Efficient Smooth Projective Hash Functions and Applications

David Pointcheval

Joint work with Olivier Blazy, Céline Chevalier and Damien Vergnaud

Ecole Normale Supérieure

Motivation

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

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

Implicit proof of knowledge of \( sk \)
\[ \rightarrow \quad \text{the authority does not learn the final status!} \]
Examples

**DH Language** [Cramer, Shoup, 2002]

\[ L_{gh} = \{(u, v)\} \text{ where } (g, h, u, v) \text{ is DH tuple:} \\
\text{there exists } r \text{ such that } u = g^r \text{ and } v = h^r \]

→ Public-key Encryption with IND-CCA Security

**General Tools: Signature**

**Definition (Signature Scheme)**

\[ S = (\text{Setup}, \text{KeyGen}, \text{Sign}, \text{Verif}): \]

\[ \text{Setup}(1^k) \rightarrow \text{global parameters } \text{param} \]

\[ \text{KeyGen}(\text{param}) \rightarrow \text{pair of keys } (sk, vk) \]

\[ \text{Sign}(sk, m; s) \rightarrow \text{signature } \sigma, \text{using the random coins } s \]

\[ \text{Verif}(vk, m, \sigma) \rightarrow \text{validity of } \sigma \]

**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)

\[ (U, V, T) \text{ is (or not) a linear tuple w.r.t. } (u, v, g) \]

An even when having access to any signature of its choice (Chosen-Message Attack).

---

**Applications**

**Commitment/Encryption** [Gennaro, Lindell, 2003]

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

→ Password-Authenticated Key Exchange in the Standard Model

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

\[ L_{pk(\ell), m} = \{c\} \text{ where } c \text{ is an encryption of } m \text{ under } pk, \text{ with label } \ell \]

→ PAKE in the UC Framework (passive corruptions)

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

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

→ PAKE in the UC Framework with Adaptive Corruptions

---

**Assumptions: CDH and DLin**

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

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

For any generator \( g \overset{\$}{\leftarrow} G \), and any scalars \( a, b \overset{\$}{\leftarrow} \mathbb{Z}_p^* \),

\[ \text{given } (g, g^a, g^b), \text{ compute } g^{ab}. \]

Decisional variant easy if a bilinear map is available.

**Definition (Decision Linear Problem (DLin))** [Boneh, Boyen, Shacham, 2004]

For any generator \( g \overset{\$}{\leftarrow} G \), and any scalars \( a, b, x, y, c \overset{\$}{\leftarrow} \mathbb{Z}_p^* \),

\[ \text{given } (g, g^x, g^y, g^{xa}, g^{yb}, g^c), \text{ decide whether } c = a + b \text{ or not.} \]

Equivalently, given a reference triple \( (u = g^x, v = g^y, g) \)

\[ \text{and a new triple } (U = u^a = g^{xa}, V = v^b = g^{yb}, T = g^c), \]

\[ \text{decide whether } T = g^{a+b} \text{ or not (that is } c = a + b). \]

\[ (U, V, T) \text{ is (or not) a linear tuple w.r.t. } (u, v, g) \]
**Signature: Waters**

\[ G = \langle g \rangle \] group of order \( p \), and a bilinear map \( e : G \times G \to G_T \)

**Waters Signature**

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 \( x \overset{\$}{\leftarrow} \mathbb{Z}_p \)
- \( \text{Sign}(sk = X, M; s) \), for \( M \in \{0, 1\}^k \) and \( s \overset{\$}{\leftarrow} \mathbb{Z}_p \)
  \[ \sigma = (\sigma_1 = X \cdot \mathcal{F}(M)^s, \sigma_2 = g^{-s}) \]
- \( \text{Verify}(vk = X, M, \sigma = (\sigma_1, \sigma_2)) \) 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)**

\[ \mathcal{E} = (\text{Setup}, \text{KeyGen}, \text{Encrypt}, \text{Decrypt}) : \]

- \( \text{Setup}(1^k) \to \) global parameters \( \text{param} \)
- \( \text{KeyGen}(\text{param}) \to \) pair of keys \( (pk, dk) \)
- \( \text{Encrypt}(pk, m; r) \to \) ciphertext \( c \), using the random coins \( r \)
- \( \text{Decrypt}(dk, c) \to \) plaintext, or \( \bot \) if the ciphertext is invalid

**Definition (Security: IND-CPA)**

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 (Chosen-Plaintext Attack).

---

**Encryption: Linear**

\[ G = \langle g \rangle \] group of order \( p \)

**Linear Encryption**

- Keys: \( dk = (x_1, x_2) \overset{\$}{\leftarrow} \mathbb{Z}_p^2 \), \( pk = (X_1 = g^{x_1}, X_2 = g^{x_2}) \)
- \( \text{Encrypt}(pk = (X_1, X_2), M; (r_1, r_2)) \), for \( M \in G \) and \( (r_1, r_2) \overset{\$}{\leftarrow} \mathbb{Z}_p^2 \)
  \[ C = (C_1 = X_1^{r_1}, C_2 = X_2^{r_2}, C_3 = g^{r_1 + r_2} \cdot M) \]
- \( \text{Decrypt}(dk = (x_1, x_2), C = (C_1, C_2, C_3)) \to M = C_3 / C_1^{x_1} C_2^{x_2} \)

**Security**

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

For any pairing product equation of the form:

$$\prod e(A_i, X_i)^{\alpha_i} \prod e(X_i, Y_j)^{\gamma_{ij}} = e(A, B),$$

where the $A, B, A_i \in G$ are constant group elements, $\alpha_i \in \mathbb{Z}_p$, and $\gamma_{ij} \in \mathbb{Z}_p$ are constant scalars, and $X_i$ are unknowns

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

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

**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 $i = 1, 2$:

$$HashKG_L(r = r_1 || r_2) = hk = (hk_1, hk_2)$$
$$ProjKG_L(hk, c) = hp = (hp_1, hp_2)$$
$$Hash_L(hk, c) = Hash_1(hk_1, c) \oplus Hash_2(hk_2, c)$$
$$ProjHash_L(hp, c; (w_1, w_2)) = ProjHash_1(hp_1, c; w_1) \oplus ProjHash_2(hp_2, c; w_2)$$

- 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

**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:

$$SHS_i = \{HashKG_i, ProjKG_i, Hash_i, ProjHash_i\}$$

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

$$hk_1 = HashKG_1(r_1) \quad hk_2 = HashKG_2(r_2)$$
$$hp_1 = ProjKG_1(hk_1, c) \quad hp_2 = ProjKG_2(hk_2, c)$$
Conjunctions and Disjunctions

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 $i \in \{1, 2\}$:

$$\text{HashKG}_L(r = r_1 || r_2) = h_k = (hk_1, hk_2)$$

$$\text{ProjKG}_L(h_k, c) = h_p = (hp_1, hp_2, hp_\Delta)$$

where $hp_\Delta = \text{Hash}_1(hk_1, c) \oplus \text{Hash}_2(hk_2, c)$

$$\text{Hash}_L(h_k, c) = \text{Hash}_1(hk_1, c)$$

$$\text{ProjHash}_L(hp, c; w) = \text{ProjHash}(hp_1, c; w)$$ if $c \in L_1$

$$\text{ProjHash}_L(hp, c; w) = \text{ProjHash}_L(hp_2, c; w)$$ if $c \in L_2$

$hp_\Delta$ helps to compute the missing hash value, if and only if at least one can be computed

$$\bar{c}_i = (u_i^1, u_i^2, g_i^\delta \cdot X_i) \quad \text{for } i = 1, \ldots, m$$

$$\bar{C}_i = (U_i^1, U_i^2, G_i^\delta \cdot Z_i) \quad \text{for } i = m + 1, \ldots, n$$

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

$$\bar{c}_i = (U_i^1, U_i^2, G_i^\delta \cdot Z_i)$$

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 $i = m + 1, \ldots, n$

ECO Normale Superieure

Pairing Product Equations

Commitments

$$\bar{c}_i = (u_i^1, u_i^2, g_i^\delta \cdot X_i) \quad \text{for } i = 1, \ldots, m$$

$$\bar{C}_i = (U_i^1, U_i^2, G_i^\delta \cdot Z_i) \quad \text{for } i = m + 1, \ldots, n$$

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

$$\bar{c}_i = (U_i^1, U_i^2, G_i^\delta \cdot Z_i)$$

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

$$(\lambda, (\eta_i, \theta_i)_{i=1,\ldots,n}) \xleftarrow{\$} \mathbb{Z}_p^{2n+1}, \text{ one sets } h_{k_i} = (\eta_i, \theta_i, \lambda)$$

and $hp_i = (u_i^1 g_i^\lambda, u_i^2 g_i^\lambda) \in \mathbb{G}_2$

where $\zeta_i = 1$ for $i = 1, \ldots, m$.

The associated projection keys in $\mathbb{G}_T$ are

$$\text{HP}_i = e(hp_{i,1}, A_i), e(hp_{i,2}, A_i)), \text{ for } i = 1, \ldots, n,$$

where $A_i = g$ for $i = m + 1, \ldots, n$.

The hash value is

$$H = \left( \prod_{i=1}^{n} C_{i,1}^{\eta_i} \cdot C_{i,2}^{\theta_i} \cdot C_{i,3}^{\zeta_i} \right) \times B^{-\lambda}$$

$$= \left( \prod_{i=1}^{n} \text{HP}_{i,1} \text{HP}_{i,2} \right) \times \left( \prod_{i=1}^{m} e(X_i, A_i) \prod_{i=m+1}^{n} Z_i^{\zeta_i} / B \right)^{\lambda}$$

Equality indeed holds if and only if the equation is satisfied

Ecole Normale Superieure

Pairing Product Equations

Commitments

$$\bar{c}_i = (u_i^1, u_i^2, g_i^\delta \cdot X_i) \quad \text{for } i = 1, \ldots, m$$

$$\bar{C}_i = (U_i^1, U_i^2, G_i^\delta \cdot Z_i) \quad \text{for } i = m + 1, \ldots, n$$

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

$$\bar{c}_i = (U_i^1, U_i^2, G_i^\delta \cdot Z_i)$$

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

$$(\lambda, (\eta_i, \theta_i)_{i=1,\ldots,n}) \xleftarrow{\$} \mathbb{Z}_p^{2n+1}, \text{ one sets } h_{k_i} = (\eta_i, \theta_i, \lambda)$$

and $hp_i = (u_i^1 g_i^\lambda, u_i^2 g_i^\lambda) \in \mathbb{G}_2$

where $\zeta_i = 1$ for $i = 1, \ldots, m$.

The associated projection keys in $\mathbb{G}_T$ are

$$\text{HP}_i = e(hp_{i,1}, A_i), e(hp_{i,2}, A_i)), \text{ for } i = 1, \ldots, n,$$

where $A_i = g$ for $i = m + 1, \ldots, n$.

The hash value is

$$H = \left( \prod_{i=1}^{n} C_{i,1}^{\eta_i} \cdot C_{i,2}^{\theta_i} \cdot C_{i,3}^{\zeta_i} \right) \times B^{-\lambda}$$

$$= \left( \prod_{i=1}^{n} \text{HP}_{i,1} \text{HP}_{i,2} \right) \times \left( \prod_{i=1}^{m} e(X_i, A_i) \prod_{i=m+1}^{n} Z_i^{\zeta_i} / B \right)^{\lambda}$$

Equality indeed holds if and only if the equation is satisfied

Ecole Normale Superieure

Pairing Product Equations
### Multiple Equations

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

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

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

This is a conjunction of languages

→ Similar Hash Proofs on Linear Cramer-Shoup Commitments

---

### Blind RSA

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 $M' = M \cdot r^e \mod n$
- The signer signs $M'$ into $\sigma' = M'^d \mod n$
- The user unblinds the signature: $\sigma = \sigma'/r \mod n$

Indeed,

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

→ Proven under the One-More RSA

[Blazy, Namprempre, Pointcheval, Semanko, 2001]

---

### Randomizable Commutative Signature/Encryption

**Randomizable Commutative Signature/Encryption**

- 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 $\sigma(M)$

→ Randomizable Signature

**Security**

- Encryption hides $M$ (blinding of the message)
- Re-randomization hides $\sigma(M)$ (blinding of the signature)
**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 $C = E_{pk}(\mathcal{F}(M))$ has to be sent

---

**Blind Signature**

In order to get the $\ell$-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:

→ It was the most efficient blind signature up to 2011

Why NIZK, since there are already two flows?

---

**Oblivious Transfers**

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

**With SPHF**

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

- $U$ sends encryptions of $M$, into $C_1$, and $\mathcal{F}(M)$, into $C_2$;
- $A$ generates
  - a signature $\sigma$ on $C_2$,
  - masks it using $\text{Hash} = \text{Hash}(hk; (C_1, C_2))$
- $U$ computes $\text{Hash} = \text{ProjHash}(hp; (C_1, C_2), (r, s))$, and gets $\sigma$. 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!
A sender $S$ wants to send a message $M$ to $U$ such that

- $U$ gets $M$ if and only if it owns a signature $\sigma$ 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/use a valid signature

---

**A New OSBE**

[Blazy, Pointcheval, Vergnaud, 2012]

$S$ wants to send a message $M$ to $U$, if $U$ owns/uses a valid signature $\sigma$ on $m$ under $vk$:

**With a Smooth Projective Hash Function**

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

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

→ round-optimal

Standard model with Waters Signature + Linear Encryption

→ $\text{CDH}$ and $\text{DLin}$ assumptions
### General Construction

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

**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).

**Our Construction**

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

**Languages**

- Password: PAKE secure under $DLin$
- Waters Signature: Secret Handshake, Credentials secure under $DLin + CDH$
- Any Linear Pairing Product Equation Systems in both $G$ and $GT$
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
  → Round optimal!

More general: Language-based Authenticated Key Exchange