Randomizable Commutative Signature and Encryption Schemes

David Pointcheval

Joint work with Olivier Blazy, Georg Fuchsbauer and Damien Vergnaud

Ecole Normale Supérieure

June 23rd, 2011
Grenoble

Outline

Outline

Dessert Choice

If one wants to get preferences for the desserts,
one asks people to vote for

- ☐ Chocolate Cake
- ☐ Cheese Cake
- ☐ Ice Cream
- ☐ Apple

with *e.g.*, possibly 2 choices
After collection of the ballots, one counts the number of choices:

Chocolate Cake	243		1	Chocolate Cake
Cheese Cake	111	$\rightarrow$	2	Ice Cream
Ice Cream	167		3	Cheese Cake
Apple	52		4	Apple

Electronic Voting: Basic Properties

Authentication

- Only people authorized to vote should be able to vote
- Voters should vote only once

Anonymity

- Votes and voters should be unlinkable

Main Approaches

- Blind Signatures
- Homomorphic Encryption

General Approach: Homomorphic Encryption

Homomorphic Encryption & Signature

- The voter generates V_i his vote $v_i \in \{0, 1\}$ (for each $\Box$)
- The voter encrypts v_i to the server $\rightarrow \quad c_i = \mathcal{E}_{pk}(v_i; r_i)$
- The voter signs his vote $\rightarrow \quad \sigma_i = \mathcal{S}_{usk_i}(c_i; s_i)$

Such a pair (c_i, σ_i) is a ballot

- unique per voter, because it is *signed* by the voter
- anonymous, because the vote is *encrypted*

Counting: granted homomorphic encryption, anybody can compute

$$C = \prod c = \prod \mathcal{E}_{pk}(v_i; r_i) = \mathcal{E}_{pk}(\sum v_i; \sum r_i) = \mathcal{E}_{pk}(V; R)$$

The server decrypts the tally $V = \mathcal{D}_{sk}(C)$, and proves it

General Approach: Homomorphic Encryption

Security

- uniqueness per voter: the voter *signs* his vote
- anonymity: the voter *encrypts* his vote

Universal Verifiability

Soundness: every step can be proven and publicly checked

- identity of voter: proof of identity = signature
- validity of the vote: proof of bit encryption + more
- decryption: proof of decryption

All the steps (voting + counting) can be checked afterwards

General Approach: Homomorphic Encryption

Weaknesses

- Anonymity: the server can decrypt any individual vote
  $\rightarrow$ use of distributed decryption (threshold decryption)
- Receipt: if a voter wants to sell his vote, r_i is a proof
  (a coercer can also provide a modified voting client system
  in order to generate a receipt or even receive it directly)
  $\rightarrow$ re-randomization of the ciphertext

Distributed decryption is easy (*e.g.*, ElGamal allows it),
while re-randomization of the ciphertext requires more work!

Receipt-Freeness

Our goal is to prevent receipts
$\rightarrow$ receipt-free electronic system

Outline

Assumptions: Diffie-Hellman

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

$\mathbb{G}$ a cyclic group of prime order p.
The *CDH* assumption in $\mathbb{G}$ states:
for any generator $g \stackrel{\$}{\leftarrow} \mathbb{G}$, and any scalars $a, b \stackrel{\$}{\leftarrow} \mathbb{Z}_p^*$,
given (g, g^a, g^b), it is hard to compute g^{ab}.

Definition (The Decisional Diffie-Hellman problem (*DDH*))

$\mathbb{G}$ a cyclic group of prime order p.
The *DDH* assumption in $\mathbb{G}$ states:
for any generator $g \stackrel{\$}{\leftarrow} \mathbb{G}$, and any scalars $a, b, c \stackrel{\$}{\leftarrow} \mathbb{Z}_p^*$,
given (g, g^a, g^b, g^c), it is hard to decide whether $c = ab$ or not.

In some pairing-friendly groups, the latter assumption is wrong.

Assumptions: Linear Problem

Definition (Decision Linear Assumption (*DLin*))

$\mathbb{G}$ a cyclic group of prime order p.
The *DLin* assumption states:
for any generator $g \stackrel{\$}{\leftarrow} \mathbb{G}$, and any scalars $a, b, x, y, c \stackrel{\$}{\leftarrow} \mathbb{Z}_p^*$,
given $(g, g^x, g^y, g^{xa}, g^{yb}, g^c)$,
it is hard to decide whether $c = a + b$ or not.

Equivalently, given a reference triple $(u = g^x, v = g^y, g)$
and a new triple $(U = u^a = g^{xa}, V = v^b = g^{yb}, T = g^c)$,
decide whether $T = g^{a+b}$ or not (that is $c = a + b$).

General Tools: Signature

Definition (Signature Scheme)

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

- $\mathit{Setup}(1^k) \rightarrow$ global parameters *param*;
- $\mathit{SKeyGen}(\mathit{param}) \rightarrow$ pair of keys $(\mathit{sk}, \mathit{vk})$;
- $\mathit{Sign}(\mathit{sk}, m; s) \rightarrow$ signature σ, using the random coins s;
- $\mathit{Verif}(\mathit{vk}, m, \sigma) \rightarrow$ validity of σ

Signature: Examples

In a group $\mathbb{G}$ of order p, with a generator g, and a bilinear map $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$

Waters Signature [Waters, 2005]

For a message $M = (M_1, \ldots, M_k) \in \{0,1\}^k$, we define $\mathcal{F}(M) = u_0 \prod_{i=1}^k u_i^{M_i}$, where $\vec{u} = (u_0, \ldots, u_k) \xleftarrow{\$} \mathbb{G}^{k+1}$.
For an additional generator $h \xleftarrow{\$} \mathbb{G}$.

- *SKeyGen*: $vk = X = g^x$, $sk = Y = h^x$, for $x \xleftarrow{\$} \mathbb{Z}_p$;
- $\mathit{Sign}(sk = Y, M; s)$, for $M \in \{0,1\}^k$ and $s \xleftarrow{\$} \mathbb{Z}_p$
  $\to \quad \sigma = (\sigma_1 = Y \cdot \mathcal{F}(M)^s, \sigma_2 = g^{-s})$;
- $\mathit{Verif}(vk = X, M, \sigma = (\sigma_1, \sigma_2))$ checks whether
  $$e(g, \sigma_1) \cdot e(\mathcal{F}(M), \sigma_2) = e(X, h).$$

General Tools: Encryption

Definition (Encryption Scheme)

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

- $\mathit{Setup}(1^k) \quad \to \quad$ global parameters *param*;
- $\mathit{EKeyGen}(\mathit{param}) \quad \to \quad$ pair of keys (pk, dk);
- $\mathit{Encrypt}(pk, m; r) \quad \to \quad$ ciphertext c, using the random coins r;
- $\mathit{Decrypt}(dk, c) \quad \to \quad$ plaintext, or $\perp$ if the ciphertext is invalid.

Homomorphic Encryption

For some group laws: $\oplus$ on the plaintext, $\otimes$ on the ciphertext, and $\odot$ on the randomness

$$\mathit{Encrypt}(pk, m_1; r_1) \otimes \mathit{Encrypt}(pk, m_2; r_2) = \mathit{Encrypt}(pk, m_1 \oplus m_2; r_1 \odot r_2)$$
$$\mathit{Decrypt}(sk, \mathit{Encrypt}(pk, m_1; r_1) \otimes \mathit{Encrypt}(pk, m_2; r_2)) = m_1 \oplus m_2$$

Encryption: Examples

In a group $\mathbb{G}$ of order p, with a generator g:

ElGamal Encryption [ElGamal, 1985]

- *EKeyGen*: $dk = x \xleftarrow{\$} \mathbb{Z}_p$, $pk = X = g^x$;
- $\mathit{Encrypt}(pk = X, m; r)$, for $m \in \mathbb{G}$ and $r \xleftarrow{\$} \mathbb{Z}_p$
  $\to \quad c = (c_1 = g^r, c_2 = X^r \cdot m)$;
- $\mathit{Decrypt}(dk = x, c = (c_1, c_2)) \quad \to \quad m = c_2 / c_1^x$.

Linear Encryption [Boneh, Boyen, Shacham, 2004]

- *EKeyGen*: $dk = (x_1, x_2) \xleftarrow{\$} \mathbb{Z}_p^2$, $pk = (X_1 = g^{x_1}, X_2 = g^{x_2})$;
- $\mathit{Encrypt}(pk = (X_1, X_2), m; (r_1, r_2))$, for $m \in \mathbb{G}$ and $(r_1, r_2) \xleftarrow{\$} \mathbb{Z}_p^2$
  $\to \quad c = (c_1 = X_1^{r_1}, c_2 = X_2^{r_2}, c_3 = g^{r_1 + r_2} \cdot m)$;
- $\mathit{Decrypt}(dk = (x_1, x_2), c = (c_1, c_2, c_3)) \quad \to \quad m = c_3 / c_1^{1/x_1} c_2^{1/x_2}$.

Encryption: Properties

In a group $\mathbb{G}$ of order p, with a generator g:

ElGamal Encryption

$dk = x \xleftarrow{\$} \mathbb{Z}_p$, $pk = X = g^x$

$$\begin{aligned} & \mathit{Encrypt}(X, m_1; r_1) \times \mathit{Encrypt}(X, m_2; r_2) \\ & = (g^{r_1}, X^{r_1} \cdot m_1) \times (g^{r_2}, X^{r_2} \cdot m_2) \\ & = (g^{r_1 + r_2}, X^{r_1 + r_2} \cdot m_1 \cdot m_2) = \mathit{Encrypt}(X, m_1 \cdot m_2; r_1 + r_2) \end{aligned}$$

$\to \quad (\oplus_M = \times, \otimes_C = \times, \odot_R = +)$ homomorphism
$\to \quad$ re-randomization: multiplication by $\mathit{Encrypt}(X, 1; r)$.

With $m = g^M$: $\mathit{Encrypt}^*(pk, M; (r_1, r_2)) = \mathit{Encrypt}(pk, g^M; (r_1, r_2))$

$\to \quad (\oplus_M = +, \otimes_C = \times, \odot_R = +)$ homomorphism
$\to \quad$ re-randomization: multiplication by $\mathit{Encrypt}^*(X, 0; r)$.

Encryption: Properties

In a group $\mathbb{G}$ of order p, with a generator g:

Linear Encryption

$dk = (x_1, x_2) \stackrel{\$}{\leftarrow} \mathbb{Z}_p^2$, $pk = (X_1 = g^{x_1}, X_2 = g^{x_2})$

$$\begin{aligned} & \mathit{Encrypt}((X_1, X_2), m_1; (r_1, r_1')) \times \mathit{Encrypt}((X_1, X_2), m_2; (r_2, r_2')) \\ &= (X_1^{r_1}, X_2^{r_1'}, g^{r_1 + r_1'} \cdot m_1) \times (X_1^{r_2}, X_2^{r_2'}, g^{r_2 + r_2'} \cdot m_2) \\ &= (X_1^{r_1 + r_2}, X_2^{r_1' + r_2'}, g^{r_1 + r_1' + r_2 + r_2'} \cdot m_1 \cdot m_2) \\ &= \mathit{Encrypt}((X_1, X_2), m_1 \cdot m_2; (r_1 + r_2, r_1' + r_2')) \end{aligned}$$

$\rightarrow \quad (\oplus_M = \times, \otimes_C = \times, \odot_R = +)$ homomorphism

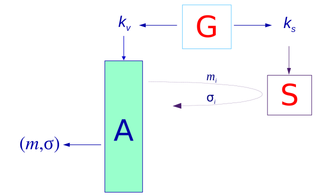
With $m = g^M \quad \rightarrow \quad (\oplus_M = +, \otimes_C = \times, \odot_R = +)$ homomorphism

Security Notions: Signature

Signature: EF-CMA

Existential Unforgeability under Chosen-Message Attacks

An adversary should not be able to generate a new valid message-signature pair even if it is allowed to ask signatures on any message of its choice



Impossibility to forge signatures

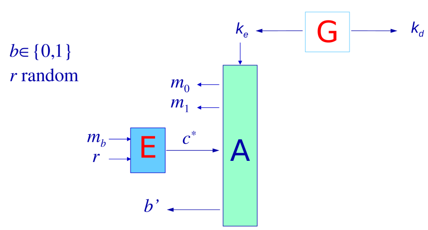
Waters signature reaches EF-CMA under the *CDH* assumption

Security Notions: Encryption

Encryption: IND-CCA

Indistinguishability under Chosen-Plaintext Attacks

An adversary that chooses two messages, and receives the encryption of one of them, should not be able to decide which one has been encrypted



Impossibility to learn any information about the plaintext

ElGamal (resp. Linear) encryption reaches IND-CPA under the *DDH* (resp. *DLin*) assumption

Groth-Sahai Commitments

[Groth, Sahai, 2008]

Under the *DLin* assumption, the commitment key is:

$$(\mathbf{u}_1 = (u_{1,1}, 1, g), \mathbf{u}_2 = (1, u_{2,2}, g), \mathbf{u}_3 = (u_{3,1}, u_{3,2}, u_{3,3})) \in (\mathbb{G}^3)^3$$

Initialization

$$\mathbf{u}_3 = \mathbf{u}_1^{\lambda} \odot \mathbf{u}_2^{\mu} = (u_{3,1} = u_{1,1}^{\lambda}, u_{3,2} = u_{2,2}^{\mu}, u_{3,3} = g^{\lambda+\mu})$$

with $\lambda, \mu \stackrel{\$}{\leftarrow} \mathbb{Z}_p^*$, and random elements $u_{1,1}, u_{2,2} \stackrel{\$}{\leftarrow} \mathbb{G}$.

It means that $\mathbf{u}_3$ is a linear tuple w.r.t. $(u_{1,1}, u_{2,2}, g)$.

Groth-Sahai Commitments

Group Element Commitment

To commit a group element $X \in \mathbb{G}$,
one chooses random coins $s_1, s_2, s_3 \in \mathbb{Z}_p$ and sets

$$\mathcal{C}(X) := (1, 1, X) \odot \mathbf{u}_1^{s_1} \odot \mathbf{u}_2^{s_2} \odot \mathbf{u}_3^{s_3} = (u_{1,1}^{s_1} \cdot u_{3,1}^{s_3}, u_{2,2}^{s_2} \cdot u_{3,2}^{s_3}, X \cdot g^{s_1+s_2} \cdot u_{3,3}^{s_3}).$$

Scalar Commitment

To commit a scalar $x \in \mathbb{Z}_p$,
one chooses random coins $\gamma_1, \gamma_2 \in \mathbb{Z}_p$ and sets

$$\mathcal{C}'(x) := (u_{3,1}^x, u_{3,2}^x, (u_{3,3}g)^x) \odot \mathbf{u}_1^{\gamma_1} \odot \mathbf{u}_3^{\gamma_2} = (u_{3,1}^{x+\gamma_2} \cdot u_{1,1}^{\gamma_1}, u_{3,2}^{x+\gamma_2}, u_{3,3}^{x+\gamma_2} \cdot g^{x+\gamma_1}).$$

Groth-Sahai Commitments

- If correct initialization of commitment key ($\mathbf{u}_3$ a linear tuple), these commitments are perfectly binding
- With some initialization parameters, the committed values can even be extracted $\rightarrow$ extractable commitments
- Using pairing product equations, one can make proofs on many relations between scalars and group elements:

$$\prod_j e(A_j, X_j)^{\alpha_j} \prod_i e(Y_i, B_i)^{\beta_i} \prod_{i,j} e(X_i, Y_j)^{\gamma_{i,j}} = t,$$

  where the A_j, B_i, and t are constant group elements, α_i, β_j, and $\gamma_{i,j}$ are constant scalars, and X_j and Y_i are either group elements in $\mathbb{G}_1$ and $\mathbb{G}_2$, or of the form $g_1^{x_j}$ or $g_2^{y_i}$, respectively, to be committed.
- The proofs are perfectly sound

Groth-Sahai Proofs

- If $\mathbf{u}_3$ a linear tuple, these commitments are perfectly binding
- The proofs are perfectly sound

- If $\mathbf{u}_3$ is a random tuple, the commitments are perfectly hiding
- The proofs are perfectly witness hiding

- Under the *DLin* assumption, with a correct initialization, proofs are witness hiding

Outline

Dessert Choice

A ballot consists of one or two crosses in

- ☐ Chocolate Cake
- ☐ Cheese Cake
- ☐ Ice Cream
- ☐ Apple

Each box is thus expressed as a bit: $v_i \in \{0, 1\}$, for $i = 1, 2, 3, 4$
With the additional constraint (at most 2 choices): $\sum_i v_i \in \{0, 1, 2\}$

In the following, we focus on one box only:

- V_i is the i-th voter
- v_i is the value of the box for this voter: 0 or 1

Voting Procedure

Cryptographic Primitives

- Signature $\mathcal{S} = (\mathit{Setup}, \mathit{SKeyGen}, \mathit{Sign}, \mathit{Verif})$ that is EF-CMA, *e.g.*, Waters Signature;
- Homomorphic enc. $\mathcal{E} = (\mathit{Setup}, \mathit{EKeyGen}, \mathit{Encrypt}, \mathit{Decrypt})$ that is IND-CPA, *e.g.*, ElGamal or Linear Encryption

+ distributed decryption, as ElGamal and Linear schemes allow

Initialization

- The authority owns a signing/verification key-pair (sk, vk)
- The ballot-box owns an encryption key pk, which decryption capability is distributed among the board members
- Each voter V_i owns a signing/verification key-pair (usk_i, uvk_i)

Voting Procedure

Voting Phase

Voter V_i — Server S

$c_i = \mathit{Encrypt}(pk, v_i; r_i)$
$\sigma_i = \mathit{Sign}(usk_i, c_i; s_i)$
$\Pi_c =$ Proof of bit encryption $\xrightarrow{c_i, \sigma_i, \Pi_c}$

$\xleftarrow{\Sigma_i}$ $\Sigma_i = \mathit{Sign}(sk, c_i; s'_i)$

- from (σ_i, Π_c): authorization and uniqueness of a voter
- from c_i: privacy for the voter unless individual votes are decrypted
- with Σ_i: a voter can complain if his vote is not in the ballot-box

Counting Procedure

Counting Phase

- Anybody can check all the votes (c_i, σ_i, Π_c)
- Anybody can compute

$$C = \prod c_i = \prod \mathcal{E}_{pk}(v_i; r_i) = \mathcal{E}_{pk}(\sum v_i; \sum r_i) = \mathcal{E}_{pk}(V; R)$$

- The board members decrypt C in a distributed and verifiable way, into V

- Everything is verifiable: universal verifiability
- Board members accept to participate to one decryption only: C
  - $\rightarrow$ individual votes are protected
  - $\rightarrow$ anonymity

Summary

Security

- uniqueness per voter: signature
- anonymity: encryption and distributed decryption
- universal verifiability: every step is publicly verifiable
  - Soundness: the server cannot add ballots
  - Dispute: the server cannot remove ballots

Weakness: Receipt

To sell his vote, the voter reveals his random coins r_i as a receipt

Receipt-freeness: the voter should not know the random coins!

Re-Randomization

Voting Phase

Voter V_i | Server S

$c_i = Encrypt(pk, v_i; r_i)$

$\Pi_c =$ Proof of bit encryption

$\xrightarrow{c_i, \Pi_c}$

$\xleftarrow{c'_i}$ $\quad c'_i = Random(c_i; r'_i)$

$\xleftrightarrow{\text{Proof}(c'_i \equiv c_i)}$

$\sigma_i = Sign(usk_i, c'_i; s_i)$ $\quad \xrightarrow{\sigma_i}$

$\xleftarrow{\Sigma_i}$ $\quad \Sigma_i = Sign(sk, c_i; s'_i)$

Non-transferable proof of $c'_i \equiv c_i$: verifier-designated proof
Proof of knowledge of $[r'_i$ such that $c'_i = Random(c_i, r'_i)]$ or $[usk_i]$

Security

Re-Randomization

- re-randomization: the voter no longer knows the random coins
- designated-verifier proof:
  - Voter convinced: c'_i contains his vote
  - Receipt-freeness: the server cannot transfer this proof

Weakness: verifiability

The proof Π_c can be verified by the server on c
but not by users on c': no universal verifiability
The proof should be re-randomized (adapted) by the server:
Possible with Groth-Sahai methodology

Security

Weakness: interactions

- interactive proof
- 2-round voting (at best!)

Non-Interactive Receipt-Freeness

Our goal is to achieve receipt-freeness
but in a non-interactive way

Outline

Signatures on Randomizable Ciphertexts

Voting Phase

Voter V_i | Server S

$c_i = Encrypt(pk, v_i; r_i)$

$\sigma_i = Sign(usk_i, c_i; s_i)$

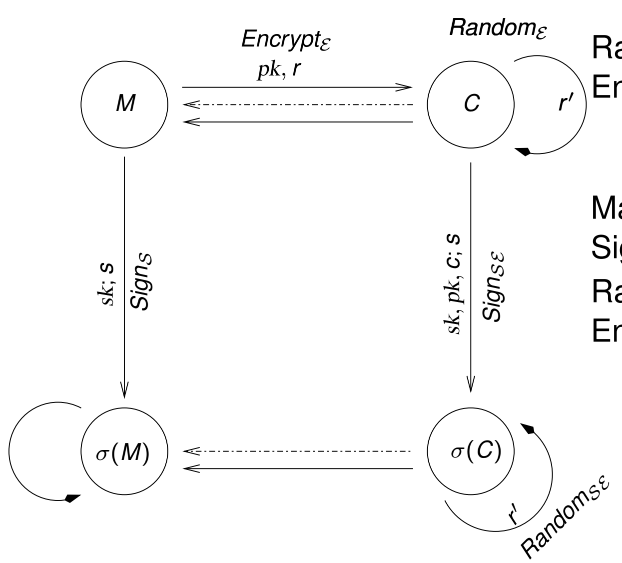
Π_c = Proof of bit encryption $\xrightarrow{c_i, \sigma_i, \Pi_c}$ $(c'_i, \sigma'_i, \Pi'_c) = Random(c_i, \sigma_i, \Pi_c; r'_i)$

$\xleftarrow{c'_i, \Pi'_c, \Sigma_i}$ $\Sigma_i = Sign(sk, (c'_i, \Pi'_c); s'_i)$

The server not only adapts the proof, but the signature too!

- from (σ_i, Π_c): authorization and uniqueness of a voter
- from c_i: privacy for the voter
- from *Random*: receipt-freeness (unknown random coins $r_i + r'_i$)

Signatures on Randomizable Ciphertexts



Randomizable Encryption

Malleable Signature on Randomizable Encryption

Linear Encryption

In a group $\mathbb{G}$ of order p, with a generator g, and a bilinear map $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$

Linear Encryption

[Boneh, Boyen, Shacham, 2004]

- $EKeyGen$: $dk = (x_1, x_2) \stackrel{\$}{\leftarrow} \mathbb{Z}_p^2$, $pk = (X_1 = g^{x_1}, X_2 = g^{x_2})$;
- $Encrypt(pk = (X_1, X_2), m; (r_1, r_2))$, for $m \in \mathbb{G}$ and $(r_1, r_2) \stackrel{\$}{\leftarrow} \mathbb{Z}_p^2$
  $\to \quad c = (c_1 = X_1^{r_1}, c_2 = X_2^{r_2}, c_3 = g^{r_1+r_2} \cdot m)$;
- $Decrypt(dk = (x_1, x_2), c = (c_1, c_2, c_3)) \quad \to \quad m = c_3 / c_1^{1/x_1} c_2^{1/x_2}$.

Re-Randomization

- $Random_\mathcal{E}(pk = (X_1, X_2), c = (c_1, c_2, c_3); (r'_1, r'_2))$, for $(r'_1, r'_2) \stackrel{\$}{\leftarrow} \mathbb{Z}_p^2$
  $\to \quad c' = (c'_1 = c_1 \cdot X_1^{r'_1}, c'_2 = c_2 \cdot X_2^{r'_2}, c'_3 = c_3 \cdot g^{r'_1 + r'_2})$.

Waters Signature

In a group $\mathbb{G}$ of order p, with a generator g,
and a bilinear map $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$

Waters Signature [Waters, 2005]

For a message $M = (M_1, \ldots, M_k) \in \{0,1\}^k$,
we define $F = \mathcal{F}(M) = u_0 \prod_{i=1}^k u_i^{M_i}$, where $\vec{u} = (u_0, \ldots, u_k) \stackrel{\$}{\leftarrow} \mathbb{G}^{k+1}$.
For an additional generator $h \stackrel{\$}{\leftarrow} \mathbb{G}$.

- *SKeyGen*: $vk = X = g^x$, $sk = Y = h^x$, for $x \stackrel{\$}{\leftarrow} \mathbb{Z}_p$;
- *Sign*$(sk = Y, F; s)$, for $M \in \{0,1\}^k$, $F = \mathcal{F}(M)$, and $s \stackrel{\$}{\leftarrow} \mathbb{Z}_p$
  $\to \quad \sigma = (\sigma_1 = Y \cdot F^s, \sigma_2 = g^{-s})$;
- *Verif*$(vk = X, M, \sigma = (\sigma_1, \sigma_2))$ checks whether
$$e(g, \sigma_1) \cdot e(F, \sigma_2) = e(X, h).$$

Waters Signature on a Linear Ciphertext: Idea

We define $F = \mathcal{F}(M) = u_0 \prod_{i=1}^k u_i^{M_i}$, and encrypt it
$$c = (c_1 = X_1^{r_1}, c_2 = X_2^{r_2}, c_3 = g^{r_1+r_2} \cdot F)$$

- *KeyGen*: $vk = X = g^x$, $sk = Y = h^x$, for $x \stackrel{\$}{\leftarrow} \mathbb{Z}_p$
  $dk = (x_1, x_2) \stackrel{\$}{\leftarrow} \mathbb{Z}_p^2$, $pk = (X_1 = g^{x_1}, X_2 = g^{x_2})$
- *Sign*$((X_1, X_2), Y, c; s)$, for $c = (c_1, c_2, c_3)$
  $\to \quad \sigma = (\sigma_1 = Y \cdot c_3^s, \sigma_2 = (c_1^s, c_2^s), \sigma_3 = (g^s, X_1^s, X_2^s))$
- *Verif*$((X_1, X_2), X, c, \sigma)$ checks $\quad e(g, \sigma_1) = e(X, h) \cdot e(\sigma_{3,0}, c_3)$
$$e(\sigma_{2,0}, g) = e(c_1, \sigma_{3,0}) \qquad e(\sigma_{2,1}, g) = e(c_2, \sigma_{3,0})$$
$$e(\sigma_{3,1}, g) = e(X_1, \sigma_{3,0}) \qquad e(\sigma_{3,2}, g) = e(X_2, \sigma_{3,0})$$

σ_3 is needed for ciphertext re-randomization

Re-Randomization of Ciphertext

$$c = (c_1 = X_1^{r_1}, \quad c_2 = X_2^{r_2}, \quad c_3 = g^{r_1+r_2} \cdot F \quad)$$
$$\sigma = (\sigma_1 = Y \cdot c_3^s, \quad \sigma_2 = (c_1^s, c_2^s), \quad \sigma_3 = (g^s, X_1^s, X_2^s) \quad)$$

after re-randomization by $\quad (r'_1, r'_2)$

$$c' = (c'_1 = c_1 \cdot X_1^{r'_1}, \quad c'_2 = c'_2 \cdot X_2^{r'_2}, \quad c'_3 = c_3 \cdot g^{r'_1+r'_2} \quad)$$
$$\sigma' = (\sigma'_1 = \sigma_1 \cdot \sigma_{3,0}^{r'_1+r'_2}, \; \sigma'_2 = (\sigma_{2,0} \cdot \sigma_{3,1}^{r'_1}, \sigma_{2,1} \cdot \sigma_{3,2}^{r'_2}), \; \sigma'_3 = \sigma_3 \quad)$$

Anybody can publicly re-randomize c into c'
with additional random coins (r'_1, r'_2),
and adapt the signature σ of c into σ' of c'

Unforgeability under Chosen-Ciphertext Attacks

Chosen-Ciphertext Attacks

The adversary is allowed to ask any valid ciphertext of his choice to the signing oracle

Because of the re-randomizability of the ciphertext-signature,
we cannot expect resistance to existential forgeries,
but we should allow a restricted malleability only:

Forgery

A valid ciphertext-signature pair, so that the plaintext is different from all the plaintexts in the ciphertexts sent to the signing oracle

Unforgeability

From a valid ciphertext-signature pair:

$$c = (c_1 = X_1^{r_1}, c_2 = X_2^{r_2}, c_3 = g^{r_1+r_2} \cdot F)$$
$$\sigma = (\sigma_1 = Y \cdot c_3^s, \sigma_2 = (c_1^s, c_2^s), \sigma_3 = (g^s, X_1^s, X_2^s))$$

and the decryption key (x_1, x_2), one extracts

$$\begin{array}{lll} F = & c_3/(c_1^{1/x_1} c_2^{1/x_2}) & \\ \Sigma = (& \Sigma_1 = \sigma_1/(\sigma_{2,0}^{1/x_1} \sigma_{2,1}^{1/x_2}), & \Sigma_2 = \sigma_{3,0}) \\ = (& = Y \cdot F^s & = g^s) \end{array}$$

Security of Waters signature is for a pair (M, Σ)

$\rightarrow$ needs of a proof of knowledge Π_M of M in $F = \mathcal{F}(M)$

bit-by-bit commitment of M and Groth-Sahai proof

Chosen-Message Attacks

From a valid ciphertext $c = (c_1 = X_1^{r_1}, c_2 = X_2^{r_2}, c_3 = g^{r_1+r_2} \cdot F)$,
and the additional proof of knowledge of M,
one extracts M and asks for a Waters signature:

$$\Sigma = (\Sigma_1 = Y \cdot F^s, \Sigma_2 = g^s)$$

In this signature, the random coins s are unknown,
we thus need to know the coins in c

$\rightarrow$ needs of a proof of knowledge Π_r of r_1, r_2 in c

bit-by-bit commitment of r_1, r_2 and Groth-Sahai proof

From the random coins r_1, r_2 (and the decryption key):

$$\begin{array}{llll} \sigma = (\sigma_1 = \Sigma_1 \cdot \Sigma_2^{r_1+r_2}, & \sigma_2 = (\Sigma_2^{x_1 r_1}, \Sigma_2^{x_2 r_2}), & \sigma_3 = (\Sigma_2, \Sigma_2^{r_1}, \Sigma_2^{r_2}) &) \\ = Y \cdot c_3^s, & = (c_1^s, c_2^s), & = (g^s, X_1^s, X_2^s) & \end{array}$$

Security

Chosen-Ciphertext Attacks

A valid ciphertext $C = (c_1, c_2, c_3, \Pi_M, \Pi_r)$ is a

- ciphertext $c = (c_1, c_2, c_3)$
- a proof of knowledge Π_M of the plaintext M in $F = \mathcal{F}(M)$
- a proof of knowledge Π_r of the random coins r_1, r_2

From such a ciphertext and the decryption key (x_1, x_2),
and a Waters signing oracle, one can generate a signature on C

Forgery

From a valid ciphertext-signature pair (C, σ), where C encrypts M,
one can generate a Waters signature on M

Security

- From the Waters signing oracle,
  we answer Chosen-Ciphertext Signing queries
- From a Forgery, we build a Waters Existential Forgery

Security Level

Since the Waters signature is EF-CMA under the *CDH* assumption,
our signature on randomizable ciphertext is Unforgeable
against Chosen-Ciphertext Attacks
under the *CDH* assumption

Properties

Proofs

Since we use the Groth-Sahai methodology for the proofs Π_M and Π_r

- in case of re-randomization of c, one can adapt Π_M and Π_r
- because of the need of M, but also r_1 and r_2 in the simulation, we need bit-by-bit commitments:
  - M can be short (ℓ bit-long)
  - r_1 and r_2 are random in $\mathbb{Z}_p$

  $\rightarrow$ C is large!

Efficiency

We can improve efficiency: shorter signatures

Outline

Electronic Cash

Electronic Coins [Chaum, 1981]

Expected properties:

- coins are signed by the bank, for unforgeability
- coins must be distinct to detect/avoid double-spending
- the bank should not know to whom it gave a coin, for anonymity

Electronic Cash

The process is the following one:

- Withdrawal: the user gets a coin c from the bank
- Spending: the user spends a coin c in a shop
- Deposit: the shop gives back the money to the bank

Blind Signatures

We thus want:

- Anonymity: the bank cannot link a withdrawal to a deposit to know where a user spent a coin

  $\rightarrow$ blind signature
- No double-spending: a coin should not be used twice

  $\rightarrow$ fair blind signature

Blind RSA

[Chaum, 1981]

The easiest way for blind signatures, is to blind the message:
To get an RSA signature on m under public key (n, e),

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

Indeed,

$$\sigma = \sigma'/r = M'^d/r = (M \cdot r^e)^d/r = M^d \cdot r/r = M^d \bmod n$$

$\rightarrow$ Proven under the One-More RSA

[Bellare, Namprempre, Pointcheval, Semanko, 2001]

$\rightarrow$ Perfectly blind signature

Extractability

As already noted, from a valid ciphertext-signature pair:

$$c = (c_1 = X_1^{r_1}, c_2 = X_2^{r_2}, c_3 = g^{r_1+r_2} \cdot F)$$
$$\sigma = (\sigma_1 = Y \cdot c_3^s, \sigma_2 = (c_1^s, c_2^s), \sigma_3 = (g^s, X_1^s, X_2^s))$$

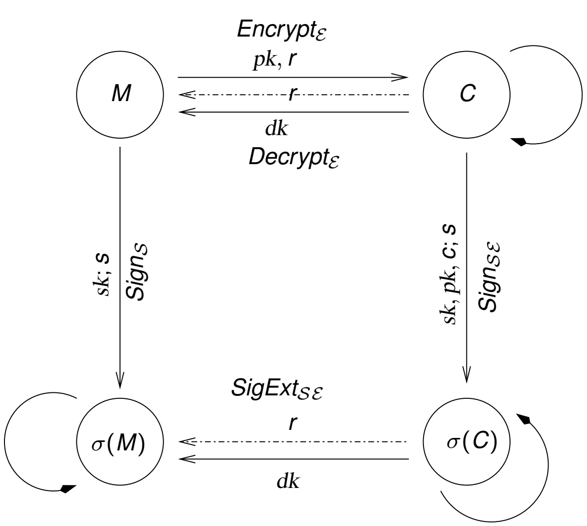
and the decryption key (x_1, x_2), one extracts

$$\begin{array}{lll} F = & c_3/(c_1^{1/x_1} c_2^{1/x_2}) & \\ \Sigma = (& \Sigma_1 = \sigma_1/(\sigma_{2,0}^{1/x_1} \sigma_{2,1}^{1/x_2}), & \Sigma_2 = \sigma_{3,0}) \\ = (& = Y \cdot F^s & = g^s) \end{array}$$

A plain Waters Signature

One can do the same from the random coins (r_1, r_2)

Extractable Signatures



Blind Signatures

Our Approach

To get a signature on M,

- The user commits/encrypts M into C, under random coins r
- The signer signs C into $\sigma(C)$, under random coins s
- The user extracts a signature $\sigma(M)$, granted the random coins r

Weakness

The signer can recognize his signature: the random coins s in $\sigma(M)$

$\rightarrow$ Randomizable Signature

Security

- Encryption hides M
- Re-randomization of signature hides $\sigma(M)$

Randomizable Signatures

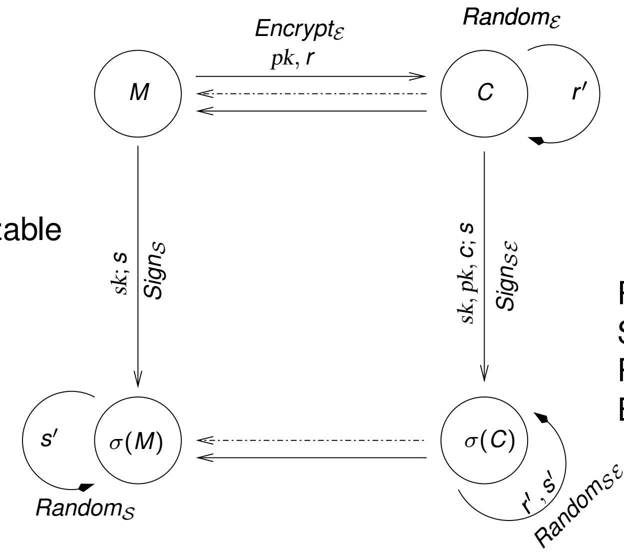
Waters Signature

- *SKeyGen*: $vk = X = g^x$, $sk = Y = h^x$, for $x \stackrel{\$}{\leftarrow} \mathbb{Z}_p$;
- *Sign*$(sk = Y, M; s)$, for $M \in \{0,1\}^k$ and $s \stackrel{\$}{\leftarrow} \mathbb{Z}_p$
  $\rightarrow \quad \sigma = (\sigma_1 = Y \cdot \mathcal{F}(M)^s, \sigma_2 = g^{-s})$;
- *Verif*$(vk = X, M, \sigma = (\sigma_1, \sigma_2))$ checks whether
$$e(g, \sigma_1) \cdot e(\mathcal{F}(M), \sigma_2) = e(X, h).$$

Re-Randomization

$Random_{\mathcal{S}}(vk = X, M, \sigma; s') : \sigma' = (\sigma'_1 = \sigma_1 \cdot \mathcal{F}(M)^{s'}, \sigma'_2 = \sigma_2 \cdot g^{-s'})$
this is exactly *Sign*$(sk = Y, M; s + s')$

Randomizable Signatures



Randomizable Encryption

Randomizable Signature

Randomizable Signature on Randomizable Encryption

Blind Signatures

Such a primitive can be used for a Waters Blind Signature:

- Unforgeability: one-more forgery would imply a forgery against the signature scheme (*CDH* assumption)
- Blindness: a distinguisher would break indistinguishability of the encryption scheme (*DLin* assumption)

Efficiency

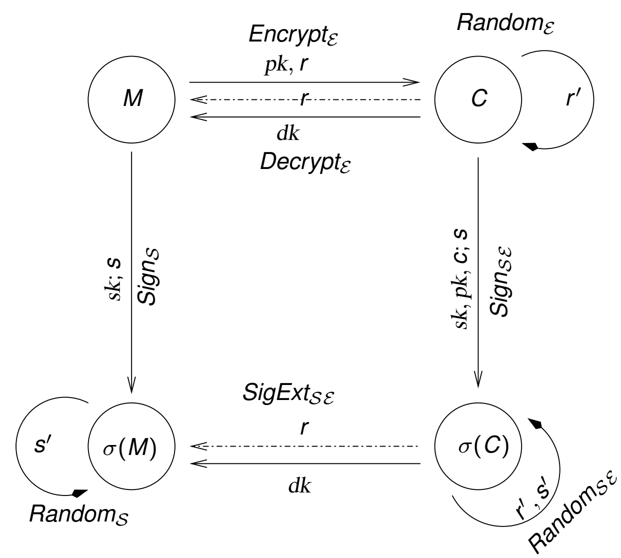
We obtain a plain Waters Signature

$\rightarrow$ Blind Signature: with a real Waters Signature

Fair Blind Signature

The user encrypts M into C, under random coins r, and the authority encryption key

Randomizable Commutative Signature/Encryption



Conclusion

Randomizable Commutative Signature/Encryption

Various Applications

- non-interactive receipt-free electronic voting scheme
- (fair) blind signature

Security relies on the *CDH* and the *DLin* assumptions
For an ℓ-bit message, ciphertext-signature:
$9\ell + 24$ group elements

A more efficient variant with asymmetric pairing
on the *CDH** and the *SXDH* assumptions
Ciphertext-signature: $6\ell + 7$ group elements in $\mathbb{G}_1$
and $6\ell + 5$ group elements in $\mathbb{G}_2$