Round-Optimal Privacy-Preserving Protocols with Smooth Projective Hash Functions

David Pointcheval

Joint work with Olivier Blazy and Damien Vergnaud

Ecole Normale Supérieure

January 13th, 2012
Grenoble

Outline

1. Blind Signatures
2. Cryptographic Tools
3. Smooth Projective Hash Functions
4. Oblivious Signature-Based Encryption

Outline

1. Blind Signatures
2. Cryptographic Tools
3. Smooth Projective Hash Functions
4. Oblivious Signature-Based Encryption

Outline

1. Blind Signatures
2. Cryptographic Tools
3. Smooth Projective Hash Functions
4. Oblivious Signature-Based Encryption

Outline

1 Blind Signatures

2 Cryptographic Tools

3 Smooth Projective Hash Functions
- Intuition
- Applications

4 Oblivious Signature-Based Encryption

Smooth Projective Hash Functions [Cramer, Shoup, 2002]

Family of Hash Function H

Let $\{H\}$ be a family of functions:
- X, domain of these functions
- L, subset (a language) of this domain

such that, for any point x in L, $H(x)$ can be computed by using
- either a *secret* hashing key hk: $H(x) = \text{Hash}_L(\text{hk}; x)$;
- or a *public* projected key hp: $H(x) = \text{ProjHash}_L(\text{hp}; x, w)$

While the former works for all points in the domain X,
the latter works for $x \in L$ only, and requires a witness w to this fact.
There is a public mapping that converts the hashing key hk into the projected key hp: $\text{hp} = \text{ProjKG}_L(\text{hk})$

Properties

For any $x \in X$, $H(x) = \text{Hash}_L(\text{hk}; x)$
For any $x \in L$, $H(x) = \text{ProjHash}_L(\text{hp}; x, w)$ $\quad w$ witness that $x \in L$

Smoothness

For any $x \notin L$, $H(x)$ and hp are independent

Pseudo-Randomness

For any $x \in L$, $H(x)$ is pseudo-random, without a witness w

The latter property requires L to be a hard-partitioned subset of X:

Hard-Partitioned Subset

L is a hard-partitioned subset of X if it is computationally hard to distinguish a random element in L from a random element in $X \setminus L$

Element-Based Projection

Initial Definition [Cramer, Shoup, 2002]

The projected key hp depends on the hashing key hk only:
$\text{hp} = \text{ProjKG}_L(\text{hk})$

New Definition [Gennaro, Lindell, 2003]

The projected key hp depends on the hashing key hk, and x:
$\text{hp} = \text{ProjKG}_L(\text{hk}; x)$

$\rightarrow$ More applications

Examples

DH Language [Cramer, Shoup, 2002]

$L_{g,h} = \{u, v\}$ such that (g, h, u, v) is DH tuple:
there exists r such that $u = g^r$ and $v = h^r$

→ Public-key Encryption with IND-CCA Security

Commitment [Gennaro, Lindell, 2003]

$L_{pk,m} = \{c\}$ such that c is a commitment of m
using public parameter pk:
there exists r such that $c = \mathrm{com}_{pk}(m; r)$
where com is the committing algorithm

→ Password-Authenticated Key Exchange in the Standard Model

Examples (Con'd)

Labeled Encryption [Canetti, Halevi, Katz, Lindell, MacKenzie, 2005]

$L_{pk,(\ell,m)} = \{c\}$ such that c is an encryption of m
with label ℓ, under the public key pk:
there exists r such that $c = \mathcal{E}^{\ell}_{pk}(m; r)$
where $\mathcal{E}$ is the encryption algorithm

→ PAKE in the UC Framework (passive corruptions)

Extractable/Equivocable Commitment [Abdalla, Chevalier, Pointcheval, 2009]

$L_{pk,m} = \{c\}$ such that c is a equivocable/extractable commitment of m
using public parameter pk

→ PAKE in the UC Framework secure against Active Corruptions

Smooth Projective HF Family for ElGamal

The CRS: $\rho = (G, q, g, pk = h)$

Language: $L = L_{(\mathbf{EG}^+,\rho),M} = \{C = (u_1, e) = \mathrm{EG}^+_{pk}(M; r), r \stackrel{\$}{\leftarrow} \mathbb{Z}_q\}$

- L is a hard-partitioned subset of $X = G^2$, under the semantic security of the ElGamal encryption scheme (DDH assumption)
- the random r is the witness to L-membership

Algorithms

- $\mathrm{HashKG}(M) = \mathrm{hk} = (\gamma_1, \gamma_3) \stackrel{\$}{\leftarrow} \mathbb{Z}_q \times \mathbb{Z}_q$
- $\mathrm{Hash}(\mathrm{hk}; M, C) = (u_1)^{\gamma_1}(eg^{-M})^{\gamma_3}$
- $\mathrm{ProjKG}(\mathrm{hk}; M, C) = \mathrm{hp} = (g)^{\gamma_1}(h)^{\gamma_3}$
- $\mathrm{ProjHash}(\mathrm{hp}; M, C; r) = (\mathrm{hp})^r$

Certification of Public Keys [Abdalla, Chevalier, Pointcheval, 2009]

For the certification Cert of an ElGamal public key $y = g^x$, in most of the protocols, the simulator needs to be able to extract the secret key:

Classical Process

- the user U sends his public key $y = g^x$;
- U and the authority A run a ZK proof of knowledge of x
- if convinced, A generates and sends the certificate Cert for y

For extracting x, the reduction requires a rewinding
(that is not always allowed: *e.g.*, in the UC Framework)

Certification of Public Keys

[Abdalla, Chevalier, Pointcheval, 2009]

For the certification Cert of an ElGamal public key $y = g^x$, in most of the protocols, the simulator needs to be able to extract the secret key:

New Process

The user U and the authority A use a smooth projective hash system for L: $y = g^x$ and $C = \mathcal{E}_{pk}(x; r)$ contain the same x

- U sends $y = g^x$, with $C = \mathcal{E}_{pk}(x; r)$, for a random r;
- A generates
  - a hashing key $\mathsf{hk} \overset{\$}{\leftarrow} \mathsf{HashKG}()$,
  - the corresponding projected key on (y, C),
  - the hash value $\mathsf{Hash} = \mathsf{Hash}(\mathsf{hk}; (y, C))$

  and sends hp along with $\mathsf{Cert} \oplus \mathsf{Hash}$;
- U computes $\mathsf{Hash} = \mathsf{ProjHash}(\mathsf{hp}; (y, C), r)$, and gets Cert.

Blind Signature

[Blazy, Fuchsbauer, Pointcheval, Vergnaud, 2011]

In order to get M blindly signed under a Waters' signature:

Previous Process

- the user U encrypts M into C_1, and $\mathcal{F}(M)$ into C_2;
- U produces a Groth-Sahai NIZK that
  C_1 and C_2 contain the same M
- if convinced, A generates a signature on C_2
- granted the commutativity, U decrypts it
  into a Waters' signature of M,
  and eventually re-randomizes the signature

Such a NIZK requires $9\ell + 24$ group elements

Blind Signature

[Blazy, Pointcheval, Vergnaud, 2012]

In order to get M blindly signed under a Waters' signature:

Previous Process

The user U and the authority A use a smooth projective hash system for L: $C_1 = \mathcal{E}_{pk_1}(M; r)$ and $C_2 = \mathcal{E}_{pk_2}(\mathcal{F}(M); s)$ contain the same M

- U sends encryptions of M, into C_1, and $\mathcal{F}(M)$, into C_2;
- A generates
  - a signature σ on C_2,
  - masks it using $\mathsf{Hash} = \mathsf{Hash}(\mathsf{hk}; (C_1, C_2))$
- U computes $\mathsf{Hash} = \mathsf{ProjHash}(\mathsf{hp}; (C_1, C_2), (r, s))$, and gets σ. Granted the commutativity, U decrypts it into a Waters' signature of M, and eventually re-randomizes it

Such a protocol requires $8\ell + 12$ group elements in total

Outline

1 Blind Signatures
2 Cryptographic Tools
3 Smooth Projective Hash Functions
4 Oblivious Signature-Based Encryption
- Example
- Security Notions

Linear Encryption

In a group $\mathbb{G}$ of order p, with a generator g,
and a bilinear map $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$

Linear Encryption

[Boneh, Boyen, Shacham, 2004]

- *EKeyGen*: $dk = (x_1, x_2) \stackrel{\$}{\leftarrow} \mathbb{Z}_p^2$, $pk = (X_1 = g^{x_1}, X_2 = g^{x_2})$;
- *Encrypt*$(pk = (X_1, X_2), m; (r_1, r_2))$, for $m \in \mathbb{G}$ and $(r_1, r_2) \stackrel{\$}{\leftarrow} \mathbb{Z}_p^2$
  $\to \quad c = (c_1 = X_1^{r_1}, c_2 = X_2^{r_2}, c_3 = g^{r_1+r_2} \cdot m)$;
- *Decrypt*$(dk = (x_1, x_2), c = (c_1, c_2, c_3)) \quad \to \quad m = c_3 / c_1^{1/x_1} c_2^{1/x_2}$.

Re-Randomization

- *Random*$_{\mathcal{E}}(pk = (X_1, X_2), c = (c_1, c_2, c_3); (r_1', r_2'))$, for $(r_1', r_2') \stackrel{\$}{\leftarrow} \mathbb{Z}_p^2$
  $\to \quad c' = (c_1' = c_1 \cdot X_1^{r_1'}, c_2' = c_2 \cdot X_2^{r_2'}, c_3' = c_3 \cdot g^{r_1'+r_2'})$.

Waters Signature

In a group $\mathbb{G}$ of order p, with a generator g,
and a bilinear map $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$

Waters Signature

[Waters, 2005]

For a message $M = (M_1, \ldots, M_k) \in \{0,1\}^k$,
we define $F = \mathcal{F}(M) = u_0 \prod_{i=1}^k u_i^{M_i}$, where $\vec{u} = (u_0, \ldots, u_k) \stackrel{\$}{\leftarrow} \mathbb{G}^{k+1}$.
For an additional generator $h \stackrel{\$}{\leftarrow} \mathbb{G}$.

- *SKeyGen*: $vk = X = g^x$, $sk = Y = h^x$, for $x \stackrel{\$}{\leftarrow} \mathbb{Z}_p$;
- *Sign*$(sk = Y, F; s)$, for $M \in \{0,1\}^k$, $F = \mathcal{F}(M)$, and $s \stackrel{\$}{\leftarrow} \mathbb{Z}_p$
  $\to \quad \sigma = (\sigma_1 = Y \cdot F^s, \sigma_2 = g^{-s})$;
- *Verif*$(vk = X, M, \sigma = (\sigma_1, \sigma_2))$ checks whether
  $$e(g, \sigma_1) \cdot e(F, \sigma_2) = e(X, h).$$

Waters Signature on a Linear Ciphertext: Idea

We define $F = \mathcal{F}(M) = u_0 \prod_{i=1}^k u_i^{M_i}$, and encrypt it

$$c = (c_1 = X_1^{r_1}, c_2 = X_2^{r_2}, c_3 = g^{r_1+r_2} \cdot F)$$

- *KeyGen*: $vk = X = g^x$, $sk = Y = h^x$, for $x \stackrel{\$}{\leftarrow} \mathbb{Z}_p$
  $dk = (x_1, x_2) \stackrel{\$}{\leftarrow} \mathbb{Z}_p^2$, $pk = (X_1 = g^{x_1}, X_2 = g^{x_2})$
- *Sign*$((X_1, X_2), Y, c; s)$, for $c = (c_1, c_2, c_3)$
  $\to \quad \sigma = (\sigma_1 = Y \cdot c_3^s, \sigma_2 = (c_1^s, c_2^s), \sigma_3 = (g^s, X_1^s, X_2^s))$
- *Verif*$((X_1, X_2), X, c, \sigma)$ checks $\quad e(g, \sigma_1) = e(X, h) \cdot e(\sigma_{3,0}, c_3)$
  $$e(\sigma_{2,0}, g) = e(c_1, \sigma_{3,0}) \qquad e(\sigma_{2,1}, g) = e(c_2, \sigma_{3,0})$$
  $$e(\sigma_{3,1}, g) = e(X_1, \sigma_{3,0}) \qquad e(\sigma_{3,2}, g) = e(X_2, \sigma_{3,0})$$

σ_3 is needed for ciphertext re-randomization

Re-Randomization of Ciphertext

$$c = (c_1 = X_1^{r_1}, \quad c_2 = X_2^{r_2}, \quad c_3 = g^{r_1+r_2} \cdot F \quad)$$
$$\sigma = (\sigma_1 = Y \cdot c_3^s, \quad \sigma_2 = (c_1^s, c_2^s), \quad \sigma_3 = (g^s, X_1^s, X_2^s) \quad)$$

after re-randomization by (r_1', r_2')

$$c' = (c_1' = c_1 \cdot X_1^{r_1'}, \quad c_2' = c_2' \cdot X_2^{r_2'}, \quad c_3' = c_3 \cdot g^{r_1'+r_2'} \quad)$$
$$\sigma' = (\sigma_1' = \sigma_1 \cdot \sigma_{3,0}^{r_1'+r_2'}, \quad \sigma_2' = (\sigma_{2,0} \cdot \sigma_{3,1}^{r_1'}, \sigma_{2,1} \cdot \sigma_{3,2}^{r_2'}), \quad \sigma_3' = \sigma_3 \quad)$$

Anybody can publicly re-randomize c into c'
with additional random coins (r_1', r_2'),
and adapt the signature σ of c into σ' of c'

Unforgeability under Chosen-Ciphertext Attacks

Chosen-Ciphertext Attacks

The adversary is allowed to ask any valid ciphertext of his choice to the signing oracle

Because of the re-randomizability of the ciphertext-signature, we cannot expect resistance to existential forgeries, but we should allow a restricted malleability only:

Forgery

A valid ciphertext-signature pair, so that the plaintext is different from all the plaintexts in the ciphertexts sent to the signing oracle

Unforgeability

From a valid ciphertext-signature pair:

$$c = (c_1 = X_1^{r_1}, c_2 = X_2^{r_2}, c_3 = g^{r_1+r_2} \cdot F)$$
$$\sigma = (\sigma_1 = Y \cdot c_3^s, \sigma_2 = (c_1^s, c_2^s), \sigma_3 = (g^s, X_1^s, X_2^s))$$

and the decryption key (x_1, x_2), one extracts

$$\begin{array}{llll} F = & c_3/(c_1^{1/x_1} c_2^{1/x_2}) & & \\ \Sigma = (& \Sigma_1 = \sigma_1/(\sigma_{2,0}^{1/x_1}\sigma_{2,1}^{1/x_2}), & \Sigma_2 = \sigma_{3,0}) \\ = (& = Y \cdot F^s & = g^s) \end{array}$$

Security of Waters signature is for a pair (M, Σ)

$\rightarrow$ needs of a proof of knowledge Π_M of M in $F = \mathcal{F}(M)$

bit-by-bit commitment of M and Groth-Sahai proof

Chosen-Message Attacks

From a valid ciphertext $c = (c_1 = X_1^{r_1}, c_2 = X_2^{r_2}, c_3 = g^{r_1+r_2} \cdot F)$, and the additional proof of knowledge of M, one extracts M and asks for a Waters signature:

$$\Sigma = (\Sigma_1 = Y \cdot F^s, \Sigma_2 = g^s)$$

In this signature, the random coins s are unknown, we thus need to know the coins in c

$\rightarrow$ needs of a proof of knowledge Π_r of r_1, r_2 in c

bit-by-bit commitment of r_1, r_2 and Groth-Sahai proof

From the random coins r_1, r_2 (and the decryption key):

$$\begin{array}{llll} \sigma = (\sigma_1 = \Sigma_1 \cdot \Sigma_2^{r_1+r_2}, & \sigma_2 = (\Sigma_2^{x_1 r_1}, \Sigma_2^{x_2 r_2}), & \sigma_3 = (\Sigma_2, \Sigma_2^{r_1}, \Sigma_2^{r_2}) &) \\ = Y \cdot c_3^s, & = (c_1^s, c_2^s), & = (g^s, X_1^s, X_2^s) & \end{array}$$

Security

Chosen-Ciphertext Attacks

A valid ciphertext $C = (c_1, c_2, c_3, \Pi_M, \Pi_r)$ is a

- ciphertext $c = (c_1, c_2, c_3)$
- a proof of knowledge Π_M of the plaintext M in $F = \mathcal{F}(M)$
- a proof of knowledge Π_r of the random coins r_1, r_2

From such a ciphertext and the decryption key (x_1, x_2), and a Waters signing oracle, one can generate a signature on C

Forgery

From a valid ciphertext-signature pair (C, σ), where C encrypts M, one can generate a Waters signature on M

Security

- From the Waters signing oracle,
  we answer Chosen-Ciphertext Signing queries
- From a Forgery, we build a Waters Existential Forgery

Security Level

Since the Waters signature is EF-CMA under the *CDH* assumption, our signature on randomizable ciphertext is Unforgeable
against Chosen-Ciphertext Attacks
under the *CDH* assumption

Properties

Proofs

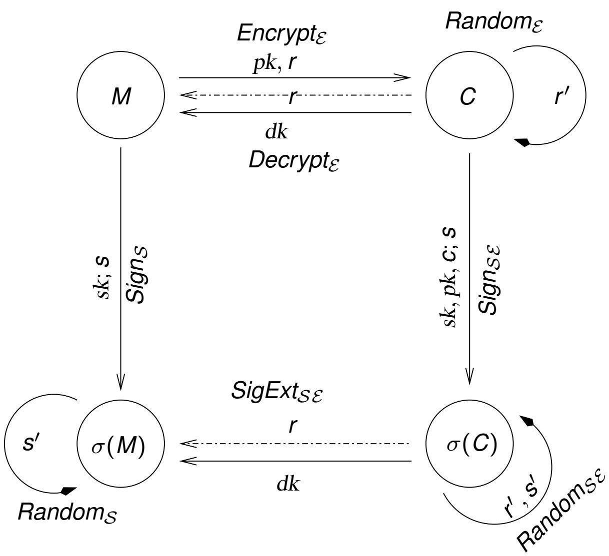
Since we use the Groth-Sahai methodology for the proofs Π_M and Π_r

- in case of re-randomization of c, one can adapt Π_M and Π_r
- because of the need of M, but also r_1 and r_2 in the simulation, we need bit-by-bit commitments:
  - M can be short (ℓ bit-long)
  - r_1 and r_2 are random in $\mathbb{Z}_p$
  - $\rightarrow$ C is large!

Efficiency

We can improve efficiency: shorter signatures

Randomizable Commutative Signature/Encryption



Conclusion

Randomizable Commutative Signature/Encryption

Various Applications

- non-interactive receipt-free electronic voting scheme
- (fair) blind signature

Security relies on the *CDH* and the *DLin* assumptions
For an ℓ-bit message, ciphertext-signature:
$9\ell + 24$ group elements

A more efficient variant with asymmetric pairing
on the CDH^* and the *SXDH* assumptions
Ciphertext-signature: $6\ell + 7$ group elements in $\mathbb{G}_1$
and $6\ell + 5$ group elements in $\mathbb{G}_2$