# Efficient Smooth Projective Hash Functions and Applications

David Pointcheval

Joint work with Olivier Blazy, Céline Chevalier and Damien Vergnaud

Ecole Normale Supérieure

## 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)

→ 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 \( x \in L \), \( H(x) = \text{ProjHash}_L(hp; x, w) \quad w \) 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

**DH Language**

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

→ Public-key Encryption with IND-CCA Security

**Algorithms**

- HashKG() = \( hk = (y_1,y_3) \leftarrow Z_p^2 \)
- ProjKG(hk) = \( hp = g^{y_1}h^{y_3} \)

\[ \text{Hash}(hk,(u,v)) = u^{y_1}v^{y_3} = hp^r = \text{ProjHash}(hp,(u,v);r) \]

---

### General Tools: Signature

**Assumptions: CDH and DLin**

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

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

For any generator \( g \leftarrow \mathbb{G} \), and any scalars \( a,b \leftarrow Z_p^* \),

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 \( g \leftarrow \mathbb{G} \), and any scalars \( a,b,x,y,c \leftarrow Z_p^* \),

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. \((u, v, g)\)

---

**Examples (Con’d)**

**Commitment/Encryption**

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

→ Password-Authenticated Key Exchange in the Standard Model

**Labeled Encryption**

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

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

→ 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

---

**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)
even when having access to any signature of its choice

(Chosen-Message Attack).
### Signature: Waters

$G = \langle g \rangle$ group of order $p$, and a bilinear $e : G \times G \rightarrow G_T$

**Waters Signature**

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

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

---

### Encryption: Linear

$G = \langle g \rangle$ group of order $p$

**Linear Encryption**

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

**Security**

Linear encryption reaches IND-CPA under the DLin assumption

---

### General Tools: Encryption

**Definition (Encryption Scheme)**

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

- $\text{Setup}(1^k) \rightarrow \text{global parameters } param$
- $\text{KeyGen}(param) \rightarrow \text{pair of keys } (pk, dk)$
- $\text{Encrypt}(pk, m, r) \rightarrow \text{ciphertext } c$, using the random coins $r$
- $\text{Decrypt}(dk, c) \rightarrow \text{plaintxt, or } \bot$ 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 Cramer-Shoup

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

**Linear Cramer-Shoup Encryption**

- Keys: $dk = (x_1, x_2, x_3, y_1, y_2, y_3, z_1, z_2, z_3) \leftarrow \mathbb{Z}_p^9$,
  \[ pk = \left( \begin{array}{c} g_1, c_1 = g_1^{x_1} g_2^{y_1}, c_2 = g_1^{x_2} g_3^{y_2}, c_3 = g_2^{x_3} g_3^{y_3} \\ g_3, d_1 = g_1^{z_1}, d_2 = g_2^{z_2}, d_3 = g_3^{z_3} \end{array} \right) \]
- $Encrypt(pk = (g_1, g_2, g_3, c_1, c_2, d_1, d_2, h_1, h_2, h_3), m; (r, s))$, for $M \in G$:
  \[ C = (\sigma = (u_1 = g_1^{f_1}, u_2 = g_2^{f_2}, u_3 = g_3^{f_3 + s}), e = M \cdot h_1^{s}, v = v_1^{v_2}) \]
  where $v_1 = c_1^{d_1^{\xi}}, v_2 = c_2^{d_2^{\xi}},$ and $\xi = \mathcal{H}(\sigma, e)$
- $Decrypt(dk = (x_1, x_2, x_3, y_1, y_2, y_3, z_1, z_2, z_3), C = (\sigma, e, v))$:
  one checks $v = \left( u_1^{x_1 + \xi_1} u_2^{x_2 + \xi_2} u_3^{x_3 + \xi_3} \rightarrow M = e / u_1^{x_1} u_2^{x_2} u_3^{x_3} \right)$
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 $g^{X_i}$,

one can make a proof of knowledge of values for the $X_i$’s or $X_j$’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 [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:

$$SHS_i = \{\text{HashKG}_i, \text{ProjKG}_i, \text{Hash}_i, \text{ProjHash}_i\}$$

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

- $h_{k_1} = \text{HashKG}_1(r_1)$ $h_{k_2} = \text{HashKG}_2(r_2)$
- $h_{p_1} = \text{ProjKG}_1(h_{k_1}, c)$ $h_{p_2} = \text{ProjKG}_2(h_{k_2}, c)$

**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 $i = 1, 2$:

- $\text{HashKG}_L(r = r_1 \| r_2) = h_k = (h_{k_1}, h_{k_2})$
- $\text{ProjKG}_L(h_k, c) = h_p = (h_{p_1}, h_{p_2})$
- $\text{Hash}_L(h_k, c) = \text{Hash}_1(h_{k_1}, c) \oplus \text{Hash}_2(h_{k_2}, c)$
- $\text{ProjHash}_L(h_p, c; (w_1, w_2)) = \text{ProjHash}_1(h_{p_1}, c; w_1) \oplus \text{ProjHash}_2(h_{p_2}, c; w_2)$

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

- $\text{HashKG}_L(r = r_1 || r_2) = \text{hk} = (hk_1, hk_2)$
- $\text{ProjKG}_L(hk, c) = \text{hp} = (hp_1, hp_2, hp_\Delta)$
  where $hp_\Delta = \text{Hash}_1(hk_1, c) \oplus \text{Hash}_2(hk_2, c)$
- $\text{Hash}_L(hk, c) = \text{Hash}_1(hk_1, c)$
- $\text{ProjHash}_L(hp, c; w) = \text{ProjHash}(hp_1, c; w)$ if $c \in L_1$
  or $hp_\Delta \oplus \text{ProjHash}_2(hp_2, c; w)$ if $c \in L_2$

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

The $c_i's$ can be transposed into $G_T$, for $i = 1, \ldots, m$:

- $c_i = (u_i^{\lambda}, u_i^{\theta_i}, g_i^{r_i+s_i} \cdot X_i)$ for $i = 1, \ldots, m$
- $C_i = (U_1^{\lambda}, U_2^{\lambda}, G_i^{r_i+s_i} \cdot Z_i)$ for $i = m + 1, \ldots, n$

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

The hash value is

\[
H = \left( \prod_{i=1}^{n} C_{i,1}^{\theta_i} \cdot C_{i,2}^{\theta_i} \cdot C_{i,3}^{\theta_i} \right) \cdot \left( \prod_{i=m+1}^{n} Z_i^{\lambda} \right)
\]

Equality indeed holds if and only if the equation is satisfied.
**Introduction**

We have $X_i$ committed in $G_i$ 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

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

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

\[
\rightarrow \text{ Similar Hash Proofs on Linear Cramer-Shoup Commitments}
\]

**Blind RSA**

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

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

Indeed,

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

\[
\rightarrow \text{ Proven under the One-More RSA}
\]

[Blazy, Fuchsbauer, Pointcheval, Vergnaud, 2011]
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 $C = \mathcal{E}_{pk}(F(M))$ has to be sent

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

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?

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

With SPHF

The user $U$ and the authority $A$ use a smooth projective hash system for $L$: $C_1 = \mathcal{E}_{\rho_1}(M; r)$ and $C_2 = \mathcal{E}_{\rho_2}(M; s)$ 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 $\text{Hash} = \text{Hash}(h \cdot (C_1, C_2))$
- $U$ computes $\text{Hash} = \text{ProjHash}(h \cdot (C_1, C_2), (r, s))$, and gets $\sigma$. Granted the commutativity, $U$ decrypts it into a Waters signature of $M$, and eventually re-randomizes it

Such a protocol requires $8` + 12$ group elements in total only!
Introduction
Cryptographic Tools
More Languages
Blind Signatures
OSBE
LAKE

Definitions

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 $\sigma$ 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

Our Scheme

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 $\sigma$ on $m$ under $vk$:

With a Smooth Projective Hash Function

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

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)

### GL – Generic Approach

**[Gennaro, Lindell, 2003]**

Additional tricks are required for the security!

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

The language is: valid commitments of $pw$

### Our Construction

- With a Linear Cramer-Shoup UC commitment
- Using the GL approach
- 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 $G$ and $G_T$
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