Efficient Smooth Projective Hash Functions and Applications

David Pointcheval

Joint work with Olivier Blazy, Céline Chevalier and Damien Vergnaud

Ecole Normale Supérieure

Isaac Newton Institute for Mathematical Sciences, Cambridge, UK
Semantics and Syntax: A Legacy of Alan Turing
Formal and Computational Cryptographic Proofs – April 12th, 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)

$\rightarrow$ Proof of validity/knowledge

Why should the authority learn the final status?

$\rightarrow$ Implicit proof of validity/knowledge?

Certification of Public Keys: ZKPoK

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

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)

And the authority learns the final status!

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 only if he knows the associated secret key sk:

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 only if he knows the associated secret key sk:

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 $\text{Hash} = \text{Hash}(\text{hk}; (pk, C))$
- U computes $\text{Hash} = \text{ProjHash}(\text{hp}; (pk, C), r))$, and gets Cert

Implicit proof of knowledge of sk
$\rightarrow$ the authority does not learn the final status!

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: $H(x) = \text{Hash}_L(\text{hk}; x)$;
- 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.

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

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 $X \setminus L$

Examples

DH Language [Cramer, Shoup, 2002]

$L_{g,h} = \{(u,v)\}$ where (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

- $\mathsf{HashKG}() = \mathsf{hk} = (\gamma_1, \gamma_3) \stackrel{\$}{\leftarrow} \mathbb{Z}_p^2$
- $\mathsf{ProjKG}(\mathsf{hk}) = \mathsf{hp} = g^{\gamma_1} h^{\gamma_3}$

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

Examples (Con'd)

Commitment/Encryption [Gennaro, Lindell, 2003]

$L_{pk,m} = \{c\}$ where 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]

$L_{pk,(\ell,m)} = \{c\}$ where 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]

$L_{pk,m} = \{c\}$ where c is an equivocable/extractable commitment of m

$\rightarrow$ PAKE in the UC Framework with Adaptive 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*) [Boneh, Boyen, Shacham, 2004])

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 $c = a + b$).
(U, V, T) is (or not) a linear tuple w.r.t. (u, v, g)

General Tools: Signature

Definition (Signature Scheme)

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

- $\mathit{Setup}(1^k) \rightarrow$ global parameters param
- $\mathit{KeyGen}(\mathit{param}) \rightarrow$ 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 [Goldwasser, Micali, Rivest, 1984])

An adversary should not be able to generate a new valid message-signature pair for a new message (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)

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

- $Setup(1^k) \quad \to \quad$ global parameters *param*
- $KeyGen(param) \quad \to \quad$ 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 [Goldwasser, Micali, 1984])

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 $(r_1, r_2) \stackrel{\$}{\leftarrow} \mathbb{Z}_p^2$
  $\to \quad C = (C_1 = X_1^{r_1}, C_2 = X_2^{r_2}, C_3 = g^{r_1+r_2} \cdot M)$
- $Decrypt(dk = (x_1, x_2), C = (C_1, C_2, C_3)) \to M = C_3 / C_1^{1/x_1} C_2^{1/x_2}$

Security

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

Encryption: Linear Cramer-Shoup

$\mathbb{G}$ group of order p, with three independent generators $g_1, g_2, g_3 \in \mathbb{G}$

Linear Cramer-Shoup Encryption [Shacham, 2007]

- Keys: $dk = (x_1, x_2, x_3, y_1, y_2, y_3, z_1, z_2, z_3) \stackrel{\$}{\leftarrow} \mathbb{Z}_p^9$,
  $$pk = \begin{pmatrix} g_1, & c_1 & = & g_1^{x_1} g_3^{x_3}, & c_2 & = & g_2^{x_2} g_3^{x_3} & \\ g_2, & d_1 & = & g_1^{y_1} g_3^{y_3}, & d_2 & = & g_2^{x_2} g_3^{y_3}, & \mathcal{H} \\ g_3, & h_1 & = & g_1^{z_1} g_3^{z_3}, & h_2 & = & g_2^{z_2} g_3^{z_3} & \end{pmatrix}$$
- $Encrypt(pk = (g_1, g_2, g_3, c_1, c_2, d_1, d_2, h_1, h_2, \mathcal{H}), m; (r, s))$, for $M \in \mathbb{G}$:
  $C = (\vec{u} = (u_1 = g_1^r, u_2 = g_2^s, u_3 = g_3^{r+s}), e = M \cdot h_1^r h_2^s, v = v_1^r v_2^s)$
  where $v_1 = c_1 d_1^{\xi}$, $v_2 = c_2 d_2^{\xi}$, and $\xi = \mathcal{H}(\vec{u}, e)$
- $Decrypt(dk = (x_1, x_2, x_3, y_1, y_2, y_3, z_1, z_2, z_3), C = (\vec{u}, e, v))$
  one checks $v \stackrel{?}{=} u_1^{x_1+\xi y_1} u_2^{x_2+\xi y_2} u_3^{x_3+\xi y_3} \quad \to \quad M = e / u_1^{z_1} u_2^{z_3} u_2^{z_3}$

Encryption: CCA Security

Definition (Security: IND-CCA [Rackoff, Simon, 1991]**)**

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

Security: Non-Malleability [Dolev,Dwork, Naor, 1991]

IND-CCA implies Non-Malleability [Bellare, Desai, Pointcheval, Rogaway, 1998]

Security of the Linear Cramer-Shoup [Shacham, 2007]

Linear Cramer-Shoup encryption reaches IND-CCA
under the *DLin* assumption

Groth-Sahai Proofs [Groth, Sahai, 2008]

For any pairing product equation of the form:

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

where the $A, B, A_i \in \mathbb{G}$ 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,

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

Notations [Abdalla, Chevalier, Pointcheval, 2009]

We assume that G possesses a group structure, and we denote by $\oplus$ the commutative law of the group (and by $\ominus$ the opposite operation)
We assume to be given two smooth hash systems SHS_1 and SHS_2, on the sets G_1 and G_2 (included in G) corresponding to the languages L_1 and L_2 respectively:

$$\mathsf{SHS}_i = \{\mathsf{HashKG}_i, \mathsf{ProjKG}_i, \mathsf{Hash}_i, \mathsf{ProjHash}_i\}$$

Let $c \in X$, and r_1 and r_2 two random elements:

$$\begin{array}{lcllcl} \mathsf{hk}_1 & = & \mathsf{HashKG}_1(r_1) & \mathsf{hk}_2 & = & \mathsf{HashKG}_2(r_2) \\ \mathsf{hp}_1 & = & \mathsf{ProjKG}_1(\mathsf{hk}_1, c) & \mathsf{hp}_2 & = & \mathsf{ProjKG}_2(\mathsf{hk}_2, c) \end{array}$$

Conjunction of Languages

A hash system for the language $L = L_1 \cap L_2$ is defined as follows, if $c \in L_1 \cap L_2$ and w_i is a witness that $c \in L_i$, for $i = 1, 2$:

$$\begin{aligned} \mathsf{HashKG}_L(r = r_1 \| r_2) &= \mathsf{hk} = (\mathsf{hk}_1, \mathsf{hk}_2) \\ \mathsf{ProjKG}_L(\mathsf{hk}, c) &= \mathsf{hp} = (\mathsf{hp}_1, \mathsf{hp}_2) \\ \mathsf{Hash}_L(\mathsf{hk}, c) &= \mathsf{Hash}_1(\mathsf{hk}_1, c) \oplus \mathsf{Hash}_2(\mathsf{hk}_2, c) \\ \mathsf{ProjHash}_L(\mathsf{hp}, c; (w_1, w_2)) &= \mathsf{ProjHash}_1(\mathsf{hp}_1, c; w_1) \\ & \quad \oplus \mathsf{ProjHash}_2(\mathsf{hp}_2, c; w_2) \end{aligned}$$

- if c is not in one of the languages, then the corresponding hash value is perfectly random: smoothness
- without one of the witnesses, then the corresponding hash value is computationally unpredictable: pseudo-randomness

Disjunction of Languages

A hash system for the language $L = L_1 \cup L_2$ is defined as follows, if $c \in L_1 \cup L_2$ and w is a witness that $c \in L_i$ for $i \in \{1, 2\}$:

$$
\begin{array}{rcll}
\mathsf{HashKG}_L(r = r_1 \| r_2) & = & \mathsf{hk} = (\mathsf{hk}_1, \mathsf{hk}_2) & \\
\mathsf{ProjKG}_L(\mathsf{hk}, c) & = & \mathsf{hp} = (\mathsf{hp}_1, \mathsf{hp}_2, \mathsf{hp}_\Delta) & \\
\text{where } \mathsf{hp}_\Delta & = & \mathsf{Hash}_1(\mathsf{hk}_1, c) \oplus \mathsf{Hash}_2(\mathsf{hk}_2, c) & \\
\mathsf{Hash}_L(\mathsf{hk}, c) & = & \mathsf{Hash}_1(\mathsf{hk}_1, c) & \\
\mathsf{ProjHash}_L(\mathsf{hp}, c; w) & = & \mathsf{ProjHash}(\mathsf{hp}_1, c; w) & \text{if } c \in L_1 \\
 & \text{or} & \mathsf{hp}_\Delta \ominus \mathsf{ProjHash}_2(\mathsf{hp}_2, c; w) & \text{if } c \in L_2
\end{array}
$$

hp_Δ helps to compute the missing hash value, if and only if at least one can be computed

Pairing Product Equations

$A_i \in \mathbb{G}$ $(i = 1, \ldots, m)$, $\zeta_i \in \mathbb{Z}_p$ $(i = m+1, \ldots, n)$, and $B \in \mathbb{G}_T$ public.
One wants to show its knowledge of $X_i \in \mathbb{G}$ (for $i = 1, \ldots, m$) and $Z_i \in \mathbb{G}_T$ (for $i = m+1, \ldots, n$) that simultaneously satisfy

$$\left(\prod_{i=1}^{m} e(X_i, A_i)\right) \cdot \left(\prod_{i=m+1}^{n} Z_i^{\zeta_i}\right) = B$$

One thus commits X_i (linear encryption) in $\mathbb{G}$, into $\vec{c}_i$, for $i = 1, \ldots, m$,
encrypted under $pk = (g, u_1, u_2)$,
and Z_i (linear encryption) in $\mathbb{G}_T$, into $\vec{C}_i$, for $i = m+1, \ldots, n$,
encrypted under $PK_i = (G, U_1, U_2)$
where $G = e(g, g)$, $U_1 = e(u_1, g)$, $U_2 = e(u_2, g)$.

Commitments

$$
\begin{array}{rcll}
\vec{c}_i & = & (u_1^{r_i}, u_2^{s_i}, g^{r_i+s_i} \cdot X_i) & \text{for } i = 1, \ldots, m \\
\vec{C}_i & = & (U_1^{r_i}, U_2^{s_i}, G^{r_i+s_i} \cdot Z_i) & \text{for } i = m+1, \ldots, n
\end{array}
$$

The $\vec{c}_i$'s can be transposed into $\mathbb{G}_T$, for $i = 1, \ldots, m$:

$$\vec{C}_i = (U_{i,1}^{r_i}, U_{i,2}^{s_i}, G_i^{r_i+s_i} \cdot Z_i)$$

where $U_{i,1} = e(u_1, A_i)$, $U_{i,2} = e(u_2, A_i)$, $G_i = e(g, A_i)$,
but also, $Z_i = e(X_i, A_i)$, for $i = 1, \ldots, m$

We also denote $U_{i,1} = U_1$, $U_{i,2} = U_2$, $G_i = G$, for $i = m+1, \ldots, n$

Smooth Projective Hash Function

$(\lambda, (\eta_i, \theta_i)_{i=1,\ldots,n}) \xleftarrow{\$} \mathbb{Z}_p^{2n+1}$, one sets $\mathsf{hk}_i = (\eta_i, \theta_i, \lambda)$
and $\mathsf{hp}_i = (u_1^{\eta_i} g^{\zeta_i \lambda}, u_2^{\theta_i} g^{\zeta_i \lambda}) \in \mathbb{G}^2$
where $\zeta_i = 1$ for $i = 1, \ldots, m$.
The associated projection keys in $\mathbb{G}_T$ are
$\mathsf{HP}_i = (e(\mathsf{hp}_{i,1}, A_i), e(\mathsf{hp}_{i,2}, A_i))$, for $i = 1, \ldots, n$,
where $A_i = g$ for $i = m+1, \ldots, n$.
The hash value is

$$
\begin{array}{rcl}
H & = & \left(\prod_{i=1}^{n} C_{i,1}^{\eta_i} \cdot C_{i,2}^{\theta_i} \cdot C_{i,3}^{\zeta_i \lambda}\right) \times B^{-\lambda} \\
 & = & \left(\prod_{i=1}^{n} \mathsf{HP}_{i,1}^{r_i} \mathsf{HP}_{i,2}^{s_i}\right) \times \left(\prod_{i=1}^{m} e(X_i, A_i) \prod_{i=m+1}^{n} Z_i^{\zeta_i} / B\right)^{\lambda}
\end{array}
$$

Equality indeed holds if and only if the equation is satisfied

Multiple Equations

We have X_i committed in $\mathbb{G}$, in $\vec{c}_i$, for $i = 1, \ldots, m$
and Z_i committed in $\mathbb{G}_T$, in $\vec{C}_i$, for $i = m+1, \ldots, n$.
We want to show they simultaneously satisfy

$$\left(\prod_{i \in \mathcal{A}_k} e(X_i, A_{k,i})\right) \cdot \left(\prod_{i \in \mathcal{B}_k} Z_i^{\zeta_{k,i}}\right) = B_k, \text{ for } k = 1, \ldots, t$$

where $A_{k,i} \in \mathbb{G}$, $B_k \in \mathbb{G}_T$, and $\zeta_{k,i} \in \mathbb{Z}_p$ are public,
as well as $\mathcal{A}_k \subseteq \{1, \ldots, m\}$ and $\mathcal{B}_k \subseteq \{m+1, \ldots, n\}$

This is a conjunction of languages

$\rightarrow$ Similar Hash Proofs on Linear Cramer-Shoup Commitments

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

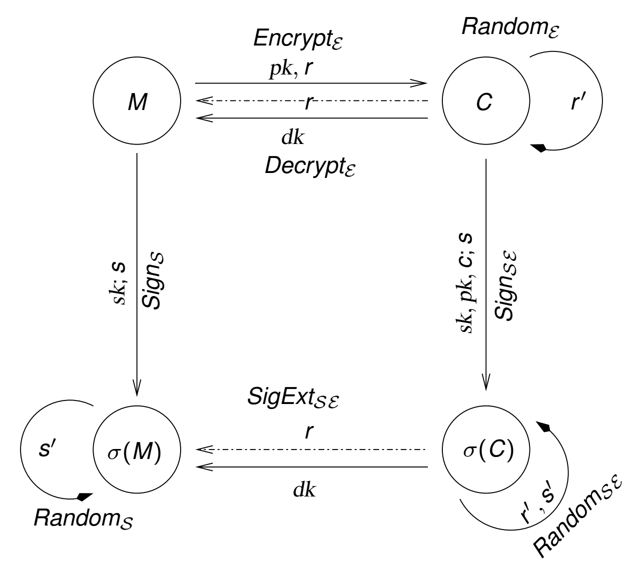
The signer can recognize his signature: the random coins s in $\sigma(M)$
$\rightarrow$ Randomizable Signature

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

Blind Signature

[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 Proof 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 $\mathsf{Hash} = \mathsf{Hash}(\mathsf{hk}; (C_1, C_2))$
- 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 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 Envelope [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

A Stronger Security Model

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

Security Notions

- 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 [Blazy, Pointcheval, Vergnaud, 2012]

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;
- S 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 (even w.r.t. the Authority): 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

General Construction

- The user U sends a commitment C of a word w
- S generates a hk and the associated hp,
  computes $\text{Hash} = \text{Hash}(\text{hk}; C)$,
  and sends hp together with $c = M \oplus \text{Hash}$;
- U computes $\text{Hash} = \text{ProjHash}(\text{hp}; C, r)$, and gets M.

U gets M iff w is in the appropriate language:

- a signature on a public message: OSBE
- a signature on a private message: Anonymous Credential
- a private message (low entropy): Password

Password-based Authenticated Key Exchange

GL – Generic Approach [Gennaro, Lindell, 2003]

Additional tricks are required for the security!

Alice		Bob
$C_1 = Commit(pw; r_1)$	$\xrightarrow{C_1}$	$C_2 = Commit(pw; r_2)$
	$\xleftarrow{C_2, \text{hp}_1}$	hk_1, hp_1 on C_1
hk_2, hp_2 on C_2	$\xleftarrow{\text{hp}_2}$	

$$\text{ProjHash}(\text{hp}_1; C_1, r_1) = H_1 = \text{Hash}(\text{hk}_1; C_1)$$
$$\text{Hash}(\text{hk}_2; C_2) = H_2 = \text{ProjHash}(\text{hp}_2; C_2, r_2)$$

$$K = H_1 \cdot H_2$$

The language is: valid commitments of pw

Language-based Authenticated Key Exchange

Definition

- Alice owns a word w_1 is a language $L_1(Pub_1, Priv_1)$;
- Bob owns a word w_2 is a language $L_2(Pub_2, Priv_2)$;
- If Alice and Bob agree on the languages,
  and actually own valid words (implicit authentication),
  they will agree on a common session key (semantic security)

- $Pub = \emptyset$, $Priv = pw$ and $L(Pub, Priv) = \{Priv\}$: PAKE
- $Pub = M$, $Priv = vk$, $L(Pub, Priv) = \{\sigma, Verif(Priv, Pub, \sigma) = 1\}$:
  Secret Handshake
- $Pub = \emptyset$, $Priv = (vk, M)$, $L(Pub, Priv) = \{\sigma, Verif(Priv, \sigma) = 1\}$:
  CAKE – Credential-based AKE [Camenisch, Casati, Gross, Shoup, 2010]

Language-based Authenticated Key Exchange

Our Construction

- With a Linear Cramer-Shoup UC commitment [Lindell, 2011]
- Using the GL approach [Gennaro, Lindell, 2003]

$\rightarrow$ UC Secure LAKE

Languages

- Password: PAKE secure under $DLin$
- Waters Signature: Secret Handshake, Credentials
  secure under $DLin + CDH$

Any Linear Pairing Product Equation Systems in both $\mathbb{G}$ and $\mathbb{G}_T$

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 [Blazy, Pointcheval, Vergnaud, 2012]
- Oblivious Signature-Based Envelope

$\rightarrow$ Round optimal!

More general: Language-based Authenticated Key Exchange