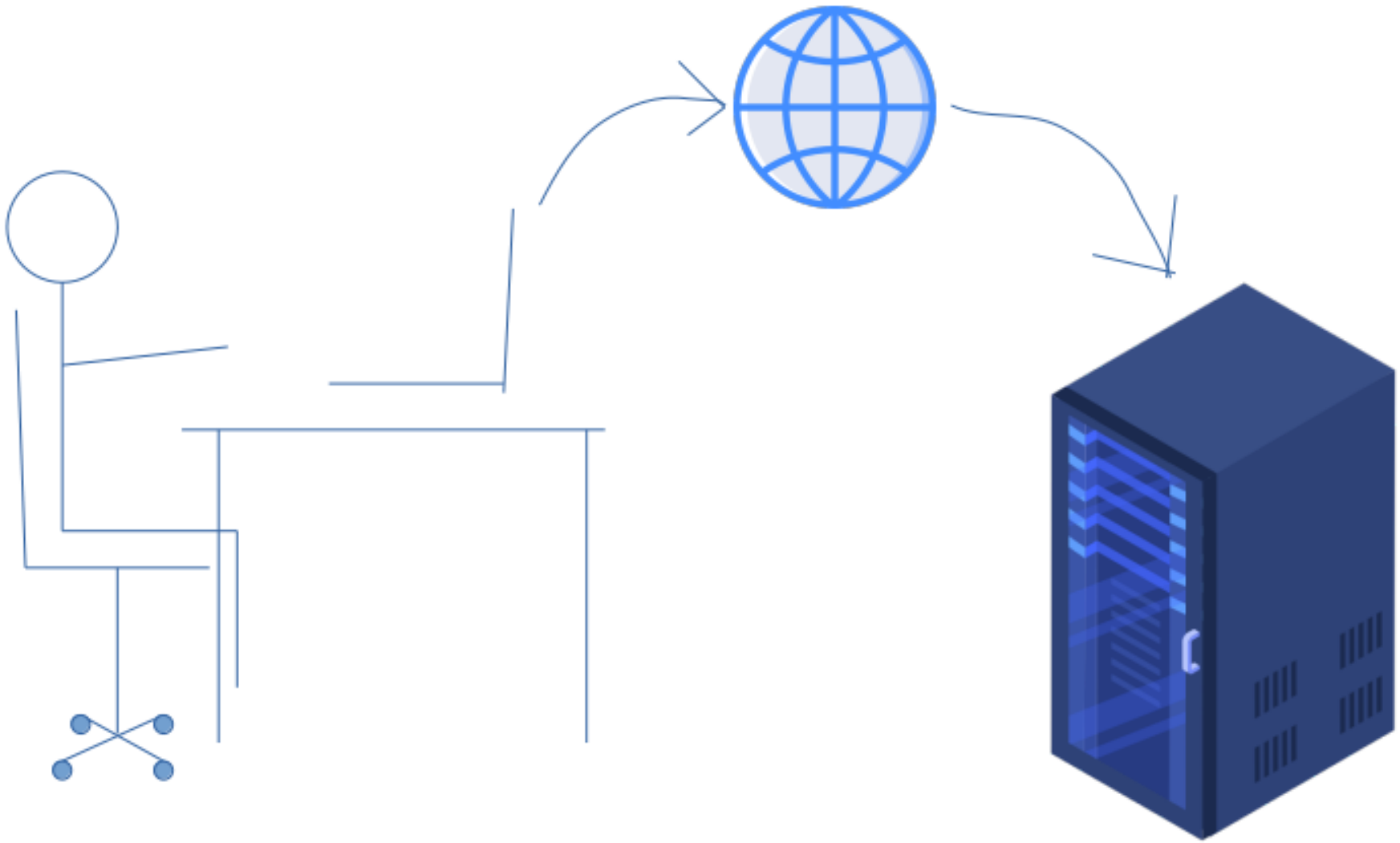
Traceable Encryption for Verifiable Receipt-free Online Voting

Quentin Yang
in collaboration with
Henri Devillez, Thomas Peters and Olivier Pereira

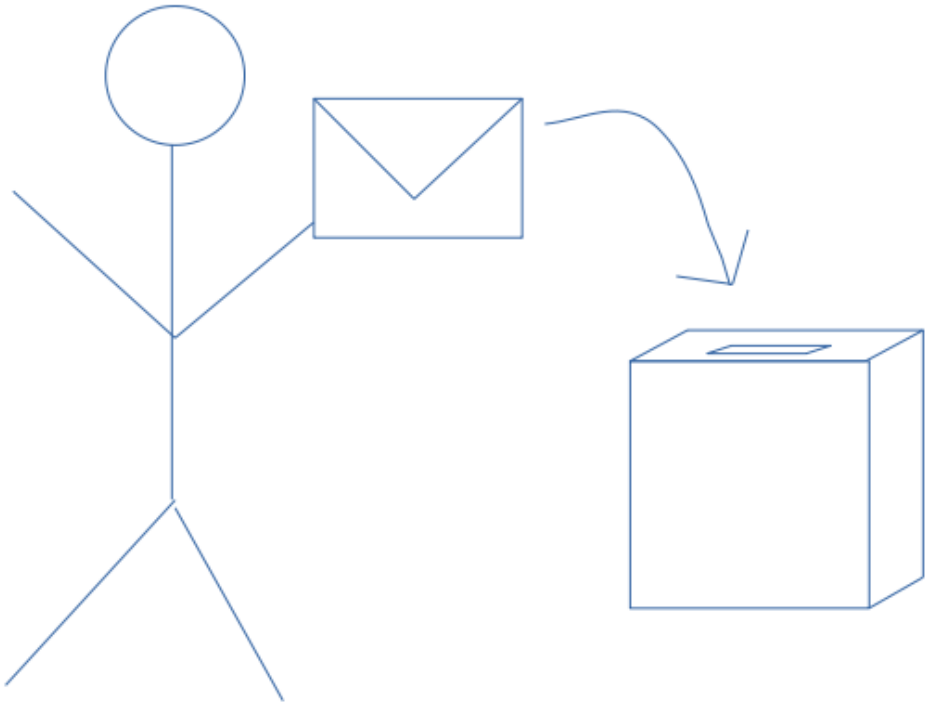
Université Catholique de Louvain

Journées au Vert, September 2022

What is electronic voting?



The security goals in electronic voting



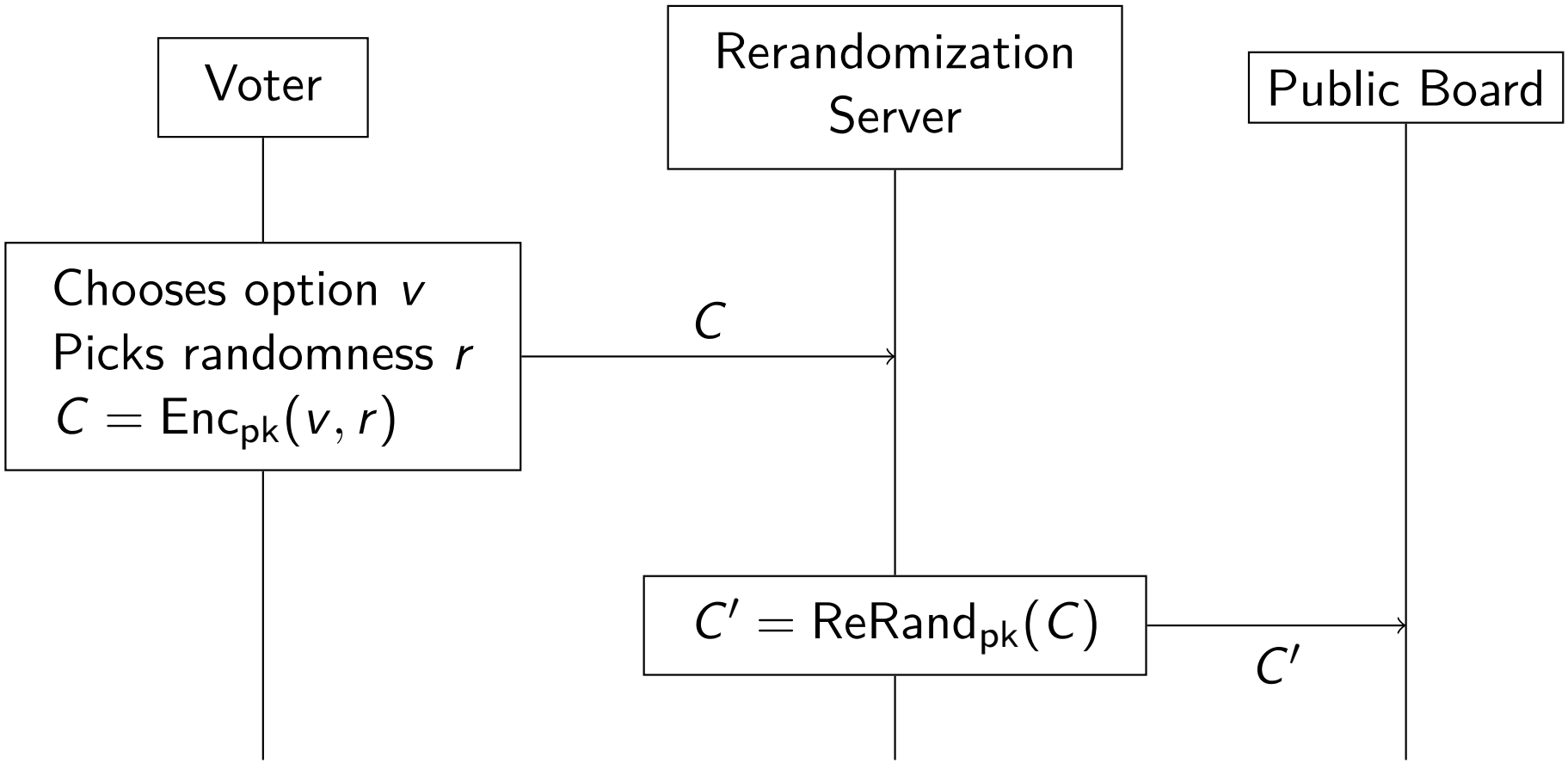
Security properties
✓Eligibility
✓Cast-as-intended
✓Vote secrecy
✓Confidence in the result

Receipt-freeness

Receipt-freeness

Receipt-freeness: a voter cannot prove how they voted.

The rerandomization paradigm



An example: **BeleniosRF**, Chaidos, Cortier, Fuchsbauer and Galindo (CCS'16)

Outline

Traceable encryption

Definition

A traceable encryption (TREnc) is a public key encryption scheme $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$, augmented with:

- $\mathsf{LGen}(\mathsf{pk})$: Generates a (random) link key lk;
- $\mathsf{LEnc}_{\mathsf{pk}}(\mathsf{lk}, m, r)$: Encrypts a message m using the link key lk and the randomness r. We have that $\mathsf{Enc}_{pk}(m, r) = \mathsf{LEnc}(\mathsf{pk}, \mathsf{LGen}(\mathsf{pk}), m, r)$;
- $\mathsf{Trace}_{\mathsf{pk}}(C)$: Outputs a trace from a ciphertext;
- $\mathbf{ReRand}_{\mathsf{pk}}(C, r)$: Rerandomizes a ciphertext C. We require that $Dec_{\mathsf{sk}}(C) = Dec_{\mathsf{sk}}(\mathsf{ReRand}_{\mathsf{pk}}(C))$;
- $\mathsf{Ver}_{\mathsf{pk}}(C)$: Verifies the validity of a ciphertext.

Traceable encryption

Definition

A traceable encryption (TREnc) is a public key encryption scheme $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$, augmented with:

- $\mathsf{LGen}(\mathsf{pk})$: Generates a (random) link key lk;
- $\mathsf{LEnc}_{\mathsf{pk}}(\mathsf{lk}, m, r)$: Encrypts a message m using the link key lk and the randomness r. We have that $\mathsf{Enc}_{pk}(m, r) = \mathsf{LEnc}(\mathsf{pk}, \mathsf{LGen}(\mathsf{pk}), m, r)$;
- $\mathbf{Trace}_{\mathsf{pk}}(C)$: Outputs a trace from a ciphertext;
- $\mathsf{ReRand}_{\mathsf{pk}}(C, r)$: Rerandomizes a ciphertext C. We require that $Dec_{\mathsf{sk}}(C) = Dec_{\mathsf{sk}}(\mathsf{ReRand}_{\mathsf{pk}}(C))$;
- $\mathsf{Ver}_{\mathsf{pk}}(C)$: Verifies the validity of a ciphertext.

Traceable encryption

Definition

A traceable encryption (TREnc) is a public key encryption scheme $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$, augmented with:

- **LGen**(pk): Generates a (random) link key lk;
- **LEnc**$_{\mathsf{pk}}(\mathsf{lk}, m, r)$: Encrypts a message m using the link key lk and the randomness r. We have that $\mathsf{Enc}_{pk}(m, r) = \mathsf{LEnc}(\mathsf{pk}, \mathsf{LGen}(\mathsf{pk}), m, r)$;
- $\mathsf{Trace}_{\mathsf{pk}}(C)$: Outputs a trace from a ciphertext;

$\vdots$

Definition (Link traceability)

A TREnc has link traceability if, for all $(\mathsf{pk}, \mathsf{lk}, m, m', r, r')$, we have

$$\mathsf{Trace}_{\mathsf{pk}}(\mathsf{LEnc}_{\mathsf{pk}}(\mathsf{lk}, m, r)) = \mathsf{Trace}_{\mathsf{pk}}(\mathsf{LEnc}_{\mathsf{pk}}(\mathsf{lk}, m', r')).$$

Traceable encryption

Definition

A traceable encryption (TREnc) is a public key encryption scheme $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$, augmented with:

$$\vdots$$

- $\mathbf{Ver}_{\mathsf{pk}}(C)$: Verifies the validity of a ciphertext.

Definition (Verifiability)

A TREnc is verifiable is no PPT adversary can produce some C such that $\mathsf{Ver}_{\mathsf{pk}}(C) = 1$ while C is not in the range of Enc.

Traceable encryption

Definition

A traceable encryption (TREnc) is a public key encryption scheme $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$, augmented with:

- $\mathsf{LGen}(\mathsf{pk})$: Generates a (random) link key lk;
- $\mathsf{LEnc}_{\mathsf{pk}}(\mathsf{lk}, m, r)$: Encrypts a message m using the link key lk and the randomness r. We have that $\mathsf{Enc}_{pk}(m, r) = \mathsf{LEnc}(\mathsf{pk}, \mathsf{LGen}(\mathsf{pk}), m, r)$;
- $\mathsf{Trace}_{\mathsf{pk}}(C)$: Outputs a trace from a ciphertext;
- $\mathsf{ReRand}_{\mathsf{pk}}(C, r)$: Rerandomizes a ciphertext C. We require that $Dec_{\mathsf{sk}}(C) = Dec_{\mathsf{sk}}(\mathsf{ReRand}_{\mathsf{pk}}(C))$;
- $\mathsf{Ver}_{\mathsf{pk}}(C)$: Verifies the validity of a ciphertext.

We also require the link traceability and verifiability properties.

Traceable encryption

Definition (Traceability)

A TREnc is traceable if, for every PPT adversary $\mathbb{A}$, the following experiment returns 1 with negligible probability.

$\mathsf{Exp}^{\mathrm{trace}}_{\mathbb{A}}(\lambda)$

$(\mathsf{pk}, \mathsf{sk}) \leftarrow\!\!\$\ \mathsf{Gen}(\lambda)$

$m \leftarrow\!\!\$\ \mathbb{A}(\mathsf{pk}, \mathsf{sk})$

$C \leftarrow\!\!\$\ \mathsf{Enc}_{\mathsf{pk}}(m)$

$C^\star \leftarrow\!\!\$\ \mathbb{A}(C)$

if $\mathsf{Trace}_{\mathsf{pk}}(C) = \mathsf{Trace}_{\mathsf{pk}}(C^\star)$ and $\mathsf{Ver}_{\mathsf{pk}}(C^\star) = 1$ and $\mathsf{Dec}_{\mathsf{sk}}(C^\star) \neq m$

then return 1

else return 0

Traceable encryption

Definition (IND-TCCA)

A TREnc is IND-TCCA secure if for every PPT adversary $\mathbb{A}$, the following experiment returns 1 with a probability negligibly close to $1/2$.

$\mathsf{Exp}^{\mathrm{tcca}}_{\mathbb{A}}(\lambda)$

$(\mathsf{pk}, \mathsf{sk}) \leftarrow\!\!\$\ \mathsf{Gen}(\lambda)$
$(C_0, C_1) \leftarrow\!\!- \ \mathbb{A}^{\mathsf{Dec}(\cdot)}(\mathsf{pk})$
$b \leftarrow\!\!\$\ \{0,1\}$
if $\mathsf{Trace}_{\mathsf{pk}}(C_0) \neq \mathsf{Trace}_{\mathsf{pk}}(C_1)$ or
$\mathsf{Ver}_{\mathsf{pk}}(C_0) = 0$ or $\mathsf{Ver}_{\mathsf{pk}}(C_1) = 0$
then return b
$C^\star \leftarrow\!\!\$\ \mathsf{ReRand}_{\mathsf{pk}}(C_b)$
$b' \leftarrow\!\!\$\ \mathbb{A}^{\mathsf{Dec}^\star(\cdot)}(C^\star)$
return $b' == b$

Outline

Application to BeleniosRF

BeleniosRF uses an encryption which is:

- ✓ Rerandomizable: ElGamal encryption.
- ✓ Verifiable: Groth-Sahai proofs.
- ✓ Traceable: Asymmetric Waters Signature.

Application to BeleniosRF

BeleniosRF uses an encryption which is:

- ✓ Rerandomizable: ElGamal encryption.
- ✓ Verifiable: Groth-Sahai proofs.
- ✓ Traceable: Asymmetric Waters Signature.

However, it comes with a few limitations:

- Restricted to small bitstrings.
- Low modularity.

Asymmetric Waters Signatures

Setting:

- Bilinear groups $(\mathbb{G}, \hat{\mathbb{G}}, \mathbb{G}_T)$ of prime order p with generators $g \in \mathbb{G}$ and $\hat{g} \in \hat{\mathbb{G}}$ and a pairing e.
- Random elements $z, u_0, \cdots, u_k \in \mathbb{G}$.
- A function $\mathcal{F} : \{0,1\}^k \longrightarrow \mathbb{G}$ such that $\mathcal{F}(m) = u_0 \prod_{i=1}^{k} u_i^{m_i}$.

Secret key: $\alpha \in \mathbb{Z}_p$ and $y = z^\alpha$.
Public key: $x = g^\alpha$ and $\hat{x} = \hat{g}^\alpha$.

Sign$(\mathbf{m}, \mathbf{s})$: $\sigma_1 = y\mathcal{F}(m)^s$, $\sigma_2 = g^s$ and $\hat{\sigma} = \hat{g}^s$.
Verify$(\mathbf{m}, \sigma)$: Check that

$$e(\sigma_1, \hat{g}) = e(z, \hat{x})e(\mathcal{F}(m), \hat{\sigma}) \text{ and } e(\sigma_2, \hat{g}) = e(\sigma_1, \hat{\sigma}).$$

BeleniosRF's limitations

Definition

A traceable encryption (TREnc) is a public key encryption scheme $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$, augmented with:

- $\mathsf{LGen}(\mathsf{pk})$: Generates a (random) link key lk;
- $\mathsf{LEnc}_{\mathsf{pk}}(\mathsf{lk}, m, r)$: Encrypts a message m using the link key lk and the randomness r. We have that $\mathsf{Enc}_{pk}(m, r) = \mathsf{LEnc}(\mathsf{pk}, \mathsf{LGen}(\mathsf{pk}), m, r)$;
- $\mathsf{Trace}_{\mathsf{pk}}(C)$: Outputs the public verification key;
- $\mathsf{ReRand}_{\mathsf{pk}}(C, r)$: Rerandomizes the ciphertext and the signature;
- $\mathsf{Ver}_{\mathsf{pk}}(C)$: Verifies the signature.

We also require the link traceability and verifiability properties.

BeleniosRF's limitations

Definition

A traceable encryption (TREnc) is a public key encryption scheme $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$, augmented with:

- $\mathsf{LGen}(\mathsf{pk})$: Generates a (random) link key lk;
- $\mathsf{LEnc}_{\mathsf{pk}}(\mathsf{lk}, m, r)$: Encrypts a message m using the link key lk and the randomness r. We have that $\mathsf{Enc}_{pk}(m, r) = \mathsf{LEnc}(\mathsf{pk}, \mathsf{LGen}(\mathsf{pk}), m, r)$;
- $\mathsf{Trace}_{\mathsf{pk}}(C)$: Outputs the public verification key;
- $\mathsf{ReRand}_{\mathsf{pk}}(C, r)$: Rerandomizes the ciphertext and the signature;
- $\mathsf{Ver}_{\mathsf{pk}}(C)$: Verifies the signature.

We also require the link traceability and verifiability properties.

Outline

Groth-Sahai proofs

Setting:

- Bilinear groups $\mathbb{G}, \hat{\mathbb{G}}, \mathbb{G}_T$ of prime order p with a pairing e.
- We rely on the SXDH assumption.

Groth-Sahai proofs are Non-Interactive Witness Indistinguishable proofs for bilinear equations, such as

$$\prod_{i=1}^{n} e(A_i, Y_i) \prod_{j=1}^{n} e(X_j, B_i) \prod_{i=1}^{n} \prod_{j=1}^{n} e(X_j, Y_i)^{\alpha_{i,j}} = t_T,$$

where red symbols represent the (secret) witnesses.

Groth-Sahai proofs

Setting:

- Bilinear groups $\mathbb{G}, \hat{\mathbb{G}}, \mathbb{G}_T$ of prime order p with a pairing e.
- We rely on the SXDH assumption.

Groth-Sahai proofs are Non-Interactive Witness Indistinguishable proofs for bilinear equations, such as

$$\prod_{i=1}^{n} e(A_i, Y_i) \prod_{j=1}^{n} e(X_j, B_i) \prod_{i=1}^{n} \prod_{j=1}^{n} e(X_j, Y_i)^{\alpha_{i,j}} = t_T,$$

where red symbols represent the (secret) witnesses.

Property: They are perfectly rerandomizable.

Asymmetric Waters Signatures

Setting:

- Bilinear groups $(\mathbb{G}, \hat{\mathbb{G}}, \mathbb{G}_T)$ of prime order p with generators $g \in \mathbb{G}$ and $\hat{g} \in \hat{\mathbb{G}}$ and a pairing e.
- Random elements $z, u_0, \cdots, u_k \in \mathbb{G}$.
- A function $\mathcal{F} : \{0,1\}^k \longrightarrow \mathbb{G}$ such that $\mathcal{F}(m) = u_0 \prod_{i=1}^{k} u_i^{m_i}$.

Secret key: $\alpha \in \mathbb{Z}_p$ and $y = z^\alpha$.
Public key: $x = g^\alpha$ and $\hat{x} = \hat{g}^\alpha$.

Sign$(\mathbf{m}, \mathbf{s})$: $\sigma_1 = y\mathcal{F}(m)^s$, $\sigma_2 = g^s$ and $\hat{\sigma} = \hat{g}^s$.
Verify$(\mathbf{m}, \sigma)$: Check that

$$e(\sigma_1, \hat{g}) = e(z, \hat{x})e(\mathcal{F}(m), \hat{\sigma}) \text{ and } e(\sigma_2, \hat{g}) = e(\sigma_1, \hat{\sigma}).$$

Groth-Sahai proofs

Setting:

- Bilinear groups $\mathbb{G}, \hat{\mathbb{G}}, \mathbb{G}_T$ of prime order p with a pairing e.
- We rely on the SXDH assumption.

Groth-Sahai proofs are Non-Interactive Witness Indistinguishable proofs for bilinear equations, such as

$$\prod_{i=1}^{n} e(A_i, Y_i) \prod_{j=1}^{n} e(X_j, B_i) \prod_{i=1}^{n} \prod_{j=1}^{n} e(X_j, Y_i)^{\alpha_{i,j}} = t_T,$$

where red symbols represent the (secret) witnesses.

Property: They are perfectly rerandomizable.

Linearly Homomorphic Structure-Preserving Signature

Setting: Bilinear groups $(\mathbb{G}, \hat{\mathbb{G}}, \mathbb{G}_T)$ of prime order p with independent generators $\hat{g}, \hat{h} \in \hat{\mathbb{G}}$ and a pairing e.
Secret key: $(\alpha_i, \beta_i)_{i=1}^n \in \mathbb{Z}_p$
Public key: $\hat{k}_i = \hat{g}^{\alpha_i} \hat{h}^{\beta_i}$ for $i = 1$ to n.

$$\textbf{Sign}(\mathbf{M_1}, \cdots, \mathbf{M_n}) : \left(Z = \prod_{i=1}^{n} M_i^{\alpha_i}, R = \prod_{i=1}^{n} M_i^{\beta_i} \right)$$

$$\textbf{Verify}(\mathbf{Z}, \mathbf{R}) : \text{Check that } e(Z, \hat{g}) e(R, \hat{h}) = \prod_{i=1}^{n} e(M_i, \hat{k}_i)$$

Outline

Security proofs

Things I have proven:

- ✓ Verifiability
- ✓ Traceability
- ✓ TCCA security

Security proofs

An example: unforgeability of the LHSP signature under SXDH.

Setting: Bilinear groups $(\mathbb{G}, \hat{\mathbb{G}}, \mathbb{G}_T)$ of prime order p with independent generators $\hat{g}, \hat{h} \in \hat{\mathbb{G}}$ and a pairing e.
Secret key: $(\alpha_i, \beta_i)_{i=1}^n \in \mathbb{Z}_p$
Public key: $\hat{k}_i = \hat{g}^{\alpha_i} \hat{h}^{\beta_i}$ for $i = 1$ to n.

$$\textbf{Sign}(\mathbf{M_1}, \cdots, \mathbf{M_n}) : \left(Z = \prod_{i=1}^{n} M_i^{\alpha_i}, R = \prod_{i=1}^{n} M_i^{\beta_i} \right)$$

$$\textbf{Verify}(\mathbf{Z}, \mathbf{R}) : \text{Check that } e(Z, \hat{g}) e(R, \hat{h}) = \prod_{i=1}^{n} e(M_i, \hat{k}_i)$$

Security proofs

An example: unforgeability of the LHSP signature under SXDH.

The adversary is given:

- $\hat{g}, \hat{h}, \hat{k}_1, \cdots, \hat{k}_n$ such that $\hat{k}_i = \hat{g}^{\alpha_i} \hat{h}^{\beta_i}$.
- Some valid signatures on messages $\overrightarrow{M}_1, \cdots, \overrightarrow{M}_k$: (Z_i, R_i) such that $Z_i = \prod_{j=1}^{n} M_{i,j}^{\alpha_j}$ and $R_i = \prod_{j=1}^{n} M_{i,j}^{\beta_j}$.
- A message $\overrightarrow{M} \notin < \overrightarrow{M}_1, \cdots, \overrightarrow{M}_k >$ to sign.

The adversary outputs Z', R' such that $e(Z', \hat{g})e(R', \hat{h}) = \prod_{i=1}^{n} e(M_i, \hat{k}_i)$.

By computing $Z = \prod_{i=1}^{n} M_i^{\alpha_i}$ and $R = \prod_{i=1}^{n} M_i^{\beta_i}$, we have

$$e(Z/Z', \hat{g})e(R/R', \hat{h}) = 1.$$

Security proofs

An example: unforgeability of the LHSP signature under SXDH.

$$e(Z/Z', \hat{g})e(R/R', \hat{h}) = 1.$$

First step: Rule out $(Z/Z', R/R') \neq (1, 1)$.

Security proofs

An example: unforgeability of the LHSP signature under SXDH.

$$e(Z/Z', \hat{g})e(R/R', \hat{h}) = 1.$$

First step: Rule out $(Z/Z', R/R') \neq (1, 1)$.

If I have a DDH challenge $(\hat{g}_1, \hat{g}_2, \hat{g}_3, \hat{g}_4)$, I use $(\hat{g}_1, \hat{g}_2)$ as $(\hat{g}, \hat{h})$.

Security proofs

An example: unforgeability of the LHSP signature under SXDH.

$$e(Z/Z', \hat{g})e(R/R', \hat{h}) = 1.$$

First step: Rule out $(Z/Z', R/R') \neq (1, 1)$.

If I have a DDH challenge $(\hat{g}_1, \hat{g}_2, \hat{g}_3, \hat{g}_4)$, I use $(\hat{g}_1, \hat{g}_2)$ as $(\hat{g}, \hat{h})$.

DDH tuple: there exists α such that

$$\begin{aligned}(\hat{g}_3, \hat{g}_4) &= (\hat{g}_1, \hat{g}_2)^{\alpha} \\ e(Z/Z', \hat{g}_3)e(R/R', \hat{g}_4) &= (e(Z/Z', \hat{g}_1)e(R/R', \hat{g}_2))^{\alpha} \\ &= 1.\end{aligned}$$

Random tuple: the pairing product is random.

Security proofs

An example: unforgeability of the LHSP signature under SXDH.

$$e(Z/Z', \hat{g})e(R/R', \hat{h}) = 1.$$

First step: Rule out $(Z/Z', R/R') \neq (1, 1)$.

Security proofs

An example: unforgeability of the LHSP signature under SXDH.

$$e(Z/Z', \hat{g})e(R/R', \hat{h}) = 1.$$

First step: Rule out $(Z/Z', R/R') \neq (1, 1)$.

Second step: Conclude.

The adversary is given:

- $\hat{g}, \hat{h}, \hat{k}_1, \cdots, \hat{k}_n$ such that $\hat{k}_i = \hat{g}^{\alpha_i}\hat{h}^{\beta_i}$.
- Some valid signatures on messages $\overrightarrow{M}_1, \cdots, \overrightarrow{M}_k$: (Z_i, R_i) such that $Z_i = \prod_{j=1}^{n} M_{i,j}^{\alpha_j}$ and $R_i = \prod_{j=1}^{n} M_{i,j}^{\beta_j}$.
- A message $\overrightarrow{M} \notin < \overrightarrow{M}_1, \cdots, \overrightarrow{M}_k >$ to sign.

The adversary outputs $Z = \prod_{i=1}^{n} M_i^{\alpha_i}$ and $R = \prod_{i=1}^{n} M_i^{\beta_i}$.

Security proofs

Things I have proven:

- ✓ Verifiability
- ✓ Traceability
- ✓ TCCA security

Security proofs

Things I have proven:

- ✓ Verifiability
- ✓ Traceability
- ✓ TCCA security
- ✓ IND-TCCA + Verifiable + "nice" tally $\Longrightarrow$ receipt-freeness

The definition of Receipt-freeness

The control flow in receipt-freeness

1. The voter votes.
2. They receive or produce a receipt.
3. They try to convince the adversary.

The control flow in receipt-freeness

1. The voter votes.
2. They receive or produce a receipt.
3. They try to convince the adversary.

1. The adversary gives some "instruction".
2. The voter votes.
3. They receive or produce a receipt.
4. They try to convince the adversary.

The definition of Receipt-freeness

The control flow in receipt-freeness

1. The voter votes.
2. They receive or produce a receipt.
3. They try to convince the adversary.

1. The adversary gives some "instruction".
2. The voter votes.
3. They receive or produce a receipt.
4. They try to convince the adversary.

The adverary could instruct to use a specific trace!

The definition of Receipt-freeness

$\mathsf{Exp}^{\beta}_{\mathcal{A},\mathcal{V},\mathcal{D}}(\lambda)$

$\mathsf{pk} \leftarrow \mathcal{O}\mathsf{init}(\lambda); \mathcal{A}^{\mathcal{O}\mathsf{cast},\mathcal{O}\mathsf{board},\mathcal{O}\mathsf{voteLR},\mathcal{O}\mathsf{receiptLR}}(\mathsf{pk}); (\mathsf{r},\Pi) \leftarrow \mathcal{O}\mathsf{tally}();$
$\mathsf{g} \leftarrow \mathcal{A}(\mathsf{r},\Pi);$ **return** $(\mathsf{g} == \beta)$

$\mathcal{O}\mathsf{init}(\lambda)$

if $\beta = 0$
  then $(\mathsf{pk},\mathsf{sk}) \leftarrow \mathsf{Setup}(\lambda)$
else $(\mathsf{pk},\mathsf{sk},\tau) \leftarrow \mathsf{SimSetup}(1^{\lambda})$
$\mathsf{PB}_0 \leftarrow \varnothing; \mathsf{PB}_1 \leftarrow \varnothing$
return pk

$\mathcal{O}\mathsf{cast}(\mathsf{b})$

if $\mathsf{Valid}(\mathsf{PB}_0,\mathsf{b}) = 0$
  or $\mathsf{Valid}(\mathsf{PB}_1,\mathsf{b}) = 0$
  then return $\perp$
else $\mathsf{Append}(\mathsf{PB}_0,\mathsf{b});$
  $\mathsf{Append}(\mathsf{PB}_1,\mathsf{b})$

$\mathcal{O}\mathsf{board}()$

return PB_{β}

$\mathcal{O}\mathsf{receiptLR}(\mathsf{I},\mathsf{v})$

$\mathsf{s}_0,\mathsf{b}_0 \leftarrow \mathsf{Vote}(\mathsf{I},\mathsf{RS},\mathsf{PB}_0)$
$\mathsf{s}_1,\mathsf{b}_1 \leftarrow \mathsf{Vote}(\mathcal{D}_{\mathsf{I}}(\mathsf{v}),\mathsf{RS},\mathsf{PB}_1)$
if $\mathsf{Valid}(\mathsf{PB}_0,\mathsf{b}_0) = 0$ **or** $v \notin \mathcal{C}$
 or $\mathsf{Valid}(\mathsf{PB}_1,\mathsf{b}_0) = 0$ **then return** $\perp$
else $\mathsf{Append}(\mathsf{PB}_0,\mathsf{b}_0); \mathsf{Append}(\mathsf{PB}_1,\mathsf{b}_1)$
return s_{β}

$\mathcal{O}\mathsf{voteLR}(\mathsf{v}_0,\mathsf{v}_1)$

if $\mathsf{v}_0 \notin \mathcal{C}$ **or** $\mathsf{v}_1 \notin \mathcal{C}$ **then return** $\perp$
$\mathsf{b}_0 \leftarrow \mathsf{Vote}(\mathsf{v}_0,\mathsf{RS},\mathsf{PB}_0)$
$\mathsf{b}_1 \leftarrow \mathsf{Vote}(\mathsf{v}_1,\mathsf{RS},\mathsf{PB}_1)$
$\mathsf{Append}(\mathsf{PB}_0,\mathsf{b}_0); \mathsf{Append}(\mathsf{PB}_1,\mathsf{b}_1)$

$\mathcal{O}\mathsf{tally}()$

$(\mathsf{r},\Pi) \leftarrow \mathsf{Tally}(\mathsf{PB}_0,\mathsf{sk})$
if $\beta = 1$ **then** $\Pi \leftarrow \mathsf{SimProof}(\mathsf{PB}_1,\mathsf{r},\tau)$
return (r,Π)

Conclusion

Our contributions:

- Formalized the notion of Traceable Encryption
- Overcame the limitations of BeleniosRF
- Took the notion of receipt-freeness to the next step
- Constructed a voting scheme and proved its security

Conclusion

Our contributions:

- Formalized the notion of Traceable Encryption
- Overcame the limitations of BeleniosRF
- Took the notion of receipt-freeness to the next step
- Constructed a voting scheme and proved its security
- Adapted the scheme for cast-as-intended verification

Our work was submited at S&P in August.

Bonus: the voting protocol