LOW COMMUNICATION THRESHOLD FULLY HOMOMORPHIC ENCRYPTION

Alain Passelègue & Damien Stehlé

Kolkata --- December 13, 2024

Eprint 2024/1984

HEAAN CRYPTO LAB

* K. Boudgoust, P. Scholl: Simple threshold (fully *homomorphic encryption) from LWE with polynomial modulus*. ASIACRYPT'23

MAIN RESULTS

Contribution #1

Cryptanalysis of the BS23* Threshold-FHE with **moderate** decryption modulus

⇒ All known (general-purpose) Threshold-FHE's need **exponential** decr. modulus

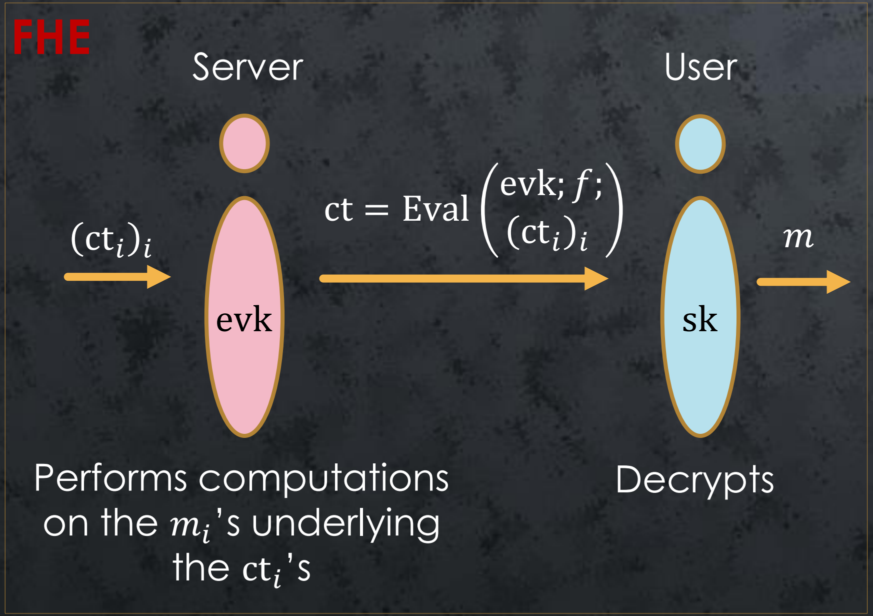
Contribution #2

Construction of a Threshold-FHE with **tiny** decryption modulus…

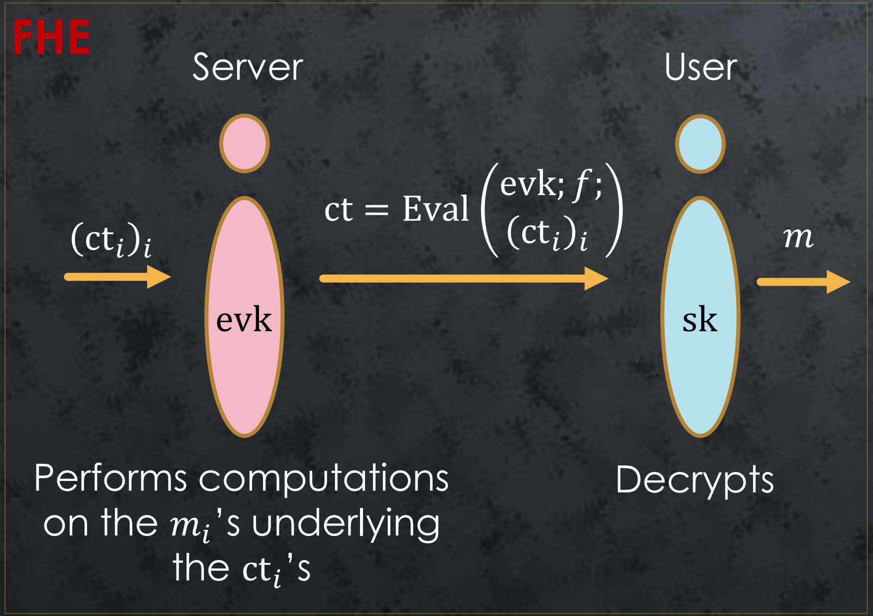
… for the (specific) case where the computing party is not corrupted.

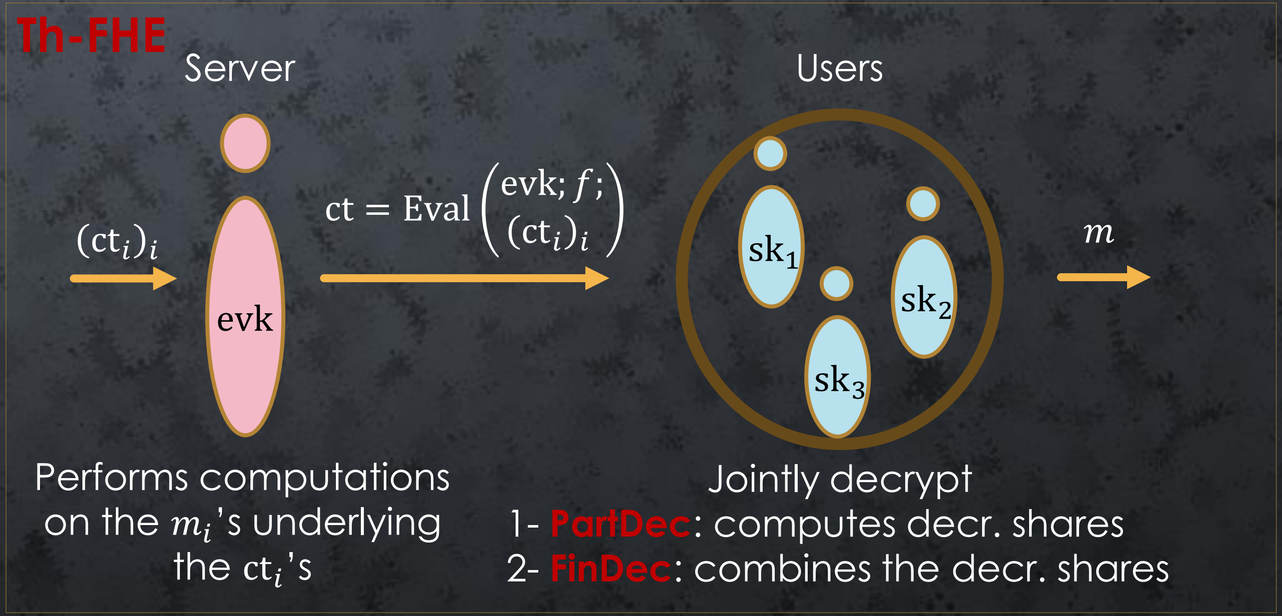
Disclaimer: we only look at the N-out-of-N case

THRESHOLD-FHE



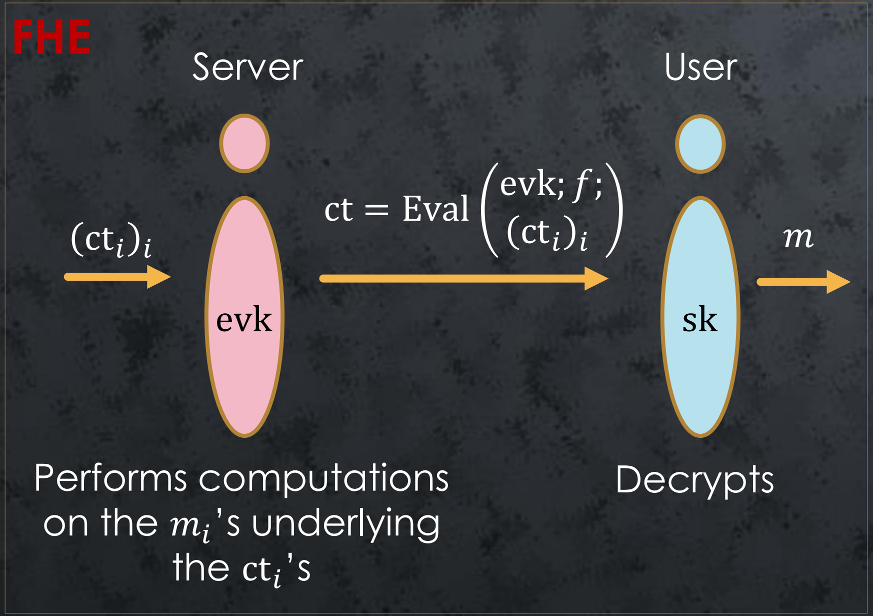
THRESHOLD-FHE

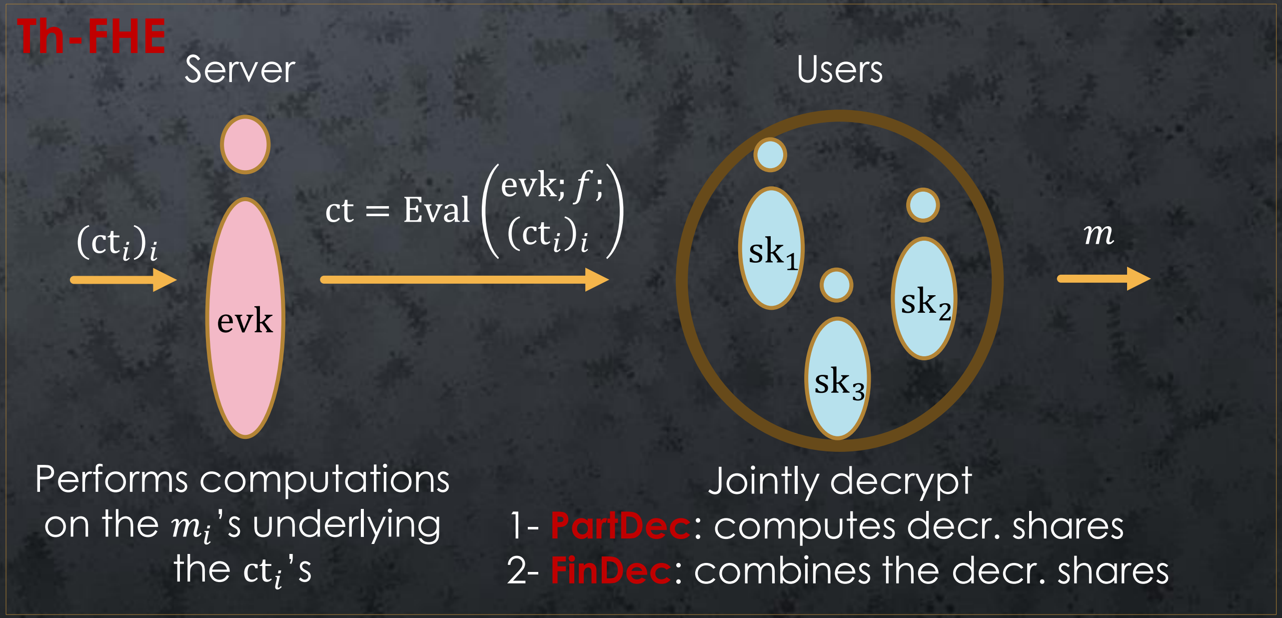




THRESHOLD-FHE

* G. Asharov, A. Jain, A. López-Alt, E. Tromer, V. Vaikuntanathan, D. Wichs. Multiparty computation with low communication, computation and interaction via threshold FHE. EUROCRYPT'12
** D. Boneh, R. Gennaro, S. Goldfeder, A. Jain, S. Kim, P. Rasmussen, A. Sahai. Threshold cryptosystems from threshold fully homomorphic encryption. CRYPTO'18





Th-FHE: sk is shared between users
- Protects sk
- Enables secure multi-party computations*
- Allows to thresholdize cryptographic constructions**

* B. Li, D. Micciancio. On the security of homomorphic encryption on approximate numbers. EUROCRYPT'21

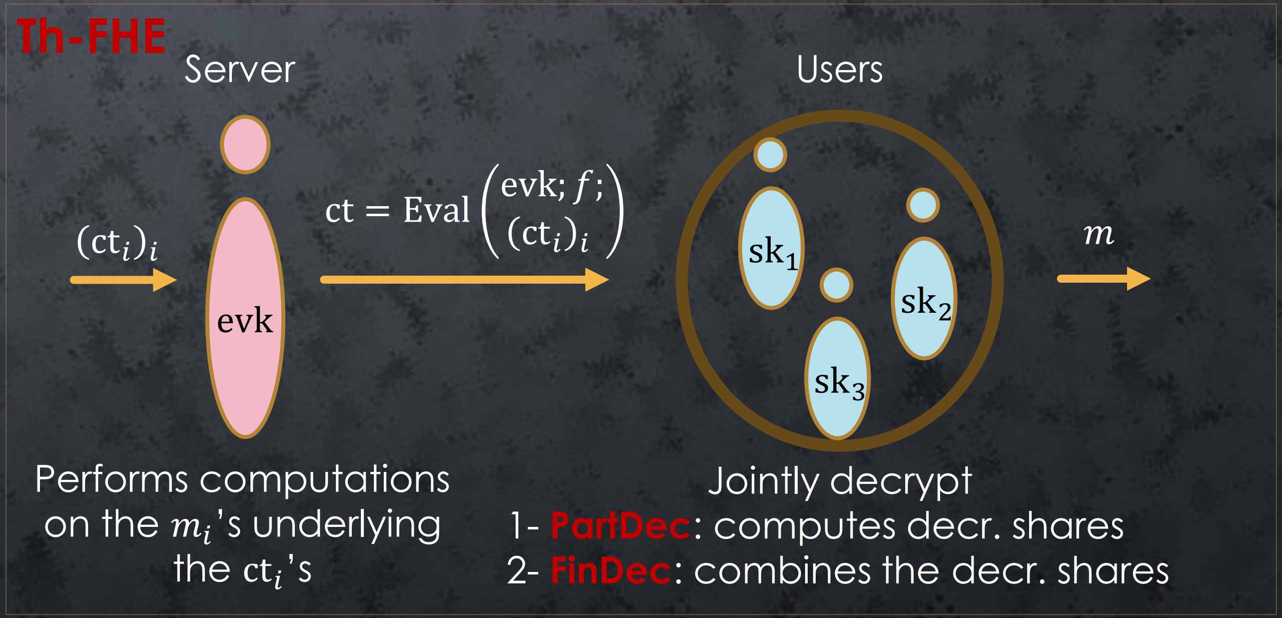
THRESHOLD-FHE: SECURITY (INFORMAL)

Adversary can:

- corrupt $N-1$ users
- request encr. of ptxts
- request evaluations on generated ctxts
- request decr. of any generated ctxt
  (unless its ptxt trivially solves the challenge)

Adervsary's challenge: distinguish between the encryptions of two ptxts of its choice

(we actually consider simulation-based security)



Side note: for $N=1$ user, this matches the IND-CPA-D security notion from LM21*.

THRESHOLD-FHE: GENERAL DESIGN

Start from an FHE scheme, with ciphertexts of the form:

$$\mathrm{ct} = (a, b): \ a \cdot \mathrm{sk} + b = \mathrm{Ecd}(m) + e \ [q]$$

- a, b can be over $\mathbb{Z}_q$ (LWE) or R_q (Ring-LWE), and e is small
- Ecd can be most/least/… significant bits

Split the key as

$$\mathrm{sk} = \sum \mathrm{sk}_i$$

PartDec $(\mathrm{ct}, \mathrm{sk}_i)$

$$\mathrm{sh}_i := a \cdot \mathrm{sk}_i + e_i$$

FinDec $(\mathrm{ct}, (\mathrm{sh}_i)_i)$

$$\mathrm{Dcd}\left(\sum \mathrm{sh}_i + b\right)$$

* M. Dahl, D. Demmler, S. El Kazdadi, A. Meyre, J.-B. Orfila, D. Rotaru, N. Smart, S. Tap, M. Walter. Noah's ark: Efficient threshold-FHE using noise flooding. WAHC'23

HOW LARGE SHOULD THE FLOODING BE?

$$\mathrm{ct} = (a, b):\ \ a \cdot \mathrm{sk} + b = \mathrm{Ecd}(m) + e\ \ [q]$$

Split the key as

$$\mathrm{sk} = \sum \mathrm{sk}_i$$

PartDec $(\mathrm{ct}, \mathrm{sk}_i)$

$$\mathrm{sh}_i \ \coloneqq\ a \cdot \mathrm{sk}_i + e_i$$

FinDec $(\mathrm{ct}, (\mathrm{sh}_i)_i)$

$$\mathrm{Dcd}\left(\sum \mathrm{sh}_i + b\right)$$

The size of e_i drives the choice of q during decryption
⇒ drives amount of communication in decryption

(if need be, we can switch to a large q just before decryption*)

* M. Dahl, D. Demmler, S. El Kazdadi, A. Meyre, J.-B. Orfila, D. Rotaru, N. Smart, S. Tap, M. Walter. Noah's ark: Efficient threshold-FHE using noise flooding. WAHC'23

HOW LARGE SHOULD THE FLOODING BE?

$$\mathrm{ct} = (a, b):\ \ a \cdot \mathrm{sk} + b = \mathrm{Ecd}(m) + e\ \ [q]$$

Split the key as

$$\mathrm{sk} = \sum \mathrm{sk}_i$$

PartDec $(\mathrm{ct}, \mathrm{sk}_i)$

$$\mathrm{sh}_i \ \coloneqq\ a \cdot \mathrm{sk}_i + e_i$$

FinDec $(\mathrm{ct}, (\mathrm{sh}_i)_i)$

$$\mathrm{Dcd}\ (\textstyle\sum \mathrm{sh}_i + b)$$

The size of e_i drives the choice of q during decryption
⇒ drives amount of communication in decryption
(if need be, we can switch to a large q just before decryption*)

No e_i: $e_i = 0$
Adversary can recover sk_i

Exponential e_i: $|e_i| \geq 2^{\lambda} \cdot |e|$
We can simulate the adversary's view

HOW LARGE SHOULD THE FLOODING BE?

$$\mathrm{ct} = (a, b)\colon\ a \cdot \mathrm{sk} + b = \mathrm{Ecd}(m) + e\ \ [q]$$

Split the key as

$$\mathrm{sk} = \sum \mathrm{sk}_i$$

PartDec $(\mathrm{ct}, \mathrm{sk}_i)$

$$\mathrm{sh}_i \ \coloneqq\ a \cdot \mathrm{sk}_i + e_i$$

FinDec $(\mathrm{ct}, (\mathrm{sh}_i)_i)$

$$\mathrm{Dcd}\ (\textstyle\sum \mathrm{sh}_i + b)$$

No e_i: $e_i = 0$
Adversary can recover sk_i

Very small e_i: $|e_i| \approx \mathrm{poly}(\lambda) \cdot |e|$
Adversary can "average-out" e_i in sh_i

Exponential e_i: $|e_i| \geq 2^{\lambda} \cdot |e|$
We can simulate the adversary's view

What about moderate e_i?

$$|e_i| \approx \mathrm{poly}(Q_{dec}) \cdot |e|$$

(Q_{dec} is the number of decr. queries)

* S. Agrawal, D. Stehlé, A. Yadav. Round-optimal lattice-based threshold signatures, revisited. ICALP'22
** B. Li, D. Micciancio, M. Schultz, J. Sorrell. Securing approximate homomorphic encryption using differential privacy. CRYPTO'22

CAN MODERATE FLOODING WORK?

$$\mathrm{ct} = (a, b)\text{:}\;\; a \cdot \mathrm{sk} + b = \mathrm{Ecd}(m) + e \;\; [q]$$

Split the key as

$$\mathrm{sk} = \sum \mathrm{sk}_i$$

PartDec $(\mathrm{ct}, \mathrm{sk}_i)$

$$\mathrm{sh}_i \;\coloneqq\; a \cdot \mathrm{sk}_i + e_i$$
$$|e_i| \approx \mathrm{poly}(Q_{dec}) \cdot |e|$$

FinDec $(\mathrm{ct}, (\mathrm{sh}_i)_i)$

$$\mathrm{Dcd}\left(\sum \mathrm{sh}_i + b\right)$$

ASY22*: When used to thresholdize a signature scheme, this can be proved secure using Rényi Divergence

LMSS22** attack, e.g. using BFV:

- Encrypt $(10,10)$ or $(0,0)$
- Perform an inner product with $(1, -1)$
- In both cases, the result is 0
- But the noise is larger for $(10,10)$

$\Rightarrow$ Gives a $\mathrm{poly}\left(\frac{1}{Q_{dec}}\right)$ dist. advantage

** K. Boudgoust, P. Scholl: Simple threshold (fully *homomorphic encryption) from LWE with polynomial modulus*. ASIACRYPT'23

THE BS23 APPROACH

$$\mathrm{ct} = (a, b)\colon\ a \cdot \mathrm{sk} + b = \mathrm{Ecd}(m) + e\ \ [q]$$

Split the key as

$$\mathrm{sk} = \sum \mathrm{sk}_i$$

PartDec $(\mathrm{ct}, \mathrm{sk}_i)$

$$\mathrm{sh}_i \ \coloneqq\ a \cdot \mathrm{sk}_i + e_i$$
$$|e_i| \approx \mathrm{poly}(Q_{dec}) \cdot |e|$$

FinDec $(\mathrm{ct}, (\mathrm{sh}_i)_i)$

$$\mathrm{Dcd}\ (\textstyle\sum \mathrm{sh}_i + b)$$

ASY22: When used to thresholdize a signature scheme, this can be proved secure using Rényi Divergence

BS23* (informal): Assuming the FHE scheme is **circuit-private**, then the threshold FHE scheme is secure with moderate noise

LMSS22: The decryption noise may carry **information on past computations**, including the challenge plaintexts

Circuit privacy: the distribution of the decryption noise does not depend on past computations, even if sk is given to the adversary

CONTRIBUTION #1: CRYPTANALYSIS OF BS23

We are given $\mathrm{ct}^* = (a^*, b^*)$: $a^* \cdot \mathrm{sk} + b^* = \mathrm{Ecd}(m_\beta) + e \ \ [q]$
We want to distinguish $\beta = 0$ from $\beta = 1$

Assumption 1: the scheme allows "Rescale"

$$\mathrm{ct} = (a, b):\ a \cdot \mathrm{sk} + b = \mathrm{Ecd}(m) + e \ \ [q]$$

$$\downarrow$$

$$\mathrm{ct}' = (a', b') = \left(\left\lfloor \frac{q'}{q} a \right\rceil, \left\lfloor \frac{q'}{q} b \right\rceil\right) \ \ [q'] \ \text{ with } q' \ll q$$

We have

$$a' \cdot \mathrm{sk} + b' \approx \mathrm{Ecd}(m) + e_{rnd} \cdot \mathrm{sk}\,,$$

where $e_{rnd} = \left\{\frac{q'}{q} a\right\}$ is **known**

CONTRIBUTION #1: CRYPTANALYSIS OF BS23

We are given $\mathrm{ct}^* = (a^*, b^*)$: $a^* \cdot sk + b^* = \mathrm{Ecd}(m_\beta) + e \;\; [q]$
We want to distinguish $\beta = 0$ from $\beta = 1$

Assumption 1: the scheme allows "Rescale"

$$\mathrm{ct} = (a, b):\;\; a \cdot \mathrm{sk} + b = \mathrm{Ecd}(m) + e \;\; [q]$$

$$\downarrow$$

$$\mathrm{ct}' = (a', b') = \left(\left\lfloor \frac{q'}{q} a \right\rceil, \left\lfloor \frac{q'}{q} b \right\rceil \right) \quad [q'] \;\; \text{with } q' \ll q$$

We have

$$a' \cdot \mathrm{sk} + b' \approx \mathrm{Ecd}(m) + e_{rnd} \cdot \mathrm{sk}\,,$$

where $e_{rnd} = \left\{ \frac{q'}{q} a \right\}$ is **known**

Assumption 2: "nice" homomorphic mult. noise

$$\mathrm{ct}_1 = (a_1, b_1):\;\; a_1 \cdot \mathrm{sk} + b_1 = \mathrm{Ecd}(m_1) + e_1 \;\; [q]$$
$$\mathrm{ct}_2 = (a_2, b_2):\;\; a_2 \cdot sk + b_2 = \mathrm{Ecd}(m_2) + e_2 \;\; [q]$$

$$\downarrow$$

$$\mathrm{ct}_\times = (a_\times, b_\times):\;\; a_\times \cdot \mathrm{sk} + b_\times = \mathrm{Ecd}(m_1 \cdot m_2) + e_\times \;\; [q']$$

with

$$e_\times \approx m_1 \cdot e_2 + m_2 \cdot e_1$$

BFV and CKKS can be parametrized to fit

CONTRIBUTION #1: CRYPTANALYSIS OF BS23

We are given $\mathrm{ct}^* = (a^*, b^*)$: $a^* \cdot \mathrm{sk} + b^* = \mathrm{Ecd}(m_\beta) + e \ [q]$
We want to distinguish $\beta = 0$ from $\beta = 1$

If Eval ends up with $\mathrm{Enc}(A)$,
we post-process as follows:

$$\mathrm{Enc}(1) \xrightarrow{\text{Rescale}} \mathrm{Enc}(1)$$

noise $\approx e_{rnd} \cdot \mathrm{sk}$
with known e_{rnd}

CONTRIBUTION #1: CRYPTANALYSIS OF BS23

We are given $\mathrm{ct}^* = (a^*, b^*)$: $a^* \cdot \mathrm{sk} + b^* = \mathrm{Ecd}(m_\beta) + e \;\; [q]$
We want to distinguish $\beta = 0$ from $\beta = 1$

If Eval ends up with $\mathrm{Enc}(A)$,
we post-process as follows:

$\mathrm{Enc}(1)$

$\downarrow$ Rescale

$\mathrm{Enc}(1)$

noise $\approx e_{rnd} \cdot \mathrm{sk}$
with known e_{rnd}

$\mathrm{Enc}(A)$

Multiply

$\mathrm{Enc}(A)$

noise $\approx A \cdot e_{rnd} \cdot \mathrm{sk}$
if A is large

CONTRIBUTION #1: CRYPTANALYSIS OF BS23

We are given $\mathrm{ct}^* = (a^*, b^*)$: $a^* \cdot \mathrm{sk} + b^* = \mathrm{Ecd}(m_\beta) + e \ [q]$
We want to distinguish $\beta = 0$ from $\beta = 1$

If Eval ends up with $\mathrm{Enc}(A)$,
we post-process as follows:

$\mathrm{Enc}(1)$

Rescale

$\mathrm{Enc}(1)$ $\quad$ $\mathrm{Enc}(A)$

noise $\approx e_{rnd} \cdot \mathrm{sk}$
with known e_{rnd}

Multiply

$\mathrm{Enc}(A)$

noise $\approx A \cdot e_{rnd} \cdot \mathrm{sk}$
if A is large

1. If the initial scheme is circuit-private, then so is the modified scheme
2. Request encryption and decryption of $\mathrm{MSB}(a^*)$
3. Recover $e_\times \approx \mathrm{MSB}(a^*) \cdot e_{rnd} \cdot \mathrm{sk}$
4. Compute $e_\times + e_{rnd} \cdot b^* \approx \mathrm{e_{rnd}} \cdot \mathrm{Ecd}(m_\beta)$
5. As the decr. modulus is small, we can distinguish

FORGETTING HISTORY REQUIRES RANDOMNESS

Deeper issue with the BS23 approach:

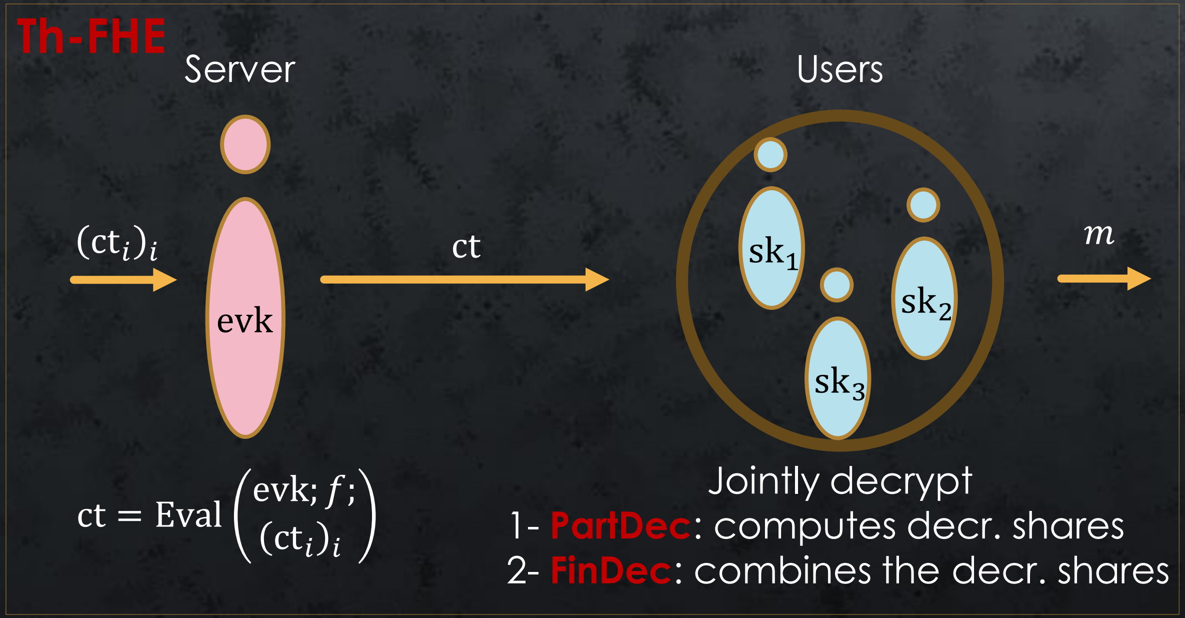
- Current circuit-privacy techniques require the server to **inject randomness**
- But the server is potentially a corrupted user ⇒ **not random to the adversary**

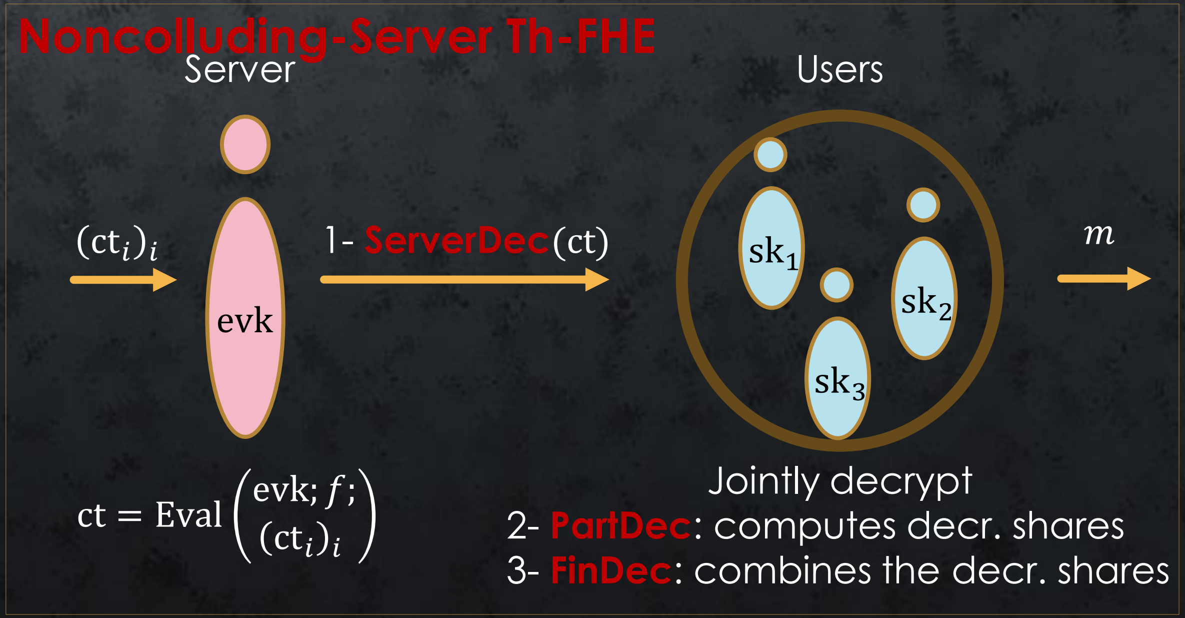
FORGETTING HISTORY REQUIRES RANDOMNESS

Deeper issue with the BS23 approach:

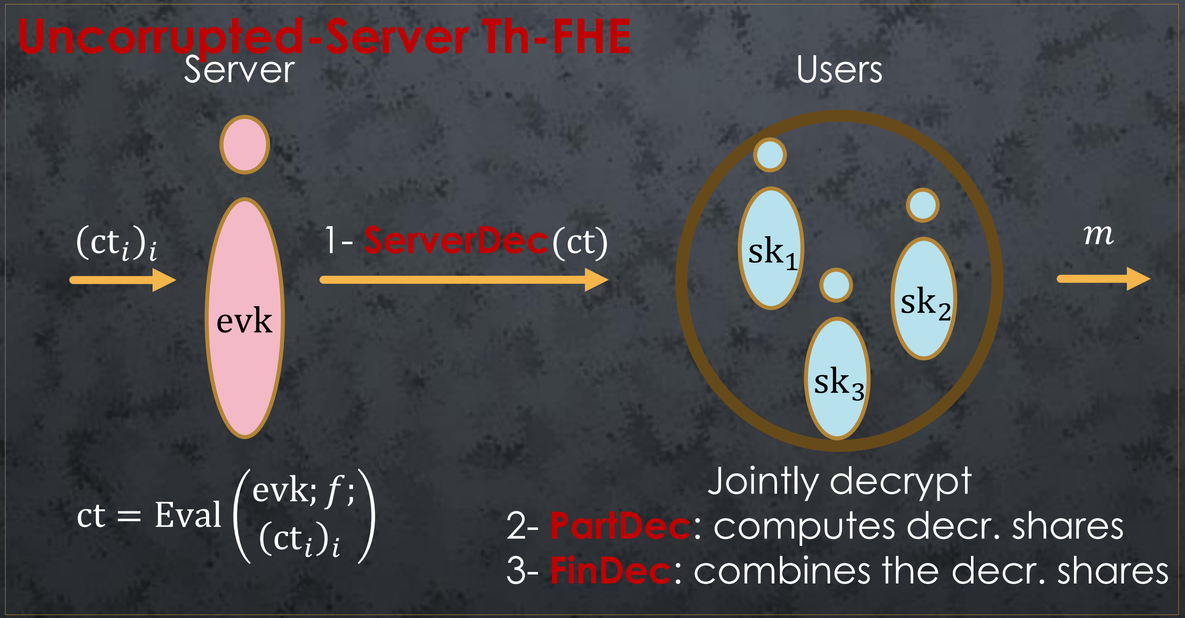
- Current circuit-privacy techniques require the server to **inject randomness**
- But the server is potentially a corrupted user ⇒ **not random to the adversary**

We propose an **noncolluding-server variant of Threshold-FHE**





DOES THE MODEL MAKE SENSE?



This depends on applications!

- OK if the group of users externalizes the computation to an outsider server
  ⇒ Somewhat trusted third party (may eavesdrop, may not collude)

- Not OK for the universal thresholdizer, which requires Eval to be deterministic

CONTRIBUTION #2: DOUBLE FLOOD & ROUND

$\mathrm{ct} = (a, b)\colon\ a \cdot \mathrm{sk} + b = \mathrm{Ecd}(m) + e\ \ [q]$

Split the key as

$\mathrm{sk} = \sum \mathrm{sk}_i$

ServerDec (ct)

Add $\mathrm{Enc}(0)$ to ct
Add exponential flooding to b-part
Rdm-Rescale to a **$\mathbf{poly}(\lambda)$ modulus**

PartDec $(\mathrm{ct}, \mathrm{sk}_i)$

$\mathrm{sh}_i \coloneqq a \cdot sk_i + e_i$
Using **tiny** e_i

FinDec $(\mathrm{ct}, (\mathrm{sh}_i)_i)$

$\mathrm{Dcd}\ (\sum \mathrm{sh}_i + b)$

* D. Micciancio, A. Suhl. Simulation-secure threshold PKE from LWE with polynomial modulus. eprint 2023/1728

CONTRIBUTION #2: DOUBLE FLOOD & ROUND

$$\mathrm{ct} = (a, b): \ a \cdot \mathrm{sk} + b = \mathrm{Ecd}(m) + e \ [q]$$

Split the key as

$\mathrm{sk} = \sum \mathrm{sk}_i$

ServerDec (ct)

Add $\mathrm{Enc}(0)$ to ct
Add exponential flooding to b-part
Rdm-Rescale to a **$\mathrm{poly}(\lambda)$ modulus**

PartDec $(\mathrm{ct}, \mathrm{sk}_i)$

$\mathrm{sh}_i := a \cdot sk_i + e_i$
Using **tiny** e_i

FinDec $(\mathrm{ct}, (\mathrm{sh}_i)_i)$

$\mathrm{Dcd}\ (\sum \mathrm{sh}_i + b)$

- Everything that users get and send is with a **$\mathrm{poly}(\lambda)$ modulus**
- This requires exponential flooding, but only internally to the server
- Proof technique closely related to MS23*

WRAP-UP

Contribution #1

Cryptanalysis of the BS23* Threshold-FHE with **moderate** decryption modulus

⇒ All known (general-purpose) Threshold-FHE's need **exponential** decr. modulus

Contribution #2

Construction of a Threshold-FHE with **tiny** decryption modulus…

… for the (specific) case where the server is not colluding.

Open problems:

- Can we get general-purpose Threshold-FHE with $\mathrm{poly}(Q_{dec})$ decryption modulus?
- Can we weaken the noncolluding-server assumption?

QUESTIONS?

Eprint 2024/1984

{ alain.passelegue, damien.stehle } @ cryptolab.co.kr