Using Authenticated Channels To Achieve Secure Communications

Sylvain Pasini
joint work with Serge Vaudenay

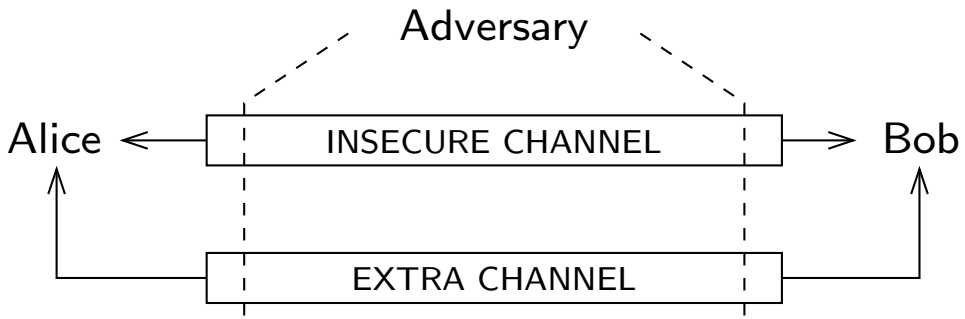
LASEC EPFL

CCA seminar, Paris

March the 30^{th} 2007

Setting up a Secure Communication

Suppose Alice and Bob want to communicate securely:



- No prior exchanged key (no PKI)
- Insecure channel:
  - Adversaries have full control
- Extra channel:
  - Additionnal assumptions (C, A, I, ...)

Possible Extra Channels

	Interactive		Non-interactive	
	Encounter	Telephone	Voice mail	Email
Authenticity	✓	✓	✓	
Confidentiality	✓			
Low cost		✓	✓	✓
Availability			✓	✓

Using symmetric cryptography, we need confidentiality:

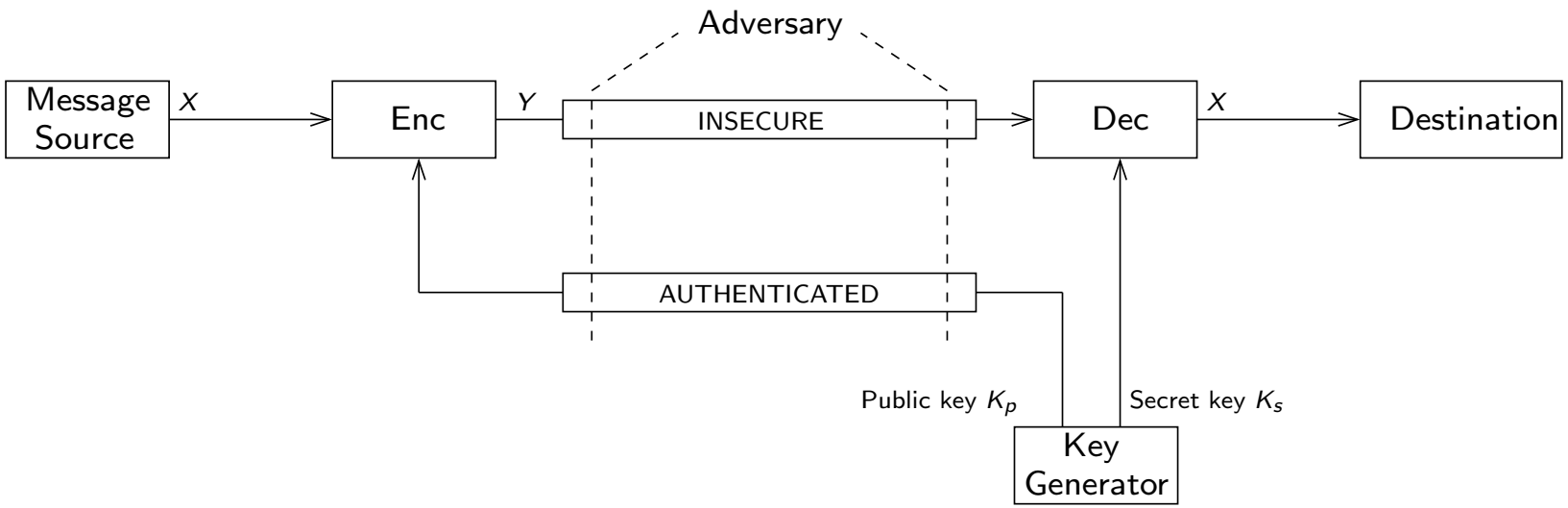
→ encounter.

Using public-key cryptography, we need authentication:

→ e.g. telephone or voice mail.

Exchanging a public-key...

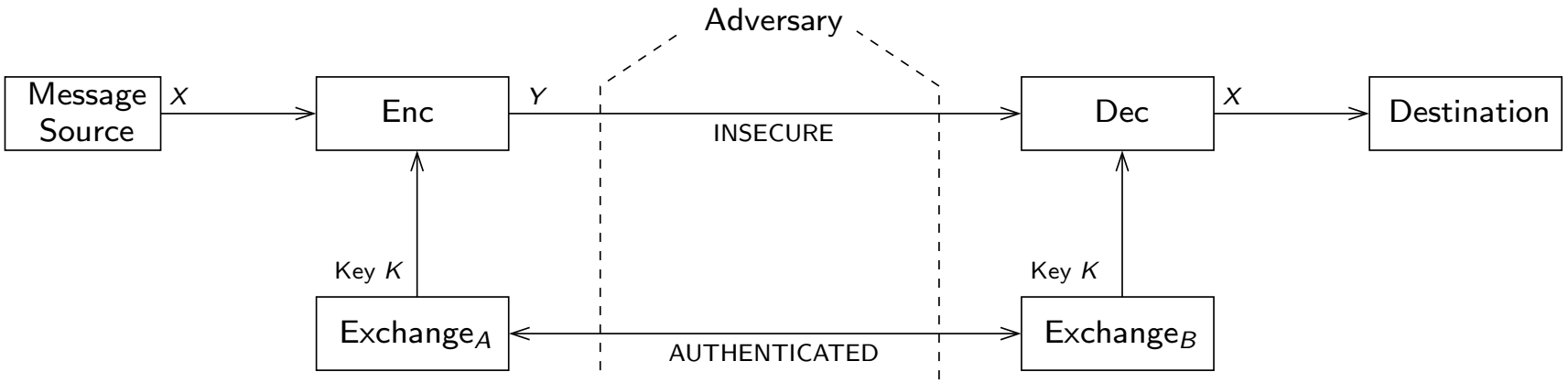
The semi-authenticated key transfer:



- E.g. the RSA cryptosystem.
- We no longer need confidentiality.
- An authenticated (extra) channel is enough.

Running a key agreement...

The Merkle-Diffie-Hellman model:



- E.g. the Diffie-Hellman protocol.
- We no longer need confidentiality.
- An authenticated (extra) channel is enough.

Secure Communication

In a nutshell, a secure communication can be setup by

- exchanging a public key (and authenticating)
- running an Authenticated Key Agreement (AKA)

Hypothesis:

- No prior exchanged key (no PKI).

Secure Communication

Example of an RSA 1024-bit key:

ssh-rsa AAAAB3NzaC1yc2EAAAABIwAAAIEApZTX
ilQgosFxevR9ewub/qE1/BoHXCkpzWwopTHkiY2e
8pMxMXOc/DzKV0qgsdC3X9pQODRy+awoANAgttPX
h6JM4ZlYgaEN6azJSyrK0SlOLDn+YmjjhaKEn1uf
LbroQ6Cpg0lj3lXvHEN52P32IfhY08ivC0pBmO4Y
eyErBiE=

Objective

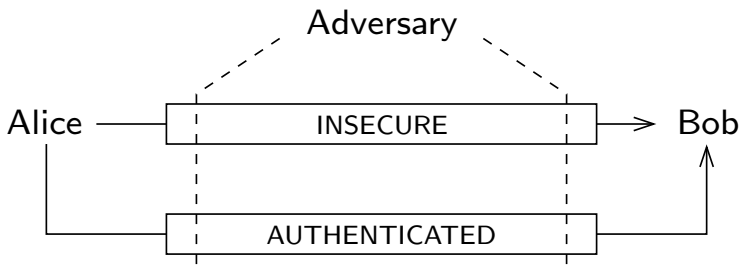
Do user-friendly protocols.
So, reduce the amount of authenticated data.

Authenticated Channels

How does a message authentication protocol work?

- χ It does not send the message over the authenticated channel
- ✓ It sends the message through the insecure channel
- ✓ The authentication is done by authenticating a shorter string

Channels model:



Authenticated channel:

- Authenticity and integrity are protected.
- Anyone can read, delay, remove, ...

Using a Fingerprint...

Authenticate a public key P_K by

$\longrightarrow$ checking its fingerprint

An example is the SSH protocol:

- ✓ Sends P_K through the insecure channel
- ✓ Authenticate $H(P_K)$

The fingerprint is of the form:

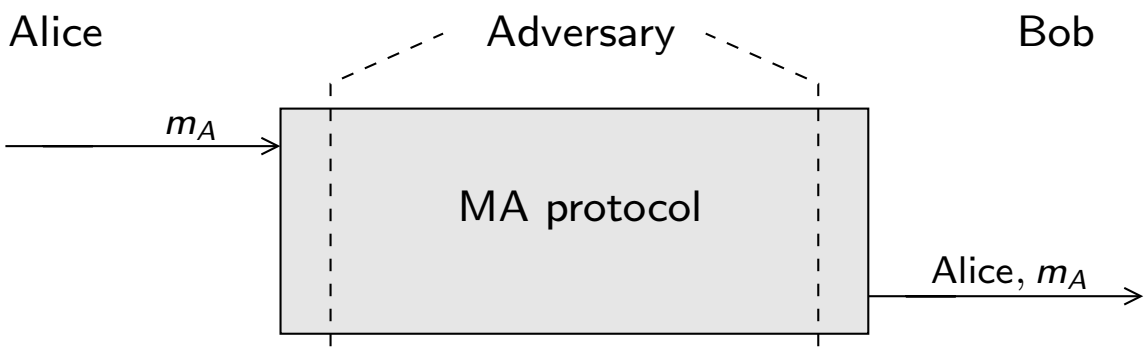
bc:a1:12:30:bc:17:08:eb:31:43:eb:e1:15:12:ca:1a

How many users really check this?

Overview

Message Authentication Protocol

- Alice authenticates her message to Bob (i.e. m_A).



Exchanging a Public-key

Today...

Today...

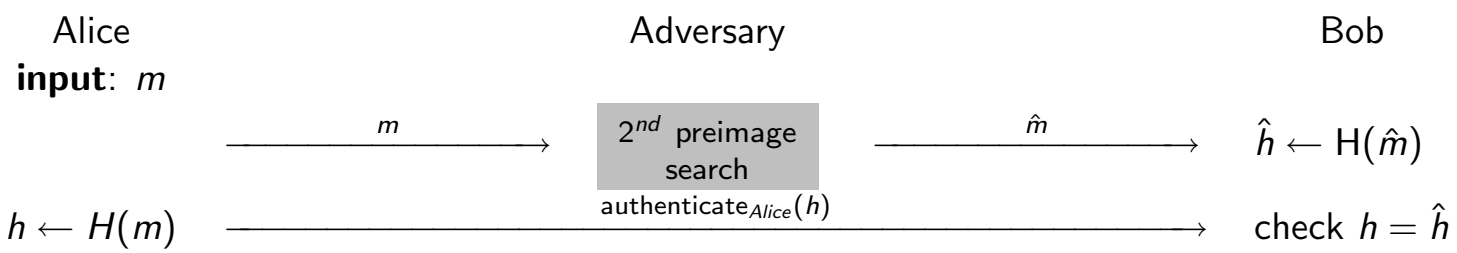
SSH and GPG use the following:

Alice		Bob
input: m		
	$\xrightarrow{m}$	$\hat{h} \leftarrow \mathrm{H}(\hat{m})$
$h \leftarrow H(m)$	$\xrightarrow{\text{authenticate}_{Alice}(h)}$	check $h = \hat{h}$
		output: $Alice, \hat{m}$

The symbol ˆ on a received message indicates that it may be different from the one originally sent.
(e.g. when an attack is performed)

What about Security?

Known message attack:



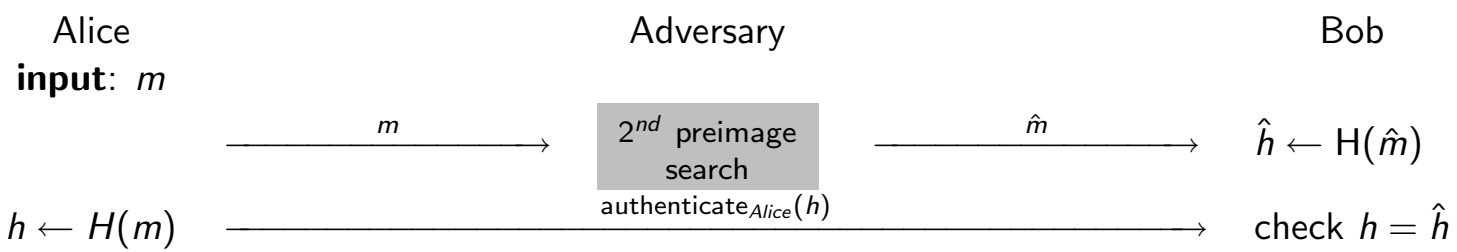
H **only** has to be weakly collision resistant (80 bits).

Chosen message attack:

H must be collision resistant (160 bits).

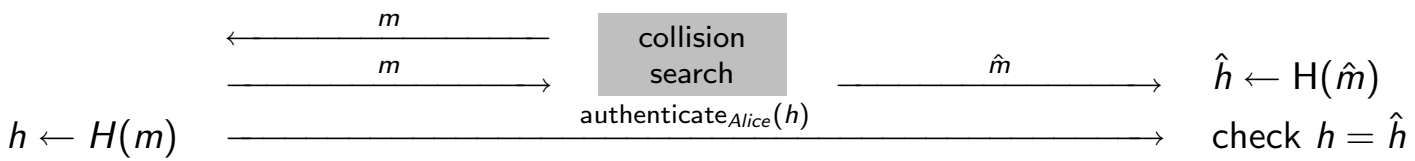
What about Security?

Known message attack:



H **only** has to be weakly collision resistant (80 bits).

Chosen message attack:



H must be collision resistant (160 bits).

What about Security?

Known message attack:

H **only** has to be weakly collision resistant (80 bits).

Chosen message attack:

H must be collision resistant (160 bits).


Exchanging a Public-key
A Generic Attack

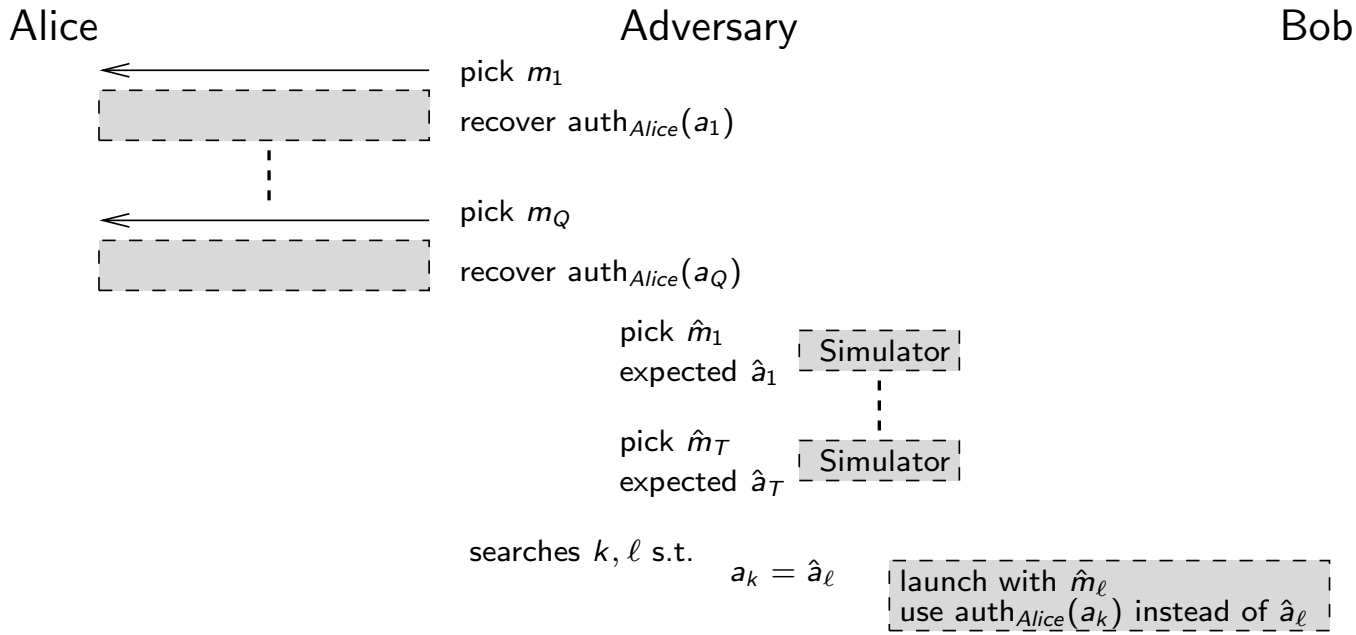
Generic Attack

Generic assumptions:

- Consider any message authentication protocol $\mathcal{P}$.
  - $\mathcal{P}$ is non-interactive.
  - $\mathcal{P}$ uses k authenticated bits.
- Consider an adversary $\mathcal{A}$ against $\mathcal{P}$.
  - $\mathcal{A}$ is a chosen message adversary.
  - $\mathcal{A}$ is limited to Q_A runs with Alice (online).
  - $\mathcal{A}$ is bounded by a time complexity T (offline).

Sketch

Instances of Bob can be simulated.



Success probability:

$$\Pr[\text{success}] \approx 1 - e^{-\frac{T \cdot Q_A}{2^k}}$$

Generic Attack

Theorem

For a non-interactive message authentication protocol which uses a weak authenticated channel, there exists a generic attack s.t.

$$\Pr[\text{success}] \approx 1 - e^{-\frac{T \cdot Q_A}{2^k}}$$

Consequence

No protocol can remain secure when
$T \cdot Q_A$ is non negligible against 2^k

Definition

If a protocol reaches this security level, it is **optimal**.

Generic Attack

Theorem

For a non-interactive message authentication protocol which uses a weak authenticated channel, there exists a generic attack s.t.

$$\Pr[\text{success}] \approx 1 - e^{-\frac{T \cdot Q_A}{2^k}}$$

Consequence

No protocol can remain secure when

$T \cdot Q_A$ is non negligible against 2^k

Definition

If a protocol reaches this security level, it is **optimal**.

Generic Attack

Theorem

For a non-interactive message authentication protocol which uses a weak authenticated channel, there exists a generic attack s.t.

$$\Pr[\text{success}] \approx 1 - e^{-\frac{T \cdot Q_A}{2^k}}$$

Consequence

No protocol can remain secure when

$T \cdot Q_A$ is non negligible against 2^k

Definition

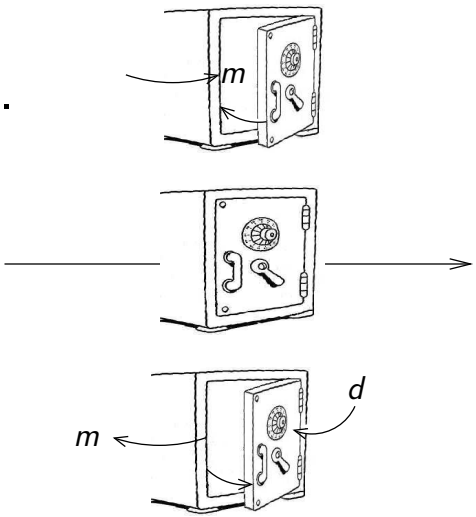
If a protocol reaches this security level, it is **optimal**.


Exchanging a Public-key
Recalls on Commitment Schemes

Commitment Schemes

A commitment is like a locked combination safe:

- When Alice wants to commit on m,
  she places m inside the safe and closes it.
- The safe is the commit object c,
  it can be given to Bob.
- When Alice wants to reveal m,
  she gives the combination d (decommit).

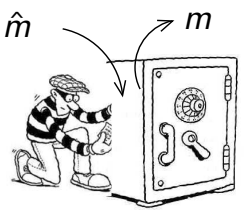


Hiding property:

m cannot be known before c is opened

Binding property:

m cannot be modified after c is closed



Commitment Schemes, More Formally

There are two algorithms:

- $(c, d) \leftarrow \textbf{commit}(m)$
- $m \leftarrow \textbf{open}(c, d)$

Completeness property: $\forall m, c, d$ where $(c, d) \leftarrow \text{commit}(m)$,

we always have $m = \text{open}(c, d)$.

Hiding property: $\forall m, c, d$ where $(c, d) \leftarrow \text{commit}(m)$,

Given c, it is impossible to deduce information about m.

Binding property: $\forall m, c, d$ where $(c, d) \leftarrow \text{commit}(m)$,

it is impossible to find d' s.t. $m' \leftarrow \text{open}(c, d')$ and $\hat{m} \neq m$.

Flipping a coin by telephone

Commitment scheme based on a hash function:

- commit(m)
  - picks a random r
  - outputs $c \leftarrow H(d)$ and $d \leftarrow (m \| r)$
- open(c, d)
  - checks that $c = H(d)$ and ouptuts m or $\perp$

The game:

Alice		Bob
choose $b \in \{head, tail\}$		
pick r, $d \leftarrow (b \| r)$, $c \leftarrow H(d)$	$\xrightarrow{c}$	
	$\xleftarrow{r}$	$r \leftarrow$ [flip the coin]
	$\xrightarrow{d}$	check $c = H(d)$


Exchanging a Public-key
An Improved Protocol

Overview

Main idea

Avoid the authenticated message to be predictable by adding randomness.

Given an input message m:

1. commit on m

   yield c and d (not deterministic).
2. reveal c and d.

   given (c, d), anyone can recover m (deterministic)
3. authenticate $H(c)$

   c is not foreseeable, thus $H(c)$ neither.

The Proposed Protocol [PV06a]

Alice		Bob
input: m		
$(c, d) \leftarrow \text{commit}(m)$	$\xrightarrow{c\|\|d}$	$\hat{m} \leftarrow \text{open}(\hat{c}, \hat{d})$
$h \leftarrow H(c)$	$\xrightarrow{\text{authenticate}_{Alice}(h)}$	check $h = H(\hat{c})$
		output: $Alice, \hat{m}$

Example using the commitment based on a hash function:

pick r		
$c \leftarrow H'(m\|\|r)$	$\xrightarrow{c\|\|m\|\|r}$	check $\hat{c} = H'(\hat{m}\|\|\hat{r})$
$h \leftarrow H(c)$	$\xrightarrow{\text{authenticate}_{Alice}(h)}$	check $h = H(\hat{c})$

Security

Overall Security

Consider an adversary bounded by complexity T and Q_A protocol runs with Alice.

He succeeds with probability at most $p \leq Q_A(\epsilon_c + \epsilon_h)$

by assuming a (T, ϵ_c)-binding commitment scheme and a (T, ϵ_h)-weakly collision resistant hash function.

Note that

- ϵ_c can be as small as desired (c sent over the broadband channel)
- h must be as short as possible (h sent over the authenticated channel)

Considering $T \approx 2^{80}$, $Q_A \approx 2^{20}$, we need at least $k = 100$.


Exchanging a Public-key
SAS-based protocol

(Interactive) SAS-based Protocol [Vau05]

Main idea

Avoid offline attacks by running an interactive protocol.

A 3-move MCA:

Alice		**Bob**
input: m		
pick $R_A \in_U \{0,1\}^k$		pick $R_B \in_U \{0,1\}^k$
$(c,d) \leftarrow \mathsf{commit}(m, R_A)$	$\xrightarrow{m,c}$	
	$\xleftarrow{R_B}$	
	$\xrightarrow{d}$	$\hat{R}_A \leftarrow \mathsf{open}(\hat{m}, \hat{c}, \hat{d})$
$\mathsf{SAS} \leftarrow R_A \oplus \hat{R}_B$	$\xleftrightarrow{\mathsf{authenticate}_{\mathrm{Alice}}(\mathsf{SAS})}$	$\mathsf{SAS} \leftarrow \hat{R}_A \oplus R_B$
		output: Alice, $\hat{m}$

Security

Overall Security

Consider an adversary bounded by complexity T and Q_A (Q_B) protocol runs with Alice (Bob).

He succeeds with probability at most $p \leq Q_A Q_B 2^{-k} + \epsilon$,

under reasonable assumptions on the commitment scheme.

To reach the same security level as credit cards: $k = 15$ is enough.

(i.e. 4-digit PIN, $Q_A = Q_B \leq 3$ and $p = 3 \cdot 10^{-4}$)

Applications

- Distant host authentication, e.g. SSH
- E-mail authentication, e.g. GPG signature
- Secure e-mail, e.g. GPG encryption
- Digital signature, e.g. RSA signature with MD5:

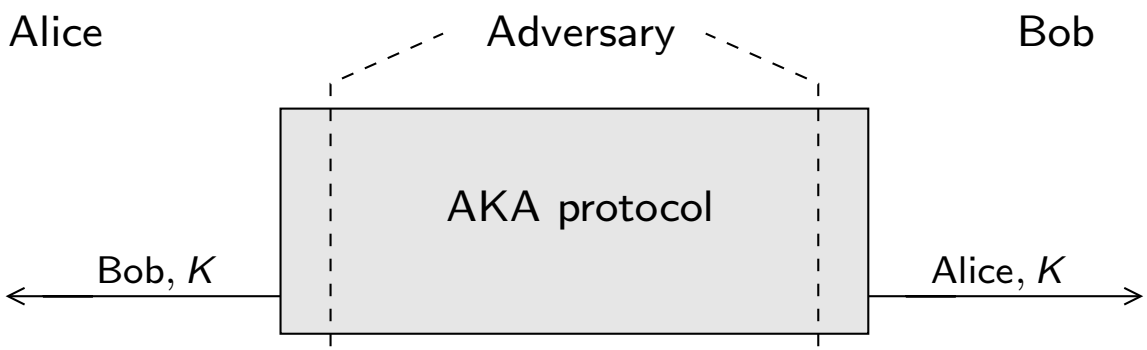
$$\mathrm{Sig}'(m) = c||d||\mathrm{Sig}(c)$$

Is interactivity possible for each?


Running a Key Agreement...

AKA as a building block

- Alice and Bob agree on a same key (i.e. K).



Running a Key Agreement...

Early Proposals

PGPfone Key Agreement Protocol (1995)

A 3-move AKA:

Alice		**Bob**
pick x_A, $y_A \leftarrow g^{x_A}$		pick x_B, $y_B \leftarrow g^{x_B}$
	$\xrightarrow{\text{commit}(y_A)}$	
	$\xleftarrow{y_B}$	
$z_A \leftarrow \hat{y}_B^{x_A}$	$\xrightarrow{\text{open}(y_A)}$	$z_B \leftarrow \hat{y}_A^{x_B}$
$\text{SAS} \leftarrow \text{truncH}(y_A \Vert \hat{y}_B)$	$\xrightarrow{\text{authenticate}_{\text{Alice}}(\text{SAS})}$	$\text{SAS} \stackrel{?}{=} \text{truncH}(\hat{y}_A \Vert y_B)$
check SAS is the same	$\xleftarrow{\text{authenticate}_{\text{Bob}}(\text{SAS})}$	
final state: Bob, z_A		**final state**: Alice, z_B

Problem: no security proof

Hoepman Key Agreement Protocol (2004)

A 4-move AKA:

Alice		**Bob**
pick x_A, $y_A \leftarrow g^{x_A}$	$\xrightarrow{\text{commit}(y_A)}$	pick x_B, $y_B \leftarrow g^{x_B}$
	$\xleftarrow{\text{commit}(y_B)}$	
$\text{SAS}_A \leftarrow \text{pieceof}(y_A)$	$\xrightarrow{\text{authenticate}_{\text{Alice}}(\text{SAS}_A)}$	
	$\xleftarrow{\text{authenticate}_{\text{Bob}}(\text{SAS}_B)}$	$\text{SAS}_B \leftarrow \text{pieceof}(y_B)$
	$\xrightarrow{\text{open}(y_A)}$	check commit, SAS
check commit, SAS	$\xleftarrow{\text{open}(y_B)}$	
$z_A \leftarrow (\hat{y}_B)^{x_A}$		$z_B \leftarrow (\hat{y}_A)^{x_B}$
final state: Bob, z_A		**final state**: Alice, z_B

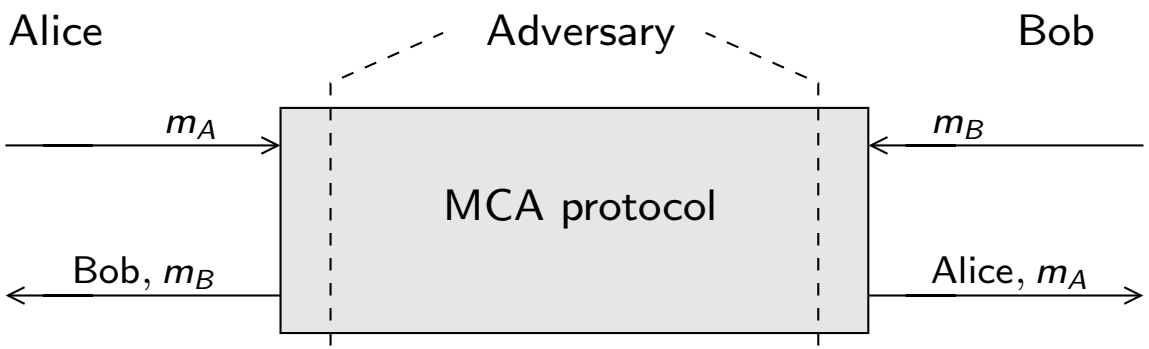
Problem: no security proof

Running a Key Agreement...

Key Agreement to Message Authentication

Message Cross-Authentication (MCA) Protocol

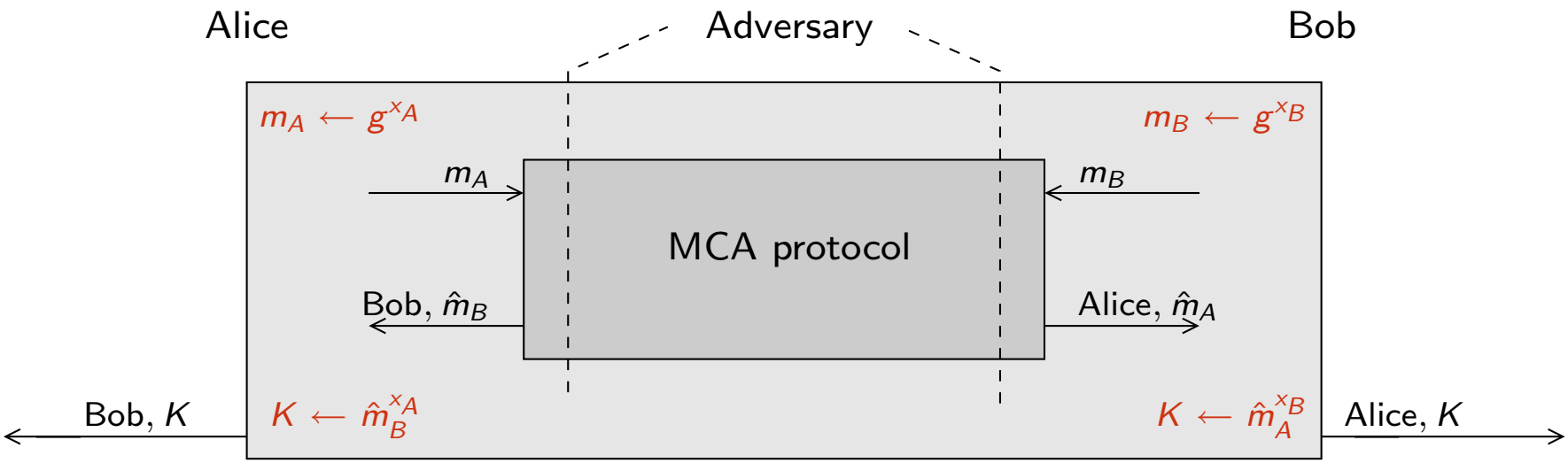
- Alice and Bob authenticate their message (i.e. m_A, m_B).



An Example of Construction

We can build an AKA in two steps

1. Both users run a MCA protocol.
2. Input messages are DH values g^{x_A}, g^{x_B}.



A Generic Construction [PV06b]

1. Run an AKA_0 (over the insecure channel).
2. Run a MCA,
   - input messages are the view of the protocol transcript
   - AKA_0's last message is send with the MCA's first (each user saves a move)

Theorem

Consider:

- an n_k-move (T, ε_k)-AKA_0 protocol
- an n_a-move (T, ε_a)-MCA protocol

There exists a μ s.t. the generic construction is an

$$\max(n_k, n_k + n_a - 2)\text{-move } (T\mu, \varepsilon_k + \varepsilon_a)\text{-AKA protocol.}$$

SAS-based MCA protocol [Vau05]

A 4-move MCA:

Alice		**Bob**
input: m_A		**input**: m_B
pick $R_A \in_U \{0,1\}^k$		pick $R_B \in_U \{0,1\}^k$
$(c_A, d_A) \leftarrow \text{commit}(0\|\|m_A, R_A)$	$\xrightarrow{m_A, c_A}$	
	$\xleftarrow{m_B, c_B}$	$(c_B, d_B) \leftarrow \text{commit}(1\|\|m_B, R_B)$
	$\xrightarrow{d_A}$	$\hat{R}_A \leftarrow \text{open}(0\|\|\hat{m}_A, \hat{c}_A, \hat{d}_A)$
$\hat{R}_B \leftarrow \text{open}(1\|\|\hat{m}_B, \hat{c}_B, \hat{d}_B)$	$\xleftarrow{d_B}$	
$\text{SAS} \leftarrow R_A \oplus \hat{R}_B$	$\xrightarrow{\text{authenticate}_{\text{Alice}}(\text{SAS})}$	$\text{SAS} \stackrel{?}{=} \hat{R}_A \oplus R_B$
check SAS is the same	$\xleftarrow{\text{authenticate}_{\text{Bob}}(\text{SAS})}$	
output: Bob, $\hat{m}_B$		**output**: Alice, $\hat{m}_A$


Running a Key Agreement...
A New MCA Protocol

A New SAS-Based MCA Protocol [PV06b]

Our new 3-move MCA:

Alice		**Bob**
input: m_A		**input**: m_B
pick $K \in_U \{0,1\}^\kappa$		pick $R \in_U \{0,1\}^\rho$
$(c,d) \leftarrow \text{commit}(m_A, K)$	$\xrightarrow{m_A, c}$	
	$\xleftarrow{m_B, R}$	
	$\xrightarrow{d}$	$\hat{K} \leftarrow \text{open}(\hat{m}_A, \hat{c}, \hat{d})$
$\text{SAS} \leftarrow \hat{R} \oplus h_K(\hat{m}_B)$	$\xrightarrow{\text{authenticate}_{\text{Alice}}(\text{SAS})}$	$\text{SAS} \stackrel{?}{=} R \oplus h_{\hat{K}}(m_B)$
check SAS is the same	$\xleftarrow{\text{authenticate}_{\text{Bob}}(\text{SAS})}$	
check Alice $\neq$ Bob		
output: Bob, $\hat{m}_B$		**output**: Alice, $\hat{m}_A$

We have a 3-move AKA (e.g. using DH).

Our New SAS-Based MCA Protocol

Overall Security

Consider:

- an ε-random oracle commitment scheme
- an ε_h-universal hash function family h

Any adversary bounded by Q instances of Alice or Bob cannot succeed with probability higher than

$$Q^2(2^{-\rho} + \varepsilon + \varepsilon_h).$$

A Typical Application

Secure VoIP:

- **Start** in an **insecure** mode.
  - The channel is authenticated (not confidential).
- Run a DH through our SAS-based MCA.
  Now,
  - both share a secret key K (thanks to DH)
  - K is cross-authenticated (thanks to our MCA)
- **Switch** to a **secure** conversation.
  - Use K with standard cryptography, e.g. AES.
  - The channel is now authenticated and confidential!

Running a Key Agreement...

Group settings

A Group SAS-Based Protocol...

Context:

- Consider n participants, $n \geq 2$.
- How to setup a secure communications between them?
- Example: Conference VoIP

Group Key Agreement:

- There exist protocols extending DH to groups.
- As for DH, they use authenticated channels.

Missing primitive:

- There do not exist Group MCA protocol.
- Ongoing work...

Summary of our results

Today, authentication is done with 128-160 bits.

A new **non-interactive** protocol which

- uses a commitment scheme.
- is optimal and requires only 80-100 bits.

A new **interactive** SAS-based MCA protocol which

- uses a commitment scheme.
- is optimal and requires only 15-40 bits.

Now, we have a **3-move SAS-based AKA** which uses an authenticated channel.

Future work: Group settings.