Non-Interactive CCA2-Secure Threshold Cryptosystems: Achieving Adaptive Security in the Standard Model Without Pairings

Julien Devevey[1] Benoît Libert[2,1] Khoa Nguyen[3] Thomas Peters[4] Moti Yung[5]

ENS de Lyon, Laboratoire LIP (U. Lyon, CNRS, ENSL, Inria, UCBL), France

CNRS, Laboratoire LIP, France

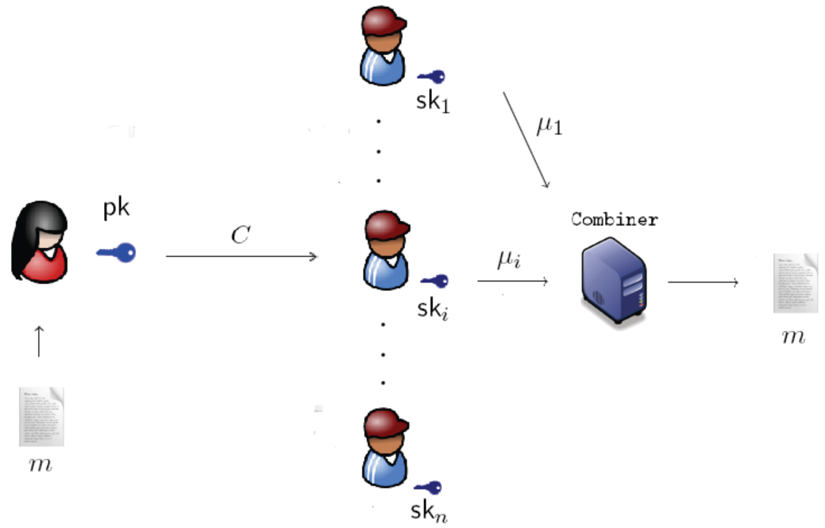
Nanyang Technological University, SPMS, Singapore

FNRS and Université catholique de Louvain, Belgium

Google and Columbia University, USA

Table of contents

Threshold Cryptography



Our goal

Build a Threshold Public-Key Encryption scheme satisfying:

- Compactness: size of C and pk independent of the number of servers,
- IND-CCA2 security, as in non-threshold PKE,
- … under adaptive corruptions: the adversary can obtain any sk_i, at any time.
- Without using pairings.

Main results and previous works

Construction	Assumption	Adaptive	IND-CCA2	Compactness
[SG98]	CDH/DDH	✗	✓ (ROM)	✓
[FP01]	DDH	✓	✓ (ROM)	✓
[BBH06]	DBDH*	✗	✓	✓
[LY12]	SXDH*	✓	✓	✓
[BGG+18]	FHE (LWE)	✗	✓	✓
This work (1)	LWE & DCR	✓	✓	✓
This work (2)	LWE	✓	✓	✓

*: In a group with pairings.

Ciphertext size:

- Construction (1): About three times the size of a Camenisch-Shoup encryption
- Construction (2): Super-polynomial modulus (but quantum-safe)

Main results and previous works

Construction	Assumption	Adaptive	IND-CCA2	Compactness
[SG98]	CDH/DDH	✗	✓ (ROM)	✓
[FP01]	DDH	✓	✓ (ROM)	✓
[BBH06]	DBDH*	✗	✓	✓
[LY12]	SXDH*	✓	✓	✓
[BGG+18]	FHE (LWE)	✗	✓	✓
This work (1)	LWE & DCR	✓	✓	✓
This work (2)	LWE	✓	✓	✓

*: In a group with pairings.

Ciphertext size:

- Construction (1): About three times the size of a Camenisch-Shoup encryption
- Construction (2): Super-polynomial modulus (but quantum-safe)

Definitions and Building Blocks

Threshold Public-Key Encryption

A compact TPKE is a 5-uple (KeyGen, Enc, PartDec, PartVerify, Combine) of algorithms that interact the following way:

Server i	Alice	Bob
	$(\mathsf{pk}, \{\mathsf{sk}_i\}) \leftarrow \mathsf{KeyGen}(1^\lambda, t)$	
$\longleftarrow \mathsf{sk}_i$		$\mathsf{pk} \longrightarrow$
		$\mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{pk}, m)$
		$\longleftarrow \mathsf{ct}$
$\longleftarrow \mathsf{ct}$		
$m_i \leftarrow \mathsf{PartDec}(\mathsf{sk}_i, \mathsf{ct})$		
$m_i \longrightarrow$		
	$\mathsf{PartVerify}(\mathsf{pk}, \mathsf{ct}, m_i)$	
	$m' \leftarrow \mathsf{Combine}(\mathsf{pk}, \{m_j\}_{j \in \mathcal{S}}, \mathsf{ct})$	

Under the condition that $|\mathsf{pk}|, |\mathsf{ct}| = \mathsf{poly}(\lambda)$.

It is correct if $\forall |\mathcal{S}| \geq t, m = m'$ with proba $\geq 1 - \mathsf{negl}(\lambda)$.

Threshold Public-Key Encryption

A compact TPKE is a 5-uple (KeyGen, Enc, PartDec, PartVerify, Combine) of algorithms that interact the following way:

Server i	Alice	Bob
	$(\mathsf{pk}, \{\mathsf{sk}_i\}) \leftarrow \mathsf{KeyGen}(1^\lambda, t)$	
$\longleftarrow \mathsf{sk}_i$		$\mathsf{pk} \longrightarrow$
		$\mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{pk}, m)$
		$\longleftarrow \mathsf{ct}$
$\longleftarrow \mathsf{ct}$		
$m_i \leftarrow \mathsf{PartDec}(\mathsf{sk}_i, \mathsf{ct})$		
$m_i \longrightarrow$		
	$\mathsf{PartVerify}(\mathsf{pk}, \mathsf{ct}, m_i)$	
	$m' \leftarrow \mathsf{Combine}(\mathsf{pk}, \{m_j\}_{j \in \mathcal{S}}, \mathsf{ct})$	

Under the condition that $|\mathsf{pk}|, |\mathsf{ct}| = \mathsf{poly}(\lambda)$.

It is correct if $\forall |\mathcal{S}| \geq t, m = m'$ with proba $\geq 1 - \mathsf{negl}(\lambda)$.

Adaptive IND-CCA2 security for TPKE

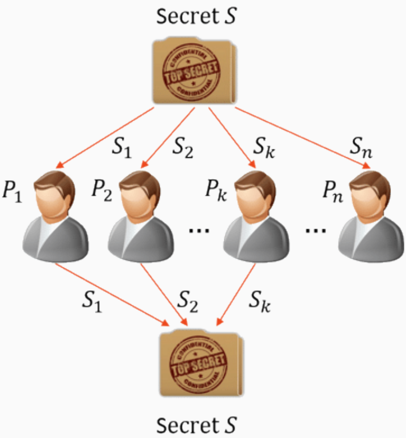
No PPT adversary $\mathcal{A}$ with a $\mathsf{PartDec}(\mathsf{sk}_i, \cdot)$ oracle for any $i \in [\ell]$ has non-negligible advantage:

$\mathcal{C}$		$\mathcal{A}$
	$\longleftarrow$	t
$(\mathsf{pk}, \{\mathsf{sk}_i\}_{i\in[N]}) \leftarrow \mathsf{KeyGen}(1^\lambda, t)$		
pk	$\longrightarrow$	
	$\longleftarrow$	(m_0, m_1)
$b \leftarrow U(\{0, 1\})$		
$c \leftarrow \mathsf{Enc}(\mathsf{pk}, m_b)$		
c	$\longrightarrow$	
	$\longleftarrow$	b'

- $\mathcal{A}$ can obtain any sk_i at any time,
- $\mathcal{A}$ can make partial decryption queries (i, c) for the challenge,

as long as it cannot trivially win. Its advantage is $|\Pr(b = b') - 1/2|$.

Building block: Linear Integer Secret Sharing



Building block: Linear Integer Secret Sharing

Monotone Access Structure

A family of sets $\mathbb{A} \subseteq \mathbf{2}^{[\ell]}$ is a monotone access structure if $\emptyset \notin \mathbb{A}$ and

$$\forall A \in \mathbb{A}, \forall B \subseteq [\ell], A \subseteq B \implies B \in \mathbb{A}.$$

The threshold family $T_{t,\ell} := \{A \subseteq [\ell], |A| \geq t\}$ is a monotone access structure.

Integer Span Program (Damgård-Thorbek; PKC'06)

For any monotone access structure $\mathbb{A}$ there exist a matrix $\mathbf{M} \in \mathbb{Z}^{d \times e}$ and a surjective map $\psi : [d] \mapsto [\ell]$ such that the following slide is true.

Building block: Linear Integer Secret Sharing

LISS `(Damgård-Thorbek; PKC'06)`

To share an integer $s \in [-2^l, 2^l]$ among parties $[\ell]$, use $\mathbf{M} \in \mathbb{Z}^{d \times e}$,

- Choose random $\rho_2, \ldots \rho_e$ and define $\vec{\rho} = (s, \rho_2, \ldots, \rho_e)^\top$
- Compute $\vec{s} = (s_1, \ldots, s_d)^\top = \mathbf{M} \cdot \vec{\rho}$
- Give s_i to party $\psi(i)$

Shares $\mathbf{s} \in \mathbb{Z}_q^m$ into $(\mathsf{sk}_1, \ldots, \mathsf{sk}_\ell) \in \mathbb{Z}_q^{d_1 \times m} \times \cdots \times \mathbb{Z}_q^{d_\ell \times m}$ such that for any $\mathcal{S}, |\mathcal{S}| \geq t$, there exist $\vec{\lambda}_i \in \{-1, 0, 1\}^{d_i}$ for $i \in \mathcal{S}$ such that:

$$\sum_{i \in \mathcal{S}} \vec{\lambda}_i^\top \cdot \mathsf{sk}_i = \mathbf{s}.$$

Building block: OTSS NIZK

A Non-Interactive Zero-Knowledge proof system for a language $\mathcal{L} = (\mathcal{L}_{\text{zk}}, \mathcal{L}_{\text{sound}})$ associated to two NP relations $(R_{\text{zk}}, R_{\text{sound}})$ is a tuple $(\text{Setup}, \text{P}, \text{V})$ of algorithms that interact the following way:

$\text{Alice}(x \in \mathcal{L}_{\text{zk}})$	$\text{Bob}((x, w) \in R_{\text{zk}})$
$\text{crs} \leftarrow \text{Setup}(1^\lambda, \mathcal{L}, \tau_{\mathcal{L}})$	
$\text{crs} \longrightarrow$	
	$\pi \leftarrow \text{P}(\text{crs}, x, w, \text{lbl})$
	$\longleftarrow \pi, \text{lbl}$
$\text{V}(\text{crs}, x, \pi, \text{lbl})$	

It is complete if V almost always outputs 1 in this case.

Properties

The proof system is zero-knowledge if there is a simulator $(\mathsf{Sim}_0, \mathsf{Sim}_1)$ such that:

$\mathcal{C}$	$\mathcal{A}$
$b \hookleftarrow U(\{0,1\})$	
$\mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda, \mathcal{L}, \tau_\mathcal{L})$ if $b = 0$	
$(\mathsf{crs}, \tau_{\mathsf{zk}}) \leftarrow \mathsf{Sim}_0(1^\lambda, \mathcal{L}, \tau_\mathcal{L})$ else	
$\mathsf{crs} \longrightarrow$	
	$\longleftarrow x, w, \mathsf{lbl}$
$\pi \leftarrow \mathsf{P}(\mathsf{crs}, x, w, \mathsf{lbl})$ if $b = 0$	
$\pi \leftarrow \mathsf{Sim}_1(\mathsf{crs}, x, \tau_{\mathsf{zk}}, \mathsf{lbl})$ else	
$\pi \longrightarrow$	
	$\longleftarrow b'$

$|\Pr(b' = b) - 1/2| = \mathsf{negl}(\lambda)$ for any ppt adversary $\mathcal{A}$.

One-Time Simulation Soundness

The proof system is One-Time Simulation Sound if the following experiment outputs 1 with negligible probability for any ppt $\mathcal{A}$:

$\mathcal{C}$		$\mathcal{A}$
$(\mathsf{crs}, \tau_{\mathsf{zk}}) \leftarrow \mathsf{Sim}_0(1^\lambda, \mathcal{L}, \tau_{\mathcal{L}})$		
	$\mathsf{crs} \longrightarrow$	
	$\longleftarrow$	(x, lbl)
$\pi \leftarrow \mathsf{Sim}_1(\mathsf{crs}, \tau_{\mathsf{zk}}, x, \mathsf{lbl})$		
	$\pi \longrightarrow$	
	$\longleftarrow$	$(x^*, \mathsf{lbl}^*, \pi^*)$
Output $\mathsf{V}(\mathsf{crs}, x^*, \pi^*, \mathsf{lbl}^*)$ if $x^* \notin \mathcal{L}_{\mathsf{sound}}$.		

ζ-Decision Composite Residuosity assumption [Pai99, DJ01]

Given $N = pq$ and $\zeta > 1$ for primes p, q. The distributions $\{x = w^{N^\zeta} \bmod N^{\zeta+1} \mid w \leftarrow U(\mathbb{Z}_N^\star)\}$ and $\{x \mid x \leftarrow U(\mathbb{Z}_{N^{\zeta+1}}^\star)\}$ are computationally indistinguishable.

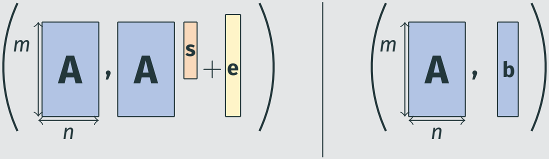
Equivalent to the 1-DCR assumption for any $\zeta > 1$ [DJ01].

Hardness assumptions

The Learning-With-Errors (LWE) problem (Regev, STOC'05)

Parameters: dimension n, number of samples $m \geq n$, modulus q.

For $\mathbf{A} \hookleftarrow \mathbb{Z}_q^{m \times n}$, $\mathbf{s} \hookleftarrow \mathbb{Z}_q^n$ and $\mathbf{e}$ a small error $\approx \alpha q$, distinguish



for uniform $\mathbf{b} \hookleftarrow \mathbb{Z}_q^m$.

Constructions

Construction from DCR+LWE: Intuition

- Pairing-free adaptation of [LY12]
- Exploits the entropy of shared secret keys "à la Cramer-Shoup"; build a DCR-based hash proof system (similar to `Camenisch-Shoup; Crypto'03`)
- Ciphertext (C_0, C_1, π) contains a simulation-sound proof that C_0 is an N^ζ-th residue in $\mathbb{Z}^*_{N^{\zeta+1}}$
- NIZK component instantiated from Fiat-Shamir and CI-hash functions (implied by LWE, cf. `Peikert-Shiehian; Crypto'19`)
- We provide a new construction of one-time simulation-sound (OT-SS) argument from DCR

Based on DCR and LWE

- KeyGen($1^\lambda, t$):
  1. Set $N = pq$, where $p, q, \frac{p-1}{2}$ and $\frac{q-1}{2} \geq 2^\lambda$ are primes, and $\zeta \geq 1$.
  2. Generate $\mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda)$ for a NIZK $\Pi^{OTSS} = (\mathsf{Setup}, \mathsf{P}, \mathsf{V})$ for $\mathcal{L}^{\mathsf{DCR}} := \{x \in \mathbb{Z}^*_{N^{\zeta+1}} \mid \exists w \in \mathbb{Z}^*_N : x = w^{N^\zeta} \bmod N^{\zeta+1}\}$.
  3. Sample $g_0 \hookleftarrow U(\mathbb{Z}^*_N)$ and set $h = g_0^{4N^\zeta \cdot x} \bmod N^{\zeta+1}$, where $x \hookleftarrow D_{\mathbb{Z},\sigma}$.
  4. LISS: key shares are $\mathsf{sk}_i = \left(\mathbf{M} \cdot \begin{pmatrix} x \\ \rho_1 \\ \vdots \\ \rho_{e-1} \end{pmatrix} \right)_{j \in \psi^{-1}(i)} \in \mathbb{Z}^{d_i}, \forall i \in [\ell]$,
     where $\rho_j \hookleftarrow D_{\mathbb{Z},\sigma}, \forall j \leq e-1$.

Output $\mathsf{pk} = (N, \zeta, g_0, h, \mathsf{crs})$ and $(\mathsf{sk}_1, \mathsf{sk}_2, \ldots, \mathsf{sk}_\ell)$.

Based on DCR and LWE

- KeyGen($1^\lambda, t$):
  1. Set $N = pq$, where $p, q, \frac{p-1}{2}$ and $\frac{q-1}{2} \geq 2^\lambda$ are primes, and $\zeta \geq 1$.
  2. Generate $\mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda)$ for a NIZK $\Pi^{OTSS} = (\mathsf{Setup}, \mathsf{P}, \mathsf{V})$ for $\mathcal{L}^{\mathsf{DCR}} := \{x \in \mathbb{Z}^*_{N^{\zeta+1}} \mid \exists w \in \mathbb{Z}^\star_N : x = w^{N^\zeta} \bmod N^{\zeta+1}\}$.
  3. Sample $g_0 \hookleftarrow U(\mathbb{Z}^*_N)$ and set $h = g_0^{4N^\zeta \cdot x} \bmod N^{\zeta+1}$, where $x \hookleftarrow D_{\mathbb{Z},\sigma}$.
  4. LISS: key shares are $\mathsf{sk}_i = \left(\mathbf{M} \cdot \begin{pmatrix} x \\ \rho_1 \\ \vdots \\ \rho_{e-1} \end{pmatrix} \right)_{j \in \psi^{-1}(i)} \in \mathbb{Z}^{d_i}, \forall i \in [\ell]$, where $\rho_j \hookleftarrow D_{\mathbb{Z},\sigma}, \forall j \leq e-1$.

Output $\mathsf{pk} = (N, \zeta, g_0, h, \mathsf{crs})$ and $(\mathsf{sk}_1, \mathsf{sk}_2, \ldots, \mathsf{sk}_\ell)$.

- $\mathsf{KeyGen}(1^\lambda, t)$:
  1. Set $N = pq$, where $p, q, \frac{p-1}{2}$ and $\frac{q-1}{2} \geq 2^\lambda$ are primes, and $\zeta \geq 1$.
  2. Generate $\mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda)$ for a NIZK $\Pi^{OTSS} = (\mathsf{Setup}, \mathsf{P}, \mathsf{V})$ for $\mathcal{L}^{\mathsf{DCR}} := \{x \in \mathbb{Z}^*_{N^{\zeta+1}} \mid \exists w \in \mathbb{Z}^\star_N : x = w^{N^\zeta} \bmod N^{\zeta+1}\}$.
  3. Sample $g_0 \hookleftarrow U(\mathbb{Z}^*_N)$ and set $h = g_0^{4N^\zeta \cdot x} \bmod N^{\zeta+1}$, where $x \hookleftarrow D_{\mathbb{Z},\sigma}$.
  4. LISS: key shares are $$\mathsf{sk}_i = \left(\mathbf{M} \cdot \begin{pmatrix} x \\ \rho_1 \\ \vdots \\ \rho_{e-1} \end{pmatrix}\right)_{j \in \psi^{-1}(i)} \in \mathbb{Z}^{d_i}, \forall i \in [\ell],$$ where $\rho_j \hookleftarrow D_{\mathbb{Z},\sigma}, \forall j \leq e-1$.

Output $\mathsf{pk} = (N, \zeta, g_0, h, \mathsf{crs})$ and $(\mathsf{sk}_1, \mathsf{sk}_2, \ldots, \mathsf{sk}_\ell)$.

Based on DCR and LWE (2)

- Encrypt(pk, Msg): To encrypt $\mathsf{Msg} \in \mathbb{Z}_{N^\zeta}$,
  1. Sample $r \hookleftarrow U(\{0, \ldots, \lfloor N/4 \rfloor\})$.
  2. Compute
  $$C_0 = {g_0}^{2N^\zeta \cdot r} \bmod N^{\zeta+1} \text{ and } C_1 = (1+N)^{\mathsf{Msg}} \cdot h^r \bmod N^{\zeta+1}.$$
  3. Compute $\vec{\pi} \leftarrow \mathsf{P}\big(\mathsf{crs}, C_0, g_0^{2r} \bmod N, \mathsf{lbl}\big)$, a proof that $C_0 \in \mathcal{L}^{\mathsf{DCR}}$ using the label $\mathsf{lbl} = C_1$.
  4. Return $\mathsf{ct} := (C_0, C_1, \vec{\pi})$.

Based on DCR and LWE (2)

- Encrypt(pk, Msg): To encrypt $\mathsf{Msg} \in \mathbb{Z}_{N^\zeta}$,
  1. Sample $r \hookleftarrow U(\{0, \ldots, \lfloor N/4 \rfloor\})$.
  2. Compute
  $$C_0 = {g_0}^{2N^\zeta \cdot r} \bmod N^{\zeta+1} \text{ and } C_1 = (1+N)^{\mathsf{Msg}} \cdot h^r \bmod N^{\zeta+1}.$$
  3. Compute $\vec{\pi} \leftarrow \mathsf{P}\big(\mathsf{crs}, C_0, g_0^{2r} \bmod N, \mathsf{lbl}\big)$, a proof that $C_0 \in \mathcal{L}^{\mathsf{DCR}}$ using the label $\mathsf{lbl} = C_1$.
  4. Return $\mathsf{ct} := (C_0, C_1, \vec{\pi})$.

Based on DCR and LWE (3)

- PartDec$(\mathsf{sk}_i, \mathsf{ct})$: To decrypt with $\mathsf{sk}_i = (s_j)_{j \in \psi^{-1}(i)}$, server i does:
  1. If $\mathsf{V}(\mathsf{crs}, C_0, \vec{\pi}, \mathsf{lbl}) = 0$, return $\perp$.
  2. For each $j \in \psi^{-1}(i) = \{j_1, \ldots, j_{d_i}\}$, compute $\mu_{i,j} = C_0^{2 \cdot s_j} \bmod N^{\zeta+1}$ and return

$$\vec{\mu}_i = (\mu_{i,j_1}, \ldots, \mu_{i,j_{d_i}}).$$

Based on DCR and LWE (3)

- PartDec$(\mathsf{sk}_i, \mathsf{ct})$: To decrypt with $\mathsf{sk}_i = (s_j)_{j \in \psi^{-1}(i)}$, server i does:
  1. If $\mathsf{V}(\mathsf{crs}, C_0, \vec{\pi}, \mathsf{lbl}) = 0$, return $\perp$.
  2. For each $j \in \psi^{-1}(i) = \{j_1, \ldots, j_{d_i}\}$, compute $\mu_{i,j} = C_0^{2 \cdot s_j} \bmod N^{\zeta+1}$ and return

$$\vec{\mu}_i = (\mu_{i,j_1}, \ldots, \mu_{i,j_{d_i}}).$$

Based on DCR and LWE (3)

- PartDec$(\mathsf{sk}_i, \mathsf{ct})$: To decrypt with $\mathsf{sk}_i = (s_j)_{j \in \psi^{-1}(i)}$, server i does:
  1. If $\mathsf{V}(\mathsf{crs}, C_0, \vec{\pi}, \mathsf{lbl}) = 0$, return $\perp$.
  2. For each $j \in \psi^{-1}(i) = \{j_1, \ldots, j_{d_i}\}$, compute $\mu_{i,j} = C_0^{2 \cdot s_j} \bmod N^{\zeta+1}$ and return

$$\vec{\mu}_i = (\mu_{i,j_1}, \ldots, \mu_{i,j_{d_i}}).$$

Based on DCR and LWE (4)

- Combine$(\mathcal{B} = (\mathcal{S}, |\mathcal{S}| \geq t, \{\vec{\mu}_i\}_{i \in \mathcal{S}}), \mathsf{ct} = (C_0, C_1, \vec{\pi}))$: Letting $\mathcal{S} = \{j_1, \ldots, j_t\}$,
  1. LISS: find a reconstruction vector $\vec{\lambda}_{\mathcal{S}} = [\vec{\lambda}_{j_1}^\top \mid \ldots \mid \vec{\lambda}_{j_t}^\top]^\top \in \{-1, 0, 1\}^{d_{\mathcal{S}}}$.
  2. LISS: compute
  $$\hat{\mu} \triangleq \prod_{i \in [t]} \prod_{k \in [d_{j_i}]} \mu_{j_i,k}^{\lambda_{j_i,k}} = C_0^{2x} \bmod N^{\zeta+1}.$$
  3. Compute $\hat{C}_1 = C_1 / \hat{\mu} \bmod N^{\zeta+1}$ and return $\perp$ if $\hat{C}_1 \not\equiv 1 \pmod{N}$. Otherwise, return $\mathsf{Msg} = (\hat{C}_1 - 1)/N \in \mathbb{Z}_{N^\zeta}$.

Based on DCR and LWE (4)

- Combine$(\mathcal{B} = (\mathcal{S}, |\mathcal{S}| \geq t, \{\vec{\mu}_i\}_{i \in \mathcal{S}}), \mathsf{ct} = (C_0, C_1, \vec{\pi}))$: Letting $\mathcal{S} = \{j_1, \ldots, j_t\}$,
  1. LISS: find a reconstruction vector $\vec{\lambda}_{\mathcal{S}} = [\vec{\lambda}_{j_1}^\top \mid \ldots \mid \vec{\lambda}_{j_t}^\top]^\top \in \{-1, 0, 1\}^{d_{\mathcal{S}}}$.
  2. LISS: compute
  $$\hat{\mu} \triangleq \prod_{i \in [t]} \prod_{k \in [d_{j_i}]} \mu_{j_i,k}^{\lambda_{j_i,k}} = C_0^{2x} \bmod N^{\zeta+1}.$$
  3. Compute $\hat{C}_1 = C_1 / \hat{\mu} \bmod N^{\zeta+1}$ and return $\perp$ if $\hat{C}_1 \not\equiv 1 \pmod{N}$. Otherwise, return $\mathsf{Msg} = (\hat{C}_1 - 1)/N \in \mathbb{Z}_{N^\zeta}$.

Security

Theorem

The scheme is CCA2 secure under adaptive corruptions, assuming that: (i) DCR holds; (ii) The NIZK argument is one-time simulation-sound.

- We give a one-time simulation sound Π^{OTSS} for $\mathcal{L}^{\mathsf{DCR}}$ under the DCR and LWE assumption.

  (shorter public parameters; improves an unbounded SS construction [LNPY20])

- Security proof exploits the entropy of secret keys (sampled from a discrete Gaussian) and the properties of a LISS (similarly to `Libert-Stehlé-Titiu; TCC'18`).

Security

Theorem

The scheme is CCA2 secure under adaptive corruptions, assuming that: (i) DCR holds; (ii) The NIZK argument is one-time simulation-sound.

- We give a one-time simulation sound Π^{OTSS} for $\mathcal{L}^{\mathsf{DCR}}$ under the DCR and LWE assumption.

  (shorter public parameters; improves an unbounded SS construction [LNPY20])

- Security proof exploits the entropy of secret keys (sampled from a discrete Gaussian) and the properties of a LISS (similarly to `Libert-Stehlé-Titiu; TCC'18`).

Security

Proof idea.

- DCR allows moving to a game that encrypts using the secret key x
- Message hidden by $x \bmod N^\zeta$
- Conditionally on $\mathcal{A}$'s view, $x \in \mathbb{Z}$ is Gaussian in a shift of $p'q' \cdot \mathbb{Z}$
  $\Rightarrow$ The distribution of $x \bmod N^\zeta$ is statistically close to $U(\mathbb{Z}_{N^\zeta})$.

Construction from LWE: Threshold Dual Regev

- Exploits the entropy of secret $\mathbf{R} \in \mathbb{Z}^{m \times L}$ conditionally on public keys $\mathbf{U} = \mathbf{A} \cdot \mathbf{R} \in \mathbb{Z}_q^{n \times L}$
- Shares each column of $\mathbf{R} \in \mathbb{Z}^{m \times L}$ using a LISS scheme
- Uses noise flooding in partial decryption shares
- Security proof follows idea from distributed PRFs (Libert-Stehlé-Titiu; TCC'18)
- Uses a simulation-sound argument that ciphertext components are of the form $(\mathbf{c}_0, \mathbf{c}_1)^\top = \mathbf{B} \cdot \mathbf{s} + \mathbf{e} \bmod q$ (Libert et al.; Asiacrypt'20)
- **Open problem:** avoid noise flooding; use a polynomial modulus while keeping compact ciphertexts

Construction from LWE: Threshold Dual Regev

- Exploits the entropy of secret $\mathbf{R} \in \mathbb{Z}^{m \times L}$ conditionally on public keys $\mathbf{U} = \mathbf{A} \cdot \mathbf{R} \in \mathbb{Z}_q^{n \times L}$
- Shares each column of $\mathbf{R} \in \mathbb{Z}^{m \times L}$ using a LISS scheme
- Uses noise flooding in partial decryption shares
- Security proof follows idea from distributed PRFs (`Libert-Stehlé-Titiu; TCC'18`)
- Uses a simulation-sound argument that ciphertext components are of the form $(\mathbf{c}_0, \mathbf{c}_1)^\top = \mathbf{B} \cdot \mathbf{s} + \mathbf{e} \bmod q$ (`Libert et al.; Asiacrypt'20`)
- **Open problem:** avoid noise flooding; use a polynomial modulus while keeping compact ciphertexts

Based on LWE solely

Lemma: Proof system [LNPT20, Section 3]

There exist one-time simulation-sound NIZK arguments $\Pi^{OTSS} = (\mathsf{Setup}, \mathsf{P}, \mathsf{V})$ for the gap language

$$\mathcal{L}_{\mathsf{zk}} = \{\mathbf{c} : \exists(\mathbf{s}, \mathbf{e}) \in \mathbb{Z}_q^{n+L} \times \mathbb{Z}^{m+L} : \|\mathbf{e}\| \leq \tilde{d} \ \wedge \ \mathbf{c} = \mathbf{B}\mathbf{s} + \mathbf{e}\}$$

$$\mathcal{L}_{\mathsf{sound}} = \{\mathbf{c} : \exists(\mathbf{s}, \mathbf{e}) \in \mathbb{Z}_q^{n+L} \times \mathbb{Z}^{m+L} : \|\mathbf{e}\| \leq \gamma\tilde{d} \ \wedge \ \mathbf{c} = \mathbf{B}\mathbf{s} + \mathbf{e}\},$$

for any matrix $\mathbf{B} \in \mathbb{Z}_q^{(m+L)\times(n+L)}$, where $m, n, L \in \mathsf{poly}(\lambda)$.

Based on LWE solely

- KeyGen($1^\lambda, t$):
  1. Set $\mathsf{pp} = \{m, n, q, p, L, \mathcal{L}_{\mathsf{LISS}}\}$, with p prime and $q = p \cdot K$. Pick two Gaussian parameters $\beta, \beta_s \in (0, 1)$.
  2. Sample $\mathbf{A} \hookleftarrow U(\mathbb{Z}_q^{n\times m})$, $\mathbf{R} \hookleftarrow D_{\mathbb{Z},\sigma}^{m\times L}$ and compute $\mathbf{U} := \mathbf{AR} \in \mathbb{Z}_q^{n\times L}$. Define $\mathsf{pk}' := (\mathbf{A}, \mathbf{U})$, $\mathsf{sk} := \mathbf{R}$.
  3. Set $\gamma, \tilde{d}$. Generate $\mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda)$ for $\mathbf{B} = \begin{bmatrix} \mathbf{A}^\top & \mathbf{0}^{m\times L} \\ \mathbf{U}^\top & K \cdot \mathbf{I}_L \end{bmatrix}$.
  4. LISS: parse $\mathbf{R}$ as $\mathbf{R} = \begin{bmatrix} \mathbf{r}_1 \mid & \mathbf{r}_2 \mid & \cdots \mid & \mathbf{r}_L \end{bmatrix} \in \mathbb{Z}^{m\times L}$. Set

     $$\mathbf{R}_\tau = \mathbf{M} \cdot [\mathbf{r}_\tau | \vec{\rho}_\tau^{\,\top}]^\top \in \mathbb{Z}^{d\times m}, \text{where } \vec{\rho}_\tau \hookleftarrow (D_{\mathbb{Z},\sigma})^{(e-1)\times m}, \forall \tau \in [L].$$

     Define the key shares as $\mathsf{sk}_i = \left\{\mathbf{R}_{\tau,\psi^{-1}(i)} \in \mathbb{Z}^{d_i \times m}\right\}_{\tau\in[L]} \forall i \in [\ell]$.

Finally, return $(\mathsf{pp}, \mathsf{pk} := (\mathsf{pk}', \mathsf{crs}), \mathsf{sk}_1, \mathsf{sk}_2, \ldots, \mathsf{sk}_\ell)$.

Based on LWE solely

- KeyGen($1^\lambda, t$):
  1. Set $\mathsf{pp} = \{m, n, q, p, L, \mathcal{L}_{\mathsf{LISS}}\}$, with p prime and $q = p \cdot K$. Pick two Gaussian parameters $\beta, \beta_{\mathsf{s}} \in (0, 1)$.
  2. Sample $\mathbf{A} \hookleftarrow U(\mathbb{Z}_q^{n\times m})$, $\mathbf{R} \hookleftarrow D_{\mathbb{Z},\sigma}^{m\times L}$ and compute $\mathbf{U} := \mathbf{A}\mathbf{R} \in \mathbb{Z}_q^{n\times L}$. Define $\mathsf{pk}' := (\mathbf{A}, \mathbf{U})$, $\mathsf{sk} := \mathbf{R}$.
  3. Set $\gamma, \tilde{d}$. Generate $\mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda)$ for $\mathbf{B} = \begin{bmatrix} \mathbf{A}^\top & \mathbf{0}^{m\times L} \\ \mathbf{U}^\top & K \cdot \mathbf{I}_L \end{bmatrix}$.
  4. LISS: parse $\mathbf{R}$ as $\mathbf{R} = \begin{bmatrix} \mathbf{r}_1 \mid & \mathbf{r}_2 \mid & \cdots \mid & \mathbf{r}_L \end{bmatrix} \in \mathbb{Z}^{m\times L}$. Set

     $$\mathbf{R}_\tau = \mathbf{M} \cdot [\mathbf{r}_\tau | \vec{\rho}_\tau^{\top}]^\top \in \mathbb{Z}^{d\times m}, \text{where } \vec{\rho}_\tau \hookleftarrow (D_{\mathbb{Z},\sigma})^{(e-1)\times m}, \forall \tau \in [L].$$

     Define the key shares as $\mathsf{sk}_i = \left\{ \mathbf{R}_{\tau,\psi^{-1}(i)} \in \mathbb{Z}^{d_i \times m} \right\}_{\tau \in [L]} \forall i \in [\ell]$.

  Finally, return $(\mathsf{pp}, \mathsf{pk} := (\mathsf{pk}', \mathsf{crs}), \mathsf{sk}_1, \mathsf{sk}_2, \ldots, \mathsf{sk}_\ell)$.

- KeyGen($1^\lambda, t$):
  1. Set $\mathsf{pp} = \{m, n, q, p, L, \mathcal{L}_{\mathsf{LISS}}\}$, with p prime and $q = p \cdot K$. Pick two Gaussian parameters $\beta, \beta_s \in (0, 1)$.
  2. Sample $\mathbf{A} \hookleftarrow U(\mathbb{Z}_q^{n\times m})$, $\mathbf{R} \hookleftarrow D_{\mathbb{Z},\sigma}^{m\times L}$ and compute $\mathbf{U} := \mathbf{AR} \in \mathbb{Z}_q^{n\times L}$. Define $\mathsf{pk}' := (\mathbf{A}, \mathbf{U})$, $\mathsf{sk} := \mathbf{R}$.
  3. Set $\gamma, \tilde{d}$. Generate $\mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda)$ for $\mathbf{B} = \begin{bmatrix} \mathbf{A}^\top & \mathbf{0}^{m\times L} \\ \mathbf{U}^\top & K \cdot \mathbf{I}_L \end{bmatrix}$.
  4. LISS: parse $\mathbf{R}$ as $\mathbf{R} = \left[\mathbf{r}_1 \mid \mathbf{r}_2 \mid \cdots \mid \mathbf{r}_L\right] \in \mathbb{Z}^{m\times L}$. Set

     $$\mathbf{R}_\tau = \mathbf{M} \cdot [\mathbf{r}_\tau | \vec{\bar{\rho}}_\tau^\top]^\top \in \mathbb{Z}^{d\times m}, \text{where } \vec{\bar{\rho}}_\tau \hookleftarrow (D_{\mathbb{Z},\sigma})^{(e-1)\times m}, \forall \tau \in [L].$$

     Define the key shares as $\mathsf{sk}_i = \left\{\mathbf{R}_{\tau,\psi^{-1}(i)} \in \mathbb{Z}^{d_i \times m}\right\}_{\tau\in[L]} \forall i \in [\ell]$.

  Finally, return $(\mathsf{pp}, \mathsf{pk} := (\mathsf{pk}', \mathsf{crs}), \mathsf{sk}_1, \mathsf{sk}_2, \ldots, \mathsf{sk}_\ell)$.

Based on LWE solely (2)

- Encrypt(pp, pk, Msg): To encrypt $\mathsf{Msg} \in \mathbb{Z}_p^L$,
  1. Sample $\mathbf{s} \hookleftarrow \mathbb{Z}_q^n$, $\mathbf{e}_0 \hookleftarrow D_{\mathbb{Z}^m, \beta q}$, $\mathbf{e}_1 \hookleftarrow D_{\mathbb{Z}^L, 2\beta \cdot \sqrt{m}\sigma \cdot q}$
  2. Compute:
     $$\mathbf{c}_0 = \mathbf{A}^\top \cdot \mathbf{s} + \mathbf{e}_0 \in \mathbb{Z}_q^m \text{ and } \mathbf{c}_1 = \mathbf{U}^\top \cdot \mathbf{s} + \mathbf{e}_1 + K \cdot \mathsf{Msg} \in \mathbb{Z}_q^L$$
     and a proof $\vec{\pi} \leftarrow \mathsf{P}\big(\mathsf{crs}, (\mathbf{c}_0^\top \mid \mathbf{c}_1^\top)^\top, (\bar{\mathbf{s}}, \bar{\mathbf{e}})\big)$ using the witnesses $\bar{\mathbf{s}} = (\mathbf{s}^\top \mid \mathsf{Msg}^\top)^\top \in \mathbb{Z}_q^{n+L}$, $\bar{\mathbf{e}} = (\mathbf{e}_0^\top \mid \mathbf{e}_1^\top)^\top \in \mathbb{Z}^{m+L}$.
  3. Return $\mathsf{ct} := (\mathbf{c}_0, \mathbf{c}_1, \vec{\pi})$.

Based on LWE solely (2)

- Encrypt(pp, pk, Msg): To encrypt $\mathsf{Msg} \in \mathbb{Z}_p^L$,
  1. Sample $\mathbf{s} \hookleftarrow \mathbb{Z}_q^n$, $\mathbf{e}_0 \hookleftarrow D_{\mathbb{Z}^m,\beta q}$, $\mathbf{e}_1 \hookleftarrow D_{\mathbb{Z}^L,2\beta\cdot\sqrt{m}\sigma\cdot q}$
  2. Compute:
     $$\mathbf{c}_0 = \mathbf{A}^\top \cdot \mathbf{s} + \mathbf{e}_0 \in \mathbb{Z}_q^m \text{ and } \mathbf{c}_1 = \mathbf{U}^\top \cdot \mathbf{s} + \mathbf{e}_1 + K \cdot \mathsf{Msg} \in \mathbb{Z}_q^L$$
     and a proof $\vec{\pi} \leftarrow \mathsf{P}\big(\mathsf{crs}, (\mathbf{c}_0^\top \mid \mathbf{c}_1^\top)^\top, (\bar{\mathbf{s}}, \bar{\mathbf{e}})\big)$ using the witnesses $\bar{\mathbf{s}} = (\mathbf{s}^\top \mid \mathsf{Msg}^\top)^\top \in \mathbb{Z}_q^{n+L}$, $\bar{\mathbf{e}} = (\mathbf{e}_0^\top \mid \mathbf{e}_1^\top)^\top \in \mathbb{Z}^{m+L}$.
  3. Return $\mathsf{ct} := (\mathbf{c}_0, \mathbf{c}_1, \vec{\pi})$.

Based on LWE solely (3)

- PartDec(pp, sk$_i$, ct): Given ct $= (\mathbf{c}_0, \mathbf{c}_1, \vec{\pi})$ and sk$_i = \{\mathbf{R}_{\tau,\psi^{-1}(i)}\}_{\tau\in[L]}$, server i does:
  1. If $V(\mathsf{crs}, (\mathbf{c}_0, \mathbf{c}_1), \vec{\pi}) = 0$, return $\perp$.
  2. Otherwise, compute $\bar{\vec{\mu}}_{i,\tau} = \mathbf{R}^\top{}_{\tau,\psi^{-1}(i)} \cdot \mathbf{c}_0 \in \mathbb{Z}_q^{d_i}, \forall \tau \in [\ell]$. Sample $\mathbf{e}'_{i,\tau} \hookleftarrow D_{\mathbb{Z}_q^{d_i}, \beta_s \cdot q}, \forall \tau \in [L]$ and return

$$\vec{\mu}_i = \{\vec{\mu}_{i,\tau}\}_{\tau\in[L]} := \{\bar{\vec{\mu}}_{i,\tau} + \mathbf{e}'_{i,\tau}\}_{\tau\in[L]}.$$

Based on LWE solely (3)

- PartDec(pp, sk_i, ct): Given $\mathsf{ct} = (\mathbf{c}_0, \mathbf{c}_1, \vec{\pi})$ and $\mathsf{sk}_i = \{\mathbf{R}_{\tau,\psi^{-1}(i)}\}_{\tau\in[L]}$, server i does:
    1. If $V(\mathsf{crs}, (\mathbf{c}_0, \mathbf{c}_1), \vec{\pi}) = 0$, return $\perp$.
    2. Otherwise, compute $\bar{\vec{\mu}}_{i,\tau} = \mathbf{R}^\top{}_{\tau,\psi^{-1}(i)} \cdot \mathbf{c}_0 \in \mathbb{Z}_q^{d_i}, \forall \tau \in [\ell]$. Sample $\mathbf{e}'_{i,\tau} \hookleftarrow D_{\mathbb{Z}_q^{d_i}, \beta_s \cdot q}, \forall \tau \in [L]$ and return

$$\vec{\mu}_i = \{\vec{\mu}_{i,\tau}\}_{\tau\in[L]} := \{\bar{\vec{\mu}}_{i,\tau} + \mathbf{e}'_{i,\tau}\}_{\tau\in[L]}.$$

Based on LWE solely (3)

- PartDec(pp, sk_i, ct): Given $\mathsf{ct} = (\mathbf{c}_0, \mathbf{c}_1, \vec{\pi})$ and $\mathsf{sk}_i = \{\mathbf{R}_{\tau,\psi^{-1}(i)}\}_{\tau\in[L]}$, server i does:
  1. If $V(\mathsf{crs}, (\mathbf{c}_0, \mathbf{c}_1), \vec{\pi}) = 0$, return $\perp$.
  2. Otherwise, compute $\bar{\vec{\mu}}_{i,\tau} = \mathbf{R}_{\tau,\psi^{-1}(i)} \cdot \mathbf{c}_0 \in \mathbb{Z}_q^{d_i}, \forall \tau \in [\ell]$. Sample $\mathbf{e'}_{i,\tau} \hookleftarrow D_{\mathbb{Z}_q^{d_i}, \beta_s \cdot q}, \forall \tau \in [L]$ and return

$$\vec{\mu}_i = \{\vec{\mu}_{i,\tau}\}_{\tau\in[L]} := \{\bar{\vec{\mu}}_{i,\tau} + \mathbf{e'}_{i,\tau}\}_{\tau\in[L]}.$$

Based on LWE solely (3)

- PartDec$(\mathsf{pp}, \mathsf{sk}_i, \mathsf{ct})$: Given $\mathsf{ct} = (\mathbf{c}_0, \mathbf{c}_1, \vec{\pi})$ and $\mathsf{sk}_i = \{\mathbf{R}_{\tau,\psi^{-1}(i)}\}_{\tau\in[L]}$, server i does:
  1. If $\mathsf{V}(\mathsf{crs}, (\mathbf{c}_0, \mathbf{c}_1), \vec{\pi}) = 0$, return $\perp$.
  2. Otherwise, compute $\bar{\vec{\mu}}_{i,\tau} = \mathbf{R}_{\tau,\psi^{-1}(i)} \cdot \mathbf{c}_0 \in \mathbb{Z}_q^{d_i}, \forall \tau \in [\ell]$. Sample $\mathbf{e}'_{i,\tau} \hookleftarrow D_{\mathbb{Z}_q^{d_i}, \beta_s \cdot q}, \forall \tau \in [L]$ and return
  $$\vec{\mu}_i = \{\vec{\mu}_{i,\tau}\}_{\tau\in[L]} := \{\bar{\vec{\mu}}_{i,\tau} + \mathbf{e}'_{i,\tau}\}_{\tau\in[L]}.$$

Based on LWE solely (4)

- Combine$(\mathsf{pp}, \mathcal{B} = (\mathcal{S}, |\mathcal{S}| \geq t, \{\vec{\mu}_i = \{\vec{\mu}_{i,\tau}\}_{\tau \in [L]}\}_{i \in \mathcal{S}}), (\mathbf{c}_0, \mathbf{c}_1))$:

1. LISS: find a reconstruction vector $\vec{\lambda}_{\mathcal{S}} = [\vec{\lambda}_{j_1}^\top \mid \ldots \mid \vec{\lambda}_{j_t}^\top]^\top \in \{-1, 0, 1\}^{d_{\mathcal{S}}}$.
2. LISS: compute

$$\vec{\mu}_\tau \triangleq \sum_{i \in \mathcal{S}} \langle \vec{\lambda}_i, \vec{\mu}_{i,\tau} \rangle = \langle \mathbf{r}_\tau, \mathbf{c}_0 \rangle + \underbrace{\sum_{i \in \mathcal{S}} \langle \vec{\lambda}_i, \mathbf{e}'_{i,\tau} \rangle}_{=:\mathbf{e}''[\tau]} \quad \forall \tau \in [L].$$

3. Compute

$$\mathbf{v} := \mathbf{c}_1 - \mathbf{R}^\top \mathbf{c}_0 - \mathbf{e}'' = K \cdot \mathsf{Msg} + \mathbf{e}_1 - \mathbf{R}^\top \mathbf{e}_0 - \mathbf{e}'' \in \mathbb{Z}_q^L.$$

4. Return $\mathsf{Msg} \in \mathbb{Z}_p^L$ s.t. $|\mathbf{v}[i] - K \cdot \mathsf{Msg}[i]|$ is minimal $\forall i \in [L]$.

Based on LWE solely (4)

- Combine$\big(\mathsf{pp}, \mathcal{B} = (\mathcal{S}, |\mathcal{S}| \geq t, \{\vec{\mu}_i = \{\vec{\mu}_{i,\tau}\}_{\tau \in [L]}\}_{i \in \mathcal{S}}), (\mathbf{c}_0, \mathbf{c}_1)\big)$:
  1. LISS: find a reconstruction vector $\vec{\lambda}_{\mathcal{S}} = [\vec{\lambda}_{j_1}^\top \mid \ldots \mid \vec{\lambda}_{j_t}^\top]^\top \in \{-1, 0, 1\}^{d_{\mathcal{S}}}$.
  2. LISS: compute
  $$\vec{\mu}_\tau \triangleq \sum_{i \in \mathcal{S}} \langle \vec{\lambda}_i, \vec{\mu}_{i,\tau} \rangle = \langle \mathbf{r}_\tau, \mathbf{c}_0 \rangle + \underbrace{\sum_{i \in \mathcal{S}} \langle \vec{\lambda}_i, \mathbf{e}'_{i,\tau} \rangle}_{=:\mathbf{e}''[\tau]} \quad \forall \tau \in [L].$$
  3. Compute
  $$\mathbf{v} := \mathbf{c}_1 - \mathbf{R}^\top \mathbf{c}_0 - \mathbf{e}'' = K \cdot \mathsf{Msg} + \mathbf{e}_1 - \mathbf{R}^\top \mathbf{e}_0 - \mathbf{e}'' \in \mathbb{Z}_q^L.$$
  4. Return $\mathsf{Msg} \in \mathbb{Z}_p^L$ s.t. $|\mathbf{v}[i] - K \cdot \mathsf{Msg}[i]|$ is minimal $\forall i \in [L]$.

Based on LWE solely (4)

- Combine$\big(\mathsf{pp}, \mathcal{B} = (\mathcal{S}, |\mathcal{S}| \geq t, \{\vec{\mu}_i = \{\vec{\mu}_{i,\tau}\}_{\tau \in [L]}\}_{i \in \mathcal{S}}), (\mathbf{c}_0, \mathbf{c}_1)\big)$:
  1. LISS: find a reconstruction vector $\vec{\lambda}_{\mathcal{S}} = [\vec{\lambda}_{j_1}^{\top} \mid \ldots \mid \vec{\lambda}_{j_t}^{\top}]^{\top} \in \{-1, 0, 1\}^{d_{\mathcal{S}}}$.
  2. LISS: compute
  $$\vec{\mu}_{\tau} \triangleq \sum_{i \in \mathcal{S}} \langle \vec{\lambda}_i, \vec{\mu}_{i,\tau} \rangle = \langle \mathbf{r}_{\tau}, \mathbf{c}_0 \rangle + \underbrace{\sum_{i \in \mathcal{S}} \langle \vec{\lambda}_i, \mathbf{e}'_{i,\tau} \rangle}_{=:\mathbf{e}''[\tau]} \quad \forall \tau \in [L].$$
  3. Compute
  $$\mathbf{v} := \mathbf{c}_1 - \mathbf{R}^{\top}\mathbf{c}_0 - \mathbf{e}'' = K \cdot \mathsf{Msg} + \mathbf{e}_1 - \mathbf{R}^{\top}\mathbf{e}_0 - \mathbf{e}'' \ \in \mathbb{Z}_q^L.$$
  4. Return $\mathsf{Msg} \in \mathbb{Z}_p^L$ s.t. $|\mathbf{v}[i] - K \cdot \mathsf{Msg}[i]|$ is minimal $\forall i \in [L]$.

Results

Properties of the scheme

The scheme is compact, correct and adaptive-CCA secure under the LWE assumption.

Did we forget something?

Detecting Corrupted Servers

Consistency

Assume that there is a public map Φ, such that $\Phi(\mathsf{PartDec}(\mathsf{sk}_i, \mathsf{ct}, \mu))$ is deterministic and Combine only needs it to recover μ. No PPT adversary $\mathcal{A}$ has non-negligible advantage in the game:

$\mathcal{C}$	$\mathcal{A}$
$\longleftarrow$	t
$(\mathsf{pk}, \{\mathsf{sk}_i\}_{i \in [N]}) \leftarrow \mathsf{KeyGen}(1^\lambda, t)$	
$(\mathsf{pk}, \mathsf{sk}_1, \ldots, \mathsf{sk}_\ell) \longrightarrow$	
$\longleftarrow$	$(\mathsf{ct}^*, \mu_i^*, i)$
Output 1 if $\Phi(\mu_i^*) \neq \Phi(\mathsf{PartDec}(\mathsf{pk}, \mathsf{sk}_i, \mathsf{ct}^*))$ and $\mathsf{PartVerify}(\mathsf{pk}, \mathsf{ct}^*, \mu_i^*) = 1$.	

The advantage of $\mathcal{A}$ is $\Pr(\mathcal{C} \text{ outputs } 1)$.

Robustness

No PPT adversary $\mathcal{A}$ has non-negligible advantage in the game:

$\mathcal{C}$	$\mathcal{A}$
$\longleftarrow$	t
$(\mathsf{pk}, \{\mathsf{sk}_i\}_{i\in[N]}) \leftarrow \mathsf{KeyGen}(1^\lambda, t)$	
$(\mathsf{pk}, \mathsf{sk}_1, \ldots, \mathsf{sk}_\ell) \longrightarrow$	
$\longleftarrow$	$(\mathsf{ct}^*, \{\mu_i^0\}_{i\in\mathcal{S}_0}, \mathcal{S}_0, \{\mu_j^1\}_{j\in\mathcal{S}_1}, \mathcal{S}_1)$
Output 1 if $\forall b \in \{0,1\}, i \in \mathcal{S}_b$ $\mathsf{PartVerify}(\mathsf{pk}, \mathsf{ct}^*, \mu_i^b) = 1$ and $\mathsf{Combine}(\mathsf{pk}, (\mathcal{S}_0, \{\mu_i^0\}), \mathsf{ct}^*)$ $\neq \mathsf{Combine}(\mathsf{pk}, (\mathcal{S}_1, \{\mu_j^1\}), \mathsf{ct}^*)$.	

The advantage of $\mathcal{A}$ is $\Pr(\mathcal{C}$ outputs $1)$.

Consistency implies robustness.

A recipe to attain consistency/robustness

- Add a commitment to the secret key shares in the public key.
- Add a proof that the decryption was done with the committed key share.
- PartVerify checks both the proof of good encryption and the proof of good decryption.
- Witness-Indistinguishability is important to keep the IND-CCA security under adaptive corruptions.

A recipe to attain consistency/robustness

- Add a commitment to the secret key shares in the public key.
- Add a proof that the decryption was done with the committed key share.
- PartVerify checks both the proof of good encryption and the proof of good decryption.
- Witness-Indistinguishability is important to keep the IND-CCA security under adaptive corruptions.

The DCR case

- The commitment is $\mathsf{vk} := \{g_0^{4N^\zeta \mathsf{sk}_{i,j}}, \forall j \in \psi^{(-1)}(i) \forall i \in [\ell]\}$
- We build a WI Σ-protocol, which can be turned into a NIWI/NIZK argument system for the language

$$\mathcal{L}_i^{\log} := \{(g_1, \{h_{i,j}, \mu_{i,j}\}_{j \in \psi^{(-1)}(i)} | \\ \forall j \in \psi^{(-1)}(i), \exists s_j \in [-B^*, B^*] : h_{i,j} = g_0^{4N^\zeta s_j} \wedge \mu_{i,j} = g_1^{2s_j}\}$$

- Use a transformation to turn it into a trapdoor Σ-protocol, due to Ciampi et al.; SCN'20
- Compile it into a NIWI/NIZK unbounded simulation-sound argument system $(\mathsf{Setup}^{\log}, \mathsf{P}_i^{\log}, \mathsf{V}_i^{\log})$

Description of the modified DCR scheme

- KeyGen$'(1^\lambda, t)$:
  1. Run $(\mathsf{pp}, \mathsf{pk}, \mathsf{sk}_1, \ldots, \mathsf{sk}_\ell) \leftarrow \mathsf{KeyGen}(1^\lambda, t)$.
  2. Generate $\mathsf{crs}^{\log} = (\mathsf{Setup}^{\log}(1^\lambda))$ the global CRS.
  3. Update the public key to

$$\mathsf{pk}' = (\mathsf{pk}, \mathsf{crs}^{\log}, \mathsf{vk} := \{g_0^{2N^\zeta \mathsf{sk}_{i,j}}\}_{(i,j)\in[\ell]\times\psi^{(-1)}(i)}).$$

  4. Return $(\mathsf{pp}, \mathsf{pk}', \mathsf{sk}_1, \ldots \mathsf{sk}_\ell)$.
- PartDec$'(\mathsf{pp}, \mathsf{sk}_i, \mathsf{ct} = (C_0, C_1, \vec{\pi}))$:
  1. Run $\vec{\mu}_i \leftarrow \mathsf{PartDec}(\mathsf{pp}, \mathsf{sk}_i, \mathsf{ct})$.
  2. Then, generate

$$\pi_i = \mathrm{P}_i^{\log}(\mathsf{crs}^{\log}, (C_0, \mathsf{vk}_{\psi^{(-1)}(i)}, \vec{\mu}_i), \mathsf{sk}_i).$$

  3. Return $\vec{\mu}_i' = (\vec{\mu}_i, \pi_i)$.

Description of the modified DCR scheme

- $\mathsf{KeyGen}'(1^\lambda, t)$:
  1. Run $(\mathsf{pp}, \mathsf{pk}, \mathsf{sk}_1, \ldots, \mathsf{sk}_\ell) \leftarrow \mathsf{KeyGen}(1^\lambda, t)$.
  2. Generate $\mathsf{crs}^{\log} = (\mathsf{Setup}^{\log}(1^\lambda))$ the global CRS.
  3. Update the public key to

$$\mathsf{pk}' = (\mathsf{pk}, \mathsf{crs}^{\log}, \mathsf{vk} := \{g_0^{2N^\zeta \mathsf{sk}_{i,j}}\}_{(i,j)\in[\ell]\times\psi^{(-1)}(i)}).$$

  4. Return $(\mathsf{pp}, \mathsf{pk}', \mathsf{sk}_1, \ldots \mathsf{sk}_\ell)$.
- $\mathsf{PartDec}'(\mathsf{pp}, \mathsf{sk}_i, \mathsf{ct} = (C_0, C_1, \vec{\pi}))$:
  1. Run $\vec{\mu}_i \leftarrow \mathsf{PartDec}(\mathsf{pp}, \mathsf{sk}_i, \mathsf{ct})$.
  2. Then, generate

$$\pi_i = \mathsf{P}_i^{\log}(\mathsf{crs}^{\log}, (C_0, \mathsf{vk}_{\psi^{(-1)}(i)}, \vec{\mu}_i), \mathsf{sk}_i).$$

  3. Return $\vec{\mu}_i' = (\vec{\mu}_i, \pi_i)$.

Description of the modified DCR scheme (2)

- $\mathsf{PartVerify}(\mathsf{pp}, \mathsf{pk}, \mathsf{ct}, \vec{\mu}'_i = (\vec{\mu}_i, \pi_i))$:
  1. Check that $\mathsf{ct} = (C_0, C_1, \vec{\pi})$ is a valid ciphertext by running $\mathsf{V}(\mathsf{crs}, (\mathbf{c}_0, \mathbf{c}_1), \vec{\pi})$. If it is not, return 0.
  2. If $\mathsf{V}_i^{\log}(\mathsf{crs}^{\log}, \vec{\mu}_i, \pi_i) = 0$, then return 0.
  3. Else, return 1.

The LWE case

- The commitment is $\mathbf{V}_{i,\tau} := \mathbf{A} \cdot \mathbf{R}_{\tau,\Psi^{(-1)}(i)} \in \mathbb{Z}_q^{n \times d_i}$
- We build a WI trapdoor Σ-protocol, which can be turned into a NIZK/NIWI argument system $(\mathsf{Setup}_i^{\mathrm{lwe}}, \mathsf{P}_i^{\mathrm{lwe}}, \mathsf{V}_i^{\mathrm{lwe}})$ for the language $\mathcal{L}_i^{\mathrm{lwe}} = (\mathcal{L}_{i,\mathsf{zk}}^{\mathrm{lwe}}, \mathcal{L}_{i,\mathsf{sound}}^{\mathrm{lwe}})$, where

$$\begin{aligned}\mathcal{L}_{i,c}^{\mathrm{lwe}} = \Big\{(\mathbf{c}_0, \mathbf{V}_{i,\tau}, \mu_{i,\tau}) | \exists \mathsf{sk}_{i,\tau} \in \mathbb{Z}^{d_i \times m}, \mathbf{V}_{i,\tau} &= \mathbf{A}\mathsf{sk}_{i,\tau} \\ \wedge \quad & \|\mu_{i,\tau}^\top - \mathbf{c}_0^\top \mathsf{sk}_{i,\tau}\| \leq B_e^c \\ \wedge \quad & \|(\mathsf{sk}_{i,\tau})_j\| \leq B_r^c, \forall j \in [m]\Big\},\end{aligned}$$

for $c \in \{\mathsf{zk}, \mathsf{sound}\}$, using the construction from `Libert et al.; Asiacrypt'20`.

Description of the modified LWE scheme

- $\mathsf{KeyGen}'(1^\lambda, t)$:
  1. Run $(\mathsf{pp}, \mathsf{pk}, \mathsf{sk}_1, \ldots, \mathsf{sk}_\ell) \leftarrow \mathsf{KeyGen}(1^\lambda, t)$.
  2. Generate $\mathsf{crs}^{\mathrm{lwe}} = (\mathsf{Setup}_i^{\mathrm{lwe}}(1^\lambda))_{i \in [\ell]}$ the global CRS.
  3. Update the public key to

$$\mathsf{pk}' = (\mathbf{A}, \mathbf{U}, \mathsf{crs}, \mathsf{crs}^{\mathrm{lwe}}, \{\mathbf{V}_{i,\tau} = \mathbf{A} \cdot \mathbf{R}_{\tau,\psi^{-1}(i)}^\top, (i,\tau) \in [\ell] \times [L]\}).$$

  4. Return $(\mathsf{pp}, \mathsf{pk}', \mathsf{sk}_1, \ldots \mathsf{sk}_\ell)$.
- $\mathsf{PartDec}'(\mathsf{pp}, \mathsf{sk}_i, \mathsf{ct} = (\mathbf{c}_0, \mathbf{c}_1, \vec{\pi}))$:
  1. Run $\vec{\mu}_i = \{\vec{\mu}_{i,\tau}\}_{\tau \in [L]} \leftarrow \mathsf{PartDec}(\mathsf{pp}, \mathsf{sk}_i, \mathsf{ct})$.
  2. Then, for each $\tau \in [L]$, generate

$$\pi_{i,\tau} = \mathsf{P}_i^{\mathrm{lwe}}(\mathsf{crs}^{\mathrm{lwe}}, \vec{\mu}_{i,\tau}, \mathbf{R}_{\tau,\psi^{-1}(i)}).$$

  3. Return $\vec{\mu}_i' = \{\vec{\mu}_{i,\tau}, \pi_{i,\tau}\}_{\tau \in [L]}$.

Description of the modified LWE scheme

- $\mathsf{KeyGen}'(1^\lambda, t)$:
  1. Run $(\mathsf{pp}, \mathsf{pk}, \mathsf{sk}_1, \ldots, \mathsf{sk}_\ell) \leftarrow \mathsf{KeyGen}(1^\lambda, t)$.
  2. Generate $\mathsf{crs}^{\mathrm{lwe}} = (\mathsf{Setup}_i^{\mathrm{lwe}}(1^\lambda))_{i \in [\ell]}$ the global CRS.
  3. Update the public key to

$$\mathsf{pk}' = (\mathbf{A}, \mathbf{U}, \mathsf{crs}, \mathsf{crs}^{\mathrm{lwe}}, \{\mathbf{V}_{i,\tau} = \mathbf{A} \cdot \mathbf{R}_{\tau,\psi^{-1}(i)}^\top, (i, \tau) \in [\ell] \times [L]\}).$$

  4. Return $(\mathsf{pp}, \mathsf{pk}', \mathsf{sk}_1, \ldots \mathsf{sk}_\ell)$.
- $\mathsf{PartDec}'(\mathsf{pp}, \mathsf{sk}_i, \mathsf{ct} = (\mathbf{c}_0, \mathbf{c}_1, \vec{\pi}))$:
  1. Run $\vec{\mu}_i = \{\vec{\mu}_{i,\tau}\}_{\tau \in [L]} \leftarrow \mathsf{PartDec}(\mathsf{pp}, \mathsf{sk}_i, \mathsf{ct})$.
  2. Then, for each $\tau \in [L]$, generate

$$\pi_{i,\tau} = \mathsf{P}_i^{\mathrm{lwe}}(\mathsf{crs}^{\mathrm{lwe}}, \vec{\mu}_{i,\tau}, \mathbf{R}_{\tau,\psi^{-1}(i)}).$$

  3. Return $\vec{\mu}_i' = \{\vec{\mu}_{i,\tau}, \pi_{i,\tau}\}_{\tau \in [L]}$.

Description of the modified LWE scheme (2)

- $\mathsf{PartVerify}(\mathsf{pp}, \mathsf{pk}, \mathsf{ct}, \vec{\mu}_i' = \{\vec{\mu}_{i,\tau}, \pi_{i,\tau}\}_{\tau \in [L]})$:
  1. Check that $\mathsf{ct} = (\mathbf{c}_0, \mathbf{c}_1, \vec{\pi})$ is a valid ciphertext by running $\mathsf{V}(\mathsf{crs}, (\mathbf{c}_0, \mathbf{c}_1), \vec{\pi})$. If it is not, return 0.
  2. For every $\tau \in [L]$, if $\mathsf{V}_{i,\tau}^{\mathrm{lwe}}(\mathsf{crs}^{\mathrm{lwe}}, \vec{\mu}_{i,\tau}, \pi_{i,\tau}) = 0$, then return 0.
  3. Else, return 1.

Properties

Security of the modified DCR-based scheme

The modified scheme is IND-CCA secure under adaptive corruptions and consistent.

Security of the modified LWE-based scheme

The modified scheme is IND-CCA secure under adaptive corruptions and robust.

Properties

Security of the modified DCR-based scheme

The modified scheme is IND-CCA secure under adaptive corruptions and consistent.

Security of the modified LWE-based scheme

The modified scheme is IND-CCA secure under adaptive corruptions and robust.

Thank you for your attention!