Pourquoi les versions théoriques d'ElGamal et de RSA ne sont pas sûres ?

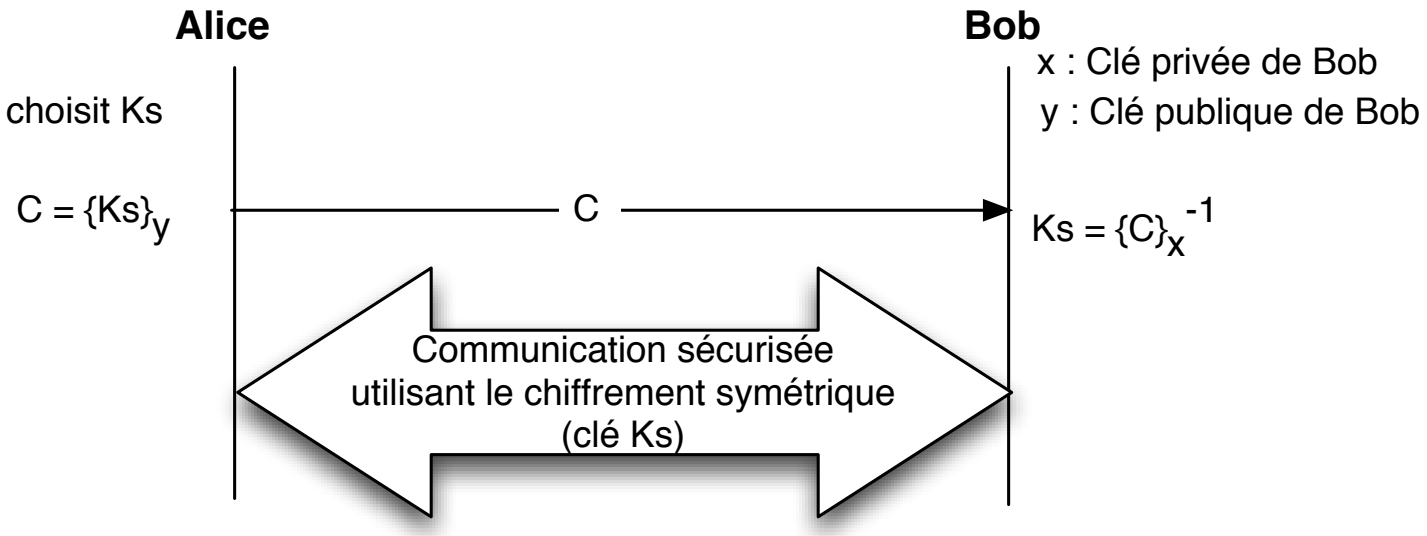
Sylvain Pasini

EPFL / LASEC

le 8 février 2005

Introduction

- Montrer l'insécurité des versions théoriques d'ElGamal et de RSA
- Application : Chiffrement hybride



- Objectif : clés DES

Basé sur la publication de Dan Boneh, Antoine Joux et Phong Q. Nguyen (AsiaCrypt 2000)

Aperçu

ElGamal en bref

- Sécurité repose sur la difficulté du logarithme discret
- Particularité : chiffrement non déterministe

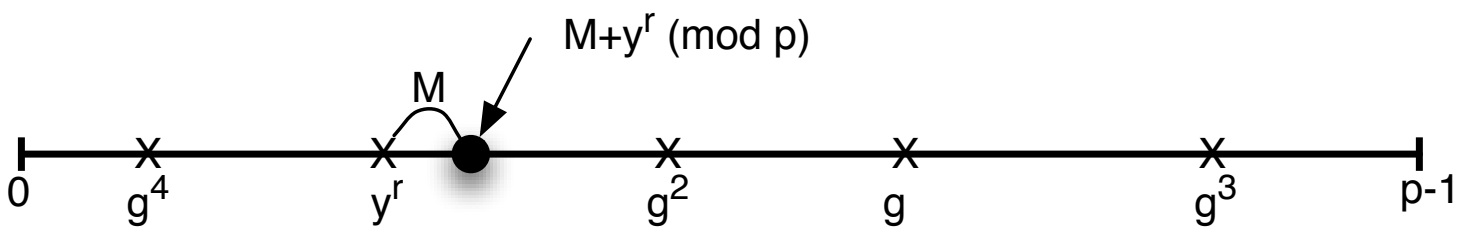
Paramètres publiques :	p, un grand nombre premier
	g, un générateur de $\mathbb{Z}_p^*$
Initialisation :	générer aléatoirement $x \in \mathbb{Z}_{p-1}$
	$y = g^x \bmod p$
Clé secrète :	$K_s = x$
Clé publique :	$K_p = y$
Message :	M, un élément de $\mathbb{Z}_p^*$
Chiffrement :	génère aléatoirement $r \in \mathbb{Z}_{p-1}$
	$C = \langle u = g^r mod\ p, v = M \cdot y^r \bmod p \rangle$
Déchiffrement :	$M = vu^{-x}$

Le problème des arrondis dans un sous-groupe

- Supposons : $g \in \mathbb{Z}_p^*$
  - avec q l'ordre de g, et $q \ll p$
- Soit G_q le sous-groupe de $\mathbb{Z}_p^*$ généré par g
- Maintenant, modifions le chiffrement :

$$r \in \mathbb{Z}_q^*, u = g^r \text{ et } v = M + y^r \bmod p$$

- Intuitivement :



Le problème des arrondis dans un sous-groupe(2)

Les additions arrondies dans un sous-groupe

Soit $z \in G_q$ et soit un message M.
Connaissant $u = z + M \bmod p$, trouver z.

Supposant G_q distribué uniformément dans $\mathbb{Z}_p^*$:
si $M < \frac{p}{q}$, z est unique avec une grande probabilité .

Le problème des arrondis dans un sous-groupe(3)

Les multiplications arrondies dans un sous-groupe

Soit $z \in G_q$ et soit un message M.
Connaissant $u = z \cdot M \bmod p$, trouver z.

Supposant G_q distribué uniformément dans $\mathbb{Z}_p^*$:
si $M < \frac{p}{q}$, z est unique avec une grande probabilité .

- Une solution performante à ce problème impliquerais que plain ElGamal ne serait pas sûre.

Algorithmes pour les multiplications arrondies dans un sous-groupe

- Supposons que $M = M_1 \cdot M_2$ avec $M_i \leq 2^{m_i}$
- On peut écrire :

$$v = y^r \cdot M_1 \cdot M_2 \bmod p$$

$$(v/M_2)^q = y^{rq} \cdot M_1^q = M_1^q \bmod p$$

- Attaque du type *meet-in-the-middle*
  - Mémoire utilisée = $2^{m_1} \log_2(p)$ bits
  - Temps en $O(2^{m_1} + 2^{m_2})$ exponentielles modulaires
- Pour DES : 16 GB !

Réduction à un problème du type sac à dos

- On suppose que $p - 1 = q \cdot r \cdot s$

  où $s \geq 2^m$ tel que le logarithme discret n'est pas difficile dans le sous-groupe G_s
- Utiliser Pohlig-Hellman pour trouver $\log(x^{qr})$ où $x \in \mathbb{Z}_p^*$
  - nécessite la factorisation de s
- Utiliser Pollard p-1 pour trouver s et sa factorisation

Réduction à un problème du type sac à dos(2)

- On peut écrire :

$$v = z \cdot M_1 \cdot M_2 \bmod p$$

$$v^{qr} = M_1^{qr} \cdot M_2^{qr} \bmod p$$

- Ici, le logarithme est pas difficile

$$\log(v^{qr}) = \log(M_1^{qr}) + \log(M_2^{qr}) \bmod s$$

- Identification d'un problème du type sac à dos

Résolution du sac à dos

En entrée : La cible n ainsi que les deux tables T_1 et T_2
En sortie : Toutes les solutions du type $n = t_1 + t_2$
avec $t_1 \in T_1$ et $t_2 \in T_2$
Algorithme :

Trier T_1 de manière croissante

Trier T_2 de manière décroissante

Répéter jusqu'à que T_1 ou T_2 soit vide :

Calculer $t = first(T_1) + first(T_2)$

Selon t :

Si $t = n$, afficher la solution trouvée
supprimer $first(T_1)$ et $first(T_2)$

Si $t < n$, supprimer $first(T_1)$ de T_1

Si $t > n$, supprimer $first(T_2)$ de T_2

Résolution du sac à dos, exemple

- Groupe est $\mathbb{Z}_{11}^*$
- Cible = 7
- Deux tables de 4 éléments

Itération 1 T1 = 10, 7, 5, 3
T2 = 2, 3, 4, 8
10+2$>$7

Itération 2 T1 = 7, 5, 3
T2 = 2, 3, 4, 8
7+2$>$7

Itération 3 T1 = 5, 3
T2 = 2, 3, 4, 8
5+2$=$7 $\rightarrow$ solution

Itération 4 T1 = 3
T2 = 3, 4, 8
3+3$<$7

Itération 5 T1 = 3
T2 = 4, 8
3+4$=$7 $\rightarrow$ solution

Itération 6 T1 = $\emptyset$
T2 = 8
T1 vide $\rightarrow$ stop

Résolution du sac à dos, 3 tables

- Trois tables ?

$$\log_{W^{qr}}(v^{qr}) = \sum_{i=1}^{3} \log_{W^{qr}}(M_i^{qr}) \bmod s$$
$$\log_{W^{qr}}(v^{qr}) - t_1 = t_2 + t_3 \bmod s$$

En entrée : La cible n ainsi que les trois tables T_1, T_2 et T_3
En sortie : Toutes les solutions du type $n = t_1 + t_2 + t_3$ avec $t_i \in T_i \ \forall i = 1..3$
Algorithme :

Trier T_2 de manière croissante

Trier T_3 de manière décroissante

Pour chacun des $t_i \in T_1$

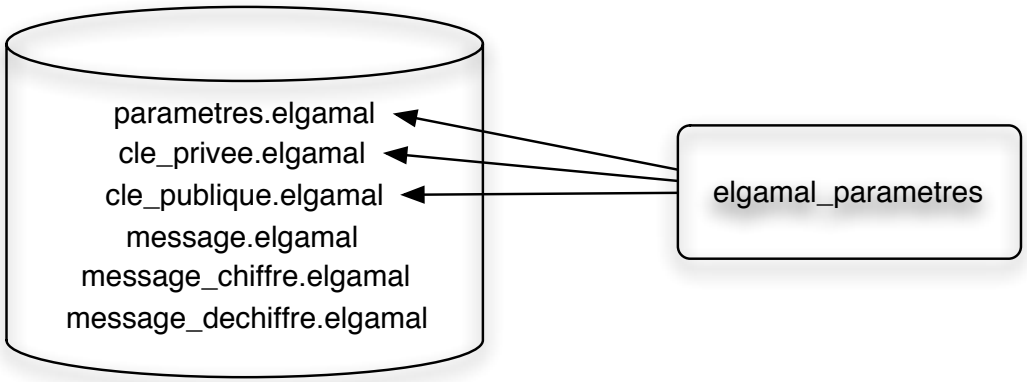
La sous-cible vaut $n' = n - t_i$
Résoudre le problème à deux tables, T_2 et T_3, en utilisant comme cible n'.

Résolution du sac à dos, comparaison

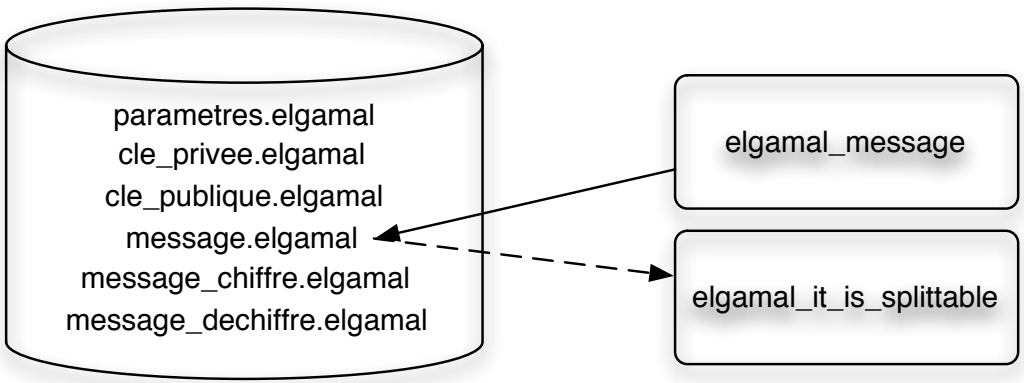
	2 tables	3 tables
mémoire utiliée *(éléments de $\mathbb{Z}_p^*$)*	$O(2 \cdot 2^{m/2})$	$O(3 \cdot 2^{m/3})$
temps remplissage des tables *(logarithmes discrets)*	$O(2^{m/2})$	$O(2^{m/3})$
temps résolution *(additions modulaire)*	$O(2^{m/2})$	$O(2^{\frac{2}{3}m})$

Structure des applications

Génération des paramètres et des clés

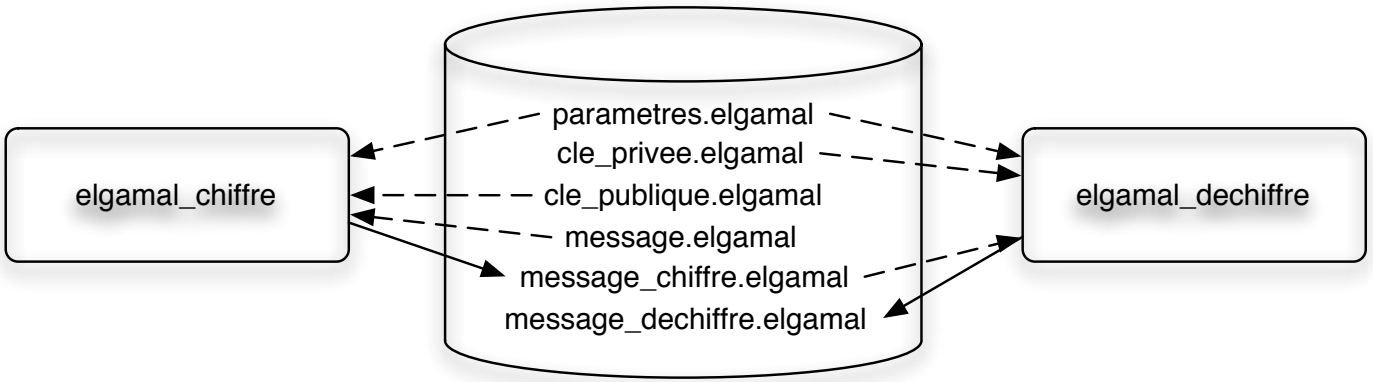


Génération du message

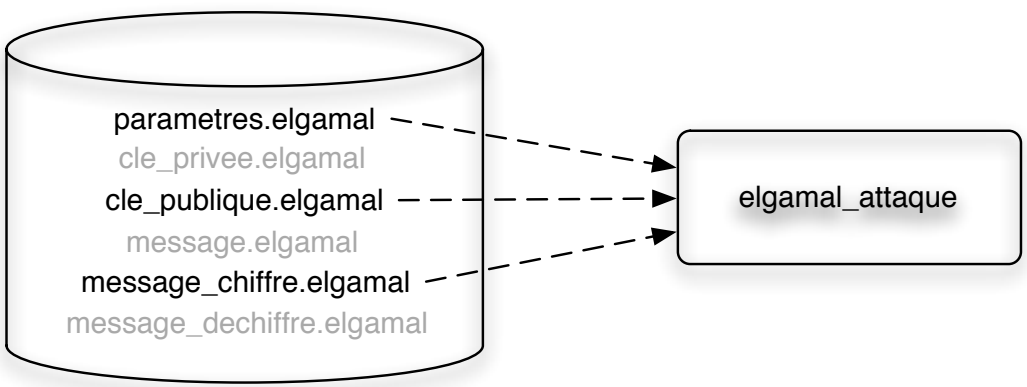


Structure des applications(2)

Chiffrement - Déchiffrement



Attaque



Vue globale de l'attaque

Prétraitement :

1. Lecture des paramètres p et g
2. Factoriser partiellement l'ordre du groupe, donc $p-1$
3. Extraire s
4. Remplir les tables avec $\log(M^{qr}), \forall M = 0, ..., 2^{\max(m_1,m_2,m_3)} - 1$
5. Trier les tables

Dépend uniquement des paramètres
Utilisable pour plusieurs messages

Vue globale de l'attaque(2)

Récupération d'un message :

1. Lecture du message chiffré $\langle u, v \rangle$
2. Déterminer la cible du problème du sac à dos
3. Résoudre le problème du sac à dos
4. Rechercher les M_i correspondants aux $t_i \in T_i$
5. Afficher les candidats du message

RSA en bref

Paramètres publiques :	s, un entier
Set up :	générer 2 nombres premiers aléatoires
	p et q différents de $s/2$ bits
	$N = pq$, donc $\phi(N) = (p-1)(q-1)$
	e, aléatoire tel que $\gcd(e, \phi(N)) = 1$
	$d = e^{-1} \bmod ((p-1)(q-1))$
Clé secrète :	$K_s = (d, N)$
Clé publique :	$K_p = (e, N)$
Message :	M, un élément de $\mathbb{Z}_N^*$
Chiffrement :	$C = M^e \bmod N$
Déchiffrement :	$M = C^d \bmod N$

L'attaque du type Meet-in-the-middle

- Supposons que $M = M_1 \cdot M_2$ avec $M_i \leq 2^{m_i}$
- Alors :

  $C = (M_1 \cdot M_2)^e \bmod N$

  $\frac{C}{M_2^e} = M_1^e \bmod N$
- L'attaque :

  Créer la table T contenant $M_1^e \bmod N$, $\forall M_1$

  $\forall M_2$ vérifier si $\frac{C}{M_2^e} \bmod N \in T$

  Chaque collision révèle une couple candidat

Probabilités de réussite

- Si message découpable, alors attaquable
- Possible de savoir si un message est découpable :

En entrée : Le message M, les paramètres m_1 et m_2 bits.
En sortie : Découpable ou non
Algorithme :

Factoriser M

Pour toutes les répartitions possibles des facteurs premiers entre les 2 parties de M :

Si cette répartition est une solution

Afficher : découpable

Si aucune solution n'a été trouvée alors :

Afficher : non découpable

- Quelle probabilité ?

Probabilités de réussite(2)

m	m_1	m_2	P(découpable)
40	20	20	11%
	21	21	32%
	22	22	38%
	20	22	36%
	20	25	53%
56	28	28	9%
	29	29	27%
	30	30	33%
	28	30	33%
64	32	32	10%
	33	33	23%
	34	34	32%
	30	36	35%

m	m_1	m_2	m_3	P(découpable)
40	13	13	14	1%
	14	14	14	4%
	13	14	15	7%
56	18	19	19	1%
	19	19	19	3%
	20	20	20	5%
64	22	22	22	3%
	23	23	23	5%
	24	24	24	9%
	25	25	25	12%

Influence de la taille du message, ElGamal

- Voir l'influence de la taille du message M
- p de 512 bits, dont un grand facteur premier de 160 bits

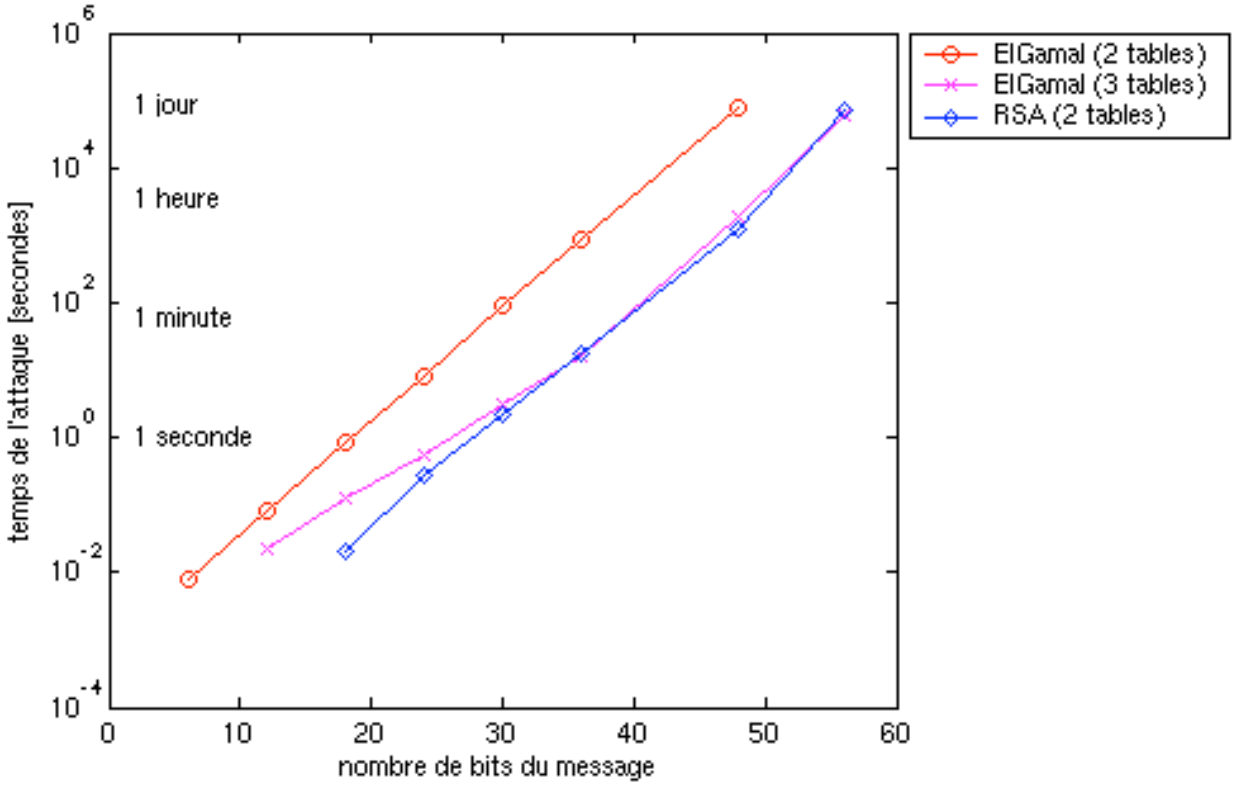
m	temps d'attaque avec 2 tables	P(réussite) avec 2 tables	temps d'attaque avec 3 tables	P(réussite) avec 3 tables
6	8 ms	14%	-	0%
12	80ms	14%	20ms	1.5%
18	800ms	14%	50ms	1.0%
24	8s	13%	500ms	1.0%
30	1m 30s	12%	3s	1.0%
36	13m	10%	16s	0.5%
42	2h	10%	2m50s	0.5%
48	22h	9%	30m	0.5%
56	*long...*	9%	16h	0.5%

Influence de la taille du message, RSA

- Voir l'influence de la taille du message M
- N de 512 bits

m	temps d'attaque	P(réussite)
18	20ms	14%
24	270ms	13%
30	2s	12%
36	18s	10%
48	21m	9%
56	19h	9%

Influence de la taille du message



Etude de l'attaque

- Analyser les parties coûteuses de l'attaque sur ElGamal
- Message de 42 bits

		2 tables	3 tables
1.	Lecture des paramètres p et g	< 1s	< 1s
2.	Factoriser partiellement $p - 1$	≈ 1s	≈ 1s
3.	Extraire s	< 1s	< 1s
4.	Remplir les tables T_i	62m	43s
5.	Trier les tables	7s	< 1s
6.	Lecture du message chiffré $\langle u, v \rangle$	< 1s	< 1s
7.	Déterminer la cible	< 1s	< 1s
8.	Résoudre le problème du sac à dos	≈ 1s	84s
9.	Rechercher les M_i correspondants aux $t_i \in T_i$	55m	42s
10.	Afficher les résultats	< 1s	< 1s
	Au total :	1h57	2m50s

Conclusion

- Réussi à retrouver une clé DES chiffrée avec plain ElGamal ou plain RSA en moins de 20 heures
- Versions théoriques pas sûre dans certains cas
- Montre l'importance du prétraitement
- Motivation pour utiliser des standards, tel que RSA-OAEP

Démonstration

- A partir d'un nombre (message)
- Possibilité de savoir s'il est attaquable :
  - Si oui : résultats
  - Si non : que va faire l'attaque ?

Merci
de votre attention