Arithmétique et Cryptographie, Partie II
Applications cryptographiques des pairings

A. Bonnecaze

Institut de Mathématiques de Luminy (IML)

Oujda, Mai 2009

Historique

- 1993 : Attaques MOV (Weil) puis FR (Tate)
  - Réduction : ECDLP -> DLP sur F_{q^k}
  - Lorsque k (embedding degree) est petit
  - Par ex courbes supersingulières $k \in \{1, \ldots, 6\}$
- 2000 : Utilisation des pairings pour construire des protocoles
  - Echange de clé à trois
  - Cryptographie basée sur l'identité
  - Signatures courtes
  - Clés hiérarchiques
  - ...

Deux types de protocoles :

- Ceux qui ne peuvent pas être construits par d'autres techniques (IdBased Encryption, aggregate signature,...)
- Ceux qui améliorent la fonctionnalité des protocoles existants

A l'heure actuelle

- Déjà des centaines de protocoles proposés !
- Calcul d'un pairing un peu lent mais recherche très active (optimisation, sécurité)

Couplage bilinéaire

- $\mathbb{G}, \mathbb{G}_1$: groupes cycliques finis d'ordre premier p.
- Une application $e : \mathbb{G} \times \mathbb{G} \mapsto \mathbb{G}_1$ est dite un couplage bilinéaire si :
  - Bilinéaire : $\forall u, v \in \mathbb{G}$, et $\forall a, b \in \mathbb{Z}_p$, $e\left(u^a, v^b\right) = e\left(u, v\right)^{ab}$.
  - Non-dégénéré : g générateur $\mathbb{G} \Longrightarrow e\left(g, g\right)$ générateur $\mathbb{G}_1$
  - Calculable : il existe un algorithme polynomial pour calculer $e\left(u, v\right)$ pour $\forall u, v \in \mathbb{G}$.
- Les couplages de Weil [JN01] et de Tate [BLS04,BGHS07] sont les plus utilisés.

Echange de clés à trois : A, B, C (Joux 2000)

A, B, C ont les clés secrètes resp $a, b, c \in Z_p$

- A envoie aP à B et C
- B envoie bP à A et C
- C envoie cP à A et B

- A calcule $k_A = e(bP, cP)^a$
- B calcule $K_B = e(aP, cP)^b$
- C calcule $K_C = e(aP, bP)^c$

La clé commune est $K_{ABC} = K_A = K_B = K_C = e(P, P)^{abc}$

- Sécurité :
  sûr contre une attaque passive sous l'hypothèse que le problème BDH est dur (connaissant (P, aP, bP, cP) déterminer $e(P, P)^{abc}$)
- Ce protocole se généralise à n parties

Echange de clés à trois : A, B, C (Joux 2000)

A, B, C ont les clés secrètes resp $a, b, c \in Z_p$

- A envoie aP à B et C
- B envoie bP à A et C
- C envoie cP à A et B

- A calcule $k_A = e(bP, cP)^a$
- B calcule $K_B = e(aP, cP)^b$
- C calcule $K_C = e(aP, bP)^c$

La clé commune est $K_{ABC} = K_A = K_B = K_C = e(P, P)^{abc}$

- Sécurité :
  sûr contre une attaque passive sous l'hypothèse que le problème BDH est dur (connaissant (P, aP, bP, cP) déterminer $e(P, P)^{abc}$)
- Ce protocole se généralise à n parties

Chiffrement basé sur l'identité

Un problème majeur en cryptographie : la gestion des clés

- Comment obtenir la (vraie) clé publique de Bob ?
- BD de clés publiques peuvent être falsifiées (pb d'intégrité)
- Certificats ? oui mais il faut une autorité de certification digne de confiance pour toutes les parties
  - Actuellement il existe beaucoup d'AC. Laquelle choisir ?
- Le nombre de clés publiques augmente donc aussi le nombre de certificats et de CA...

Chiffrement basé sur l'identité

En 1984 Shamir propose que la clé publique soit directement liée à l'identité de la personne

- Par exemple son email avec éventuellement des informations supplémentaires :
  Exemple : "Bob@.acrypta.fr"
- Comment réaliser un tel schéma de chiffrement ?
- Open problem jusqu'au début des années 2000.
- Boneh & Franklin en 2001 réalisent le premier protocole en utilisant des pairings

Schéma de chiffrement

Alice veut envoyer un message chiffré à Bob

- Alice chiffre son message en utilisant
  - l'identité de Bob, par exemple "bob@acrypta.fr"
  - la clé publique du PKG (Private Key Generator)
- Bob obtient sa clé privée auprès du PKG après s'être authentifié
- La clé privée de Bob dépend
  1. de l'identité de Bob : "bob@acrypta.fr"
  2. de la clé "Master" du PKG (clé secrète seulement connue du PKG)

Schéma de chiffrement

Avantages :

- Pour envoyer un message à une personne, Alice n'a besoin que de l'identité de la personne et de la clé publique du PKG.
- Pas de PKI (pas de certificats)
- Nouvelles fonctionnalités

Désavantages :

- Le PKG connaît la clé privée de Bob (Key escrow). Mais possibilité de construire un schéma distribué ou les PKG ne connaissent qu'un fragment de la clé.
- Bob doit s'authentifier pour obtenir sa clé privée (il le fait une fois pour toute)
- Le PKG doit utiliser un canal sûr pour envoyer à Bob sa clé
- Alice doit obtenir les paramètres du PKG avant de pouvoir chiffrer le message

Protocole

Setup
- Clé "Master" du PKG : choisir $s \in_R Z_q^*$
- Clé publique du PKG : $P_{pub} = sP$
- Fonction de hachage (map-to-point) $H_1 : \{0,1\}^* \to \mathbb{G}$
- Fonction de hachage $H_2 : \mathbb{G}_1 \to \{0,1\}^n$
- n est la longueur du message M en bits

Extract Connaissant l'identité publique du l'utilisateur : $ID \in \{0,1\}^*$
- Calculer sa clé publique : $Q_{ID} = H_1(ID) \in \mathbb{G}$
- Calculer sa clé privée : $S_{ID} = sQ_{ID}$

Encrypt Choisir $r \in_R Z_q^*$, le chiffré C est

$$C = \langle rP, M \oplus H_2(g_{ID}^r) \rangle$$

où $g_{ID} = e(Q_{ID}, P_{pub})$

Decrypt Connaissant $C = \langle U, V \rangle$, calculer

$$V \oplus H_2(e(S_{ID}, U))$$

Propriétés de IBE

Sécurité

- Utilise l'hypothèse que BDH est dur
- Peut devenir IND-CCA2 dans le modèle de l'oracle aléatoire

Efficacité

- Setup : 1 scalar multiplication scalaire dans $\mathbb{G}$
- Extract : 1 opération de hachage Map-to-point ; 1 multiplication scalaire dans $\mathbb{G}$.
- Encrypt : 1 opération de hachage Map-to-point ; 1 multiplication scalaire dans $\mathbb{G}$; 1 fonction de hachage (H_2) ; 1 XOR ; 1 pairing ; 1 exponentiation dans $\mathbb{G}_1$.
- Decrypt : 1 fonction de hachage (H_2) ; 1 XOR ; 1 pairing.

Fonctionnalités intéressantes du chiffrement

- PEKS : Chiffrement avec mot clés
- Chiffrement hiérarchique
- Révocation de clé publique
  - "Bob@mathcrypto.fr || 2009" clé publique de Bob pour l'année 2009
  - Alice n'a pas besoin d'un nouveau certificat à chaque fois que Bob met à jour sa clé privée (ici chaque année)
- Si Alice connaît la clé privée de Bob en 2009, elle ne peut déchiffrer les messages des années précédentes
- Alice peut envoyer un message à Bob que celui-ci ne pourra déchiffrer que dans le futur.
- Délégation de service : Alice chiffre des mails à Bob en utilisant le sujet du mail comme clé de chiffrement : "Bob@Dell.fr || 2009 || vendeur" Bob donne la clé privée correspondant à certains collaborateurs

PEKS

Exemple de chiffrement avec recherche de mots clé

Alice veut recevoir ses mails

- "urgents" sur son téléphone portable,
- "privés" sur son PDA,
- les autres mails sur son PC

Propriétés

- Les mails sont chiffrés
- Les mails sont accompagnés d'un ou plusieurs mots clé chiffrés
- Alice envoie les instructions de routage accompagnés d'une clé secrète à son gateway
- Le gateway d'Alice route correctement les mails sans pouvoir déchiffrer les mots clé
- aussi appelé SPKE (Searchable public-key encryption)

PEKS

Pour envoyer un message M avec mots clé $W_1, \ldots, W_n$, Bob envoie

$$E_{A_{pub}}(M) || PEKS(A_{pub}, W_1) || \ldots || PEKS(A_{pub}, W_n)$$

- Le gateway teste les mots clé grâce au trapdoor que Alice lui a donné
- Le gateway peut être par exemple le serveur IMAP d'Alice

PEKS : protocole

- KeyGen clé privée : $s \in_R \mathbb{Z}_q^*$, clé publique : $P_{pub} = sP$. Ensemble des mots clé : K. Deux fonctions de hachage $H_1 : K \to \mathbb{G}$, $H_2 : \mathbb{G}_1 \to \mathbb{Z}_q^*$
- PEKS A partir d'un mot clé W et une clé publique P_{pub}, choisir $r \in_R \mathbb{Z}_q^*$ et calcule
$$\langle rP, H_2(e(H_1(W), P_{pub})^r)\rangle = \langle U, V\rangle$$
- Trapdoor étant donné un mot clé W et la clé s, calcule $T_w = sH_1(W)$
- Test étant donné T_w, $\langle U, V\rangle$ et la clé publique P_{pub}, teste si
$$V = H_2(e(T_w, U))$$

Id-Based forward-secure Encryption Scheme

- Key generation : détermine une paire de clé privée/publique. La clé privée est notée d_0 la clé publique e
- Encryption algorithm : à partir de la clé publique, du message m et du temps t, calcule le chiffré de m au temps t :

$$C = E_e(m, t)$$

- Update mechanism : pour chaque nouvelle période de temps t, le déchiffrement utilise une nouvelle clé privée d_t. La clé d_t doit se calculer facilement a partir de d_0 et du temps courant t. $d_t = UPDATE(t, D_{t-1})$, UPDATE étant une foncion à sens unique.
- Decryption : pour déchiffrer m qui a été envoyé au temps t, il faut utiliser la clé d_t :

$$D_{d_t}(E_e(m, t)) = m$$

Schémas de signature

- La signature numérique est un mécanisme permettant d'authentifier l'auteur d'un document électronique et de garantir son intégrité.
- La plupart des schémas existants sont prouvés sûrs dans le modèle de l'oracle aléatoire.
- Des schémas sont prouvés sûrs dans le modèle standard (c-à-d. sans oracles aléatoires)

- Il existe des signatures courtes (environ 160 bits)
- signatures agrégées et multisignatures
- signatures de groupes
- signatures en aveugle
- ...

Signatures courtes : BLS (Boneh, Lynn, Shacham, 2001)

Clé privée $s \in_R \mathbb{Z}_q^*$; Clé publique $y = sP$
$\mathbb{G} = \langle P \rangle$; fonction de hachage (map-to-point) $H_1 : \{0,1\}^* \rightarrow \mathbb{G}$

$$\sigma(m) = s.H(m)$$

Vérification la signature est aceptée si

$$e(P, \sigma) = e(y, H(m))$$

Sécurité basé sur CDH (dans le modèle de l'oracle aléatoire)

BLS (suite)

Efficacité

- signature : 1 hash, 1 mutiplication par scalaire dans $\mathbb{G}$
- vérification : 2 pairings
- la taille de la signature est de 154 bits avec une courbe elliptique sur $\mathbb{F}_{3^{97}}$
- prend de l'ordre de la milliseconde pour les deux opérations
- Remarque : la fonction de hachage est l'opération la plus lente

Multisignature (Boldyreva 2003)

Plusieurs personnes veulent signer un même message.
Clé privée $s_i \in_R \mathbb{Z}_q^*$; Clé publique $y_i = s_i P$

$$\sigma_i(m) = s_i.H(m)$$

$$\sigma(m) = \sum_i \sigma_i = H(m).\sum_i s_i$$

La signature est $(\sigma, y = \sum_i y_i,$ liste des signataires)
Vérification la signature est aceptée si

$$e(P, \sigma) = e(y, H(m))$$

Même sécurité que BLS

Signatures à seuil (Boldyreva 2003)

Groupe de n personnes. Tout sous-groupe de t personnes peut signer le message. La clé privée "master" est $s = \sum_i s_i L_i$ où les L_i sont les coefficients de Lagrange La clé public "master" est $y = x.P$
La clé public $y_i = x_i.P$

- Chacun des t signataires signe : $\sigma_i(m) = s_i.H(m)$
- $\sigma(m) = \sum_i \sigma_i.L_i$
- Vérification la signature est aceptée si

$$e(P, \sigma) = e(y, H(m))$$

Signatures agrégées

Plusieurs signataires veulent signer différents messages pour produire une signature de petite taille

- Chaque signataire i signe : $\sigma_i(m) = s_i.H(m)$
- $\sigma(m) = \sum_i \sigma_i$
- Vérification la signature est aceptée si

$$e(P,\sigma) = \prod_{i=1}^{n} e(y_i, H(m_i))$$

Signature en anneau (signature anonyme)

Un membre d'un groupe veut créer une signature avec sa clé privée. On peut vérifier que la signature a été créée par un membre du groupe.
Mais on ne sait pas qui a signé.

- Soit s le signataire, x_i/y_i les clés privées/publiques, $i \in [1 \ldots n]$.
- Le signataire choisit $r_i \in_R \mathbb{Z}_q^*$ pour $i \neq s$
- Le signataire calcule $\sigma_i = r_i.P$ et $\sigma_s = (H(m) - \sum_{i \neq s} y_i.r_i).(1/x_s)$
- La signature est $(\sigma_1, \ldots, \sigma_n)$
- Vérification la signature est aceptée si

$$e(P, H(m)) = \prod_{i=1}^{n} e(y_i, \sigma_i)$$

Implantation

- Il existe une librairie gratuite maintenue par l'université de Stanford : The Pairing-Based Cryptography Library, PBC
- http ://crypto.stanford.edu/pbc/
- Cette librairie est écrite en C
- Elle permet d'implanter très facilement des protocoles

Avenir de la cryptographie basée sur les pairings

- Pas encore utilisée dans des applications industrielles
- Implantation dans des cartes à puces
- La recherche est très dynamique aussi bien pour le côté mathématique que cryptographique
- Formes multilinéaires ?