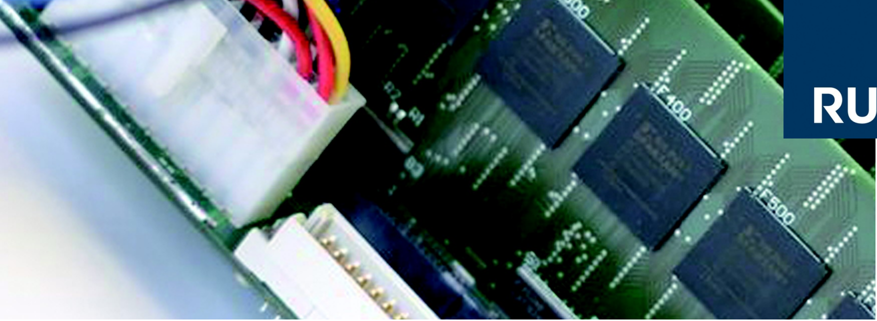


RUHR-UNIVERSITÄT BOCHUM

Round-optimal Signature, developing new tools to improve efficiency

, 2013

O. Blazy
Horst Görtz Institute for IT Security / Ruhr-University Bochum

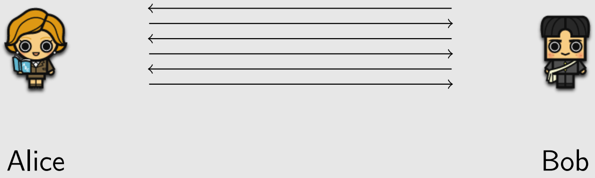
Horst Görtz Institut für IT-Sicherheit

RUB

1 General Remarks

hgi
Horst Görtz Institut
für IT-Sicherheit

Proof of Knowledge



- 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.

Zero-Knowledge Proof Systems

- Introduced in 1985 by Goldwasser, Micali and Rackoff.

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

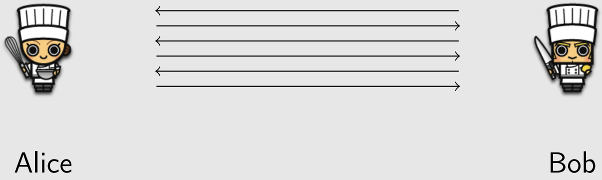
Zero-Knowledge Proof Systems

- 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



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

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

Alice Bob

§ 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 $\rightsquigarrow$ verifier will be convinced of this fact
2. **Soundness:** $\mathcal{S}$ is false $\rightsquigarrow$ no cheating prover can convince the verifier that $\mathcal{S}$ is true
3. **Zero-knowledge:** $\mathcal{S}$ is true $\rightsquigarrow$ 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.

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

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
- ...

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

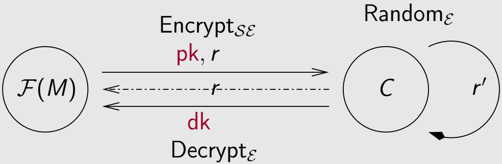
$(p, \mathbb{G}, \mathbb{G}_T, e, g)$ bilinear structure:

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

Definition 1 (Encryption Scheme)

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

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



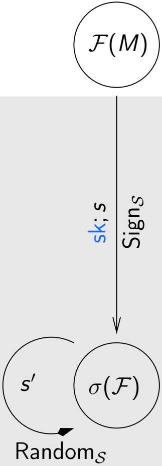
Indistinguishability:
Given M_0, M_1, it should be hard to guess which one is encrypted in C.

Definition 2 (Linear Encryption) (BBS04)

- Setup($1^{\mathfrak{K}}$): 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 $\alpha, \beta \stackrel{\$}{\leftarrow} \mathbb{Z}_p^2$, $\mathcal{C} = \left(c_1 = X_1^{\alpha}, c_2 = X_2^{\beta}, c_3 = g^{\alpha+\beta} \cdot M\right)$.
- Decrypt(dk $= (\mu, \nu), \mathcal{C} = (c_1, c_2, c_3)$): Computes $M = c_3/(c_1^{1/\mu} c_2^{1/\nu})$.

Randomization

Random(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 3 (Signature Scheme)

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

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

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

Definition 4 (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 $x \xleftarrow{\$} \mathbb{Z}_p$ 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 5 (DL)

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

Definition 6 (CDH)

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

Definition 7 (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

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.

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

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

Properties:

- correctness
- soundness
- witness-indistinguishability

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

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.

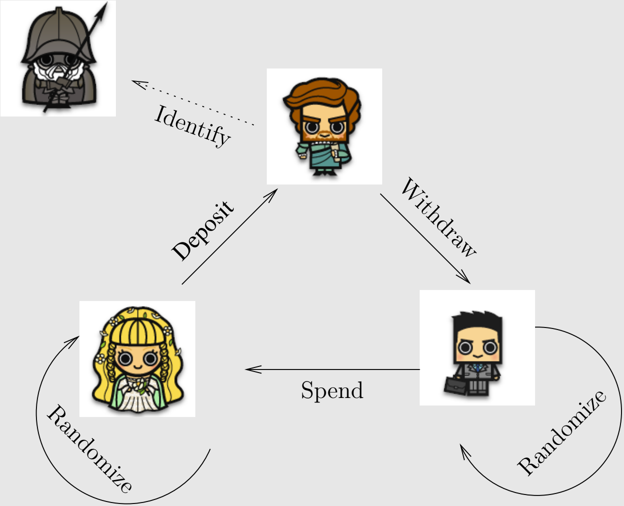
Vote without receipt freeness

Voter		Voting Center
	$\xrightarrow{c(v)}$ $\xrightarrow{\sigma(c;s)}$	

Vote with receipt freeness

Voter		Voting Center
	$\xrightarrow{\quad c(v) \quad}$ $\rightleftharpoons c' \equiv c \rightleftharpoons$ $\xrightarrow{\quad \sigma(c';s) \quad}$	

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

Definition 8 (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;

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:

Definition 9 (Signature on Ciphertexts)

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

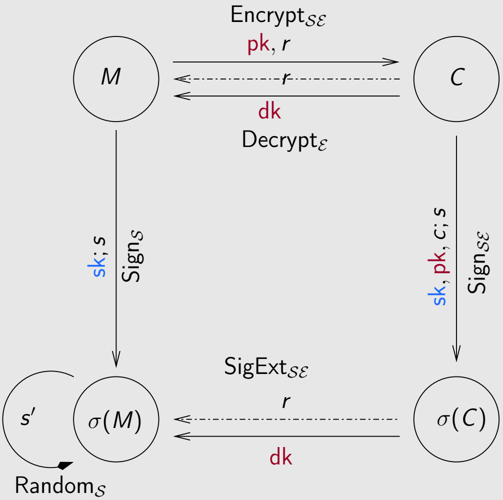
- Setup($1^{\mathfrak{K}}$): param$_e$, param$_s$;
- EKeyGen(param$_e$): pk, dk;
- SKeyGen(param$_s$): vk, sk;
- Encrypt(pk, vk, m; r): produces c on $m \in \mathcal{M}$ and pk;
- Sign(sk, pk, c; s): produces σ, on the input c under sk;
- Decrypt(dk, vk, c): decrypts c under dk;
- Verif(vk, pk, c, σ):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

Definition 10 (Extractable Randomizable Signature on Ciphertexts)

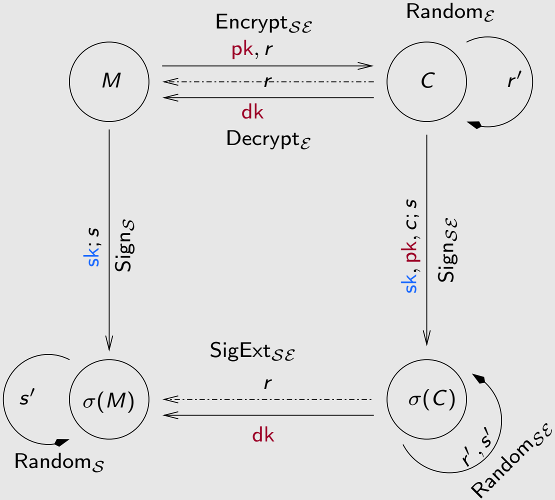
$\mathcal{SE}$=(Setup, SKeyGen, EKeyGen, Encrypt, Sign, Random, Decrypt, Verif, SigE

- 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]

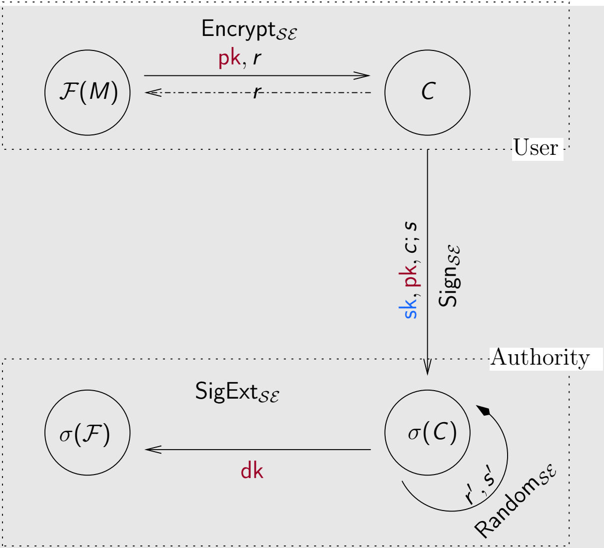


Extractable SRC

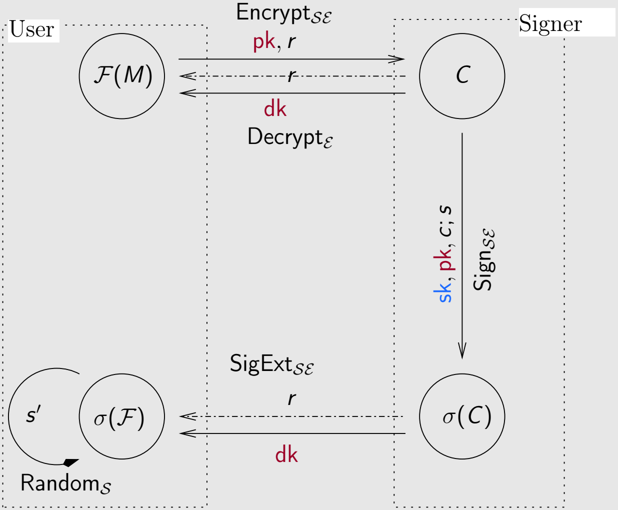


E-Voting

[PKC 2011: BFPV]



Blind Signature [PKC 2011: BFPV]



Partially-Blind Signature

User 

Signer 

$$\xleftrightarrow{\text{info}}$$

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

$$\xleftarrow{\sigma(C')}$$

Partially-Blind Signature

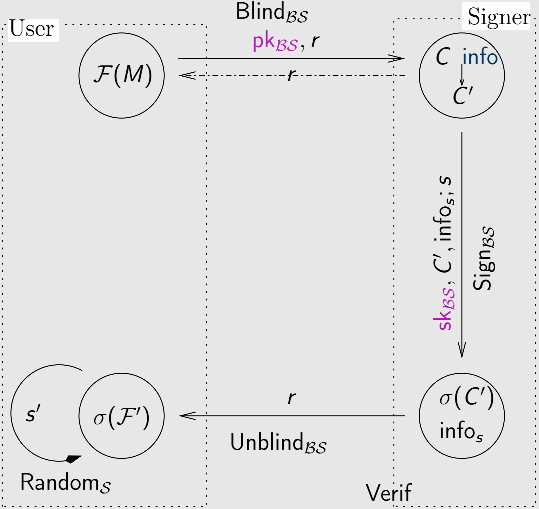
User

Signer

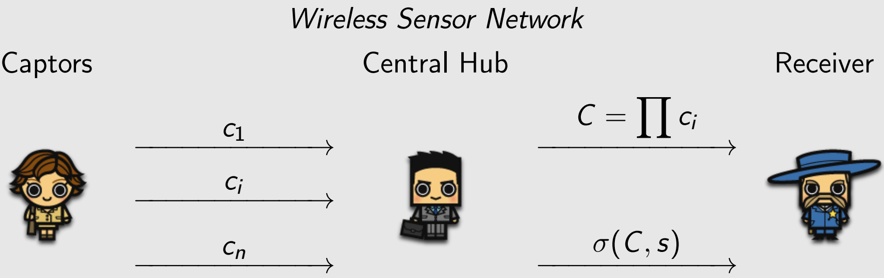
$$\xrightarrow{C' = C(M, \text{info})}$$

$$\xleftarrow{\sigma(C', \text{info}_s)}$$

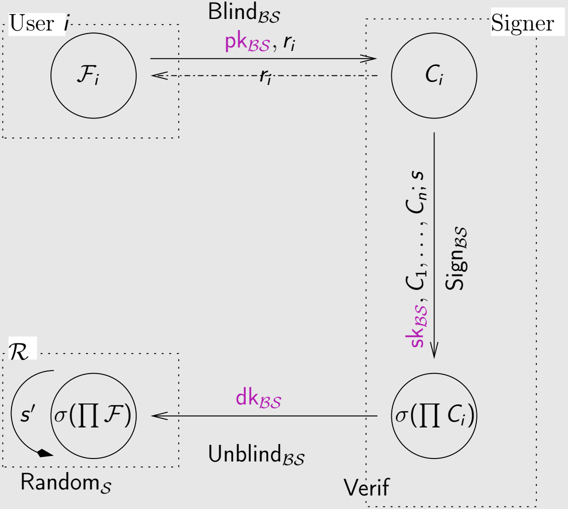
Signer-Friendly Partially Blind Signature [SCN 2012: BPV]



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

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

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 11 ((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, $Pr[a_{X_1} = \ldots = a_{X_m} = 0 \wedge a_{Z_1} \cdot \ldots \cdot a_{Z_n} \neq 0]$ 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 $u_i = g^{a_i} h^{b_i}$.

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

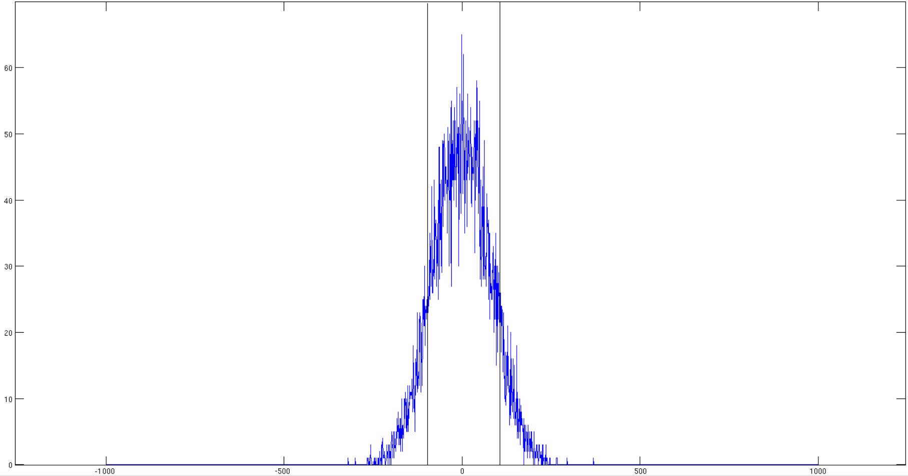
Non $(2, 1)$-programmability

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 $\rightleftharpoons$ 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.

Smooth Projective Hash Functions [CS02]

Definition [CS02,GL03,KV11]

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

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

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

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

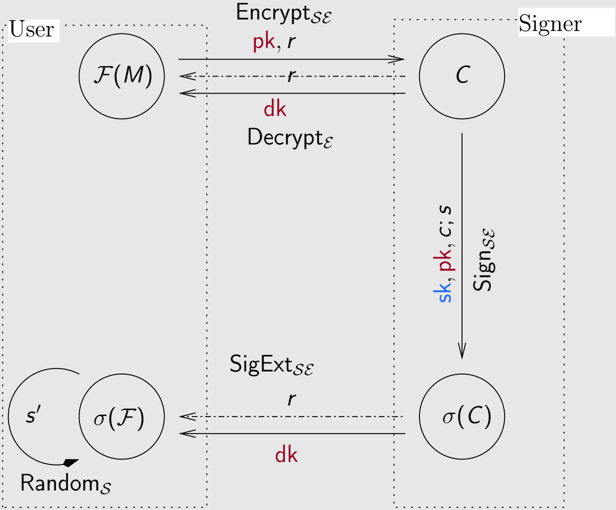
For any $x \notin L$, $H(x)$ and hp are independent

Pseudo-Randomness

For any $x \in L$, $H(x)$ is pseudo-random, without a witness w

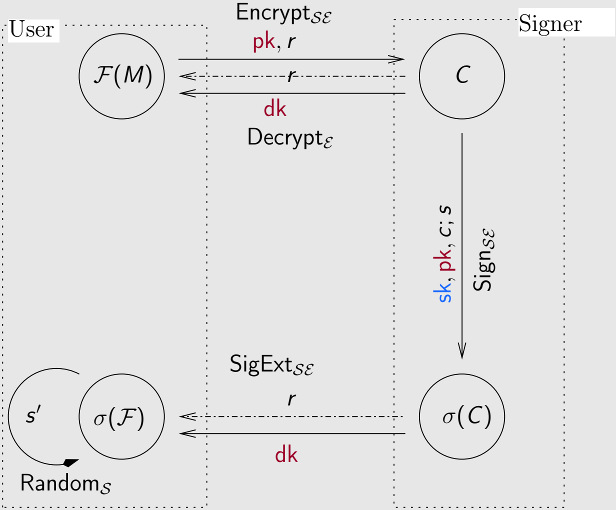
The latter property requires L to be a hard-partitioned subset of X.

Blind-Signatures [TCC 2012, PKC 13, Crypto 13, . . .]



PKC / SCN
$6\,\ell+7, 6\,\ell+5$

Blind-Signatures [TCC 2012, PKC 13, Crypto 13, . . .]



PKC / SCN

$6\,\ell + 7, 6\,\ell + 5$

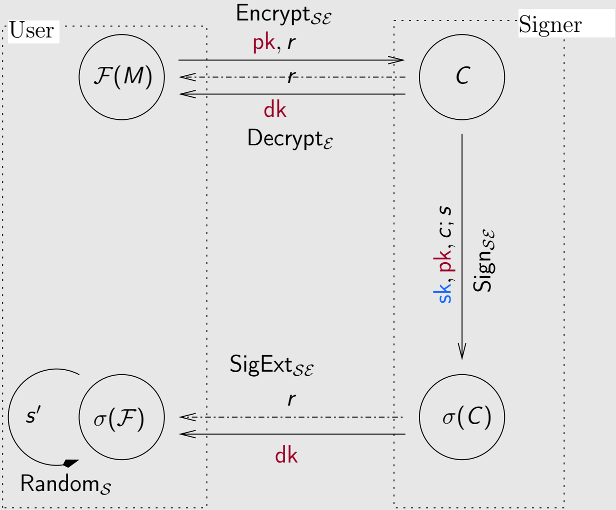
TCC

$5\,\ell + 6, 1$

Languages

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

Blind-Signatures [TCC 2012, PKC 13, Crypto 13, ...]



PKC / SCN

$6\,\ell + 7, 6\,\ell + 5$

TCC

$5\,\ell + 6, 1$

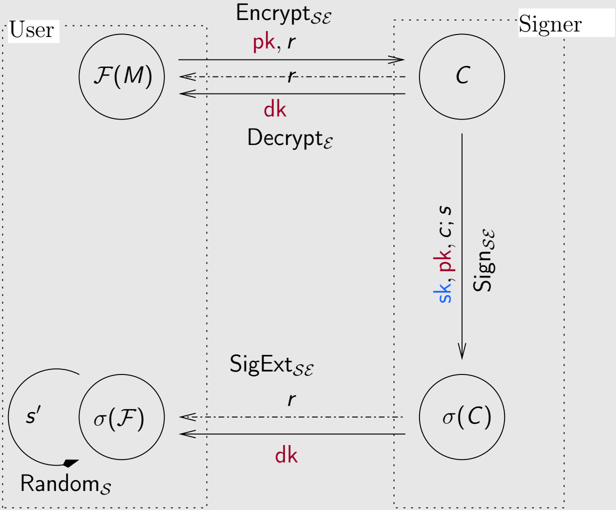
Crypto

$3\,\ell + 7, 1$

Languages

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

Blind-Signatures [TCC 2012, PKC 13, Crypto 13, . . .]



PKC / SCN

$6\,\ell + 7, 6\,\ell + 5$

TCC

$5\,\ell + 6, 1$

Soon

$(2 + o(1))\ell, 1$

Languages

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

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

Many thanks for your attention!

Any questions?

More details are available in the full version...

Horst Görtz Institut
für IT-Sicherheit