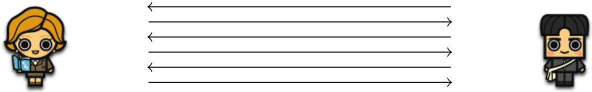
Interactive and Non-Interactive Proofs of Knowledge

Olivier Blazy

ENS / CNRS / INRIA / Paris 7 $\rightarrow$ RUB

Sept 2012

Proof of Knowledge



Alice Bob

- interactive method for one party to prove to another the knowledge of a secret $\mathcal{S}$.

1. **Completeness:** $\mathcal{S}$ is true $\leadsto$ verifier will be convinced of this fact
2. **Soundness:** $\mathcal{S}$ is false $\leadsto$ no cheating prover can convince the verifier that $\mathcal{S}$ is true

Classical Instantiations : Schnorr proofs, Sigma Protocols ...

Zero-Knowledge Proof Systems

- Introduced in 1985 by Goldwasser, Micali and Rackoff.

$\rightsquigarrow$ 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.

  $\rightsquigarrow$ Reveal nothing other than the validity of assertion being proven

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

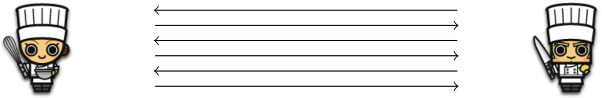
Zero-Knowledge Proof Systems

- Introduced in 1985 by Goldwasser, Micali and Rackoff.

  $\rightsquigarrow$ 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

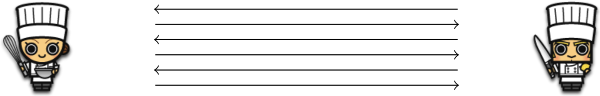


Alice Bob

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

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

Zero-Knowledge Interactive Proof

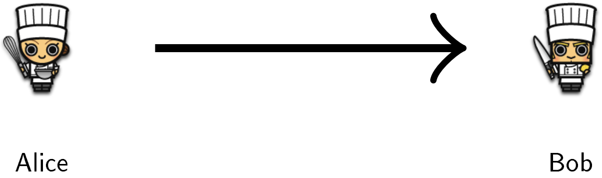


Alice Bob

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

1. **Completeness:** if $\mathcal{S}$ is true, the honest verifier will be convinced of this fact
2. **Soundness:** if $\mathcal{S}$ is false, no cheating prover can convince the honest verifier that it is true
3. **Zero-knowledge:** if $\mathcal{S}$ 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 $\mathcal{S}$ is true, without revealing anything other than the veracity of $\mathcal{S}$.

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

History of NIZK Proofs

Inefficient NIZK

- Blum-Feldman-Micali, 1988.
- . . .
- De Santis-Di Crescenzo-Persiano, 2002.

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

Efficient NIZK

- Groth-Ostrovsky-Sahai, 2006.
- Groth-Sahai, 2008.

History of NIZK Proofs

Inefficient NIZK

- Blum-Feldman-Micali, 1988.
- ...
- De Santis-Di Crescenzo-Persiano, 2002.

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

Efficient NIZK

- Groth-Ostrovsky-Sahai, 2006.
- Groth-Sahai, 2008.

History of NIZK Proofs

Inefficient NIZK

- Blum-Feldman-Micali, 1988.
- ...
- De Santis-Di Crescenzo-Persiano, 2002.

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

Efficient NIZK

- Groth-Ostrovsky-Sahai, 2006.
- Groth-Sahai, 2008.

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

$\mathsf{pk} \leftarrow$

$\rightarrow \mathsf{Cert}$

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

Signature of a blinded message

Signer

User

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

$\rightarrow \sigma$

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

Conditional Actions

Transmission of private information

Server

Request $\leftarrow$

$\rightarrow$ info

User

$\pi \rightsquigarrow$ The User should possess some credentials.

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.

Symmetric bilinear structure

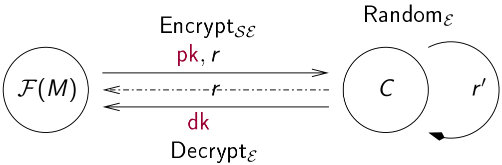
$(p, \mathbb{G}, \mathbb{G}_T, e, g)$ bilinear structure:

- $\mathbb{G}$, $\mathbb{G}_T$ multiplicative groups of order p
  - $p =$ **prime integer**
- $\langle g \rangle = \mathbb{G}$
- $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$
  - $\langle e(g,g) \rangle = \mathbb{G}_T$
  - $e(g^a, g^b) = e(g,g)^{ab}$, $a, b \in \mathbb{Z}$
- deciding group membership, group operations, bilinear map } efficiently computable.

Definition (Encryption Scheme)

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

- Setup($1^{\mathfrak{K}}$): 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.



Indistinguishability:
Given M_0, M_1, it should be hard to guess which one is encrypted in C.

Definition (Linear Encryption) (BBS04)

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

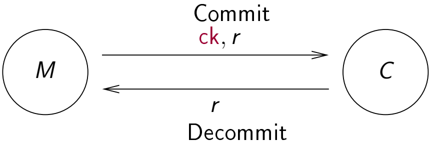
Randomization

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

Definition (Commitment Scheme)

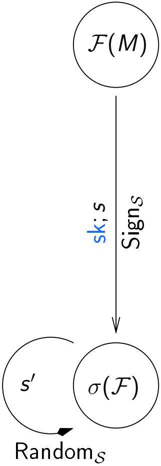
$\mathcal{E} = (\mathsf{Setup}, \mathsf{Commit}, \mathsf{Decommit})$:

- $\mathsf{Setup}(1^{\mathfrak{K}})$: param, ck;
- $\mathsf{Commit}(\mathsf{ck}, m; r)$: **c** on the input message $m \in \mathcal{M}$ using $r \stackrel{\$}{\leftarrow} \mathcal{R}$;
- $\mathsf{Decommit}(\mathbf{c}, m; w)$ opens **c** and reveals m, together with w that proves the correct opening.



Pedersen

- Setup$(1^{\mathfrak{K}})$: $g, h \in \mathbb{G}$;
- Commit$(m; r)$: $\mathbf{c} = g^m h^r$;
- Decommit$(\mathbf{c}, m; r)$: $\mathbf{c} \stackrel{?}{=} g^m h^r$.



Definition (Signature Scheme)

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

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

Unforgeability:
Given q pairs (m_i, σ_i), it should be hard to output a valid σ on a fresh m.

Definition (Waters Signature) (Wat05)

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

Randomization

$\mathsf{Random}(\sigma; r) : \sigma' = \left(\sigma_1 \mathcal{F}(m)^r, \sigma_2 g^r\right) = \left(\mathsf{sk}\mathcal{F}(m)^{r+s}, g^{r+s}\right)$

Definition (DL)

Given $g, h \in \mathbb{G}^2$, it is hard to compute α 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 $c = a + b$.

Groth-Sahai Proof System

- **Pairing product equation (PPE):** for variables $\mathcal{X}_1, \ldots, \mathcal{X}_n \in \mathbb{G}$

$$(E) : \prod_{i=1}^{n} e(A_i, \mathcal{X}_i) \prod_{i=1}^{n} \prod_{j=1}^{n} e(\mathcal{X}_i, \mathcal{X}_j)^{\gamma_{i,j}} = t_T$$

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

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

Setup($\mathbb{G}$): commitment key **ck**;
Com(**ck**, $X \in \mathbb{G}$; ρ): commitment $\vec{c_X}$ to X;
Prove(**ck**, $(X_i, \rho_i)_{i=1,\ldots,n}$, (E)): proof ϕ;
Verify(**ck**, $\vec{c_{X_i}}$, (E), ϕ): checks whether ϕ is valid.

Groth-Sahai Proof System

- **Pairing product equation (PPE):** for variables $\mathcal{X}_1, \ldots, \mathcal{X}_n \in \mathbb{G}$

$$(E) : \prod_{i=1}^{n} e(A_i, \mathcal{X}_i) \prod_{i=1}^{n} \prod_{j=1}^{n} e(\mathcal{X}_i, \mathcal{X}_j)^{\gamma_{i,j}} = t_T$$

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

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

Setup($\mathbb{G}$): commitment key **ck**;
Com(**ck**, $X \in \mathbb{G}$; ρ): commitment $\vec{c_X}$ to X;
Prove(**ck**, $(X_i, \rho_i)_{i=1,\ldots,n}$, (E)): proof ϕ;
Verify(**ck**, $\vec{c_{X_i}}$, (E), ϕ): checks whether ϕ is valid.

$$(E) : \prod_{i=1}^{n} e(A_i, \mathcal{X}_i) \prod_{i=1}^{n} \prod_{j=1}^{n} e(\mathcal{X}_i, \mathcal{X}_j)^{\gamma_{i,j}} = t_T$$

Assumption	DLin	SXDH	*SD*
Variables	3	2	1
PPE	9	(2,2)	1
Linear	3	2	1
Verification	$12n + 27$	$5m + 3n + 16$	$n + 1$
[ACNS 2010: BFI+]	$3n + 6$	$m + 2n + 8$	$n + 1$

Properties:

- correctness
- soundness
- witness-indistinguishability
- randomizability Commitments and proofs are publicly randomizable.

$$(E) : \prod_{i=1}^{n} e(A_i, \mathcal{X}_i) \prod_{i=1}^{n} \prod_{j=1}^{n} e(\mathcal{X}_i, \mathcal{X}_j)^{\gamma_{i,j}} = t_T$$

Assumption	DLin	SXDH	*SD*
Variables	3	2	1
PPE	9	(2,2)	1
Linear	3	2	1
Verification	$12n + 27$	$5m + 3n + 16$	$n + 1$
[ACNS 2010: BFI+]	$3n + 6$	$m + 2n + 8$	$n + 1$

Properties:

- correctness
- soundness
- witness-indistinguishability
- randomizability Commitments and proofs are publicly randomizable.

$$(E): \prod_{i=1}^{n} e(A_i, \mathcal{X}_i) \prod_{i=1}^{n} \prod_{j=1}^{n} e(\mathcal{X}_i, \mathcal{X}_j)^{\gamma_{i,j}} = t_T$$

Assumption	DLin	SXDH	*SD*
Variables	3	2	1
PPE	9	(2,2)	1
Linear	3	2	1
Verification	$12n + 27$	$5m + 3n + 16$	$n + 1$
[ACNS 2010: BFI+]	$3n + 6$	$m + 2n + 8$	$n + 1$

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

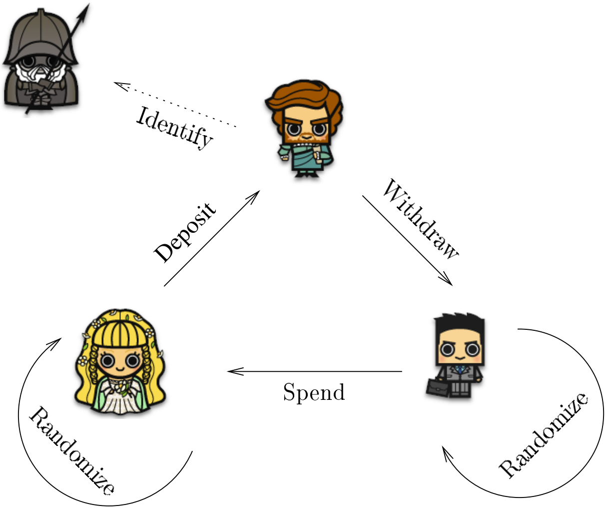
- Votes and voters should be anonymous

△ Receipt freeness

Homomorphic Encryption and Signature approach

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

Electronic Cash



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

Electronic Coins — Chaum 81

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 σ so that the authority *even powerful* cannot recognize later the pair (m, σ).

Round-Optimal Blind Signature — Fischlin 06

- The user encrypts his message m in c.
- The signer then signs c in σ.
- The user verifies σ.
- He then encrypts σ and c into $\mathcal{C}_\sigma$ and $\mathcal{C}$ and generates a proof π.
- π: $\mathcal{C}_\sigma$ is an encryption of a signature over the ciphertext c encrypted in $\mathcal{C}$, and this c is indeed an encryption of m.
- Anyone can then use $\mathcal{C}, \mathcal{C}_\sigma, \pi$ 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;
- $\leadsto$ Revisited Waters, Commutative encryption / signature.

Commutative properties

Encrypt

To encrypt a message m:

$$c = (\mathsf{pk}_1{}^{r_1}, \mathsf{pk}_2{}^{r_2}, \mathcal{F}(m) \cdot g^{r_1+r_2})$$

Commutative properties

Encrypt

To encrypt a message m:

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

Sign ∘ Encrypt

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

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

Commutative properties

Encrypt

To encrypt a message m:

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

Sign ∘ Encrypt

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

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

Decrypt ∘ Sign ∘ Encrypt

Using dk.

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

Definition (Signature on Ciphertexts)

$\mathcal{SE}$ = (Setup, SKeyGen, EKeyGen, Encrypt, Sign, Decrypt, Verif):

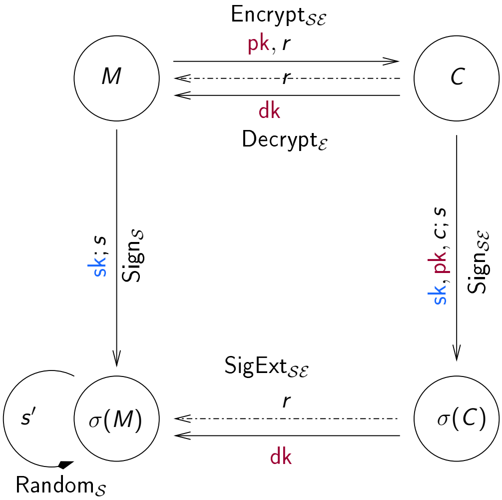
- Setup($1^{\mathfrak{K}}$): param_e, param_s;
- EKeyGen(param_e): pk, dk;
- SKeyGen(param_s): vk, sk;
- Encrypt(pk, vk, m; r): produces c on $m \in \mathcal{M}$ and pk;
- Sign(sk, pk, c; s): produces σ, on the input c under sk;
- Decrypt(dk, vk, c): decrypts c under dk;
- Verif(vk, pk, c, σ): checks whether σ is valid.

Definition (Extractable Randomizable Signature on Ciphertexts)

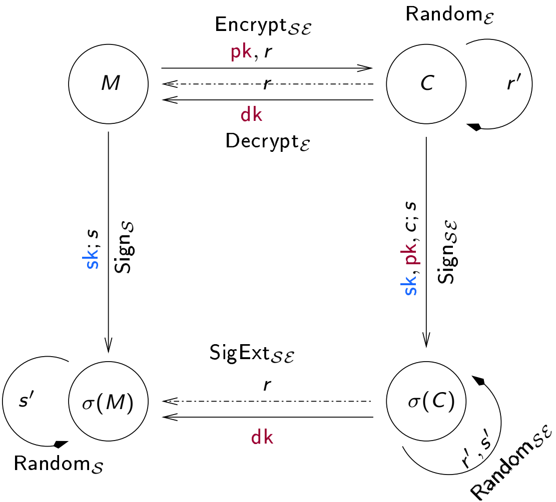
$\mathcal{SE}$=(Setup, SKeyGen, EKeyGen, Encrypt, Sign, Random, Decrypt, Verif, SigExt):

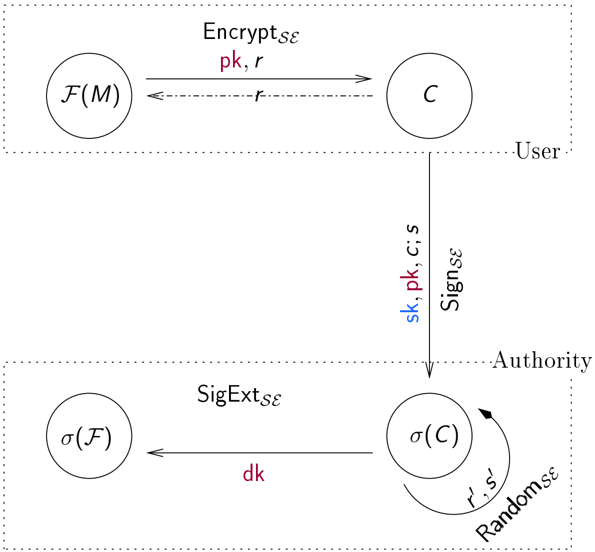
- Random(vk, pk, c, σ; r', s') produces c' and σ' on c', using additional coins;
- SigExt(dk, vk, σ) outputs a signature σ^*.

Randomizable Signature on Ciphertexts [PKC 2011: BFPV]

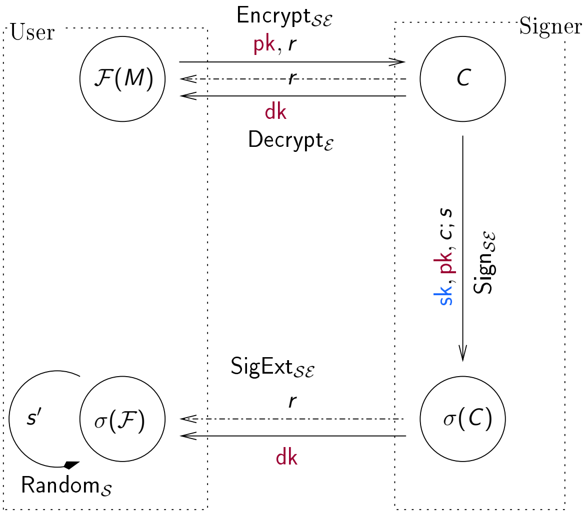


Extractable SRC




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

Blind Signature [PKC 2011: BFPV]



Partially-Blind Signature

User

Signer

info

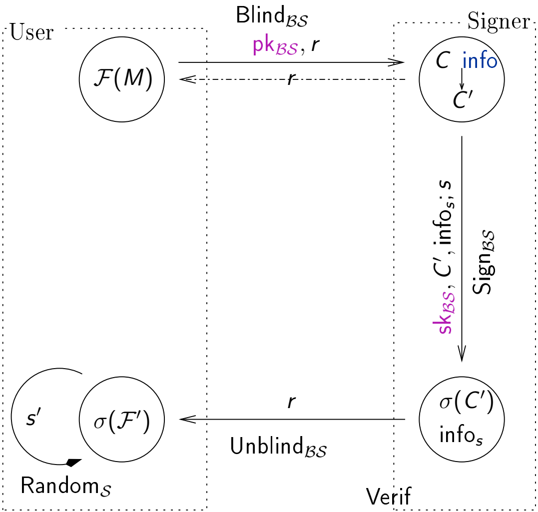
$C' = C(M, \text{info})$ $\longrightarrow$

$\sigma(C')$ $\longleftarrow$

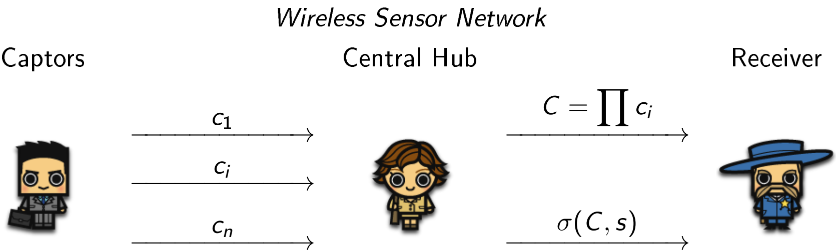
Partially-Blind Signature

User		Signer
	$\xrightarrow{C' = C(M, \text{info})}$ $\xleftarrow{\sigma(C', \text{info}_s)}$	

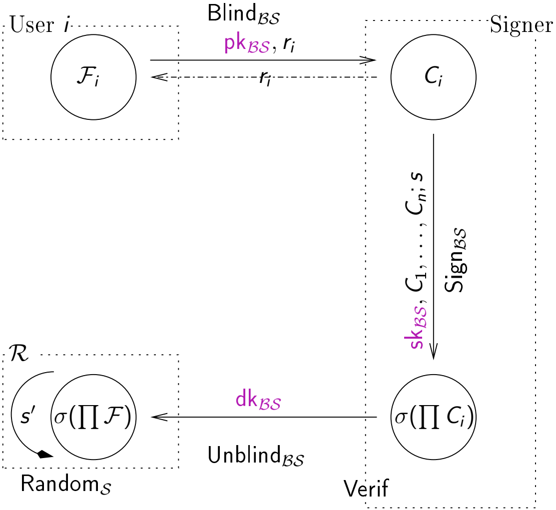
Signer-Friendly Partially Blind Signature [SCN 2012: BPV]



Multi-Source Blind Signatures



Multi-Source Blind Signatures [SCN 2012: BPV]



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

$(1, q)$-Programmability of Waters function

Why do we need it: Unforgeabilty, q signing queries, 1 signature to exploit.
$\rightsquigarrow$ 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 $u_i = g^{a_i} h^{b_i}$.

$\mathcal{F}(m) = u_0 \prod u_i^{m_i} = g^{\sum_{\delta_i} a_i} h^{\sum_{\delta_i} b_i} = g^{a_m} h^{b_m}$.

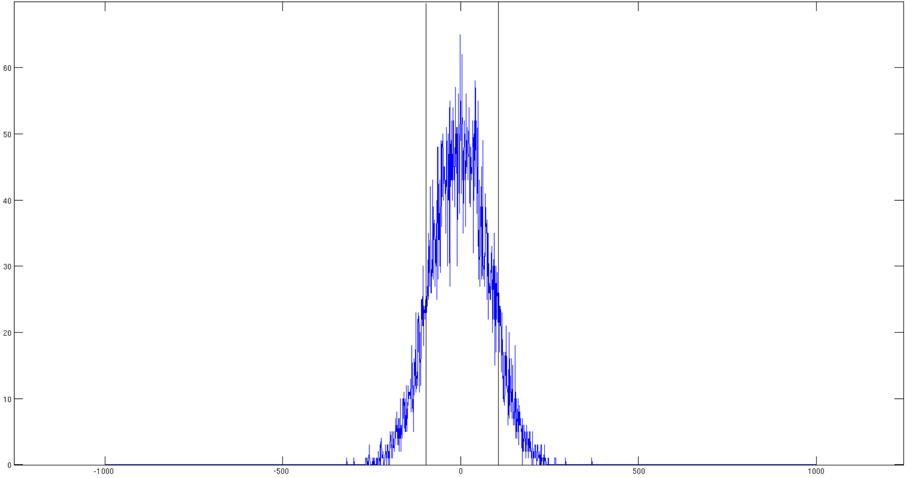
Non $(2,1)$-programmability

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+]

Certification of Public Keys: (NI)ZKPoK

Certification of a public key

Server

$\mathsf{pk} \leftarrow$

$\rightarrow \pi(\mathsf{sk}) \leftarrow$

$\rightarrow \mathsf{Cert}$

User

Certification of Public Keys: (NI)ZKPoK

Certification of a public key

Server

$\mathsf{pk} \leftarrow$

$\rightarrow \pi(\mathsf{sk}) \leftarrow$

$\rightarrow \mathsf{Cert}$

User

Certification of Public Keys: (NI)ZKPoK

Certification of a public key

Server		User
	pk $\leftarrow$ π(sk) $\rightarrow$ Cert	

Certification of Public Keys: (NI)ZKPoK

Certification of a public key

Server

$\mathsf{pk} \leftarrow$

$\pi(\mathsf{sk})$

$\rightarrow \mathsf{Cert}$

User

Certification of Public Keys: (NI)ZKPoK

Certification of a public key

Server		User
	pk ← π(sk) → Cert	

π can be forwarded

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

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

SPHF Properties

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

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.

SPHF Properties

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

w witness that $x \in L, \text{hp} = \text{ProjKG}_L(\text{hk}, x)$

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.

SPHF Properties

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

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.

SPHF Properties

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

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 Public Keys: SPHF [ACP09]

Certification of a public key

Server		User
	$\mathsf{pk}, C = \mathcal{C}(\mathsf{sk}; r) \leftarrow$ $\rightarrow P = \mathsf{Cert} \oplus \mathsf{Hash}(\mathsf{hk}; (\mathsf{pk}, C))$ $\mathsf{hp} = \mathsf{ProjKG}(\mathsf{hk}, C)$	

$$P \oplus \mathsf{ProjHash}(\mathsf{hp}; (\mathsf{pk}, C), r) = \mathsf{Cert}$$

Certification of Public Keys: SPHF [ACP09]

Certification of a public key

Server | User

$$\mathsf{pk}, C = \mathcal{C}(\mathsf{sk}; r) \leftarrow$$

$$\rightarrow P = \mathsf{Cert} \oplus \mathsf{Hash}(\mathsf{hk}; (\mathsf{pk}, C))$$

$$\mathsf{hp} = \mathsf{ProjKG}(\mathsf{hk}, C)$$

$$P \oplus \mathsf{ProjHash}(\mathsf{hp}; (\mathsf{pk}, C), r) = \mathsf{Cert}$$

Implicit proof of knowledge of sk

Oblivious Signature-Based Envelope (OSBE) [LDB03]

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.

Oblivious Signature-Based Envelope (OSBE) [LDB03]

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.

Oblivious Signature-Based Envelope (OSBE) [LDB03]

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!

A Stronger Security Model

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

$\mathcal{L}$: $C = \mathcal{C}(\sigma, r)$ 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.

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

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

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

Security Properties

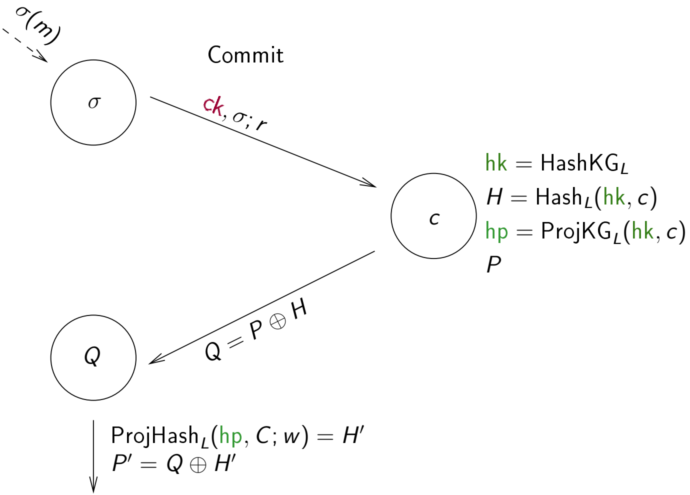
- ✓ Oblivious w.r.t. the authority: IND-CPA of the encryption scheme (Hard-partitioned Subset of the SPHF);
- ✓ Semantic Security: Smoothness of the SPHF
- ✓ Semantic Security w.r.t. the Authority: Pseudo-randomness of the SPHF

Semantic Security w.r.t. the Authority requires one interaction $\rightsquigarrow$ round-optimal
Standard model with Waters Signature + Linear Encryption $\rightsquigarrow$ CDH and DLin

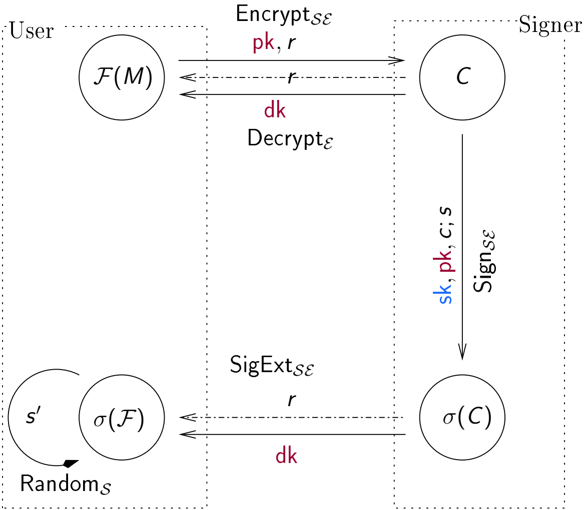
Security Properties

- ✓ 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

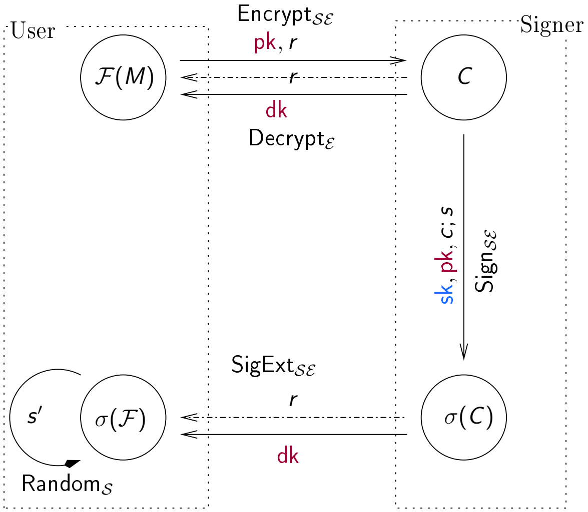


$$L = \mathsf{WLin}(\mathsf{ck}, \mathsf{vk}, m) \rightsquigarrow e(\underline{\mathcal{X}}, g) = e(\mathcal{F}(m), \sigma_2) \cdot e(\mathsf{vk}, h)$$



Groth Sahai

$9\,\ell + 24$



Groth Sahai

$9\,\ell + 24$

SPHF

$8\,\ell + 12$

Languages

BLin: $\{0, 1\}$,
ELin: $\{\mathcal{C}(\mathcal{C}(\ldots))\}$.

Password Authenticated Key Exchange

Alice

$\rightarrow \mathcal{C}(pw_B)$

$\mathcal{C}(pw_A), \mathrm{hp}_B \leftarrow$

$\rightarrow \mathrm{hp}_A$

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

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

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

$$\rightarrow \mathsf{hp}_A$$

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?

$\mathrm{hp} = g^\kappa h^\lambda$ $\mathrm{hp}^a = G^\kappa H^\lambda$

Oblivious Transfer, Implicit Opening of a ciphertext

$(U = u^a, V = v^b, W = g^{a+b})$ Valid Linear tuple?

$\mathrm{hp} = u^\kappa g^\lambda, v^\mu g^\lambda$ $\mathrm{hp}_1^a \mathrm{hp}_2^b = U^\kappa V^\mu W^\lambda$

- Diffie Hellman / Linear Tuple
- Conjunction / Disjunction

$(g, h, G = g^a, H = h^a)$ Valid Diffie Hellman tuple?

$\mathsf{hp} = g^\kappa h^\lambda$ $\mathsf{hp}^a = G^\kappa H^\lambda$

Oblivious Transfer, Implicit Opening of a ciphertext

$(U = u^a, V = v^b, W = g^{a+b})$ Valid Linear tuple?

$\mathsf{hp} = u^\kappa g^\lambda, v^\mu g^\lambda$ $\mathsf{hp}_1^a \mathsf{hp}_2^b = U^\kappa V^\mu W^\lambda$

- Diffie Hellman / Linear Tuple
- Conjunction / Disjunction

$\mathcal{L}_1 \cap \mathcal{L}_2$

Simultaneous verification

$\text{hp} = \text{hp}_1, \text{hp}_2$

$H'_1 \cdot H'_2 = H_1 \cdot H_2$

$\wedge A_i$

- Diffie Hellman / Linear Tuple
- Conjunction / Disjunction

$\mathcal{L}_1 \cap \mathcal{L}_2$
$\mathrm{hp} = \mathrm{hp}_1, \mathrm{hp}_2$
$\wedge A_i$

Simultaneous verification
$H'_1 \cdot H'_2 = H_1 \cdot H_2$

$\mathcal{L}_1 \cup \mathcal{L}_2$
$\mathrm{hp} = \mathrm{hp}_1, \mathrm{hp}_2, \mathrm{hp}_\Delta$
Is it a bit?

One out of 2 conditions
$H' = \mathcal{L}_1 ? \mathrm{hp}_1^{w_1} : \mathrm{hp}_2^{w_2} \cdot \mathrm{hp}_\Delta = X_1^{\mathrm{hk}_1}$

- Diffie Hellman / Linear Tuple
- Conjunction / Disjunction

$\mathcal{L}_1 \cap \mathcal{L}_2$ — Simultaneous verification

$\mathsf{hp} = \mathsf{hp}_1, \mathsf{hp}_2$ — $H'_1 \cdot H'_2 = H_1 \cdot H_2$

$\wedge A_i$

$\mathcal{L}_1 \cup \mathcal{L}_2$ — One out of 2 conditions

$\mathsf{hp} = \mathsf{hp}_1, \mathsf{hp}_2, \mathsf{hp}_\Delta$ — $H' = \mathcal{L}_1 ? \mathsf{hp}_1^{w_1} : \mathsf{hp}_2^{w_2} \cdot \mathsf{hp}_\Delta = X_1^{\mathsf{hk}_1}$

Is it a bit?

$\rightsquigarrow$ BLin.

- (Linear) Cramer-Shoup Encryption
- Commitment of a commitment
- Linear Pairing Equations
- Quadratic Pairing Equation

$(e = h^r M, u_1 = g_1^r, u_2 = g_2^r, v = (cd^\alpha)^r)$
$\mathrm{hp} = g_1^\kappa g_2^\mu (cd^\alpha)^\eta h^\lambda$

Verifiability of the CS
$\mathrm{hp}^r = u_1^\kappa u_2^\mu v^\eta (e/M)^\lambda$

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

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

Verifiability of the LCS
$\mathrm{hp}_1^r \cdot \mathrm{hp}_2^s = u_1^\kappa u_2^\mu u_3^\theta v^\eta (e/M)^\lambda$

- (Linear) Cramer-Shoup Encryption
- Commitment of a commitment
- Linear Pairing Equations
- Quadratic Pairing Equation

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

$\text{hp} = g_1^\kappa g_2^\mu (cd^\alpha)^\eta h^\lambda$

Verifiability of the CS

$\text{hp}^r = u_1^\kappa u_2^\mu v^\eta (e/M)^\lambda$

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

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

$\text{hp} = g_1^\kappa g_3^\theta (c_1 d_1^\alpha)^\eta h^\lambda, g_2^\mu g_3^\theta (c_1 d_1^\alpha)^\eta h^\lambda$

Verifiability of the LCS

$\text{hp}_1^r \cdot \text{hp}_2^s = u_1^\kappa u_2^\mu u_3^\theta v^\eta (e/M)^\lambda$

- (Linear) Cramer-Shoup Encryption
- Commitment of a commitment
- Linear Pairing Equations
- Quadratic Pairing Equation

$(U = u^a, V = v^s, G = h^s g^a)$
$\mathsf{hp} = u^\eta g^\lambda, v^\theta h^\lambda$

Previous Language ELin
$\mathsf{hp}_1^a \mathsf{hp}_2^s = U^\eta V^\theta G^\lambda$

- (Linear) Cramer-Shoup Encryption
- Commitment of a commitment
- Linear Pairing Equations
- Quadratic Pairing Equation

$$\left(\prod_{i \in A_k} e(\mathcal{Y}_i, \mathcal{A}_{k,i})\right) \cdot \left(\prod_{i \in B_k} \mathcal{Z}_i^{\mathfrak{z}_{k,i}}\right) = \mathcal{D}_k$$

For each variables: $\mathsf{hp}_i = u^{\kappa_i} g^{\lambda}, v^{\mu_i} g^{\lambda}$
$\left(\prod_{i \in A_k} e(\mathsf{hp}_i^{w_i}, \mathcal{A}_{k,i})\right) \cdot \left(\prod_{i \in B_k} \mathsf{HP}_i^{\mathfrak{z}_{k,i} w_i}\right) =$
$\left(\prod_{i \in A_k} e(H_i, \mathcal{A}_{k,i})\right) \cdot \left(\prod_{i \in B_k} H_i^{\mathfrak{z}_{k,i}}\right) / \mathcal{D}_k^{\lambda}$

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

$$\left(\prod_{i \leq j \in A_k} e(\mathcal{Y}_i, \mathcal{A}_{k,i}) \cdot e(\mathcal{Y}_i, \mathcal{Y}_j)^{\gamma_{i,j}}\right) \cdot \left(\prod_{i \in B_k} \mathcal{Z}_i^{\mathfrak{z}_{k,i}}\right) = \mathcal{D}_k$$

Anonymous membership to a group, other way to do BLin,...
$e(g^b, g^{1-b}) = 1_T$

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

Various Applications:

Privacy-preserving protocols:

△ Many more Round optimal applications?

Various Applications:

✓ IND-CCA [CS02]

Privacy-preserving protocols:

△ Many more Round optimal applications?

Smooth Projective Hash Functions $\triangleq$ *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?

Various Applications:

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

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:

- ✓ Blind signatures [TCC 2012: BPV]

△ Many more Round optimal applications?

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

△ 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:

- ✓ 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 ≙ *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?

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 [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