Cryptographie
Gestion des clés

Carlos Aguilar
carlos.aguilar@enseeiht.fr

IRIT-IRT

Références

Livres électroniques

Cornell, https://www.cs.cornell.edu/courses/cs4830/2010fa/lecnotes.pdf
Bristol, http://www.cs.umd.edu/~waa/414-F11/IntroToCrypto.pdf
Stanford, http://crypto.stanford.edu/~dabo/cryptobook/

Cours fortement inspiré de :

[1] Stanford, https://www.coursera.org/course/crypto
[2] Univ. of Virginia, http://www.udacity.com/view#Course/cs387/

Plan

Plan

La gestion des clés dans un monde symétrique

Deux personnes

S'ils partagent un secret ok, sinon ils doivent l'échanger par un canal sûr

Un groupe de n personnes (fixe)

Un secret partagé : dangereux
Un secret par couple d'utilisateurs : n^2 secrets

Un groupe dynamique

Procédure de révocation de clés, systèmes de trousseaux, etc.

Et toujours un besoin de canal sécurisé

Utilisation d'une tierce partie de confiance

Principe

- Toute nouvelle personne s'enregistre auprès de la tierce partie de confiance TTP et échange par un canal sécurisé une clé unique
- Toute communication avec TTP est sécurisée (confidentialité + intégrité) par cette clé
- Quand un utilisateur A veut communiquer avec B elle demande à TTP de transférer un secret k_{AB} à B

Défauts

- Confiance de tout le monde dans une même entité
- Besoin constant d'accéder à la TTP même quand A et B sont à portée de communication directe
- Single point of failure ?

Utilisation d'une tierce partie de confiance

Principe

- Toute nouvelle personne s'enregistre auprès de la tierce partie de confiance TTP et échange par un canal sécurisé une clé unique
- Toute communication avec TTP est sécurisée (confidentialité + intégrité) par cette clé
- Quand un utilisateur A veut communiquer avec B elle demande à TTP de transférer un secret k_{AB} à B

Défauts

- Confiance de tout le monde dans une même entité
- Besoin constant d'accéder à la TTP même quand A et B sont à portée de communication directe
- Single point of failure ? pas nécessairement

Plan

Note historique

Les puzzles de Merkle

1974, premier à proposer une solution
Basée intégralement sur le chiffrement symétrique

Principe

- A envoie 2^{k_1} puzzles $Enc(wk_i, id_i||sk_i)$, wk_i ayant k_2 bits aléatoires, id_i, sk_i aléatoires de $k_1 + k_2$ bits
- B choisi un au hasard tente 2^{k_2} clés et renvoie id_i
- A et B utilisent sk_i pour chiffrer/authentifier leurs communications

Bien tenté !

Coûts : pour l'émetteur 2^{k_1}, pour le récepteur 2^{k_2}
Pour l'attaquant $2^{k_1+k_2}$
Pas vraiment sûr mais purement basé sur la crypto existante

Diffie-Hellman (1976)

New directions in cryptography

Schéma permettant d'échanger une clé secrète avec un coût polynomial pour les utilisateurs et un coût exponentiel pour un attaquant

Résistant aux observateurs passifs mais pas à ceux actifs

Bases

p un nombre premier de grande taille (typiquement plus de 1024 bits) de la forme $p = 1 + K * q$ pour q premier (typiquement de plus de 100 bits).
On note $\mathbb{Z}_p$ (ou $\mathbb{Z}/p\mathbb{Z}$) l'ensemble $\{0, \ldots, p-1\}$ muni de l'addition et de la multiplication réduites modulo p (simplification)

Il existe (et on peut trouver facilement) un élément g de $\mathbb{Z}_p$ générant q éléments distincts : c'est-à-dire $g \in \{0, \ldots, p-1\}$ tq $\{g \bmod p, g^2 \bmod p, \ldots, g^q \bmod p\}$ soit de taille q

Diffie-Hellman (1976) : Principe

Paramètres publiques

A et B se mettent d'accord (e.g. utilisent un standard) sur p et g

Échange

- A tire $priv_A \leftarrow \{1, \ldots, q\}$ et envoie $pub_A = g^{priv_A} \bmod p$ à B
- B tire $priv_B \leftarrow \{1, \ldots, q\}$ et envoie $pub_B = g^{priv_B} \bmod p$ à A

Calcul du secret

- A calcule $sk_A = pub_B^{priv_A} \bmod p$
- B calcule $sk_B = pub_A^{priv_B} \bmod p$

Questions

Montrer que $sk_A = sk_B$. Temps de calcul en $N = \log_2 p$?

Le logarithme discret (1/2)

Un petit problème

p premier g générateur de $\mathbb{Z}_p$

Log discret : À partir de y trouver x tq $g^x \bmod p = y$
Permet de casser Diffie-Hellman

Fonction L pour la complexité

$L_p[\alpha, c] = 2^{(c+o(1))(\log p)^{\alpha}(\log\log p)^{1-\alpha}}$

*simplification : formule standard avec $e^{\cdots}$ et log népérien

Permet d'étalonner les coûts entre polynomial et exponentiel en $\log p$
$\alpha = 0 \rightarrow L_p[0, c] = (\log p)^{(c+o(1))}$
$\alpha = 1 \rightarrow L_p[1, c] = 2^{(c+o(1))\log p}$

Meilleur algo

Crible algébrique (number field sieve) pour le log discret
$L_p[1/3, 1.92] = 2^{(1.92+o(1))(\log p)^{1/3}(\log\log p)^{2/3}}$
En pratique pour doubler la sécurité k, il faut multiplier $\log p$ par $2^3 = 8$

Le logarithme discret (2/2)

NIST SP 800 56A

Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography

Recommandations (basés sur le crible algébrique)

Sécurité k=80 $\Rightarrow \log_2 p = 1024, \log_2 q = 160$
Sécurité k=112 $\Rightarrow \log_2 p = 2048, \log_2 q = 224$

Et pour plus de sécurité

Log discret sur les corps finis non recommandé

Réciproque

Est-ce que quelqu'un qui casse DH casse le log discret ?

Personne n'a réussi à le prouver en 40 ans

Est-ce que casser DH est plus simple que résoudre le log discret ?

Personne n'a réussi à le prouver en 40 ans

Sur quoi repose DH ?

Computational Diffie-Hellman (CDH) Problem : pour (p, g, g^a, g^b) trouver g^{ab}
CDH Assumption : Algorithmes en temps poly $\Rightarrow$ avantage négligeable

DH dans un groupe générique

Contexte

- Trouver un groupe avec une opération OP tel que le log discret soit difficile
- Prendre un g du groupe générant une sous-groupe de grande taille q
- À partir de $\underbrace{g\ OP\ g\ OP\ \dots\ OP\ g}_{\text{x fois}}$ trouver x est difficile

Protocole

- A tire $priv_A \leftarrow \{1, ..., q\}$
- A calcule $pub_A = g\ OP\ \dots\ OP\ g\ priv_A$ fois et l'envoie à B
- B fait de même
- On a bien

$$\underbrace{pub_A\ OP\ pub_A\ OP\ \dots\ OP\ pub_A}_{\text{priv}_B\ \text{fois}} = \underbrace{g\ OP\ g\ OP\ \dots\ OP\ g}_{\text{priv}_A * \text{priv}_B\ \text{fois}} = \underbrace{pub_B\ OP\ pub_B\ OP\ \dots\ OP\ pub_B}_{\text{priv}_A\ \text{fois}}$$

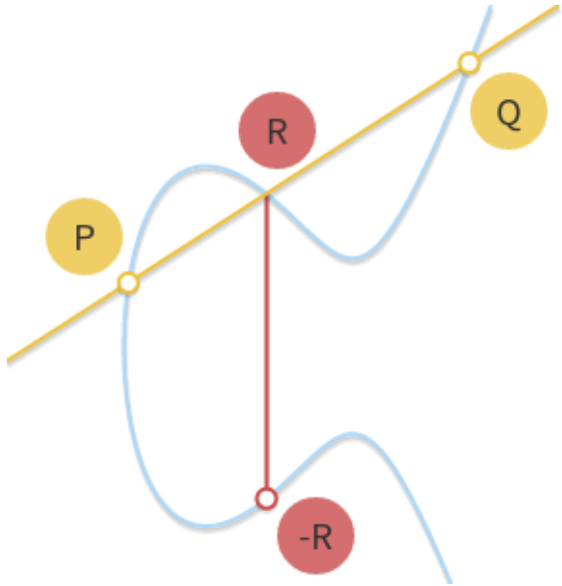
$\Rightarrow$ A et B obtiennent bien la même clé

Sécurité : CDH Assumption dans le corps associé (pas de groupe connu ou CDH est facile et le log discret difficile)

Le logarithme discret sur les courbes elliptiques

Les courbes elliptiques

- Équation $y^2 = x^3 + ax + b$
- Points à coordonnées entières forment un groupe
- Loi de composition $P + Q = R$ utilisant une tangente
- Elliptic-Curve Diffie-Hellman (ECDH) au tableau



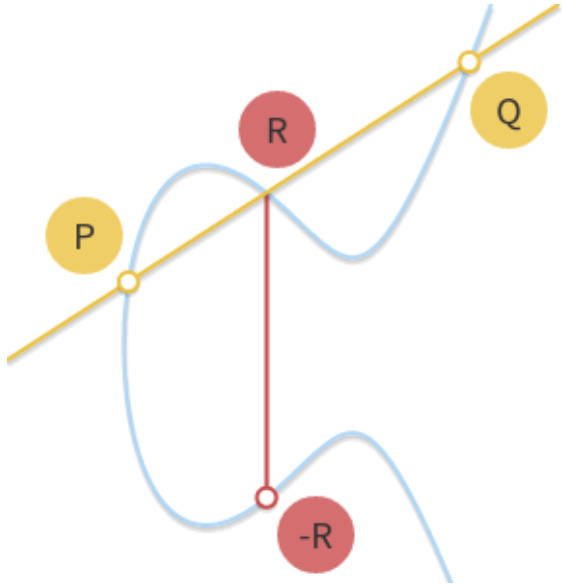
FFC $\rightarrow$ ECC

Multiplication $g * g'$ mod $p \rightarrow$ Somme de deux points $G + G'$
Exponentiation g^a mod $p \rightarrow$ Multiplication d'un point par un scalaire aG
Square and Multiply $\rightarrow$ Double and Add
$A = g^a$, $B = g^b$, $sk = g^{ab} \rightarrow A = aG$, $B = bG$, $sk = abG$

Le logarithme discret sur les courbes elliptiques

Les courbes elliptiques

- Équation $y^2 = x^3 + ax + b$
- Points à coordonnées entières forment un groupe
- Loi de composition $P + Q = R$ utilisant une tangente
- Elliptic-Curve Diffie-Hellman (ECDH) au tableau



Motivation

Meilleur algo pour résoudre le log discret en $\sqrt{p}$, pas sous-exponentiel

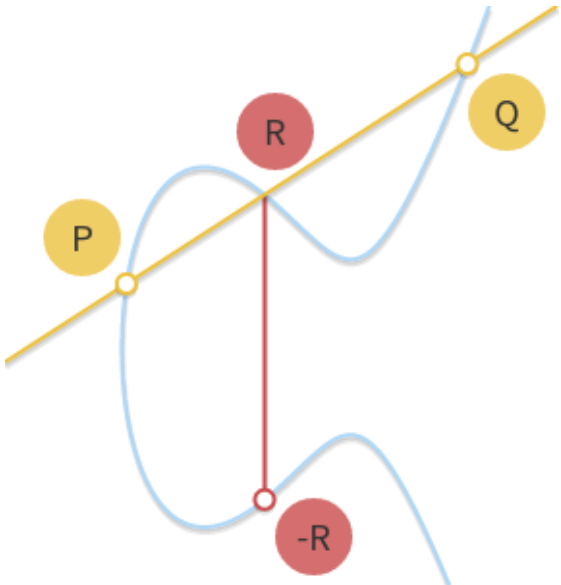
Sécurité $k = 80 \Rightarrow \log_2 p = 160$, $k = 112 \Rightarrow \log_2 p = 224$,
$k = 128 \Rightarrow \log_2 p = 256$, $k = 192 \Rightarrow \log_2 p = 384$, $k = 256 \Rightarrow \log_2 p = 521$

Opérations sur les courbes elliptiques

Les courbes elliptiques

Loi de composition $P + Q = R$ utilisant une tangente
Coût : 14 Muls + 6 Adds [Cohen-Miyaji-Ono 98]
Coordonnées projectives :
$(x, y) \Leftrightarrow (X, Y, Z)$ avec $x = X/Z, y = Y/Z$

```
Input: P = (X1,Y1,Z1), Q = (X2,Y2,Z2)
Y1Z2 = Y1*Z2; X1Z2 = X1*Z2; Z1Z2 = Z1*Z2
u = Y2*Z1-Y1Z2; uu = u2
v = X2*Z1-X1Z2; vv = v2; vvv = v*vv
R = vv*X1Z2; A = uu*Z1Z2-vvv-2*R
X3 = v*A
Y3 = u*(R-A)-vvv*Y1Z2
Z3 = vvv*Z1Z2
Output: P+Q = (X3, Y3, Z3)
```



Exemple pour $k = 128$

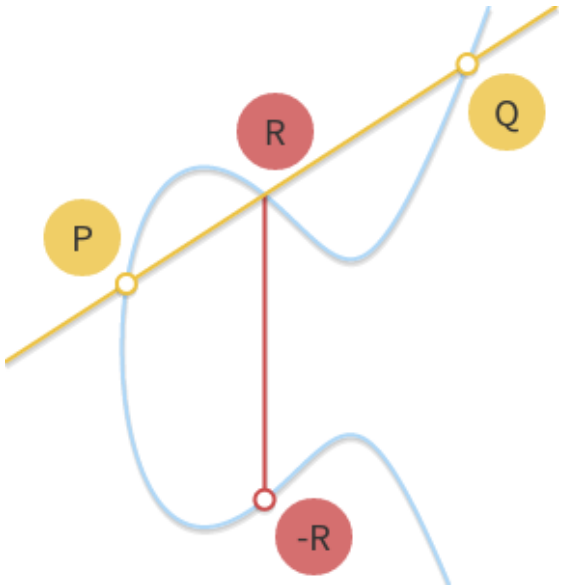
ECC $\Rightarrow$ attaque en $L_p[1, 1/2]$ (c'est à dire $\sqrt{p}$) $\Rightarrow \log_2 p = 256$
FFC $\Rightarrow$ attaque en $L_p[1/3, c] \Rightarrow \log_2 p = 3072$

Opérations sur les courbes elliptiques

Les courbes elliptiques

Loi de composition $P + Q = R$ utilisant une tangente
Coût : 14 Muls + 6 Adds [Cohen-Miyaji-Ono 98]
Coordonnées projectives :
$(x, y) \Leftrightarrow (X, Y, Z)$ avec $x = X/Z, y = Y/Z$

```
Input: P = (X1,Y1,Z1), Q = (X2,Y2,Z2)
Y1Z2 = Y1*Z2; X1Z2 = X1*Z2; Z1Z2 = Z1*Z2
u = Y2*Z1-Y1Z2; uu = u2
v = X2*Z1-X1Z2; vv = v2; vvv = v*vv
R = vv*X1Z2; A = uu*Z1Z2-vvv-2*R
X3 = v*A
Y3 = u*(R-A)-vvv*Y1Z2
Z3 = vvv*Z1Z2
Output: P+Q = (X3, Y3, Z3)
```



Exemple pour $k = 128$

ECC $\Rightarrow$ attaque en $L_p[1, 1/2]$ (c'est à dire $\sqrt{p}$) $\Rightarrow \log_2 p = 256$
FFC $\Rightarrow$ attaque en $L_p[1/3, c] \Rightarrow \log_2 p = 3072$

Surcoût par multiplication de FFC $x144$, surcoût en nb d'opérations de ECC $x14$
$\Rightarrow$ ECC dix fois plus rapide

Échange de clés non interactif [1]

Publication des clés sur Facebook

A,B,C,D publient leur clé publique sur Facebook
$pub_A = g^{priv_A} \bmod p$, $pub_B = g^{priv_B} \bmod p$,
$pub_C = g^{priv_C} \bmod p$, $pub_D = g^{priv_D} \bmod p$

A souhaite communiquer avec C, combien de messages doivent-ils échanger pour arriver à un secret commun ?

Devenez une starlette de la crypto !

Comment faire pour obtenir de la même façon sk_{XYZ} ? [Joux 2004]
Comment faire pour obtenir de la même façon sk_{WXYZ} ? (problème ouvert)

Échange de clés non interactif [1]

Publication des clés sur Facebook

A,B,C,D publient leur clé publique sur Facebook
$pub_A = g^{priv_A} \bmod p$, $pub_B = g^{priv_B} \bmod p$,
$pub_C = g^{priv_C} \bmod p$, $pub_D = g^{priv_D} \bmod p$

A souhaite communiquer avec C, combien de messages doivent-ils échanger pour arriver à un secret commun ?

0 : A récupère sur la page de C sa clé et calcule $sk_{AC} = pub_C^{priv_A} \bmod p$

Devenez une starlette de la crypto !

Comment faire pour obtenir de la même façon sk_{XYZ} ? [Joux 2004]
Comment faire pour obtenir de la même façon sk_{WXYZ} ? (problème ouvert)

Man in the Middle Attacks

Attaquants actifs

Un attaquant voyant passer (g, p, g^a, g^b) ne peut pas obtenir g^{ab} mais que peut faire un attaquant qui modifie les communications ?

Principe

Au tableau

Définition

On dit d'un échange de clés qu'il résiste au MITM s'il résiste même à un attaquant interceptant et modifiant les communications

Solutions

Simple : obtention des clés par un canal sûr et mémorisation (coût linéaire)
Sans canal sûr : il faudrait un moyen de certifier les clés

Confidentialité persistante (Forward Security)

Et si...

Quelqu'un volait la clé secrète de A sans que A s'en rende compte ...
Les échanges de clés futurs avec A seraient sécurisés ?

Et si...

Quelqu'un avait enregistré tous les échanges passés et les communications chiffrées avec des clés dérivées des secrets communs obtenus ?
Il pourrait utiliser la clé secrète volée à A a posteriori ?

Confidentialité persistante (Forward Security)

Et si...

Quelqu'un volait la clé secrète de A sans que A s'en rende compte ...
Les échanges de clés futurs avec A seraient sécurisés ? Non

Et si...

Quelqu'un avait enregistré tous les échanges passés et les communications chiffrées avec des clés dérivées des secrets communs obtenus ?
Il pourrait utiliser la clé secrète volée à A a posteriori ?

Confidentialité persistante (Forward Security)

Et si...

Quelqu'un volait la clé secrète de A sans que A s'en rende compte ...
Les échanges de clés futurs avec A seraient sécurisés ? Non

Et si...

Quelqu'un avait enregistré tous les échanges passés et les communications chiffrées avec des clés dérivées des secrets communs obtenus ?
Il pourrait utiliser la clé secrète volée à A a posteriori ? Oui

Confidentialité persistante (Forward Security)

Et si...

Quelqu'un volait la clé secrète de A sans que A s'en rende compte ...
Les échanges de clés futurs avec A seraient sécurisés ? Non

Et si...

Quelqu'un avait enregistré tous les échanges passés et les communications chiffrées avec des clés dérivées des secrets communs obtenus ?
Il pourrait utiliser la clé secrète volée à A a posteriori ? Oui

L'utilisation de clés éphémères

A et B font deux échanges :
- un avec leur clé publique habituelle $sk_{AB} = g^{priv_A * priv_B} \bmod p$
- un avec deux clés générées pour l'occasion $sk'_{AB} = g^{priv'_A * priv'_B} \bmod p$

Ils définissent $sk_{AB} || sk'_{AB}$ comme leur secret commun
Puis ils détruisent $priv'_A$ et $priv'_B$
Les clés habituelles sont dites statiques et les autres éphémères

Standards pour les protocoles de type DH

NIST SP 800 56A

Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography

Plusieurs familles en fonction du nombre de clés statiques et éphémères :
$C(2s, 2e)$, $C(2s, 1e)$, $C(1s, 2e)$, $C(1s, 1e)$, $C(2s)$, $C(2e)$
Les protocoles à deux clés peuvent utiliser DH ou ECDH, celles avec plus peuvent utiliser DH, ECDH ou des variantes de MQV (protocole proche hors programme)

Pour les familles $C(2s, 1e)$ et $C(1s, 2e)$ la partie ayant une seule clé l'utilise deux fois (pour l'échange statique et pour l'échange éphémère)

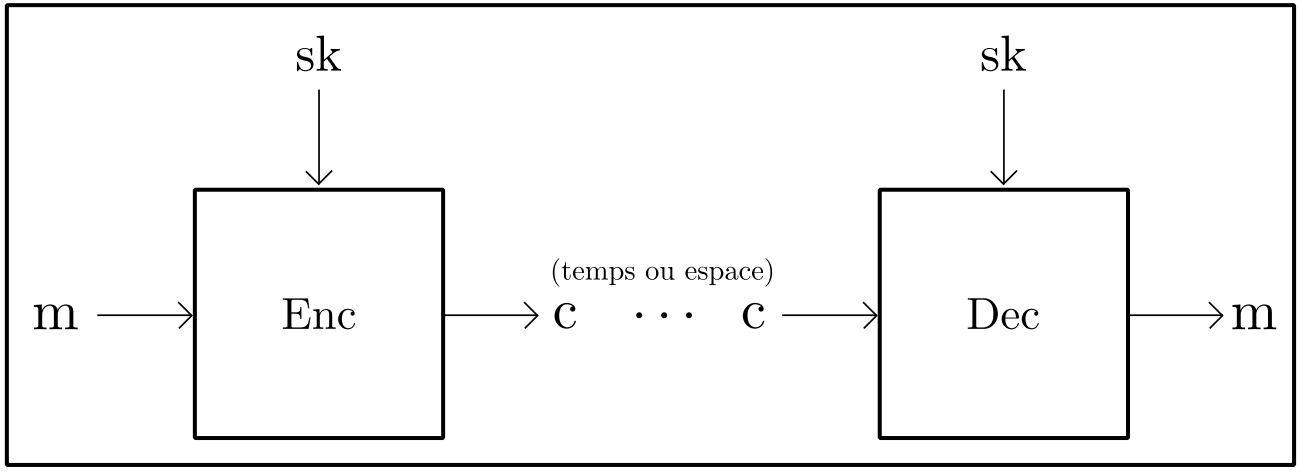
Clés et sécurité

Les utilisateurs sont supposés avoir un certain niveau de confiance du lien entre le clés statiques et les identités

Pour une partie, utiliser une clé statique lui permet de prouver son identité et utiliser une clé éphémère lui permet de protéger les échanges passés en cas de vol de clé privée statique (e.g. AA protègerait ses clients avec un protocole de type $C(1s, 2e)$)

Plan

L'arrivée de la cryptographie à clé publique



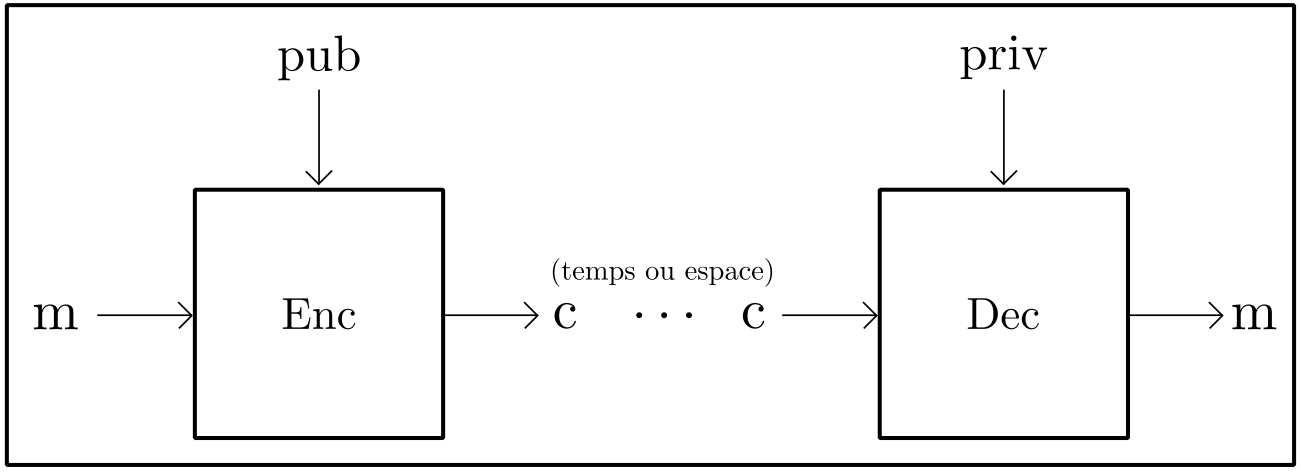
Une machine à laver ...

Inversible ET avec un design publique DONT on dévoilerait la clé ...
Comment ça peut être sûr ?

RSA (1977)

Rivest Shamir et Adleman
Premier protocole permettant de générer $(pub, priv)$ tq dévoiler pub en gardant $priv$ secret ne casse pas complètement la sécurité

L'arrivée de la cryptographie à clé publique



Une machine à laver ...

Inversible ET avec un design publique DONT on dévoilerait la clé ...
Comment ça peut être sûr ?

RSA (1977)

Rivest Shamir et Adleman
Premier protocole permettant de générer ($pub, priv$) tq dévoiler pub en gardant $priv$ secret ne casse pas complètement la sécurité

Cryptosystèmes à clé publique : Définition

Algorithmes

Un cryptosystème à clé publique (PKC) est composé de trois algorithmes

- $KeyGen(1^k)$: alg. randomisé donnant en sortie une bi-clé $(pub, priv) \in K$
- $Enc(pub, m)$: alg. randomisé donnant en sortie un chiffré $c \in C$ de m
- $Dec(priv, c)$: alg. déterministe donnant en sortie un clair $m \in M$

Consistance

$\forall k, \forall (pub, priv) \leftarrow KeyGen(1^k), \forall m \in M, Dec(priv, Enc(pub, m)) = m$

Cryptosystèmes à clé publique : Sécurité

Minimum indispensable : Fonction à trappe (Trapdoor Function)

Pour $(pub, priv) \leftarrow KeyGen(1^k)$, $x \overset{U}{\leftarrow} M$ et $c \leftarrow Enc(pub, x)$ tout algo polynomial a une chance négligeable de donner en sortie $x' = x$

Propriétés de sécurité souhaitées

Objectif : Indistingabilité $\Rightarrow$ Sécurité sémantique (on apprend rien sur le clair)
L'attaquant dispose de la clé publique et l'indistingabilité doit tenir :

- IND-CPA : contre clairs choisis (attaquant choisi le challenge)
- IND-CCA1 : contre chiffrés choisis (à déchiffrer par un oracle avant le challenge)
- IND-CCA2 : contre chiffrés choisis adaptatifs (à déchiffrer par un oracle avant et après le challenge, sauf le challenge)

En pratique un cryptosystème sans IND-CCA2 est toujours dangereux, et il faut se restreindre dans ces cas à des utilisations particulières

Passage : Trapdoor Function ⇒ IND-CCA2 (1/2)

Obtention d'un cryptosystème à clé publique IND-CCA2

Théorème (prouvé sous le modèle du RO) :
si $(KeyGen, Enc, Dec)$ fonction à trappe, H CSHF, (E_s, D_S) Chiffrement Symmétrique Authentifié (avec MAC sur le chiffré) **alors** pour

- $Enc'(pub, m) = \{x \xleftarrow{U} M, sk = CSHF(x), y \leftarrow Enc(pub, x),$ return $y \| E_S(sk, m)\}$
- $Dec'(priv, y\|c) = \{x = Dec(priv, y), sk = CSHF(x),$ return $D_S(sk, c)\}$
- $KeyGen' = KeyGen$

$(KeyGen', Enc', Dec')$ est un chiffrement à clé publique IND-CCA2

Explication

Dans $y\|E_S(sk, m)$, y est un chiffré par PKC contenant une clé sk pour E_S
Si on déchiffre y par PKC on peut obtenir la clé sk et on peut déchiffrer le message

TF ⇒ L'attaquant ne peut pas obtenir x dans sa totalité
CSHF ⇒ Si l'attaquant n'a pas tout x il ne connaît rien sur sk
Puis ... Chiffrement symétrique + MAC sur chiffré ⇒ IND-CCA2 (hors programme)

Passage : Trapdoor Function $\Rightarrow$ IND-CCA2 (1/2)

Obtention d'un cryptosystème à clé publique IND-CCA2

Théorème (prouvé sous le modèle du RO) :
si ($KeyGen$, Enc, Dec) fonction à trappe, H CSHF, (E_s, D_S) Chiffrement Symmétrique Authentifié (avec MAC sur le chiffré) **alors** pour

- $Enc'(pub, m) = \{x \xleftarrow{U} M, sk = CSHF(x), y \leftarrow Enc(pub, x),$ return $y \| E_S(sk, m)\}$
- $Dec'(priv, y\|c) = \{x = Dec(priv, y), sk = CSHF(x),$ return $D_S(sk, c)\}$
- $KeyGen' = KeyGen$

($KeyGen'$, Enc', Dec') est un chiffrement à clé publique IND-CCA2

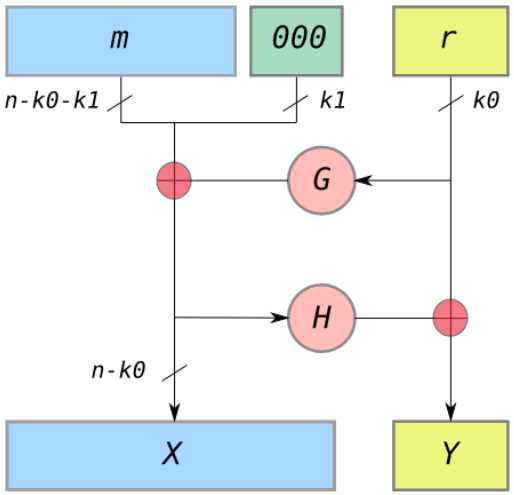
Standards

Paradigme KEM-DEM : Key Encryption Mechanism + Data Encryption Mechanism
On parle souvent de cryptographie hybride (avantage supplémentaire vitesse)

NIST SP 800 56B (échanges de clés basés sur le pb de la factorisation)
$\rightarrow$ RSA-KEM-KWS (Key Wrapping Scheme et pas DEM car pour transport de clés)

Passage : Trapdoor Function ⇒ IND-CCA2 (2/2)

Optimal Assimetric Encryption Padding (OAEP, 1994)



- n taille en bits des clairs
- G et H fonctions de type SHA récentes
- $k0$ longueur de sortie de la fonction H
- m paddé avec des zéros jusqu'à une taille $n - k0$
- G utilisée avec un compteur de 32 bits pour avoir une sortie assez grande $G(r||0)||G(r||1)\ldots$ (puis on tronque)

Source https ://en.wikipedia.org/wiki/Optimal_asymmetric_encryption_padding

PKCS#1 v2

Thm (Fujisaki Okamoto Pointcheval Stern 2001) :
RSA Fonction à trappe ⇒ RSA-OAEP est IND-CCA2 dans le modèle du RO
Fonction à trappe générique ⇒ PKC avec IND-CCA1 dans le modèle du RO

Rq1 : OAEP+ [Shoup 2001] FT générique ⇒ PKC avec IND-CCA2
Rq2 : PKCS#1 v1 et v1.5 n'avaient pas de preuve ... et sont cassés

Chiffrement à clé publique : Utilisations

Envoi de messages de n utilisateurs vers 1

Clé publique $\Rightarrow$ Partage de clé par un canal assurant l'intégrité
Pas de confidentialité $\Rightarrow$ Peut être connue de tout le monde

Envoi de message confidentiel : $Enc(pub_{Carlos}, m)$
Sécurité sémantique : un attaquant ne peut rien apprendre du chiffré (dans le modèle ou la sécurité sémantique a été prouvée)

Authentification

Un serveur associe pub_A à l'utilisateur A, quand il veut s'authentifier :
- Le serveur prend $r \leftarrow \{0,1\}^k$, et envoie $defi = Enc(pub_A, r)$ à A
- A déchiffre $defi$ et renvoie r au serveur

Si sécurité sémantique : $min(k, k_{PKC})$ bits de sécurité
Quel avantage par rapport à utiliser des mots de passe ?

RSA : Bases

Modules et fonction phi

Pour un module m la fonction $\phi(m)$ définit la taille du groupe multiplicatif
Th d'Euler : $\forall x \in \mathbb{Z}_m, x^{\phi(m)} = 1 \bmod m$

Fonction phi pour un module RSA

Si $N = p * q$ avec p, q premiers, on a $\phi(N) = (p-1)(q-1)$

Secret phi $\Leftrightarrow$ Factorisation

$\phi(N) = p * q - (p+q) + 1$
Si on connait $\phi(N)$ et $N = p * q$ on connait $p+q$
Quelles sont les racines de $X^2 - (p+q) * X + (p*q)$?

RSA : Algorithmes

$KeyGen(1^k)$

- Choisir une taille de module $\log_2 N$ en suivant les consignes du NIST pour une sécurité k
- Choisir deux premiers p, q de $1/2 \log_2 N$ et noter $N = p * q$ et $\phi(N) = (p-1)(q-1)$
- Définir $e = 2^{16} + 1$ et choisir d et $const$ tq $e * d = 1 + const * \phi(N)$ (alg. d'Euclide étendu)
- Retourner $(pub, priv)$ avec $pub = (e, N)$ et $priv = (d, N)$

$Enc((e, N), m)$

- Retourner m^e mod N

$Dec((d, N), c)$

- Retourner c^d mod N

Remarques

Espaces : $M = C = \mathbb{Z}_N$, $d \in \mathbb{Z}_{\phi(N)}$
Consistance (opérations mod N) :
$c^d = m^{e*d} = m^{1+const*\phi(N)} = m * m^{const*\phi(N)} = m * (m^{\phi(N)})^{const} = m$

RSA : Sécurité (1/2)

Exemple avec $e = 3$

$N = 15$

$1 \longrightarrow 1$

$2 \longrightarrow 8$

$4 \longrightarrow 4$

$7 \longrightarrow 13$

$8 \longrightarrow 2$

$11 \longrightarrow 8$

$13 \longrightarrow 7$

$14 \longrightarrow 14$

RSA : Sécurité (1/2)

Exemple avec $e = 3$

$N = 15$

$? \longleftarrow 1$

$? \longleftarrow 2$

$? \longleftarrow 4$

$? \longleftarrow 7$

$? \longleftarrow 8$

$? \longleftarrow 11$

$? \longleftarrow 13$

$? \longleftarrow 14$

RSA : Sécurité (2/2)

Chiffrement déterministe

Aucune indistingabilité : l'attaquant choisit m_0, m_1, quand il reçoit c il calcule $c_b = Enc(pub, m_b)$ et vérifie si $c = c_0$ ou $c = c_1$

RSA Assumption

RSA est une fonction à trappe

Comme pour le Log discret et DH. On sait casser RSA avec la factorisation mais en 40 ans personne n'a pu montrer : ni la réciproque (comme quoi casser RSA impliquerai un algo de factorisation) ; ni à l'opposé que RSA soit plus simple que la factorisation

Attention à RSA sans IND-CCA2 !

Impact sur la confidentialité

Envoi de message confidentiel : $Enc(pub_{Carlos}, m)$. Cas non-sûrs ?

Impact sur l'authentification

Challenge : Le serveur prend $r \leftarrow \{0,1\}^k$, et envoie $defi = Enc(pub_A, r)$.
Sécurité ? Indice $P[r = r_1 * r_2 \mid r_1 \text{ et } r_2 \text{ de } k/2 \text{ bits}] \simeq 0.2$

Attention à RSA sans IND-CCA2 !

Impact sur la confidentialité

Envoi de message confidentiel : $Enc(pub_{Carlos}, m)$. Cas non-sûrs ?
m petit (inférieur à k_{PKC} bits)

Impact sur l'authentification

Challenge : Le serveur prend $r \leftarrow \{0,1\}^k$, et envoie $defi = Enc(pub_A, r)$.
Sécurité ? Indice $P[r = r_1 * r_2 \mid r_1 \text{ et } r_2 \text{ de } k/2 \text{ bits}] \simeq 0.2$

Attention à RSA sans IND-CCA2 !

Impact sur la confidentialité

Envoi de message confidentiel : $Enc(pub_{Carlos}, m)$. Cas non-sûrs ?
m petit (inférieur à k_{PKC} bits)

Impact sur l'authentification

Challenge : Le serveur prend $r \leftarrow \{0,1\}^k$, et envoie $defi = Enc(pub_A, r)$.
Sécurité ? Indice $P[r = r_1 * r_2 \mid r_1 \text{ et } r_2 \text{ de } k/2 \text{ bits}] \simeq 0.2$
Table avec les r_1^e mod N chercher $c/(r_2^e)$ mod N dedans $\rightarrow 2^{k/2} + 2^{k/2}$ opérations

Attention à RSA sans IND-CCA2 !

Impact sur la confidentialité

Envoi de message confidentiel : $Enc(pub_{Carlos}, m)$. Cas non-sûrs ?
m petit (inférieur à k_{PKC} bits)

Impact sur l'authentification

Challenge : Le serveur prend $r \leftarrow \{0,1\}^k$, et envoie $defi = Enc(pub_A, r)$.
Sécurité ? Indice $P[r = r_1 * r_2 \mid r_1 \text{ et } r_2 \text{ de } k/2 \text{ bits}] \simeq 0.2$
Table avec les r_1^e mod N chercher $c/(r_2^e)$ mod N dedans $\rightarrow 2^{k/2} + 2^{k/2}$ opérations

Conclusion

Ne jamais utiliser RSA seul (dit souvent RSA-raw) pour autre chose que chiffrer des valeurs uniformément aléatoires !

Autres attaques à prendre en compte

Petit e, ou petit d

Ne jamais prendre un d petit ! (nombreuses attaques)
e est déjà optimisé (16 carrés + 1 multiplication), le laisser comme ça

Analyse de temps ou de courant [Kocher 97 et 99]

Square and Multiply : très sensible aux analyses de temps et de courant
Opérations dépendent des bits de d, si on reconnaît les opérations on obtient les bits
Utiliser des algorithmes calculant les deux possibilités à chaque étape

Manque d'aléa pour Keygen $\rightarrow$ Factorisation massive

Si N_1 et N_2 ont un facteur commun $pgcd(N_1, N_2)$ puis factorisation
Lenstra et al. factorisation de 0.4% des clés OpenSSL sur Internet

Faute avec le CRT

Déchiffrement CRT : Calculer mod$p$ et mod$q$ puis faire un relèvement (x4)
Si une faute quelconque se glisse dans l'un des deux alors on a pour x la sortie :
$pgcd(x, N) \in \{p, q\} \Rightarrow$ vérifier en rechiffrant avant de donner la sortie

Et les échanges de clés ? (1/2)

NIST SP 800 56B

Recommendation for Pair-Wise Key Establishment Schemes Using Integer Factorization Cryptography

Deux possibilités : Key Agreement Schemes (KAS1 et KAS2) et Key Transport Schemes (KTS-OAEP, KTS-KEM-KWS)

Primitives

Pour $n = \log_2 N$
RSA-SVE (Secret Value Encapsulation) : $x \leftarrow \{0,1\}^n$, $Enc(pub, x)$
RSA-KEM-KWS : $x \leftarrow \{0,1\}^n$, $sk = CSHF(x)$, $y = Enc(pub, x)$, $y \| E_S(sk, m)$
RSA-OAEP : $Enc(pub, OAEP(m))$

Et les échanges de clés ? (2/2)

Les accords de clés

Les deux parties contribuent à l'aléa du secret commun
KAS1 : $A \rightarrow B$ RSA-SVE(Z), $B \rightarrow A$ *Random*, sortie $Z \| Random$
KAS2 : $A \rightarrow B$ RSA-SVE(Z), $B \rightarrow A$ RSA-SVE(Z'), sortie $Z \| Z'$

Les transports de clés

Une seule partie définit le secret
KTS-OAEP : $A \rightarrow B$ RSA-OAEP(Z), sortie Z
KTS-KEM-KWS : $A \rightarrow B$ RSA-KEM-KWS(Z), sortie Z

Finalisation d'un échange de clés

Dérivation

Un échange se termine par un appel à une fonction de dérivation de clés (KDF)
Tous les standards demandent cela (je ne vous l'ai pas présenté pour isoler la partie échange de la partie dérivation)

Binding

On prend le secret échangé en entrée et toujours un contexte
Contexte : acteurs, date, utilisation, application, PID, etc.
Puis on utilise la fonction de dérivation KDF sur la concaténation

Confirmation

Les utilisateurs ne s'authentifient pas en envoyant leur clé publique ...
Mais en montrant qu'ils ont obtenu le secret commun

NIST SP 800 56A/B
KDF $\rightarrow$ $MacKey||SharedSecret$
Envoi (pot. bilateral) de $MAC(MacKey, "KC_{2,A}" || ID_A || ID_B || Envois_A || Envois_B)$

Plan

Utilisations de la dérivation de clés

Extraction

La sortie d'un échange de clés peut ne pas être uniforme (pour Diffie-Hellman $g^{priv_A * priv_B \bmod q} \bmod p$, e.g. $\log_2 q = 224$ bits d'entropie, $\log_2 p = 2048$ bits de taille)

Dérivation de clés $\rightarrow$ secret plus uniforme de plus petite taille

Rattachement à un contexte (binding)

Vu pour l'échange de clés : assurer que le résultat soit unique pour une transaction/application/utilisation/date/etc.

Expansion

À partir d'un secret de taille k obtenir une chaîne pseudo-aléatoire de plus grande taille ou une série de secrets à la demande

NIST SP800 56A/B : éviter les utilisation publiques (sels, IVs, etc.)

Extraction + Renforcement

Utiliser une KDF lente pour dériver une clé ou un haché à partir d'un mot de passe

La dérivation en une étape

Les fonctions KDF1/MGF1 et KDF2

KDF1/MGF1 (ISO 18033-2, IEEE P1363A, PKCS#1 v2) :
$Hash(Z||C||Context)$ avec C compteur sur 32 bits commençant à 0

KDF2 (ISO 18033-2, ANSI X9.42) : Même chose avec compteur commençant à 1

La fonction KDF3

KDF3 (ISO 18033-2, NIST SP 800 56A/B) :
$Hash(C||Z||Context)$ avec C compteur sur 32 bits commençant à 0

KDF3 (alternative NIST SP 800 56A/B) :
$HMAC(salt, C||Z||Context)$ si *salt* non défini, considérer que c'est des zéros

La dérivation en deux étapes

Standards : HKDF (HMAC-based KDF)

NIST SP 800 56C “Recommendation for Key Derivation through Extraction-then-Expansion”
Approuvé pour NIST SP 800 56A/B
Extraction et Expansion basés sur un HMAC (bits de sécurité $<$ bits du haché) ou AES-CMAC (bits de sécurité $<$ 128)

Extraction

$Extraction(salt, Z) = MAC(salt, Z) = KDK$
Entrée avec beaucoup d’entropie mais potentiellement non uniforme

Expansion

$Expansion(KDK, Context, L) = MatCrypto$ de L octets
On veut dériver plusieurs secrets avec une sécurité de k bits

NIST SP 800 108 “Recommendation for Key Derivation Using Pseudorandom Functions”
Implémentation de Expansion : MAC en CTR ou avec feedback (tableau)

Les Password-Based KDF (1/2)

Principe

Mots de passe salés : déjà vu (rainbow tables)
Ralentissement des attaques par dictionnaire (Password Stretching)

crypt (Unix, Linux, BSD)

Fonction historique de hachage des MDP

`r=0; Pour i de 1 à 25: r=DES(salt||password, r)`

Puis stockage de `$salt$r`

De nos jours le nb de tours et paramétrable et la fonction aussi :
DES, MD5, Blowfish, SHA-256, SHA-512 (par défaut) sur une Ubuntu
Stockage : `$algo_nb$rounds=X$salt$r`

Les Password-Based KDF (2/2)

PKCS#5

Password-based Cryptography Standard
v1.5 définit PBKDF1, v2 définit PBKDF2

PBKDF2

`CléDérivée = PBKDF2(PRF, mdp, salt, boucles, L)`

`i=0; CléDérivée`$_i$ `= PRF(mdp, salt||i);`

`Pour j de 1 à boucles: CléDérivée`$_i$ `= PRF(mdp, CléDérivée`$_i$`);`

Si la taille de la sortie demandée `L` n'est pas exactement la sortie de PRF on itère en incrémentant `i`, on concatène les `CléDérivée`$_i$, puis on tronque à la bonne taille
Exemple WPA2 : `PBKFD2(HMAC-SHA1, passphrase, SSID, 4096, 256)`

Utilisé dans ...

Applications : WPA, chiffrement OpenDocument, WinZip, Apple iOS, Mac OSX, Cisco iOS, Firefox sync, etc.

Chiffrement : FileVault (Mac), LUKS, Truecrypt, EncFS, GRUB2, etc.

L'utilisation hiérarchique

Au tableau !

Fin

Prochain cours

Signature électronique