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
1. Introduction
2. Cryptographic Tools
3. More Languages
4. Blind Signatures
5. Oblivious Signature-Based Envelope
6. Language-based Authenticated Key Exchange

Motivation
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)

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

Implicit proof of knowledge of \( sk \)
- 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 \( hk \): \( H(x) = Hash_L(hk;x) \);
- or a public projected key \( hp \): \( H(x) = ProjHash_L(hp;x,w) \)

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 \( hk \mapsto hp = ProjKG_L(hk,x) \)

**Properties**

For any \( x \in X \), \( H(x) = Hash_L(hk;x) \)
For any \( x \in L \), \( H(x) = ProjHash_L(hp;x,w) \) \( w \) 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 \( X \setminus L \).
### Blind Signatures

**Introduction** Cryptographic Tools More Languages Blind Signatures OSBE LAKE

**Applications**

**Examples**

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

  \[ \rightarrow \text{ Public-key Encryption with IND-CCA Security} \]

- **Algorithms**
  
  - HashKG() = \( hk = (y_1, y_3) \sim \mathbb{Z}_p^2 \)
  
  - ProjKG(hk) = hp = \( g^{y_1}h^{y_3} \)
  
  \[ \text{Hash(hk,(u,v)) = } u^{y_1}h^{y_3} = hp^r = \text{ProjHash(hp,(u,v);r)} \]

### Computational Assumptions

**Assumptions:** 

- **CDH and DLin**
  
  \[ g \text{ a cyclic group of prime order } p \text{ with or without bilinear map}. \]

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

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

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

- \( \text{Setup}(1^k) \rightarrow \) global parameters \( \text{param} \)
- \( \text{KeyGen}(\text{param}) \rightarrow \) pair of keys \( (sk, vk) \)
- \( \text{Sign}(sk, m; s) \rightarrow \) signature \( \sigma \), using the random coins \( s \)
- \( \text{Verif}(vk, m, \sigma) \rightarrow \) 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) even when having access to any signature of its choice (Chosen-Message Attack).
**Signature: Waters**

\[ \mathcal{G} = \langle g \rangle \text{ group of order } p, \text{ and a bilinear } e : \mathcal{G} \times \mathcal{G} \to \mathcal{G}_T \]

### Waters Signature

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

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

---

**Encryption: Linear**

\[ \mathcal{G} = \langle g \rangle \text{ group of order } p \]

### Linear Encryption

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

---

**Signature & Encryption**

**General Tools: Encryption**

<table>
<thead>
<tr>
<th>Definition (Encryption Scheme)</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \mathcal{E} = (\text{Setup}, \text{KeyGen}, \text{Encrypt}, \text{Decrypt}) : )</td>
</tr>
<tr>
<td>( \text{Setup}(1^k) \to \text{global parameters } param )</td>
</tr>
<tr>
<td>( \text{KeyGen}(param) \to \text{pair of keys } (pk, dk) )</td>
</tr>
</tbody>
</table>

---

**Signature & Encryption**

**Encryption: Linear Cramer-Shoup**

\[ \mathcal{G} \text{ group of order } p, \text{ with three independent generators } g_1, g_2, g_3 \in \mathcal{G} \]

### Linear Cramer-Shoup Encryption

- **Keys**: \( dk = (x_1, x_2, x_3, y_1, y_2, y_3, z_1, z_2, z_3) \overset{\$}{\leftarrow} \mathbb{Z}_p^9 \)
  \[ g_1, c_1 = g_1^{x_1} g_3^{y_1}, c_2 = g_2^{x_2} g_3^{y_2}, g_3, h_1 = g_1^{x_1} g_3^{y_3}, h_2 = g_2^{x_2} g_3^{y_3}, h_3 = \mathcal{H}(\tau, e) \]
- **Encrypt**: \( pk = (g_1, g_2, g_3, c_1, c_2, d_1, d_2, h_1, h_2, h_3, m; (r, s)), \) for \( M \in \mathcal{G} \):
  \[ C = \sigma = (u_1 = g_1^r, u_2 = g_2^s, u_3 = g_3^{r+s}), e = M \cdot h_1^r h_2^s, v = v_1^r v_2^s \]
  \[ \text{where } v_1 = c_1 d_1^e, v_2 = c_2 d_2^e \] and \( \xi = \mathcal{H}(\tau, e) \)
- **Decrypt**: \( dk = (x_1, x_2, x_3, y_1, y_2, y_3, z_1, z_2, z_3), C = (\sigma, e, v) \)
  \[ \text{one checks } v = v_1^{x_1+\xi y_1} v_2^{x_2+\xi y_2} v_3^{x_3+\xi y_3} \to M = e / u_1^2 u_2^2 u_3^2 \]
**Encryption: CCA Security**

**Definition (Security: IND-CCA)** \[\text{Rackoff, Simon, 1991}\]

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** \[\text{Dolev,Dwork, Naor, 1991}\]

IND-CCA implies Non-Malleability \[\text{Bellare, Desai, Pointcheval, Rogaway, 1998}\]

**Security of the Linear Cramer-Shoup** \[\text{Shacham, 2007}\]

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

---

**Groth-Sahai Methodology**

**Groth-Sahai Proofs** \[\text{Groth, Sahai, 2008}\]

For any pairing product equation of the form:

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

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 $g^{X_i}$,

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,

→ Under the $DLin$ assumption: Efficient NIZK

---

**Conjunctions and Disjunctions**

**Notations** \[\text{Abdalla, Chevalier, Pointcheval, 2009}\]

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 $i = 1, 2$:

- $\text{HashKG}_L(r = r_1 \parallel r_2) = \text{hk} = (\text{hk}_1, \text{hk}_2)$
- $\text{ProjKG}_L(\text{hk}, c) = \text{hp} = (\text{hp}_1, \text{hp}_2)$
- $\text{Hash}_L(\text{hk}, c) = \text{Hash}_1(\text{hk}_1, c) \oplus \text{Hash}_2(\text{hk}_2, c)$
- $\text{ProjHash}_L(\text{hp}, c; (w_1, w_2)) = \text{ProjHash}_1(\text{hp}_1, c; w_1) \oplus \text{ProjHash}_2(\text{hp}_2, c; w_2)$

- 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
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 \( i \in \{1, 2\} \):

\[
\begin{align*}
\text{HashKG}_1(r_1 || r_2) &= \text{hk} = (\text{hk}_1, \text{hk}_2) \\
\text{ProjKG}_1(\text{hk}, c) &= \text{hp} = (\text{hp}_1, \text{hp}_2, \text{hp}_A) \\
&\quad \text{where } \text{hp}_A = \text{Hash}_1(\text{hk}_1, c) \oplus \text{Hash}_2(\text{hk}_2, c) \\
\text{Hash}_k(\text{hk}, c) &= \text{Hash}_1(\text{hk}_1, c) \\
\text{ProjHash}_k(\text{hp}, c; w) &= \text{ProjHash}(\text{hp}_1, c; w) \quad \text{if } c \in L_1 \\
&\quad \text{or } \text{hp}_A \oplus \text{ProjHash}_2(\text{hp}_2, c; w) \quad \text{if } c \in L_2
\end{align*}
\]

\( \text{hp}_A \) helps to compute the missing hash value, if and only if at least one can be computed

\[ \text{Commitments} \]

\[
\begin{align*}
\varphi_i &= (u_{i,1}^s, u_{i,2}^s, \cdot, X_i) &\text{for } i = 1, \ldots, m \\
C_i &= (U_{i,1}^s, U_{i,2}^s, \cdot, Z_i) &\text{for } i = m + 1, \ldots, n
\end{align*}
\]

The \( \varphi_i \)'s can be transposed into \( G_T \), for \( i = 1, \ldots, m \):

\[
C_i = (U_{i,1}^s, U_{i,2}^s, G_i^{s_i} \cdot Z_i)
\]

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 \( i = m + 1, \ldots, n \)

Pairing Product Equations

\[ \text{A}_i \in G \ (i = 1, \ldots, m), \ \zeta_i \in \mathbb{Z}_p \ (i = m + 1, \ldots, n), \text{ and } B \in G_T \public. \]

One wants to show its knowledge of \( X_i \in G \) (for \( i = 1, \ldots, m \)) and \( Z_i \in G_T \) (for \( i = m + 1, \ldots, n \)) that simultaneously satisfy

\[
\forall i \in \{1, \ldots, n\} \quad e(X_i, A_i) \cdot Z_i^{\zeta_i} = B
\]

One thus commits \( X_i \) (linear encryption) in \( G \), into \( \varphi_i \), for \( i = 1, \ldots, m \), encrypted under \( pk = (g, u_1, u_2) \), and \( Z_i \) (linear encryption) in \( G_T \), into \( C_i \), for \( i = m + 1, \ldots, n \), encrypted under \( PK_i = (G, U_1, U_2) \)

where \( G = e(g, g), U_1 = e(u_1, g), U_2 = e(u_2, g) \).

Smooth Projective Hash Function

\[
(\lambda, (\eta_i, \theta_i), i = 1, \ldots, n) \leftarrow \mathbb{Z}_p^{2n+1}, \text{ one sets } \text{hk}_i = (\eta_i, \theta_i, \lambda)
\]

and \( \text{hp}_i = (u_{i,1} \cdot g^{\lambda}, u_{i,2} \cdot g^{\lambda}) \in G^2 \)

where \( \zeta_i = 1 \) for \( i = 1, \ldots, m \).

The associated projection keys in \( G_T \) are

\[
\text{HP}_i = (e(\text{hp}_{i,1}, A_i), e(\text{hp}_{i,2}, A_i)), \text{ for } i = 1, \ldots, n,
\]

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

The hash value is

\[
H = \prod_{i=1}^{n} \prod_{i=1}^{n} \prod_{i=1}^{n} \prod_{i=1}^{n} e(X_i, A_i) \cdot Z_i^{\zeta_i} / B
\]

Equality indeed holds if and only if the equation is satisfied
Blind Signatures

We have $X_i$ committed in $G$, in $C_i$, for $i = 1, \ldots, m$ and $Z_i$ committed in $G_T$, in $C_i$, for $i = m+1, \ldots, n$. We want to show they simultaneously satisfy

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

where $A_{k,i} \subseteq G$, $B_k \subseteq G_T$, 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:

- 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 $\sigma' = M'^d \mod n$

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]
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

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` + 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?

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 $\sigma$ 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!
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

Our Scheme
A New OSBE [Blazy, Pointcheval, Vergnaud, 2012]

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 M₀ from multiple interactions with S sending M₁ 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.

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 → CDH and DLin assumptions
Extension to More Languages

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)

**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

GL – Generic Approach

<table>
<thead>
<tr>
<th>Alice</th>
<th>Bob</th>
</tr>
</thead>
<tbody>
<tr>
<td>$C_1 = \text{Commit}(pw; r_1)$</td>
<td>$C_2 = \text{Commit}(pw; r_2)$</td>
</tr>
<tr>
<td>$C_2, hp_1$</td>
<td>$hk_1, hp_1$ on $C_1$</td>
</tr>
<tr>
<td>$hp_2$</td>
<td>$hp_2$ on $C_2$</td>
</tr>
<tr>
<td>$\text{ProjHash}(hp_1; C_1, r_1) = H_1 = \text{Hash}(hk_1; C_1)$</td>
<td>$\text{Hash}(hk_2; C_2) = H_2 = \text{ProjHash}(hp_2; C_2, r_2)$</td>
</tr>
<tr>
<td>$K = H_1 \cdot H_2$</td>
<td></td>
</tr>
</tbody>
</table>

The language is: valid commitments of $pw$
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