Round-Optimal Privacy-Preserving Protocols with Smooth Projective Hash Functions

O. Blazy, D.Pointcheval, D. Vergnaud

ENS,CNRS, INRIA

Conditional Actions

Certification of a public key

Group Manager

$\mathsf{pk}, \pi \leftarrow$

$\rightarrow \mathsf{Cert}$

User

$\rightsquigarrow$ The User should know the associated sk.

Conditional Actions

Signature of a blinded message

Signer

$\mathcal{C}(M), \pi \leftarrow$

$\rightarrow \sigma$

User

$\rightsquigarrow$ The User should know the plaintext M.

Conditional Actions

Transmission of private information

Server

$Request, \pi \leftarrow$

$\rightarrow$ info

User

$\rightsquigarrow$ The User should possess some credentials.

Semantic security

- Only people with the requested secret should be able to access the information

Semantic security

- Only people with the requested secret should be able to access the information

Escrow-Freeness

- The authority should not learn whether he possesses the requested secret.

Certification of Public Keys: ZKPoK

A user can ask for the certification of a pk, but if he knows the associated sk only:

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

Certification of Public Keys: ZKPoK

A user can ask for the certification of a pk, but if he knows the associated sk only:

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

~~Escrow-Freeness~~

Certification of Public Keys: ZKPoK

A user can ask for the certification of a pk, but if he knows the associated sk only:

With a Non-Interactive Zero-Knowledge Proof of Membership

- the user U sends his public key pk, and an encryption $\mathcal{C}$ of sk together with a NIZK proof, π
- if convinced, A generates and sends the certificate Cert for pk

Certification of Public Keys: ZKPoK

A user can ask for the certification of a pk, but if he knows the associated sk only:

With a Non-Interactive Zero-Knowledge Proof of Membership

- the user U sends his public key pk, and an encryption $\mathcal{C}$ of sk together with a NIZK proof, π
- if convinced, A generates and sends the certificate Cert for pk

~~Escrow-Freeness~~

Certification of Public Keys: SPHF [ACP09]

A user can ask for the certification of pk, but if he knows the associated sk only:

With a Smooth Projective Hash Function

$\mathcal{L}$: pk and $C = \mathcal{C}(\mathsf{sk}; r)$ are associated to the same sk

- U sends his pk, and an encryption C of sk;
- A generates the certificate Cert for pk, and sends it, masked by $\mathsf{Hash} = \mathsf{Hash}(\mathsf{hk}; (\mathsf{pk}, C))$;
- U computes $\mathsf{Hash} = \mathsf{ProjHash}(\mathsf{hp}; (\mathsf{pk}, C), r))$, and gets Cert.

Certification of Public Keys: SPHF [ACP09]

A user can ask for the certification of pk, but if he knows the associated sk only:

With a Smooth Projective Hash Function

$\mathcal{L}$: pk and $C = \mathcal{C}(\mathsf{sk}; r)$ are associated to the same sk

- U sends his pk, and an encryption C of sk;
- A generates the certificate Cert for pk, and sends it, masked by $\mathsf{Hash} = \mathsf{Hash}(\mathsf{hk}; (\mathsf{pk}, C))$;
- U computes $\mathsf{Hash} = \mathsf{ProjHash}(\mathsf{hp}; (\mathsf{pk}, C), r))$, and gets Cert.

Implicit proof of knowledge of sk $\rightsquigarrow$ Escrow-Freeness

Smooth Projective Hash Functions [CS02]

Definition [CS02,GL03]

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

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

Properties

For any $x \in X$, $H(x) = \mathrm{Hash}_L(\mathrm{hk}; x)$
For any $x \in L$, $H(x) = \mathrm{ProjHash}_L(\mathrm{hp}; x, w)$ $\quad w$ 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 $X \setminus L$

Properties

For any $x \in X$, $H(x) = \mathsf{Hash}_L(\mathsf{hk}; x)$
For any $x \in L$, $H(x) = \mathsf{ProjHash}_L(\mathsf{hp}; x, w)$ $\quad$ 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 $X \setminus L$

Properties

For any $x \in X$, $H(x) = \mathrm{Hash}_L(\mathrm{hk}; x)$
For any $x \in L$, $H(x) = \mathrm{ProjHash}_L(\mathrm{hp}; x, w)$ $\quad w$ 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 $X \setminus L$

Properties

For any $x \in X$, $H(x) = \mathsf{Hash}_L(\mathsf{hk}; x)$
For any $x \in L$, $H(x) = \mathsf{ProjHash}_L(\mathsf{hp}; x, w)$ $\quad$ 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 $X \setminus L$

Transmission of private information

Server

$\textcolor{purple}{Request}, \mathcal{C} \leftarrow$

$\rightarrow \text{info} \oplus H_L$

User

Definition (CDH)

Given $g, g^a, h \in \mathbb{G}^3$, it is hard to compute h^a.

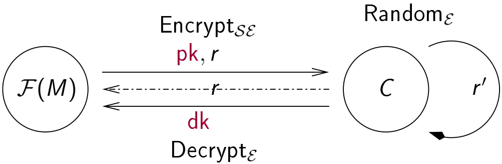
Definition (DLin)

Given $u, v, w, u^a, v^b, w^c \in \mathbb{G}^6$, it is hard to decide wether $c = a + b$.

Definition (Encryption Scheme)

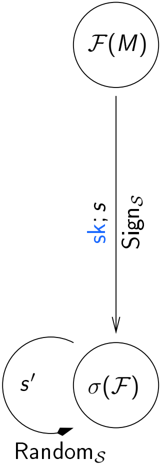
$\mathcal{E} = (\mathsf{Setup}, \mathsf{EKeyGen}, \mathsf{Encrypt}, \mathsf{Decrypt})$:

- $\mathsf{Setup}(1^\lambda)$: param;
- EKeyGen(param): public *encryption* key pk, private *decryption* key dk;
- $\mathsf{Encrypt}(\mathsf{pk}, m; r)$: ciphertext c on $m \in \mathcal{M}$ and pk;
- $\mathsf{Decrypt}(\mathsf{dk}, c)$: decrypts c under dk.



Definition (Linear Encryption (BBS04))

- Setup(1^λ): Generates a multiplicative group $(p, \mathbb{G}, g)$.
- EKeyGen$_\mathcal{E}$(param):dk $= (\mu, \nu) \stackrel{\$}{\leftarrow} \mathbb{Z}_p^2$, and pk $= (X_1 = g^\mu, X_2 = g^\nu)$.
- Encrypt(pk $= (X_1, X_2), M; \alpha, \beta$): For M, and random$\alpha, \beta \stackrel{\$}{\leftarrow} \mathbb{Z}_p^2$, defines as $\mathcal{C} = \left(c_1 = X_1^\alpha, c_2 = X_2^\beta, c_3 = g^{\alpha+\beta} \cdot M\right)$.
- Decrypt(dk $= (\mu, \nu), \mathcal{C} = (c_1, c_2, c_3)$): Computes $M = c_3/(c_1^{1/\mu} c_2^{1/\nu})$.



Definition (Signature Scheme)

$\mathcal{S} = (\mathsf{Setup}, \mathsf{SKeyGen}, \mathsf{Sign}, \mathsf{Verif})$:

- $\mathsf{Setup}(1^\lambda)$: param;
- $\mathsf{SKeyGen}(\mathsf{param})$: public *verification* key vk, private *signing* key sk;
- $\mathsf{Sign}(\mathsf{sk}, m; s)$: signature σ on m, under sk;
- $\mathsf{Verif}(\mathsf{vk}, m, \sigma)$: checks whether σ is valid on m.

Definition (Waters Signature (Wat05))

- $\mathsf{Setup}_{\mathcal{S}}(1^\lambda)$: Generates $(p, \mathbb{G}, \mathbb{G}_T, e, g)$, an extra h, and (u_i) for the Waters function ($\mathcal{F}(m) = u_0 \prod_i u_i^{m_i}$).
- $\mathsf{SKeyGen}_{\mathcal{S}}(\mathsf{param})$: Picks $x \xleftarrow{\$} \mathbb{Z}_p$ and outputs $\mathsf{sk} = Y = h^x$, and $\mathsf{vk} = X = g^x$;
- $\mathsf{Sign}(\mathsf{sk}, m; \mu)$: Outputs $\sigma(m) = (Y\mathcal{F}(m)^\mu, g^{-\mu})$;
- $\mathsf{Verif}(\mathsf{vk}, m, \sigma)$: Checks the validity of σ,$(e(g, \sigma_1) \cdot e(\mathcal{F}(m), \sigma_2) \stackrel{?}{=} e(X, h))$

Oblivious Signature-Based Envelope [LDB03]

A sender S wants to send a message M to U such that

- U gets M iff it owns $\sigma(m)$ 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.

Oblivious Signature-Based Envelope [LDB03]

A sender S wants to send a message M to U such that

- U gets M iff it owns $\sigma(m)$ 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.

Oblivious Signature-Based Envelope [LDB03]

A sender S wants to send a message M to U such that

- U gets M iff it owns $\sigma(m)$ 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.

One-Round OSBE from IBE

The authority owns the master key of an IBE scheme,
and provides the decryption key (signature) associated to m to U.
S wants to send a message M to U, if U owns a valid signature.

- S encrypts M under the identity m.

Security properties

- Correct: trivial
- Oblivious: no message sent!
- Semantic Security: IND-CPA of the IBE

But the authority can decrypt everything!

One-Round OSBE from IBE

The authority owns the master key of an IBE scheme,
and provides the decryption key (signature) associated to m to U.
S wants to send a message M to U, if U owns a valid signature.

- S encrypts M under the identity m.

Security properties

- Correct: trivial
- Oblivious: no message sent!
- Semantic Security: IND-CPA of the IBE

But the authority can decrypt everything!

One-Round OSBE from IBE

The authority owns the master key of an IBE scheme,
and provides the decryption key (signature) associated to m to U.
S wants to send a message M to U, if U owns a valid signature.

- S encrypts M under the identity m.

Security properties

- Correct: trivial
- Oblivious: no message sent!
- Semantic Security: IND-CPA of the IBE

But the authority can decrypt everything!

A Stronger Security Model

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

Security Notions

- Escrow-free (Oblivious w.r.t. the authority):
  the authority does not know whether U uses a valid signature;
- Semantic Security: U cannot distinguish multiple interactions with :
  S sending M_0 from those 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.

Our New OSBE

S wants to send a message M to U, if U owns a valid $\sigma(M)$ under vk:

With a Smooth Projective Hash Function

$\mathcal{L}$: $C = \mathcal{C}(\sigma, r)$ contains a valid $\sigma(m)$ under vk

- the user U sends an encryption C of σ;
- A generates hk and the associated hp,
  computes $H = \mathsf{Hash}(\mathsf{hk}; C)$,
  and sends hp together with $c = P \oplus H$;
- U computes $X = \mathsf{ProjHash}(\mathsf{hp}; C, r)$, and gets P.

$Lin(\mathsf{pk}, m) : \mathcal{L}(\mathcal{C}(m)) \quad \rightsquigarrow \quad WLin(\mathsf{pk}, \mathsf{vk}, M) : \mathcal{L}(\mathcal{C}(\sigma(m)))$

Security Properties

- ✓ Oblivious/Escrow-free: 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 $\rightsquigarrow$ round-optimal
Standard model with Waters Signature + Linear Encryption $\rightsquigarrow$ CDH and DLin

Security Properties

- ✓ Oblivious/Escrow-free: 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 $\rightsquigarrow$ round-optimal

Standard model with Waters Signature + Linear Encryption $\rightsquigarrow$ CDH and DLin

Security Properties

- ✓ Oblivious/Escrow-free: 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 $\rightsquigarrow$ round-optimal
Standard model with Waters Signature + Linear Encryption $\rightsquigarrow$ CDH and DLin

Electronic Coins — Chaum 81

Expected properties

- ✓ Coins are signed by the bank: *Unforgeability*
- ✓ Coins should be distinct to prevent *Double-Spending*
- ✓ Bank should not know to whom it gave a coin: *Anonymity*

Protocol

- Withdrawal: A user get a coin c from the bank
- Spending: A user pays a shop with the coin c
- Deposit: The shop gives the coin c back to the bank

Anonymity

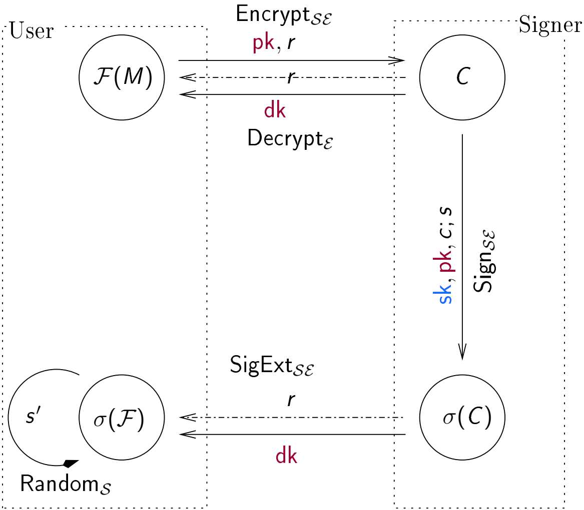
- The bank cannot link a withdrawal to a deposit

No double-spending

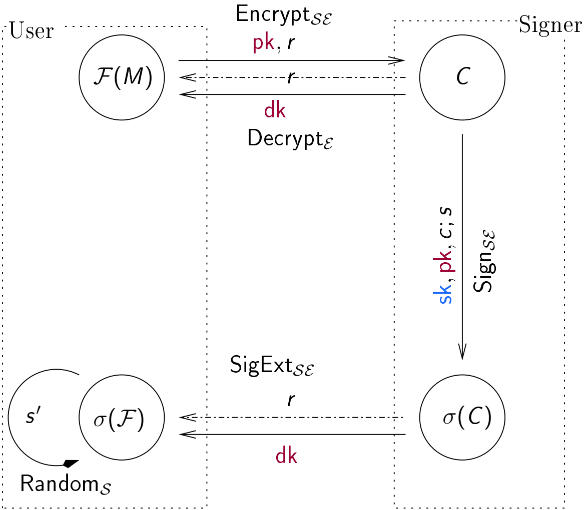
- A coin should not be used twice

Definition (Blind Signature)

A blind signature allows a user to get a message m signed by an authority into σ so that the authority *even powerful* cannot recognize later the pair (m, σ).

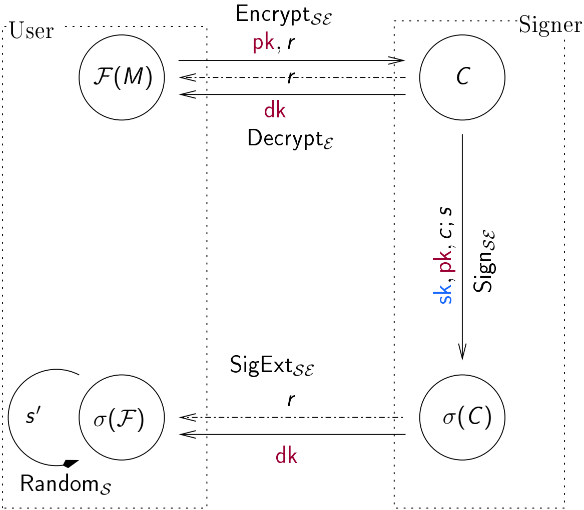


User
Signer
Encrypt$_{\mathcal{SE}}$
pk, r
$\mathcal{F}(M)$
C
r
dk
Decrypt$_{\mathcal{E}}$
sk, pk, c; s
Sign$_{\mathcal{SE}}$
SigExt$_{\mathcal{SE}}$
r
s'
$\sigma(\mathcal{F})$
$\sigma(C)$
dk
Random$_{\mathcal{S}}$



Groth Sahai

$9\,\ell + 24$



Groth Sahai

$9\,\ell + 24$

SPHF

$8\,\ell + 12$

Smooth Projective Hash Functions $\hat{=}$ *implicit* proofs of knowledge

Smooth Projective Hash Functions $\hat{=}$ *implicit* proofs of knowledge

Various Applications:

Privacy-preserving protocols:

△ Many more Round optimal application?

Smooth Projective Hash Functions $\triangleq$ *implicit* proofs of knowledge

Various Applications:

✓ IND-CCA [CS02]

Privacy-preserving protocols:

△ Many more Round optimal application?

Smooth Projective Hash Functions $\hat{=}$ *implicit* proofs of knowledge

Various Applications:

- ✓ IND-CCA [CS02]
- ✓ PAKE [GL03]

Privacy-preserving protocols:

△ Many more Round optimal application?

Smooth Projective Hash Functions $\hat{=}$ *implicit* proofs of knowledge

Various Applications:

- ✓ IND-CCA [CS02]
- ✓ PAKE [GL03]
- ✓ Certification of Public Keys [ACP09]

Privacy-preserving protocols:

△ Many more Round optimal application?

Smooth Projective Hash Functions $\hat{=}$ *implicit* proofs of knowledge

Various Applications:

- ✓ IND-CCA [CS02]
- ✓ PAKE [GL03]
- ✓ Certification of Public Keys [ACP09]

Privacy-preserving protocols:

△ Many more Round optimal application?

Smooth Projective Hash Functions $\hat{=}$ *implicit* proofs of knowledge

Various Applications:

- ✓ IND-CCA [CS02]
- ✓ PAKE [GL03]
- ✓ Certification of Public Keys [ACP09]

Privacy-preserving protocols:

- ✓ Blind signatures

△ Many more Round optimal application?

Smooth Projective Hash Functions $\hat{=}$ *implicit* proofs of knowledge

Various Applications:

- ✓ IND-CCA [CS02]
- ✓ PAKE [GL03]
- ✓ Certification of Public Keys [ACP09]

Privacy-preserving protocols:

- ✓ Blind signatures
- ✓ Oblivious Signature-Based Envelope

△ Many more Round optimal application?

Smooth Projective Hash Functions $\triangleq$ *implicit* proofs of knowledge

Various Applications:

- ✓ IND-CCA [CS02]
- ✓ PAKE [GL03]
- ✓ Certification of Public Keys [ACP09]

Privacy-preserving protocols:

- ✓ Blind signatures
- ✓ Oblivious Signature-Based Envelope

△ Many more Round optimal application?