Interactive and Non-Interactive Proofs of Knowledge

Olivier Blazy

ENS / CNRS / INRIA / Paris 7 → RUB

Sept 2012
1 General Remarks

2 Building blocks

3 Non-Interactive Proofs of Knowledge

4 Interactive Implicit Proofs
1. General Remarks
2. Building blocks
3. Non-Interactive Proofs of Knowledge
4. Interactive Implicit Proofs
1 General Remarks

2 Building blocks

3 Non-Interactive Proofs of Knowledge

4 Interactive Implicit Proofs
1. General Remarks

2. Building blocks

3. Non-Interactive Proofs of Knowledge

4. Interactive Implicit Proofs
Proof of Knowledge

- Interactive method for one party to prove to another the knowledge of a secret $S$.

1. **Completeness:** $S$ is true $\implies$ verifier will be convinced of this fact

2. **Soundness:** $S$ is false $\implies$ no cheating prover can convince the verifier that $S$ is true

Classical Instantiations: Schnorr proofs, Sigma Protocols...
Zero-Knowledge Proof Systems

- Introduced in 1985 by Goldwasser, Micali and Rackoff.
  - Reveal nothing other than the validity of assertion being proven
- Used in many cryptographic protocols
  - Anonymous credentials
  - Anonymous signatures
  - Online voting
  - ...
Zero-Knowledge Proof Systems

- Introduced in 1985 by Goldwasser, Micali and Rackoff.

  ~⇒ Reveal nothing other than the validity of assertion being proven

- Used in many cryptographic protocols
  - Anonymous credentials
  - Anonymous signatures
  - Online voting
  - ...

Zero-Knowledge Proof Systems

- Introduced in 1985 by Goldwasser, Micali and Rackoff.

  ⇒ Reveal nothing other than the validity of assertion being proven

- Used in many cryptographic protocols
  - Anonymous credentials
  - Anonymous signatures
  - Online voting
  - ...
Zero-Knowledge Interactive Proof

- Interactive method for one party to prove to another that a statement $S$ is true, without revealing anything other than the veracity of $S$.

1. Completeness: if $S$ is true, the honest verifier will be convinced of this fact.
2. Soundness: if $S$ is false, no cheating prover can convince the honest verifier that it is true.
3. Zero-knowledge: if $S$ is true, no cheating verifier learns anything other than this fact.
Zero-Knowledge Interactive Proof

interactive method for one party to prove to another that a statement \( S \) is true, without revealing anything other than the veracity of \( S \).

1. **Completeness:** if \( S \) is true, the honest verifier will be convinced of this fact
2. **Soundness:** if \( S \) is false, no cheating prover can convince the honest verifier that it is true
3. **Zero-knowledge:** if \( S \) is true, no cheating verifier learns anything other than this fact.
Non-Interactive Zero-Knowledge Proof

- non-interactive method for one party to prove to another that a statement $S$ is true, without revealing anything other than the veracity of $S$.

1. **Completeness**: $S$ is true $\Rightarrow$ verifier will be convinced of this fact
2. **Soundness**: $S$ is false $\Rightarrow$ no cheating prover can convince the verifier that $S$ is true
3. **Zero-knowledge**: $S$ is true $\Rightarrow$ no cheating verifier learns anything other than this fact.
History of NIZK Proofs

Inefficient NIZK
- ...
- De Santis-Di Crescenzo-Persiano, 2002.

Alternative: Fiat-Shamir heuristic, 1986: interactive ZK proof $\leadsto$ NIZK
But limited by the Random Oracle

Efficient NIZK
History of NIZK Proofs

Inefficient NIZK

- ...
- De Santis-Di Crescenzo-Persiano, 2002.

Alternative: Fiat-Shamir heuristic, 1986: interactive ZK proof $\leadsto$ NIZK
But limited by the Random Oracle

Efficient NIZK

History of NIZK Proofs

**Inefficient NIZK**


  ...  

- De Santis-Di Crescenzo-Persiano, 2002.

**Alternative:** Fiat-Shamir heuristic, 1986: interactive ZK proof $\leadsto$ NIZK

But limited by the Random Oracle

**Efficient NIZK**


Applications of NIZK Proofs

- Fancy signature schemes
  - group signatures
  - ring signatures
  - traceable signatures

- Efficient non-interactive proof of correctness of shuffle

- Non-interactive anonymous credentials

- CCA-2-secure encryption schemes (with public verifiability)

- Identification

- E-voting, E-cash

...
Conditional Actions

Certification of a public key

Group Manager

User

\[ \pi \rightsquigarrow \text{The User should know the associated } sk. \]
Conditional Actions

Signature of a blinded message

Signer $\rightarrow \sigma \pi \rightarrow C(M) \leftarrow \sigma$

$\pi \rightsquigarrow$ The User should know the plaintext $M$. 
Conditional Actions

Transmission of private information

Server

Request ←
→ info

π ⇝ The User should possess some credentials.

User
**Soundness**

- Only people proving they know the expected secret should be able to access the information.

**Zero-Knowledge**

- The authority should not learn said secret.
1 General Remarks

2 Building blocks
- Bilinear groups aka Pairing-friendly environments
- Commitment / Encryption
- Signatures
- Security hypotheses

3 Non-Interactive Proofs of Knowledge

4 Interactive Implicit Proofs
(p, G, GT, e, g) bilinear structure:

- G, GT multiplicative groups of order p
  - p = prime integer
- ⟨g⟩ = G
- e : G × G → GT
  - ⟨e(g, g)⟩ = GT
  - e(g^a, g^b) = e(g, g)^{ab}, a, b ∈ Z

deciding group membership,

- group operations,
- bilinear map

\{ \text{efficiently computable.} \}
Definition (Encryption Scheme)

\[ \mathcal{E} = (\text{Setup, EKeyGen, Encrypt, Decrypt}): \]
- Setup(1^k): param;
- EKeyGen(param): public encryption key \( pk \), private decryption key \( dk \);
- Encrypt(pk, m; r): ciphertext \( c \) on \( m \in \mathcal{M} \) and \( pk \);
- Decrypt(dk, c): decrypts \( c \) under \( dk \).

\[ \mathcal{F}(M) \xrightarrow{\text{Encrypt}_{\mathcal{E}} \text{ pk, r}} \text{C} \xleftarrow{\text{Random}_{\mathcal{E}}} \]

\[ \text{Decrypt}_{\mathcal{E}} \text{ dk} \]

\textbf{Indistinguishability:}
Given \( M_0, M_1 \), it should be hard to guess which one is encrypted in \( C \).
**Definition (Linear Encryption)**  
(BBS04)

- **Setup**($1^κ$): Generates a multiplicative group $(p, \mathbb{G}, g)$.

- **EKeyGen**$_{\mathcal{E}}$(param):  
  $dk = (\mu, \nu) \xleftarrow{\$} \mathbb{Z}_p^2$, and $pk = (X_1 = g^\mu, X_2 = g^\nu)$.

- **Encrypt**($pk = (X_1, X_2), M; \alpha, \beta$): For $M$, and random $\alpha, \beta \xleftarrow{\$} \mathbb{Z}_p^2$,  
  $C = (c_1 = X_1^\alpha, c_2 = X_2^\beta, c_3 = g^{\alpha+\beta} \cdot M)$.

- **Decrypt**($dk = (\mu, \nu), C = (c_1, c_2, c_3)$): Computes $M = c_3/(c_1^{1/\mu} c_2^{1/\nu})$.

---

**Randomization**

Random($pk, C; r, s$) :  
$C' = (c_1 X_1^r, c_2 X_2^s, c_3 g^{r+s}) = (X_1^{\alpha+r}, X_2^{\beta+s}, g^{\alpha+r+\beta+s} \cdot M)$
Definition (Commitment Scheme)

\[ \mathcal{E} = (\text{Setup, Commit, Decommit}): \]
- \text{Setup}(1^\mathbb{R}): \text{param, ck}\
- \text{Commit}(\text{ck}, m; r): \text{c on the input message } m \in \mathcal{M} \text{ using } r \leftarrow \mathcal{R}\
- \text{Decommit}(\text{c}, m; w) \text{ opens } \text{c and reveals } m, \text{together with } w \text{ that proves the correct opening.}
Pedersen

- Setup($1^k$): $g, h \in \mathbb{G}$;
- Commit($m; r$): $c = g^m h^r$;
- Decommit($c, m; r$): $c \overset{?}{=} g^m h^r$. 
Definition (Signature Scheme)

\[ S = (\text{Setup}, \text{SKeyGen}, \text{Sign}, \text{Verif}) : \]
- \( \text{Setup}(1^k) : \text{param} ; \)
- \( \text{SKeyGen}(\text{param}) : \text{public verification key } \text{vk}, \text{private signing key } \text{sk} ; \)
- \( \text{Sign}(\text{sk}, m; s) : \text{signature } \sigma \text{ on } m, \text{under } \text{sk} ; \)
- \( \text{Verif}(\text{vk}, m, \sigma) : \text{checks whether } \sigma \text{ is valid on } m. \)

**Unforgeability:**
Given \( q \) pairs \( (m_i, \sigma_i) \), it should be hard to output a valid \( \sigma \) on a fresh \( m \).
Definition (Waters Signature) (Wat05)

- **Setup**\(_S(1^k)\): Generates \((p, \mathbb{G}, \mathbb{G}_T, e, g)\), an extra \(h\), and \((u_i)\) for the Waters function \(\mathcal{F}(m) = u_0 \prod_i u_i^{m_i}\).

- **SKeyGen**\(_S(\text{param})\): Picks \(x \xleftarrow{\$} \mathbb{Z}_p\) and outputs \(sk = h^x\), and \(vk = g^x\);

- **Sign**(sk, m; s): Outputs \(\sigma(m) = (sk\mathcal{F}(m)^s, g^s)\);

- **Verif**(vk, m, σ): Checks the validity of \(\sigma\): 
  \[ e(g, \sigma_1) = e(\mathcal{F}(m), \sigma_2) \cdot e(vk, h) \]

Randomization

Random(σ; r) : \(\sigma' = (\sigma_1\mathcal{F}(m)^r, \sigma_2g^r) = (sk\mathcal{F}(m)^{r+s}, g^{r+s})\)
Definition (DL)

Given $g, h \in \mathbb{G}^2$, it is hard to compute $\alpha$ such that $h = g^\alpha$.

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 whether $c = a + b$. 
1. General Remarks

2. Building blocks

3. Non-Interactive Proofs of Knowledge
   - Groth Sahai methodology
   - Motivation
   - Signature on Ciphertexts
   - Application to other protocols
   - Waters Programmability

4. Interactive Implicit Proofs
Groth-Sahai Proof System

- **Pairing product equation (PPE):** for variables \(X_1, \ldots, X_n \in G\)

\[
    (E) : \prod_{i=1}^{n} e(A_i, X_i) \prod_{i=1}^{n} \prod_{j=1}^{n} e(X_i, X_j)^{\gamma_{i,j}} = t_T
\]

determined by \(A_i \in G, \gamma_{i,j} \in \mathbb{Z}_p\) and \(t_T \in G_T\).

- Groth-Sahai \(\Rightarrow\) WI proofs that elements in \(G\) that were committed to satisfy PPE

**Setup**(\(G\)): commitment key \(ck\);
**Com**(\(ck, X \in G; \rho\)): commitment \(\overrightarrow{c_X}\) to \(X\);
**Prove**(\(ck, (X_i, \rho_i)_{i=1,\ldots,n}, (E)\)): proof \(\phi\);
**Verify**(\(ck, \overrightarrow{c_X}, (E), \phi\)): checks whether \(\phi\) is valid.
Groth-Sahai Proof System

- **Pairing product equation (PPE):** for variables $X_1, \ldots, X_n \in \mathbb{G}$

  $$(E) : \prod_{i=1}^{n} e(A_i, X_i) \prod_{i=1}^{n} \prod_{j=1}^{n} e(X_i, X_j)^{\gamma_{i,j}} = t_T$$

  determined by $A_i \in \mathbb{G}$, $\gamma_{i,j} \in \mathbb{Z}_p$ and $t_T \in \mathbb{G}_T$.

- **Groth-Sahai $\rightsquigarrow$ W1 proofs** that elements in $\mathbb{G}$ that were committed to satisfy PPE

  - **Setup($\mathbb{G}$):** commitment key $\mathbf{ck}$;
  - **Com($\mathbf{ck}, X \in \mathbb{G}; \rho$):** commitment $\mathbf{c}_X$ to $X$;
  - **Prove($\mathbf{ck}, (X_i, \rho_i)_{i=1,\ldots,n}, (E)$):** proof $\phi$;
  - **Verify($\mathbf{ck}, \mathbf{c}_X, (E), \phi$):** checks whether $\phi$ is valid.
\[(E) : \prod_{i=1}^{n} e(A_i, \mathcal{X}_i) \prod_{i=1}^{n} \prod_{j=1}^{n} e(\mathcal{X}_i, \mathcal{X}_j)^{\gamma_{i,j}} = t_T\]

<table>
<thead>
<tr>
<th>Assumption</th>
<th>DLin</th>
<th>SXDH</th>
<th>SD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Variables</td>
<td>3</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>PPE</td>
<td>9</td>
<td>(2,2)</td>
<td>1</td>
</tr>
<tr>
<td>Linear</td>
<td>3</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>Verification</td>
<td>$12n + 27$</td>
<td>$5m + 3n + 16$</td>
<td>$n + 1$</td>
</tr>
<tr>
<td>[ACNS 2010: BFI+]</td>
<td>$3n + 6$</td>
<td>$m + 2n + 8$</td>
<td>$n + 1$</td>
</tr>
</tbody>
</table>

Properties:
- correctness
- soundness
- witness-indistinguishability
- randomizability Commitments and proofs are publicly randomizable.
\[(E) : \prod_{i=1}^{n} e(A_i, \chi_i) \prod_{i=1}^{n} \prod_{j=1}^{n} e(\chi_i, \chi_j)^{\gamma_{i,j}} = t_T\]

<table>
<thead>
<tr>
<th>Assumption</th>
<th>DLin</th>
<th>SXDH</th>
<th>SD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Variables</td>
<td>3</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>PPE</td>
<td>9</td>
<td>(2,2)</td>
<td>1</td>
</tr>
<tr>
<td>Linear</td>
<td>3</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>Verification</td>
<td>12n + 27</td>
<td>5m + 3n + 16</td>
<td>n + 1</td>
</tr>
<tr>
<td>[ACNS 2010: BFI+]</td>
<td>3n + 6</td>
<td>m + 2n + 8</td>
<td>n + 1</td>
</tr>
</tbody>
</table>

Properties:

- correctness
- soundness
- witness-indistinguishability
- randomizability Commitments and proofs are publicly randomizable.
\[(E) : \prod_{i=1}^{n} e(A_i, X_i) \prod_{i=1}^{n} \prod_{j=1}^{n} e(X_i, X_j)^{\gamma_{i,j}} = t_T\]

<table>
<thead>
<tr>
<th>Assumption</th>
<th>DLin</th>
<th>SXDH</th>
<th>SD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Variables</td>
<td>3</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>PPE</td>
<td>9</td>
<td>(2,2)</td>
<td>1</td>
</tr>
<tr>
<td>Linear</td>
<td>3</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>Verification</td>
<td>12n + 27</td>
<td>5m + 3n + 16</td>
<td>n + 1</td>
</tr>
<tr>
<td>[ACNS 2010: BFI+]</td>
<td>3n + 6</td>
<td>m + 2n + 8</td>
<td>n + 1</td>
</tr>
</tbody>
</table>

**Properties:**
- correctness
- soundness
- witness-indistinguishability
- randomizability Commitments and proofs are publicly randomizable.
Electronic Voting

For dessert, we let people vote

✓ Chocolate Cake
✓ Cheese Cake
✓ Fruit Salad
✓ Brussels Sprout

After collection, we count the number of ballots:

Chocolate Cake  123
Cheese Cake     79
Fruit Salad     42
Brussels sprout  1
Authentication
- Only people authorized to vote should be able to vote
- People should be able to vote only once

Anonymity
- Votes and voters should be anonymous
- Receipt freeness
The voter generates his vote $v$.
- The voter encrypts $v$ to the server as $c$.
- The voter signs $c$ and outputs $\sigma$.
- $(c, \sigma)$ is a ballot unique per voter, and anonymous.
- Counting: granted homomorphic encryption $C = \prod c$.
- The server decrypts $C$. 

Homomorphic Encryption and Signature approach
Electronic Cash

Identify

Deposit

Withdraw

Spend

Randomize

Randomize
Protocol

- Withdrawal: A user gets 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

Electronic Coins

Expected properties

- **Unforgeability** $\rightsquigarrow$ Coins are signed by the bank
- **No Double-Spending** $\rightsquigarrow$ Each coin is unique
- **Anonymity** $\rightsquigarrow$ Blind Signature

Definition (Blind Signature)

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

Chaum 81
Round-Optimal Blind Signature

- The user encrypts his message \( m \) in \( c \).
- The signer then signs \( c \) in \( \sigma \).
- The user verifies \( \sigma \).
- He then encrypts \( \sigma \) and \( c \) into \( C_{\sigma} \) and \( C \) and generates a proof \( \pi \).
- \( \pi \): \( C_{\sigma} \) is an encryption of a signature over the ciphertext \( c \) encrypted in \( C \), and this \( c \) is indeed an encryption of \( m \).
- Anyone can then use \( C, C_{\sigma}, \pi \) to check the validity of the signature.
Vote
- A user should be able to encrypt a ballot.
- He should be able to sign this encryption.
- Receiving this vote, one should be able to randomize for Receipt-Freeness.

E-Cash
- A user should be able to encrypt a token
- The bank should be able to sign it providing Unforgeability
- This signature should now be able to be randomized to provide Anonymity

Our Solution
- Same underlying requirements;
- Advance security notions in both schemes requires to extract some kind of signature on the associated plaintext;
- General Framework for Signature on Randomizable Ciphertexts;
- Revisited Waters, Commutative encryption / signature.
Commutative properties

Encrypt

To encrypt a message $m$:

$$c = (pk_1 r_1, pk_2 r_2, \mathcal{F}(m) \cdot g^{r_1 + r_2})$$
Commutative properties

Encrypt

To encrypt a message $m$:

$$c = (pk_1^{r_1}, pk_2^{r_2}, \mathcal{F}(m) \cdot g^{r_1+r_2})$$

Sign $\circ$ Encrypt

To sign a valid ciphertext $c_1, c_2, c_3$, one has simply to produce.

$$\sigma = (c_1^s, c_2^s, sk \cdot c_3^s, pk_1^s, pk_2^s, g^s).$$
Commutative properties

Encrypt

To encrypt a message $m$:

$$c = (pk_1^{r_1}, pk_2^{r_2}, \mathcal{F}(m) \cdot g^{r_1+r_2})$$

Sign $\circ$ Encrypt

To sign a valid ciphertext $c_1, c_2, c_3$, one has simply to produce.

$$\sigma = (c_1^s, c_2^s, sk \cdot c_3^s, pk_1^s, pk_2^s, g^s).$$

Decrypt $\circ$ Sign $\circ$ Encrypt

Using $dk$.

$$\sigma = (\sigma_3 / \sigma_1^{dk_1} \cdot \sigma_2^{dk_2}, \sigma_6) = (sk \cdot \mathcal{F}(m)^s, g^s).$$
**Definition (Signature on Ciphertexts)**

\[ SE = (\text{Setup, SKeyGen, EKeyGen, Encrypt, Sign, Decrypt, Verif}) \]

- **Setup**\( (1^R) \): param\(_e\), param\(_s\);
- **EKeyGen**\( (\text{param}_e) \): pk, dk;
- **SKeyGen**\( (\text{param}_s) \): vk, sk;
- **Encrypt**\( (\text{pk}, \text{vk}, m; r) \): produces c on \( m \in \mathcal{M} \) and pk;
- **Sign**\( (\text{sk}, \text{pk}, c; s) \): produces \( \sigma \), on the input c under sk;
- **Decrypt**\( (\text{dk}, \text{vk}, c) \): decrypts c under dk;
- **Verif**\( (\text{vk}, \text{pk}, c, \sigma) \): checks whether \( \sigma \) is valid.

**Definition (Extractable Randomizable Signature on Ciphertexts)**

\[ SE' = (\text{Setup, SKeyGen, EKeyGen, Encrypt, Sign, Random, Decrypt, Verif, SigExt}) \]

- **Random**\( (\text{vk}, \text{pk}, c, \sigma; r', s') \) produces \( c' \) and \( \sigma' \) on \( c' \), using additional coins;
- **SigExt**\( (\text{dk}, \text{vk}, \sigma) \) outputs a signature \( \sigma^* \).
Randomizable Signature on Ciphertexts

\[ \text{Encrypt}_{SE} \quad \text{pk}, r \]

\[ \text{Decrypt}_{E} \]

\[ \text{Sign}_{S} \quad \text{sk}, s \]

\[ \text{Sign}_{SE} \quad \text{sk}, \text{pk}, c, s \]

\[ \text{Random}_{S} \]

\[ \sigma(M) \]

\[ \sigma(C) \]
Extractable SRC

Diagram:

- $M$ interacts with $C$ through $\sigma(M)$ and $\sigma(C)$.
- $sk, s$ and $sk, pk, c, s$ are involved in signing.
- $Encrypt_{SE}$ and $Decrypt_{SE}$ are used for encryption and decryption.
- $SigExt_{SE}$ and $Sign_{SE}$ are for signature extensions and signing.
- $Random_S$ and $Random_{SE}$ are used for randomness.

Equations:

- $\sigma(M)$
- $\sigma(C)$
- $Encrypt_{SE}(pk, r)$
- $Decrypt_{SE}(sk)$

O. Blazy (ENS → RUB)
E-Voting

[PKC 2011: BFPV]

\[ \mathcal{F}(M) \xrightarrow{\text{Encrypt}_{SE}} \mathcal{C} \xleftarrow{r} \]

User

\[ \text{Sign}_{SE}^{sk, pk, c, s} \]

Authority

\[ \mathcal{F} \]

\[ \sigma(\mathcal{F}) \xleftarrow{dk} \]

\[ \text{SigExt}_{SE} \]

\[ \sigma(\mathcal{C}) \rightarrow r', s \]

\[ \text{Random}_{SE} \]
Blind Signature

[PKC 2011: BFPV]

\[ \sigma(C) \]

\[ \sigma(F) \]

User

Signer

\[ F(M) \]

\[ \mathcal{E} \]

Encrypt\(_{\mathcal{E}}\)

Decrypt\(_{\mathcal{E}}\)

Sign\(_{\mathcal{E}}\)

SigExt\(_{\mathcal{E}}\)

Random\(_S\)

\[ s' \]

\[ \sigma(F) \]

\[ \mathcal{F}(M) \]

\[ \mathcal{E} \]

pk, r

dk

sk, pk, c, s

O. Blazy (ENS → RUB)
Partially-Blind Signature

User

info

$C' = C(M, \text{info})$

Signer

$\sigma(C')$
Partially-Blind Signature

\[ C' = C(M, \text{info}) \]

\[ \sigma(C', \text{info}_s) \]
Signer-Friendly Partially Blind Signature [SCN 2012: BPV]

User

\[ F(M) \]

\[ s' \]

Random

\[ s' \]

\[ \sigma(F') \]

Signer

\[ C_{\text{info}} \]

\[ C' \]

\[ \sigma(C') \]

info_s

Verif

Unblind_{BS}

\[ r \]

Blind_{BS}

\[ pk_{BS}, r \]

\[ \text{Sig}_{BS} \]

\[ \text{sk}_{BS}, C', \text{info}_s, s \]
Multi-Source Blind Signatures

Wireless Sensor Network

Captors

Central Hub

Receiver

\[ C = \prod c_i \]

\[ \sigma(C, s) \]
Multi-Source Blind Signatures

\[ \text{Random}_S \]
\[ s' \]
\[ \mathcal{R} \]

\[ \mathcal{F}_i \]

\[ \text{User } i \]

\[ \text{Blind}_{BS} \]
\[ p_{k_{BS}}, r_i \]

\[ C_i \]

\[ \text{Signer} \]

\[ \sigma(\prod \mathcal{F}) \]

\[ \text{Random}_S \]

\[ \sigma(\prod C_i) \]

\[ \text{Unblind}_{BS} \]

\[ \text{Verif} \]

\[ \text{Unblind}_{BS} \]

\[ \text{Verif} \]
Two solutions

Different Generators

- Each captor has a disjoint set of generators for the Waters function
- Enormous public key
Two solutions

Different Generators

- Each captor has a disjoint set of generators for the Waters function
- Enormous public key
Two solutions

**Different Generators**
- Each captor has a disjoint set of generators for the Waters function
- Enormous public key

**A single set of generators**
- The captors share the same set of generators
- Waters over a non-binary alphabet?
Two solutions

Different Generators
- Each captor has a disjoint set of generators for the Waters function
- Enormous public key

A single set of generators
- The captors share the same set of generators
- Waters over a non-binary alphabet?
Programmability of Waters over a non-binary alphabet

**Definition ((m, n)-programmability)**

$F$ is $(m, n)$ programmable if given $g, h$ there is an efficient trapdoor producing $a_X, b_X$ such that $F(X) = g^{a_X} h^{b_X}$, and for all $X_i, Z_j$,

$Pr[a_{X_1} = \ldots = a_{X_m} = 0 \land a_{Z_1} \cdot \ldots \cdot a_{Z_n} \neq 0]$ is not negligible.

**Why do we need it:** Unforgeability, $q$ signing queries, 1 signature to exploit.

Choose independent and uniform elements $(a_i)_{(1, \ldots, \ell)}$ in $\{-1, 0, 1\}$, and random exponents $(b_i)_{(0, \ldots, \ell)}$, and setting $a_0 = -1$.

Then $u_i = g^{a_i} h^{b_i}$.

$F(m) = u_0^m \prod u_i^{m_i} = g^{\sum_i a_i} h^{\sum_i b_i} = g^{a_m} h^{b_m}$.
Non $(2, 1)$-programmability

Waters over a non-binary alphabet is not $(2, 1)$-programmable.

$(1, q)$-programmability

Waters over a polynomial alphabet remains $(1, q)$-programmable.
Sum of random walks on polynomial alphabets

Local Central Limit Theorem ⇔ Lindeberg Feller
- New primitive: Signature on Randomizable Ciphertexts [PKC 2011: BFPV]
- ✓ One Round Blind Signature [PKC 2011: BFPV]
- ✓ Receipt Free E-Voting [PKC 2011: BFPV]
- ✓ Signer-Friendly Blind Signature [SCN 2012: BPV]
- ✓ Multi-Source Blind Signature [SCN 2012: BPV]

**Efficiency**

- DLin + CDH: $9\ell + 24$ Group elements.
- SXDH + CDH+: $6\ell + 15, 6\ell + 7$ Group elements.

Other results based on Groth-Sahai Methodology:

- Traceable Signatures [2012: BP]
- Transferable E-Cash [Africacrypt 2011: BCF+]
1. General Remarks

2. Building blocks

3. Non-Interactive Proofs of Knowledge

4. Interactive Implicit Proofs
   - Motivation
   - Smooth Projective Hash Function
   - Application to various protocols
   - Manageable Languages
Certification of a public key

Server

User

pk ←

→ \( \pi(sk) \) ←

→ Cert
Certification of Public Keys: (NI)ZKPoK

Certification of a public key

Server

pk ←
→ \pi(sk) ←
→ Cert

User
Certification of Public Keys: (NI)ZKPoK

Certification of a public key

Server

pk ←
π(sk)
→ Cert

User
Certification of Public Keys: (NI)ZKPoK

Certification of a public key

Server

pk ← \[\pi(sk)\] → Cert

User
Certification of Public Keys: (NI)ZKPoK

Certification of a public key

Server

\[ \pi(\text{sk}) \]

\[ \rightarrow \text{Cert} \]

User

\[ \text{pk} \leftarrow \]

\[ \pi \]

\[ \text{can be forwarded} \]
A user can ask for the certification of \( pk \), but if he knows the associated \( sk \) only:

**With a Smooth Projective Hash Function**

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

- \( U \) sends his \( \text{pk} \), and an encryption \( C \) of \( sk \);
- \( A \) generates the certificate \( \text{Cert} \) for \( \text{pk} \), and sends it, masked by \( \text{Hash} = \text{Hash}(hk; (pk, C)); \)
- \( U \) computes \( \text{Hash} = \text{ProjHash}(hp; (pk, C), r)) \), and gets \( \text{Cert} \).
A user can ask for the certification of $\text{pk}$, but if he knows the associated $\text{sk}$ only:

**With a Smooth Projective Hash Function**

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

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

Implicit proof of knowledge of $\text{sk}$
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 \( hk \): \( H(x) = \text{Hash}_L(hk; x) \);
- or a public projected key \( hp \): \( H'(x) = \text{ProjHash}_L(hp; x, w) \)

Public mapping \( hk \mapsto hp = \text{ProjKG}_L(hk, x) \)
SPHF 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 $x \in L$, $hp = \text{ProjKG}_L(hk, x)$

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 $X$. 

O. Blazy (ENS → RUB)
SPHF 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 $x \in L$, $hp = \text{ProjKG}_L(hk, x)$

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 $X$. 
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 $x \in L$, $hp = \text{ProjKG}_L(hk, x)$

**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 $X$. 

O. Blazy (ENS → RUB)
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 $x \in L$, $hp = \text{ProjKG}_L(hk, x)$

**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 $X$. 
Certification of a public key

Server

\[ \text{pk, } C = C(\text{sk}; r) \leftarrow \]

\[ \rightarrow P = \text{Cert} \oplus \text{Hash}(hk; (pk, C)) \]

\[ hp = \text{ProjKG}(hk, C) \]

User

\[ P \oplus \text{ProjHash}(hp; (pk, C), r) = \text{Cert} \]
Certification of Public Keys: SPHF

**Certification of a public key**

Server

\[
\text{pk, } C = C(sk; r) \leftarrow \\
\rightarrow P = \text{Cert} \oplus \text{Hash}(hk; (pk, C)) \\
hp = \text{ProjKG}(hk, C)
\]

User

\[
P \oplus \text{ProjHash}(hp; (pk, C), r) = \text{Cert}
\]

Implicit proof of knowledge of \(sk\)
A sender $S$ wants to send a message $P$ to $U$ such that
- $U$ gets $P$ iff it owns $\sigma(m)$ valid under $vk$
- $S$ does not learn whereas $U$ gets the message $P$ or not

Correctness: if $U$ owns a valid signature, he learns $P$

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 $P$ if he does not own a valid signature.
A sender $S$ wants to send a message $P$ to $U$ such that
- $U$ gets $P$ iff it owns $\sigma(m)$ valid under $vk$
- $S$ does not learn whereas $U$ gets the message $P$ or not

Correctness: if $U$ owns a valid signature, he learns $P$

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 $P$ if he does not own a valid signature.
A sender $S$ wants to send a message $P$ to $U$ such that
- $U$ gets $P$ iff it owns $\sigma(m)$ valid under $vk$
- $S$ does not learn whereas $U$ gets the message $P$ or not

Correctness: if $U$ owns a valid signature, he learns $P$

**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 $P$ 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 $P$ to $U$, if $U$ owns a valid signature.

- $S$ encrypts $P$ 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 $P$ to $U$, if $U$ owns a valid signature.

- $S$ encrypts $P$ 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 $P$ to $U$, if $U$ owns a valid signature.

- $S$ encrypts $P$ 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!
S wants to send a message $P$ 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;

- **Semantic Security:** $U$ cannot distinguish multiple interactions with:
  $S$ sending $P_0$ from those with $S$ sending $P_1$
  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 $P$. 

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

With a Smooth Projective Hash Function

Let $C = C(\sigma, r)$ contains a valid $\sigma(m)$ under $vk$

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

$$\text{Lin}(pk, m) : \{C(m)\} \rightsquigarrow \text{WLin}(pk, vk, m) : \{C(\sigma(m))\}$$

$$(U, V, W, G) \in \text{WLin}(pk, vk, m):
\exists r, s \in \mathbb{Z}_p, (U, V, W) = (u^r, v^s, g^{r+s}\sigma), e(\sigma, g) = e(h, vk) \cdot e(f(m), G)$$
Security Properties

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

Standard model with Waters Signature + Linear Encryption $\rightsquigarrow$ CDH and DLin
✓ Oblivious 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 \(\leadsto\) round-optimal
Standard model with Waters Signature + Linear Encryption \(\leadsto\) CDH and DLin
\[ \sigma(m) \xrightarrow{\text{Commit}} \sigma \]

\[ c_k, \sigma; r \]

\[ \text{hk} = \text{HashKG}_L \]
\[ H = \text{Hash}_L(\text{hk}, c) \]
\[ \text{hp} = \text{ProjKG}_L(\text{hk}, c) \]

\[ P \]

\[ Q = P \oplus H \]

\[ \text{ProjHash}_L(\text{hp}, C; w) = H' \]
\[ P' = Q \oplus H' \]

\[ L = \text{WLIn}(c_k, v_k, m) \xrightarrow{\sim} e(\chi, g) = e(\mathcal{F}(m), \sigma_2) \cdot e(v_k, h) \]
Blind-Signatures

\[ \text{Encrypt}_{SE}(pk, r) \]

\[ \text{Decrypt}_E \]

\[ \text{Sign}_{SE}(sk, pk, c, s) \]

\[ \text{SigExt}_{SE} \]

Groth Sahai

9 \( \ell + 24 \)
Blind-Signatures

\[ \text{Encrypt}_{SE}(pk, r) \]

\[ \text{Decrypt}_{\varepsilon} \]

\[ \text{Sign}_{SE}(sk, pk, c, s) \]

\[ \text{SigExt}_{SE} \]

\[ \text{Random}_{S} \]

\[ \sigma(\mathcal{F}) \]

\[ \mathcal{F}(M) \]

\[ \sigma(C) \]

\[ C \]

\[ \text{Groth Sahai} \]

\[ 9 \ell + 24 \]

\[ \text{SPHF} \]

\[ 8 \ell + 12 \]

\[ \text{Languages} \]

\[ \text{BLin}: \{0, 1\}, \]

\[ \text{ELin}: \{C(C(\ldots))\} \]
Password Authenticated Key Exchange

Alice

→ \text{C}(pw_B)

C(pw_A), hp_B ←

→ hp_A

Bob

H_B \cdot H'_A

H'_B \cdot H_A

Same value iff both passwords are the same, and users know witnesses.
Language Authenticated Key Exchange

Alice

$\rightarrow C(\mathcal{L}_B), C(\mathcal{L}_A'), C(M_B)$

$C(\mathcal{L}_A), C(\mathcal{L}_B'), C(M_A), h_p_B \leftarrow$

$\rightarrow h_p_A$

Bob

$H_B \cdot H_A'$

$H_B' \cdot H_A$

Same value iff languages are as expected, and users know witnesses.
**Diffie Hellman / Linear Tuple**

**Conjunction / Disjunction**

\[(g, h, G = g^a, H = h^a)\]

Valid Diffie Hellman tuple?

\[hp^a = G^\kappa H^\lambda\]

\[hp = g^\kappa h^\lambda\]

Oblivious Transfer, Implicit Opening of a ciphertext

\[(U = u^a, V = v^b, W = g^{a+b})\]

Valid Linear tuple?

\[hp_1^a hp_2^b = U^\kappa V^\mu W^\lambda\]

\[hp = u^\kappa g^\lambda, v^\mu g^\lambda\]
**Diffie Hellman / Linear Tuple**

- **Conjunction / Disjunction**

\[(g, h, G = g^a, H = h^a)\]

Valid Diffie Hellman tuple?

\[hp^a = G^\kappa H^\lambda\]

\[hp \equiv g^\kappa h^\lambda\]

Valid Linear tuple?

\[hp_1^a hp_2^b = U^\kappa V^\mu W^\lambda\]

Oblivious Transfer, Implicit Opening of a ciphertext

\[(U = u^a, V = v^b, W = g^{a+b})\]

\[hp = u^\kappa g^\lambda, v^\mu g^\lambda\]
Diffie Hellman / Linear Tuple
Conjunction / Disjunction

\[ L_1 \cap L_2 \]

\[ \text{hp} = \text{hp}_1, \text{hp}_2 \]

\[ \land A_i \]

Simultaneous verification

\[ H'_1 \cdot H'_2 = H_1 \cdot H_2 \]
Diffie Hellman / Linear Tuple
Conjunction / Disjunction

\( \mathcal{L}_1 \cap \mathcal{L}_2 \)
\( hp = hp_1, hp_2 \)
\( \land A_i \)

\( \mathcal{L}_1 \cup \mathcal{L}_2 \)
\( hp = hp_1, hp_2, hp_\Delta \)
Is it a bit?

Simultaneous verification

\[ H'_1 \cdot H'_2 = H_1 \cdot H_2 \]

One out of 2 conditions

\[ H' = \mathcal{L}_1 ? hp_{w_1}^1 : hp_{w_2} \cdot hp_\Delta = X_1^{hk_1} \]
Diffie Hellman / Linear Tuple
Conjunction / Disjunction

\[ \mathcal{L}_1 \cap \mathcal{L}_2 \]
\[ \text{hp} = \text{hp}_1, \text{hp}_2 \]
\[ \land A_i \]

\[ \mathcal{L}_1 \cup \mathcal{L}_2 \]
\[ \text{hp} = \text{hp}_1, \text{hp}_2, \text{hp}_\Delta \]
Is it a bit?

Simultaneous verification
\[ H'_1 \cdot H'_2 = H_1 \cdot H_2 \]

One out of 2 conditions
\[ H' = \mathcal{L}_1 ? \text{hp}_1^{w_1} : \text{hp}_2^{w_2} : \text{hp}_\Delta = \chi_1^{hk_1} \]
(Linear) Cramer-Shoup Encryption

Commitment of a commitment

Linear Pairing Equations

Quadratic Pairing Equation

\[(e = h^r M, u_1 = g_1^r, u_2 = g_2^r, \nu = (cd^\alpha)^r)\]
\[hp = g_1^\kappa g_2^\mu (cd^\alpha)^\eta h^\lambda\]

Verifiability of the CS
\[hp^r = u_1^\kappa u_2^\mu \nu^\eta (e/M)^\lambda\]

Implicit Opening of a ciphertext, verifiability of a ciphertext, PAKE

\[(g_1^r, g_2^s, g_3^{r+s}, h_1^r h_2^s M, (c_1 d_1^\alpha)^r)(c_2 d_2^\alpha)^s)\]
\[hp = g_1^\kappa g_3^\theta (c_1 d_1^\alpha)^\eta h^\lambda, g_2^\mu g_3^\theta (c_1 d_1^\alpha)^\eta h^\lambda\]

Verifiability of the LCS
\[hp_1^r \cdot hp_2^s = u_1^\kappa u_2^\mu u_3^\theta \nu^\eta (e/M)^\lambda\]
(Linear) Cramer-Shoup Encryption

Commitment of a commitment

Linear Pairing Equations

Quadratic Pairing Equation

\( e = h^r M, u_1 = g_1^r, u_2 = g_2^r, \nu = (cd^\alpha)^r \)

\[ \text{Verifiability of the CS} \]

\[ hp^r = u_1^\kappa u_2^\mu \nu^\eta (e/M)^\lambda \]

Implicit Opening of a ciphertext, verifiability of a ciphertext, PAKE

\( (g_1^r, g_2^s, g_3^{r+s}, h_1^r h_2^s M, (c_1 d_1^\alpha)^r) (c_2 d_2^\alpha)^s \)

\[ \text{Verifiability of the LCS} \]

\[ hp_1^r \cdot hp_2^s = u_1^\kappa u_2^\mu u_3^\theta \nu^\eta (e/M)^\lambda \]
(Linear) Cramer-Shoup Encryption
Commitment of a commitment
Linear Pairing Equations
Quadratic Pairing Equation

\[(U = u^a, V = v^s, G = h^s g^a)\]
\[hp = u^\eta g^\lambda, v^\theta h^\lambda\]

Previous Language ELin
\[hp_1^a hp_2^s = U^\eta V^\theta G^\lambda\]
(Linear) Cramer-Shoup Encryption
Commitment of a commitment
Linear Pairing Equations
Quadratic Pairing Equation

\[
\left( \prod_{i \in A_k} e(Y_i, A_{k,i}) \right) \cdot \left( \prod_{i \in B_k} Z_{i,3k,i} \right) = D_k
\]

For each variables: \( h_{pi} = u^{\kappa_i} g^{\lambda}, v^{\mu_i} g^{\lambda} \)

\[
\left( \prod_{i \in A_k} e(h_{pi}^{w_i}, A_{k,i}) \right) \cdot \left( \prod_{i \in B_k} H_{pi}^{3k,i,w_i} \right) = \\
\left( \prod_{i \in A_k} e(H_i, A_{k,i}) \right) \cdot \left( \prod_{i \in B_k} H_i^{3k,i} \right) / D_k^\lambda
\]

Knowledge of a secret key, Knowledge of a (secret) signature on a (secret) message valid under a (secret) verification key, ...
(Linear) Cramer-Shoup Encryption
Commitment of a commitment
Linear Pairing Equations
Quadratic Pairing Equation

\[
\left( \prod_{i \leq j \in A_k} e(Y_i, A_k, i) \cdot e(Y_i, Y_j)^{\gamma_{i,j}} \right) \cdot \left( \prod_{i \in B_k} Z_i^{3k,i} \right) = D_k
\]

Anonymous membership to a group, other way to do BLin,…
\[ e(g^b, g^{1-b}) = 1_T \]
Smooth Projective Hash Functions \(\Rightarrow\) implicit proofs of knowledge
Smooth Projective Hash Functions $\Rightarrow$ implicit proofs of knowledge

Various Applications:

Privacy-preserving protocols:

△ Many more Round optimal applications?
Smooth Projective Hash Functions $\Rightarrow$ implicit proofs of knowledge

Various Applications:

- IND-CCA [CS02]

Privacy-preserving protocols:

△ Many more Round optimal applications?
Smooth Projective Hash Functions \( \Rightarrow \) implicit proofs of knowledge

Various Applications:

- IND-CCA [CS02]
- PAKE [GL03]

Privacy-preserving protocols:

△ Many more Round optimal applications?
Smooth Projective Hash Functions ⇒ implicit proofs of knowledge

Various Applications:

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

Privacy-preserving protocols:

△ Many more Round optimal applications?
Smooth Projective Hash Functions $\Rightarrow$ implicit proofs of knowledge

Various Applications:

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

Privacy-preserving protocols:

⚠️ Many more Round optimal applications?
Smooth Projective Hash Functions $\Rightarrow$ implicit proofs of knowledge

Various Applications:
- IND-CCA [CS02]
- PAKE [GL03]
- Certification of Public Keys [ACP09]

Privacy-preserving protocols:
- Blind signatures [TCC 2012: BPV]

△ Many more Round optimal applications?
Smooth Projective Hash Functions \( \Rightarrow \) implicit proofs of knowledge

Various Applications:

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

Privacy-preserving protocols:

- ✓ Blind signatures [TCC 2012: BPV]
- ✓ Oblivious Signature-Based Envelope [TCC 2012: BPV]

△ Many more Round optimal applications?
Smooth Projective Hash Functions $\Rightarrow$ implicit proofs of knowledge

Various Applications:

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

Privacy-preserving protocols:

- ✓ Blind signatures [TCC 2012: BPV]
- ✓ Oblivious Signature-Based Envelope [TCC 2012: BPV]
- ✓ (v)-PAKE, LAKE, Secret Handshakes [eprint/sub 2012: BPCV]

△ Many more Round optimal applications?
Smooth Projective Hash Functions \(\Rightarrow\) implicit proofs of knowledge

Various Applications:
- \(\checkmark\) IND-CCA [CS02]
- \(\checkmark\) PAKE [GL03]
- \(\checkmark\) Certification of Public Keys [ACP09]

Privacy-preserving protocols:
- \(\checkmark\) Blind signatures [TCC 2012: BPV]
- \(\checkmark\) Oblivious Signature-Based Envelope [TCC 2012: BPV]
- \(\checkmark\) (v)-PAKE, LAKE, Secret Handshakes [eprint/sub 2012: BPCV]
- \(\checkmark\) E-Voting [sub 2012: BP]

△ Many more Round optimal applications?
Smooth Projective Hash Functions $\triangle$ implicit proofs of knowledge

**Various Applications:**
- ✓ IND-CCA [CS02]
- ✓ PAKE [GL03]
- ✓ Certification of Public Keys [ACP09]

**Privacy-preserving protocols:**
- ✓ Blind signatures [TCC 2012: BPV]
- ✓ Oblivious Signature-Based Envelope [TCC 2012: BPV]
- ✓ (v)-PAKE, LAKE, Secret Handshakes [eprint/sub 2012: BPCV]
- ✓ E-Voting [sub 2012: BP]

△ Many more Round optimal applications?
Groth-Sahai
- Allows to combine efficiently classical building blocks
- Allows several kind of new signatures under standard hypotheses

Smooth Projective Hash Functions
- Can handle more general languages under better hypotheses
- Do not add any extra-rounds in an interactive scenario
- More efficient in the usual cases
**Groth-Sahai**
- Allows to combine efficiently classical building blocks
- Allows several kind of new signatures under standard hypotheses

**Smooth Projective Hash Functions**
- Can handle more general languages under better hypotheses
- Do not add any extra-rounds in an interactive scenario
- More efficient in the usual cases