Cryptographie

Une introduction

A. Bonnecaze
Polytech/IML

Cours 3

- Cryptographie à clé publique
- Protocoles particuliers

Cryptographie à clef publique

- 1976 Diffie & Hellman
- 1978 RSA, 1985 ElGamal, ECC
- Basée sur des problèmes difficiles
  - Logarithme discret
  - Factorisation de grands entiers
- Clef de chiffrement et de déchiffrement distinctes
- Trop lent pour chiffrer directement des données
  - Chiffrer une clef de session, échange de clefs, signatures

Crypto à clé publique : principales applications

- Confidentialité :
  - Chiffrement/déchiffrement
- Authentification :
  - Signature digitale
- Clés de session :
  - Échange de clés

Problèmes mathématiques

Repose sur l'existence de fonction à sens unique (avec trappe)

Par exemple :

Factoriser un grand nombre

$$\begin{tikzcd} F : x,y \arrow[rrr, bend left, "\textit{facile}"] & & & z=xy \arrow[lll, bend left, "\textit{difficile}"] \end{tikzcd}$$

Problèmes algorithmiquement difficiles

Factorisation dans Zp*

$p, q \rightarrow$ calculer $n=pq$ (facile, quadratique)

$n=pq \rightarrow$ déterminer p, q (difficile)

Problème RSA dans Zp*

$g^a, a \rightarrow$ déterminer g

DLP (Discrete Log Problem)

$g^a, g \rightarrow$ déterminer a

CDH (Computational Diffie-Hellman)

$g, g^a, g^b \rightarrow$ déterminer g^{ab}

DDH (Decisional Diffie-Hellman)

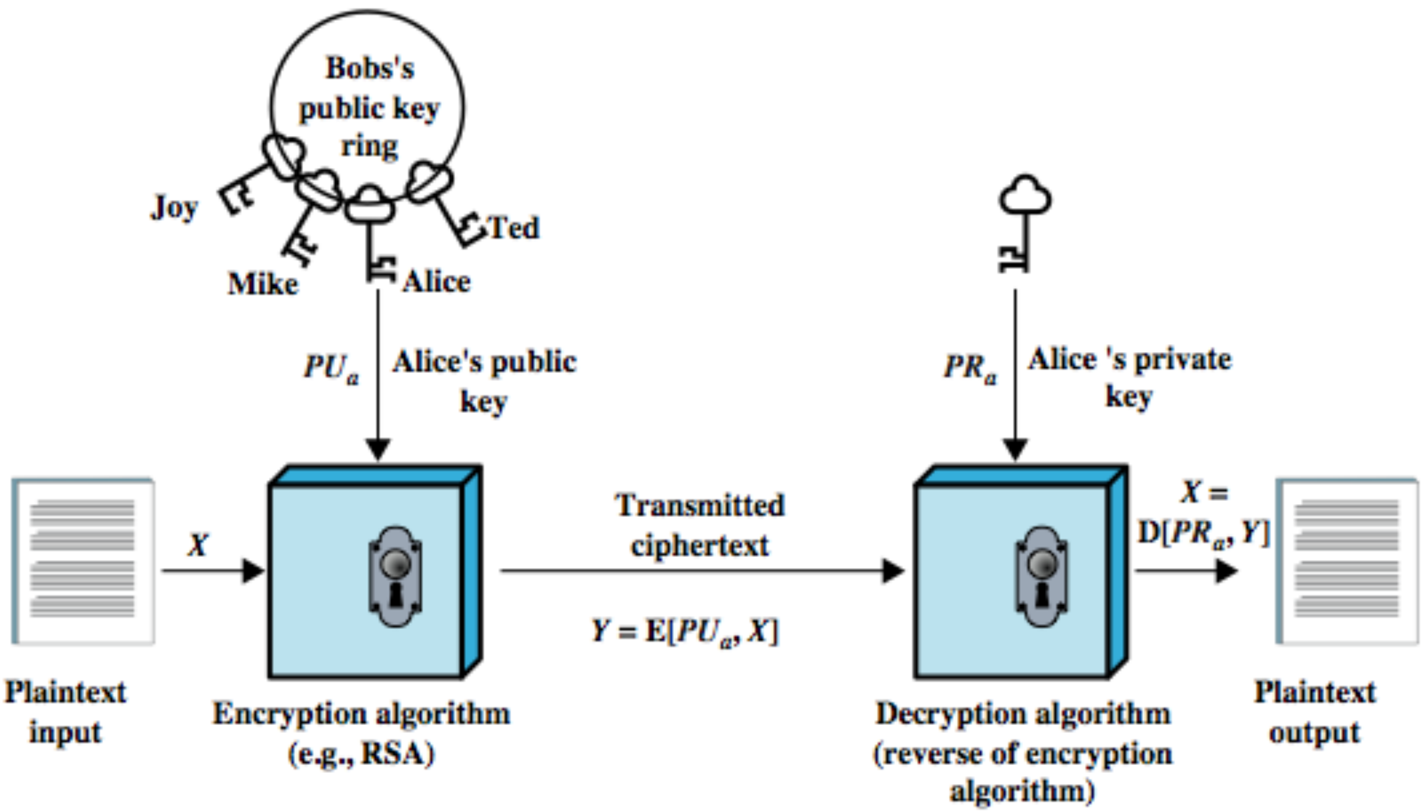
$G, g^a, g^b, g^c \rightarrow$ a-t-on $c = ab$?

Fonctionnement (chiffrement, signature)

- Chiffrement
  - Clef publique pour le chiffrement, clef privée pour déchiffrer
  - Tout le monde peut chiffrer

- Signature
  - Seul le détenteur de la clef privée peut signer,
  - mais tout le monde peut vérifier la signature

Chiffrement à clé publique



(a) Confidentiality

beaucoup plus lent que le
chiffrement symétrique

Algorithmes de clés publiques

- RSA (Rivest, Shamir, Adleman)
  – Développé en 1977
  – Le seul algo de chiffrement de clé publique largement accepté
  – Longueur de clé : au moins 1024 bits
- Diffie-Hellman key exchange algorithm
  – Permet d'échanger une valeur secrete
- Digital Signature Standard (DSS)
  – Fonction de signature digitale utilisant SHA-1
- Elliptic curve cryptography (ECC)
  – nouveau, même sécurité que RSA, mais avec des clés plus courtes

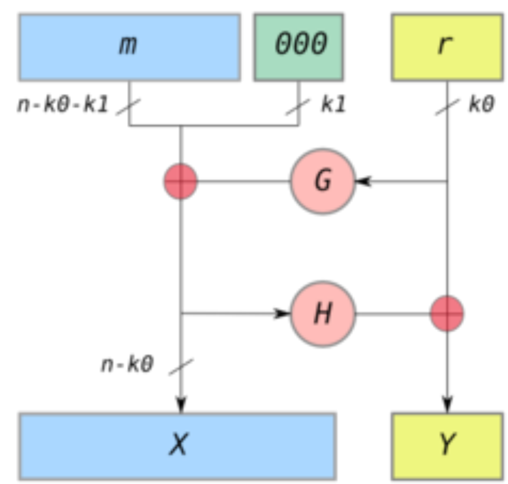
Exemple : RSA

- Chiffrement
- Déchiffrement
- Chiffrement probabiliste?
  - OAEP
- Longueur des clés
  - Sécurité correcte : 4096 bits
  - Sécurité excellente : 15000 bits
    - Trop long!!!

RSA-OAEP

- m = message ; r = valeur aléatoire
- G et H deux fonctions de hachage

$OAEP(m) = [(m \oplus G(r)) \,||\, (r \oplus H(m \oplus G(r)))]$
$= x \,||\, y$



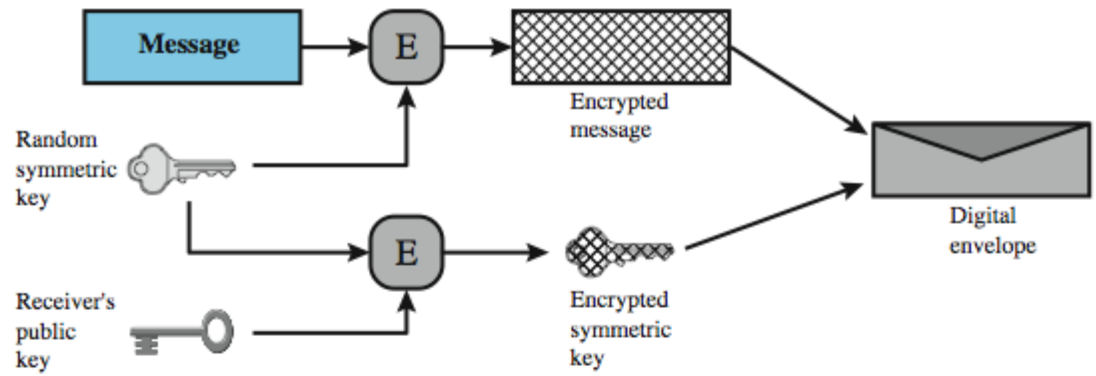
Pour retrouver m

- $r = y \oplus H(x)$
- $m = x \oplus G(r)$

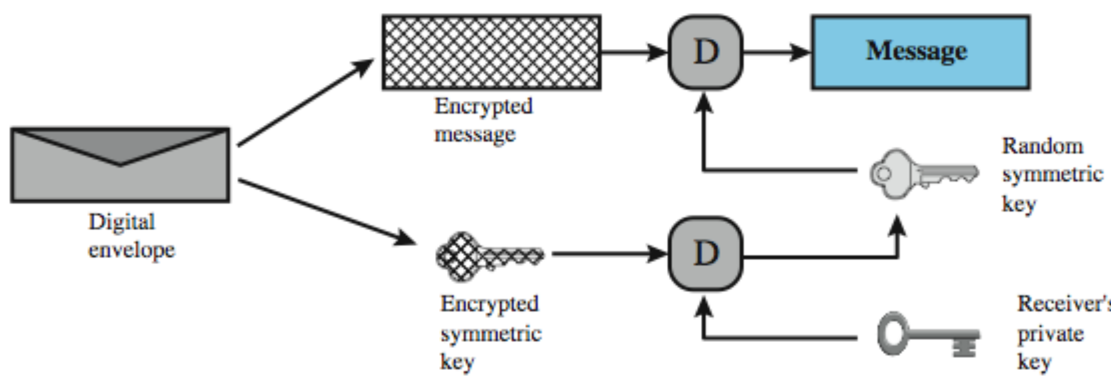
Chiffrement El Gamal (1984)

- Alice calcule h = g^x, avec g, h dans Z*p pour un grand nombre premier p, g étant un élément générateur de Z*p, et divulgue sa clé publique (p,g,h). La valeur x est sa clé privée.
- Si Bob veut envoyer un message à Alice, il convertit d'abord son message sous la forme d'un nombre m de Z*p.
- Bob génère un nombre entier k aléatoirement et calcule c1 = g^k et $c2=m.h^k$.
- Il envoie (c1,c2) à Alice.
- Alice peut reconstruire le message initial m en calculant $c2/c1^x$.
- Il s'agit d'un chiffrement probabiliste car k est choisi aléatoirement pour chaque chiffrement

Enveloppes Digitales



(a) Creation of a digital envelope



(b) Opening a digital envelope

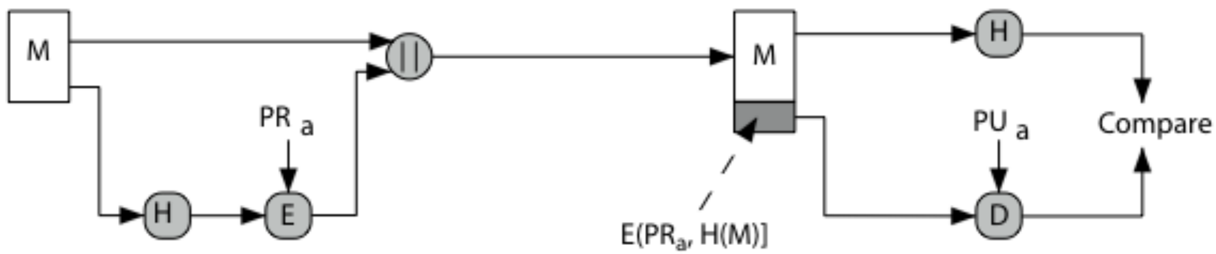
Digital Signature Standard (DSS)

- Approuvé par US Govt
- Choisi par NIST & NSA debut 90's
- Publié comme FIPS-186 en 1991
- Revisé en 1993, 1996 & 2000
- Utilise SHA hash algorithm
- DSS est le standard, DSA est l'algorithme
- FIPS 186-2 (2000) variantes utilisant RSA & elliptic curve

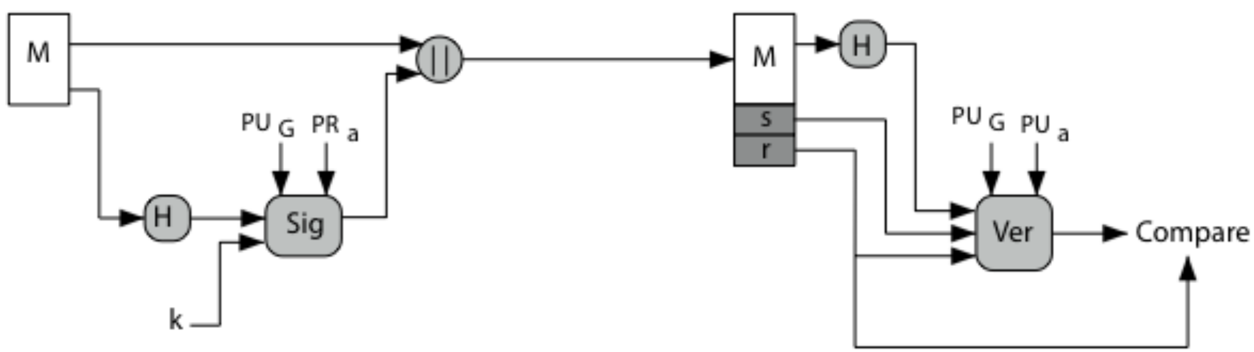
Digital Signature Algorithm (DSA)

- Signature de 320 bits
- Plus courte et plus rapide que RSA
- Seulement un schéma de signature
- Sécurité : dépend de la difficulté de résoudre le pb du log discret
- variante de ElGamal & Schnorr

Digital Signature Algorithm



(a) RSA Approach

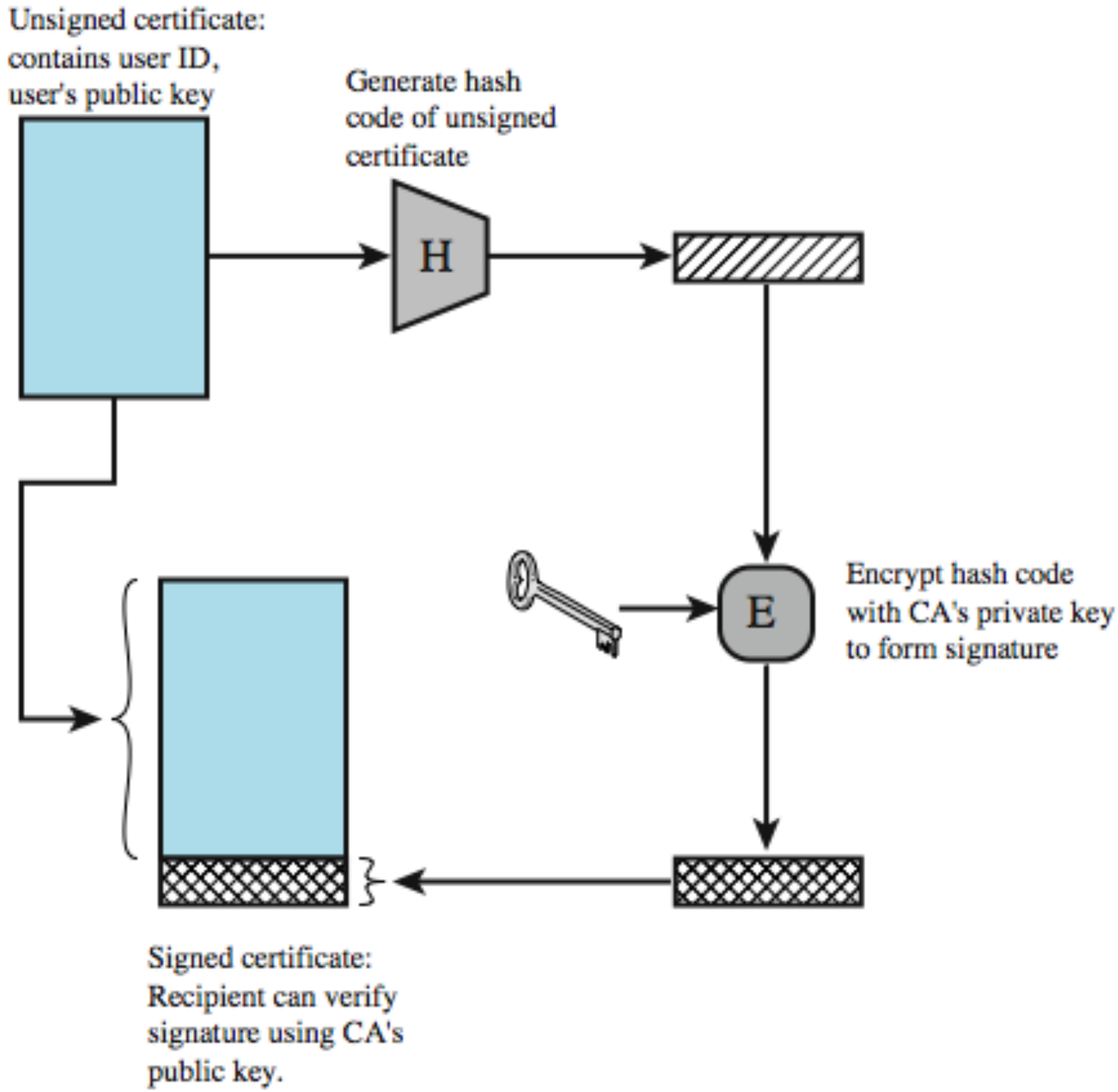


(b) DSS Approach

Echange de clés (de session)

- Le premier protocole d'échange de clés est celui de **Diffie-Hellman**
- On peut aussi utiliser la crypto asymétrique
- Attention à l'attaque man in the middle !
- Il existe actuellement beaucoup de protocoles d'échange de clés (voir cours de sécurité)

Certificats



Certificate:
Data:

Version: 3 (0x2)

Serial Number: 1 (0x1)

Signature Algorithm: md5WithRSAEncryption

Issuer: C=ZA, ST=Western Cape, L=Cape Town, O=Thawte Consulting cc,
OU=Certification Services Division,
CN=Thawte Server CA/Email=server-certs@thawte.com

Validity Not Before: Aug 1 00:00:00 1996 GMT
Not After : Dec 31 23:59:59 2020 GMT

Subject: C=ZA, ST=Western Cape, L=Cape Town, O=Thawte Consulting cc, OU=Certification Services Division,
CN=Thawte Server CA/Email=server-certs@thawte.com

Subject Public Key Info:

Public Key Algorithm: rsaEncryption RSA Public Key: (1024 bit) Modulus (1024 bit):
00:d3:a4:50:6e:c8:ff:56:6b:e6:cf:5d:b6:ea:0c: 68:75:47:a2:aa:c2:da:84:25:fc:a8:f4:47:51:da: 85:b5:20:74:94:86:1e:0f:75:c9:e9:08:61:f5:06:
6d:30:6e:15:19:02:e9:52:c0:62:db:4d:99:9e:e2: 6a:0c:44:38:cd:fe:be:e3:64:09:70:c5:fe:b1:6b:
29:b6:2f:49:c8:3b:d4:27:04:25:10:97:2f:e7:90:d:c0:28:42:99:d7:4c:43:de:c3:f5:21:6d:54:9f:d:c3:58:e1:c0:e4:d9:5b:b0:b8:dc:b4:7b:df:36:3a:c2:b5:66:22:1
2:d6:87:0d

Exponent: 65537 (0x10001)

X509v3 extensions: X509v3 Basic Constraints: critical

CA:TRUE

Signature Algorithm: md5WithRSAEncryption

07:fa:4c:69:5c:fb:95:cc:46:ee:85:83:4d:21:30:8e:ca:d9: a8:6f:49:1a:e6:da:51:e3:60:70:6c:84:61:11:a1:1a:c8:48: 3e:59:43:7d:4f:95:3d:a1:8b:b7:0b:62:98:7a:75:8a:dd:88:
4e:4e:9e:40:db:a8:cc:32:74:b9:6f:0d:c6:e3:b3:44:0b:d9: 8a:6f:9a:29:9b:99:18:28:3b:d1:e3:40:28:9a:5a:3c:d5:b5: e7:20:1b:8b:ca:a4:ab:8d:e9:51:d9:e2:4c:2c:59:a9:da:b9:
b2:75:1b:f6:42:f2:ef:c7:f2:18:f9:89:bc:a3:ff:8a:23:2e: 70:47

Taille des systèmes

- Les tailles sont données en bits

Système	attaque	Taille 1	Taille 2	Taille 3	Taille 4
block	brutale	80	112	128	256
RSA	factorisation	1024	2048	4096	15000
hachage	anniversaire	160	224	256	512
EC	ECDLP	160	200	256	400

Résumé

- Le chiffrement à clé publique est lent et ne s'utilise pas pour chiffrer des données
- Il s'utilise pour chiffrer des clés
- Il DOIT être probabiliste
- La gestion des clés est primordiale
  - Génération
  - distribution
  - Stockage
  - Remplacement, destruction

Web sites

- http://www.faqs.org/faqs/cryptography-faq/
- http://www.rsa.com/rsalabs/node.asp?id=2152
- http://www.cryptography-tutorial.com/cryptofaq.htm
- http://www.bouncycastle.org/
- http://www.di-mgt.com.au/crypto.html
- http://www.simonsingh.net/Crypto_Corner.html
- http://www.cryptogram.org/

Quelques protocoles/outils intéressants

- Fonction caméléon
- Identification
- Engagement
- Partage de secret

Fonction caméléon

Alice est la signataire

la banque la bénéficiaire

Alice ne veut pas que la banque divulgue à un tiers l’information signée sans son consentement

Fonction caméléon

- Une fonction de hachage caméléon est une fonction « collision resistant » à trappe
- Il existe une clé privée (trapdoor) qui permet de trouver des collisions
- Connaissant la clé « trapdoor » et le haché $H(m)$, il est possible de trouver m' tel que $H(m') = H(m)$

Fonction caméléon

pour signer, Alice utilise la fonction de hachage de la banque

La banque connait la clé privée de la fonction de hachage (trapdoor)

La banque envoie à Alice la clé publique de la fonction de hachage

La banque est la seule à pouvoir trouver des collisions

Fonction caméléon

- $H : \{0,1\}^k \times \{0,1\}^r \rightarrow \{0,1\}^k$
- g un générateur d'un groupe cyclique G
- Clé trapdoor : x ; Clé publique : $y = g^x$
- m = message, r est aléatoire
- $H(m,r) = y^m g^r$
- Connaissant x, on peut trouver m' tel que

  $H(m',r') = H(m,r)$ collision !!
- Il suffit de prendre $r' = r+x(m-m')$

En cas de dispute

Alice nie avoir signé m' et fait appel à un juge

2 cas :

- Alice a effectivement signé m'
- La banque a modifié le message m signé par Alice en utilisant la trapdoor

Vérification

La banque envoie au juge la signature sign(m’) = (m’, r’, sig’)

Si la vérif échoue : signature rejetée

Sinon le juge demande à Alice d’accepter cette signature

Si elle accepte finalement : ok

Sinon, elle doit produire une collision sur la fonction de hachage

vérification

Si la banque a triché, Alice peut toujours produire la signature qu’elle a originellement envoyée à la banque

Sign(m) = (m, r, sign)

Le juge vérifie que (m, r) est une collision

Or, la banque est la seule à pouvoir produire une collision (elle connait trapdoor)

Donc Sign(m’) est rejetée

résumé

La signature d’Alice est recevable pour la banque mais

La banque ne peut pas prouver à un tiers qu’Alice à signé un document

Sauf si Alice est d’accord

Identification de Schnorr

- P veut prouver qu'il détient un secret, mais sans révéler le secret
- Basé sur la difficulté de résoudre le problème du logarithme discret
- Autre protocole : Fiat-Shamir

 veut prouver à  qu'il sait calculer le logarithme discret en base g d'une valeur u

$u = g^{a}$

sans révéler à Bob la valeur a

- Calculer le logarithme discret est un problème difficile lorsque les valeurs sont grandes
- Il décide d'utiliser le protocole de Schnorr

Identification de Schnorr (simplifiée)

Paramètres publics : $q|(p-1)$, H ss-groupe d'ordre q de Zp^*, g générateur de H

Paramètre secret de Iwan : a où $a < q$

- Valeur publique de Iwan : $u = g^a \bmod p$
- Iwan veut prouver qu'il connait le log discret de u
- Iwan tire au sort r et envoie $x = g^r \bmod p$ à Bob
- Bob tire au sort z dans Zq et l'envoie à Iwan
- Iwan calcule $v = r - az \bmod q$ et envoie v à Bob
- Bob vérifie que $x = g^v u^z \bmod p$

Partage de secret

- Alice détient un secret s
- Elle ne veut pas le sauvegarder sur un support informatique de peur que le support lui soit volé
- Elle décide de donner à n amis un fragment du secret
- Pour retrouver le secret, Alice contacte t amis parmi les n

Secret sharing scheme…(Shamir)

- Soit S un secret et n personnes
- Comment distribuer un fragment de S à chacune des personnes de telle manière que n'importe quel groupe de $t < n$ personnes puisse reconstituer le secret ?
- Un groupe de moins de t personnes ne doit pas pouvoir reconstituer le secret

SSS suite

- Un polynôme de degré t -1
- $F(x) = a_0+\ldots+a_{t-1}\, x^{t-1}$
- Le secret est a_0
- Alice donne à chacun de ses amis un point du polynôme
- Il faut au moins t points pour retrouver le polynôme grâce à l'interpolation de Lagrange et retrouver le secret a_0

Le nombre d'amis d'Alice est très supérieur à t

Propriétés

Sécurisé : la sécurité de l'information est théorique (non basé sur un pb difficile) et la connaissance de moins de t fragments ne donne pas d'info sur le secret

Minimal : La taille de chaque fragment égale la taille du secret

Extensible : des fragments peuvent être ajoutés ou supprimés

Dynamique : La sécurité peut être facilement améliorée en modifiant le polynôme (avec même secret et nouveaux fragments)

La crypto elliptique (un aperçu)

- 1985 : Koblitz/Miller : inventent la crypto sur courbes elliptiques
- Chiffrement El Gamal fondé sur le problème du log discret
- Signature ECDSA (320 bits)
- Utilisation de pairings pour cryptanalyser des algos cryptographiques
- Intérêt de ECC?

Les courbes elliptiques

- Avantage en complexité
  - Même coût en puissance de calcul pour résoudre
    - log discret sur une courbe de taille cardinal **160** bits
    - factoriser un entier de **1024** bits
- Avantage au niveau des applications
  - Clés plus courtes (1024 -> 163), calculs plus rapides, moindre consommation d'énergie et de mémoire
  - Pairings (Weil, Tate)
    - Identity-Based crypto (absence de PKI)
    - Key agreement
    - Signatures courtes, …

Courbes elliptiques

- Equation :

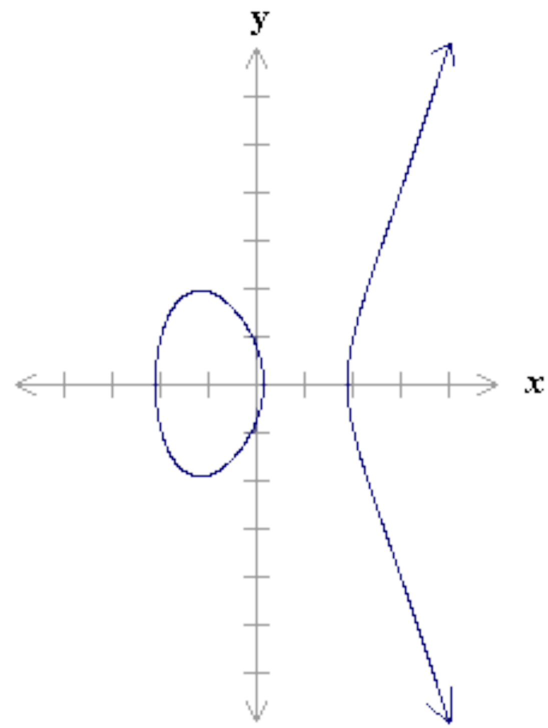
  $$y^2=x^3+ax+b$$

  avec $4a^3+27b^2 <> 0$

- Définie sur K
  - K=R
  - $K=F_p$ (nombre fini de points)
  - $K=F_{2^m}$ (caractéristique 2)

Courbes elliptiques

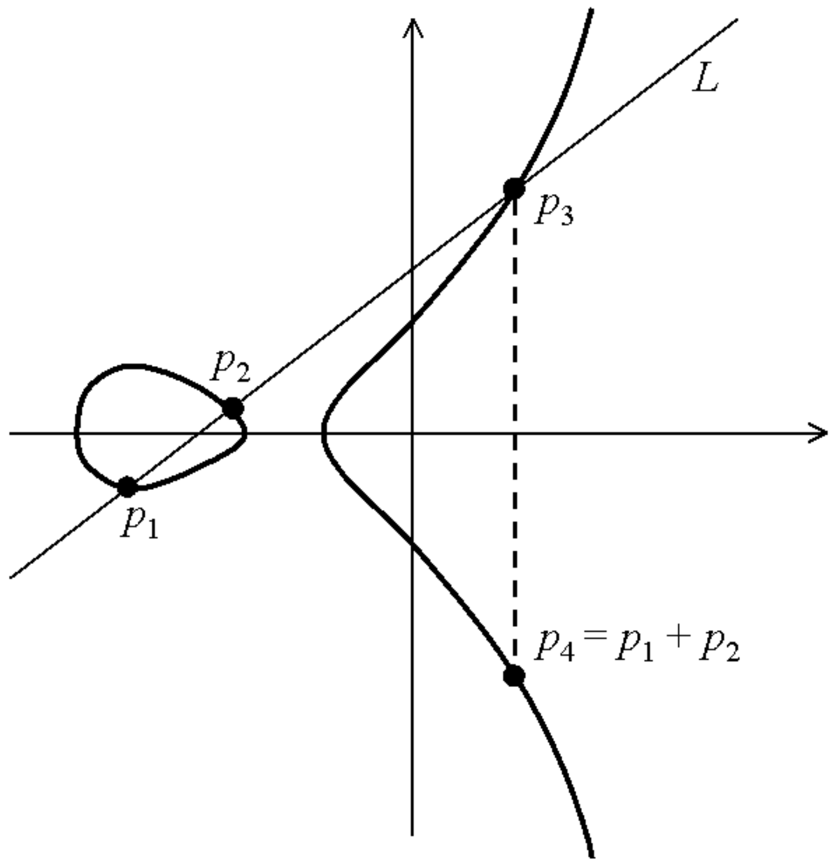
E(K) est l'ensemble des points $P=(x,y) \in K^2$ vérifiant l'équation de Weierstrass, plus le point $O=[0,1,0] \in P^2$ à l'infini



$y^2 = x^3 - 4x + 0.67$

Remarque : *x fixé, 2 possibilités pour y*

Opération d’addition



Opération d'addition

- $P_1=(x_1,y_1)$ et $P_2=(x_2,y_2)$ de E
- Alors la somme $P_1+P_2=P_3=(x_3,y_3)$ est un point de la courbe vérifiant

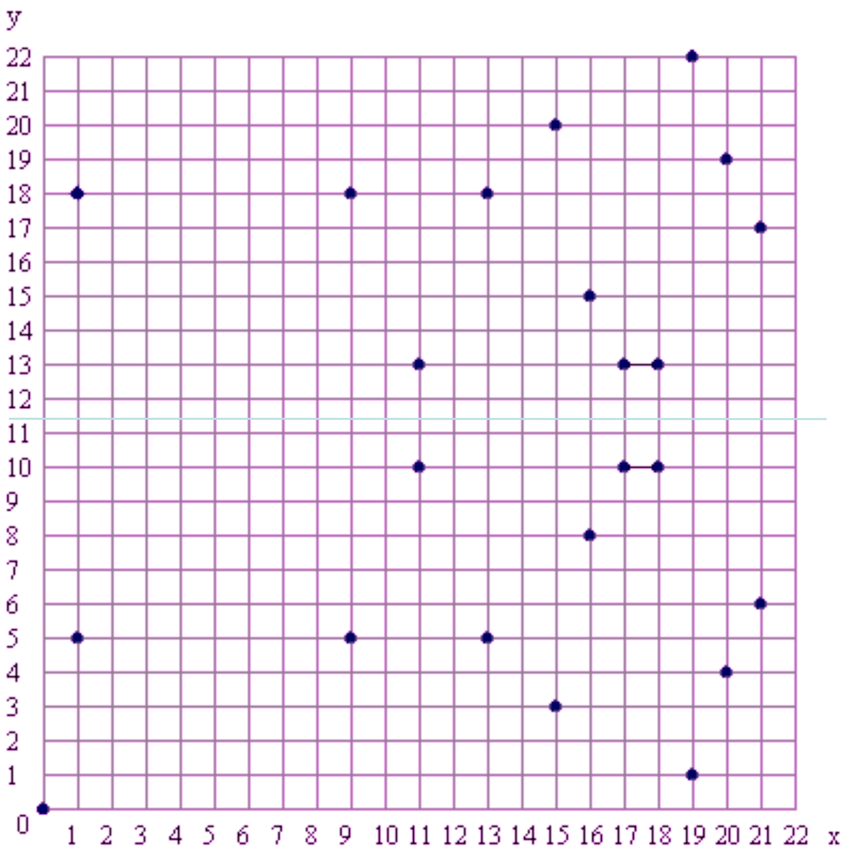
$$\begin{cases} x_3 = ((y_2-y_1)/(x_2-x_1))^2-x_1-x_2 \\ y_3 = y_1+((y_2-y_1)/(x_2-x_1))(x_1-x_3) \end{cases}$$

Courbes elliptiques sur Z_p

- $Z_p = \{0,1,\ldots,p-1\}$
- $y^2 = x^3 + ax + b \bmod p$
  - a et b dans Z_p, x, y sont aussi dans Z_p.
  - $4a^3 + 27b^2 <> 0 \bmod p$.
  - Addition modulo p.
- Example p=23, $Z_p=Z_{23}$. $y^2 = x^3 + x$

(0,0) (1,5) (1,18) (9,5) (9,18) (11,10) (11,13) (13,5) (13,18) (15,3) (15,20) (16,8) (16,15) (17,10) (17,13) (18,10) (18,13) (19,1) (19,22) (20,4) (20,19) (21,6) (21,17)

$y^2 \bmod 23 = x^3 + x \bmod 23$



Elliptic curve equation: $y^2 = x^3 + x$ over F_{23}

Source Certicom

Chiffrement à base d'identité (IBE)

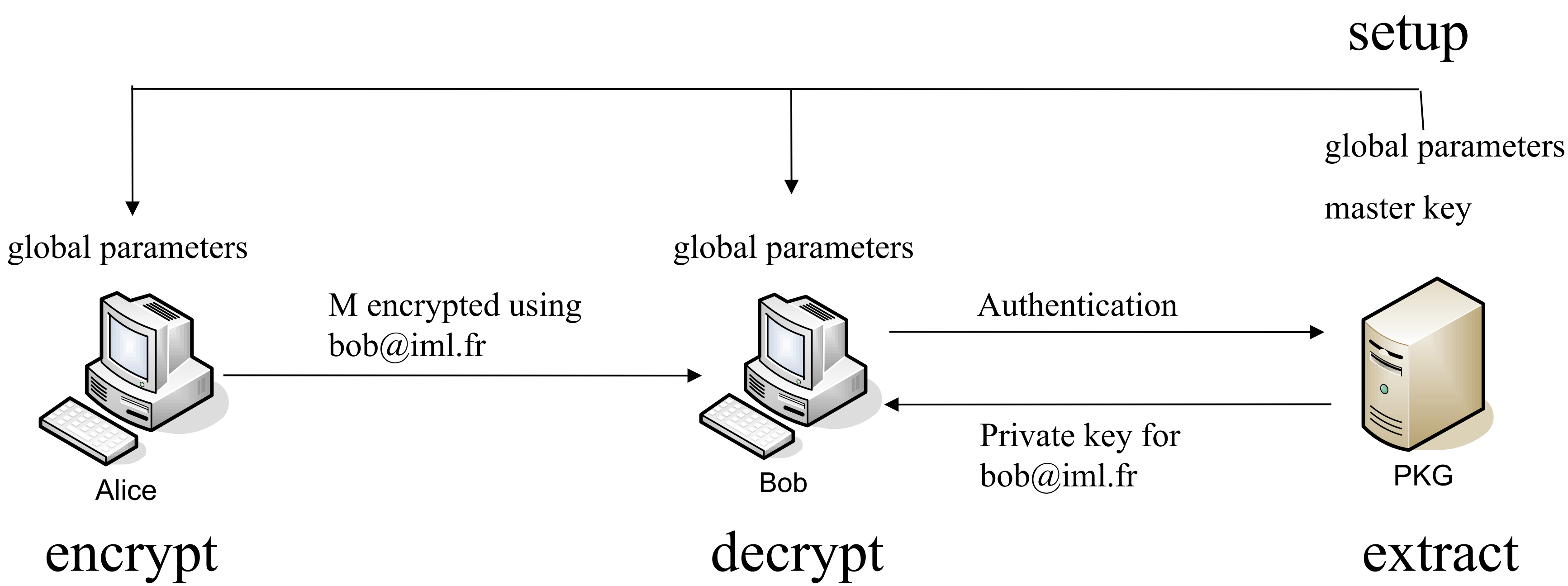
- Proposé par Shamir en 1984
- IBE est un schéma de chiffrement à clé publique
- La clé publique peut être une chaîne arbitraire
- Evite l'utilisation de certificats
- Utilise un tiers de confiance (PKG)

Schémas de chiffrement

- Alice veut envoyer un message à Bob
- Comment trouver la clé publique de Bob?
- IBE permet de simplifier cette étape :
  - La clé publique de Bob se calcule à partir de l'identité de Bob (ex : **bob@iml.fr**)
  - Bob obtient sa clé privée d'un tiers «*private key generator* » (PKG) après s'être authentifié.

IBE

Alice envoie un message à Bob



Propriétés de IBE

- L'infrastructure de clefs publiques est remplacée par les PKGs.
- Les certificats deviennent superflus
- Le chiffrement fonctionne sans communiquer avec le PKG.
- La vérification des signatures fonctionne sans communiquer avec le PKG.

Propriétés de IBE

- PKG peut signer ou déchiffrer tout

mais

- La clé maître et le calcul des clés privées peuvent être distribués sur plusieurs PKGs par un schéma de partage de secrets.

Applications de IBE

- Bob chiffre mail avec pub-key = “alice@hotmail”
  - Utilisation facile : pas besoin de vérifier le certificat d’Alice
  - Bob peut envoyer un mail à Alice avant qu’elle n’ait de clé privée

- Bob chiffre avec pub-key = “alice@hotmail || current-date”
  - Clés privées à vie courte
  - Bob peut envoyer des mails à lire dans le futur

- Credentials: inclus dans la clé publique
  - Chiffre avec : “alice@hotmail || date || clearance=secret”
  - Alice ne peut déchiffrer que si elle détient secret à la date donnée