**Introduction**

Cryptographic Tools

More Languages

Blind Signatures

OSBE

LAKE

---

**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)

→ Proof of validity/knowledge

Why should the authority learn the final status?

→ 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

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

Implicit proof of knowledge of \( sk \)
- \( \rightarrow \) the authority does not learn the final status!

Smooth Projective Hash Functions

**Definition**

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

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

**Properties**

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

**Smoothness**

For any \( x \not\in L \), \( H(x) \) 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 (Con’d)

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

\[ L_{pk,m} = \{ c \} \text{ where } c \text{ is an encryption of } m \text{ under } pk: \]

\[ \text{there exists } r \text{ such that } c = E_{pk}(m; r) \]

→ 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**

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

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

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

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

Decision variant easy if a bilinear map is available.

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

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

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. \((u, v, g)\)
**Signature: Waters**

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

**Waters Signature** [Waters, 2005]

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

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

**Security**

Waters signature reaches EF-CMA under the \( CDH \) assumption

---

**General Tools: Encryption**

**Definition (Encryption Scheme)**

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

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

**Definition (Security: IND-CPA)** [Goldwasser, Micali, 1984]

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** [Boneh, Boyen, Shacham, 2004]

- **Keys**: \( dk = (x_1, x_2) \leftarrow \mathbb{Z}_p^2 \), \( pk = (X_1 = g^{x_1}, X_2 = g^{x_2}) \)
- **Encrypt**(pk = (X_1, X_2), M; (r_1, r_2)), for \( M \in G \) and \( (r_1, r_2) \leftarrow \mathbb{Z}_p^2 \)
  \[ C = (C_1 = X_1^{r_1} \cdot X_2^{r_2}, C_2 = X_2^{r_1} \cdot X_1^{r_2} \cdot M) \]
- **Decrypt**(dk = (x_1, x_2), C = (C_1, C_2, C_3)) \( \rightarrow \) \( M = C_3 / C_1^{1/x_1} \cdot C_2^{1/x_2} \)

**Security**

Linear encryption reaches IND-CPA under the \( DLin \) assumption

---

**Encryption: Linear Cramer-Shoup**

\[ G \] group of order \( p \), with three independent generators \( g_1, g_2, g_3 \in G \)

**Linear Cramer-Shoup Encryption** [Shacham, 2007]

- **Keys**: \( dk = (x_1, x_2, x_3, y_1, y_2, y_3, z_1, z_2, z_3) \leftarrow \mathbb{Z}_p^9 \)
  \[ pk = \left( \begin{array}{c} g_1, \ c_1 = g_1^{x_1} \cdot g_3^{z_1}, \ c_2 = g_2^{x_2} \cdot g_3^{z_2} \\ g_2, \ d_1 = g_1^{y_1} \cdot g_3^{y_3}, \ d_2 = g_2^{y_2} \cdot g_3^{y_3} \\ g_3, \ h_1 = g_1^{x_3}, \ h_2 = g_2^{x_3} \end{array} \right) \]
- **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 \( M \in G \):
  \[ C = (\bar{u} = (u_1 = g_1^{x_1} \cdot u_2 = g_2^{x_2} \cdot u_3 = g_3^{x_3 + s}), \ e = M \cdot h_1^{r_1} \cdot h_2^{s}, \ v = v_1 \cdot v_2) \]
  where \( v_1 = c_1^{d_1} \cdot v_2 = c_2^{d_2} \), and \( \xi = \mathcal{H}(\bar{u}, e) \)
- **Decrypt**(dk = (x_1, x_2, x_3, y_1, y_2, y_3, z_1, z_2, z_3), C = (\( \bar{u}, e, v \))
  one checks \( v = u_1^{x_1 + \xi y_1} \cdot u_2^{x_2 + \xi y_2} \cdot u_3^{x_3 + \xi y_3} \)
  \( \rightarrow \) \( M = e / u_1^{x_1} \cdot u_2^{x_2} \cdot u_3^{x_3} \)

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 Proofs**

Groth, Sahai, 2008

For any pairing product equation of the form:

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

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 $g^{\alpha_i}$,

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**

Abdalla, Chevalier, Pointcheval, 2009

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**

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)$$\\hk_2 = HashKG_2(r_2)$$\\hp_1 = ProjKG_1(hk_1, c)$$\\hp_2 = ProjKG_2(hk_2, c)$
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 $i \in \{1, 2\}$:

- $\text{Hash}_{KG_1}(r = r_1 || r_2) = h_k = (h_k_1, h_k_2)$
- $\text{Proj}_{KG_1}(h_k, c) = h_p = (h_{p1}, h_{p2}, h_{p\Delta})$
  where $h_{p\Delta} = \text{Hash}_1(h_k_1, c) \oplus \text{Hash}_2(h_k_2, c)$
- $\text{Hash}_L(h_k, c) = \text{Hash}_1(h_k_1, c)$

$\text{ProjHash}_L(h_p, c; w) = \text{ProjHash}(h_{p1}, c; w)$ if $c \in L_1$

or $h_{p\Delta} \oplus \text{ProjHash}_2(h_{p2}, c; w)$ if $c \in L_2$

$h_{p\Delta}$ helps to compute the missing hash value, if and only if at least one can be computed.

Pairing Product Equations

Conjunctions and Disjunctions

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

- $\text{Hash}_{KG_1}(r = r_1 || r_2) = h_k = (h_k_1, h_k_2)$
- $\text{Proj}_{KG_1}(h_k, c) = h_p = (h_{p1}, h_{p2}, h_{p\Delta})$
  where $h_{p\Delta} = \text{Hash}_1(h_k_1, c) \oplus \text{Hash}_2(h_k_2, c)$
- $\text{Hash}_L(h_k, c) = \text{Hash}_1(h_k_1, c)$

$\text{ProjHash}_L(h_p, c; w) = \text{ProjHash}(h_{p1}, c; w)$ if $c \in L_1$

or $h_{p\Delta} \oplus \text{ProjHash}_2(h_{p2}, c; w)$ if $c \in L_2$

$h_{p\Delta}$ helps to compute the missing hash value, if and only if at least one can be computed.

Ecole Normale Supérieure

David Pointcheval

21/41

Introduction Cryptographic Tools
More Languages Blind Signatures OSBE LAKE

Introduction Cryptographic Tools
More Languages Blind Signatures OSBE LAKE

Pairing Product Equations

Commitments

$\tilde{c}_i = \left( u_i^1, u_i^2, g_i^{r_i+s_i} \cdot X_i \right)$ for $i = 1, \ldots, m$

$\tilde{C}_i = \left( U_i^1, U_i^2, G_i^{r_i+s_i} \cdot Z_i \right)$ for $i = m + 1, \ldots, n$

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

$\tilde{c}_i = \left( u_i^1, u_i^2, g_i^{r_i+s_i} \cdot Z_i \right)$

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

Ecole Normale Supérieure

David Pointcheval

22/41

Introduction Cryptographic Tools
More Languages Blind Signatures OSBE LAKE

Introduction Cryptographic Tools
More Languages Blind Signatures OSBE LAKE

Pairing Product Equations

Smooth Projective Hash Function

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

and $h_{p_i} = (u_i^1 g_i^{\xi_i}, u_i^2 g_i^{\lambda}) \in \mathbb{G}^2$

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

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

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

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

The hash value is

$H = \left( \prod_{i=1}^{n} C_{i,1}^{\eta_i}, C_{i}^{\theta_i} \right)$

Ecole Normale Supérieure

David Pointcheval

23/41
### Blind Signatures

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

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

where $A_{k,i} \in G_i$, $B_k \in G_T$, 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

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

### Blind RSA

The easiest way for blind signatures, is to blind the message:

- 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,

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

$\rightarrow$ Proven under the One-More RSA

[Bellare, Namprempre, Pointcheval, Semanko, 2001]
Blind Signatures

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

- 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 $C = E_{pk}(F(M))$ has to be sent

Oblivious Transfers

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 $F(M)$ 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

Such a protocol requires $8\ell + 24$ group elements in total only!

With SPHF

The user $U$ and the authority $A$ use a smooth projective hash system for $L$: $C_1 = E_{\rho_1}(M; r)$ and $C_2 = E_{\rho_2}(F(M); s)$ contain the same $M$

- $U$ sends encryptions of $M$, into $C_1$, and $F(M)$, into $C_2$;
- $A$ generates
  - a signature $\sigma$ on $C_2$,
  - masks it using Hash = 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!

Oblivious Transfer

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

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

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 a valid signature

A New OSBE

$S$ 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 $M$.

With a Smooth Projective Hash Function

The user $U$ and the sender $S$ use a smooth projective hash system for $L$: $C = E_{pk}(\sigma; r)$ 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 $M$. 
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)

- $Pub = \emptyset$, $Priv = pw$ and $L(Pub, Priv) = \{Priv\}$: PAKE
- $Pub = M$, $Priv = vk$, $L(Pub, Priv) = \{\sigma, \text{Verif}(Priv, Pub, \sigma) = 1\}$: Secret Handshake
- $Pub = \emptyset$, $Priv = (vk, M)$, $L(Pub, Priv) = \{\sigma, \text{Verif}(Priv, \sigma) = 1\}$: CAKE – Credential-based AKE

**GL – Generic Approach**

Additional tricks are required for the security!

- $C_1 = \text{Commit}(pw; r_1)$
- $C_2 = \text{Commit}(pw; r_2)$
- $hk_1, hp_1$ on $C_1$
- $hp_2$ on $C_2$
- $\text{ProjHash}(hp_1; C_1, r_1) = H_1 = \text{Hash}(hk_1; C_1)$
- $\text{Hash}(hk_2; C_2) = H_2 = \text{ProjHash}(hp_2; C_2, r_2)$
- $K = H_1 \cdot H_2$

The language is: valid commitments of $pw$

**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 $G_T$

Our Construction

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

$\rightarrow$ UC Secure LAKE
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