

During a training phase, $\mathcal{A}$ *is allowed to query a sign oracle denoted* Sign*. On input message* $m \in \mathbb{M}$*,* Sign *answers the complete valid signature* (σ_p, σ_s)*. After this training*

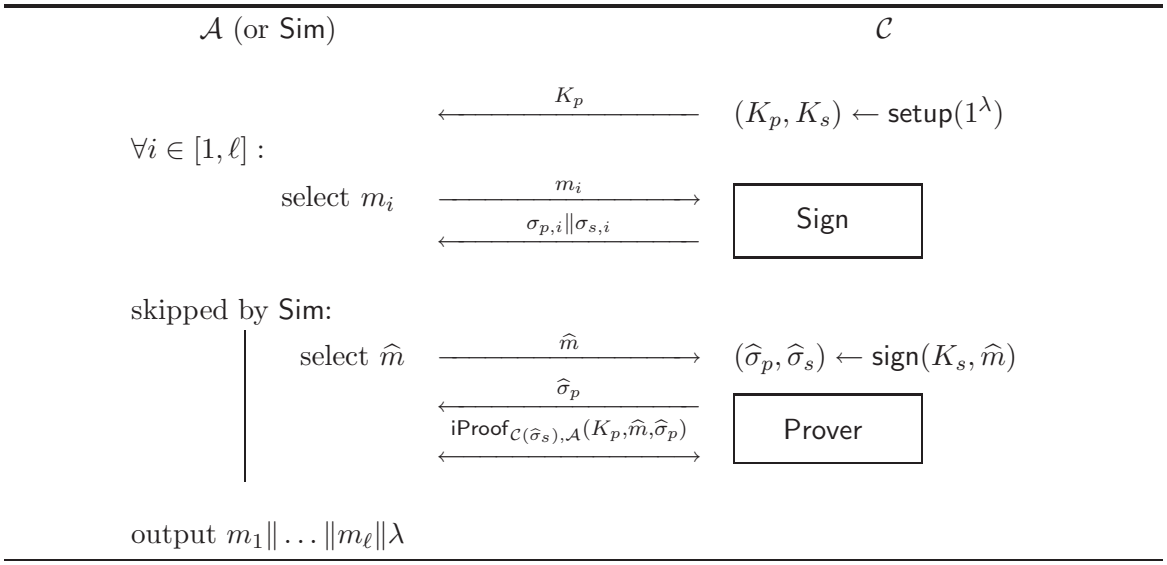


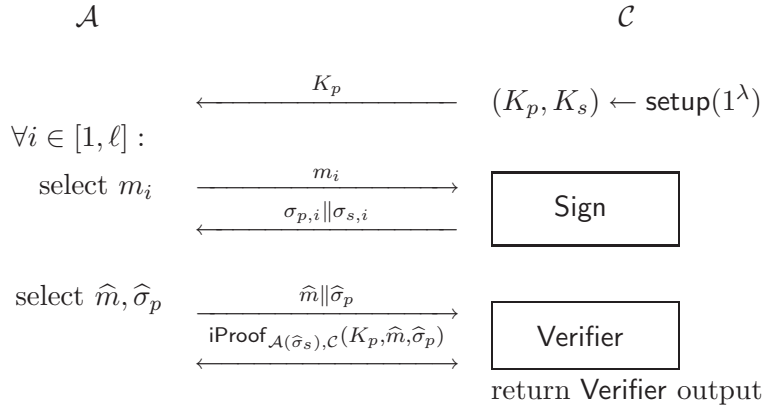
Figure 12.1. *ONTAP Non-Transferable Game.*

phase, $\mathcal{A}$ selects a $\widehat{m}$ and a $\widehat{\sigma}_p$ with $\widehat{m}$ not sent to Sign*. $\mathcal{A}$ simulates a prover to an honest verifier as depicted in Figure 12.2.*

The ONTAP scheme is said unforgeable *if no* PPT *adversary $\mathcal{A}$ can make the honest verifier accepting in the game of Figure 12.2 with non-negligible probability.*

Clearly, Definition 12.4 implies classical unforgeability in the sense of Definition 11.2 since anyone able to forge a signature is also able to win the game of Fig. 12.2.

In the ONTAP unforgeability definition (Definition 12.4), we could give access to non-concurrent prover oracles to the adversary $\mathcal{A}^*$. Suppose it is the case and we denote them by Prover_j's. Each oracle simulates an honest prover $\mathbf{P}$ in $\mathsf{iProof}_{\mathbf{P}(\sigma^*_{s,j}),\mathcal{A}^*}(K_p, m^*_j, \sigma^*_{p,j})$. Each oracle Prover_j is setup with a given message m^*_j and several iProof executions can be requested for the same m^*_j and some signature (σ_p, σ_s). Executions to the same Prover_j cannot be performed concurrently. This definition of unforgeability can be reduced to Definition 12.4 which uses no prover oracle. Suppose $\mathcal{A}^*$ is limited to m Prover oracles. We split $\mathcal{A}^*$ in several adversaries $\mathcal{A}^*_1$ to $\mathcal{A}^*_m$ playing modified games. Each $\mathcal{A}^*_i$ plays with $\mathcal{C}$ where all Prover_j for $j \neq i$ are replaced by a query to the sign oracle and a simulation for the iProof protocol. So, only the Prover_i in the game with the adversary $\mathcal{A}_i$ uses a Prover oracle. Clearly, $\Pr[\mathcal{A}^* \text{ succeeds}] \leq \sum_{i=1}^m \Pr[\mathcal{A}^*_i \text{ succeeds}]$. Now, we define adversaries $\mathcal{A}'_i$: each one plays the same game than $\mathcal{A}^*_i$ except that Prover_i is simulated by Sim as defined in Definition 12.3. By using the offline non-transferability property, the state of adversary $\mathcal{A}^*_i$ is computationally indistinguishable from the one of adversary $\mathcal{A}'_i$, so $\Pr[\mathcal{A}^* \text{ succeeds}] \leq \sum_{i=1}^m \Pr[\mathcal{A}'_i \text{ succeeds}] + \mathsf{negl}$. This proves that introducing a Prover



$\mathcal{A}$ **wins if** Verifier accepts and $\widehat{m}$ not queried to Sign.

Figure 12.2. *ONTAP Unforgeability Game.*

oracle in Definition 12.4 does not strengthen our unforgeability notion when offline non-transferability is granted.

Theorem 12.5 (ONTAP Construction).
Let S *be a classical digital signature scheme in which the* sign *algorithm outputs a signature splittable in two parts: a public part σ_p and a private part σ_s. We assume there exists an algorithm* simulate *such that $\sigma_p \leftarrow \mathsf{simulate}(K_p, m)$ is computationally indistinguishable from the one generated by $\mathsf{sign}(K_s, m)$. Let* iProof *be a deniable zero-knowledge proof of knowledge for witness σ_s in the relation*

$$R(K_p \| m \| \sigma_p, \sigma_s) \iff \mathsf{verify}(K_p, m, \sigma_p \| \sigma_s) \ .$$

If S *is EF-CMA-secure, then the ONTAP based on* setup, sign, *and* iProof *is secure.*

The deniable zero-knowledge property of the iProof protocol guarantees that the proof is offline non-transferable. So, the ability to *prove* that σ_s is known by someone is not possible after the protocol is complete.

The required signature scheme S should be in the class $\mathbb{C}$ defined in [SSN08] which includes many signature schemes. Note that there exists a Σ-protocol for any signature scheme since any NP relation has one [SSN08]. However, such a protocol is in general not efficient. Thanks to the next section, we transform Σ-protocols into denial ZK proofs of knowledge.

Proof.
We start with the constructed ONTAP scheme and consider the ONTAP security games. Assuming that S is EF-CMA-secure, the public signature is simulatable, and iProof is deniable ZK, we want to show that the ONTAP is unforgeable in the sense of Definition 12.4 and offline non-transferable in the sense of Definition 12.3.

Unforgeability: We consider a T-time adversary $\mathcal{A}$ playing the ONTAP unforgeability game with a challenger $\mathcal{C}$ as depicted in Figure 12.2. $\mathcal{A}$ is limited by ℓ queries to the oracle Sign.

We split $\mathcal{A}$ in two parts: $\mathcal{A}_1$, which represents the three first moves of $\mathcal{A}$ in Figure 12.2 and outputs a state λ, and $\mathcal{A}_2(\lambda)$, which represents the last two moves of $\mathcal{A}$.

Thanks to the soundness, Ext fed with $\mathcal{A}_2(\lambda)$ produces $\widehat{\sigma}_s$ such that $\mathsf{verify}(K_p, \widehat{m}, \widehat{\sigma}_p \| \widehat{\sigma}_s)$ holds. Hence, running $\lambda \leftarrow \mathcal{A}_1^{\mathsf{Sign}}$, then $\mathsf{Ext}^{\mathcal{A}_2(\lambda)}$ wins in the EF-CMA game which is not possible.

Offline Non-Transferability: We construct Sim by running $\mathcal{A}$ until $\widehat{m}$ is submitted. Then, Sim runs $\widehat{\sigma}'_p \leftarrow \mathsf{simulate}(K_p, \widehat{m})$ and continues to simulate $\mathcal{A}$ by feeding it with $\widehat{\sigma}'_p$. Clearly, $\mathcal{A}$ with the simulated $\widehat{\sigma}'_p$ reaches a state which is indistinguishable from $\mathcal{A}$ with a true signature $\widehat{\sigma}_p$. Then, we use the simulator for iProof to simulate the final state (and output) from $\mathcal{A}$.

■

12.4 Deniable ZK from Σ-Protocols

Generally, Σ-protocols are designed to prove the knowledge of a witness in a binary relation. The notion of Σ-protocol represents an important tool for the design of zero-knowledge protocols. It generalizes well-known proofs of knowledge such as Guillou-Quisquater [GQ88, GQ90] or Schnorr [Sch90, Sch91] protocols. Below, we first briefly recall the required material and refer to Damgård [Dam05] for a detailed treatment.

(Classical) Σ-protocols are only honest verifier zero-knowledge (HVZK) in the sense of Definition 11.5. Clearly, in many applications we cannot assume that verifiers will be honest. So, later, we will present a generic construction allowing to strengthen Σ-protocols. It transforms any Σ-protocol into a deniable zero-knowledge proof of knowledge in the sense of Definition 11.10 either in the standard, CRS, or RO model.

12.4.1 Σ-Protocols

A Σ-protocol is a special 3-move honest-verifier zero-knowledge (sHVZK) proof of knowledge between a prover $\mathbf{P}$ and a verifier $\mathbf{V}$ for a relation R. We recall that for a pair $(x, w) \in R$, x is a common input for $\mathbf{P}$ and $\mathbf{V}$ and w is a private input for $\mathbf{P}$. A Σ-protocol consists of three moves: a, e, and z where the first is from $\mathbf{P}$ to $\mathbf{V}$. We usually call the three exchanged messages the transcript and we denote it by (a, e, z). We call the transcript "accepting" if an honest verifier $\mathbf{V}$ would accept the corresponding interactive proof execution. In Σ-protocols, e is a random bit-string which is (for the honest verifier) independent from a.

To fully characterize a Σ-protocol, we specify the algorithms which generate a and z, the domain of e, and the verifying algorithm executed by the verifier at the end. Let us denote them by PR_1, PR_2 and VER respectively. Finally, a Σ-protocol can be formally described as depicted in Figure 12.3 where the notation ϖ_{P} (resp. ϖ_{V}) represents the random tape of $\mathbf{P}$ (resp. of $\mathbf{V}$).

$\mathbf{P}$		$\mathbf{V}$
input: w	**common input:** x	
random tape: ϖ_{P}		**random tape:** ϖ_{V}
$a = \mathsf{PR}_1(x, w; \varpi_{\mathrm{P}})$	$\xrightarrow{a}$	
	$\xleftarrow{e}$	$e \in_{\mathsf{u}} \{0,1\}^t$
$z = \mathsf{PR}_2(x, w, e; \varpi_{\mathrm{P}})$	$\xrightarrow{z}$	$b = \mathsf{VER}(x, a, e, z)$

Figure 12.3. *A Generic Σ-protocol.*

In addition to the above restrictions, a Σ-protocol must achieve efficiency and completeness following Definition 11.4, and must satisfy two additional properties: special soundness and special HVZK (sHVZK).

Definition 12.6 (Σ-protocol).
Let $(\mathbf{P}, \mathbf{V})$ *be a pair of interactive Turing machines (ITMs) and* proof *a 3-move protocol as depicted in Figure 12.3.* $\mathsf{proof}_{\mathbf{P}(w),\mathbf{V}}(x)$ *is a* Σ-protocol *for the relation* R *if the following holds :*

- Efficiency and completeness: *See Definition 11.4.*
- Special Soundness: *For any* $x \in L_R$ *and any two accepting transcripts on input* x, (a, e_1, z_1), (a, e_2, z_2) *with* $e_1 \neq e_2$, *there exists a polynomial-time extractor* $\mathsf{Ext}(x, a, e_1, e_2, z_1, z_2)$ *which outputs a bit-string* w *such that* $(x, w) \in R$.
- Special HVZK (sHVZK): *There exists a polynomial-time simulator* $\mathsf{Sim}_{\mathsf{zk}}$ *which for any* x *and a random* e *outputs a transcript* (a, e, z) *with identical probability distribution as a transcript generated by honest* $\mathbf{P}$ *and* $\mathbf{V}$ *on input* x.

The special soundness (resp. sHVZK) guarantees that a Σ-protocol is sound (resp. HVZK) (for details, see Damgård [Dam05]). Note also that the sHVZK is a restricted version of HVZK, where the simulator cannot freely choose the message e sent by $\mathbf{V}$.

12.4.2 Weak Σ-Protocols

We define a weaker notion of Σ-protocols as follows.

Definition 12.7 ($\kappa(x)$-weak Σ-protocol).
Let κ be a real function. A $\kappa(x)$-weak Σ-protocol *is a Σ-protocol with the special soundness property modified as follows:*

> *For any $x \in L_R$, any a, and any $e \in \{0,1\}^t$, there exists an* unique *z such that* $\mathsf{VER}(x, a, e, z) = 1$. *Denote* $z = \mathsf{Resp}(x, a, e)$.
>
> *There exists a polynomial-time extractor* Ext *such that for any $x \in L_R$, any a, and any $e_1 \in \{0,1\}^t$, we have*
>
> $$\Pr_{e_2 \in_u \{0,1\}^t}[(x, \mathsf{Ext}(x, a, e_1, e_2, \mathsf{Resp}(x, a, e_1), \mathsf{Resp}(x, a, e_2))) \in R] \geq 1 - \kappa(x)$$

$\kappa(x)$-weak Σ-protocols are sound with soundness error $\kappa(x)$. This comes from a simplified version of the proof of Theorem 12.10 and Theorem 12.11 below. Special soundness is achieved for $\kappa(x) = 2^{-t}$.

In the following, we give two examples of $\kappa(x)$-weak Σ-protocols.

Example 1. The first example is the Guillou-Quisquater (GQ) protocol [GQ88, GQ90]. Let $N = pq$ be an RSA modulus, e be the RSA public exponent, and $d = e^{-1} \pmod{\varphi(N)}$ be the RSA private exponent. For simplicity we assume that e is prime[2]. Given a public X, the GQ protocol allows to prove the knowledge of x such that $X = x^e \bmod N$. The GQ protocol is depicted in Figure 12.4.

P		**V**
input: x	**common input:** $(N, e), X$	
picks $y \in_u \mathbb{Z}_N^*$		
$Y = y^e \bmod N$	$\xrightarrow{Y}$	
	$\xleftarrow{r}$	picks $r \in_u \{0,1\}^t$
$z = yx^r \bmod N$	$\xrightarrow{z}$	check $z^e \stackrel{?}{=} YX^r \pmod N$

Figure 12.4. *The Guillou-Quisquater (GQ) Protocol.*

[2] In practice, RSA keys use $e = 3$ or $e = 65537$.

Theorem 12.8 (The GQ Weak-Σ-protocol).
Let t be the bit-length of the second move. The GQ protocol with prime exponent e is a $\frac{\lceil \frac{2^t}{e} \rceil}{2^t}$-weak Σ-protocol.

Proof.
Clearly, we can define PR_1, PR_2, and VER. Special HVZK is straightforward: on input (x, r), pick a random z and lets $Y = \frac{z^e}{X^r} \pmod N$. Here, we only need to prove that it is $\kappa(x)$-weak. Note that given the two first moves (Y,r), there exists an unique third move (z) for which $\mathbf{V}$ will accept. It remains to prove the soundness and for that we should build an extractor Ext which outputs the witness given two transcripts with the same first move (Y), i.e., any $(I, Y, r_1, \mathsf{Resp}(I, Y, r_1))$ and a random $(r_2, \mathsf{Resp}(I, Y, r_2))$ with $I = (N, e, X)$.

Given (N, e, X), Y, r_1, r_2, z_1, z_2 such that $z_1^e = YX^{r_1} \pmod N$ and $z_2^e = YX^{r_2} \pmod N$, if $\gcd(r_1 - r_2, e) = 1$, then we can find some integers a and b such that $ae + b(r_1 - r_2) = 1$ by using the Extended Euclid algorithm and then can compute $x = X^a z_1^b z_2^{-b} \bmod N$ which satisfies

$$x^e = X^{ae}(z_1^e)^b(z_2^e)^{-b} = X^{ae}(YX^{r_1})^b(YX^{r_2})^{-b} = X^{ae+b(r_1-r_2)} = X \pmod N$$

so a valid witness is extracted. Clearly, the GQ protocol is $\kappa(x)$-weak where $\kappa(x) = \max_{r_1} \Pr_{r_2}[\mathsf{gcd}(r_1 - r_2, e) \neq 1]$ and we find that

$$\kappa(x) = \sum_{k=0}^{2^t-1} 1_{\mathsf{gcd}(r_1-k,e)\neq 1} \Pr[r_2 = k] = \frac{1}{2^t}\#\{\text{multiple of } e \text{ in } [r_1, r_1 + 2^t - 1]\} \leq \frac{\lceil \frac{2^t}{e} \rceil}{2^t}.$$

■

Example 2. The second example is from Schnorr [Sch90, Sch91]. Let g be the generator of a group $\mathcal{G}$ of prime order q. The Schnorr protocol allows to prove the knowledge of the discrete logarithm x in $\mathcal{G}$ of the element $X = g^x$. The Schnorr protocol is depicted in Figure 12.5.

Theorem 12.9 (The Schnorr Weak-Σ-protocol).
Let t be the bit-length of the second move. The Schnorr protocol in a group of prime order is a 2^{-t}-weak Σ-protocol.

Proof.
As in proof of Theorem 12.8, it suffices to prove the soundness and for that we should build an extractor Ext.

First, note that

$$z_1 - z_2 = (y + r_1 x) - (y - r_2 x) = (r_1 - r_2)x \ .$$

P		V
input: x	**common input:** $(g,q), X$	
picks $y \in_{\mathsf{u}} \mathbb{Z}_q$		
$Y = g^y$	$\xrightarrow{Y}$	
	$\xleftarrow{r}$	picks $r \in_{\mathsf{u}} \{0,1\}^t$
$z = y + rx \bmod q$	$\xrightarrow{z}$	check $g^z \stackrel{?}{=} YX^r$

Figure 12.5. *The Schnorr Protocol.*

So, a trivial extractor $\mathsf{Ext}(X, Y, r_1, r_2, z_1, z_2)$ exists and simply consists in computing $x = (z_1 - z_2)(r_1 - r_2)^{-1}$. Clearly, if $(r_1 - r_2)$ is invertible in $\mathbb{Z}_q$, then Ext succeeds. Recall that $2^t \ll q$ and so assuming q prime, all elements are invertible except 0. The Schnorr protocol is $\kappa(x)$-weak where $\kappa(x) = 2^{-t}$.

■

12.4.3 Generic Transform of Σ-Protocols

A malicious verifier can convert an interactive Σ-protocol into a non-interactive proof following the Fiat-Shamir technique [FS87]. Indeed, in a Σ-protocol, the move (e) should be independent from the move (a). However, a malicious verifier can use a random oracle R and return $e = \mathsf{R}(x, a)$ where x is the common input. Them, (a, z) becomes an universally verifiable signature for x. As a consequence, no Σ-protocol can be ZK considering malicious verifiers [CDM00].

In this section, we show how to transform an honest-verifier zero-knowledge (HVZK) protocol into a deniable ZK (dZK) one. For that, we should force the verifier to commit on its challenge at the beginning of the protocol. A solution is to add a commitment step at the beginning. The idea was proposed by Goldreich-Micali-Wigderson [GMW91] and then reused by Goldreich-Kahan [GK96]. They prove that it is possible to achieve ZK in the standard model with a polynomial round complexity.

Here, we want to prove that it is possible to achieve deniable ZK in the CRS or RO models with only 4 moves. At the same time, we achieve ZK in the standard model with one extra move. The extra move is necessary for sending the fresh public key which replaces the common reference string. Note that Cramer, Damgård, and MacKenzie [CDM00] proposed a transform to achieve ZK but with a bigger round complexity while Damgård [Dam00] proposed an efficient construction but without deniability. Clearly, for our application, deniability is mandatory in the CRS and RO models.

The protocol in the standard model is depicted in Figure 12.6. Clearly, the prover should be ensured that nobody knows the trapdoor K_s. Consequently, the prover generates the key pair himself, he gives the public key on the first (extra) move and the private key on the last one.

P		**V**
input: w **random tape:** ϖ_{P}	**common input:** x	**random tape:** ϖ_{V}
$R = \mathsf{random}(\varpi_{\mathrm{P}})$		
$(K_p, K_s) = \mathsf{setup}(1^\lambda, R)$	$\xrightarrow{K_p}$	$e \in_{\mathsf{u}} \{0,1\}^t$
	$\xleftarrow{c}$	$(c, d) = \mathsf{commit}(K_p, e; \varpi_{\mathrm{V}})$
$a = \mathsf{PR}_1(x, w; \varpi_{\mathrm{P}})$	$\xrightarrow{a}$	
$e = \mathsf{open}(K_p, c, d)$	$\xleftarrow{d}$	
$z = \mathsf{PR}_2(x, w, e; \varpi_{\mathrm{P}})$	$\xrightarrow{z \Vert R}$	$(\widehat{K}_p, \widehat{K}_s) = \mathsf{setup}(1^\lambda, R)$
		check $\widehat{K}_p \stackrel{?}{=} K_p$
		$b = \mathsf{VER}(x, a, e, z)$

Figure 12.6. *A Generic Transform of Σ-protocol in the Standard Model.*

Theorem 12.10 (Generic Transform of Σ-protocol in the Standard Model). *Let $\mathcal{C}$ be a trapdoor commitment scheme, π be a $\kappa(x)$-weak Σ-protocol, and π' be its generic transform as depicted in Figure 12.6.*

For any arbitrary large integer k, π' is a zero-knowledge proof of knowledge in the standard model with soundness error $\kappa'(x) = \max\left(\kappa(x), 1/|x|^k\right)$.

Clearly, if the trapdoor of the commitment scheme K_s is known by the verifier, the protocol remains honest-verifier zero-knowledge (this is essentially the Σ-protocol) but loses deniability. Indeed, a malicious verifier could take $e = \mathsf{OW}(a)$ and open c to e. The response z would become a transferable proof following the Fiat-Shamir paradigm [FS87] to transform interactive proofs into non-interactive ones. Thus, the public key of the commitment scheme K_p should be trusted by the prover who believes that the verifier does not know the corresponding private key. In practice, a first solution is to instantiate the protocol in the CRS model as depicted in Figure 12.7. Another costless pragmatic solution could consist of using a hash function at the place of the commitment. This is essentially the instantiated variant with the RO commitment scheme depicted in Figure 12.8.

Theorem 12.11 (Generic Transform of Σ-protocol in the CRS and RO Models). *Let $\mathcal{C}$ be a trapdoor commitment scheme (resp. the extractable RO commitment scheme). Let π be a $\kappa(x)$-weak Σ-protocol and let π' be its generic transform as depicted in Fig-*

P		**V**
input: w	**common input:** crs, x	
random tape: ϖ_{P}		**random tape:** ϖ_{V}
		$e \in_{\mathsf{u}} \{0,1\}^t$
	$\xleftarrow{c}$	$(c,d) = \mathsf{commit}(\mathsf{crs}, e; \varpi_{\mathrm{V}})$
$a = \mathsf{PR}_1(x, w; \varpi_{\mathrm{P}})$	$\xrightarrow{a}$	
$e = \mathsf{open}(\mathsf{crs}, c, d)$	$\xleftarrow{d}$	
$z = \mathsf{PR}_2(x, w, e; \varpi_{\mathrm{P}})$	$\xrightarrow{z}$	$b = \mathsf{VER}(x, a, e, z)$

Figure 12.7. *A Generic Transform of Σ-protocol in the CRS model.*

P		**V**
input: w	**common input:** x	
random tape: ϖ_{P}		**random tape:** ϖ_{V}
		$e \in_{\mathsf{u}} \{0,1\}^t, r = \mathsf{random}(\varpi_{\mathrm{V}})$
	$\xleftarrow{c}$	$c = \mathsf{H}(e, r),\ d = e \Vert r$
$a = \mathsf{PR}_1(x, w; \varpi_{\mathrm{P}})$	$\xrightarrow{a}$	
$e \Vert r = d,\ c \stackrel{?}{=} \mathsf{H}(e, r)$	$\xleftarrow{d}$	
$z = \mathsf{PR}_2(x, w, e; \varpi_{\mathrm{P}})$	$\xrightarrow{z}$	$b = \mathsf{VER}(x, a, e, z)$

Figure 12.8. *A Generic Transform of Σ-protocol in the RO model.*

ure 12.7 where crs *is the public key as setup in the commitment scheme (resp. in Figure 12.8 where* H *is a random oracle).*

For any arbitrary large integer k, π' is a deniable zero-knowledge proof of knowledge in the CRS model (resp. in the RO model) with soundness error $\kappa'(x) = \max\left(\kappa(x), 1/|x|^k\right)$.

Proof (of Theorems 12.10 and 12.11)).
We have a Σ-protocol, denoted by π, satisfying Definition 12.6. We build a protocol, denoted by π', and we want to prove that it satisfies Definition 11.10.

Efficiency and completeness of the protocols of Figures 12.6, 12.7, and 12.8 are trivial. So, we concentrate in proving the properties of soundness and deniable zero-knowledge.

Soundness. To prove that the constructed protocol π' is sound, we need to show that there exists an extractor Ext' with some properties as defined in Definition 12.6. We start the proof in the standard model.

Let π be a $\kappa(x)$-weak Σ-protocol. Recall that for any $x \in L_R$, given the input x and two random accepting transcripts (a, e_1, z_1) and (a, e_2, z_2) of the protocol π, there exists a polynomial-time extractor Ext which outputs the witness w such that $(x, w) \in R$ with probability $1 - \kappa(x)$ (over e_2) (special soundness property, see Definition 12.6).

Let k be an arbitrary positive large integer. Let $\mathbf{P}^*$ be any malicious prover. Let $\varepsilon(x)$ be the probability that $\mathbf{P}^*$ passes the protocol π' with an honest verifier $\mathbf{V}$ with input x. Let $\kappa'(x)$ be the soundness error of the protocol π'. By Definition 11.7 it is assumed that $\varepsilon(x) > \kappa'(x)$. We construct the extractor Ext' as described in Figure 12.9. Thanks to

1. Ext' picks ϖ_{P} and set up $\mathbf{P}^*$ with ϖ_{P}.
2. Ext' plays the role of the verifier and runs a complete protocol with $\mathbf{P}^*$ who gives the trapdoor at the end. If this protocol does not fail (event A_1), this defines a first transcript $c, a, d_1, z_1 \| K_s$ such that $\mathsf{VER}(x, a_1, e_1, z_1)$ with $e_1 = \mathsf{open}(K_p, c, d_1)$ outputs 1.
3. Ext' picks e_2 and computes $d_2 \leftarrow \mathsf{equivocate}(K_s, c, e_2)$. Ext' runs another complete protocol with $\mathbf{P}^*$ set up with the same ϖ_{P} and uses messages c and d_2. If this protocol does not fail (event A_2), this defines a second transcript $c, a, d_2, z_2 \| K_s$ such that $\mathsf{VER}(x, a, e_2, z_2)$ with $e_2 = \mathsf{open}(K_p, c, d_2)$ outputs 1.
4. If one of the two protocols failed, Ext' aborts. Otherwise, using Ext with inputs (a, e_1, z_1) and (a, e_2, z_2), Ext' recovers w such that $(x, w) \in R$.

Figure 12.9. *The Knowledge Extractor* Ext.

the property of $\mathsf{equivocate}$, the extractor Ext' simulates perfectly an honest verifier for the

malicious prover $\mathbf{P}^*$.

It remains to prove that Ext' runs within limited expected time as in Definition 11.10. Given ϖ_{P} and c, both protocols are independent and succeed with the same probability, let us denote it by

$$\Pr[A_j|\varpi_{\mathrm{P}}, c] = p_{\varpi_{\mathrm{P}},c} \text{ for } j = 1, 2 \ .$$

The expected value of $p_{\varpi_{\mathrm{P}},c}$ over the random choice of ϖ_{P} and c is $\varepsilon(x)$. No matter whether A_j holds, let z_j be the unique z such that $\mathsf{VER}(x, a, e_j, z_j) = 1$. Let B the event that $\mathsf{Ext}(a, e_1, e_2, z_1, z_2)$ succeeds, i.e.,

$$\Pr[\neg B] \leq \kappa(x) \ .$$

Furthermore, $\Pr[\neg B|A_1] \leq \kappa(x)$. We want to compute $\Pr[A_1 \wedge A_2 \wedge B]$ and we start by writing :

$$\Pr[A_1 \wedge A_2 \wedge B|\varpi_{\mathrm{P}}, c] = \Pr[A_1 \wedge A_2|\varpi_{\mathrm{P}}, c] - \Pr[A_1 \wedge A_2 \wedge \neg B|\varpi_{\mathrm{P}}, c] \ .$$

Focusing on the right term, we have

$$\Pr[A_1 \wedge A_2 \wedge \neg B|\varpi_{\mathrm{P}}, c] \leq \Pr[A_1 \wedge \neg B|\varpi_{\mathrm{P}}, c] \leq p_{\varpi_{\mathrm{P}},c}\kappa(x) \ ,$$

while the left term is

$$\Pr[A_1 \wedge A_2|\varpi_{\mathrm{P}}, c] = p^2_{\varpi_{\mathrm{P}},c} \ .$$

So, we obtain

$$\Pr[A_1 \wedge A_2 \wedge B|\varpi_{\mathrm{P}}, c] \geq p_{\varpi_{\mathrm{P}},c}(p_{\varpi_{\mathrm{P}},c} - \kappa(x)) \ .$$

Finally, we compute the expected value over the ϖ's and c's and using the Jensen's inequality on the function $x \mapsto x^2$, we obtain

$$\Pr[A_1 \wedge A_2 \wedge B] \geq \varepsilon(x)(\varepsilon(x) - \kappa(x)) \ .$$

We conclude that the average number of running time of Ext' before it succeeds is

$$\frac{1/\varepsilon(x)}{\varepsilon(x) - \kappa(x)} \ .$$

Following Definition 11.10, it remains to prove that the following inequality

$$\frac{1/\varepsilon(x)}{\varepsilon(x) - \kappa(x)} \leq \frac{|x|^k}{\varepsilon(x) - \kappa'(x)}$$

is true for any $\varepsilon(x) > \kappa'(x)$. It is the case when $\kappa'(x) = \max\left(\kappa(x), 1/|x|^k\right)$.

Note that in the standard model (Figure 12.6), Ext' learns the trapdoor when event A_1 holds.

In the case of the CRS model, the extractor Ext' can be assumed to *know the trapdoor of the commitment* (see Definition 11.7).

In the case of the RO commitment, the proof is essentially the same: Ext' creates two entries $\mathsf{H}(e_1, r_1) = \mathsf{H}(e_2, r_2) = c$ in the H table where r_1 and r_2 are random values from the RO commitment (see Section 3.5.9.1). Then, it executes both protocols by using one entry for each. If by any chance $\mathbf{P}^*$ queries H with the other, the extraction fails. But this happens with negligible probability.

Deniable Zero-knowledge. First, note that deniable zero-knowledge (dZK) and zero-knowledge (ZK) are equivalent in the standard model. So, in this proof we will show that all three protocols are dZK. This will imply that the protocol of Figure 12.6 is ZK in the standard model.

We need to build a simulator $\mathsf{Sim}_{\mathsf{zk}}$ able to simulate the interactions between an honest prover $\mathbf{P}$ and any verifier $\mathbf{V}^*$ as described in Definition 11.10. In the CRS model, the simulator $\mathsf{Sim}_{\mathsf{zk}}$ is no longer allowed to generate the common reference string crs. Let $K_p = \mathsf{crs}$ be any uniformly distributed random string. K_p is given to all: to the prover $\mathbf{P}$, to the verifier $\mathbf{V}^*$, and to the simulator $\mathsf{Sim}_{\mathsf{zk}}$.

Recall that given any $x \in L_R$ and a random e there exists a polynomial-time simulator $\mathsf{Sim}'_{\mathsf{zk}}$ which outputs a transcript (a, e, z) which has identical probability distribution than a transcript generated by the honest prover $\mathbf{P}$ and an honest verifier $\mathbf{V}$ on input x.

We construct the simulator $\mathsf{Sim}_{\mathsf{zk}}$ as depicted in Figure 12.10. Clearly, $\mathsf{Sim}_{\mathsf{zk}}$ always returns a complete protocol view from $\mathbf{V}^*$. It is either of type I $(\varpi_{\mathrm{V}}, K_p, x, a)$ or of type II $(\varpi_{\mathrm{V}}, K_p, x, a, z)$. The ϖ_{V} distribution is perfect as well as the view of type I. Let $A_{\varpi_{\mathrm{V}}, K_p, x}$ be the set of all possible a such that $\mathbf{V}^*(\varpi_{\mathrm{V}}, K_p, x, a)$ returns a valid c. The distribution of a' is the marginal distribution from PR_1 conditioned to set $A_{\varpi_{\mathrm{V}}, K_p, x}$. So, it is perfectly simulated as well. Finally, the unique z' is well simulated (the negligible probability of breaking the commitment has a negligible influence on the distribution) so we have a computationally indistinguishable simulator.

We still have to show that the average number of rewindings is polynomial. Let ϖ_{V} be a fix random tape of $\mathbf{V}^*$. Given $x \in L_R$, for w s.t. $(x, w) \in R$ we consider $\mathbf{V}^*_{\varpi_{\mathrm{V}}}$ interacting with $\mathbf{P}(x, w)$. We denote by $p = p_{\varpi_{\mathrm{V}}}(x)$ the probability that the commit value c is incorrectly opened to $\mathbf{P}$. Since the distribution of a can be simulated, $p_{\varpi_{\mathrm{V}}}(x)$ does not depend on w.

Let C be the number of partial or complete executions that $\mathsf{Sim}_{\mathsf{zk}}$ do with $\mathbf{V}^*$. Clearly, when $C = 1$ the first commitment was incorrect and we have $\Pr[C = 1] = p$. When we have $C = c$, the first and the last commitment were correct, but there were $c - 2$ incorrect ones in the middle. So, $\Pr[C = c] = (1 - p) \cdot p^{c-2} \cdot (1 - p)$ for $c \geq 2$. Clearly, the probability that we need c iterations before reaching step 5(c)ii is

$$\Pr[C = c] = (1 - p)^2 p^{c-2} \text{ for } c \geq 2 \ .$$

Incidentally, the probability to stop before step 5a is p. The expected number of iterations is constant. The expected complexity, i.e., the expected number of rounds over the distribution

1. $\mathsf{Sim}_{\mathsf{zk}}$ launches $\mathbf{V}^*$ with input x and a fresh random tape ϖ_{V}.
2. $\mathsf{Sim}_{\mathsf{zk}}$ receives c from $\mathbf{V}^*$.
3. $\mathsf{Sim}_{\mathsf{zk}}$ picks a random a using the same distribution than $\mathsf{PR}_1(\cdot)$. Thanks to the special HVZK property, this can be simulated by using $\mathsf{Sim}'_{\mathsf{zk}}$ and obtaining (a, e_*, z_*).
4. $\mathsf{Sim}_{\mathsf{zk}}$ then gives a to $\mathbf{V}^*$.
5. $\mathsf{Sim}_{\mathsf{zk}}$ receives d from $\mathbf{V}^*$, computes $e = \mathsf{open}(K_p, c, d)$, and checks $e \stackrel{?}{\neq} \perp$.
   - If the commitment is not valid, i.e., $e = \perp$, then $\mathsf{Sim}_{\mathsf{zk}}$ stops the simulation and releases the transcript $(\varpi_{\mathrm{V}}, K_p, x, a)$.
   - Otherwise, i.e., if the commitment is valid,
     (a) $\mathsf{Sim}_{\mathsf{zk}}$ rewinds $\mathbf{V}^*$ with the same random tape ϖ_{V} and receives the same c from $\mathbf{V}^*$ since ϖ_{V} is unchanged. $\mathsf{Sim}_{\mathsf{zk}}$ can thus guess that c will again open to e.
     (b) $\mathsf{Sim}_{\mathsf{zk}}$ gives x and e to the simulator $\mathsf{Sim}'_{\mathsf{zk}}$ in order to obtain a "good" transcript (a', e, z'). $\mathsf{Sim}_{\mathsf{zk}}$ sends a' to $\mathbf{V}^*$.
     (c) $\mathsf{Sim}_{\mathsf{zk}}$ receives d' from $\mathbf{V}^*$, computes $e' = \mathsf{open}(K_p, c, d')$, and checks $e' \stackrel{?}{\neq} \perp$.
       - If the commitment is not valid, i.e., $e' = \perp$, then $\mathsf{Sim}_{\mathsf{zk}}$ goes back to step 5a.
       - Otherwise, i.e., if the commitment is valid,
         i. If $e \neq e'$ (double opening of c), $\mathsf{Sim}_{\mathsf{zk}}$ aborts.
         ii. $\mathsf{Sim}_{\mathsf{zk}}$ finishes by yielding $(\varpi_{\mathrm{V}}, K_p, x, a', z')$ of the last interaction with $\mathbf{V}^*$.

Figure 12.10. *The Simulator* $\mathsf{Sim}_{\mathsf{zk}}$.

of the random tape, is computed by

$$\begin{aligned}
\mathrm{E}[C] &= \sum_{c=0}^{\infty} c \Pr[C = c] \\
&= \Pr[C = 1] + \sum_{c=2}^{\infty} c \cdot \Pr[C = c] \\
&= p + \cdot \sum_{c=2}^{\infty} c \cdot (1-p)^2 p^{c-2} \\
&= p + (1-p)^2 \cdot \sum_{j=0}^{\infty} (j+2) \cdot p^j \ .
\end{aligned}$$

We can now compute the real value of the expected value. We first split the sum in two parts as follows:

$$\mathrm{E}[C] = p + (1-p^2) \cdot \left(\sum_{j=0}^{\infty} (j+1) \cdot p^j + \sum_{j=0}^{\infty} p^j \right) \ .$$

Then, we integrate the left sum in order to easily compute the value:

$$\begin{aligned}
\mathrm{E}[C] &= p + (1-p^2) \cdot \left(\frac{\mathrm{d}}{\mathrm{dp}} \left[\sum_{j=0}^{\infty} p^{j+1} \right] + \sum_{j=0}^{\infty} p^j \right) \\
&= p + (1-p^2) \cdot \left(\frac{\mathrm{d}}{\mathrm{dp}} \left[\sum_{j=0}^{\infty} p^j - 1 \right] + \sum_{j=0}^{\infty} p^j \right) \\
&= p + (1-p^2) \cdot \left(\frac{\mathrm{d}}{\mathrm{dp}} \left[\frac{p}{1-p} \right] + \frac{1}{1-p} \right) \ .
\end{aligned}$$

Finally, we find a constant expected running time:

$$\begin{aligned}
\mathrm{E}[C] &= p + (1-p^2) \cdot \left(\frac{1}{1-p} + \frac{p}{(1-p)^2} + \frac{1}{1-p} \right) \\
&= p + p + 2 \cdot (1-p) \\
&= 2 \ .
\end{aligned}$$

So, the expected complexity, i.e., the expected number of rounds over the distribution of the random tape, is $\mathrm{E}[C] = 2$. This proves that the simulator runs in expected polynomial time.

■

12.5 ONTAP Constructions in Practice

12.5.1 ONTAP with a Generic RSA Signature

We propose ONTAP-RSA: an ONTAP scheme which is generic for RSA-based signatures. It is based on a zero-knowledge variant of the GQ protocol.

Consider $h \leftarrow \mathsf{H}_{\mathsf{seed}}(m)$ be a formatting function and $b = \mathsf{V}(h, m)$ be a check function returning 1 if the formatted h is consistent with the message m, and 0 otherwise.

Definition 12.12 (Generic RSA Signature Scheme).
A generic RSA signature with security parameter k works in a group $\mathbb{Z}_N^*$ with $N = pq$ where p, q are two $\frac{k}{2}$-bit random prime numbers. Let e, d such that $ed \equiv 1 \pmod{\varphi(N)}$ and e is prime. Since several variants are commonly used we do not specify further the generation algorithm. The private key is $K_s = d$ and the corresponding public key is $K_p = (N, e)$

The signature of a message m consists of the tuple $\sigma = (\sigma_p, \sigma_s)$. The algorithm picks a random seed*, computes the formatted message $\sigma_p = \mathsf{H}_{\mathsf{seed}}(m)$, computes the signature $\sigma_s = \sigma_p^d \bmod N$ by using the private key K_s, and outputs σ_p and σ_s.*

There exists a verification algorithm $\mathsf{verify}(K_p, m, \sigma_p, \sigma_s)$ *which outputs 1 if the signature is valid, i.e., if $\mathsf{V}(\sigma_p, m) = 1$ and $\sigma_s^e \bmod N = \sigma_p$, and 0 otherwise.*

Clearly, the PKCS#1v1.5, ISO/IEC 9796, RSA-PSS standards all fit into this category.

The iProof protocol works as depicted in Figure 12.11. Note that K_p is the public key related to the signature scheme while crs is the one related to the commitment scheme. The way to adapt to the plain model or random oracle model is straightforward.

P		**V**
input: σ_s	**common input:** crs, K_p, m	
$\sigma_p = \sigma_s^e \bmod N$		pick $r \in_{\mathsf{u}} \{0,1\}^t$
pick $y \in_{\mathsf{u}} \mathbb{Z}_N^*$	$\xleftarrow{c}$	$(c, d) = \mathsf{commit}(\mathsf{crs}, r)$
$Y = y^e \bmod N$	$\xrightarrow{Y \Vert \sigma_p}$	check $\mathsf{V}(\sigma_p, m) \stackrel{?}{=} 1$
$r = \mathsf{open}(\mathsf{crs}, c, d)$	$\xleftarrow{d}$	
$z = y\sigma_s^r \bmod N$	$\xrightarrow{z}$	check $z^e \stackrel{?}{=} Y\sigma_p^r \pmod N$

Figure 12.11. *The* iProof *Protocol for ONTAP-RSA.*

Theorem 12.13 (ONTAP-RSA).
Assume that the RSA-based signature is EF-CMA-secure and that the $\mathsf{commit}(\cdot)$ *is a trapdoor commitment scheme in the CRS model (resp. the RO commitment scheme). The digital signature scheme added to the signature proof* iProof *of Figure 12.11 forms an ONTAP scheme as defined in Definition 12.2 in the CRS model (resp. in the RO model). The soundness error is* $\frac{\lceil \frac{2^t}{e} \rceil}{2^t}$.

With an extra round we obtain an ONTAP in the standard model as depicted in Figure 12.6.

Proof.
Clearly, there exists an algorithm $\mathsf{simulate}(K_p, m)$ which outputs a σ_p computationally indistinguishable from the one generated by $\mathsf{sign}(K_s, m)$, i.e., $\sigma_p \leftarrow \mathsf{H}_{\mathsf{seed}}(m)$. Thanks to Theorem 12.5, we only need to prove that the signature scheme is unforgeable and the protocol deniable zero-knowledge. Unforgeability is already assumed. Efficiency and completeness of iProof are trivial. Soundness and deniable zero-knowledge properties of iProof are proven by Theorem 12.8 and by Theorem 12.11.

■

12.5.2 ONTAP with a Generic ElGamal Signature

In this section we show how to build an interactive proof for any ElGamal based signature scheme. For that, we first define a *generic ElGamal* signature scheme as follows:

Definition 12.14 (Generic ElGamal).
A generic *ElGamal signature scheme works in a group* $\mathcal{G}$ *with a generator* $g \in \mathcal{G}$ *of order* q. *The private key is* $K_s = x \in_{\mathsf{u}} \mathbb{Z}_q$ *and the corresponding public key is* $K_p = y = g^x$.

The signature of a message m *consists of the tuple* $\sigma = (u, v, \xi, s) \leftarrow \mathsf{sign}(K_s, m)$. *This tuple is split in two parts: a public part* $\sigma_p = (u, v, \xi)$ *which can be perfectly simulated without* K_s *and a secret part* $\sigma_s = s$.

There exists a verification algorithm $\mathsf{verify}(K_p, m, \sigma_p, \sigma_s)$ *which outputs 1 if the signature is valid, i.e., when* $u^s = v$ *and* $\mathsf{ver}(K_p, m, \sigma_p) = 1$, *and outputs 0 otherwise for some* ver *algorithm.*

ElGamal [ElG85] (with a group of prime order), Schnorr [Sch90, Sch91], DSA [DSS94, DSS00], and ECDSA [ECD98] signatures, all meet the *generic* ElGamal requirements and respect the parameters and key generation. We give briefly four examples :

The Plain ElGamal signature. Let p and g be respectively a prime number and a generator of the group $\mathcal{G} = \mathbb{Z}_p^*$. In this case, we have $q = p - 1$. g, p, q are public parameters.

Let $x \in_{\mathsf{u}} \mathbb{Z}_{p-1}$ be the secret key and $y = g^x \bmod p$ be the corresponding public key. The signature σ of a message m is

$$\sigma = (\sigma_r, \sigma_s)$$

$$\sigma_r = g^k \bmod p, \qquad \sigma_s = \frac{h(m) - x\sigma_r}{k} \bmod p - 1$$

for some random $k \in_{\mathsf{u}} \mathbb{Z}^*_{p-1}$. One can verify the signature by checking

$$y^{\sigma_r}\sigma_r^{\sigma_s} \stackrel{?}{\equiv} g^{h(m)} \pmod p \ .$$

Intuitively, we will makes σ_r public since it is a random value while we will keep σ_s private. So, we will define $u := \sigma_r$ and $s := \sigma_s$. Thanks to the zero knowledge proof we should be able to prove that we know a s such that $u^s = v$. So, the second public information will $v := \sigma_r^s$. With respect to the generic ElGamal, we finally define

- $u := g^k \bmod p \ (= \sigma_r)$,
- $v := g^{h(m)}y^{-u} \bmod p$,
- $\xi := \emptyset$,
- $s := \frac{h(m)-xu}{k} \pmod q \ (= \sigma_s)$,
- and the $\mathsf{ver}(y, m, (u, v, \xi))$ algorithm consists in checking $v \stackrel{?}{\equiv} g^{h(m)}y^{-u} \pmod p$.

On can verify that $u^s = v$ by writing

$$u^s = \left(g^k\right)^s = g^{k\cdot\frac{h(m)-xu}{k}} = g^{h(m)}\left(g^x\right)^{-u} = v \ .$$

The Schnorr signature. Let q be a large enough prime number, $p = aq + 1$ be a prime number, and g be a generator of $\mathcal{G} \subset \mathbb{Z}^*_p$ of order q. g, p, q are public parameters. Let $x \in_{\mathsf{u}} \mathbb{Z}_q$ be the secret key and $y = g^x \bmod p$ be the corresponding public key. The signature σ of a message m is

$$\sigma = (\sigma_e, \sigma_s)$$

$$\sigma_e = h(m \| g^k \bmod p), \qquad \sigma_s = \sigma_e x + k \bmod q$$

for some random $k \in_{\mathsf{u}} \mathbb{Z}^*_q$. One can verify the signature by checking

$$\sigma_e \stackrel{?}{=} h\left(m \| g^{\sigma_s}y^{-\sigma_e} \bmod p\right) \ .$$

Intuitively, we will keep σ_s private. So, $s := \sigma_s$. Then, we should fine u and v such that $u^s = v$. we should keep enough information to check the signature and so we let $\xi := \sigma_e$. With respect to the generic ElGamal, we finally define

- $u := g$,
- $v := g^s \bmod p$,
- $\xi := h(m \| g^k \bmod p)$ $(= \sigma_e)$,
- $s := \xi x + k \bmod q$ $(= \sigma_s)$,
- and the $\mathsf{ver}(y, m, (u, v, \xi))$ algorithm consists in checking $h\left(m \| v y^{-\xi} \bmod p\right) \stackrel{?}{=} \xi$.

We clearly have $u^s = v$.

The DSA signature. Let q be a large enough prime number, $p = aq + 1$ be a prime number, and g be a generator of $\mathcal{G} \subset \mathbb{Z}_p^*$ of order q. g, p, q are public parameters. Let $x \in_{\mathsf{u}} \mathbb{Z}_q$ be the secret key and $y = g^x \bmod p$ be the corresponding public key. The signature σ of a message m is

$$\sigma = (\sigma_r, \sigma_s)$$

$$\sigma_r = \left(g^k \bmod p\right) \bmod q, \qquad \sigma_s = \frac{h(m) + x\sigma_r}{k} \bmod q$$

for some random $k \in_{\mathsf{u}} \mathbb{Z}_q^*$. One can verify the signature by checking

$$\sigma_r \stackrel{?}{=} \left(g^{\frac{h(m)}{\sigma_s} \bmod q} y^{\frac{\sigma_r}{\sigma_s} \bmod q} \bmod p\right) \bmod q \ .$$

With respect to the generic ElGamal, we finally define

- $u := \left(g^k \bmod p\right) \bmod q$ $(= \sigma_r)$,
- $v := \left(g^{h(m) \bmod q} y^u \bmod p\right) \bmod q$,
- $\xi := \emptyset$,
- $s := \frac{h(m)+xu}{k} \bmod q$ $(= \sigma_s)$,
- and the $\mathsf{ver}(y, m, (u, v, \xi))$ algorithm checks $v \stackrel{?}{=} g^{h(m) \bmod q} y^u \pmod p$.

On can verify that $u^s = v$ by writing

$$u^s = \left(g^{k\frac{h(m)+xu}{k} \bmod q} \bmod p\right) \bmod q = \left(g^{h(m) \bmod q} (g^x)^u \bmod p\right) \bmod q = v \ .$$

The ECDSA signature. It works over an elliptic curve. The difficulty with ECDSA (see [Vau06]) is to deal with objects of many different types: elliptic curve points, field elements, integers, and bit-strings. The standard [ECD98] provides extensive details about how to represent and manipulate them.

There are two possible types of finite fields:

- fields of characteristic two for which the elliptic curve equation over $\mathsf{GF}(q)$ is defined by $y^2 + xy = x^3 + ax^2 + b$,
- large prime fields for which the elliptic curve equation over $\mathsf{GF}(q)$ is defined by $y^2 = x^3 + ax + mb$.

The public parameters consist of the field cardinality q, the selected field type, the elliptic curve $\mathbb{C}$ defined by the parameters a and b, a prime number $n > 2^{160}$, and an element $G \in \mathbb{C}$ of order n. These parameters are subject to many security criteria. Here we use additive notations. Let $d \in_{\mathsf{u}} [1, n-1]$ be the secret key and $Q = dG$ be the corresponding public key. The signature σ of a message m is

$$\sigma = (\sigma_r, \sigma_s)$$

$$\sigma_r = \overline{(kG)_x} \bmod n, \qquad \sigma_s = \frac{h(m) + d\sigma_r}{k} \bmod n$$

for some random $k \in_{\mathsf{u}} [0, n-1]$ where $(P)_x$ denotes the x-coordinate of an elliptic curve point $P \in \mathbb{C}$ and $\overline{e}$ denotes a standard way to convert a field element e into an integer. If $\sigma_r = 0$ or $\sigma_s = 0$, try with another random k. One can verify the signature by checking that $Q \stackrel{?}{\in} \mathbb{C}$, $Q \stackrel{?}{\neq} \mathcal{O}$, $nQ \stackrel{?}{=} \mathcal{O}$, $\sigma_r, \sigma_s \stackrel{?}{\in} [1, n-1]$, and

$$\sigma_r \stackrel{?}{=} \overline{\left(\frac{h(m)}{\sigma_s} G + \frac{\sigma_r}{\sigma_s} Q\right)_x} \bmod n \ .$$

Indeed, if everything works well we should have

$$\sigma_r = \overline{\left(\frac{h(m)}{\sigma_s} G + \frac{\sigma_r}{\sigma_s} dG\right)_x} \bmod n = \overline{\left(\frac{h(m) + d\sigma_r}{\sigma_s} G\right)_x} \bmod n = \overline{(kG)_x} \bmod n$$

With respect to the generic ElGamal, we finally define

- $u := kG$,
- $v := h(m)G + \overline{u_x}Q$,
- $\xi := \emptyset$,
- $s := \frac{h(m) + d\overline{u_x}}{k} \bmod n$,
- and the $\mathsf{ver}(Q, m, (u, v, \xi))$ algorithm consists in checking $v \stackrel{?}{=} h(m)G + \overline{u_x}Q \bmod n$

On can verify that $u\, s = v$ (additive notation) by writing

$$u\, s = kG \frac{h(m) + d\overline{u_x}}{k} \bmod n = h(m)G + \overline{u_x}(dG) = v \ .$$

In order to build a non-transferable signature, instead of revealing the private part of the signature, we will prove that we know it. Clearly, we will use a zero-knowledge proof as before. The required proof of knowledge should allows **P** to prove to **V** that he knows s such that $u^s = v$. Note that this is the proof of the knowledge of the discrete logarithm. The identification protocol from Schnorr [Sch90, Sch91] is a Σ-protocol when q is prime proving exactly that. Consequently, we applied our generic transform from Theorem 12.11 and we obtain the verification protocol of Figure 12.12 which is deniable zero-knowledge in the CRS model. We thus obtain several schemes: ONTAP-ElGamal, ONTAP-Schnorr, ONTAP-DSA, ONTAP-ECDSA, and so on.

P		**V**
input: σ_p, σ_s	**common input:** crs, K_p, m	
$(\sigma_p = (u, v, \xi), \sigma_s = s)$		pick $r \in_{\mathsf{u}} \{0,1\}^t$
pick $\ell \in_{\mathsf{u}} [0, q-1]$	$\xleftarrow{c}$	$(c, d) = \mathsf{commit}(\mathsf{crs}, r)$
$a = u^\ell \bmod p$	$\xrightarrow{a \Vert \sigma_p}$	check $\mathsf{ver}(K_p, m, \sigma_p) \stackrel{?}{=} 1$
$r = \mathsf{open}(\mathsf{crs}, c, d)$	$\xleftarrow{d}$	$\sigma_p = (u, v, \xi)$
$z = \ell + r \cdot s \bmod q$	$\xrightarrow{z}$	check $u^z \stackrel{?}{=} av^r \pmod p$

Figure 12.12. *The* iProof *Protocol for ONTAP-ElGamal.*

Theorem 12.15 (ONTAP-ElGamal).
Assume that the ElGamal-based signature is EF-CMA-secure and that the commit$(\cdot)$ *is a trapdoor commitment scheme in the CRS model (resp. the RO commitment scheme). The digital signature scheme added to the signature proof* iProof *of Figure 12.12 forms an ONTAP scheme in the CRS model (resp. in the RO model). The soundness error is* 2^{-t}.

With an extra round we obtain an ONTAP in the standard model as depicted in Figure 12.6.

Proof.
The proof is similar to the one of Theorem 12.13. It relies on Theorem 12.9 and Theorem 12.11.

■

12.6 Comparison with Other Works

As the UDVSP of Baek *et al.* [BSNS05] our ONTAP definition requires no PKI for verifiers. UDVSP and ONTAP are conceptually equivalent but the security notions differ. The UDVSP security from [BSNS05] uses three definitions, restrict to known message attacks and

honest verifiers while we use two definitions, chosen message attacks and malicious verifiers. In addition to this, our instantiations of ONTAP can be built efficiently on standard signature schemes and need no change for the signing algorithm. UDVSP does not apply directly to RSA- and ElGamal-based schemes while our proposed ONTAP implementations do. Our proposed implementations just require a modular exponentiation for the prover, i.e., the e-passport, while the proposed UDVSP constructions require the use of bilinear mappings as well as signature transforms.

Recently, Shahandashti and Safavi-Naini [SSN08] presented a construction for UDVS. As we saw before, UDVS requires PKI for verifiers and thus is not adapted in our case. However, they present a way for a signature holder to prove his signature knowledge to a verifier. They define a signature class $\mathbb{C}$ for which signatures can be converted in a public and a private part. The private part is simulatable and there exists a proof of knowledge for the private part. The signature holder simply needs to convert its signature, to send the public part to the verifier, and finally to prove his knowledge of the private part. They use this definition to designate a signature to a verifier by using a Fiat-Shamir transform on the interactive proof. Except the transform, we use a similar idea, i.e., a signature in two parts, one simulatable and the other provable. The authors does not give any security proof (since it is not their main contribution). They use a classical Σ-protocol and thus their proof of knowledge is HVZK only. As seen before, HVZK is clearly not enough for the application we have in mind. Here we strengthen the knowledge proofs, we give formal security proofs and examples of implementations. Finally the scheme of [SSN08] may loose deniability if a malicious verifier registers a rogue key.

As the undeniable signature scheme of Gennaro, Krawczyk, and Rabin [GKR00], a proposed instantiation of our solution is based on RSA. However, our protocol is a proof of knowledge based on Guillou-Quisquater while the confirmation protocol of Gennaro *et al.* shows that two elements were raised to the same exponent. This leads to two conceptually different protocols.

12.7 Application to Electronic Passports

Today, electronic passports, or e-passports, are available in many countries. They are formally called *machine-readable travel documents* (MRTD). An e-passport embeds an RFID tag allowing to check the traveler identity through a wireless communication. They also store private information, biometric data, and a digital signature issued by a national authority. The implementation, the use, the security, and all others things related to passports are mandated by the International Civil Aviation Organization (ICAO).

As non-electronic passports, e-passports contain a Machine Readable Zone (MRZ) as depicted in Figure 12.14. The MRZ is aimed to be read by an optical scanner. It consists of two lines with basic information such as the document type, the document number, the

Figure 12.13. *How to Distinguish an E-passport? (source: www.passeportsuisse.ch)*

document expiry date, the holder name, the holder gender, the holder date of birth, and so on.

To ensure global interoperability of e-passports, the stored information should follows the Logical Data Structure (LDS) as written in the standard [MRT04a]. The LDS is split in nineteenth Data Groups (DG). As examples, the first data group, DG1, contains the digital version of the MRZ, DG2 contains a facial digital picture in JPEG format optimized for face recognition, and DG3 may contains the encoded fingerprint(s). Note that only DG1 and DG2 are mandatory.

12.7.1 Passive versus Active Authentication

Any electronic information (with no cryptographic protection) is subject to copies and/or alterations. Stored information must be authenticated to prove that they are genuine and not forged.

Passive authentication. The mandatory solution is to use the *passive authentication*. In addition to the LDS data groups, the chip also contains a *Document Security Object* (SOD). The SOD includes the hash representations (digests) of the LDS data groups and a digital signature of this list of digests. It may contain a certificate of the signer's public key. The signature is issued by the authority of the country and prove that the data are authentic and not altered. To verify the signature, one needs to obtain the

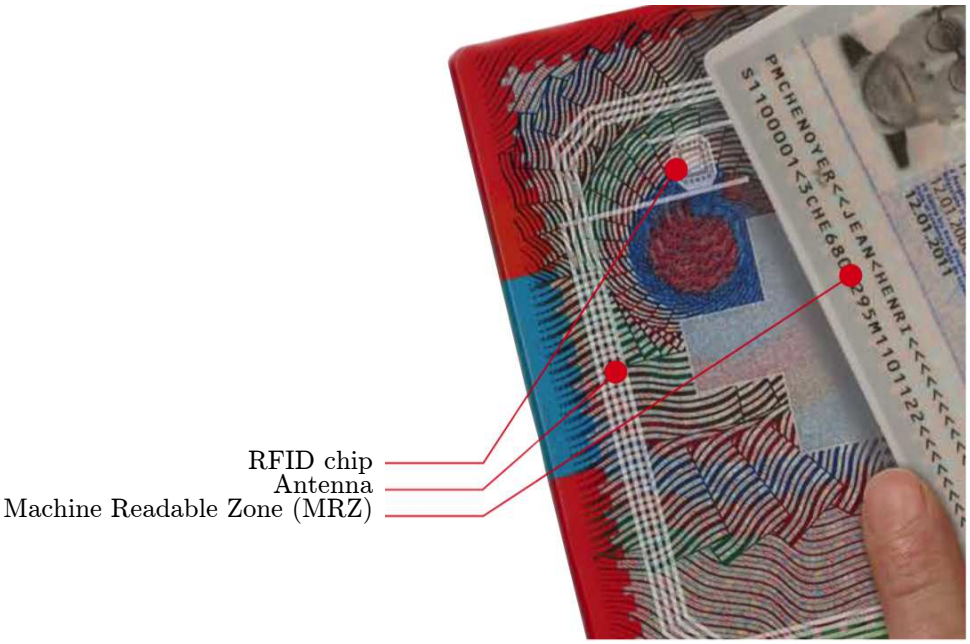


Figure 12.14. *E-passport Main Components (source: www.passeportsuisse.ch)*

certificate of the authority. Sometimes the certificate of the authority is also stored in the LDS, but if it is not the case it can be obtained from the ICAO public-key directory. The SOD are mandatory following the ICAO standard.

Optional Active authentication. *Active authentication* aims to prevent the substitution or the cloning of the chip. In short, the chip contains a pair of public/private key. The private key is stored in the chip in a secure part of the memory while the public key is accessible by the reader in DG15. The reader sends a challenge to the passport. The passport signs the challenge with its private key and sends back the signature. The reader now can check the validity of the signature by using the public key stored in DG15. Remember that data groups are authenticated by the SOD and so the public key is authenticated. A clone of the chip is impossible since there is no way to recover the private key from the secure memory and there is no way to modify the whole key pair since the public key is authenticated by the SOD.

12.7.2 Optional Basic and Extended Access Controls

Since the chip in the passport has wireless access, these data may be captured without the agreement of the holder. Let Alice be walking with her e-passport in her pocket. With no access control anyone can obtain the data from her e-passport while it is just stored in

her pocket. The ICAO standard proposes security offers to avoid unauthorized data access. First of all, e-passports may have a shield (metallic cover) abolishing the radio waves which avoid the access to the chip unless the e-passport is opened. This solution avoids a little the identity tracking, but does not really protect the data privacy. Note that it is not longer used by the countries (actually, only the US passports use it). The ICAO standard propose two optional protocols avoiding unauthorized access to the chip:

None. There is no access control.

(Optional) Basic Access Control. In short, the *basic access control* (BAC) avoids unauthorized access to the chip and avoids communications eavesdropping.

To avoid unauthorized access, the BAC forces the reader to establish a visual contact with the MRZ of the e-passport. Indeed, the access to the data stored in the chip is only possible if the reader knows a secret key which is derived from the MRZ.

To avoid communications eavesdropping, the BAC then use a key derivation mechanism in order to establish session keys for secure messaging between the chip and the reader.

It is not a real access control since anyone can implement an e-passport reader and obtains data from any passport without being authorized by public authorities. BAC is well known to provide a very small protection against unauthorized access. It is by far insufficient since the new generation of e-passports will contain more private information such as fingerprint, address, etc.

While access control is optional, today e-passports from almost all countries implement the basic access control (BAC).

(Optional) Extended Access Control. The European Union is now promoting the *extended access control* (EAC) which is based on more elaborate cryptographic protocols (ECDH key agreement with key authentication) and terminal authentication based on a specific public key infrastructure (PKI). Each border patrol would possess a certificate and the e-passport may verify it before giving sensitive data. This PKI is also known to suffer from weaknesses. First, the revocation procedure is unreliable: an e-passport is completely disconnected, so he has no reliable clock and certificate revocations become very difficult in this setting. Secondly, border patrol certificates can only be used by authorized countries. The visiting country presents a certificate to the e-passport which is able to verify its validity only if the certificate was signed by the national authority (from the home country). However, for foreign countries who has no agreement with the home country, the e-passport should be able to prove the identity of its owner. So, these countries are limited to the use of BAC. This means that the e-passport has no mean to check certificates from unauthorized readers. So, the e-passport will offer a better security in the countries with agreements than countries without it. This seems pretty weird.

In addition to this, EAC is only meant to protect non-mandatory data groups since mandatory ones should still be accessible to countries with no agreement to read extra information. This means that DG1, DG2 are not protected and worse that the SOD is not protected by EAC so will still leak evidence that a given protected data group is correct. Clearly, an adversary can still distinguish a correct EAC-protected data group from an incorrect one without being authorized to read it.

12.7.3 E-Passport Passive Authentication Issue

As pointed out by Monnerat, Vaudenay, and Vuagnoux [MVV07], the passive authentication is a big loss for privacy.

Indeed, the actual implementation following the ICAO standard allows someone having access to a passport to get the private information and a proof of it. The main problem is that the proof can be *transferred*. We note that there is two manners to transfer a proof:

- The first manner is to run a relay attack, also known as the Mafia fraud attack. It is some kind of (online) man-in-the-middle attack in which an adversary uses a distant passport to prove its identity to a verifier.
- The second manner is simply to store all information (accessible offline) and the signature.

We focus on the second threat. Note that a possible countermeasure against the first threat may be to use some distance bounding solution (see Section 2.5.2).

12.7.4 Deniable Zero-Knowledge in Signature Verification

First of all, note that to execute the optional *active authentication*, the chip has to carry out some RSA computation(s). Therefore, we can assume that chips are able to run one or two RSA computations. Following the ICAO standard, we also note that e-passports should use RSA, DSA, or ECDSA signatures.

One solution for e-passports is to avoid the signature of the SOD to be revealed. The e-passport should prove that it knows the signature (a proof of knowledge) but it should not reveal it (zero-knowledge proof). Thus, we propose to have the signature part of the SOD hidden and passive authentication replaced by an Offline Non-Transferable Authentication Protocol (ONTAP) which is a deniable zero-knowledge authentication protocol, see Section 12.3. We gave the necessary material to implement an ONTAP protocol for the signature standards used in e-passports, e.g., RSA, DSA, and ECDSA, in Sections 12.5.1 and 12.5.2.

Chapter

THIRTEEN

Building Secure Schemes based on Weak Hash Functions

A textbook signature scheme usually does a poor job because it is restricted to input messages of fixed length and is often weakly secure as described in Section 11.2. In order to sign messages of arbitrary length, hash functions [Riv91, Riv92, SHA93, SHA95] and the so-called *hash-and-sign paradigm* appeared. Clearly, collision attacks on the hash functions lead to forgeries. Therefore, hash functions must be collision resistant. The problem is that their collision resistance is not as high as expected as evidenced by the recent successful attacks by Wang *et al.* [WY05, WYY05a, WYY05b].

Here, we wonder how to recycle signature schemes that are currently implemented and based on (now) weak hash functions. To do so, we consider *generic* transforms using preprocessing based on [BR97, HK06a, Mir06].

We first recall the hash-and-sign paradigm. The first variant shows how a random oracle brings randomness (Section 13.1.1), the second one shows how a collision-resistant hash function extends the input domain (Section 13.1.2), and the last one presents the original hash-and-sign paradigm with a random oracle (Section 13.1.3).

The hash-and-sign paradigm is proven in the random oracle model while in practice hash functions are used instead. As said before, hash functions deviate more and more from this idealization. In particular, collisions may be reported.

A natural solution to avoid the collision-resistance assumption on the hash function is to add randomness in hashing. As detailed in Section 13.1.4, Bellare and Rogaway [BR97] proposed to sign $(K, H_K(m))$ with a random salt K where H is a Target Collision Resistant (TCR) hash function, also known as Universal One-Way Hash Function (UOWHF).

More recently, Halevi and Krawczyk [HK06a] proposed the concept of enhanced TCR (eTCR) hash function, some eTCR construction techniques, and the RMX construction based on current hash functions. This latter scheme only adds a randomized pre-processing on the input message and thus standard implementations can be used as-is. As detailed in Section 13.1.5, they suggest to use eTCR functions as pre-processing for signature schemes and, as a consequence, the salt K does not need to be signed.

In a first time, we will look for a security model which fits in a better way the real hash functions. Indeed, we develop in Section 13.2 the Preimage-Tractable Random Oracle Model (PT-ROM) which was introduced by Liskov [Lis07].

In Section 13.3, we prove in our PT-ROM that the construction with an eTCR pre-processing is *strongly* secure based on any textbook signature scheme which is *weakly* secure.

The disadvantage of the methods using a random seed κ is that κ must be appended to the signature. To avoid the increase in signature length, Mironov [Mir06] proposed for DSA [DSS94, DSS00], RSA-PSS [BR96], and the Cramer-Shoup [CS00] schemes to re-use the randomness from the signature scheme instead of adding a new one. In Section 13.4, we generalize this construction and propose a *generic* transform that applies to special signature schemes. Indeed, we define special signature schemes for which we can split the signature algorithm in two parts: first, there is a randomized algorithm independent from the input message, then, there is a deterministic algorithm which outputs the signature. We call these schemes Signatures with Randomized Precomputation (SRP). This assumption makes the pre-processing transform less generic because the signature must generate some random coins which must be available before the message is processed and extractable from the signature.

We published the PT-ROM, the strengthening signature construction, as well as the generic entropy recycling technique in [PV07].

13.1 Hash-and-Sign variants Today

Building a signature scheme for messages of arbitrary length based on a scheme restricted to fixed message length implies that a function H taking in input messages from an infinite domain $\mathbb{M}$ and outputs an elements of a finite space $\mathbb{M}_H$ exists. Such a signature schemes follows the *hash-and-sign* paradigm.

13.1.1 Adding Unpredictability

We first consider that we have access to a random oracle R. Our construction S' is a *strongly secure* FML-DS based on a *weakly secure* FML-DS S. The construction is depicted in Figure 13.1 and formally works as follows:

$$\mathsf{S}'.\mathsf{sign}(K_s, m) = \mathsf{S}.\mathsf{sign}\left(K_s, \mathsf{R}(m)\right) \quad . \tag{13.1}$$

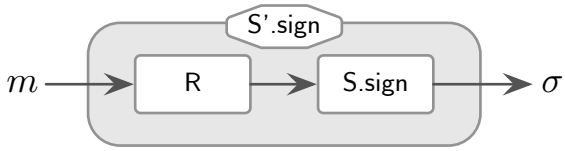


Figure 13.1. *Hash-and-Sign Bringing Unpredictability (with a Random Oracle* R*).*

Theorem 13.1 (Hash-and-Sign Paradigm, Unpredictability).
Consider R *is a random oracle bounded by* q *queries with* n*-bit input strings and* k*-bit output strings. In addition, consider* S *is any* (T, ℓ, ε)*-UF-KMA resistant FML-DS scheme with* s*-bit output strings.*

The signature construction S' *defined by Equation* (13.1) *is* $(T - \mu, \ell, q \cdot \varepsilon)$*-EF-CMA resistant where* $\mu = \mathcal{O}\left(q(k+n)\right)$*.*

Note that we could directly use this (folklore) construction to reach AML-DS, but this would prevent us from identifying the role of collision resistant hash functions. Indeed, random oracles usually play two roles: bringing *collision resistance* and *unpredictability* [Can97]. Here, we would like to separate the two roles. In Theorem 13.1 when R has a fixed input length (hence for FML-DS) we identify where unpredictability is used. The next Theorem 13.2 uses collision resistance. Finally, Theorem 13.3 will combine both.

Another motivation would be to replace collision resistant hash functions in existing signature schemes by keeping the same security as detailed in subsequent sections.

Proof.
Clearly, the steps of the construction S' are in the following order:

$$m_i \in \{0,1\}^n \xrightarrow{\mathsf{R}} r_i \in \{0,1\}^k \xrightarrow{\mathsf{sign}} \sigma_i \in \{0,1\}^s \quad .$$

Consider any $(T - \mu)$-time adversary $\mathcal{A}$ playing the EF game in the CMA model against our constructed scheme S'. $\mathcal{A}$ can access to oracles for R and for $\mathsf{S}'.\mathsf{sign}$. Using an algorithm $\mathcal{B}$ of complexity at most μ, we can transform $\mathcal{A}$ in an adversary playing the UF-KMA game against S as depicted in Figure 13.2 where $\mathcal{C}$ plays the role of the challenger in the UF-KMA game (with respect to S).

Without loss of generality, we assume that before querying the signing oracle with a message m, $\mathcal{A}$ always query R with m. We also assume that queries to R are pairwise different and the forged message $\widehat{m}$ is queried to R at some point.

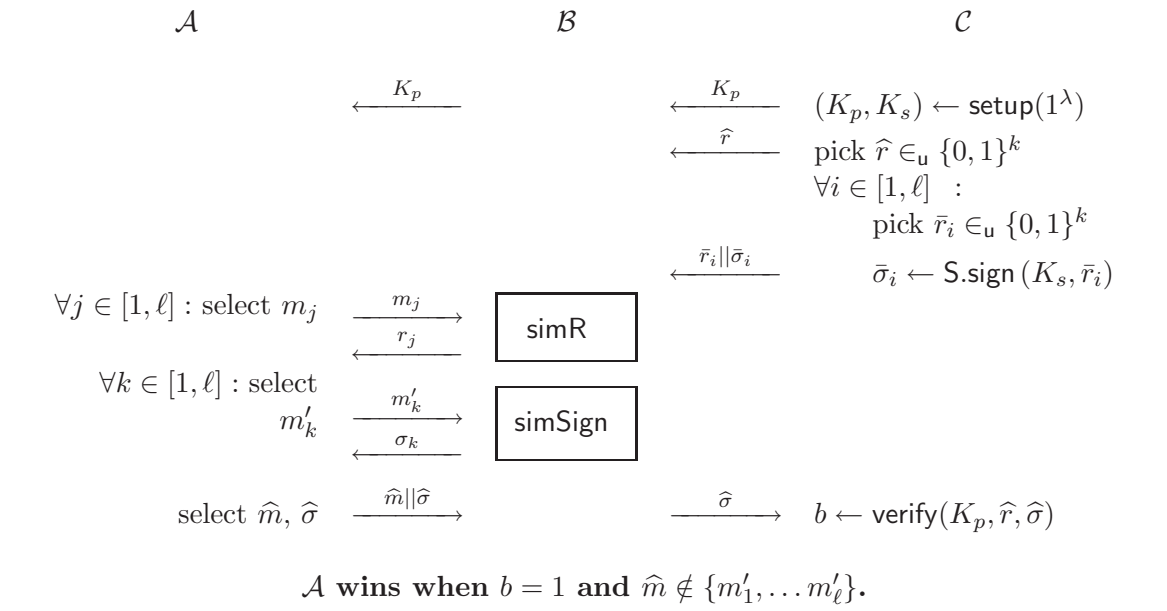


Figure 13.2. *Reduction to the UF-KMA Game Against* S.

$\mathcal{B}$ has to simulate for $\mathcal{A}$ the R and $\mathsf{S.sign}$ oracles that we refer by simR and $\mathsf{simSign}$ respectively. The simulations work as follows:

simR **algorithm.** At the beginning of the game, $\mathcal{B}$ picks a random $t \in_\mathsf{u} [1, q]$. When $\mathcal{A}$ submits an R-query with input m_j, it answers with the next $\bar{r}_i$ in the sequence, i.e., $r_j = \bar{r}_i$, except for the t^{th} query for which it answers with $\widehat{r}$, i.e., $r_t = \widehat{r}$.

$\mathsf{simSign}$ **algorithm.** When $\mathcal{A}$ submits a sign-query, the input message m'_k was queried before to simR. So, $\forall j \neq t$, the $\mathsf{R}(m_j)$ is equal to an $\bar{r}_i$ and for $j = t$, it is equal to $\widehat{r}$. Except in the $\widehat{r}$ case, $\mathcal{B}$ answers by the corresponding $\bar{\sigma}_i$, otherwise it fails.

If $\mathcal{A}$ succeeds, he will send a (forged) valid pair $(\widehat{m}, \widehat{\sigma})$ to $\mathcal{B}$. Note that $\widehat{m}$ was queried to R and thus $\mathsf{R}(\widehat{m}) = r_t = \widehat{r}$ with probability $1/q$. Since a query to R with input $\widehat{m}$ will output $\widehat{r}$ with probability $1/q$, $\mathsf{S.sign}(K_s, \widehat{r})$ will be equal to $\mathsf{S'.sign}(K_s, \widehat{m})$ with probability $1/q$. Thus, $\mathcal{B}$ simply has to forward $\widehat{\sigma}$ to $\mathcal{C}$ and wins with probability $1/q$.

Clearly, $\mathcal{B}$ perfectly simulates a challenger for the adversary $\mathcal{A}$ which plays the EF-CMA game. $\mathcal{B}$ plays the UF-KMA game with $\mathcal{C}$ and wins its game when $\mathsf{R}(\widehat{m}) = \widehat{r}$ which appears when $\mathcal{B}$ guessed the right t at the beginning, i.e., with probability $1/q$.

■

13.1.2 Domain Extension

Using a full collision resistant hash function (CRHF) H, the hash-and-sign paradigm allows to build a secure signature scheme for messages of arbitrary length. As we noted before, collision resistance is actually no longer considered as a reasonable assumption. The construction S' is a strongly secure *FML-DS* based on a strongly secure *AML-DS* S. The construction is depicted in Figure 13.3 and formally works as follows:

$$\mathsf{S}'.\mathsf{sign}(K_s, m) = \mathsf{S}.\mathsf{sign}\left(K_s, H(m)\right) \quad . \tag{13.2}$$

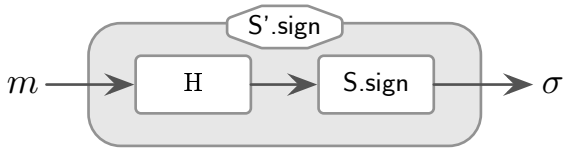


Figure 13.3. *Hash-and-Sign Extending the Domain (with a CRHF H).*

Here is the corresponding security result from [Rog06].

Theorem 13.2 (Hash-and-Sign Paradigm, Domain Extension).
Consider $H : \{0,1\}^* \to \{0,1\}^k$ be a collision resistant hash function and S be a (T, ℓ, ε)-EF-CMA resistant FML-DS scheme with at least k-bit input messages.

The AML-DS construction S' defined by Equation (13.2) is $(T-\mu, \ell, \varepsilon+\ell\cdot 2^{-k})$-EF-CMA resistant where $\mu = \mathcal{O}(\ell \cdot k + L)$ and L is a bound on the total length of messages to be signed.

In short, if H is a collision resistant hash function and S is a strongly-secure FML-DS, then S' is a strongly-secure AML-DS.

Proof.
Consider an algorithm $\mathcal{B}$ and an adversary $\mathcal{A}$ against our constructed scheme S'. As in the previous proof, we use both of then to attack the scheme S. The reduction is depicted in Figure 13.4. Note that an attack against the scheme S' exists if and only if $\widehat{m} \notin \{m_1, \ldots, m_\ell\}$. Equivalently for the scheme S, the attack exists if and only if $\widehat{h} \notin \{h_1, \ldots, h_\ell\}$.

Suppose $\mathcal{A}$ wins his game with probability ε (against S'). $\mathcal{A}$ wins if $\widehat{m} \notin \{m_1, \ldots, m_\ell\}$ but it is perhaps not the case for the $\widehat{h}$. Indeed collisions can occur between $H(\widehat{h})$ and the set $\{H(h_1), \ldots H(h_\ell)\}$ with probability $p = 1 - (1 - 2^{-k})^\ell \leq \ell \cdot 2^{-k}$. In such a case, the attack against S fails, i.e., with probability at most $\ell \cdot 2^{-k}$.

■

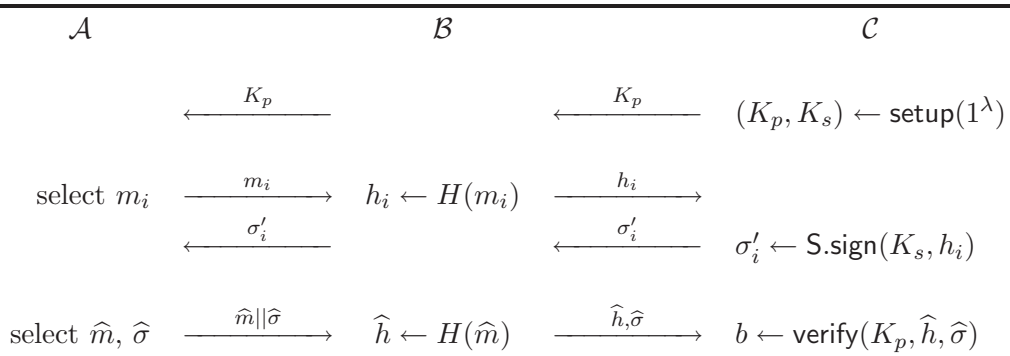


Figure 13.4. *Reduction to the EF-CMA Game Against* S.

13.1.3 Both at the Same Time

Combining Theorem 13.1 (adding unpredictability by using a random oracle) and Theorem 13.2 (extending the message space using a hash function), we obtain a *strongly secure AML-DS* that is the so called hash-and-sign paradigm. The construction is depicted in Figure 13.5.

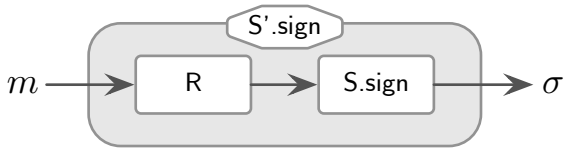


Figure 13.5. *The Hash-and-Sign Paradigm (with a Random Oracle* R*).*

Here is the corresponding security result from [Rog06].

Theorem 13.3 (Hash-and-Sign Paradigm).
Let $\mathsf{R} : \{0,1\}^* \to \{0,1\}^k$ *be a random oracle and* S *be a* (T, ℓ, ε)*-UF-KMA resistant FML-DS scheme with at least* k*-bit input messages.*

The AML-DS construction S' *defined by*

$$\mathsf{S}'.\mathsf{sign}(K_s, m) = \mathsf{S.sign}\,(K_s, \mathsf{R}(m))$$

is $(T - \mu, \ell, \varepsilon + \ell \cdot 2^{-k})$*-EF-CMA resistant where* $\mu = \mathcal{O}(\ell \cdot k + L)$ *and* L *is a bound on the total length of messages to be signed.*

The proof of this folklore result is rather straightforward. Indeed, H brings *collision resistance* in domain extension as well as *unpredictability*.

In short, if R is a random oracle and S is a weakly-secure FML-DS, then S' is a strongly-secure AML-DS.

13.1.4 Randomized Hash-and-Sign Paradigm

The idea of using a TCR function comes from Bellare and Rogaway [BR97]. It was also reused recently by Mironov [Mir06]. As depicted in Figure 13.6, the constructed signature consists of the pair randomness-signature as follows:

$$\mathsf{S}'.\mathsf{sign}(K_s, m) = (\kappa,\ \mathsf{S}.\mathsf{sign}(K_s, \kappa \| H_\kappa(m)))$$

where κ is some random coins and $H_\kappa(\cdot)$ is a TCR hash function.

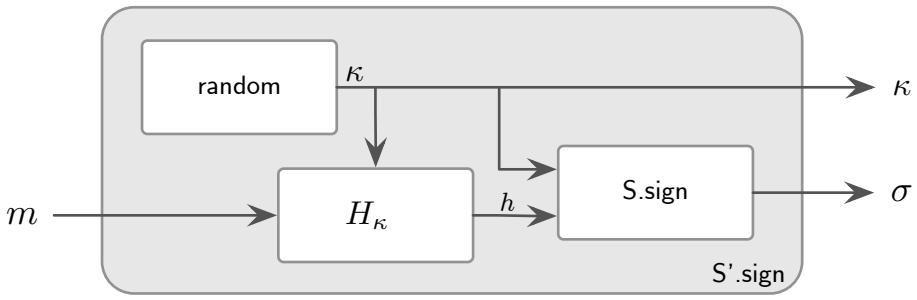


Figure 13.6. *The Randomized Hash-and-Sign Paradigm (with a TCR Function H).*

The following result is a straightforward generalization of Mironov [Mir06].

Theorem 13.4 (Randomized Hash-and-Sign Paradigm with a TCR Function). *Consider an FML-DS S with domain $\{0,1\}^r$ and a function $G : \{0,1\}^* \mapsto \{0,1\}^r$. We assume that $G(X)$ is indistinguishable from $Y \in_{\mathsf{u}} \{0,1\}^r$ when $X \in_{\mathsf{u}} \{0,1\}^{2r}$. Let $H : \{0,1\}^k \times \{0,1\}^* \mapsto \{0,1\}^n$ be a TCR hash function and $\mathsf{R} : \{0,1\}^{k+n} \mapsto \{0,1\}^r$ be a random oracle. We construct two AML-DS S^* and S' by*

$$\begin{array}{rcll} \mathsf{S}^*.\mathsf{sign}(K_s, m) & = & \mathsf{S}.\mathsf{sign}\,(K_s, G(m)) & \\ \mathsf{S}'.\mathsf{sign}(K_s, m) & = & (\kappa \parallel \mathsf{S}.\mathsf{sign}\,(K_s, \mathsf{R}\,(\kappa \| H_\kappa(m)))) & \text{with } \kappa \in_{\mathsf{u}} \{0,1\}^k \end{array}$$

Assuming that S^* is EF-CMA-secure, then S' is also EF-CMA-secure.

This means that if there exists a domain extender G that makes S^* strongly-secure, then S' is strongly-secure.

Proof.
Consider $H : \{0,1\}^k \times \{0,1\}^* \mapsto \{0,1\}^n$ is a $(T + \mu_H, \varepsilon_H)$-TCR hash function for μ_H to be defined later, $\mathsf{R} : \{0,1\}^{k+n} \mapsto \{0,1\}^r$ is a random oracle bounded to q queries,

and S an FML-DS scheme with r-bit input messages. We assume that the construction S^* is $(T+\mu_S, \ell, \varepsilon_S)$-EF-CMA secure for μ_S to be defined later. We assume that G is $(T+\mu_G, q+\ell+1, \varepsilon_d)$-PRG when restricted to $(2r)$-bit inputs. We will prove that the construction S' is $(T, \ell, \varepsilon_S + \ell\varepsilon_H + \varepsilon_c + \varepsilon_d)$-EF-CMA secure where ε_c represents a probability of collision on the outputs of the random oracle.

We consider an adversary $\mathcal{A}$ playing the EF-CMA game against S'. We assume without loss of generality that $\mathcal{A}$ queries R with $\kappa \| H_{\widehat{\kappa}}(\widehat{m})$ before releasing the final forgery $(\widehat{m}, \widehat{\kappa}, \widehat{\sigma})$ (so we have up to $q+1$ queries to R). By using an algorithm $\mathcal{B}$, we prove that we can reduce $\mathcal{A}$ to an adversary either against the signature construction S^* or either against the TCR hash function H.

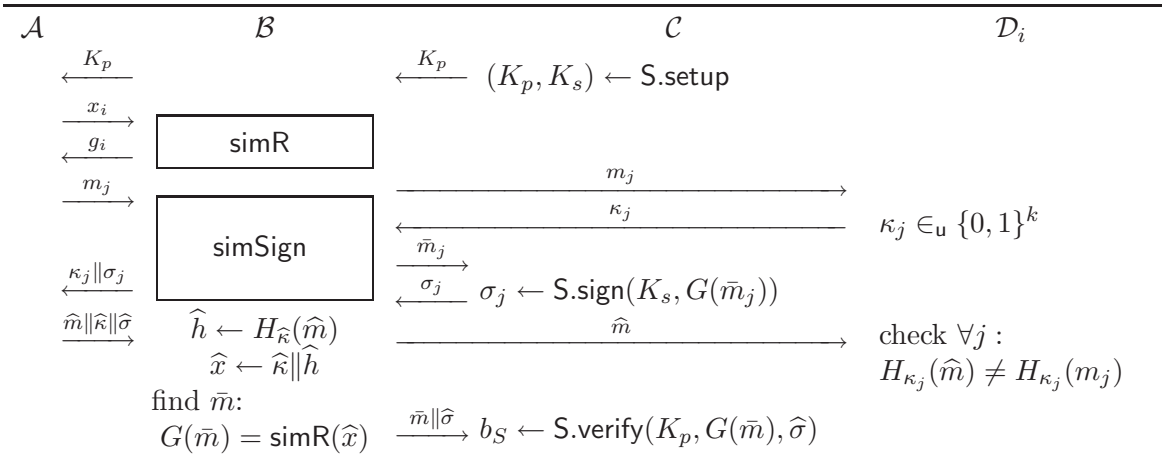


Figure 13.7. *Reduction to EF-CMA or TCR Games (from EF-CMA).*

The reduction is depicted in Figure 13.7 where $\mathcal{C}$ plays the role of the challenger in the EF-CMA game of Figure 11.2 for S^* and each $\mathcal{D}_i$ plays the role of the the challenger in the TCR game of Figure 3.3. Clearly, $\mathcal{B}$ has to simulate the random oracle R and the signing oracle that we refer to simR and $\mathsf{simSign}$ respectively. The simulations work as follows:

simR: $\mathcal{B}$ manages a table $\mathcal{T}$ initially empty. For each R-query with input x:

- if $\mathsf{simR}(x)$ is not defined in $\mathcal{T}$, $\mathcal{B}$ picks a random $\bar{m}$ uniformly in $\{0,1\}^{2r}$ and answers $g \leftarrow G(\bar{m})$. Hence, a new entry $(x, g, \bar{m})$ is inserted in $\mathcal{T}$, meaning $\mathsf{simR}(x) = g = G(\bar{m})$. Note that the third entry $\bar{m}$ will be used by $\mathsf{simSign}$ only.
- otherwise, i.e., if $\mathsf{simR}(x)$ is defined in $\mathcal{T}$, $\mathcal{B}$ answers $g = \mathsf{simR}(x)$ as defined in $\mathcal{T}$.

$\mathsf{simSign}$: For each sign-query with input m:

1. $\mathcal{B}$ computes $x \leftarrow \kappa \| H_\kappa(m)$ where κ is returned by $\mathcal{D}_i$ on query m,
2. $\mathcal{B}$ queries $\mathsf{simR}(x)$. Let $\bar{m}$ be such that $\mathsf{simR}(x) = G(\bar{m})$ from $\mathcal{T}$,

3. $\mathcal{B}$ queries $\mathcal{C}$ with $\bar{m}$ to obtain its signature σ,

4. finally, $\mathcal{B}$ returns $\kappa \| \sigma$ to $\mathcal{A}$.

$\mathcal{B}$ is allowed to ℓ queries to the $\mathsf{S.sign}$ oracle, so $\mathcal{A}$ is also allowed to ℓ queries to $\mathsf{simSign}$. Note that the $\mathsf{simSign}$ simulation is perfect but the simR simulation is not. At the end, if $\mathcal{A}$ succeeds, he returns a forged pair $(\widehat{m}, \widehat{\kappa}, \widehat{\sigma})$ to $\mathcal{B}$. We use the proof methodology of Shoup [Sho04]:

- Let game_0 be the normal EF-CMA game against S' (with R and $\mathsf{S.sign}$ oracles).
- Let E_1 be the event that there were no collision on the output of R. Let game_1 be game_0 in which the winning condition is modified by mandating E_1 to occur.

  Clearly, when E_1 does not occur, there is a collision on the R outputs. Since there is at most $q + \ell + 1$ elements in the simR table, this probability is bounded by $\varepsilon_c \leq \frac{(q+\ell+1)^2}{2} 2^{-r}$. So, $\delta_{1,0} = \Pr[\mathcal{A} \text{ wins } \mathsf{game}_0] - \Pr[\mathcal{A} \text{ wins } \mathsf{game}_1] \leq \varepsilon_c$.
- Let game_2 be game_1 where the R oracle was replaced by the simR simulator.

  Let $\mathcal{A}'$ simulate $\mathcal{A}$ and simR in which picking a random $\bar{m}$, computing $g \leftarrow G(\bar{m})$, and inserting $(x, g, \bar{m})$ in the table is replaced by getting a random g^* from a source Σ and storing (x, g^*) in the table. We consider the two following sources: Σ_0 picks g^* with uniform distribution and Σ_1 picks $\bar{m}$ and output $g^* \leftarrow G(\bar{m})$. Note that using Σ_0 perfectly simulates game_1 while using Σ_1 perfectly simulates game_2. At the end, $\mathcal{A}'$ checks whether the EF-CMA game succeeded. Clearly, this is a distinguisher of some complexity $T + \mu_G$ between Σ_0 and Σ_1 by using $q + \ell + 1$ samples. So, $\delta_{2,1} = |\Pr[\mathcal{A} \text{ wins } \mathsf{game}_1] - \Pr[\mathcal{A} \text{ wins } \mathsf{game}_2]| \leq \varepsilon_d$.
- Let game_3 be the simulated EF-CMA game of Figure 13.7. Since the simulation $\mathsf{simSign}$ of the signing oracle is perfect, we have $\Pr[\mathcal{A} \text{ wins } \mathsf{game}_3] = \Pr[\mathcal{A} \text{ wins } \mathsf{game}_2]$ and thus $\delta_{3,2} = 0$.
- Let E_4 be the event that the final $\bar{m}$ was not queried to $\mathcal{C}$. Let game_4 be the game_3 in which E_4 occurred. In that case, $\mathcal{A}$ can be perfectly reduced to an EF-CMA adversary of complexity $T + \mu_s$ against $\mathcal{C}$. So, $\Pr[\mathcal{A} \text{ wins } \mathsf{game}_4] \leq \varepsilon_S$.

  Clearly, if E_4 did not occur, $\bar{m}$ was previously queried to $\mathcal{C}$. Let $\bar{m} = \bar{m}_j$, i.e., $\bar{m}$ was queried by $\mathcal{B}$ to $\mathcal{C}$ at the j^{th} sign-query. Thus, $\mathcal{B}$ queried simR with an input x_j and obtained $(x_j, G(\bar{m}_j), \bar{m}_j)$. Since there were no collision on simR, $\bar{m} = \bar{m}_j$ implies that $\widehat{x} = x_j$ thus $\widehat{\kappa} = \kappa_j$ and $\widehat{h} = h_j$. We have $H_{\widehat{\kappa}}(\widehat{m}) = H_{\widehat{\kappa}}(m_j)$. $\widehat{m}$ is different from all m_i since $\mathcal{A}$ won his attack against S'. Hence, $\mathcal{A}$ can be perfectly reduced to a TCR adversary against $\mathcal{D}_j$ and $\delta_{4,3} = \Pr[\mathcal{A} \text{ wins } \mathsf{game}_3] - \Pr[\mathcal{A} \text{ wins } \mathsf{game}_4] \leq \ell \varepsilon_H$.

Finally,

$$\begin{aligned}\Pr[\mathcal{A} \text{ wins } \mathsf{game}_0] &\leq \Pr[\mathcal{A} \text{ wins } \mathsf{game}_4] + \delta_{4,3} + \delta_{3,2} + \delta_{2,1} + \delta_{1,0} \\ &\leq \varepsilon_S + \ell\varepsilon_H + 0 + \varepsilon_d + \varepsilon_c\end{aligned}$$

We conclude by considering the above reductions that μ_H and μ_S are within the order of magnitude of the simulation cost which is polynomial.

■

There still remains some problems related to this last construction:

1. we do not have a full reduction to the weak security of S;
2. the signature enlarges;
3. κ must be signed;
4. we still need a random oracle R (implicitly meaning collision-resistant hashing) so the role of R is to concentrate on unpredictability and nevertheless, R is now restricted to $\{0,1\}^{k+m}$.

13.1.5 Improved Randomized Hash-and-Sign Paradigm

Halevi and Krawczyk [HK06a] also use a randomized hashing but avoid signing the κ salt. Indeed, they use an eTCR hash function as depicted in Figure 13.8. In [HK06a], they

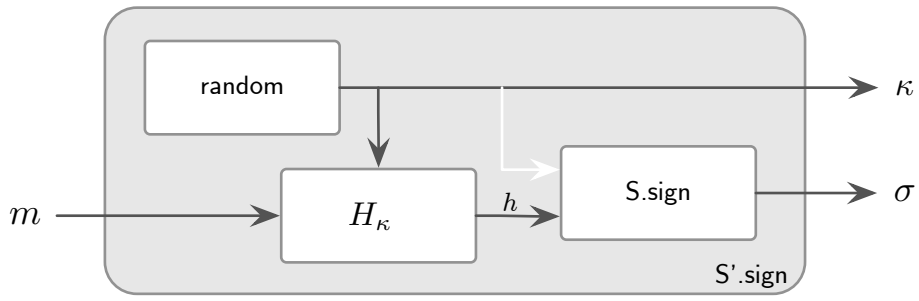


Figure 13.8. *The Randomized Hash-and-Sign Paradigm (with an eTCR Function H).*

introduced the concept of eTCR hashing and proposed a construction technique for eTCR functions. The technique is based on weak hashing. As application, they suggested to use an eTCR function as pre-processing for signature schemes. The signature consists of the pair (κ, σ) where σ is $\mathsf{S.sign}(K_s, H_\kappa(m))$. One problem is that they do no provide any proof of security for the signature. Indeed, they only focus on the problem for constructing an eTCR hash function based on weak hashing.

13.1.6 Analysis of the Above Existing Solutions

Figure 13.9 summarizes the results of the above theorems.

Function	Domain extension	Security proof	$\|\sigma'\|_2 = \|\sigma\|_2$	Avoid signing κ	ROM
RO*	☹	☺(weak to strong)	☺	n/a	☹
CRHF	☺	☺(strong to strong)	☺	n/a	☺
RO	☺	☺(weak to strong)	☺	n/a	☹
TCR	☺	$\approx$ (no reduction to S)	☹	☹	☹
eTCR	☺	☹	☹	☺	n/a

Figure 13.9. *Summary of the Hash-and-Sign Paradigm Variants.*

In a nutshell,

- using a CRHF, signatures do not enlarge and no RO is needed, however there is no security benefit and it is difficult to find strongly-secure plain signatures,
- using a random oracle, there is a security benefit, however there is the need of a random oracle while in practice hash functions deviate more and more from this idealization,
- using a TCR function, there is no reduction to the security of the signature S, there is still a need for a random oracle, and in addition the signature enlarges,
- finally, using an eTCR hash function, there is no formal security proof and the signature still enlarges.

In the rest of this chapter, we will propose a solution solving many problems and we concentrate on the four following goals:

1. The actual implementations are (almost all) proven in the random oracle model. In practice, the random oracle is just replaced by a hash function. While hash functions deviate more and more from the ideal random oracle, the security of the signature implementations which use the hash-and-sign paradigm becomes less and less secure. Our first goal is to *find a provably secure solution* to improve the security of the signature schemes which use the hash-and-sign paradigm.
2. We note that replacing the actual implementations is quite infeasible in a short time. So, we want to find a pre-processing able to *reuse the actual implementations*.
3. Before looking for a solution, first, we should *find a "better" model* which will fit the behavior of the (weak) hash functions. As said before, we emphasize that the random oracle model is too strong.

4. Finally, if it is possible we would like *to avoid that the signature enlarges.*

So, we start with Section 13.2 by defining a weaker model which seems better adapted to hash functions than the random oracle model. Then, in Section 13.3, we give a provably secure pre-processing solving goals 1, 2, and 3. Finally, in Section 13.4, we show how to avoid the increase in signature length in order to solve our last goal.

13.2 Modeling (Weak) Hash Functions

One crucial task is to find a model which fits to the current security of hash functions. Today hash functions are often modeled by random oracles (see Section 3.3) while they differ more and more from this idealization. A solution is to use the Liskov [Lis07] idea. It consists of a random oracle that is provided together with another oracle that "breaks" one (or many) hash function property(ies), e.g., a first preimage oracle. We apply the preimage-tractable random oracle model (PT-ROM) to model weak hashing in digital signatures.

13.2.1 Weak Random Oracle Hashing

Preimage-Tractable Random Oracles were introduced by Liskov [Lis07]. It is used to idealize some weak hash functions for which preimages are computable, i.e., the one-wayness is not guaranteed. It consists of two oracles:

- the first oracle WR can be used to compute images as a random oracle, i.e., $r = \mathsf{WR}(m)$,
- the second oracle $\mathsf{preimgWR}$ can be used to find a preimage of a hashed value. When $\mathsf{preimgWR}$ is queried with input r, it picks uniformly at random an element within the set of all its preimages, i.e., it outputs $m \in_{\mathsf{u}} \{\mathsf{WR}^{-1}(r)\}$.

The simulation of WR is done as for a (standard) random oracle, i.e., managing a table $\mathcal{T}$. To simulate $\mathsf{preimgWR}$, upon a new query r we first compute the probability q to answer an m that is not new, i.e.,

$$q = \Pr\left[(\mathsf{WR}^{-1}(r), r) \in \mathcal{T} | \mathbb{A}_{(m',r')\in\mathcal{T}} \mathsf{WR}(m') = r'\right] \quad .$$

Then, we flip a biased coin b with $\Pr[b = 0] = q$. If $b = 0$, then we pick uniformly one pair (m, r) in $\mathcal{T}$, otherwise we pick uniformly one m such that $(m, r) \notin \mathcal{T}$ and insert (m, r) in $\mathcal{T}$. Finally, we answer by m. Note that this oracle can be used to find collisions as well.

From a theoretical viewpoint, the preimage-tractable random oracle model (PT-ROM) is as powerful as the random oracle model (ROM) since $\mathsf{preimgWR}(0\|\alpha) \oplus \mathsf{preimgWR}(1\|\alpha)$ is indifferentiable from a random oracle even when $(\mathsf{WR}, \mathsf{preimgWR})$ is a preimage-tractable

random oracle. Our motivation is to model weak hash functions which are in place without changing the algorithm implementations.

13.3 Strong Signature Schemes with Weak Hashing

Today, most of the implementations use the hash-and-sign paradigm as depicted in Figure 13.10. The problem is that the hashing became weak. As said before, we only want to add a pre-processing and we will consider this type of construction as one soldered block in the future. In Figure 13.10, the soldered block is modeled by a *double line*.

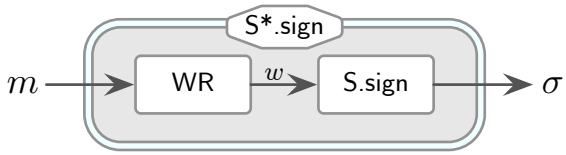


Figure 13.10. *The (Weak) Hash-and-Sign Implementations* S^* *(double boxed).*

More formally, we consider a deterministic (weak) hash-and-sign signature S^* defined by

$$\mathsf{S}^*.\mathsf{sign}(K_s, m) = \mathsf{S}.\mathsf{sign}(K_s, \mathsf{WR}(m)) \ ,$$

where $\mathsf{WR} : \{0,1\}^n \to \{0,1\}^r$ is a (weak) hash function and S is an weakly-secure FML-DS on domain $\{0,1\}^r$.

We put S^* together with the Halevi and Krawczyk [HK06a] message processing. Namely, given a weakly-secure FML-DS S we construct a strongly-secure AML-DS S' as depicted in Figure 13.11.

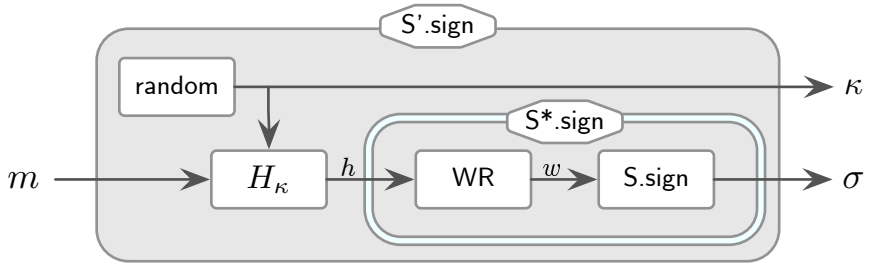


Figure 13.11. *The Secure Construction based on* S^*.

Formally we have:

$$\mathsf{S}'.\mathsf{sign}(K_s, m) = (\kappa, \mathsf{S}^*.\mathsf{sign}(K_s, H_\kappa(m))) \ ,$$

where $H : \{0,1\}^k \times \{0,1\}^* \text{ to } \{0,1\}^n$ is an eTCR hash function family. More precisely, the sign and verify algorithms work as follows:

$\sigma' \leftarrow \mathsf{S}'.\mathsf{sign}(K_s, m)$:
- pick $\kappa \in_{\mathsf{u}} \{0,1\}^k$
- $h \leftarrow H_\kappa(m)$
- $w \leftarrow \mathsf{WR}(h)$
- $\sigma \leftarrow \mathsf{S}.\mathsf{sign}(K_s, w)$
- $\sigma' \leftarrow (\kappa \| \sigma)$

$b \leftarrow \mathsf{S}'.\mathsf{verify}(K_p, m, \sigma')$: $\quad (\sigma' = \kappa \| \sigma)$
- $h \leftarrow H_\kappa(m)$
- $w \leftarrow \mathsf{WR}(h)$
- $b \leftarrow \mathsf{S}.\mathsf{verify}(K_p, w, \sigma)$

Clearly, our construction can be seen as a regular AML-DS based on a (weak) hash-and-sign with an extra randomized pre-processing $H_\kappa(\cdot)$.

Theorem 13.5 (Randomized Hash-and-Sign Paradigm with an eTCR Function). *Consider H is an OW-eTCR hash function family, and* WR *is a preimage-tractable random oracle. If* S *is an UF-KMA-secure FML-DS, then* S′ *in the above AML-DS construction is EF-CMA-secure.*

Clearly, we can build strong signature schemes for arbitrary messages based on any weak signature scheme restricted to fixed-length input messages *without* collision-resistance and *without* a full random oracle. The remaining drawback is that the signature enlarges.

Note that the OW assumption on H is necessary since WR is assumed to be preimage-tractable (otherwise, existential forgeries on S would translate in existential forgeries on S′). and eTCR hash functions may be not OW. Indeed, if H is eTCR, then H' defined by

$$H'_\kappa(m) = \begin{cases} 0\|m & \text{if } \kappa = 0\ldots0 \text{ and } |m| = n-1, \\ 1\|H_\kappa(m) & \text{otherwise.} \end{cases}$$

is eTCR as well but not OW. However, when there exists a set of messages $\mathbb{M}$ such that H is a PRG when restricted to $\{0,1\}^{k\times|\mathbb{M}|}$, then eTCR implies OW-eTCR.

Proof.
Let us assume that S is $(T+\mu, \ell, \varepsilon_S)$-UF-KMA-secure, H is $(T+\mu, \varepsilon_H)$-eTCR and $(T+\mu, \varepsilon_w)$-OW, and WR is a random oracle limited to $q < \ell$ queries where μ is some polynomially bounded complexity (namely, the overhead of some simulations). We will show that S′ is $(T, \ell-q, \varepsilon_f + q_p \cdot \varepsilon_w + (\ell-q)\cdot\varepsilon_H + q\cdot\varepsilon_S)$-EF-CMA-secure where ε_f represents a probability of failure during the reduction.

We start by considering an EF-CMA adversary $\mathcal{A}$ against our constructed scheme S′. We assume that $\mathcal{A}$ is bounded by a complexity T. By using an algorithm $\mathcal{B}$, we transform $\mathcal{A}$ either into an UF-KMA adversary against S or either into an eTCR adversary against H

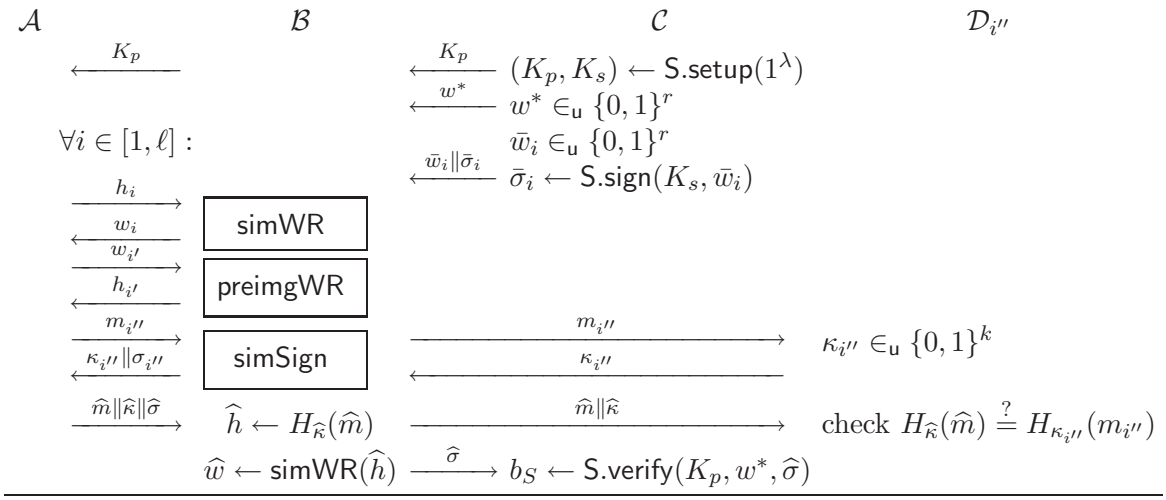


Figure 13.12. *Reduction to the UF-KMA or eTCR Games (from EF-CMA).*

as depicted in Figure 13.12. Here, $\mathcal{C}$ plays the role of the challenger in the UF-KMA game of Figure 11.1 while each $\mathcal{D}_{i''}$ plays the role of the i''^{th} challenger in the eTCR game of Figure 3.4.

Clearly, algorithm $\mathcal{B}$ has to simulates for $\mathcal{A}$ the signing oracle and the two oracles that model the preimage-tractable hash function that we refer by $\mathsf{simSign}$, simWR, and $\mathsf{preimgWR}$ respectively. To simulate WR and $\mathsf{preimgWR}$, we use another existing preimage-tractable random oracle WR_0 and $\mathsf{preimgWR}_0$ and we construct a random permutation φ such that $\mathsf{WR} = \varphi \circ \mathsf{WR}_0$. We consider a growing pool of values of h. The pool is initially empty. A new h is put in the pool if it is queried to simWR or returned by $\mathsf{preimgWR}$. Without loss of generality, we assume that $\mathcal{A}$ makes no trivial queries to simWR. Namely, he does not query simWR with an h already in the pool. Similarly, we assume that if $\widehat{h} = H_{\widehat{\kappa}}(\widehat{m})$ is not in the pool, $\mathcal{A}$ queries $\mathsf{simWR}(\widehat{h})$ before releasing $\widehat{m} \| \widehat{\kappa} \| \widehat{\sigma}$ to make sure that $\widehat{h}$ is in the pool. (So we may have $q+1$ queries to simWR.) The simulations work as follows:

simWR: At the beginning of the game, $\mathcal{B}$ picks a random $t \in_{\mathsf{u}} \{1..q\}$. When $\mathcal{A}$ submits a WR-query with input h:

- if $\varphi(\mathsf{WR}_0(h))$ is undefined, it answers the next $w = \bar{w}_i$ in the sequence, except that for the t^{th} query it answers $w = w^*$. Hence, there is a new entry $\varphi(\mathsf{WR}_0(h)) = w$ in the φ table.
- If $\varphi(\mathsf{WR}_0(h))$ is already defined, $\mathcal{B}$ aborts.

$\mathsf{preimgWR}$: When $\mathcal{A}$ submits a $\mathsf{preimgWR}$ query with input w, if $x = \varphi^{-1}(w)$ is not defined, it picks a random x on which $\varphi(x)$ is not defined and define $\varphi(x) = w$. Then, it queries $h \leftarrow \mathsf{preimgWR}_0(x)$ and answers h.

simSign: When $\mathcal{A}$ submits a sign-query with input m, $\mathcal{B}$ queries a new $\mathcal{D}_{i''}$ with input m, gets κ, and computes $h = H_\kappa(m)$. If h is in the pool, $\mathcal{B}$ abort. Otherwise, $\mathcal{B}$ runs $w \leftarrow \mathsf{simWR}(h)$ without counting this query (that is, use the next $\bar{w}_i$ in the sequence and not w^*). Thus, $\mathsf{simWR}(h)$ is equal to one of the $\bar{w}_i$ and $\mathcal{B}$ uses the corresponding signature $\bar{\sigma}_i$ to answer $\kappa \| \bar{\sigma}_i$.

Note that $\mathcal{B}$ has ℓ signed samples from $\mathcal{C}$, thus $\mathcal{A}$ is limited to ℓ queries to simWR and simSign. So, $q + q_s \leq \ell$. At the end, if $\mathcal{A}$ succeeds his EF-CMA game, he will send a tuple $(\widehat{m}, \widehat{\kappa}, \widehat{\sigma})$ to $\mathcal{B}$. We use the proof methodology of Shoup [Sho04]:

- Let game_0 be the EF-CMA game of Figure 11.2 against S'.
- Let game_1 be the simulated EF-CMA game against S' depicted in Figure 13.12.

  Clearly, the simulations fails when a $\varphi(\mathsf{WR}_0(h))$ is already defined while querying simWR with h or when $h = H_\kappa(m)$ was already in the pool while querying simSign. Let ε_f the bound on this failure probability. By using the difference lemma [Sho04] we obtain $\delta_{1,0} = \Pr[\mathcal{A} \text{ wins } \mathsf{game}_0] - \Pr[\mathcal{A} \text{ wins } \mathsf{game}_1] \leq \varepsilon_f$.

  Note that $\varepsilon_f \leq \Pr[\mathcal{B} \text{ fails on a simWR query}] + \Pr[\mathcal{B} \text{ fails on a simSign query}]$. We consider $\mathcal{A}$ is bounded by q, q_p and q_s queries to simWR, preimgWR, and simSign respectively, and a space of 2^r elements.
  First, we compute the probability that $\mathcal{B}$ fails on a simWR query, i.e., there were a collision of $\mathsf{WR}_0(h)$ for one h queried to simWR with one $\mathsf{WR}_0(h')$ for h' in the pool. By considering the queries from $\mathcal{A}$ and from simSign, there are at most $q + q_s + 1$ queries to simWR and at most $q + q_s + q_p + 1$ elements still defined in the pool. Since they are uniformly distributed, the probability that two elements collide is 2^{-r}. So, $\Pr[\mathcal{B} \text{ fails on a simWR query}] \leq (q + q_s + 1)(q + q_s + q_p + 1) \cdot 2^{-r}$.
  Now, we compute the probability that $\mathcal{B}$ fails on a simSign query, i.e., h was already in the pool. There are at most q_s queries to simSign and at most $q + q_s + q_p + 1$ elements h in the pool. For each query-h pair, we have the following scenario: $\mathcal{A}$ queries simSign with m, $\mathcal{B}$ queries $\mathcal{D}$ with m, gets κ, computes $H_\kappa(m)$, and looks if it is h. Clearly, this scenario can be described as game (a) of Figure 13.13. Let p be the maximal success probability among all random coins of the adversary $\mathcal{A}$ in the game (a).

  Now, consider game (b) depicted in Figure 13.13. Clearly, this game is harder than the eTCR game since $\mathcal{A}'$ has no control on the second message returned to $\mathcal{C}$, i.e., it is fixed to m_0. We know that ε_H is a bound on the success probability of $\mathcal{A}'$ in the eTCR game. Thus, we can deduce that $p \leq \sqrt{\varepsilon_H + 2^{-k}}$ from:

$$\begin{aligned} \varepsilon_H &\geq \Pr[H_\kappa(m_0) = H_{\kappa'}(m_0) \text{ and } \kappa' \neq \kappa] \\ &\geq \Pr[H_\kappa(m_0) = H_{\kappa'}(m_0)] - \Pr[\kappa' = \kappa] \\ &= p^2 - 2^{-k}. \end{aligned}$$

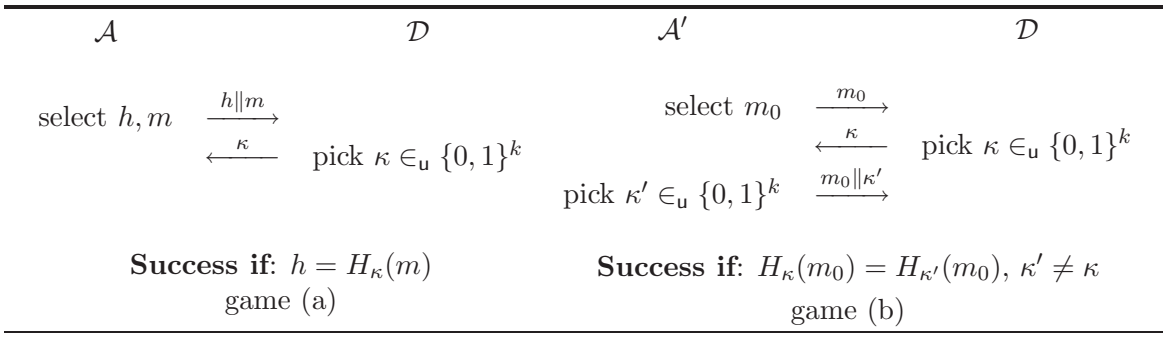


Figure 13.13. *Reduction to the eTCR Game*

We conclude that $\varepsilon_f \leq (q+q_s+1)(q+q_s+q_p+1)\cdot 2^{-r} + q_s(q+q_s+q_p+1)\cdot\sqrt{\varepsilon_H + 2^{-k}}$ and thus ε_f is negligible.

- Let E_2 be the event that the forgery $\widehat{m}\|\widehat{\kappa}\|\widehat{\sigma}$ is such that $\widehat{h} \leftarrow H_{\widehat{\kappa}}(\widehat{m})$ was queried to simWR. Let game_2 be game_1 in which E_2 occurred.

  Since we made sure that $\widehat{h}$ is in the pool, if E_2 does not occur, the $\widehat{h}$ was returned by some $\mathsf{preimgWR}(w)$ for the first time once. Note that when $\mathsf{preimgWR}$ returns an unused value, it is uniformly distributed among all unused values. Clearly, $\mathcal{A}$ has to find a pair $(\widehat{m}, \widehat{\kappa})$ with $H_{\widehat{\kappa}}(\widehat{m}) = \widehat{h}$ which breaks the one-wayness of H. So, $\delta_{2,1} = \Pr[\mathcal{A} \text{ wins } \mathsf{game}_1] - \Pr[\mathcal{A} \text{ wins } \mathsf{game}_2] \leq q_p \cdot \varepsilon_w$.

- Let E_3 be the event that $\widehat{h}$ is different from all $h_{i''} \leftarrow H_{\kappa_{i''}}(m_{i''})$. Let game_3 be game_2 in which E_3 occurred.

  Clearly, if E_3 did not occur, $\widehat{h}$ is equal to $h_{i''}$ for a certain i''. Recall that since $\mathcal{A}$ won his game $\widehat{m}$ is different from all $m_{i''}$. So, $\mathcal{A}$ found $\widehat{m}$ and $\widehat{\kappa}$ such that $H_{\widehat{\kappa}}(\widehat{m}) = H_{\kappa_{i''}}(m_{i''})$. Here, $\mathcal{A}$ can perfectly be reduced to an eTCR adversary against all $\mathcal{D}_{i''}$. So, $\delta_{3,2} = \Pr[\mathcal{A} \text{ wins } \mathsf{game}_2] - \Pr[\mathcal{A} \text{ wins } \mathsf{game}_3] \leq q_s \cdot \varepsilon_H \leq (\ell - q)\cdot\varepsilon_H$.

- Let E_4 be the event that $\widehat{w} = w^*$. In other words the forged value $\widehat{w}$ is equal to the expected value w^*. Let game_4 be game_3 in which E_4 occurred. Here, $\mathcal{A}$ can perfectly be reduced to an UF-KMA adversary against S. Clearly, $\Pr[\mathcal{A} \text{ wins } \mathsf{game}_4] \leq \varepsilon_S$.

  Finally $\Pr[\mathcal{A} \text{ wins } \mathsf{game}_3] \leq q \cdot \varepsilon_S$ since E_4 occurred with probability $1/q$ and so $\Pr[\mathcal{A} \text{ wins } \mathsf{game}_4] / \Pr[\mathcal{A} \text{ wins } \mathsf{game}_3] = 1/q$.

Finally,

$$\begin{aligned}\Pr[\mathcal{A} \text{ wins } \mathsf{game}_0] &\leq \Pr[\mathcal{A} \text{ wins } \mathsf{game}_3] + \delta_{3,2} + \delta_{2,1} + \delta_{1,0} \\ &\leq q\cdot\varepsilon_S + (\ell - q)\cdot\varepsilon_H + q_p\cdot\varepsilon_w + \varepsilon_f\end{aligned}$$

■

13.4 The Entropy Recycling Technique

To keep the same signature length, we have to avoid to append κ in the signature. The idea from [Mir06] is to use the randomness computed in the signature scheme instead of introducing a new random parameter. Mironov [Mir06] present specific modifications for the DSA [DSS94, DSS00], RSA-PSS [BR96], and Cramer-Shoup [CS00] signature schemes. In this section, we generalize the construction from Mironov. For that, we introduce a special sort of signature schemes: Signature with Randomized Precomputation.

Definition 13.6 (Signature with Randomized Precomputation).
Any Signature with Randomized Precomputation *(SRP) consists of five algorithms:* setup, presign, postsign, extract, *and* verify.

The setup *algorithm works as in a standard signature scheme, see Section 11.2.*

The signature algorithm is separated in two parts:

- *first, a* probabilistic *precomputation algorithm, called* presign*, generates the randomness without using the message to be signed,*
- *then, a signature algorithm, called* postsign*, signs the message using the previous randomness.*

The randomness must be recoverable from the signature itself, which requires the extract *algorithm.*

Finally, there must exist a verify *algorithm as in any classical signature scheme, see Section 11.2.*

Formally, the five algorithms of any SRP scheme work as follows:

$$\begin{array}{ll} (K_p, K_s) \leftarrow \mathsf{setup}(1^\lambda) & \\ (\xi, r) \leftarrow \mathsf{presign}(K_s) & r \leftarrow \mathsf{extract}(K_p, \sigma) \\ \sigma \leftarrow \mathsf{postsign}(K_s, m, \xi) & b \leftarrow \mathsf{verify}(K_p, m, \sigma) \end{array}$$

Actually, all digital signature schemes can be written this way (e.g., with r void), but we need r to have a large enough entropy. We provide the necessary quantitative definitions for that in Section 3.6. When talking about the entropy of a SRP scheme, we implicitly mean the entropy of r generated by $\mathsf{presign}(K_s)$ given a private key K_s.

Standard implementations may be represented by an SRP with pre-hashed message as depicted in Figure 13.14. Note that in the SRP representation of Figure 13.14 we conserved the weak hashing WR but in reality, this pre-hashing could be put in the postsign algorithm.

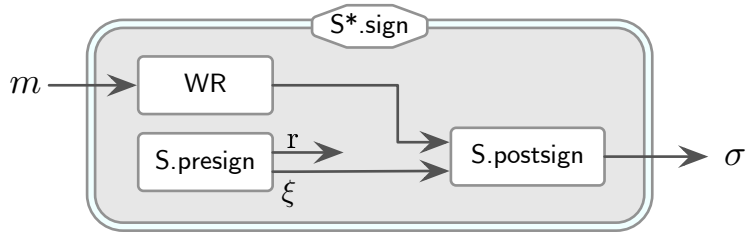


Figure 13.14. *The SRP Implementations.*

Theorem 13.7 (Randomized Hash-and-Sign Paradigm Recycling the Entropy). *Consider* $H : \{0,1\}^k \times \{0,1\}^* \mapsto \{0,1\}^n$ *is an eTCR hash function with* k*-bit keys and* S *is a FML-SRP. We assume that the signature construction* S^* *based on* S *defined by*

$\sigma' \leftarrow \mathsf{S}^*.\mathsf{sign}(K_s, m)$:
- $(\xi, r) \leftarrow \mathsf{S.presign}(K_s)$
- *pick* $\kappa \in_{\mathsf{u}} \{0,1\}^k$
- $h \leftarrow H_\kappa(m)$
- $\sigma \leftarrow \mathsf{S.postsign}(K_s, h, \xi)$
- $\sigma' \leftarrow (\kappa \| \sigma)$

$b \leftarrow \mathsf{S}^*.\mathsf{verify}(K_p, m, \kappa\|\sigma)$: $\quad(\sigma' = \kappa\|\sigma)$
- $h \leftarrow H_\kappa(m)$
- $b \leftarrow \mathsf{S.verify}\,(K_p, h, \sigma)$

is an EF-CMA secure AML-SRP requiring an additional randomness κ*. We assume that the SRP produces* k*-bit strings that are indistinguishable from uniformly distributed ones.*

Consider G *is a random oracle with* k*-bit output strings limited to* q *queries. The signature construction* S' *defined by*

$\sigma' \leftarrow \mathsf{S}'.\mathsf{sign}(K_s, m)$:
- $(\xi, r) \leftarrow \mathsf{S.presign}(K_s)$
- $\kappa \leftarrow \mathsf{G}(r)$
- $h \leftarrow H_\kappa(m)$
- $\sigma \leftarrow \mathsf{S.postsign}(K_s, h, \xi)$
- $\sigma' \leftarrow \sigma$

$b \leftarrow \mathsf{S}'.\mathsf{verify}(K_p, m, \sigma')$: $\quad(\sigma' = \sigma)$
- $r \leftarrow \mathsf{S.extract}(K_p, \sigma)$
- $\kappa \leftarrow \mathsf{G}(r)$
- $h \leftarrow H_\kappa(m)$
- $b \leftarrow \mathsf{S.verify}\,(K_p, h, \sigma)$

is also EF-CMA-secure even by re-using the randomness from the SRP.

In a nutshell if the construction of Figure 13.15 is secure, then the construction of Figure 13.16 without additional random coins is also secure. Note that in both figures, the SRP implementation could contain the hashing but here it is hidden in the postsign algorithm.

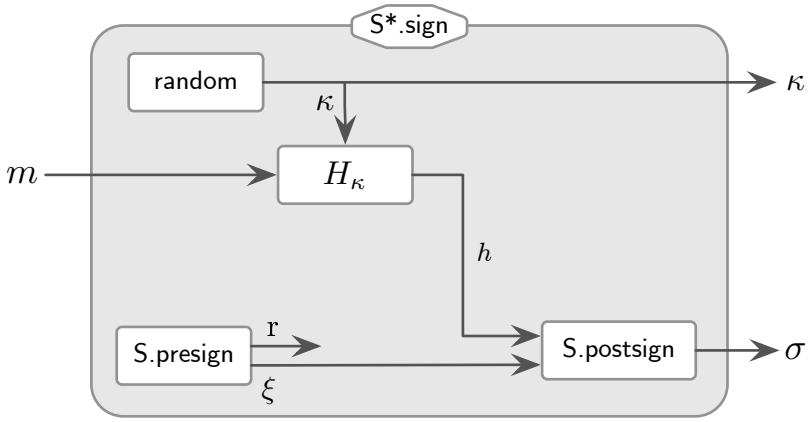


Figure 13.15. *SRP with Additional Random Coins.*

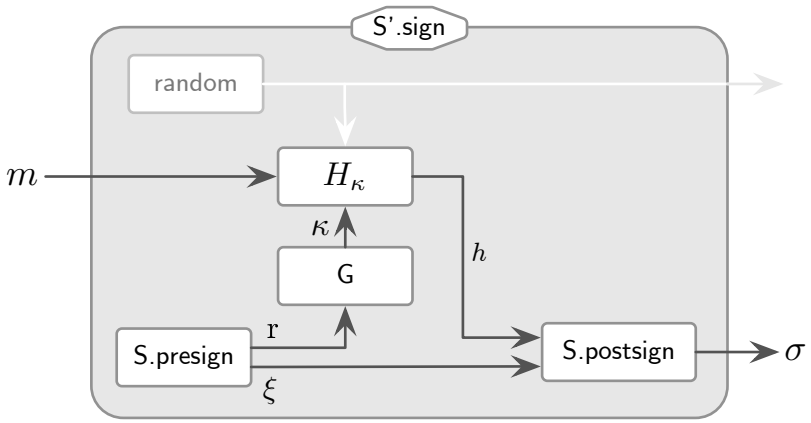
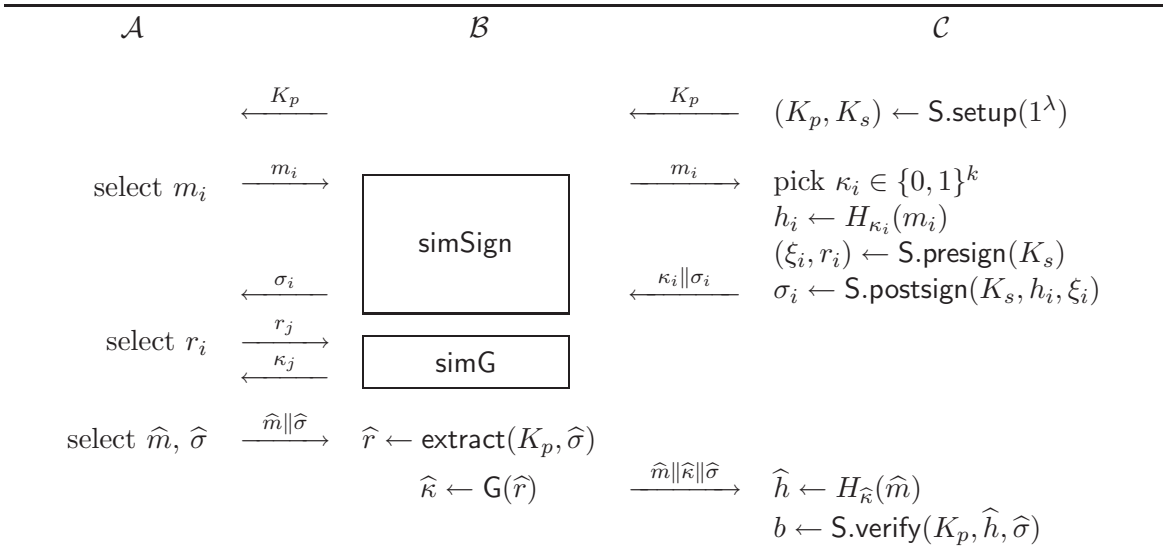


Figure 13.16. *SRP without Additional Random Coins.*

Proof.
Assume that the AML-SRP construction S^* is $(T+\mu,\ell,\varepsilon_S)$-EF-CMA secure and that r is $(T+\mu,\ell,\varepsilon_d)$-PR where μ is some polynomially bounded complexity due to the game reduction. In the following, we prove that the construction S' is $(T,\ell,\varepsilon_S+\varepsilon_c)$-EF-CMA secure where ε_c represents the probability of collision on the G outputs as defined in Lemma 3.26. We consider any EF-CMA adversary $\mathcal{A}$ against S'. As depicted in Figure 13.17, we transform $\mathcal{A}$ into an EF-CMA adversary against the (eTCR-based) scheme S^* by using an algorithm $\mathcal{B}$. $\mathcal{B}$ simulates the random oracle G, the transform of $\mathsf{S}'.\mathsf{sign}$ to $\mathsf{S}.\mathsf{sign}$, and replaces the final $(\widehat{m},\widehat{\sigma})$ by $(\widehat{m},\widehat{\kappa},\widehat{\sigma})$.



Winning condition: $b = 1$ **and** $\widehat{m} \notin \{m_1, \ldots m_\ell\}$**.**

Figure 13.17. *Reduction to the EF-CMA Game Against the eTCR-based Scheme S^*.*

The simulations works as follows:

simG. This simulation works as defined in Section 3.3.

simSign. When $\mathcal{A}$ submits a sign-query with input m, $\mathcal{B}$ obtains (κ,σ) by querying $\mathcal{C}$ with m and deduces $r \leftarrow \mathsf{extract}(K_p,\sigma)$. If r is free in the simG table, then it lets $\kappa = \mathsf{G}(r)$ and returns σ to $\mathcal{A}$, otherwise $\mathcal{B}$ fails.

$\mathcal{B}$ is allowed to ℓ queries to the $\mathsf{S}.\mathsf{sign}$ oracle, so $\mathcal{A}$ is also allowed to ℓ queries to simSign. At the end, if $\mathcal{A}$ succeeds his EF-CMA game, he will send a tuple $(\widehat{m},\widehat{\sigma})$ to $\mathcal{B}$. $\mathcal{B}$ should then deliver a tuple $(\widehat{m},\widehat{\kappa},\widehat{\sigma})$ to $\mathcal{C}$ and for that he computes $\widehat{\kappa} \leftarrow \widehat{r}$ where $\widehat{r}$ is extracted from $\widehat{\sigma}$. We use one more time the proof methodology of Shoup [Sho04]:

- Let game_0 be the EF-CMA game against S' as depicted in Figure 11.2.

- Let game_1 be the simulated EF-CMA game against S' depicted in Figure 13.12.

  Clearly, the simulation fails if $\mathsf{simSign}$ fails, i.e., if an r_j in $\mathsf{simSign}$ is not free in the simG table. Let ε_c the bound on this probability of collision. A detailed expression of ε_c is given on Lemma 3.26. It is clearly negligible.

  Let E_1 the event that all r_j are free in the simG table. So, game_1 is game_0 in which E_1 occurred.

  By using the difference lemma [Sho04] we obtain

  $$\delta_{1,0} = |\Pr[\mathcal{A} \text{ wins } \mathsf{game}_0] - \Pr[\mathcal{A} \text{ wins } \mathsf{game}_1]| \leq \varepsilon_c \ .$$

  In game_1, $\mathcal{A}$ can perfectly be reduced to an EF-CMA adversary against S^*. So $\Pr[\mathcal{A} \text{ wins } \mathsf{game}_1] \leq \varepsilon_S$.

Finally, we obtain

$$\begin{aligned} \Pr[\mathcal{A} \text{ wins } \mathsf{game}_0] &\leq \Pr[\mathcal{A} \text{ wins } \mathsf{game}_1] + \delta_{1,0} \\ &\leq \varepsilon_S + \varepsilon_c \end{aligned}$$

■

13.5 Applications

Consider any signature implementation S^* based on a textbook signature scheme S and using the original hash-and-sign paradigm with a (weak) hash function WR, i.e.,

$$\mathsf{S}^*.\mathsf{sign}(K_s, m) = \mathsf{S}.\mathsf{sign}(K_s, \mathsf{WR}(m)) \ .$$

Assume that S is weakly secure and that some weakness on WR was reported.

By using Theorem 13.5, we can build a strongly secure implementation by adding a pre-processing $H_\kappa(m)$ where H is an OW-eTCR hash function. Our new construction S' defined by

$$\mathsf{S}'.\mathsf{sign}(K_s, m) = \mathsf{S}^*.\mathsf{sign}(K_s, H_\kappa(m)) = \mathsf{S}.\mathsf{sign}(K_s, \mathsf{WR}(H_\kappa(m)))$$

is strongly secure. Note that it is strongly secure even if the WR is weak. Thanks to our construction, the actual implementations can still be used, it simply needs to "pre-process" the input message m. This assumes that WR can be modeled as a preimage-tractable random oracle.

Note that in the S' construction the signature enlarges due to the additional random coins κ.

By using Theorem 13.7, if the signature scheme S is an SRP, then we can avoid increasing the signature length by reusing the random coins from the $\mathsf{presign}$ algorithm into the pre-processing $H_\kappa(\cdot)$.

13.5.1 A Concrete Example with DSA

We apply Theorem 13.5 and Theorem 13.7 to offer a quick fix to DSA in the case that SHA-1 [SHA95] became subject to preimage attacks. Here, standard implementations of DSA could still be used: only a "message pre-processing" would be added. First, note that DSA without hashing can be described using our SRP formalism of Section 13.6. We denote by m the messages of arbitrary length (input of the sign algorithm) and by h the digest in DSA, i.e., the 160-bit string. The public parameters are a 160-bit prime q, a 1024-bit prime $p = a \cdot q + 1$, and a generator $g \in \mathbb{Z}_p$ of order q.

The DSA construction is depicted in Figure 13.18 where $f(m)$ describes some function mapping the input message m of arbitrary length to a fixed length string h. $f(m)$ represents the "message pre-processing".

$(K_s, K_p) \leftarrow \mathsf{setup}(1^\lambda)$:	pick $K_s \in_{\mathsf{u}} \mathbb{Z}_q$ $K_p \leftarrow g^{K_s} \bmod p$
$\sigma \leftarrow \mathsf{sign}(K_s, m, k, r)$:	pick $k \in_{\mathsf{u}} \mathbb{Z}_q^*$ $r \leftarrow (g^k \bmod p) \bmod q$ $h \leftarrow f(m)$ $s \leftarrow \frac{h + K_s \cdot r}{k} \bmod p$ $\sigma \leftarrow (r, s)$
$b \leftarrow \mathsf{verify}(K_p, m, \sigma)$:	$h \leftarrow f(m)$ check $r = (g^{\frac{h}{s} \bmod q} y^{\frac{r}{s} \bmod q} \bmod p) \bmod q$

Figure 13.18. *The DSA Construction.*

DSA uses the (original) hash-and-sign paradigm. $f(m)$ is simply

$$h \leftarrow H^*(m)$$

where H^* is a collision resistant hash function.

Consider textbook DSA, i.e., DSA without f, is an UF-$\emptyset$MA-secure FML-DS. Note that it is existentially forgeable. Theorem 13.5 says that the scheme of Figure 13.18 where $f(m)$ is defined by $H_\kappa \circ \mathsf{WR}(m)$, i.e.,

$$h \leftarrow \mathsf{WR}(H_\kappa(m)) \qquad \text{where } \kappa \in_{\mathsf{u}} \{0,1\}^k \ ,$$

is EF-CMA-secure when WR is a preimage-tractable random oracle (say SHA-1 in practice) and H is a one-way eTCR hash function. Thus, we build an EF-CMA-secure AML-DS based

on DSA without collision-resistance. Assuming that $\mathsf{WR}(H_\kappa(m))$ can be instantiated by $\mathsf{SHA1}(\mathsf{RMX}(\kappa, m))$ where RMX denotes the implementation from Halevi-Krawczyk [HK06b] of the message randomization, the Halevi-Krawczyk construction is secure. The drawback is that the signature enlarges sending κ.

Instead of picking some new randomness κ we re-use the randomness from the $\mathsf{presign}$ algorithm if the implementation of DSA allows it, i.e., we use $\mathsf{G}(r)$ where G is a random oracle. Theorem 13.7 says that the scheme of Figure 13.18 where $f(m)$ is defined by $H_{\mathsf{G}(r)} \circ \mathsf{WR}(m)$, i.e.,

$$h \leftarrow \mathsf{WR}(H_{\mathsf{G}(r)}(m))$$

is EF-CMA-secure as well.

From Theorem 13.5 and Theorem 13.7, we deduce that our construction is (T, Q, ε'_s)-EF-CMA-secure where $\varepsilon'_s \leq \varepsilon_f + q_p \cdot \varepsilon_w + (\ell - q) \cdot \varepsilon_H + q \cdot \varepsilon_S + \varepsilon_c$. Assuming an adversary bounded by a time complexity T and an online complexity $Q \leq T$, considering that ε_H, ε_s and ε_w are all equals to $T \cdot 2^{-160}$, k is 160-bit long, q, q_s, and ℓ are bounded by Q, and q_p is bounded by T, we obtain $\varepsilon_f \leq 9 \cdot Q \cdot T \cdot 2^{-160}$, $\varepsilon_c \leq Q^2 \cdot 2^{-160}$ and so

$$\varepsilon'_s \leq \left(12 \cdot Q \cdot T + Q^2\right) \cdot 2^{-160}.$$

Clearly, $Q \cdot T$ must be bounded by 2^{160}. Since Q is often near 2^{30}, we deduce that T can be close to 2^{130} which is much better than actual implementations requiring a complexity T bounded by 2^{80} to avoid collision attacks.

In summary, by using Theorem 13.5 and Theorem 13.7, we build a DSA-based EF-CMA-secure scheme for input messages of arbitrary length and with signatures as long as the original DSA scheme.

Chapter

FOURTEEN

Conclusion

There are several ways to establish a secure communication channel over an insecure channel. In particular, this can be achieved by using only authenticated data (relaxing the confidential assumption). Data authentication may be done based on different assumptions. In this thesis, we first consider that an extra authenticated channel is available, see Part I, and then that a public-key infrastructure is in place (or may be setup with an extra authenticated channel), see Part II.

With this thesis, we give the following main contributions:

1. We propose a detailed and elaborated model for SAS-based message authentication protocols (Chapter 4).
2. We analyze the security of *generic* SAS-based message authentication protocols (Chapter 5). In particular, we give *generic* attacks and the definition of an *optimal* protocol.
3. We propose the first optimal non-interactive message authentication protocol, called PV-NIMAP (Section 7.3).
4. We propose the first optimal two-party SAS-based message mutual-authentication protocol, called PV-SAS-MMA (Section 8.3), and the first optimal two-party SAS-based message cross-authentication protocol, called PV-SAS-MCA (Section 8.4).
5. We propose the first optimal SAS-based group message authentication protocol, called LP-SAS-GMA (Section 9.3).

6. We propose a method to build authenticated key agreements for two-party or group settings based on a message authentication protocol (Section 10.4). From that, we propose two optimal SAS-based authenticated key agreement protocols, one specific to two-party settings, called PV-SAS-AKA (Section 10.5), and another more general for any group settings, called LP-SAS-GKA (Section 10.6). We also propose a method to keep long-term pairwise authentication keys in order to setup new (sub) groups with no additional user interaction.

7. We propose the ONTAP primitive to allow non-transferable signature verification (Chapter 12). The primitive is an offline non-transferable authentication protocol (ONTAP) and is especially designed to fit the authentication mechanism of e-passports. In a more general view, we propose a way to protect the privacy of the signer, the owner of the signature, or the signed data.

8. We propose a pre-processing strengthening for the actual implementations of hash-and-sign-based signature schemes (Chapter 13). We also propose a solution to avoid the increase in signature length.

14.1 SAS-based Cryptography

Security model. First of all, we gave a detailed security model for SAS-based protocols. This kind of protocols are special in the sense that they use two types of channels. In particular, adversaries have full control on all communications but can not create authenticated messages on behalf of another identity. The security model of Chapter 4 is a logical continuation of the ones given in [Vau05b, Pas05, PV06a, PV06b, LP08, LP09]. Note that all these models, as well as the one of Chapter 4, are based on the adversarial model from Bellare and Rogaway [BR93a].

Maximal achievable security. We analyzed the security of SAS-based message authentication protocols on a global perspective. In particular, we showed that there exists a generic one-shot attack with probability of success essentially $1/n$, where n is the size of the set of all possible SAS values. We also showed that there exists a generic attack which uses q_A instances of Alice and q_B instances of Bob with probability of success essentially $1 - e^{\frac{q_A q_B}{n}}$. We emphasize that *any* SAS-based protocol is vulnerable to these generic attacks and, consequently, no protocol can achieve a better security. We conclude that *any* SAS-based message authentication protocol in which the best attack succeeds essentially with the same probability of success than a generic attack is essentially "optimal".

Security in complex settings. We proved that any SAS-based protocol is provably secure in any computational context provided that the simple and natural restrictions rules $\mathcal{R}_1$–$\mathcal{R}_4$

are fulfilled. More precisely, we first proved that all presented protocols are secure in the stand-alone model and then showed that any SAS-based message authentication protocol that is secure in the stand-alone model remains secure in more complex settings.

Interactive versus non-interactive protocols. Concerning unilateral authentication, we saw the difference between interactive and non-interactive protocols. An (optimal) interactive protocol avoids offline attacks and consequently tolerates much shorter SAS than a non-interactive protocol, e.g., 20 instead of 100 bits. Therefore, an interactive protocol is more user friendly. However, we saw that protocol interactivity may be a problem for some applications. For instance, an interactive protocol is not well-suited for SSH since there is nobody to transfer the (random) SAS from the SSH server to the client user. In short, the choice between interactive and non-interactive protocols depends on the application.

Unilateral authentication. We saw that the original two-party SAS-based protocol from Vaudenay [Vau05b] is already optimal. Hence, there was no additional work to do on unilateral interactive SAS-based message authentication protocols.

However, we saw that the previous non-interactive protocols are not optimal. We proposed an optimal non-interactive message authentication protocol based on a commitment scheme: PV-NIMAP. Thanks to the unpredictability of the authenticated value, the SAS is much shorter. Indeed, considering one-shot attacks, only 80 bits (instead of 160) are required to achieve the same security level as the SSH key authentication. Considering more general attacks, the protocol only requires 100 bits. We emphasize that our protocol only requires second preimage resistance for the hash function.

Bilateral authentication. We saw that the message cross-authentication protocol given by Vaudenay [Vau05b] is not round-optimal and given with no formal security proof. We first proposed a new 3-move MMA protocol and then a new 3-move MCA protocol using a generic commitment scheme: PV-SAS-MCA. Both constructions are optimal and can use a SAS of 20 bits.

Group authentication. We saw that the only prior work is Group-MANA IV which is not round-optimal. We proposed an optimal SAS-based group message authentication protocol: LP-SAS-GMA. This protocol is generic and can be executed between any set of participants.

Key agreements. As explained before, setting up a secure communication between two or more parties requires authenticated communication channels. In this thesis, we presented a general methodology for protecting ordinary key agreement protocols against active attacks.

More precisely, we presented an efficient construction for authenticated key agreements based on existing key agreements and SAS-based message authentication protocols.

Thanks to the Diffie-Hellman protocol, our PV-SAS-MCA protocol can make a secure and efficient SAS-based authenticated key agreement protocol with three moves. This leads to PV-SAS-AKA. Using the Burmester-Desmedt protocol, our LP-SAS-GMA protocol allows us to build a secure and efficient SAS-based group key agreement. This leads to LP-SAS-GKA. Additionally, the clever use of long-term public keys provides an efficient way of managing dynamic groups.

We emphasize here that both final protocols have an optimal security with respect to the amount of authenticated data and have an optimal number of rounds. Such key agreement protocols have the advantage that they do not require any trusted third party, any public-key infrastructure, nor any pre-shared key. Peer-to-peer security is ensured by using an authentication primitive, e.g., voice recognition for voice over IP or string copy for devices. Therefore, consumers can establish and reconfigure security associations for electronic devices with minimal effort. In a certain sense, security can be provided as an add-on feature.

Overview. To emphasize differences between various protocols, we gathered the most important aspects into Figure 14.1. The contributions of the present thesis are written in black while prior or further works are written in gray. To distinguish between unkeyed and keyed hash functions, we use shorthands $\mathsf{H}(\cdot)$ and $\mathsf{H}_K(\cdot)$. Note that for all setups, i.e., unilateral, bilateral or group, interactive or non-interactive, message authentication and key agreement, there now exists an optimal provably secure SAS-based protocol. Hence, in the same model, only small optimizations or efficiency improvements may be done. For instance, our NIMAP was recently improved in the sense that the CRS model is no longer required but only an eTCR function.

There is a gap between theoretical constructions and practical instantiations. Most practical SAS-based key agreement protocols such as the Zfone protocol [ZJC00] and the wireless USB key agreement protocol [WUS06] use collision resistant hash functions to mimic the functionality of a commitment scheme. Although this approach cannot be used in general, it may be appropriate for securing key agreement protocols, since the corresponding authenticated messages have uniform distribution. More formally, a collision resistant hash function (CRHF) as a deterministic commitment cannot be hiding and non-malleable for an arbitrary message distribution. However, hiding and non-malleability with respect to uniform distribution makes sense also for hash functions. Consequently, it should be possible to give a formal security proof for these practical protocols. On the other hand, the corresponding security requirements are very different from the standard ones such as the one-way and the collision resistance properties. Hence, interpretation of corresponding security requirements

[1]The security proof is given in a different adversarial model.

Type	*Protocol*	*Interactive*	*Weak auth.*	*Optimal*	*Sec. proof*	*Primitives*
NIMAP	CRHF-based [BSSW02]		☺		☺	CR $\mathsf{H}(\cdot)$
	MANA I, II [GMN04]	≈			☺	TCR $\mathsf{H}_K(\cdot)$
	PV-NIMAP [PV06a]		☺	☺	☺	$\mathcal{C}om$, WCR $\mathsf{H}(\cdot)$
	HCR-based [MS07]		☺	☺	☺	HCR $\mathsf{H}_K(\cdot)$
	eTCR-based [RWSN07]		☺	☺	☺	eTCR $\mathsf{H}_K(\cdot)$
IMAP	Vau-SAS-IMAP [Vau05b]	✓	☺	☺	☺	$\mathcal{C}om$
	ICR-based [MS08]	✓	☺	?	☺[1]	ICR $\mathsf{H}_K(\cdot)$
MMA	MANA III [GMN04]	✓				$\mathsf{H}_K(\cdot)$
	PV-SAS-MMA [PV06b]	✓	☺	☺	☺	$\mathcal{C}om$
MCA	Vau-SAS-MCA [Vau05b]	✓	☺			$\mathcal{C}om$
	PV-SAS-MCA [PV06b]	✓	☺	☺	☺	$\mathcal{C}om$, $\mathsf{H}_K(\cdot)$
	MANA IV [LN06a]	✓	☺	☺	☺	$\mathcal{C}om$, $\mathsf{H}_K(\cdot)$
GMA	Group-MANA IV [VAN06]	✓	☺		☺	$\mathcal{C}om$, $\mathsf{H}_K(\cdot)$
	LP-SAS-GMA [LP08]	✓	☺	☺	☺	$\mathcal{C}om$, $\mathsf{H}_K(\cdot)$
AKA	PGPfone (Zimmermann 95)	✓	☺			$\mathcal{C}om$, $\mathsf{H}(\cdot)$
	Hoepman [Hoe04]	✓	☺			$\mathcal{C}om$, $\mathsf{H}(\cdot)$
	PV-SAS-AKA [PV06b]	✓	☺	☺	☺	$\mathcal{C}om$, $\mathsf{H}_K(\cdot)$
	LP-SAS-GKA [LP08]	✓	☺	☺	☺	$\mathcal{C}om$, $\mathsf{H}_K(\cdot)$

Figure 14.1. *Overview of SAS-based Protocols.*

is an interesting theoretical and practical problem.

14.2 Preserving the Privacy of Signed Documents

We saw that releasing a (classical) digital signature faces some privacy issues. Indeed, there are cases where the prover needs to authenticate some data without making it possible for any malicious verifier to transfer the proof to anyone else, like in e-passports. To solve this problem, we proposed that the owner of the valid signature proves its knowledge without revealing it. This proof should be non-transferable.

For that reason, we studied deniability in signature verification. Deniability is essentially a weaker form of non-transferability. It holds as soon as the protocol is finished (it is often called offline non-transferability). We introduced Offline Non-Transferable Authentication Protocol (ONTAP) and we showed that they can be built by using a classical signature scheme and a deniable zero-knowledge proof of knowledge for that signature.

Usually, Σ-protocols are used as proofs of knowledge. However, they are only honest-verifier zero-knowledge. Indeed, considering malicious verifier, a protocol transcript may give evidence of interaction (with the Fiat-Shamir technique). We proposed a generic way to transform any Σ-protocol into a deniable zero-knowledge proof of knowledge (in the standard, random oracle, and common reference string) models.

Finally, we gave examples to upgrade signature standards based on RSA or ElGamal into an ONTAP. Our examples are well-suited for implementation in e-passports. In particular, they are efficient and compatible with the standard signature schemes used in e-passports.

14.3 Strengthening Signature Schemes Based on the Hash-and-Sign Paradigm

Consider any signature implementation S^* based on a textbook signature scheme S and using the original hash-and-sign paradigm with a hash function WR, i.e., $\mathsf{S}^*.\mathsf{sign}(K_s, m) = \mathsf{S}.\mathsf{sign}(K_s, \mathsf{WR}(m))$. Assume that S is weakly secure and that some weaknesses on WR was reported. Clearly, WR is a weak hash function and S^* is insecure.

Usually hash functions are modeled by random oracles, but they deviate more and more from this model. We saw how to model weak hash function and we recalled the preimage-tractable random oracle model from Liskov [Lis07].

We showed that it is possible to transform the weak implementation S^* into a strongly secure implementation. This transformation only consists in adding a pre-processing $H_\kappa(m)$ where H is an OW-eTCR hash function and κ some random coins. Our new construction S'

is defined by $\mathsf{S}'.\mathsf{sign}(K_s, m) = \mathsf{S}^*.\mathsf{sign}(K_s, H_\kappa(m)) = \mathsf{S}.\mathsf{sign}(K_s, \mathsf{WR}(H_\kappa(m)))$. S' is strongly secure and actual implementations can still be used. Indeed, it simply needs to "pre-process" the input message. This assumes that WR can be modeled as a preimage-tractable random oracle.

Usually the random coins κ should be sent with the signature. In other words, the new signature enlarges. We presented a generic solution to avoid the increase in signature length. Indeed, we showed how to reuse the random coins from the signature scheme itself.

14.4 Final Notes and Further Work

In this thesis, we analyzed SAS-based message authentication protocols in different settings, two-party versus group, unilateral versus bilateral, interactive versus non-interactive, etc. Recall that a SAS is a short authenticated string and may be transmitted from one device to another by voice, near field communication (NFC), string copy, or string comparison. The goal was to optimize them in two ways: first, a protocol should use as little as possible authenticated data, second, it should use a minimal number of rounds. We tried to keep the computational complexity as light as possible. Today, there is at least one optimal protocol for each setting as shown in Figure 14.1. Some future work may be done in order to optimize the computational complexity. Another interesting continuity will be to avoid proofs in the random oracle or common reference string models. We note that some works already started in that directions but only for non-interactive unilateral message authentication protocols, see Figure 14.1. A SAS-based (unilateral) message authentication protocol may be used

- to authenticate a public-key, as in PGP, GPG, SSH, etc.,
- in case of public-key disaster, i.e., when a trusted infrastructure is compromised, as a temporary quick fix.

We stress that SAS-based message cross-authentication protocols are very useful, in particular for authenticated key agreements. We proposed to use our (two-party) PV-SAS-AKA or our (group) LP-SAS-GKA protocols in many applications:

- to establish a secure communication between small devices, e.g., Bluetooth pairing,
- to establish a secret key between a computer and a router (WEP or WPA), a standard called Wi-Fi Protected Setup (WPS) is in development,
- to establish a secret key between a computer and its devices such as a printer or a keyboard, there is a standard Wireless USB (WUSB) in development,
- to secure Voice over IP phone calls (or any phone application),

- to establish security associations in *ad-hoc* networks (MANETs).

One may want to avoid several executions of the SAS-based message authentication protocol because each run requires user interaction. A solution consists in authenticating long-term authentication keys once. Indeed, one can use a SAS-based message authentication protocol to exchange public keys, for instance signature verification keys. After this setup phase, parties are able to authenticate data as long as they want simply by using a signature scheme. As before, one may authenticate the messages of a key agreement protocol. We noticed that using a signature scheme may lead to some privacy issues as well as some forgeability issues.

Many existing schemes protect the privacy of the signed data, however there were no scheme adapted for the case of e-passports. In this thesis, we introduced a new primitive, called Online Non-Transferable Authentication Protocol (ONTAP), filling this gap. We proposed specific implementations for RSA and ElGamal-based schemes since both may be used in e-passports. We emphasize that the computational power of an e-passport's chip is enough to execute our ONTAP implementations. Further work may be done in proposing implementations for other signature schemes.

Digital signatures are often proven to be secure in the random oracle model while hash functions deviate more and more from this idealization. Clearly, if the hash function contains weaknesses, e.g., collisions are discovered, the whole signature scheme becomes weak. We proved that by only adding a (generic) pre-processing to the (weak) insecure signature implementation, we can transform it into a strong signature scheme. We also proposed a solution to avoid the increase in signature length.

Appendix

A

Birthday Paradox

In this section, we briefly present the result of the Birthday paradox in the case we are looking for a collision between two sets. The method we use is similar to the one from Katz and Lindell [KL08].

Consider we choose uniformly and at random q_1 and q_2 elements $x_1, \ldots, x_{q_1}$ and $y_1, \ldots, y_{q_2}$ among a set of size n. The birthday paradox gives the probability that there is a collision between these two sets, i.e., there exists a pair i, j such that $x_i = y_j$. We denote by Coll the event a collision occurred and by NoColl the inverse. Clearly,

$$\Pr[\mathsf{Coll}] = \Pr[\exists\, i, j, 1 \leq i \leq q_1, 1 \leq j \leq q_2 : x_i = y_j]$$

and

$$\Pr[\mathsf{Coll}] = 1 - \Pr[\mathsf{NoColl}] \ .$$

For the proof, we denote $\mathsf{NoColl}_{i,j}$ the event that there is no collision between the sets $\{x_1, \ldots, x_i\}$ and $\{y_1, \ldots, y_j\}$. If $\mathsf{NoColl}_{q_1,q_2}$ occurs then $\mathsf{NoColl}_{q_1,j}$ must have occurred for all $j < q_2$. Thus,

$$\begin{aligned}\Pr[\mathsf{NoColl}_{q_1,j}] \quad = \quad & \Pr[\mathsf{NoColl}_{q_1,0}] \cdot \Pr[\mathsf{NoColl}_{q_1,1} | \mathsf{NoColl}_{q_1,0}] \cdot \Pr[\mathsf{NoColl}_{q_1,2} | \mathsf{NoColl}_{q_1,1}] \cdot \ldots \\ & \ldots \cdot \Pr[\mathsf{NoColl}_{q_1,q_2} | \mathsf{NoColl}_{q_1,q_2-1}] \ .\end{aligned}$$

Note that $\Pr[\mathsf{NoColl}_{q_1,0}] = 1$ since the second set is empty and thus there is no element to collide. $\Pr[\mathsf{NoColl}_{q_1,j+1} | \mathsf{NoColl}_{q_1,j}]$ is the probability that the $j + 1^{\text{th}}$ element collides with

one in the first set. So, the probability is $1 - q_1/n$.
Now, we can write

$$\Pr[\mathsf{Coll}] = 1 - \Pr[\mathsf{NoColl}] = 1 - \prod_{i=1}^{q_2} \Pr[\mathsf{NoColl}_{q_1,i} | \mathsf{NoColl}_{q_1,i-1}] \ .$$

When $q_1 < n$, we have $1 - \frac{q_1}{n} \leq e^{-\frac{q_1}{n}}$ and finally we obtain

$$\Pr[\mathsf{Coll}] \geq 1 - \prod_{i=1}^{q_2} e^{-\frac{q_1}{n}} = 1 - e^{-\frac{q_1 q_2}{n}} \ .$$

Bibliography

[AA04] Dmitri Asonov and Rakesh Agrawal. Keyboard Acoustic Emanations. In *2004 IEEE Symposium on Security and Privacy (S&P 2004), 9-12 May 2004, Berkeley, CA, USA*, pages 3–11. IEEE Computer Society, 2004.

[AARR03] Dakshi Agrawal, Bruce Archambeault, Josyula R. Rao, and Pankaj Rohatgi. The EM Side-Channel(s). In Burton S. Kaliski Jr., Çetin Kaya Koç, and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems - CHES 2002, 4th International Workshop, Redwood Shores, CA, USA, August 13-15, 2002, Revised Papers*, volume 2523 of *Lecture Notes in Computer Science*, pages 29–45. Springer, 2003.

[AES01] Advanced encryption standard (AES). Federal Information Processing Standards, Publication 197, U.S. Department of Commerce, National Institute of Standards and Technology, 2001.

[AK99] Ross J. Anderson and Markus G. Kuhn. Soft Tempest – An Opportunity for NATO. *Protecting NATO Information Systems in the 21st Century, Washington, DC, Oct 25-26*, 1999.

[AK04] Ross J. Anderson and Markus G. Kuhn. Lost Cost Countermeasures Against Compromising Electromagnetic Computer Emanations. United States Patent US 6,721,324 B1, 2004.

[ASW98] N. Asokan, Victor Shoup, and Michael Waidner. Optimistic Fair Exchange of Digital Signatures (Extended Abstract). In Kaisa Nyberg, editor, *Advances in Cryptology – EUROCRYPT '98: International Conference on the Theory and Application of Cryptographic Techniques*, volume 1403 of *Lecture Notes in Computer Science*, pages 591–606, Espoo, Finland, May 1998. Springer-Verlag.

[ASW00] N. Asokan, Victor Shoup, and Michael Waidner. Optimistic Fair Exchange of Digital Signatures (Extended Abstract). *IEEE Journal on Selected Areas in Communications*, 18(4):593–610, 2000.

[BB05] Laurent Bussard and Walid Bagga. Distance-bounding proof of knowledge to avoid real-time attacks. In Ryôichi Sasaki, Sihan Qing, Eiji Okamoto, and Hiroshi Yoshiura, editors, *Security and Privacy in the Age of Ubiquitous Computing, IFIP TC11 20th International Conference on Information Security (SEC 2005), May 30 - June 1, 2005, Chiba, Japan*. Springer, 2005.

[BBS86] Lenore Blum, Manuel Blum, and Mike Shub. A Simple Unpredictable Pseudo-Random Number Generator. *SIAM J. Comput.*, 15(2):364–383, 1986.

[BC93] Stefan Brands and David Chaum. Distance-bounding protocols. In Tor Helleseth, editor, *Advances in cryptology – EUROCRYPT '93: Workshop on the Theory and Application of Cryptographic Techniques*, volume 765 of *Lecture Notes in Computer Science*, pages 344–359, Lofthus, Norway, May 1993. Springer-Verlag.

[BCC88] Gilles Brassard, David Chaum, and Claude Crépeau. Minimum disclosure proofs of knowledge. *Journal of Computer and System Sciences*, 37(2):156–189, 1988.

[BCJ+05] Eli Biham, Rafi Chen, Antoine Joux, Patrick Carribault, Christophe Lemuet, and William Jalby. Collisions of SHA-0 and Reduced SHA-1. In Ronald Cramer, editor, *Advances in Cryptology – EUROCRYPT '05: The 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques*, volume 3494 of *Lecture Notes in Computer Science*, pages 36–57, Aarhus, Denmark, 2005. Springer-Verlag.

[BCK96] Mihir Bellare, Ran Canetti, and Hugo Krawczyk. Keying Hash Functions for Message Authentication. In Neal Koblitz, editor, *Advances in Cryptology – CRYPTO '96: The 16th Annual International Cryptology Conference*, volume 1109 of *Lecture Notes in Computer Science*, pages 1–15, Santa Barbara, California, U.S.A., August 1996. Springer-Verlag.

[BCV08] Davide Balzarotti, Marco Cova, and Giovanni Vigna. ClearShot: Eavesdropping on Keyboard Input from Video. In *2008 IEEE Symposium on Security and Privacy (S&P 2008), 18-21 May 2008, Oakland, California, USA*, pages 170–183. IEEE Computer Society, 2008.

[BD05] Mike Burmester and Yvo Desmedt. A secure and scalable Group Key Exchange system. *Information Processiong Letter*, 94(3):137–143, 2005.

[BDJR97] Mihir Bellare, Anand Desai, E. Jokipii, and Phillip Rogaway. A Concrete Security Treatment of Symmetric Encryption. In IEEE Computer Society, editor, *38th Annual Symposium on Foundations of Computer Science, FOCS '97*, pages 394–403, Miami Beach, Florida, USA, 1997.

[BDPR97] Mihir Bellare, Anand Desai, David Pointcheval, and Phillip Rogaway. Relations Among Notions of Security for Public-Key Encryption Schemes. In Hugo Krawczyk, editor, *Advances in Cryptology – CRYPTO '98: The 18th Annual International Cryptology Conference*, volume 1462 of *Lecture Notes in Computer Science*, pages 26–45, Santa Barbara, California, U.S.A., August 1997. Springer-Verlag.

[BDU08] Michael Backes, Markus Dürmuth, and Dominique Unruh. Compromising Reflections-or-How to Read LCD Monitors around the Corner. In *2008 IEEE Symposium on Security and Privacy (S&P 2008), 18-21 May 2008, Oakland, California, USA*, pages 158–169. IEEE Computer Society, 2008.

[BG93] Mihir Bellare and Oded Goldreich. On Defining Proofs of Knowledge. In Ernest F. Brickell, editor, *Advances in Cryptology – CRYPTO '92: The 14th Annual International Cryptology Conference*, volume 740 of *Lecture Notes in Computer Science*, pages 390–420, Santa Barbara, California, U.S.A., August 1993. Springer-Verlag.

[BGP06] Côme Berbain, Henri Gilbert, and Jacques Patarin. QUAD: A Practical Stream Cipher with Provable Security. In Serge Vaudenay, editor, *Advances in Cryptology – EUROCRYPT '06: The 25th Annual International Conference on the Theory and Applications of Cryptographic Techniques*, volume 4004 of *Lecture Notes in Computer Science*, pages 109–128, St-Petersburg, Russia, May 2006. Springer-Verlag.

[BKK90] Joan F. Boyar, Stuart A. Kurtz, and Mark W. Krentel. A discrete logarithm implementation of perfect zero-knowledge blobs. *Journal of Cryptology*, 2(2):63–76, 1990.

[Blu03] Bluetooth. Specification of the Bluetooth System. Version 1.2, 2003.

[BLV03] Boaz Barak, Yehuda Lindell, and Salil P. Vadhan. Lower Bounds for Non-Black-Box Zero Knowledge. In *FOCS '03: The 44th Symposium on Foundations of Computer Science*, pages 384–393, Cambridge, MA, USA, October 2003. IEEE Computer Society.

[BLV04] Boaz Barak, Yehuda Lindell, and Salil P. Vadhan. Lower Bounds for Non-Black-Box Zero Knowledge. Cryptology ePrint Archive, Report 2004/226, 2004. `http://eprint.iacr.org/`. Full version of [BLV03].

[BR93a] Mihir Bellare and Phillip Rogaway. Entity Authentication and Key Distribution. In Douglas R. Stinson, editor, *Advances in Cryptology – CRYPTO '93: The 13th Annual International Cryptology Conference*, volume 773 of *Lecture Notes in Computer Science*, pages 232–249, Santa Barbara, California, U.S.A., August 1993. Springer-Verlag.

[BR93b] Mihir Bellare and Phillip Rogaway. Random oracles are practical: a paradigm for designing efficient protocols. In Ravi Ganesan and Ravi Sandhu, editors, *CCS '93: Proceedings of the 1st ACM conference on Computer and Communications Security*, pages 62–73, Fairfax, Virginia, U.S.A., 1993. ACM Press.

[BR95] Mihir Bellare and Phillip Rogaway. Provably secure session key distribution: the three party case. In *STOC '95: Proceedings of the twenty-seventh annual ACM Symposium on Theory Of Computing*, pages 57–66, Las Vegas, Nevada, U.S.A., May 1995. ACM press.

[BR96] Mihir Bellare and Phillip Rogaway. The Exact Security of Digital Signatures – How to Sign with RSA and Rabin. In Maurer Ueli, editor, *Advances in Cryptology – EUROCRYPT '96: Workshop on the Theory and Application of Cryptographic Techniques*, volume 1070 of *Lecture Notes in Computer Science*, pages 399–416, Saragossa, Spain, May 1996. Springer-Verlag.

[BR97] Mihir Bellare and Phillip Rogaway. Towards Making UOWHFs Practical. In Burton S. Kaliski Jr., editor, *Advances in Cryptology – CRYPTO '97: The 17th Annual International Cryptology Conference*, volume 1294 of *Lecture Notes in Computer Science*, pages 470–484, Santa Barbara, California, U.S.A., August 1997. Springer-Verlag.

[BS99] Mihir Bellare and Amit Sahai. Non-malleable Encryption: Equivalence between Two Notions, and an Indistinguishability-Based Characterization. In Michael J. Wiener, editor, *Advances in Cryptology – CRYPTO '99: The 19th Annual International Cryptology Conference*, volume 1666 of *Lecture Notes in Computer Science*, pages 519–536, Santa Barbara, California, U.S.A., August 1999. Springer-Verlag.

[BSNS05] Joonsang Baek, Reihaneh Safavi-Naini, and Willy Susilo. Universal Designated Verifier Signature Proof (or How to Efficiently Prove Knowledge of a Signature). In Bimal K. Roy, editor, *Advances in Cryptology – ASIACRYPT '05: The 11th International Conference on the Theory and Application of Cryptology and Information Security*, volume 3788 of *Lecture Notes in Computer Science*, pages 644–661, Chennai, India, December 2005. Springer-Verlag.

[BSSW02] Dirk Balfanz, Diana K. Smetters, Paul Stewart, and H. Chi Wong. Talking To Strangers: Authentication in Ad-Hoc Wireless Networks. In *Proceedings of NDSS '02: The Network and Distributed System Security Symposium*, San Diego, California, U.S.A, February 2002.

[BWY06] Yigael Berger, Avishai Wool, and Arie Yeredor. Dictionary attacks using keyboard acoustic emanations. In Ari Juels, Rebecca N. Wright, and Sabrina De Capitani di Vimercati, editors, *Proceedings of the 13th ACM Conference*

on Computer and Communications Security, CCS 2006, Alexandria, VA, USA, Ioctober 30 - November 3, 2006, pages 245–254. ACM, 2006.

[Can97] Ran Canetti. Towards Realizing Random Oracles: Hash Functions that Hide All Partial Information. In Burton S. Kaliski Jr., editor, *Advances in Cryptology – CRYPTO '97: The 17th Annual International Cryptology Conference*, volume 1294 of *Lecture Notes in Computer Science*, pages 455–469, Santa Barbara, California, U.S.A., August 1997. Springer-Verlag.

[Can00] Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. Cryptology ePrint Archive, Report 2000/067. `http://eprint.iacr.org/`, 2000.

[Can01] Ran Canetti. Universally Composable Security: A New Paradigm for Cryptographic Protocols. In *FOCS*, pages 136–145, 2001.

[CCH05] Mario Cagalj, Srdjan Capkun, and Jean-Pierre Hubaux. Key Agreement in Peer-to-peer Wireless Networks. In *Proceedings of the IEEE, Special Issue in Security and Cryptography*, 2005.

[CDM00] Ronald Cramer, Ivan Damgård, and Philip D. MacKenzie. Efficient Zero-Knowledge Proofs of Knowledge Without Intractability Assumptions. In Hideki Imai and Yuliang Zheng, editors, *Public Key Cryptography – PKC '00: The 3rd International Workshop on Practice and Theory in Public Key Cryptography*, volume 1751 of *Lecture Notes in Computer Science*, pages 354–372, Melbourne, Victoria, Australia, January 2000. Springer-Verlag.

[CF01] Ran Canetti and Marc Fischlin. Universally Composable Commitments. In Joe Kilian, editor, *Advances in Cryptology – CRYPTO '01: The 21st Annual International Cryptology Conference*, volume 2139 of *Lecture Notes in Computer Science*, pages 19–40, Santa Barbara, California, USA, August 2001. Springer-Verlag.

[CGH98] Ran Canetti, Oded Goldreich, and Shai Halevi. The random oracle methodology, revisited (preliminary version). In *STOC '98: Proceedings of the thirtieth annual ACM Symposium on Theory Of Computing*, pages 209–218, Dallas, Texas, USA, 1998. ACM Press.

[CGHGN01] Dario Catalano, Rosario Gennaro, Nick Howgrave-Graham, and Phong Q. Nguyen. Paillier's Cryptosystem Revisited. In Pierangela Samarati, editor, *CCS '01: Proceedings of the 8th ACM conference on Computer and Communications Security*, pages 206–214, Philadelphia, Pennsylvania, U.S.A., 2001. ACM Press.

[Cha84] David Chaum. Blind Signature System. In David Chaum, editor, *Advances in Cryptology – CRYPTO '83*, page 153. Plenum Press, 1984.

[Cha94] David Chaum. Designated Confirmer Signatures. In Alfredo De Santis, editor, *Advances in Cryptology – EUROCRYPT '94: Workshop on the Theory and Application of Cryptographic Techniques*, volume 950 of *Lecture Notes in Computer Science*, pages 86–91, Perugia, Italy, May 1994. Springer-Verlag.

[CL08] Zhengjun Cao and Mulan Liu. Classification of signature-only signature models. *Science in China Series F: Information Sciences*, 51(8):1083–1095, 2008.

[CM00] Jan Camenisch and Markus Michels. Confirmer Signature Schemes Secure against Adaptive Adversaries. In Bart Preneel, editor, *Advances in Cryptology – EUROCRYPT '00: International Conference on the Theory and Application of Cryptographic Techniques*, volume 1807 of *Lecture Notes in Computer Science*, pages 243–258, Bruges, Belgium, May 2000. Springer-Verlag.

[CS00] Ronald Cramer and Victor Shoup. Signature Schemes Based on the Strong RSA Assumption. *ACM Transactions on Information and System Security*, 3(3):161–185, 2000.

[CS02] Ronald Cramer and Victor Shoup. Universal Hash Proofs and a Paradigm for Adaptive Chosen Ciphertext Secure Public-Key Encryption. In Lars R. Knudsen, editor, *Advances in Cryptology – EUROCRYPT '02: International Conference on the Theory and Applications of Cryptographic Techniques*, volume 2332 of *Lecture Notes in Computer Science*, pages 45–64, Amsterdam, The Netherlands, April 2002. Springer-Verlag.

[CvA90] David Chaum and Hans van Antwerpen. Undeniable Signatures. In Gilles Brassard, editor, *Advances in Cryptology – CRYPTO '89: The 9th Annual International Cryptology Conference*, volume 435 of *Lecture Notes in Computer Science*, pages 212–217, Santa Barbara, California, U.S.A., 1990. Springer-Verlag.

[CvH91] David Chaum and Eugène van Heyst. Group Signatures. In Donald W. Davies, editor, *Advances in Cryptology – EUROCRYPT '91, Workshop on the Theory and Application of Cryptographic Techniques*, volume 547 of *Lecture Notes in Computer Science*, pages 257–265, Brighton, UK, April 1991. Springer-Verlag.

[CW79] Larry Carter and Mark N. Wegman. Universal Classes of Hash Functions. *Journal of Computer and System Sciences*, 18(2):143–154, 1979.

[Dam90] Ivan Damgård. A Design Principle for Hash Functions. In Gilles Brassard, editor, *Advances in Cryptology – CRYPTO '89: The 9th Annual International Cryptology Conference*, volume 435 of *Lecture Notes in Computer Science*, pages 416–427, Santa Barbara, California, U.S.A., 1990. Springer-Verlag.

[Dam00] Ivan Damgård. Efficient Concurrent Zero-Knowledge in the Auxiliary String Model. In Bart Preneel, editor, *Advances in Cryptology – EUROCRYPT '00: International Conference on the Theory and Application of Cryptographic Techniques*, volume 1807 of *Lecture Notes in Computer Science*, pages 418–430, Bruges, Belgium, May 2000. Springer-Verlag.

[Dam05] Ivan Damgård. On Σ-protocols. Lecture Notes, 2005.

[DB94] Yvo Desmedt and Mike Burmester. A secure and efficient conference key distribution system (extended abstract). In Alfredo De Santis, editor, *Advances in Cryptology – EUROCRYPT '94: Workshop on the Theory and Application of Cryptographic Techniques*, volume 950 of *Lecture Notes in Computer Science*, pages 275–286, Perugia, Italy, May 1994. Springer-Verlag.

[DCIO98] Giovanni Di Crescenzo, Yuval Ishai, and Rafail Ostrovsky. Non-Interactive and Non-Malleable Commitment. In *STOC '98: Proceedings of the thirtieth annual ACM Symposium on Theory Of Computing*, pages 141–150, Dallas, Texas, USA, 1998. ACM Press.

[DDN91] Danny Dolev, Cynthia Dwork, and Moni Naor. Non-Malleable Cryptography (Extended Abstract). In *STOC '91: Proceedings of the Twenty Third Annual ACM Symposium on Theory of Computing*, pages 542–552, New Orleans, Louisiana, U.S.A., May 1991. ACM Press.

[DDN03] Danny Dolev, Cynthia Dwork, and Moni Naor. Nonmalleable Cryptography. *SIAM Review*, 45(4):727–784, 2003.

[DES77] Announcing the Data Encryption Standard (DES). Federal Information Processing Standard, Publication 46, National Institute of Standards and Technology (NIST), 1977.

[Des88] Yvo Desmedt. Subliminal-Free Authentication and Signature (Extended Abstract). In C. G. Günther, editor, *Advances in Cryptology – EUROCRYPT '88: Workshop on the Theory and Application of Cryptographic Techniques*, volume 330 of *Lecture Notes in Computer Science*, pages 23–33, Davos, Switzerland, August 1988. Springer-Verlag.

[DES99] Data Encryption Standard (DES). Federal Information Processing Standard, Publication 46-3, National Institute of Standards and Technology (NIST), 1999.

[DF90] Yvo Desmedt and Yair Frankel. Threshold Cryptosystems. In Gilles Brassard, editor, *Advances in Cryptology – CRYPTO '89: The 9th Annual International Cryptology Conference*, volume 435 of *Lecture Notes in Computer Science*, pages 307–315, Santa Barbara, California, U.S.A., 1990. Springer-Verlag.

[DG03] Ivan Damgård and Jens Groth. Non-interactive and reusable non-malleable commitment schemes. In *STOC '03: Proceedings of the thirty-fifth annual ACM Symposium on Theory Of Computing*, pages 426–437, San Diego, California, U.S.A., 2003. ACM Press.

[DH76] Whitfield Diffie and Martin E. Hellman. New Directions in Cryptography. *IEEE Transactions on Information Theory*, IT–22(6):644–654, November 1976.

[DN02] Ivan Damgård and Jesper Buus Nielsen. Perfect Hiding and Perfect Binding Universally Composable Commitment Schemes with Constant Expansion Factor. In Moti Yung, editor, *Advances in Cryptology – CRYPTO '02: The 22nd Annual International Cryptology Conference*, volume 2442 of *Lecture Notes in Computer Science*, pages 581–596, Santa Barbara, California, U.S.A., August 2002. Springer-Verlag.

[DSS94] Digital signature standard (DSS). Federal Information Processing Standard, Publication 186, U.S. Department of Commerce, National Institute of Standards and Technology, 1994.

[DSS00] Digital signature standard (DSS). Federal Information Processing Standard, Publication 186-2, U.S. Department of Commerce, National Institute of Standards and Technology, 2000.

[ECD98] ANSI X9.62. Public Key Cryptography for the Financial Service Industry: The Elliptic Curve Digital Signature Algorithm (ECDSA). American National Standard Institute. American Bankers Associtaion, 1998.

[EL85] Wim Van Eck and Neher Laborato. Electromagnetic Radiation from Video Display Units: An Eavesdropping Risk. *Computers & Security*, 4:269–286, 1985.

[ElG85] T. ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, 31(4):469–472, 1985.

[FF00] Marc Fischlin and Roger Fischlin. Efficient Non-malleable Commitment Schemes. In Mihir Bellare, editor, *Advances in Cryptology – CRYPTO '00: The 20th Annual International Cryptology Conference*, volume 1880 of *Lecture Notes in Computer Science*, pages 413–431, Santa Barbara, California, U.S.A., 2000. Springer-Verlag.

[FS87] Amos Fiat and Adi Shamir. How to Prove Yourself: Practical Solutions to Identification and Signature Problems. In Andrew M. Odlyzko, editor, *Advances in Cryptology – CRYPTO '86: A Conference on the Theory and Applications of Cryptographic Techniques*, volume 263 of *Lecture Notes in Computer Science*, pages 186–194, Santa Barbara, California, USA, August 1987. Springer-Verlag.

[GK96] Oded Goldreich and Ariel Kahan. How To Construct Constant-Round Zero-Knowledge Proof Systems for NP. *Journal of Cryptology*, 9(3):167–189, 1996.

[GKR00] Rosario Gennaro, Hugo Krawczyk, and Tal Rabin. RSA-Based Undeniable Signatures. *Journal of Cryptology*, 13(4):397–416, 2000.

[GMN04] Christian Gehrmann, Chris J. Mitchell, and Kaisa Nyberg. Manual Authentication for Wireless Devices. *RSA Cryptobytes*, 7(1):29–37, January 2004.

[GMO01] Karine Gandolfi, Christophe Mourtel, and Francis Olivier. Electromagnetic Analysis: Concrete Results. In Çetin Kaya Koç, David Naccache, and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems - CHES 2001, Third International Workshop, Paris, France, May 14-16, 2001, Proceedings*, volume 2162 of *Lecture Notes in Computer Science*, pages 251–261. Springer, 2001.

[GMR84] Shafi Goldwasser, Silvio Micali, and Ronald L. Rivest. A "Paradoxical"'Solution to the Signature Problem (Abstract). In G. R. Blakley and David Chaum, editors, *Advances in Cryptology, Proceedings of CRYPTO '84, Santa Barbara, California, USA, August 19-22, 1984, Proceedings*, volume 196 of *Lecture Notes in Computer Science*, pages 441–448. Springer-Verlag, 1984.

[GMR85] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The Knowledge Complexity of Interactive Proof-Systems. In *STOC '85: Proceedings of the seventeenth annual ACM Symposium on Theory Of Computing*, pages 291–304, Providence, RI, U.S.A., 1985. ACM press.

[GMR88] Shafi Goldwasser, Silvio Micali, and Ronald L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM Journal on Computing*, 17(2):281–308, 1988.

[GMR89] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The Knowledge Complexity of Interactive Proof Systems. *SIAM Journal on Computing*, 18(1):186–208, 1989.

[GMW91] Oded Goldreich, Silvio Micali, and Avi Wigderson. Proofs that Yield Nothing But Their Validity or All Languages in NP Have Zero-Knowledge Proof Systems. *Journal of the ACM*, 38(1):691–729, 1991.

[GN04] Christian Gehrmann and Kaisa Nyberg. Security in Personal Area Networks. *Security for Mobility*, pages 191–230, 2004.

[Gol01] Oded Goldreich. *Foundations of Cryptography.* Cambridge University Press, 2001.

[GQ88] Louis C. Guillou and Jean-Jacques Quisquater. A Practical Zero-Knowledge Protocol Fitted to Security Microprocessor Minimizing Both Trasmission and Memory. In C. G. Günther, editor, *Advances in Cryptology – EUROCRYPT '88: Workshop on the Theory and Application of Cryptographic Techniques*, volume 330 of *Lecture Notes in Computer Science*, pages 123–128, Davos, Switzerland, August 1988. Springer-Verlag.

[GQ90] Louis C. Guillou and Jean-Jacques Quisquater. A "Paradoxical" Identity-Based Signature Scheme Resulting from Zero-Knowledge. In Shafi Goldwasser, editor, *Advances in Cryptology – CRYPTO '88: The 8th Annual International Cryptology Conference*, volume 403 of *Lecture Notes in Computer Science*, pages 216–231, Santa Barbara, California, U.S.A., 1990. Springer-Verlag.

[HFPS99] R. Housley, W. Ford, W. Polk, and D. Solo. RFC 2459: Internet X.509 Public Key Infrastructure Certificate and CRL Profile. IETF RFC Publication, January 1999.

[HK06a] Shai Halevi and Hugo Krawczyk. Strengthening Digital Signatures via Randomized Hashing. In Victor Shoup, editor, *Advances in Cryptology – CRYPTO '06: The 26th Annual International Cryptology Conference*, volume 4117 of *Lecture Notes in Computer Science*, pages 41–59, Santa Barbara, California, U.S.A., August 2006. Springer-Verlag.

[HK06b] Shai Halevi and Hugo Krawczyk. The RMX Transform and Digital Signatures. http://www.ee.technion.ac.il/∼hugo/rhash/, 2006.

[Hoe04] Jaap-Henk Hoepman. The Ephemeral Pairing Problem. In Ari Juels, editor, *Financial Cryptography – FC '04: The 8th International Conference*, volume 3110 of *Lecture Notes in Computer Science*, pages 212–226, Key West, FL, USA, February 2004. Springer-Verlag.

[IK03] Tetsu Iwata and Kaoru Kurosawa. OMAC: One-Key CBC MAC. In Thomas Johansson, editor, *FSE '03: Fast Software Encryption: 10th International Workshop*, volume 2887 of *Lecture Notes in Computer Science*, pages 129–153, Lund, Sweden, January 2003. Springer-Verlag.

[IN83] K. Itakura, , and K. Nakamura. A public-key cryptosystem suitable for digital multisignatures. NEC Research Development 71, October 1983.

[JSI96] Markus Jakobsson, Kazue Sako, and Russell Impagliazzo. Designated Verifier Proofs and Their Applications. In Maurer Ueli, editor, *Advances in Cryptology – EUROCRYPT '96: Workshop on the Theory and Application of Cryptographic Techniques*, volume 1070 of *Lecture Notes in Computer Science*, pages 143–154, Saragossa, Spain, May 1996. Springer-Verlag.

[JV96] Mike Just and Serge Vaudenay. Authenticated Multi-Party Key Agreement. In Kwangjo Kim and Tsutomu Matsumoto, editors, *Advances in Cryptology – ASIACRYPT '96: International Conference on the Theory and Applications of Cryptology and Information Security*, volume 1163 of *Lecture Notes in Computer Science*, pages 36–49, Kyongju, Korea, November 1996. Springer-Verlag.

[JV03] Pascal Junod and Serge Vaudenay. FOX Specifications Version 1.1. In *Technical Report EPFL/IC/2003/82*, EPFL, 2003.

[KA98] Markus G. Kuhn and Ross J. Anderson. Soft Tempest: Hidden Data Transmission Using Electromagnetic Emanations. In David Aucsmith, editor, *Information Hiding, Second International Workshop, Portland, Oregon, USA, April 14-17, 1998, Proceedings*, volume 1525 of *Lecture Notes in Computer Science*, pages 124–142. Springer, 1998.

[KL08] Jonathan Katz and Yahuda Lindell. *Introduction to Modern Cryptography.* Chapman and Hall/CRC press, 2008.

[Kra94] Hugo Krawczyk. LFSR-based Hashing and Authentication. In Yvo Desmedt, editor, *Advances in Cryptology – CRYPTO '94: The 14th Annual International Cryptology Conference*, volume 839 of *Lecture Notes in Computer Science*, pages 129–139, Santa Barbara, California, U.S.A., August 1994. Springer-Verlag.

[Kuh03] Markus G. Kuhn. Compromising Emanations: Eavesdropping risks of Computer Displays. *Technical Report UCAM-CL-TR-577*, 2003.

[Kuh05] Markus G. Kuhn. Security Limits for Compromising Emanations. In Josyula R. Rao and Berk Sunar, editors, *Cryptographic Hardware and Embedded Systems - CHES 2005, 7th International Workshop, Edinburgh, UK, August 29 - September 1, 2005, Proceedings*, volume 3659 of *Lecture Notes in Computer Science*, pages 265–279. Springer, 2005.

[LdW05] Arjen K. Lenstra and Benne de Weger. On the Possibility of Constructing Meaningful Hash Collisions for Public Keys. In Colin Boyd and Juan Manuel González Nieto, editors, *ACISP '05: The 10th Australasian Conference on Information Security and Privacy*, volume 3574 of *Lecture Notes in Computer Science*, pages 267–279, Brisbane, Australia, 2005. Springer-Verlag.

[Lin03] Yehuda Lindell. General composition and universal composability in secure multi-party computation. In *FOCS '03: The 44th Symposium on Foundations of Computer Science*, pages 394–403, Cambridge, MA, USA, October 2003. IEEE Computer Society.

[Lis07] Moses Liskov. Constructing an Ideal Hash Function from Weak Ideal Compression Functions. In Eli Biham and Amr R. Youssef, editors, *SAC '06: The 13th Annual Workshop on Selected Areas in Cryptography*, pages 358–375, Montreal, Quebec, Canada, August 2007.

[LN06a] Sven Laur and Kaisa Nyberg. Efficient Mutual Data Authentication Using Manually Authenticated Strings. In David Pointceval, Yi Mu, and Kefei Chen, editors, *CANS '06: The 5th International Conference on Cryptology and Network Security*, volume 4301 of *LNCS*, pages 90–107, Suzhou, Jiangsu, China, December 2006. Springer-Verlag.

[LN06b] Sven Laur and Kaisa Nyberg. Efficient Mutual Data Authentication Using Manually Authenticated Strings (Extended version). `http://www.tcs.hut.fi/Publications/slaur/MANA-IV.pdf`, 2006.

[LP08] Sven Laur and Sylvain Pasini. SAS-Based Group Authentication and Key Agreement Protocols. In Ronald Cramer, editor, *Public Key Cryptography – PKC '08: The 11th International Conference on Theory and Practice of Public Key Cryptography*, volume 4939 of *Lecture Notes in Computer Science*, pages 197–213, Barcelona, Spain, March 2008. Springer-Verlag.

[LP09] Sven Laur and Sylvain Pasini. User-Aided Data Authentication. *International Journal of Security and Networks*, 4(1):69–86, 2009.

[LU02] Joe Loughry and David A. Umphress. Information leakage from optical emanations. *ACM Trans. Inf. Syst. Secur.*, 5(3):262–289, 2002.

[LW06] Jin Li and Yanming Wang. Universal Designated Verifier Ring Signature (Proof) Without Random Oracles. In Xiaobo Zhou, Oleg Sokolsky, Lu Yan, Eun-Sun Jung, Zili Shao, Yi Mu, Dong Chun Lee, Daeyoung Kim, Young-Sik Jeong, and Cheng-Zhong Xu, editors, *Emerging directions in Embedded and Ubiquitous Computing, EUC '06 Workshops: NCUS, SecUbiq, USN, TRUST, ESO, and MSA*, volume 4097 of *Lecture Notes in Computer Science*, pages 332–341, Seoul, Korea, August 2006. Springer-Verlag.

[LWdW05] Arjen Lenstra, Xiaoyun Wang, and Benne de Weger. Colliding X.509 Certificates. Cryptology ePrint Archive, Report 2005/067, 2005. `http://eprint.iacr.org/`.

[Mas08] Atefeh Mashatan. Message Authentication and Recognition Protocols Using Two-Channel Cryptography. PhD thesis, University of Waterloo, Ontario, Canada, 2008.

[Mer78] Ralph C. Merkle. Secure communications over insecure channels. *Communications of the ACM*, 21(4):294–299, 1978.

[Mer90] Ralph C. Merkle. One Way Hash Functions and DES. In Gilles Brassard, editor, *Advances in Cryptology – CRYPTO '89: The 9th Annual International Cryptology Conference*, volume 435 of *Lecture Notes in Computer Science*, pages 428–446, Santa Barbara, California, U.S.A., 1990. Springer-Verlag.

[Mir06] Ilya Mironov. Collision-Resistant No More: Hash-and-Sign Paradigm Revisited. In Moti Yung, Yevgeniy Dodis, Aggelos Kiayias, and Tal Malkin, editors, *Public Key Cryptography – PKC '06: The 9th International Conference on Theory and Practice of Public Key Cryptography*, volume 3958 of *Lecture Notes in Computer Science*, pages 140–156, New York, USA, April 2006. Springer-Verlag.

[Mon06] Jean Monnerat. Short Undeniable Signatures: Design, Analysis, and Applications. PhD thesis, n°3691, EPFL, Lausanne, Switzerland, 2006.

[MÖPV07] Elke De Mulder, Siddika Berna Örs, Bart Preneel, and Ingrid Verbauwhede. Differential power and electromagnetic attacks on a FPGA implementation of elliptic curve cryptosystems. *Computers & Electrical Engineering*, 33(5-6):367–382, 2007.

[MPV09] Jean Monnerat, Sylvain Pasini, and Serge Vaudenay. Efficient Deniable Authentication for Standard Signatures. In Michel Abdalla and David Pointcheval, editors, *ACNS '09: International Conference on Applied Cryptography and Network Security*, volume 5536 of *Lecture Notes in Computer Science*, pages ???–???, Paris-Rocquencourt, France, June 2009. Springer-Verlag.

[MRT04a] Machine Readable Travel Documents. Development of a Logical Data Structure — LDS For Optional Capacity Expansion Technologies. Version 1.7., 2004. `http://www.icao.int/mrtd/download/technical.cfm`.

[MRT04b] Machine Readable Travel Documents. PKI for Machine Readable Travel Documents offering ICC Read-Only Access. Version 1.1., 2004. `http://www.icao.int/mrtd/download/technical.cfm`.

[MS07] Atefeh Mashatan and Douglas R. Stinson. Noninteractive Two-Channel Message Authentication based on Hybrid-Collision Resistant Hash Functions. *IET Proceedings Information Security*, 1(3):111–118, 2007.

[MS08] Atefeh Mashatan and Douglas R. Stinson. Interactive Two-Channel Message Authentication based on Interactive-Collision Resistant Hash Functions. *International Journal of Information Security*, 2008.

[MUO96] Masahiro Mambo, Keisuke Usuda, and Eiji Okamoto. Proxy signatures for delegating signing operation. In *ACM Conference on Computer and Communications Security*, pages 48–57, 1996.

[MV04] Jean Monnerat and Serge Vaudenay. Generic Homomorphic Undeniable Signatures. In Pil Joong Lee, editor, *Advances in Cryptology – ASIACRYPT '04: The 10th International Conference on the Theory and Application of Cryptology and Information Security*, volume 3329 of *Lecture Notes in Computer Science*, pages 354–371, Jeju Island, Korea, December 2004. Springer-Verlag.

[MVV07] Jean Monnerat, Serge Vaudenay, and Martin Vuagnoux. About Machine-Readable Travel Documents – Privacy Enhancement Using (Weakly) Non-Transferable Data Authentication. In *Proceedings of RFIDSEC '07*, 2007.

[MY04] Philip MacKenzie and Ke Yang. On Simulation-Sound Trapdoor Commitments. In Christian Cachin and Jan Camenisch, editors, *Advances in Cryptology – EUROCRYPT '04: International Conference on the Theory and Applications of Cryptographic Techniques*, volume 3027 of *Lecture Notes in Computer Science*, pages 382–400, Interlaken, Switzerland, May 2004. Springer-Verlag.

[NSS06] Moni Naor, Gil Segev, and Adam Smith. Tight bounds for unconditional authentication protocols in the manual channel and shared key models. In Victor Shoup, editor, *Advances in Cryptology – CRYPTO '06: The 26th Annual International Cryptology Conference*, volume 4117 of *Lecture Notes in Computer Science*, pages 214–231, Santa Barbara, California, U.S.A., August 2006. Springer-Verlag.

[NY89] Moni Naor and Moti Yung. Universal one-way hash functions and their cryptographic applications. In *Proceedings of the Twenty First Annual ACM Symposium on Theory of Computing*, pages 33–43, New York, NY, USA, 1989. Seattle, WA, USA.

[OO91] Tatsuaki Okamoto and Kazuo Ohta. How to Utilize the Randomness of Zero-Knowledge Proofs. In Alfred J. Menezes and Scott A. Vanstone, editors, *Advances in Cryptology – CRYPTO '90: The 10th Annual International Cryptology Conference*, volume 537 of *Lecture Notes in Computer Science*, pages 456–475, Santa Barbara, California, U.S.A., 1991. Springer-Verlag.

[Pai99] Pascal Paillier. Public-Key Cryptosystems Based on Composite Degree Residuosity Classes. In Jacques Stern, editor, *Advances in Cryptology – EUROCRYPT '99: International Conference on the Theory and Application of Cryptographic Techniques*, volume 1592 of *Lecture Notes in Computer Science*, pages 223–238, Prague, Czech Republic, May 1999. Springer-Verlag.

[Pas03] Rafael Pass. On Deniability in the Common Reference String and Random Oracle Model. In Dan Boneh, editor, *Advances in Cryptology – CRYPTO '03: The 23rd Annual International Cryptology Conference*, volume 2729 of *Lecture Notes in Computer Science*, pages 316–337, Santa Barbara, California, U.S.A., August 2003. Springer-Verlag.

[Pas04] Rafael Pass. Alternative Variants of Zero-Knowledge Proofs. Licentiate Thesis, Stockholm, Sweden, 2004.

[Pas05] Sylvain Pasini. Secure Communications over Insecure Channels Using an Authenticated Channel. Master's thesis, Swiss Federal Institute of Technology (EPFL), 2005. `http://lasecwww.epfl.ch/`.

[Ped91] Torben P. Pedersen. Non-Interactive and Information-Theoretic Secure Verifiable Secret Sharing. In Joan Feigenbaum, editor, *Advances in Cryptology – CRYPTO '91: The 11th Annual International Cryptology Conference*, volume 576 of *Lecture Notes in Computer Science*, pages 129–140, Santa Barbara, California, U.S.A., 1991. Springer-Verlag.

[Pfi91] Birgit Pfitzmann. Fail-stop signatures. In *Compsec '91: 8th world conference on computer security, audit and control*, pages 125–134, 1991.

[PV05] Thomas Peyrin and Serge Vaudenay. The Pairing Problem with User Interaction. In Ryôichi Sasaki, Sihan Qing, Eiji Okamoto, and Hiroshi Yoshiura, editors, *Security and Privacy in the Age of Ubiquitous Computing, IFIP TC11 20th International Conference on Information Security (SEC 2005), May 30 - June 1, 2005, Chiba, Japan*, pages 251–266. Springer, 2005.

[PV06a] Sylvain Pasini and Serge Vaudenay. An Optimal Non-interactive Message Authentication Protocol. In David Pointcheval, editor, *Topics in Cryptology – CT-RSA '06: The Cryptographers' Track at the RSA Conference 2006*, volume 3860 of *Lecture Notes in Computer Science*, pages 280–294, San Jose, CA, USA, February 2006. Springer-Verlag.

[PV06b] Sylvain Pasini and Serge Vaudenay. SAS-based Authenticated Key Agreement. In Moti Yung, Yevgeniy Dodis, Aggelos Kiayias, and Tal Malkin, editors, *Public Key Cryptography – PKC '06: The 9th International Conference on Theory and Practice of Public Key Cryptography*, volume 3958 of *Lecture Notes in Computer Science*, pages 395–409, New York, USA, April 2006. Springer-Verlag.

[PV07] Sylvain Pasini and Serge Vaudenay. Hash-and-sign with Weak Hashing Made Secure. In Josef Pieprzyk, Hossein Ghodosi, and Ed Dawson, editors, *ACISP '07: The 12th Australasian Conference on Information Security and Privacy*, volume 4586 of *Lecture Notes in Computer Science*, pages 338–354, Townsville, Australia, 2007. Springer-Verlag.

[PW01] Birgit Pfitzmann and Michael Waidner. A model for asynchronous reactive systems and its application to secure message transmission. In *IEEE Symposium on Security and Privacy*, pages 184–, 2001.

[QS01] Jean-Jacques Quisquater and David Samyde. ElectroMagnetic Analysis (EMA): Measures and Counter-Measures for Smart Cards. In Isabelle Attali and Thomas P. Jensen, editors, *Smart Card Programming and Security, International Conference on Research in Smart Cards, E-smart 2001, Cannes, France, September 19-21, 2001, Proceedings*, volume 2140 of *Lecture Notes in Computer Science*, pages 200–210. Springer, 2001.

[Riv91] Ronald L. Rivest. The MD4 Message Digest Algorithm. In Alfred J. Menezes and Scott A. Vanstone, editors, *Advances in Cryptology – CRYPTO '90: The 10th Annual International Cryptology Conference*, volume 537 of *Lecture Notes in Computer Science*, pages 303–311, Santa Barbara, California, U.S.A., 1991. Springer-Verlag.

[Riv92] Ronald L. Rivest. The MD5 message digest algorithm. Technical Report Internet RFC-1321,IETF, 1992.

[Rog95] Phillip Rogaway. Bucket Hashing and its Application to Fast Message Authentication. In Don Coppersmith, editor, *Advances in Cryptology – CRYPTO '95: The 15th Annual International Cryptology Conference*, volume 963 of *Lecture Notes in Computer Science*, pages 29–42, Santa Barbara, California, U.S.A., August 1995. Springer-Verlag.

[Rog06] Phillip Rogaway. Formalizing Human Ignorance: Collision-Resistant Hashing without the Keys. In Phong Quang Nguyen, editor, *VietCrypt '06: International Conference on Cryptology*, volume 4341 of *Lecture Notes in Computer Science*, pages 221–228, Hanoi, Vietnam, September 2006. Springer-Verlag.

[RS84] Ronald L. Rivest and Adi Shamir. How to Expose an Eavesdropper. *Commun. ACM*, 27(4):393–394, 1984.

[RSA78] Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman. A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. *Communications of the ACM*, 21(2):120–126, Februar 1978.

[RST01] Ronald L. Rivest, Adi Shamir, and Yael Tauman. How to leak a secret. In Colin Boyd, editor, *Advances in cryptology – ASIACRYPT '01: The 7th International Conference on the Theory and Application of Cryptology and Information Security*, volume 2248 of *Lecture Notes in Computer Science*, pages 552–565, Gold Coast, Australia, January 2001. Springer-Verlag.

[RWSN07] Mohammad Reza Reyhanitabar, Shuhong Wang, and Reihaneh Safavi-Naini. Non-interactive Manual Channel Message Authentication Based on eTCR Hash Functions. In Josef Pieprzyk, Hossein Ghodosi, and Ed Dawson, editors, *ACISP '07: The 12th Australasian Conference on Information Security and*

Privacy, volume 4586 of *Lecture Notes in Computer Science*, pages 385–399, Townsville, Australia, 2007. Springer-Verlag.

[SA99] Frank Stajano and Ross Anderson. The resurrecting duckling: Security issues for ad-hoc wireless networks. In Bruce Christianson, Bruno Crispo, James A. Malcolm, and Michael Roe, editors, *Security Protocols, 7th International Workshop Proceedings*, volume 1796 of *Lecture Notes in Computer Science*, pages 172–194, Cambridge, UK, 1999. Springer-Verlag.

[SBWP03] Ron Steinfeld, Laurence Bull, Huaxiong Wang, and Josef Pieprzyk. Universal Designated-Verifier Signatures. In Chi-Sung Laih, editor, *Advances in cryptology – ASIACRYPT '03: The 9th International Conference on the Theory and Application of Cryptology and Information Security*, volume 2894 of *Lecture Notes in Computer Science*, pages 523–542, Taipei, Taiwan, January 2003. Springer-Verlag.

[Sch90] Claus-Peter Schnorr. Efficient Identification and Signatures for Smart Cards. In Gilles Brassard, editor, *Advances in Cryptology – CRYPTO '89: The 9th Annual International Cryptology Conference*, volume 435 of *Lecture Notes in Computer Science*, pages 239–252, Santa Barbara, California, U.S.A., 1990. Springer-Verlag.

[Sch91] Claus-Peter Schnorr. Efficient Signature Generation by Smart Cards. *Journal of Cryptology*, 4(3):161–174, 1991.

[Sha49] C. E. Shannon. Communication Theory of Secrecy Systems. *Bell Sys. Tech. J.*, 28:657–715, 1949.

[SHA93] Secure hash standard. Federal Information Processing Standard, Publication 180, U.S. Department of Commerce, National Institute of Standards and Technology, 1993.

[SHA95] Secure hash standard. Federal Information Processing Standard, Publication 180-1, U.S. Department of Commerce, National Institute of Standards and Technology, 1995.

[Sho04] Victor Shoup. Sequences of Games: A Tool for Taming Complexity in Security Proofs. Cryptology ePrint Archive, Report 2004/332, 2004. `http://eprint.iacr.org/`.

[Smu90] Peter Smulders. The Threat of Information Theft by Reception of Electromagnetic Radiation from RS-232 Cables. *Computers and Security*, 9(1):53–58, 1990.

[SSN08] Siamak Fayyaz Shahandashti and Reihaneh Safavi-Naini. Construction of Universal Designated-Verifier Signatures and Identity-Based Signatures from Standard Signatures. In Ronald Cramer, editor, *Public Key Cryptography – PKC '08: The 11th International Conference on Theory and Practice of Public Key Cryptography*, volume 4939 of *Lecture Notes in Computer Science*, pages 121–140, Barcelona, Spain, March 2008. Springer-Verlag.

[SSNB07] Siamak Fayyaz Shahandashti, Reihaneh Safavi-Naini, and Joonsang Baek. Concurrently-secure credential ownership proofs. In Feng Bao and Steven Miller, editors, *ASIACCS '07: Proceedings of the 2nd ACM symposium on Information, computer and communications security*, pages 161–172, Singapore, 2007. ACM Press.

[Sti91] Douglas R. Stinson. Universal Hashing and Authentication Codes. In Joan Feigenbaum, editor, *Advances in Cryptology – CRYPTO '91: The 11th Annual International Cryptology Conference*, volume 576 of *Lecture Notes in Computer Science*, pages 74–85, Santa Barbara, California, U.S.A., 1991. Springer-Verlag.

[Sti94] Douglas R. Stinson. Universal Hashing and Authentication Codes. In *Designs, Codes and Cryptography*, volume 4, pages 369–380, 1994.

[SWT01] Dawn Xiaodong Song, David Wagner, and Xuqing Tian. Timing analysis of keystrokes and timing attacks on SSH. In *SSYM'01: Proceedings of the 10th conference on USENIX Security Symposium*, pages 25–25, Berkeley, CA, USA, 2001. USENIX Association.

[Tan07] Hidema Tanaka. Information Leakage Via Electromagnetic Emanations and Evaluation of Tempest Countermeasures. In Patrick Drew McDaniel and Shyam K. Gupta, editors, *Information Systems Security, Third International Conference, ICISS 2007, Delhi, India, December 16-20, 2007, Proceedings*, volume 4812 of *Lecture Notes in Computer Science*, pages 167–179. Springer, 2007.

[TYH04] Shiang-Feng Tzeng, Cheng-Ying Yang, and Min-Shiang Hwang. A nonrepudiable threshold multi-proxy multi-signature scheme with shared verification. *Future Generation Comp. Syst.*, 20(5):887–893, 2004.

[VAN06] Jukka Valkonen, N. Asokan, and Kaisa Nyberg. Ad Hoc Security Association for Groups. In Levente Buttyán, Virgil D. Gligor, and Dirk Westhoff, editors, *Security and Privacy in Ad-Hoc and Sensor Networks, Third European Workshop, ESAS 2006*, volume 4357 of *Lecture Notes in Computer Science*, pages 150–164, Hamburg, Germany, September 2006. Springer-Verlag.

[Vau05a] Serge Vaudenay. On Bluetooth Repairing: Key Agreement based on Symmetric-Key Cryptography. In Dengguo Feng, Dongdai Lin, and Moti Yung,

editors, *Conference on Information Security and Cryptology, First SKLOIS Conference: CISC '05*, volume 3822 of *Lecture Notes in Computer Science*, pages 1–9, Beijing, China, December 2005. Springer-Verlag.

[Vau05b] Serge Vaudenay. Secure Communications over Insecure Channels Based On Short Authenticated Strings. In Victor Shoup, editor, *Advances in Cryptology – CRYPTO '05: The 25th Annual International Cryptology Conference*, volume 3621 of *Lecture Notes in Computer Science*, pages 309–326, Santa Barbara, California, U.S.A., August 2005. Springer-Verlag.

[Vau06] Serge Vaudenay. *A Classical Introduction to Cryptography: Applications for Communications Security*. Springer-Verlag, 2006.

[Vau07] Serge Vaudenay. E-Passport Threats. *IEEE Security and Privacy Magazine*, 5(6):61–64, 2007.

[vHP92] Eugène van Heyst and Torben P. Pedersen. How to make efficient fail-stop signatures. In *EUROCRYPT*, pages 366–377, 1992.

[VP09] Martin Vuagnoux and Sylvain Pasini. Compromising Electromagnetic Emanations of Wired and Wireless Keyboards. In *Proceedings of the 18th USENIX Security Symposium, August 10-14, 2009, Montreal, Canada*. USENIX Association, 2009.

[VV07] Serge Vaudenay and Martin Vuagnoux. About Machine-Readable Travel Documents. In *ICS '07*, Lecture Notes in Computer Science. Springer-Verlag, 2007.

[WBD05] Yongdong Wu, Feng Bao, and Robert H. Deng. Secure human communications based on biometrics signals. In Ryôichi Sasaki, Sihan Qing, Eiji Okamoto, and Hiroshi Yoshiura, editors, *Security and Privacy in the Age of Ubiquitous Computing, IFIP TC11 20th International Conference on Information Security (SEC 2005), May 30 - June 1, 2005, Chiba, Japan*, pages 205–221. Springer, 2005.

[WC81] Mark N. Wegman and Larry Carter. New Hash Functions and Their Use in Authentication and Set Equality. *Journal of Computer and System Sciences*, 22(3):265–279, 1981.

[WLF+05] Xiaoyun Wang, Xuejia Lai, Dengguo Feng, Hui Chen, and Xiuyuan Yu. Cryptanalysis of the hash functions MD4 and RIPEMD. In Ronald Cramer, editor, *Advances in Cryptology – EUROCRYPT '05: The 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques*, volume 3494 of *Lecture Notes in Computer Science*, pages 1–18, Aarhus, Denmark, 2005. Springer-Verlag.

[WSN08] Shuhong Wang and Reihaneh Safavi-Naini. New Results on Unconditionally Secure Multireceiver Manual Authentication. Cryptology ePrint Archive, Report 2008/039, 2008. http://eprint.iacr.org/.

[WUS06] Association Models Supplement to the Certified Wireless Universal Serial Bus Specification, 2006. http://www.usb.org/developers/wusb/wusb_2007_0214.zip.

[WY05] Xiaoyun Wang and Hongbo Yu. How to break MD5 and other hash functions. In Ronald Cramer, editor, *Advances in Cryptology – EUROCRYPT '05: The 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques*, volume 3494 of *Lecture Notes in Computer Science*, pages 19–35, Aarhus, Denmark, 2005. Springer-Verlag.

[WYY05a] Xiaoyun Wang, Yiqun Yin, and Hongbo Yu. Finding collisions in the full SHA1. In Victor Shoup, editor, *Advances in Cryptology – CRYPTO '05: The 25th Annual International Cryptology Conference*, volume 3621 of *Lecture Notes in Computer Science*, pages 17–36, Santa Barbara, California, U.S.A., August 2005. Springer-Verlag.

[WYY05b] Xiaoyun Wang, Xiuyuan Yu, and L. Y. Yin. Efficient collision search attacks on SHA-0. In Victor Shoup, editor, *Advances in Cryptology – CRYPTO '05: The 25th Annual International Cryptology Conference*, volume 3621 of *Lecture Notes in Computer Science*, pages 1–16, Santa Barbara, California, U.S.A., August 2005. Springer-Verlag.

[ZJC00] Philip Zimmermann, Alan Johnston, and Jon Callas. ZRTP: Media Path Key Agreement for Secure RTP draft-zimmermann-avt-zrtp-04, March 2000. http://www3.tools.ietf.org/html/draft-zimmermann-avt-zrtp-04.

[ZZT05] Li Zhuang, Feng Zhou, and J. D. Tygar. Keyboard acoustic emanations revisited. In Vijay Atluri, Catherine Meadows, and Ari Juels, editors, *Proceedings of the 12th ACM Conference on Computer and Communications Security, CCS 2005, Alexandria, VA, USA, November 7-11, 2005*, pages 373–382. ACM, 2005.

Glossary

Security models:

PKI	Public-key Infrastructure
CRS	Common Reference String (model)
ROM	Random Oracle Model

Hash functions:

CRHF	Collision Resistant Hash Function
WCR	Weakly Collision Resistant (hash function)
TCR	Target Collision Resistant (hash function)
UOWHF	Universal One-Way Hash Function
eTCR	enhanced Target Collision Resistant (hash function)

Message authentication protocols:

MAP	Message Authentication Protocol
UMAP	Unilateral Message Authentication Protocol
IMAP	Interactive (unilateral) Message Authentication Protocol
NIMAP	Non-Interactive (unilateral) Message Authentication Protocol
MMA	Message Mutual-Authentication
MCA	Message Cross-Authentication
GMA	Group Message Authentication
GMAP	Group Message Authentication Protocol

SAS-based protocols:

SAS	Short Authenticated String
SAS-MMA	SAS-based MMA
SAS-MCA	SAS-based MCA
SAS-GMA	SAS-based GMA

Key agreements:

KA	Key Agreement
DH	Diffie-Hellman (key agreement protocol)
AKA	Authenticated Key Agreement
SAS-AKA	SAS-based AKA
GKA	Group Key Agreement
BD	Burmester-Desmedt (group key agreement protocol)
SAS-GKA	SAS-based GKA

Our SAS-based protocols:

PV-NIMAP	The Pasini-Vaudenay NIMAP
PV-SAS-MMA	The Pasini-Vaudenay SAS-MMA (protocol)
PV-SAS-MCA	The Pasini-Vaudenay SAS-MCA (protocol)
PV-SAS-AKA	The Pasini-Vaudenay AKA (protocol)
LP-SAS-GMA	The Laur-Pasini SAS-GMA (protocol)
LP-SAS-GKA	The Laur-Pasini SAS-GKA (protocol)

Proofs of knowledge:

ITM	Interactive Turing Machine
ZK	Zero-Knowledge
HVZK	Honest Verifier Zero-Knowledge
dZK	deniable Zero-Knowledge
GQ	Guillou-Quisquater (Σ-protocol)

Signature schemes:

DS	Digital Signature
FML	Fixed Message Length
AML	Arbitrary Message Length
EF	Existential Forgery
UF	Universal Forgery
KMA	Known Message Attack
CMA	Chosen Message Attack
ONTAP	Online Non-Transferable Authentication Protocol
SRP	Signature with Randomized Precomputation
DSA	Digital Signature Algorithm

Miscellaneous:

MANET	Mobile Ad-hoc Network
PGP	Pretty Good Privacy
GPG	GNU Privacy Guard
SSH	Secure Shell
PIN	Personal Identification Number
WEP	Wired Equivalent Privacy
WPA	Wi-Fi Protected Access
WPS	Wi-Fi Protected Setup
WUSB	Wireless Universal Serial Bus

List of Figures

List of Definitions

List of Theorems

Curriculum Vitæ

Personal Situation

Sylvain Pasini
Born December 06, 1980, in Lancy, Geneva.
Swiss nationality, married, one child.
sylvain@famillepasini.ch

Education

PhD Degree in Cryptography 2005-2009
École Polytechnique Fédérale de Lausanne (EPFL)
Fellowship from the Swiss National Science Foundation (SNSF)
Supervisor: Prof. Serge Vaudenay
PhD thesis: Secure Communications Using Authenticated Channels

EPF Engineer in Communication Systems (Master degree) 2002-2005
École Polytechnique Fédérale de Lausanne (EPFL)
I&C Faculty, Communication Systems Department

HES Engineer in Electronics 1996-2001
École d'Ingénieurs de Genève (EIG)
Electronic Department

Mandatory School 1985-1996
Cycle d'orientation des Marais, Onex, Geneva (7-9P)
École du Petit-Lancy, Geneva (3-6P)
École Cérésole, Petit-Lancy, Geneva (1-2E, 1-2P)

Awards

Kudelski Prize rewarding a Master project contributing significantly to the field of cryptography and information system security. 2006

HES diploma decerned with 2002
Summa cum laude (mention très bien)
Best average grade of the EIG promotion 2002
ATG prize (Association des anciens élèves)
SIA prize (Société suisse des Ingénieurs et Architectes)
SIG prize (Services Industriels de Genève)
Charmilles Technologies prize

Academic Experience

Publications

- **Keyboard Compromising Electromagnetic Emanations** 2009
  Sylvain Pasini & Martin Vuagnoux.
  Published in the proceedings of USENIX Security '09 [Canada].
- **Efficient Deniable Authentication for Standard Signatures** 2009
  Jean Monnerat, Sylvain Pasini & Serge Vaudenay.
  Published in the proceedings of ACNS '09 [France].
- **User-Aided Data Authentication** 2009
  Sylvain Pasini & Sven Laur.
  Published in the International Journal of Security and Networks.
- **SAS-Based Group Authentication and Key Agreement Protocols** 2008
  Sylvain Pasini & Sven Laur.
  Published in the proceedings of PKC '08 [Spain].
- **Hash-and-Sign with Weak Hashing Made Secure** 2007
  Sylvain Pasini & Serge Vaudenay.
  Published in the proceedings of ACISP '07 [Australia].
- **SAS-based Authenticated Key Agreement** 2006
  Sylvain Pasini & Serge Vaudenay.
  Published in the proceedings of PKC '06 [U.S.A.].
- **An Optimal Non-interactive Message Authentication Protocol** 2006
  Sylvain Pasini & Serge Vaudenay.
  Published in the proceedings of CT-RSA'06 [U.S.A.].

Presentations

- Several oral **presentations** in conferences in U.S.A., Australia, and Europe.
- One **invited talk** at ENS-TA in Paris.

Teaching Activities

- **Scientific Projects Supervision**. 2005-2008
  Student projects supervision.
  Projects coordinator in the lab (semester and diploma).
- **Cryptography and Security Course**, 120+ students, EPFL. 2007 & 2008
  Dr. Philippe Oechslin & Prof. Serge Vaudenay.
  Teaching assistant.
  (exercise sessions, exams preparations/corrections, gradings).
- **Advanced Cryptography Course**, EPFL 2006 & 2007
  Prof. Serge Vaudenay.
  Teaching assistant.
  (exercise sessions, exams preparations/corrections, gradings).
- **(Real Time) Embedded Systems Course**, EPFL 2004 & 2005
  René Beuchat.
  Teaching assistant.
  (hardware preparation, assistant for lab sessions and projects).

Contribution to the Research World

- Involved as **external reviewer** for several prestigious conferences.
  (Crypto, Eurocrypt, Asiacrypt, Africacrypt, PKC, FSE, SAC, ACISP, ACNS, CHES, ICALP, IWSec)

Professional Experience

Technic-Hobby (Geneva), the shop of my parents Since 1987
Consultant in radio controlled models. Manager during holidays.

Neidhart & Team Orion (Geneva) 1998 & 1999
Batteries matcher, storekeeper and employee of production.

Cantonal stewardship of Geneva 1997
Storekeeper for schools of Geneva.

Relevant Projects

Keyboard Compomising Electromagnetic Emanations 2008
with Martin Vuagnoux.
We analyze the electromagnetic emanations emitted by computer keyboards. We present four different weaknesses on PS/2, USB, wireless, and laptop keyboards. Thanks to our practical implementation we were able to recover 95% of the keystrokes up to 20 meters.

Secure Communications over Insecure Channels [...] 2005
for which I received the **Kudelski Group Prize**.
Prof. Serge Vaudenay, Master thesis, EPFL.
We analyze the security of generic protocols and propose new improved solutions.

Why textbook ElGamal and RSA encryption are insecure? 2004
Prof. Serge Vaudenay, Semester Project, EPFL.
We implement the attacks to illustrate they are not secure without a pre-processing on the plaintext.

Ethernet/Internet embedded camera design 2002
René Beuchat, Semester Project, EPFL.
We build a Web server embedded in an FPGA to provide on the Internet images from a CMOS camera.

Tumours detection and segmentation of MRI images 2001
for which I received the **congratulations from the jury**.
Prof. Michel Kocher, Diploma, EIG.
We design an automatic method to detect and extract tumours of a MRI scan of the head.

Brain segmentation of MRI images 2001
Prof. Michel Kocher, Semester Project, EIG.
We design an automatic method to extract the brain of a MRI scan of the head. This is the first application of the Mathematic Morphology in 3D.

Computer Science Skills

Operating Systems	Linux, Mac OS, Windows
Programming Languages	Ada, C, C++, Java
Web development	HTML, XML, PHP, CSS, Javascript, JSP
Hardware design	VHDL, Quartus/SoPC, Leonardo Spectrum, ModelSim
Scientific Software	Maple, Matlab
Networking	Ethernet, TCP/IP and related protocols
Software Engineering	UML
Reporting Software	LaTeX, Microsoft Office

Languages

French:	mother tongue
English:	written and spoken
Italian:	spoken
German:	written

Hobbies

RC models	Radio controlled airplanes, helicopters and cars.	
	Designer, builder, and pilot of aerobatic airplanes.	
	Precision and freestyle aerobatics.	
	National team pilot for precision aerobatic (F3A).	2007
	Participation at the **F3A world championship** in Argentina.	2007
Music	Classical piano.	