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

David Pointcheval

Joint work with Olivier Blazy and Damien Vergnaud

Ecole Normale Supérieure

Grenoble – January 13th, 2012

Outline

Conditional Actions

An authority, or a server, may accept to process a request under some conditions only:

- Certification of public key: if the associated secret key is known
- Transmission of private information: if the receiver owns a credential

Blind signature on a message: if the user knows the message (for the security proof)

Certification of Public Keys: ZKPoK

In the registered key setting, a user can ask for the certification of a public key *pk*, but if he knows the associated secret key *sk* only:

With an Interactive Zero-Knowledge Proof of Knowledge

- the user *U* sends his public key *pk*;
- *U* and the authority *A* run a ZK proof of knowledge of *sk*
- if convinced, *A* generates and sends the certificate Cert for *pk*

For extracting *sk* (required in some security proofs), the reduction has to make a rewind (that is not always allowed: *e.g.*, in the UC Framework)

Certification of Public Keys: ZK and NIZK Proofs

In the registered key setting, a user can ask for the certification of a public key *pk*, but if he knows the associated secret key *sk* only:

With an Interactive Zero-Knowledge Proof of Membership

- the user U sends his public key pk, and an encryption C of sk;
- U and the authority A run a ZK proof
  that C contains the secret key sk associated to pk
- if convinced, A generates and sends the certificate Cert for pk

With a Non-Interactive Zero-Knowledge Proof of Membership

- the user U sends his public key pk, and an encryption C of sk
  together with a NIZK proof
  that C contains the secret key sk associated to pk
- if convinced, A generates and sends the certificate Cert for pk

Certification of Public Keys: SPHF

[Abdalla, Chevalier, Pointcheval, 2009]

In the registered key setting, a user can ask for the certification of a public key *pk*, but if he knows the associated secret key *sk* only:

With a Smooth Projective Hash Function

The user U and the authority A use a smooth projective hash system for L: pk and $C = \mathcal{E}_{pk'}(sk; r)$ are associated to the same sk

- the user U sends his public key pk, and an encryption C of sk;
- A generates the certificate Cert for pk, and sends it,
  masked by $\mathsf{Hash} = \mathsf{Hash}(\mathsf{hk}; (pk, C))$;
- U computes $\mathsf{Hash} = \mathsf{ProjHash}(\mathsf{hp}; (pk, C), r))$, and gets Cert.

Implicit proof of knowledge of *sk*

Smooth Projective Hash Functions

[Cramer, Shoup, 2002]

Definition [Cramer, Shoup, 2002] [Gennaro, Lindell, 2003]

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: $H(x) = \mathsf{Hash}_L(\mathsf{hk}; x)$;
- or a *public* projected key hp: $H(x) = \mathsf{ProjHash}_L(\mathsf{hp}; x, w)$

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.

Public mapping $\mathsf{hk} \mapsto \mathsf{hp} = \mathsf{ProjKG}_L(\mathsf{hk}, x)$

Properties

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

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 $v = h^r$

$\rightarrow$ Public-key Encryption with IND-CCA Security

Algorithms

- $\mathsf{HashKG}() = \mathsf{hk} = (\gamma_1, \gamma_3) \stackrel{\$}{\leftarrow} \mathbb{Z}_q \times \mathbb{Z}_q$
- $\mathsf{ProjKG}(\mathsf{hk}) = \mathsf{hp} = g^{\gamma_1} h^{\gamma_3}$

$\mathsf{Hash}(\mathsf{hk}, (u,v)) = u^{\gamma_1} v^{\gamma_3} = \mathsf{hp}^r = \mathsf{ProjHash}(\mathsf{hp}, (u,v); r)$

Examples (Con'd)

Commitment/Encryption [Gennaro, Lindell, 2003]

$L_{pk,m} = \{c\}$ such that c is an encryption of m under pk:
there exists r such that $c = \mathcal{E}_{pk}(m; r)$

$\rightarrow$ Password-Authenticated Key Exchange in the Standard Model

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

$L_{pk,(\ell,m)} = \{c\}$ such that c is an encryption of m under pk, with label ℓ

$\rightarrow$ 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

$\rightarrow$ PAKE in the UC Framework secure against Active Corruptions

Assumptions: CDH and DLin

$\mathbb{G}$ a cyclic group of prime order p (with or without bilinear map).

Definition (The Computational Diffie-Hellman problem (*CDH*))

For any generator $g \stackrel{\$}{\leftarrow} \mathbb{G}$, and any scalars $a, b \stackrel{\$}{\leftarrow} \mathbb{Z}_p^*$,
given (g, g^a, g^b), compute g^{ab}.

Decisional variant easy if a bilinear map is available.

Definition (Decision Linear Problem (*DLin*))

For any generator $g \stackrel{\$}{\leftarrow} \mathbb{G}$, and any scalars $a, b, x, y, c \stackrel{\$}{\leftarrow} \mathbb{Z}_p^*$,
given $(g, g^x, g^y, g^{xa}, g^{yb}, g^c)$, decide whether $c = a + b$ or not.

Equivalently, given a reference triple $(u = g^x, v = g^y, g)$
and a new triple $(U = u^a = g^{xa}, V = v^b = g^{yb}, T = g^c)$,
decide whether $T = g^{a+b}$ or not (that is $c = a + b$).

General Tools: Signature

Definition (Signature Scheme)

$\mathcal{S} = (\mathit{Setup}, \mathit{SKeyGen}, \mathit{Sign}, \mathit{Verif})$:

- $\mathit{Setup}(1^k) \rightarrow$ global parameters param;
- $\mathit{SKeyGen}(\mathit{param}) \rightarrow$ pair of keys (sk, vk);
- $\mathit{Sign}(sk, m; s) \rightarrow$ signature σ, using the random coins s;
- $\mathit{Verif}(vk, m, \sigma) \rightarrow$ validity of σ

Definition (Security: EF-CMA)

An adversary should not be able to generate
a new valid message-signature pair (Existential Forgery)
even when having access to any signature of its choice
(Chosen-Message Attack).

Signature: Waters

$\mathbb{G} = \langle g \rangle = \langle h \rangle$ group of order p, and a bilinear map $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$

Waters Signature

[Waters, 2005]

For a k-bit message $M = (M_i)$, we define $\mathcal{F}(M) = u_0 \prod_{i=1}^{k} u_i^{M_i}$.

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

$$e(g, \sigma_1) \cdot e(\mathcal{F}(M), \sigma_2) = e(Y, h).$$

Security

Waters signature reaches EF-CMA under the *CDH* assumption

General Tools: Encryption

Definition (Encryption Scheme)

$\mathcal{E} = (Setup, EKeyGen, Encrypt, Decrypt)$:

- $Setup(1^k) \quad \to \quad$ global parameters *param*;
- $EKeyGen(param) \quad \to \quad$ pair of keys (pk, dk);
- $Encrypt(pk, m; r) \quad \to \quad$ ciphertext c, using the random coins r;
- $Decrypt(dk, c) \quad \to \quad$ plaintext, or $\perp$ if the ciphertext is invalid.

Definition (Security: IND-CPA)

An adversary should not be able to distinguish the encrytion of m_0 from the encryption of m_1 (Indistinguishability) whereas it can encrypt any message of its choice (Chosen-Plaintext Attack).

Encryption: Linear

$\mathbb{G} = \langle g \rangle$ group of order p

Linear Encryption

[Boneh, Boyen, Shacham, 2004]

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

Security

Linear encryption reaches IND-CPA under the *DLin* assumption

Groth-Sahai Proofs

[Groth, Sahai, 2008]

For any pairing product equation of the form:

$$\prod e(A_i, X_i)^{\alpha_i} \prod e(X_i, X_j)^{\gamma_{i,j}} = t,$$

where the $A_i \in \mathbb{G}$, and $t \in \mathbb{G}_T$ are constant group elements, $\alpha_i \in \mathbb{Z}_p$, and $\gamma_{i,j} \in \mathbb{Z}_p$ are constant scalars, and X_i are unknowns

- either group elements in $\mathbb{G}$,
- or of the form g^{x_i},

one can make a proof of knowledge of values for the X_i's or x_i's so that the equation is satisfied:

- one first commits these secret values using random coins,
- and then provides proofs, that are group elements, using the above random coins,

$\to$ Under the *DLin* assumption: Efficient NIZK

Electronic Cash

Electronic Coins [Chaum, 1981]

Expected properties:

- coins are signed by the bank, for unforgeability
- coins must be distinct to detect/avoid double-spending
- the bank should not know to whom it gave a coin, for anonymity

Electronic Cash

The process is the following one:

- Withdrawal: the user gets a signed coin c from the bank
- Spending: the user spends a coin c in a shop
- Deposit: the shop gives back the money to the bank

The coin is blindly signed by the bank

Blind RSA

[Chaum, 1981]

The easiest way for blind signatures, is to blind the message:
To get an RSA signature on m under public key (n, e),

- The user computes a blind version of the hash value: $M = H(m)$ and $M' = M \cdot r^e \bmod n$
- The signer signs M' into $\sigma' = M'^d \bmod n$
- The user unblinds the signature: $\sigma = \sigma'/r \bmod n$

Indeed,

$$\sigma = \sigma'/r = M'^d/r = (M \cdot r^e)^d/r = M^d \cdot r/r = M^d \bmod n$$

$\rightarrow$ Proven under the One-More RSA

[Bellare, Namprempre, Pointcheval, Semanko, 2001]

Blind Signatures

Randomizable Commutative Signature/Encryption

[Blazy, Fuchsbauer, Pointcheval, Vergnaud, 2011]

- The user "blinds" M into C, under random coins r
- The signer signs C into $\sigma(C)$, under random coins s
- The user "unblinds" the signature $\sigma(M)$, granted the coins r

Weakness

The signer can recognize his signature: the random coins s in $\sigma(M)$

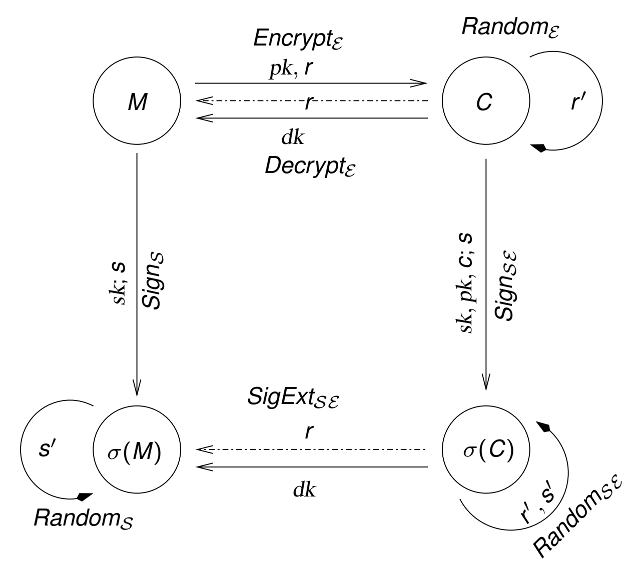
$\rightarrow$ Randomizable Signature

Security

- Encryption hides M (blinding of the message)
- Re-randomization hides $\sigma(M)$ (blinding of the signature)

Randomizable Commutative Signature/Encryption

[Blazy, Fuchsbauer, Pointcheval, Vergnaud, 2011]



Blind Signatures

Such a primitive can be used for a Waters Blind Signature,
by encrypting $\mathcal{F}(M)$:

- Unforgeability: one-more forgery would imply a forgery against the signature scheme (*CDH* assumption)
- Blindness: a distinguisher would break indistinguishability of the encryption scheme (*DLin* assumption)

Efficiency

One obtains a plain Waters Signature

Limitation

A proof of knowledge of M in $C = \mathcal{E}_{pk}(\mathcal{F}(M))$ has to be sent for the security proof: Groth-Sahai NIZK!

Blind Signatures

[Blazy, Fuchsbauer, Pointcheval, Vergnaud, 2011]

In order to get the ℓ-bit message $M = \{M_i\}$ blindly signed:

With Groth-Sahai NIZKP

- 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 (bit-by-bit proof)
- 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

$9\ell + 24$ group elements have to be sent:
$\rightarrow$ It was the most efficient blind signature up to 2011
Why NIZK, since there are already two flows?

Blind Signature

[Blazy, Pointcheval, Vergnaud, 2012]

In order to get the ℓ-bit message $M = \{M_i\}$ blindly signed:

With SPHF

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 $\mathrm{Hash} = \mathrm{Hash}(\mathrm{hk}; (C_1, C_2))$
- U computes $\mathrm{Hash} = \mathrm{ProjHash}(\mathrm{hp}; (C_1, C_2), (r, s))$, 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 only!

Oblivious Transfers

Oblivious Transfer [Rabin, 1981]

A sender S wants to send a message M to U such that

- U gets M with probability 1/2, or nothing
- S does not learn whereas U gets the message M or not

1-2 Oblivious Transfer [Even, Goldreich, Lempel, 1985]

A sender S owns two messages m_0 and m_1, and U owns a bit b

- U gets m_b but nothing on the other message
- S does not learn anything about b

Oblivious Signature-Based Encryption [Li, Du, Boneh, 2003]

A sender S wants to send a message M to U such that

- U gets M if and only if it owns a signature σ on a message m valid under vk
- S does not learn whereas U gets the message M or not

Correctness: if U owns a valid signature, he learns M

Security Notions

- Oblivious: S does not know whether U owns a valid signature (and thus gets the message);
- Semantic Security: U does not learn any information about M if he does not own a valid signature.

RSA-Based OSBE [Li, Du, Boneh, 2003]

The authority generates a FDH-RSA system $(vk = (n, e), sk = d)$, and signs m into σ for U: $\sigma = h^d \bmod n$, where $h = H(m)$.

S wants to send a message M to U, if U owns a valid signature:

- U chooses a random scalar x, and sends $u = (\sigma h^x) \bmod n$;
- S chooses a random scalar y, and computes $r = u^{ey} h^{-y} \bmod n$. It sends $v = h^{ey} \bmod n$, and a encryption of the message M under the symmetric key $k = H'(r)$;
- U computes $r' = v^x \bmod n$, and $k' = H'(r')$.

Correctness:

$r = u^{ey} h^{-y} = \sigma^{ey} h^{xey} h^{-y} = h^{dey} h^{xey} h^{-y} = h^{exy} = v^x = r' \bmod n.$

RSA-Based OSBE: Security

- Oblivious: $u = (\sigma h^x) \bmod n$ is uniformly distributed in $\mathbb{Z}_n^*$ (for an appropriate range of x);
- Semantic Security: upon reception of u, S sends $v = h^{1+ez} \bmod n$ for a random z. Then $v = h^{e(d+z)}$: formally, $v = h^{ye}$ for $y = d + z$. If U is able to compute $r = u^{ey} h^{-y}$ (extracted from H'-calls): $r = u^{1+ez} h^{-d} h^{-z}$, and thus

$$\sigma = h^d = u^{1+ez}/(rh^z) \bmod n.$$

$\rightarrow$ the knowledge of a valid signature is required to decrypt

But security in the Random Oracle Model

One-Round OSBE from IBE [Li, Du, Boneh, 2003]

The authority owns the master key of an IBE scheme, and provides the decryption key (signature) associated to m to U.

S wants to send a message M to U, if U owns a valid signature.

- S encrypts M under the identity m.

Security properties:

- Correct: trivial
- Oblivious: no message sent!
- Semantic Security: IND-CPA of the IBE

But the authority can decrypt everything!

A Stronger Security Model

S wants to send a message M to U, if U owns/uses a valid signature.

Security Notions

- Escrow-free (Oblivious w.r.t. the authority):
  the authority does not know whether U uses a valid signature (and thus gets the message);
- Semantic Security: U cannot distinguish
  multiple interactions with S sending M_0
  from multiple interactions with S sending M_1
  if he does not own/use a valid signature;
- Semantic Security w.r.t. the Authority: after the interaction,
  the authority does not learn any information about M.

A New OSBE

S wants to send a message M to U, if U owns a valid signature σ on m under vk:

With a Smooth Projective Hash Function

The user U and the sender S use a smooth projective hash system for L: $C = \mathcal{E}_{pk}(\sigma; r)$ contains a valid signature σ of m under vk

- the user U sends an encryption C of σ;
- A generates a hk and the associated hp,
  computes $\mathsf{Hash} = \mathsf{Hash}(\mathsf{hk}; C)$,
  and sends hp together with $c = M \oplus \mathsf{Hash}$;
- U computes $\mathsf{Hash} = \mathsf{ProjHash}(\mathsf{hp}; C, r)$, and gets M.

Security Properties

- Oblivious/Escrow-free: IND-CPA of the encryption scheme
  (Hard-partitioned Subset of the SPHF);
- Semantic Security: Smoothness of the SPHF
- Semantic Security w.r.t. the Authority:
  Pseudo-randomness of the SPHF

Semantic Security w.r.t. the Authority requires one interaction
$\rightarrow$ round-optimal
Standard model with Waters Signature + Linear Encryption
$\rightarrow$ CDH and DLin assumptions

Lin-compatible SPHF

- encryption key $pk = (Y_1 = g^{y_1}, Y_2 = g^{y_2})$
- ciphertext $C = (c_1 = Y_1^{r_1}, c_2 = Y_2^{r_2}, c_3 = g^{r_1+r_2} \times M)$

$Lin(pk, M)$: language of the ciphertexts of M
An SPHF for $Lin(pk, M)$ can be:

$$HashKG(Lin(pk, M)) = hk = (x_1, x_2, x_3) \stackrel{\$}{\leftarrow} \mathbb{Z}_p^3$$
$$ProjKG(hk; Lin(pk, M), C) = hp = (Y_1^{x_1} g^{x_3}, Y_2^{x_2} g^{x_3})$$

$$c_1^{x_1} c_2^{x_2} (c_3/M)^{x_3} = hp_1^{r_1} hp_2^{r_2}$$

This basically shows that
$(c_1, c_2, c_3/M)$ is a linear tuple in basis (Y_1, Y_2, g)

SPHF for Linear Encryptions of Waters Signatures

- verification key $vk = Y = g^x$ $(sk = X = h^x)$
- signature $\sigma = (\sigma_1 = X \times \mathcal{F}(M)^s, \sigma_2 = g^s)$
- encryption key $pk = (Y_1 = g^{y_1}, Y_2 = g^{y_2})$
- ciphertext $C = (c_1 = Y_1^{r_1}, c_2 = Y_2^{r_2}, c_3 = g^{r_1+r_2} \times \sigma_1, \sigma_2)$

$WLin(pk, vk, M)$: language of the ciphertexts of signatures of M

$$C_1 = e(c_1, g), C_2 = e(c_2, g), C_3 = e(c_3, g)/(e(h, vk) \cdot e(\mathcal{F}(M), \sigma_2))$$

is a linear tuple in basis $(e(Y_1, g), e(Y_2, g), e(g, g))$ in $\mathbb{G}_T$.
An SPHF for $WLin(pk, vk, M)$ can be:

$$\begin{aligned} HashKG(WLin(pk, vk, M)) &= hk = (x_1, x_2, x_3) \stackrel{\$}{\leftarrow} \mathbb{Z}_p^3 \\ ProjKG(hk; WLin(pk, vk, M), C) &= hp = (Y_1^{x_1} g^{x_3}, Y_2^{x_2} g^{x_3}) \end{aligned}$$

$$e(c_1, g)^{x_1} e(c_2, g)^{x_2} (e(c_3, g)/(e(h, Y) e(\mathcal{F}(M), \sigma_2)))^{x_3} = e(hp_1^{r_1} hp_2^{r_2}, g)$$

Conclusion

Smooth Projective Hash Functions
can be used as implicit proofs of knowledge or membership

Various Applications

- IND-CCA [Cramer, Shoup, 2002]
- PAKE [Gennaro, Lindell, 2003]
- Certification of Public Keys [Abdalla, Chevalier, Pointcheval, 2009]

Privacy-preserving protocols

- Blind signatures
- Oblivious Signature-Based Envelope

$\rightarrow$ Round optimal!

Work in progress: many more applications…