Zero-knowledge proofs and arguments in the CL framework

Séminaire Crypto, Rennes

Agathe BEAUGRAND

Joint work with G. Castagnos & F. Laguillaumie

March, 7th 2025

LIRMM

Institut de Mathématiques de Bordeaux

université de BORDEAUX

ZK in the CL framework

- CL $=$ a linearly homomorphic encryption scheme, proposed in 2015 by Castagnos & Laguillaumie
- Based on class groups of imaginary quadratic field, of which the order is hard to compute $\Rightarrow$ considered unknown
- Prove operations on the ciphertexts for applications to multiparty computation

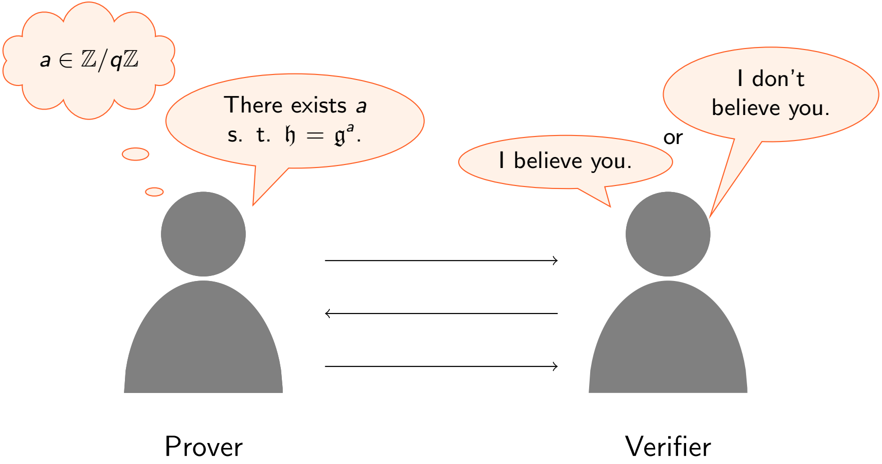
Outline

Zero-knowledge protocols

Goal

Public parameters $pp = (\mathbb{G}, \mathfrak{g}, q)$, with $\mathbb{G} = \langle \mathfrak{g} \rangle$ of order q

Statement $\mathfrak{h}$



Zero-knowledge protocols : definitions

Definition (Honest verifier zero-knowledge proof for a relation)

An *honest verifier zero-knowledge proof for* $\mathcal{R}$ is an interactive protocol between a prover and a verifier that is:

(i) *Complete:* if the prover really knows a witness, the proof is accepted.

(ii) *Sound:* a prover makes the verifier accept the proof for a false statement x only with negligible probability in λ.

(iii) *Honest verifier zero-knowledge (HVZK):* there exists a simulator, that, given a statement x, produces a transcript indistinguishable from a real accepting transcript. Sufficient to use Fiat-Shamir heuristics to obtain non interactive proofs.

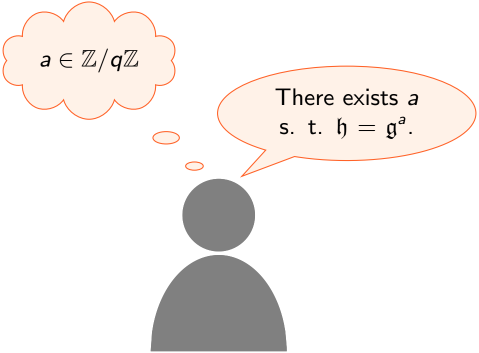
If soundness is computational, then the protocol is a HVZK *argument*.

Zero-knowledge protocols: definitions

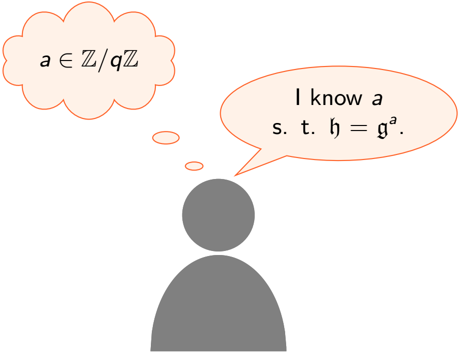
Definition (HVZK Proof of Knowledge)

Soundness $\longrightarrow$ **Knowledge Soundness:**
There exists a witness extractor that is able to compute a witness for a statement x in polynomial time, by interacting with any prover successful on x.

Notions of soundness



Prover



Prover

Soundness VS Knowledge soundness

Basic example: Schnorr protocol

Setup : $\mathbb{G} = \langle \mathfrak{g} \rangle$ group of prime order q, $\mathfrak{h} = \mathfrak{g}^a$

Prover $(\mathfrak{g}, \mathfrak{h}; a)$		Verifier $(\mathfrak{g}, \mathfrak{h})$
$\tilde{a} \xleftarrow{\$} \mathbb{Z}/q\mathbb{Z}$ $\tilde{\mathfrak{h}} \leftarrow \mathfrak{g}^{\tilde{a}}$	$\xrightarrow{\tilde{\mathfrak{h}}}$	
	$\xleftarrow{e}$	$e \xleftarrow{\$} \mathbb{Z}/q\mathbb{Z}$
$\widehat{a} = \widetilde{a} + ea \in \mathbb{Z}/q\mathbb{Z}$	$\xrightarrow{\widehat{a}}$	
		Checks if $\mathfrak{g}^{\widehat{a}} = \tilde{\mathfrak{h}} \cdot \mathfrak{h}^e$

Figure 1: Schnorr protocol for discrete logarithm

Schnorr protocol: proof

- **Completeness:** If $\mathfrak{h} = \mathfrak{g}^a$, then

$$\mathfrak{g}^{\widehat{a}} = \mathfrak{g}^{\widetilde{a}+ea} = \mathfrak{g}^{\widetilde{a}} \cdot (\mathfrak{g}^a)^e = \widetilde{\mathfrak{h}} \cdot \mathfrak{h}^e.$$

- **HV Zero-knowledge:** The simulator runs:

1. $e \xleftarrow{\$} \mathbb{Z}/q\mathbb{Z}$
2. $\widehat{a} \xleftarrow{\$} \mathbb{Z}/q\mathbb{Z}$
3. $\widetilde{\mathfrak{h}} \leftarrow \mathfrak{g}^{\widehat{a}} \cdot \mathfrak{h}^{-e}$
4. $\tau \leftarrow (\widetilde{\mathfrak{h}}, e, \widehat{a})$.

$\widehat{a} = \widetilde{a} + ea$ uniform thanks to $\widetilde{a}$ uniform $\Rightarrow$ $\widetilde{a}$ "masks" the secret a.

Schnorr protocol: proof

- **Soundness:** If the prover makes the proof accepted with proba $1/q +$ nonnegl, then there exists an algorithm (standard rewinding techniques) that extracts two accepting transcripts $\tau_1 = (\widetilde{\mathfrak{h}}, e, \widehat{a})$ and $\tau_2 = (\widetilde{\mathfrak{h}}, e', \widehat{a}')$ for $\mathfrak{h} \in \mathbb{G}$, with $e \neq e'$.

$$\begin{cases} \mathfrak{g}^{\widehat{a}} = \widetilde{\mathfrak{h}} \cdot \mathfrak{h}^{e} \\ \mathfrak{g}^{\widehat{a}'} = \widetilde{\mathfrak{h}} \cdot \mathfrak{h}^{e'} \end{cases} \Rightarrow \mathfrak{g}^{\widehat{a}-\widehat{a}'} = \mathfrak{h}^{e-e'}.$$

$e - e'$ invertible in $\mathbb{Z}/q\mathbb{Z}$ so

$$a = (\widehat{a} - \widehat{a}') \cdot (e - e')^{-1} \Rightarrow \mathfrak{g}^{a} = \mathfrak{h}.$$

$\Rightarrow$ a is a valid witness for $\mathfrak{h}$!

The case of composite order n

We now assume **$\#\mathbb{G} = n$ composite**.

- **Soundness:** There exists an algorithm that extracts two accepting transcripts $\tau_1 = (\widetilde{\mathfrak{h}}, e, \widehat{a})$ and $\tau_2 = (\widetilde{\mathfrak{h}}, e', \widehat{a}')$ for $\mathfrak{h} \in \mathbb{G}$, with $e \neq e'$.

$$\begin{cases} \mathfrak{g}^{\widehat{a}} = \widetilde{\mathfrak{h}} \cdot \mathfrak{h}^{e} \\ \mathfrak{g}^{\widehat{a}'} = \widetilde{\mathfrak{h}} \cdot \mathfrak{h}^{e'} \end{cases} \Rightarrow \mathfrak{g}^{\widehat{a}-\widehat{a}'} = \mathfrak{h}^{e-e'}.$$

$e - e'$ **not necessarily** invertible in $\mathbb{Z}/n\mathbb{Z}$... ✗

The case of composite order n

We now assume $\#\mathbb{G} = \mathbf{n}$ **composite**.

- **Soundness:** There exists an algorithm that extracts two accepting transcripts $\tau_1 = (\widetilde{\mathfrak{h}}, e, \widehat{a})$ and $\tau_2 = (\widetilde{\mathfrak{h}}, e', \widehat{a}')$ for $\mathfrak{h} \in \mathbb{G}$, with $e \neq e'$.

$$\begin{cases} \mathfrak{g}^{\widehat{a}} = \widetilde{\mathfrak{h}} \cdot \mathfrak{h}^{e} \\ \mathfrak{g}^{\widehat{a}'} = \widetilde{\mathfrak{h}} \cdot \mathfrak{h}^{e'} \end{cases} \Rightarrow \mathfrak{g}^{\widehat{a}-\widehat{a}'} = \mathfrak{h}^{e-e'}.$$

$e - e'$ **not necessarily** invertible in $\mathbb{Z}/n\mathbb{Z}$... ✗
But a wise choice of challenges might guarantee invertibility ✓

The case of composite order n

We now assume **$\#\mathbb{G} = n$ composite**.

- **Soundness:** There exists an algorithm that extracts two accepting transcripts $\tau_1 = (\widetilde{\mathfrak{h}}, e, \widehat{a})$ and $\tau_2 = (\widetilde{\mathfrak{h}}, e', \widehat{a}')$ for $\mathfrak{h} \in \mathbb{G}$, with $e \neq e'$.

$$\begin{cases} \mathfrak{g}^{\widehat{a}} = \widetilde{\mathfrak{h}} \cdot \mathfrak{h}^{e} \\ \mathfrak{g}^{\widehat{a}'} = \widetilde{\mathfrak{h}} \cdot \mathfrak{h}^{e'} \end{cases} \Rightarrow \mathfrak{g}^{\widehat{a}-\widehat{a}'} = \mathfrak{h}^{e-e'}.$$

$e - e'$ **not necessarily** invertible in $\mathbb{Z}/n\mathbb{Z}$... ✗
But a wise choice of challenges might guarantee invertibility ✓

$$a = (\widehat{a} - \widehat{a}') \cdot (e - e')^{-1} \Rightarrow \mathfrak{g}^{a} = \mathfrak{h}.$$

$\Rightarrow$ a is a valid witness for $\mathfrak{h}$!

The case of unknown order n

We now assume **$\#\mathbb{G} = n$ unknown**.

- **Soundness:** There exists an algorithm that extracts two accepting transcripts $\tau_1 = (\widetilde{\mathfrak{h}}, e, \widehat{a})$ and $\tau_2 = (\widetilde{\mathfrak{h}}, e', \widehat{a}')$ for $\mathfrak{h} \in \mathbb{G}$, with $e \neq e'$.

$$\begin{cases} \mathfrak{g}^{\widehat{a}} = \widetilde{\mathfrak{h}} \cdot \mathfrak{h}^{e} \\ \mathfrak{g}^{\widehat{a}'} = \widetilde{\mathfrak{h}} \cdot \mathfrak{h}^{e'} \end{cases} \Rightarrow \mathfrak{g}^{\widehat{a}-\widehat{a}'} = \mathfrak{h}^{e-e'}.$$

$e - e'$ **not necessarily** invertible in $\mathbb{Z}/n\mathbb{Z}$... ✗

The case of unknown order n

We now assume $\#\mathbb{G} = \mathbf{n}$ **unknown**.

- **Soundness:** There exists an algorithm that extracts two accepting transcripts $\tau_1 = (\widetilde{\mathfrak{h}}, e, \widehat{a})$ and $\tau_2 = (\widetilde{\mathfrak{h}}, e', \widehat{a}')$ for $\mathfrak{h} \in \mathbb{G}$, with $e \neq e'$.

$$\begin{cases} \mathfrak{g}^{\widehat{a}} = \widetilde{\mathfrak{h}} \cdot \mathfrak{h}^{e} \\ \mathfrak{g}^{\widehat{a}'} = \widetilde{\mathfrak{h}} \cdot \mathfrak{h}^{e'} \end{cases} \Rightarrow \mathfrak{g}^{\widehat{a} - \widehat{a}'} = \mathfrak{h}^{e-e'}.$$

$e - e'$ **not necessarily** invertible in $\mathbb{Z}/n\mathbb{Z}$... ✗
Wise choice of challenges to ensure $e - e'$ invertible:

$$a = (\widehat{a} - \widehat{a}') \cdot (e - e')^{-1} \Rightarrow \mathfrak{g}^{a} = \mathfrak{h}.$$

$\Rightarrow$ a is a valid witness for $\mathfrak{h}$!

The case of unknown order n

We now assume **$\#\mathbb{G} = n$ unknown**.

- **Soundness:** There exists an algorithm that extracts two accepting transcripts $\tau_1 = (\widetilde{\mathfrak{h}}, e, \widehat{a})$ and $\tau_2 = (\widetilde{\mathfrak{h}}, e', \widehat{a}')$ for $\mathfrak{h} \in \mathbb{G}$, with $e \neq e'$.

$$\begin{cases} \mathfrak{g}^{\widehat{a}} = \widetilde{\mathfrak{h}} \cdot \mathfrak{h}^{e} \\ \mathfrak{g}^{\widehat{a}'} = \widetilde{\mathfrak{h}} \cdot \mathfrak{h}^{e'} \end{cases} \Rightarrow \mathfrak{g}^{\widehat{a}-\widehat{a}'} = \mathfrak{h}^{e-e'}.$$

$e - e'$ **not necessarily** invertible in $\mathbb{Z}/n\mathbb{Z}$... ✗
Wise choice of challenges to ensure $e - e'$ invertible:

$$a = (\widehat{a} - \widehat{a}') \cdot (e - e')^{-1} \Rightarrow \mathfrak{g}^{a} = \mathfrak{h}.$$

$\Rightarrow$ a is a valid witness for $\mathfrak{h}$!
BUT a is not computable $\Rightarrow$ Soundness but no knowledge soundness... ✗

The case of unknown order

Setup: $\mathbb{G} = \langle \mathfrak{g} \rangle$ group of **unknown** order n, $\mathfrak{h} = \mathfrak{g}^a$, $a \in \mathbb{Z}$		
Prover $(\mathfrak{g}, \mathfrak{h}; a)$		Verifier $(\mathfrak{g}, \mathfrak{h})$
$\tilde{a} \xleftarrow{\$} [\![0, B]\!]$ $\tilde{\mathfrak{h}} \leftarrow \mathfrak{g}^{\tilde{a}}$	$\xrightarrow{\tilde{\mathfrak{h}}}$	
	$\xleftarrow{e}$	$e \xleftarrow{\$} \mathcal{C}$
$\widehat{a} = \widetilde{a} + ea \in \mathbb{Z}$	$\xrightarrow{\widehat{a}}$	
		Checks if $\mathfrak{g}^{\widehat{a}} = \tilde{\mathfrak{h}} \cdot \mathfrak{h}^e$

Figure 2: Schnorr protocol in a group of unknown order n

CL encryption scheme

Elgamal in the exponent

$\mathbb{G} = \langle g \rangle$ a DDH group of order q, we define

Algorithm 1: $\mathsf{KeyGen}_{\mathsf{EG}}$

1: $x \xleftarrow{\$} \mathbb{Z}/q\mathbb{Z}$,
2: $sk \leftarrow x$ and $pk \leftarrow g^x$
3: **return** (sk, pk)

Algorithm 2: $\mathsf{Encrypt}_{\mathsf{EG}}(pk, m)$

1: $r \xleftarrow{\$} \mathbb{Z}/q\mathbb{Z}$
2: $c_1 \leftarrow g^r$
3: $c_2 \leftarrow g^m pk^r$
4: **return** (c_1, c_2)

Algorithm 3: $\mathsf{Decrypt}_{\mathsf{EG}}((c_1, c_2), sk)$

1: $d \leftarrow c_2 c_1^{-sk}$
2: $m \leftarrow \mathsf{Solve}_{\mathsf{DL}}(d)$
3: **return** m

Theorem

Under the DDH assumption, this encryption scheme is secure against chosen-plaintext attack.

cyclic

$$G \simeq H \times F$$

F subgroup of G
cyclic of prime
order q
with easy DL

cyclic

$$G \simeq H \times F$$

F subgroup of G
cyclic of prime
order q
with easy DL

HSM assumption:
Hard to distinguish between elements of H and G

cyclic
of **unknown** order

cyclic
of **unknown** order

$$G \simeq H \times F$$

F subgroup of G
cyclic of prime
order q
with easy DL

HSM assumption:
Hard to distinguish between elements of H and G

$G = \langle g \rangle$

random

$H = \langle h \rangle$

$h = g^q$

$$G \simeq H \times F$$

$$\cap$$

$$\widehat{G} = \mathrm{Sq}\left(\mathrm{Cl}(-pq^3)\right)$$

$F = \langle f \rangle$

deterministic

$G = \langle g \rangle$

random

$H = \langle h \rangle$

$h = g^q$

$$G \simeq H \times F$$

$$\cap$$

$$\widehat{G} = \mathrm{Sq}\left(\mathrm{Cl}(-pq^3)\right)$$

$F = \langle f \rangle$

deterministic

⚠ We only know how to check $x \in \widehat{G}$ (not G)

CL encryption – scheme

Algorithm 4: $\mathsf{KeyGen}_{\mathsf{CL}}$

1: $x \xleftarrow{\$} [\![0, B[\![$,
2: $sk \leftarrow x$ and $pk \leftarrow h^x$
3: **return** (sk, pk)

Algorithm 5: $\mathsf{Encrypt}_{\mathsf{CL}}(pk, m)$

1: $r \xleftarrow{\$} [\![0, B[\![$
2: $c_1 \leftarrow h^r$
3: $c_2 \leftarrow f^m pk^r$
4: **return** (c_1, c_2)

Algorithm 6: $\mathsf{Decrypt}_{\mathsf{CL}}((c_1, c_2), sk)$

1: $d \leftarrow c_2 c_1^{-sk}$
2: $m \leftarrow \mathsf{Solve}_{\mathsf{DL}}(d)$
3: **return** m

Theorem

Under the HSM assumption, this encryption scheme is secure against chosen-plaintext attack.

Applications

- CL used for multiparty computation $\Rightarrow$ necessity to prove operations on ciphertexts (validity, homomorphic operations, shuffle...);
- MPC $\Rightarrow$ dealing with secret information and privacy $\Rightarrow$ zero-knowledge protocols
- validity ? $G \subset \widehat{G}$ of unknown order $\Rightarrow$ cannot check $c \in G^2$ $\Rightarrow$ an adversary could send invalid ciphertexts;

Application: e-voting using mixnets

Case of a referendum: the voter i chooses $m_i = 0$ (no) or $m_i = 1$ (yes), and encrypts it in $c_i = \mathrm{Enc}_{\mathrm{CL}}(m_i)$. The authority computes

$$\bigoplus_i c_i = \mathrm{Enc}_{\mathrm{CL}}\left(\sum_i m_i\right)$$

and decrypts it to count the number of yes.
But problem of anonymity $\Rightarrow$ use of mixnets.

Application: e-voting using mixnets

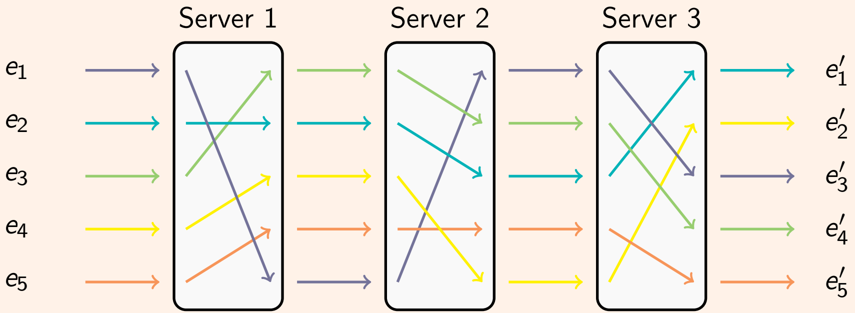


Fig. 4: A three-party mixnet

Partial extractability

A ciphertext is of the form

$$c = (c_1, c_2) = (h^r, pk^r f^m)$$

A ciphertext is of the form

Integer part:

difficult to extract

$$c = (c_1, c_2) = (h^r, pk^r f^m)$$

Part mod q:
"easier" to extract

A ciphertext is of the form

randomness:
"meaningless" part

$$c = (c_1, c_2) = (h^r, pk^r f^m)$$

message:
"meaningful" part

Partial extractability: definition

Definition

Let $\mathcal{R}$ be a relation with witness domain $\mathcal{W}_1 \times \mathcal{W}_2$. A HVZK proof for $\mathcal{R}$ has $\mathcal{W}_1$-**extractability** if there exists a witness extractor able to extract in polynomial time a partial witness $w_1 \in \mathcal{W}_1$ from any successful prover.

w_1 is a partial witness if there exists $w_2 \in \mathcal{W}_2$ such that (w_1, w_2) is a valid witness.

We denote such a proof by

$$\mathsf{HVZK} - \mathsf{PwPE}\left\{x; w_{ext} = w_1; w_2 \,\middle|\, \mathcal{R}(x, (w_1, w_2))\right\}.$$

Partial extractability: example

To prove that a CL ciphertext has the expected form, one wants to have a proof:

$$\mathsf{HVZK} - \mathsf{PoK}\left\{(c, m, r) \in \widehat{G}^2 \times \mathbb{Z}/q\mathbb{Z} \times \mathbb{Z} \,|\, c = (h^r, pk^r f^m)\right\}.$$

In many cases, it is sufficient to have a partial proof

$$\mathsf{HVZK} - \mathsf{PwPE}\left\{c; w_{ext} = m; r \,|\, c = (h^r, pk^r f^m)\right\}$$

because the goal is:

1. to guarantee c has the correct form ;
2. to guarantee that the prover actually knows the message .

.

Partial extractability: example

To prove that a CL ciphertext has the expected form, one wants to have a proof:

$$\mathsf{HVZK} - \mathsf{PoK}\left\{(c, m, r) \in \widehat{G}^2 \times \mathbb{Z}/q\mathbb{Z} \times \mathbb{Z} \,|\, c = (h^r, pk^r f^m)\right\}.$$

In many cases, it is sufficient to have a partial proof

$$\mathsf{HVZK} - \mathsf{PwPE}\left\{c; w_{ext} = m; r \,|\, c = (h^r, pk^r f^m)\right\}$$

because the goal is:

1. to guarantee c has the correct form : ✓ thanks to soundness;
2. to guarantee that the prover actually knows the message .

.

Partial extractability: example

To prove that a CL ciphertext has the expected form, one wants to have a proof:

$$\mathsf{HVZK} - \mathsf{PoK}\left\{(c, m, r) \in \widehat{G}^2 \times \mathbb{Z}/q\mathbb{Z} \times \mathbb{Z} \,|\, c = (h^r, pk^r f^m)\right\}.$$

In many cases, it is sufficient to have a partial proof

$$\mathsf{HVZK} - \mathsf{PwPE}\left\{c; w_{ext} = m; r \,|\, c = (h^r, pk^r f^m)\right\}$$

because the goal is:

1. to guarantee c has the correct form : ✓ thanks to soundness;
2. to guarantee that the prover actually knows the message : ✓ thanks to extractability.

Applications: ZK proofs in the CL framework

Example 1: Validity of a ciphertext

$pp \leftarrow \mathsf{Setup}_{\mathsf{CL}}(1^\lambda, q),\ \mathsf{pk} \in \widehat{G},\ c = (c_1, c_2) = \mathsf{Enc}_{\mathsf{CL}}(m; r)$

Prover $(h, f, c; m, r)$		Verifier (h, f, c)
$\widetilde{r} \xleftarrow{\$} [\![0, B_{\mathsf{ZK}}[\![$ $\widetilde{m} \xleftarrow{\$} \mathbb{Z}/q\mathbb{Z}$ $\widetilde{c} \leftarrow (h^{\widetilde{r}}, \mathsf{pk}^{\widetilde{r}} f^{\widetilde{m}})$	$\xrightarrow{\widetilde{c} = (\widetilde{c}_1, \widetilde{c}_2)}$	
	$\xleftarrow{e}$	$e \xleftarrow{\$} [\![0, C[\![$
$\widehat{m} = \widetilde{m} + em$ $\widehat{r} = \widetilde{r} + er$	$\xrightarrow{\widehat{m}, \widehat{r}}$	Checks if $h^{\widehat{r}} = \widetilde{c}_1 \cdot c_1^e$ $\mathsf{pk}^{\widehat{r}} \cdot f^{\widehat{m}} = \widetilde{c}_2 \cdot c_2^e$

Figure 3: HVZK-PwPE for the correctness of a ciphertext

Example 1: proof

Theorem

The protocol presented in Figure 3 is a

$$\mathsf{HVZK} - \mathsf{PwPE}\left\{c; w_{ext} = m; r \mid c = (h^r, \mathsf{pk}^r f^m)\right\}.$$

- Completeness and zero-knowledge: similar to Schnorr in a prime order group.
- Soundness: As in Schnorr, we extract two transcripts $\tau_1 = (\widetilde{c}, e, (\widehat{m}, \widehat{r}))$, $\tau_2 = (\widetilde{c}, e', (\widehat{m}', \widehat{r}'))$ with $e \neq e'$ to

$$\begin{cases} h^{\widehat{r}-\widehat{r}'} = c_1^{e-e'} \\ \mathsf{pk}^{\widehat{r}-\widehat{r}'} \cdot f^{\widehat{m}-\widehat{m}'} = c_2^{e-e'} \end{cases},$$

with $-C < e - e' < C$.

Soundness (continued)

We assume that the order of $\widehat{G}$ is C-rough (*i.e.*, it has no divisors smaller than C).
Then $e - e'$ is invertible mod $\#\widehat{G}$.
Setting $r = \delta(\widehat{r} - \widehat{r}')$ and $m = \delta(\widehat{m} - \widehat{m}')$,

$$c = (h^r, \mathrm{pk}^r \cdot f^m) = \mathrm{Enc}_{\mathrm{CL}}(m; r).$$

$\Rightarrow$ c has the correct form.

Soundness ✓

- Partial extractability: With the same computations,

$$\begin{cases} c_1 = h^{\delta(\widehat{r}-\widehat{r}')} \\ c_2 = \mathrm{pk}^{\delta(\widehat{r}-\widehat{r}')} \cdot f^{\delta(\widehat{m}-\widehat{m}')} \end{cases}$$

BUT $m, r \in \mathbb{Z}$ cannot be computed in polynomial time !
($\#\widehat{G}$ is unknown and hard to compute...)

HOWEVER, $q \mid \#\widehat{G} \Rightarrow \delta \equiv (e - e')^{-1} \mod q$
$\Rightarrow m \in \mathbb{Z}/q\mathbb{Z}$ can be computed in polynomial time from two accepting transcripts.

Partial Extractability ✓

Soundness (continued)

We assume that the order of $\widehat{G}$ is C-rough (*i.e.*, it has no divisors smaller than C). Then $e - e'$ is invertible mod $\#\widehat{G}$.
Setting $r = \delta(\widehat{r} - \widehat{r}')$ and $m = \delta(\widehat{m} - \widehat{m}')$,

$$c = (h^r, \mathrm{pk}^r \cdot f^m) = \mathrm{Enc}_{\mathrm{CL}}(m; r).$$

$\Rightarrow$ c has the correct form.

Soundness ✓

C-rough assumption

Is $\widehat{G}$ C-rough ?

In general: NO...

Cohen-Lenstra heuristics (the other CL...)

A random class groups of an imaginary quadratic field is C-rough with proba

$$\varepsilon = \prod_{p<C, p\in\mathcal{P}} \left(\prod_{i=1}^{\infty} (1 - p^{-i}) \right).$$

$+$ No way to identify the class groups that have C-rough order...

C-rough assumption

BUT

Assumption (C-rough assumption, [BDO23])

No PPT algorithm is able to distinguish between CL parameters with $\widehat{G}$ having C-rough order, and normal CL parameters.

Example 2: Batch proof for correctness of ciphertexts

$pp \leftarrow \mathsf{Setup}_{\mathsf{CL}}(1^\lambda, q)$, $\mathsf{pk} \in \widehat{G}$, $c_i = (c_{i,1}, c_{i,2}) = \mathsf{Enc}_{\mathsf{CL}}(m_i; r_i)$

Prover $(h, f, c_1, \ldots, c_n; m_1, \ldots, m_n, r_1, \ldots, r_n)$		Verifier $(h, f, c_1, \ldots, c_n)$
$\widetilde{r} \xleftarrow{\$} [\![0, B_{\mathsf{ZK},n}[\![$ $\widetilde{m} \xleftarrow{\$} \mathbb{Z}/q\mathbb{Z}$ $\widetilde{c} \leftarrow (h^{\widetilde{r}}, \mathsf{pk}^{\widetilde{r}} f^{\widetilde{m}})$	$\xrightarrow{\widetilde{c} = (\widetilde{c}_1, \widetilde{c}_2)}$	
	$\xleftarrow{\vec{e}}$	$e_1, \ldots, e_n \xleftarrow{\$} [\![0, C[\![^n$
$\widehat{m} = \widetilde{m} + \sum_{i=1}^n e_i m_i$ $\widehat{r} = \widetilde{r} + \sum_{i=1}^n e_i r_i$	$\xrightarrow{\widehat{m}, \widehat{r}}$	Checks if $h^{\widehat{r}} = \widetilde{c}_1 \cdot \prod_{i=1}^n c_{i,1}^{e_i}$ $\mathsf{pk}^{\widehat{r}} \cdot f^{\widehat{m}} = \widetilde{c}_2 \cdot \prod_{i=1}^n c_{i,2}^{e_i}$

Figure 4: HVZK-PwPE for the correctness of n ciphertexts

Example 2: Batch proof for correctness of cipheretxts

Theorem

Assuming $\widehat{G}$ has C-rough order, the protocol presented in Figure 3 is a

$$\mathsf{HVZK} - \mathsf{PwPE}\left\{c_1, \ldots, c_n; w_{ext} = \vec{m}; \vec{r} \mid \forall\, i \in [\![1, n]\!], c_i = \left(h^{r_i}, \mathsf{pk}^{r_i} f^{m_i}\right)\right\}.$$

Example 2: soundness

Let

$$\left((\widetilde{c}^{(i)}, \vec{e}^{(i,j)}, (\widehat{m}^{(i,j)}, \widehat{r}^{(i,j)}))\right)_{i\in[\![1,n]\!], j\in\{1,2\}}$$

be transcripts such that $\vec{e}^{(i,1)}$ and $\vec{e}^{(i,2)}$ differ only by their i-th component.
We have, for $i \in [\![1, n]\!], j \in \{1, 2\}$,

$$\begin{cases} h^{\widehat{r}^{(i,j)}} = \widetilde{c}_1^{(i)} \cdot \prod_{k=1}^n c_{k,1}^{e_k^{(i,j)}} \\ \mathsf{pk}^{\widehat{r}^{(i,j)}} \cdot f^{\widehat{m}^{(i,j)}} = \widetilde{c}_2^{(i)} \cdot \prod_{k=1}^n c_{k,2}^{e_k^{(i,j)}} \end{cases} \quad \text{with} \begin{cases} e_k^{(i,1)} = e_k^{(i,2)} & \text{if } k \neq i \\ e_k^{(i,1)} \neq e_k^{(i,2)} & \text{if } k = i \end{cases}.$$

So

$$\begin{cases} c_{i,1}^{e_i^{(i,1)} - e_i^{(i,2)}} = h^{\widehat{r}^{(i,1)} - \widehat{r}^{(i,2)}} \\ c_{i,2}^{e_i^{(i,1)} - e_i^{(i,2)}} = \mathsf{pk}^{\widehat{r}^{(i,1)} - \widehat{r}^{(i,2)}} \cdot f^{\widehat{m}^{(i,1)} - \widehat{m}^{(i,2)}} \end{cases}$$

Example 2: Soundness

We assume $\#\widehat{G}$ is C-rough, so that $e_i^{(i,1)} - e_i^{(i,2)}$ is invertible mod $\#\widehat{G}$, and we obtain

$$\begin{cases} c_{i,1} = h^{\delta_i(\widehat{r}^{(i,1)} - \widehat{r}^{(i,2)})} \\ c_{i,2} = \mathrm{pk}^{\delta_i(\widehat{r}^{(i,1)} - \widehat{r}^{(i,2)})} \cdot f^{\delta_i(\widehat{m}^{(i,1)} - \widehat{m}^{(i,2)})} \end{cases},$$

which gives soundness (and in a second time also partial extractability.)

Example 2: Performances

	Statement		Proof		
n	Comp. (s)	Size (MB)	Size (kB)	Prover comp.	Verifier comp.
2^9	1.4	1.7	0.634	0.011	0.092
2^{12}	2.98	13.7	0.634	0.016	0.563
2^{15}	14.95	109.7	0.635	0.049	4.469
2^{18}	110.9	877.5	0.635	0.324	36.67

Figure 5: Timings and sizes for the HVZK-PwPE for correctness of n ciphertexts of Fig. 4

Take-away

A combination of

- Partial extractability
- C-rough assumption
- (A specific transcript extractor)

allows to use efficient techniques and reduce communication for ZK proofs in the CL framework, while providing strong guarantees on messages. Similar techniques can be used for more advanced proofs, including a shuffle proof that is logarithmic in communication.

To learn some more about ZK proofs for CL:
`https://eprint.iacr.org/2024/1966` (published in *Journal of Cryptology*)

Thank you for your attention !