Practical Multi-Key Homomorphic Encryption for Federated Average Aggregation

2nd Workshop on Principles of Distributed Learning (PODL'23)
co-located with the International Symposium on Distributed Computing DISC'23
Friday, *13 October 2023*

Alberto Pedrouzo-Ulloa
apedrouzo@gts.uvigo.es , alberto.pedrouzoulloa@cea.fr

Joint work with A. Boudguiga, O. Chakraborty, R. Sirdey, O. Stan, M. Zuber
name.surname@cea.fr

Outline

- Introduction
- HE for Secure Aggregation
- Undressing HE
- What’s under the clothes
- Some outfit comparisons
- Conclusions

Introduction

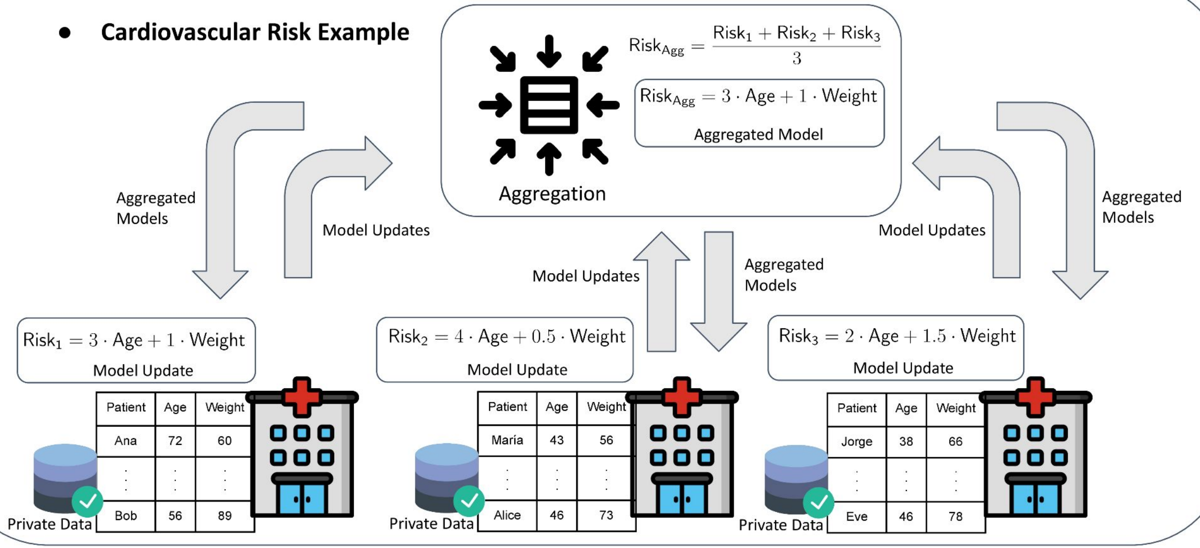
A little bit about Federated Learning and its problems

Example scenario for Federated Learning

- *FL* allows the *training of ML models without explicit sharing of training data*.
- A central server (**Aggregator**) aggregates the local training updates from Data Owners (DOs).
- **Cross-silo FL**: a model is built from the training sets of a reduced number of servers.
  - They are always available and computationally powerful.

This figure has been made using images from *www.flaticon.com* and *www.stockio.com*.

- **Cardiovascular Risk Example**



A toy example and some privacy risks

- Initially proposed to avoid moving the training data out
  - reducing communication costs and “*ensuring data privacy*.”

- Some example attacks:
  - Is  in the database of a particular hospital?
  - Can we reconstruct attributes of the people in the database?

$$\text{Risk}_1 = 3 \cdot \text{Age} + 1 \cdot \text{Weight}$$

$$\text{Risk}_2 = 4 \cdot \text{Age} + 0.5 \cdot \text{Weight}$$

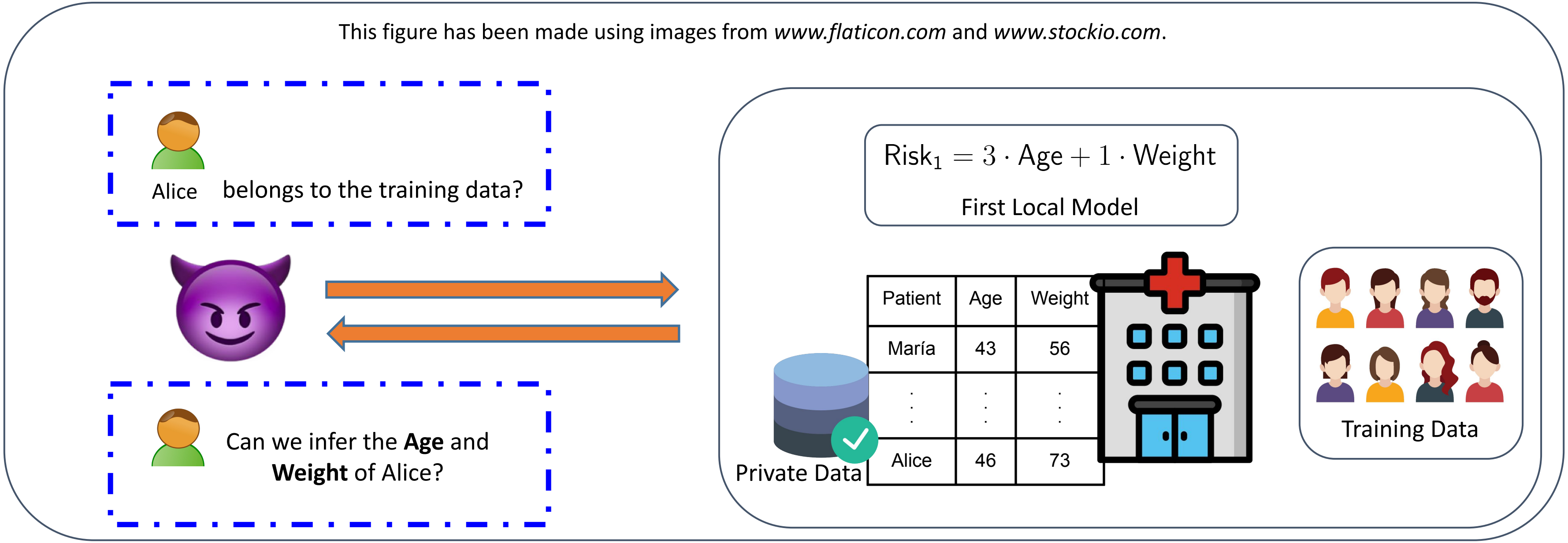
$$\text{Risk}_3 = 2 \cdot \text{Age} + 1.5 \cdot \text{Weight}$$

$$\text{Risk}_{\text{Agg}} = 3 \cdot \text{Age} + 1 \cdot \text{Weight}$$

The aggregator is the most dangerous party!

A toy example and some privacy risks

- Some example attacks:



MEMBERSHIP INFERENCE: TELL ME WHO YOU GO WITH, AND I'LL TELL YOU WHO YOU ARE

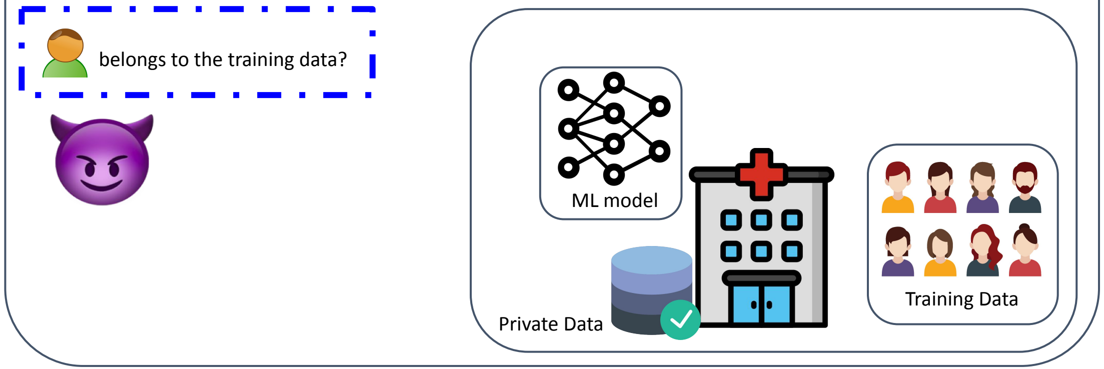
- **Membership inference:**
  - General cancer risk : 350 per 100000 people (aged 45 - 49)

https://www.cancer.gov/about-cancer/causes-prevention/risk/age

- **Membership inference:**
  - General cancer risk : 350 per 100000 people (aged 45 - 49)

https://www.cancer.gov/about-cancer/causes-prevention/risk/age

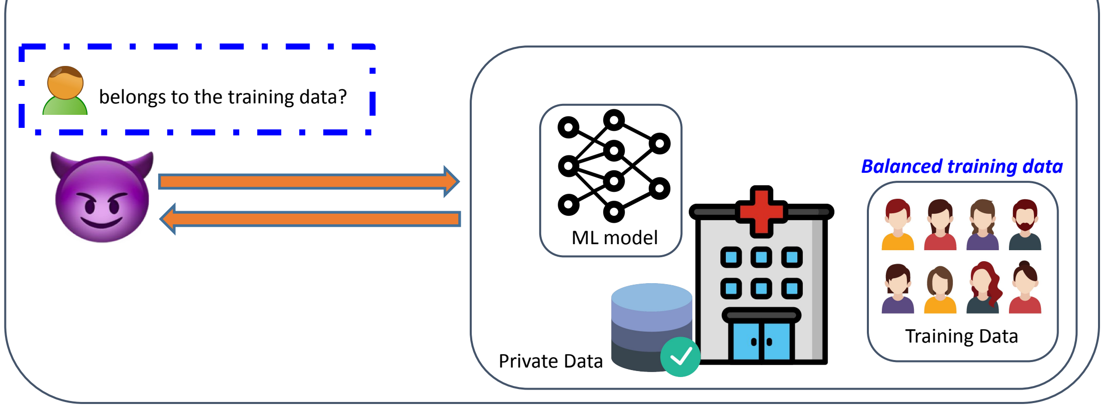
This figure has been made using images from *www.flaticon.com* and *www.stockio.com*.



- **Membership inference:**
  - General cancer risk : 350 per 100000 people (aged 45 - 49)
  - *"Cancer risk"* knowing that is contained in the training data: 1 per 2 people

https://www.cancer.gov/about-cancer/causes-prevention/risk/age

This figure has been made using images from *www.flaticon.com* and *www.stockio.com*.



Some basics of HE

RLWE and toy examples with Homomorphic Encryption (HE)

(Polynomial) Ring Learning with Errors

- **(P)RLWE problem:** RLWE relies upon the computational indistinguishability between the following pairs of samples:

$$R_q[z] = \mathbb{Z}_q[z]/(1+z^n)$$

Elements chosen uniformly at random

$$(a_i, b_i = a_i \cdot s + e_i) \approx (a_i, u_i)$$

$$\chi[z]$$

Elements drawn from the error distribution

(Polynomial) Ring Learning with Errors

- **(P)RLWE problem:** RLWE relies upon the computational indistinguishability between the following pairs of samples:

How difficult is to distinguish highly depends on the length of the polynomials.

$$R_q[z] = \mathbb{Z}_q[z]/(1+z^n)$$

Elements chosen uniformly at random

$$(a_i, b_i = a_i \cdot s + e_i) \approx (a_i, u_i)$$

$$\chi[z]$$

Elements drawn from the error distribution

PLWE/RLWE: BGV-type example for HE

- **(P)RLWE problem:** RLWE relies upon the computational indistinguishability between the following pairs of samples:

$R_q[z] = \mathbb{Z}_q[z]/(1+z^n)$

Elements chosen uniformly at random

$$(a_i, b_i = a_i \cdot s + t \cdot e_i + m) \approx (a_i, u_i)$$

$\chi[z]$

Elements drawn from the error distribution

$\mathbb{Z}_t[z]/(1+z^n)$

An example of a simple BGV-type HE

- ***Consider two encryptions:***

$$\mathsf{Enc}(m_1) = (a_1, b_1 = -a_1 s + te_1 + m_1)$$
$$\mathsf{Enc}(m_2) = (a_2, b_2 = -a_2 s + te_2 + m_2)$$

- ***Decryption:***

$$(b_1 + a_1 s \bmod 1 + z^n) \bmod q = m_1 + te_1$$

- ***Homomorphic Addition:***

$$\mathsf{Enc}(m_1 + m_2) = (a_{\mathsf{add}} = a_1 + a_2, b_{\mathsf{add}} = b_1 + b_2)$$
$$\mathsf{Enc}(m_1 + m_2) = (a_{\mathsf{add}}, b_{\mathsf{add}} = -a_{\mathsf{add}} s + t(e_1 + e_2) + (m_1 + m_2))$$

An example of a simple BGV-type HE

- ***Consider two encryptions:***

$$\mathsf{Enc}(m_1) = (a_1, b_1 = -a_1 s + te_1 + m_1)$$
$$\mathsf{Enc}(m_2) = (a_2, b_2 = -a_2 s + te_2 + m_2)$$

- ***Decryption:***

$$(b_1 + a_1 s \bmod 1 + z^n) \bmod q = m_1 + te_1$$

- ***Homomorphic Addition:***

$$\mathsf{Enc}(m_1 + m_2) = (a_{\mathsf{add}} = a_1 + a_2, b_{\mathsf{add}} = b_1 + b_2)$$
$$\mathsf{Enc}(m_1 + m_2) = (a_{\mathsf{add}}, b_{\mathsf{add}} = -a_{\mathsf{add}} s + t(e_1 + e_2) + (m_1 + m_2))$$

An example of a simple BGV-type HE

- ***Consider two encryptions:***

$$\mathsf{Enc}(m_1) = (a_1, b_1 = -a_1 s + t e_1 + m_1)$$
$$\mathsf{Enc}(m_2) = (a_2, b_2 = -a_2 s + t e_2 + m_2)$$

- ***Decryption:***

$$(b_1 + a_1 s \bmod 1 + z^n) \bmod q = m_1 + t e_1$$

- ***Homomorphic Addition:***

$$\mathsf{Enc}(m_1 + m_2) = (a_{\mathsf{add}} = a_1 + a_2, b_{\mathsf{add}} = b_1 + b_2)$$
$$\mathsf{Enc}(m_1 + m_2) = (a_{\mathsf{add}}, b_{\mathsf{add}} = -a_{\mathsf{add}} s + t(e_1 + e_2) + (m_1 + m_2))$$

An example of a simple BGV-type HE

- ***Consider two encryptions:***

$$\text{Enc}(m_1) = (a_1, b_1 = -a_1 s + te_1 + m_1)$$
$$\text{Enc}(m_2) = (a_2, b_2 = -a_2 s + te_2 + m_2)$$

- ***Decryption:***

$$(b_1 + a_1 s \bmod 1 + z^n) \bmod q = m_1 + te_1$$

- ***Homomorphic Addition:***

$$\text{Enc}(m_1 + m_2) = (a_{\text{add}} = a_1 + a_2, b_{\text{add}} = b_1 + b_2)$$
$$\text{Enc}(m_1 + m_2) = (a_{\text{add}}, b_{\text{add}} = -a_{\text{add}} s + t(e_1 + e_2) + (m_1 + m_2))$$

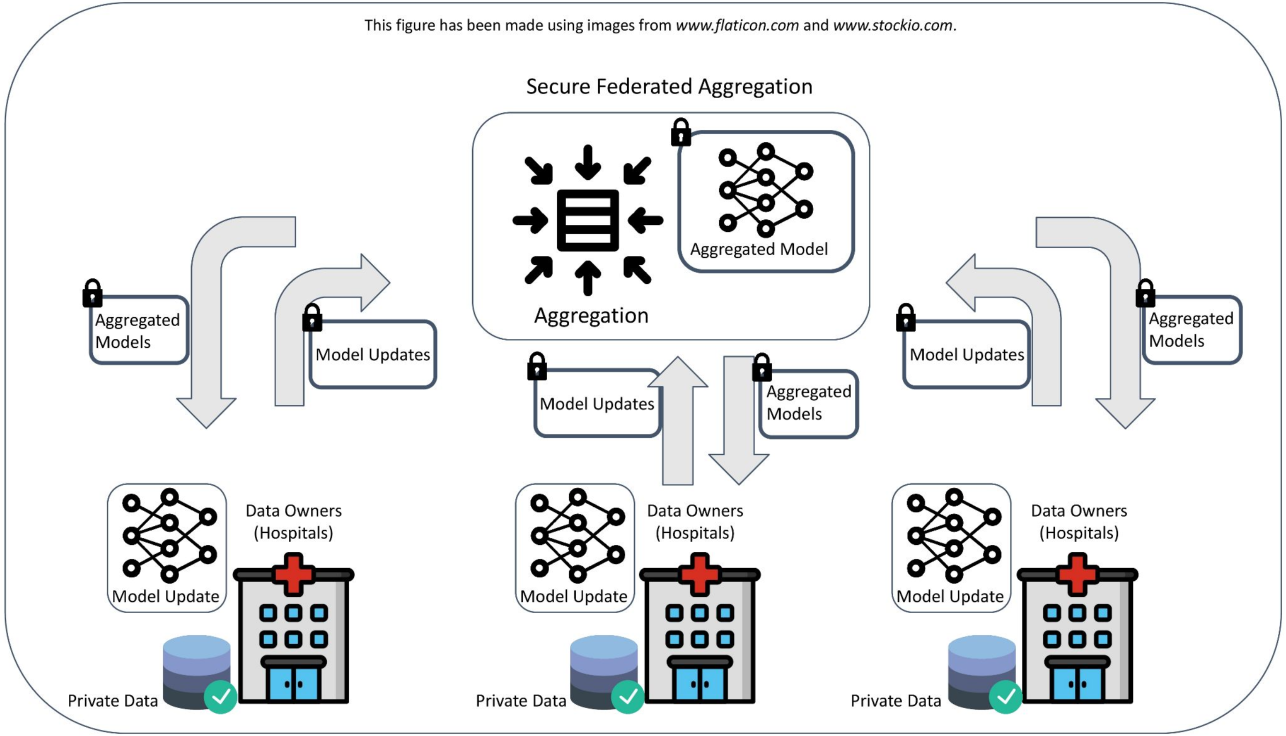
- ***Homomorphic Multiplication:***
  - It is slightly more complicated

HE for Secure Aggregation

Achieving protection against the aggregator

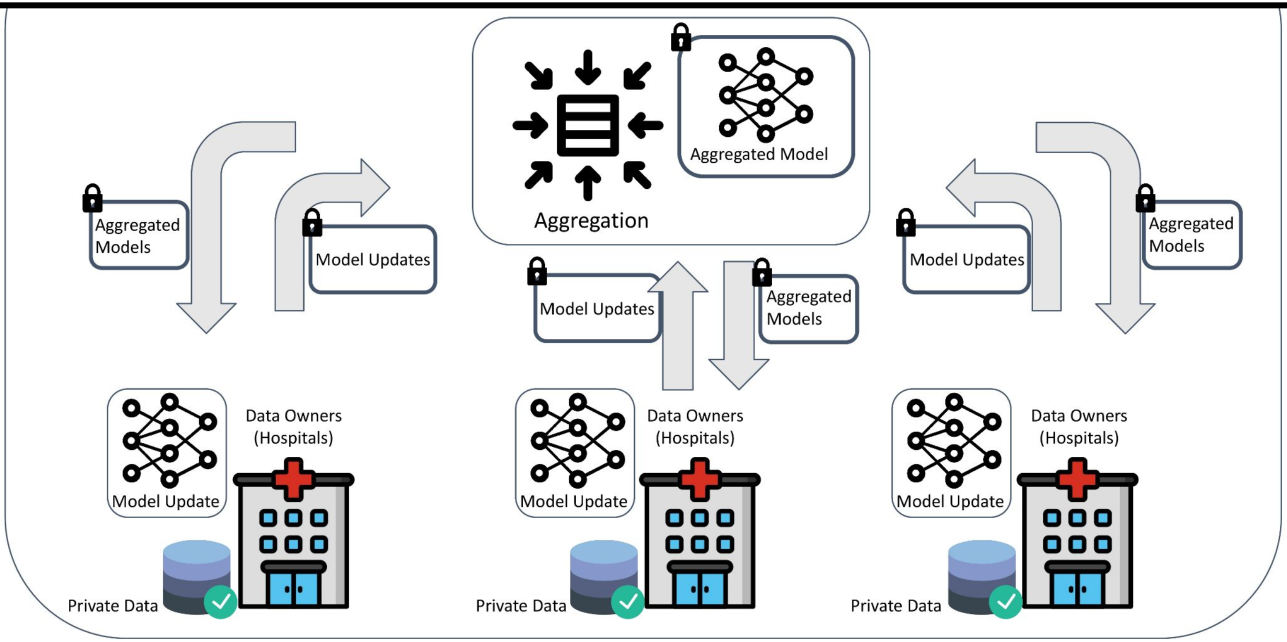
Secure Aggregation: Protection against the aggregator

- Homomorphic Encryption (HE) counters with the confidentiality threats from the Aggregator.



Secure Aggregation:
Protection against the aggregator

- Homomorphic Encryption (HE) counters with the confidentiality threats from the Aggregator.
  - It seems to be a perfect fit for secure aggregation.
  - It respects the communication flow of unprotected FL.

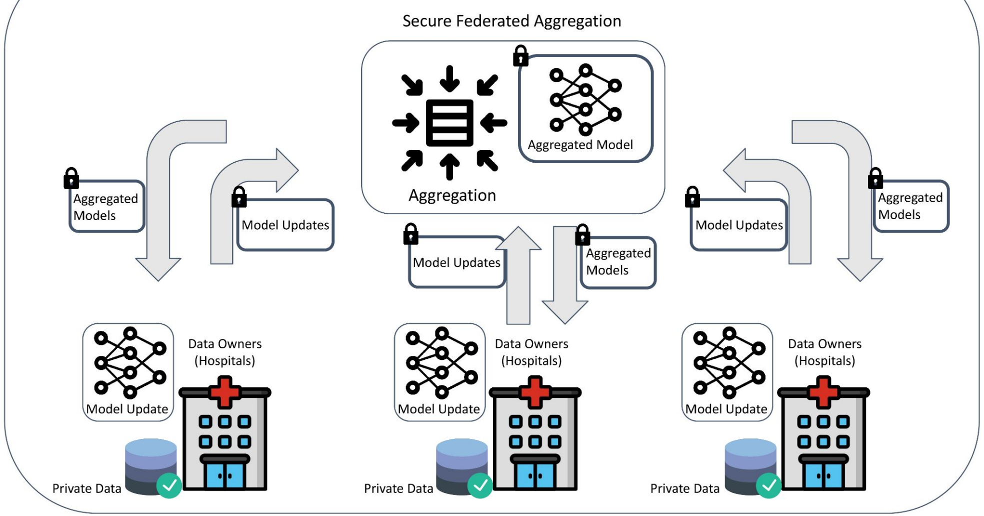


TRUMPET

Secure Aggregation: Protection against the aggregator

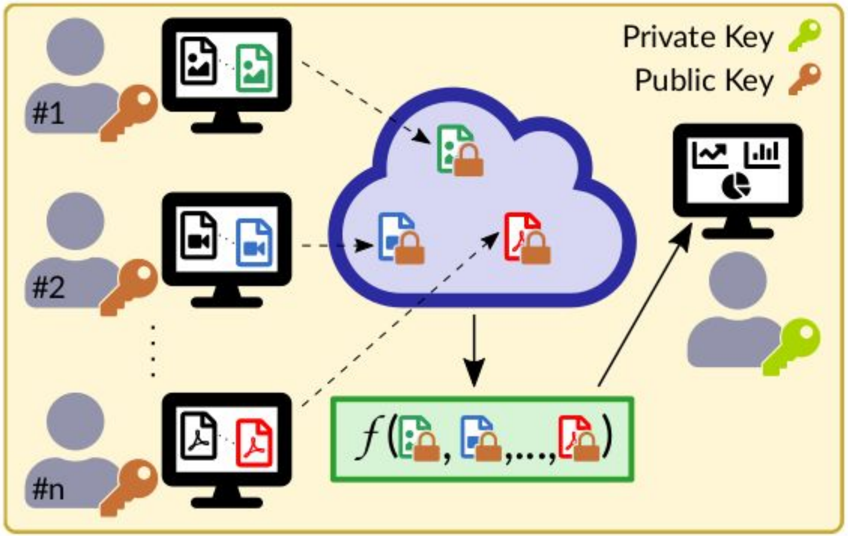
- Single-key HE imposes the need of
  - a trusted decryptor.
  - non-colluding assumption among Aggregator and decryptor.

This figure has been made using images from *www.flaticon.com* and *www.stockio.com*.

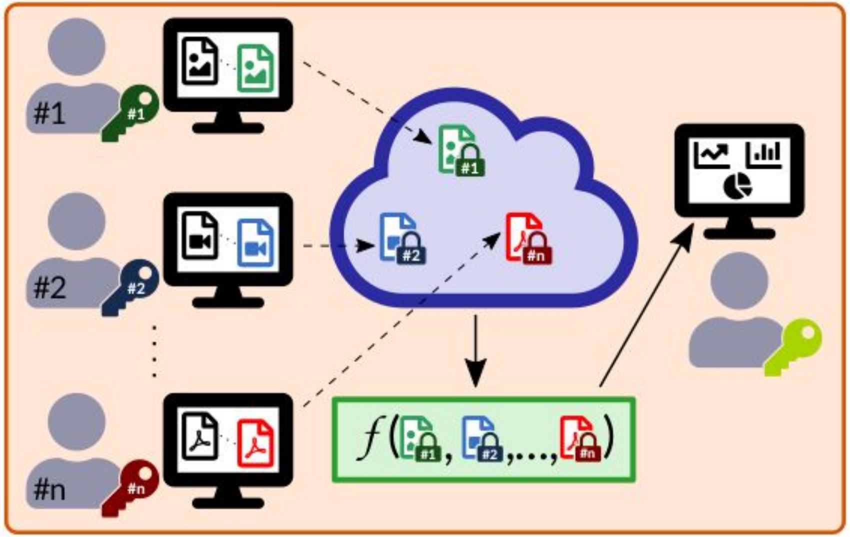


Single-key HE vs Multi-key HE

- Our scenario requires to incorporate multiple keys into HE.
  - Prevents decryption without permission of other participants.



Single key



Multiple keys

(S)HE looks nice, but maybe too many clothes for FL

Our motivation:

- Many works address the problem of secure aggregation in FL.
- To the best of our knowledge, HE has not been yet fully optimized for this setting.

Our objective:

- *Tailor and optimize HE constructions for secure average aggregation.*

We propose:

- *A lightweight communication-efficient multi-key approach suitable for the Federated Averaging rule.*

TRUMPET

Undressing HE:
a talk with “streaptease”

This is not what it seems

First outfit: Using a BFV-type encryption

- ***Public key generation:***

$$\mathsf{PK} = \mathsf{Enc}(0) = (a, b = -(as + e))$$

- ***Encryption:***
  - We encrypt a message $m \in R_p = \mathbb{Z}_p[X]/(1 + X^n)$

$$\mathsf{Enc}(m) = (c_0 = \mathsf{PK}[0]u + e_0, c_1 = \mathsf{PK}[1]u + e_1 + \underbrace{\Delta}_{\lfloor q/p \rfloor} \cdot m) \in R_q^2$$

- ***Multiple keys with an (L-out-of-L) threshold variant of BFV:***

$$\mathsf{SK} = s = s_1 + \ldots + s_L$$

First outfit: Using a BFV-type encryption

- ***Public key generation:***

$$\mathsf{PK} = \mathsf{Enc}(0) = (a, b = -(as + e))$$

- ***Encryption:***
  - We encrypt a message $m \in R_p = \mathbb{Z}_p[X]/(1 + X^n)$

$$\mathsf{Enc}(m) = (c_0 = \mathsf{PK}[0]u + e_0, c_1 = \mathsf{PK}[1]u + e_1 + \underbrace{\Delta}_{\lfloor q/p \rfloor} \cdot m) \in R_q^2$$

- ***Multiple keys with an (L-out-of-L) threshold variant of BFV:***

$$\mathsf{SK} = s = s_1 + \ldots + s_L$$

TRUMPET

First outfit: Using a BFV-type encryption

- ***Public key generation:***

$$\mathsf{PK} = \mathsf{Enc}(0) = (a, b = -(as + e))$$

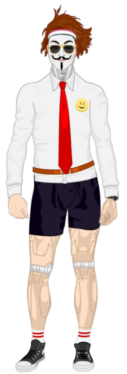
- ***Encryption:***
  - We encrypt a message $m \in R_p = \mathbb{Z}_p[X]/(1 + X^n)$

$$\mathsf{Enc}(m) = (c_0 = \mathsf{PK}[0]u + e_0, c_1 = \mathsf{PK}[1]u + e_1 + \underbrace{\Delta}_{\lfloor q/p \rfloor} \cdot m) \in R_q^2$$

- ***Multiple keys with an (L-out-of-L) threshold variant of BFV:***

$$\mathsf{SK} = s = s_1 + \ldots + s_L$$

Take it off all



- ***The public key is not needed:***
  - Each Data Owner can encrypt with its own secret key.

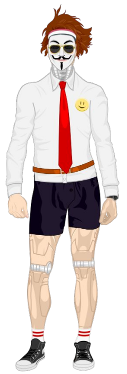
$$(a, b_i = as_i + e_i + \Delta \cdot m_i)$$

- ***Encrypted updates can be aggregated on the fly:***
  - By sharing the same "*a*", then "b" components are directly aggregated.

$$\left(a, \sum_i b_i = a(\sum_i s_i) + \sum_i e_i + \Delta \cdot \sum_i m_i = as + e + \Delta \cdot m\right)$$

  - There is no need to send "*a*".

Take it off all



- ***The public key is not needed:***
  - Each Data Owner can encrypt with its own secret key.

$$(a, b_i = as_i + e_i + \Delta \cdot m_i)$$

- ***Encrypted updates can be aggregated on the fly:***
  - By sharing the same "*a*", then "b" components are directly aggregated.

$$\left(a, \sum_i b_i = a(\sum_i s_i) + \sum_i e_i + \Delta \cdot \sum_i m_i = as + e + \Delta \cdot m\right)$$

  - There is no need to send "*a*".

Take it off all

- ***The public key is not needed:***
  - Each Data Owner can encrypt with its own secret key.

$$(a, b_i = as_i + e_i + \Delta \cdot m_i)$$

- ***Encrypted updates can be aggregated on the fly:***
  - By sharing the same "*a*", then "b" components are directly aggregated.

$$\left(a, \sum_i b_i = a(\sum_i s_i) + \sum_i e_i + \Delta \cdot \sum_i m_i = as + e + \Delta \cdot m\right)$$

  - There is no need to send "*a*".

TRUMPET

Take it off all

- ***The public key is not needed:***
  - Each Data Owner can encrypt with its own secret key.

$$(a, b_i = as_i + e_i + \Delta \cdot m_i)$$

- ***Encrypted updates can be aggregated on the fly:***
  - By sharing the same "*a*", then "b" components are directly aggregated.

$$\left(a, \sum_i b_i = a\left(\sum_i s_i\right) + \sum_i e_i + \Delta \cdot \sum_i m_i = as + e + \Delta \cdot m\right)$$

  - There is no need to send "*a*".

Take it off all

$$\lfloor as_i \rceil_p = \lfloor p/q \cdot as_i \rceil$$

- ***The public key is not needed:***
  - Each Data Owner can encrypt with its own secret key.

$$(a, b_i = as_i + e_i + \Delta \cdot m_i)$$

- ***Encrypted updates can be aggregated on the fly:***
  - By sharing the same "*a*", then "b" components are directly aggregated.

$$\left(a, \sum_i b_i = a(\sum_i s_i) + \sum_i e_i + \Delta \cdot \sum_i m_i = as + e + \Delta \cdot m\right)$$

  - There is no need to send "*a*".

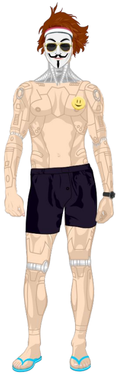
To have distributed decryption, each DO has to send $\lfloor as_i \rceil_p$

but it also decrypts the input ciphertext!

Proposed solution

Take it off all, but carefully

Proposed solution: You can leave your hat on…



Masking the secret keys: $(a, b_i = a(s_i + \mathsf{share}_i) + e_i + \Delta \cdot m_i)$

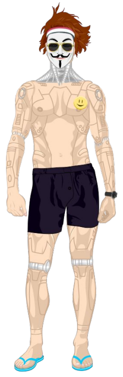
$$\left(\sum_i b_i\right) = a(s + \underbrace{\sum_i \mathsf{share}_i}_{0}) + e = a \underbrace{s}_{\sum_i s_i} + \underbrace{e}_{\sum_i e_i} + \Delta \cdot \underbrace{m}_{\sum_i m_i}$$

Building blocks:

- Additive secret shares of zero $\sum_i \mathsf{share}_i = 0$
- A PRF is used to agree in the same "*a*" per each round.
- Next lemma is used to remove the error in a distributed way:

Lemma 1 **(Lemma** 1 **[3]).** *Let* $p|q$, $\boldsymbol{x} \leftarrow R_q^N$ *and* $\boldsymbol{y} = \boldsymbol{x} + \mathbf{e} \bmod q$ *for some* $\mathbf{e} \in R_q^N$ *with* $\|\mathbf{e}\|_\infty < B < q/p$. *Then* $\Pr\left(\lfloor \boldsymbol{y} \rceil_p \neq \lfloor \boldsymbol{x} \rceil_p \bmod p\right) \leq \frac{2npNB}{q}$.

TRUMPET



Proposed solution: You can leave your hat on…

Masking the secret keys: $(a, b_i = a(s_i + \text{share}_i) + e_i + \Delta \cdot m_i)$

$$\left(\sum_i b_i\right) = a(s + \underbrace{\sum_i \text{share}_i}_{0}) + e = a \underbrace{s}_{\sum_i s_i} + \underbrace{e}_{\sum_i e_i} + \Delta \cdot \underbrace{m}_{\sum_i m_i}$$

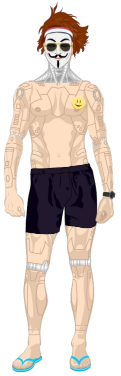
Building blocks:

- Additive secret shares of zero $\sum_i \text{share}_i = 0$
- A PRF is used to agree in the same "*a*" per each round.
- Next lemma is used to remove the error in a distributed way:

Lemma 1 (Lemma 1 [3]). *Let* $p|q$, $\boldsymbol{x} \leftarrow R_q^N$ *and* $\boldsymbol{y} = \boldsymbol{x} + \boldsymbol{e} \bmod q$ *for some* $\boldsymbol{e} \in R_q^N$ *with* $\|\boldsymbol{e}\|_\infty < B < q/p$. *Then* $\Pr\left(\lfloor \boldsymbol{y} \rceil_p \neq \lfloor \boldsymbol{x} \rceil_p \bmod p\right) \leq \frac{2npNB}{q}$.

TRUMPET

Proposed solution: You can leave your hat on…



Masking the secret keys: $(a, b_i = a(s_i + \text{share}_i) + e_i + \Delta \cdot m_i)$

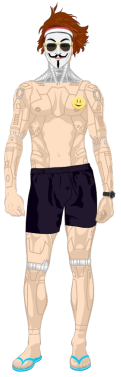
$$\left(\sum_i b_i\right) = a(s + \underbrace{\sum_i \text{share}_i}_{0}) + e = a\underbrace{s}_{\sum_i s_i} + \underbrace{e}_{\sum_i e_i} + \Delta \cdot \underbrace{m}_{\sum_i m_i}$$

Building blocks:

- Additive secret shares of zero $\sum_i \text{share}_i = 0$
- A PRF is used to agree in the same "*a*" per each round.
- Next lemma is used to remove the error in a distributed way:

Lemma 1 (Lemma 1 [3]). *Let* $p|q$, $\boldsymbol{x} \leftarrow R_q^N$ *and* $\boldsymbol{y} = \boldsymbol{x} + \mathbf{e} \bmod q$ *for some* $\mathbf{e} \in R_q^N$ *with* $\|\mathbf{e}\|_\infty < B < q/p$. *Then* $\Pr\left(\lfloor \boldsymbol{y} \rceil_p \neq \lfloor \boldsymbol{x} \rceil_p \bmod p\right) \leq \frac{2npNB}{q}$.

Proposed solution: You can leave your hat on…



Masking the secret keys: $(a, b_i = a(s_i + \mathsf{share}_i) + e_i + \Delta \cdot m_i)$

$$\left(\sum_i b_i\right) = a(s + \underbrace{\sum_i \mathsf{share}_i}_{0}) + e = a \underbrace{s}_{\sum_i s_i} + \underbrace{e}_{\sum_i e_i} + \Delta \cdot \underbrace{m}_{\sum_i m_i}$$

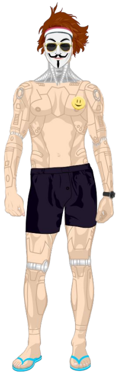
Building blocks:

- Additive secret shares of zero $\sum_i \mathsf{share}_i = 0$
- A PRF is used to agree in the same "*a*" per each round.
- Next lemma is used to remove the error in a distributed way:

Lemma 1 (Lemma 1 [3]). *Let* $p|q$, $\boldsymbol{x} \leftarrow R_q^N$ *and* $\boldsymbol{y} = \boldsymbol{x} + \mathbf{e} \bmod q$ *for some* $\mathbf{e} \in R_q^N$ *with* $\|\mathbf{e}\|_\infty < B < q/p$. *Then* $\Pr\left(\lfloor \boldsymbol{y} \rceil_p \neq \lfloor \boldsymbol{x} \rceil_p \bmod p\right) \leq \frac{2npNB}{q}$.

TRUMPET

Proposed solution: You can leave your hat on…



Masking the secret keys: $(a, b_i = a(s_i + \mathsf{share}_i) + e_i + \Delta \cdot m_i)$

$$\left(\sum_i b_i\right) = a(s + \underbrace{\sum_i \mathsf{share}_i)}_{0} + e = a \underbrace{s}_{\sum_i s_i} + \underbrace{e}_{\sum_i e_i} + \Delta \cdot \underbrace{m}_{\sum_i m_i}$$

Building blocks:

- Additive secret shares of zero $\sum_i \mathsf{share}_i = 0$
- A PRF is used to agree in the same "*a*" per each round.
- Next lemma is used to remove the error in a distributed way:

Lemma 1 (Lemma 1 [3]). *Let* $p|q$, $\boldsymbol{x} \leftarrow R_q^N$ *and* $\boldsymbol{y} = \boldsymbol{x} + \boldsymbol{e} \bmod q$ *for some* $\boldsymbol{e} \in R_q^N$ *with* $\|\mathbf{e}\|_\infty < B < q/p$. *Then* $\Pr\left(\lfloor \boldsymbol{y} \rceil_p \neq \lfloor \boldsymbol{x} \rceil_p \bmod p\right) \leq \frac{2npNB}{q}$.

Proposed solution: You can leave your hat on

- Next lemma is used to remove the error in a distributed way:

  Lemma 1 (Lemma 1 [3]). *Let* $p|q$, $\boldsymbol{x} \leftarrow R_q^N$ *and* $\boldsymbol{y} = \boldsymbol{x} + \mathbf{e} \bmod q$ *for some* $\mathbf{e} \in R_q^N$ *with* $\|\mathbf{e}\|_\infty < B < q/p$. *Then* $\Pr\left(\lfloor \boldsymbol{y} \rceil_p \neq \lfloor \boldsymbol{x} \rceil_p \bmod p\right) \leq \frac{2npNB}{q}$.

- It can be used to show that $\lfloor b \rceil_p = \lfloor as + e \rceil_p + m \neq \lfloor as \rceil_p + m$ with at most probability $\Pr(\mathsf{Ev})$

- By bounding $\Pr(\mathsf{Ev}) \leq 2^{-\kappa}$:

$$q \geq 4 \cdot n^2 \cdot N_{\mathsf{AggRounds}} \cdot N_{\mathsf{Ctxts.PerRound}} \cdot p \cdot L^2 \cdot B_{\mathsf{Init}}^2 \cdot 2^{\kappa}$$

TRUMPET

Proposed solution: You can leave your hat on

- Next lemma is used to remove the error in a distributed way:

  Lemma 1 (Lemma 1 [3]). *Let* $p|q$, $\boldsymbol{x} \leftarrow R_q^N$ *and* $\boldsymbol{y} = \boldsymbol{x} + \mathbf{e} \bmod q$ *for some* $\mathbf{e} \in R_q^N$ *with* $\|\mathbf{e}\|_\infty < B < q/p$. *Then* $\Pr\left(\lfloor \boldsymbol{y} \rceil_p \neq \lfloor \boldsymbol{x} \rceil_p \bmod p\right) \leq \frac{2npNB}{q}$.

- It can be used to show that $\lfloor b \rceil_p = \lfloor as + e \rceil_p + m \neq \lfloor as \rceil_p + m$ with at most probability $\mathsf{Pr}(\mathsf{Ev})$

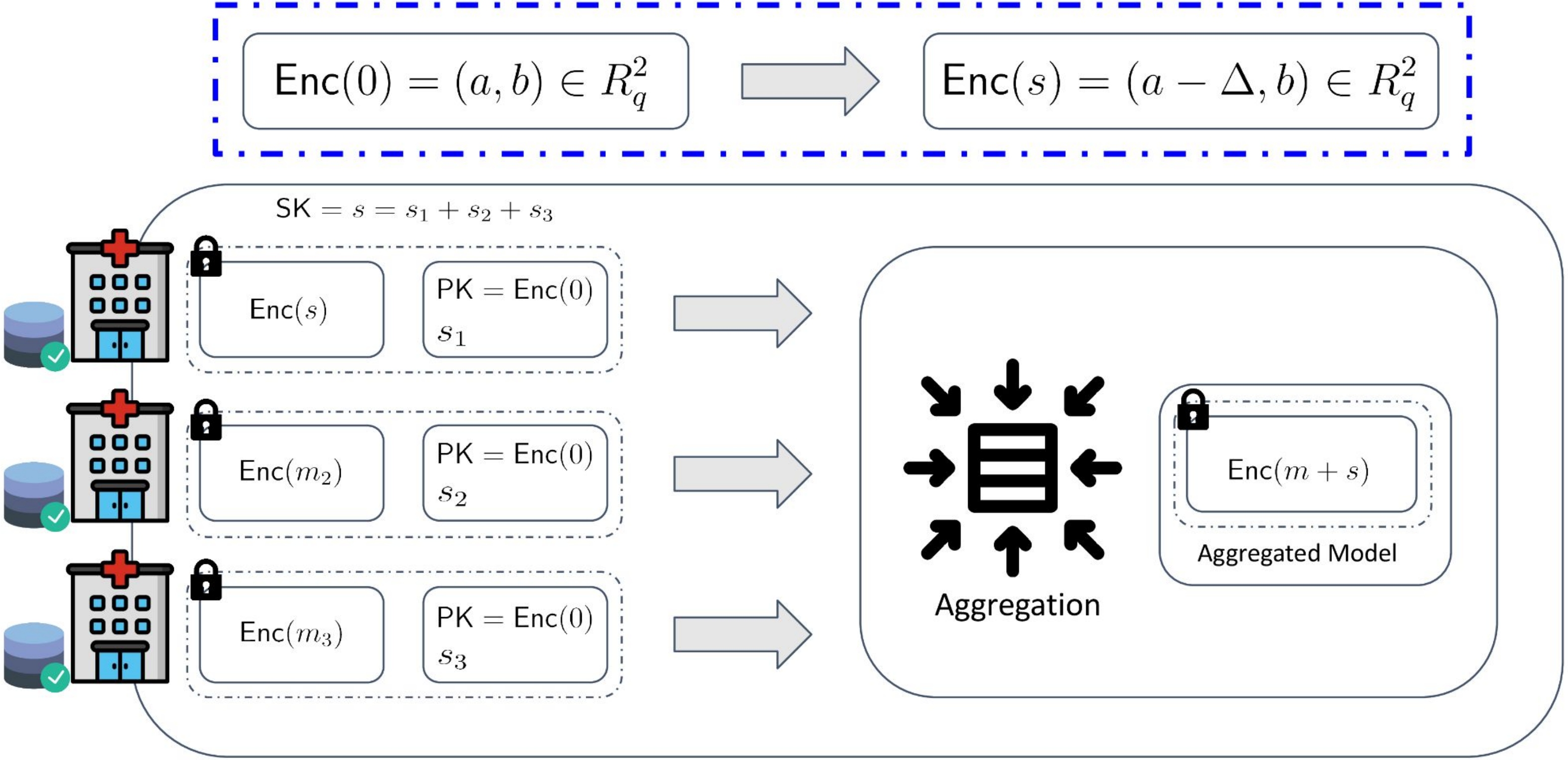
- By bounding $\mathsf{Pr}(\mathsf{Ev}) \leq 2^{-\kappa}$:

$$q \geq 4 \cdot n^2 \cdot N_{\mathsf{AggRounds}} \cdot N_{\mathsf{Ctxts.PerRound}} \cdot p \cdot L^2 \cdot B_{\mathsf{Init}}^2 \cdot 2^{\kappa}$$

What’s under the clothes

Some nice surprises

Dishonest Data Owners



Some nice properties

- **LImiting ciphertexts' malleability**
  - By assuming the Common Reference String (*CRS*) model, a different "*a*" term is fixed per each aggregation round.
- **Upgrade to malicious aggregators**
  - The Aggregator can only apply additive transformations without being detected.
  - An extra condition check can be embedded into ciphertexts to verify honest behavior.
- **Stronger semi-honest DOs:**
  - As there is no public key, DOs cannot generate encryptions of the global secret key.

Some nice properties

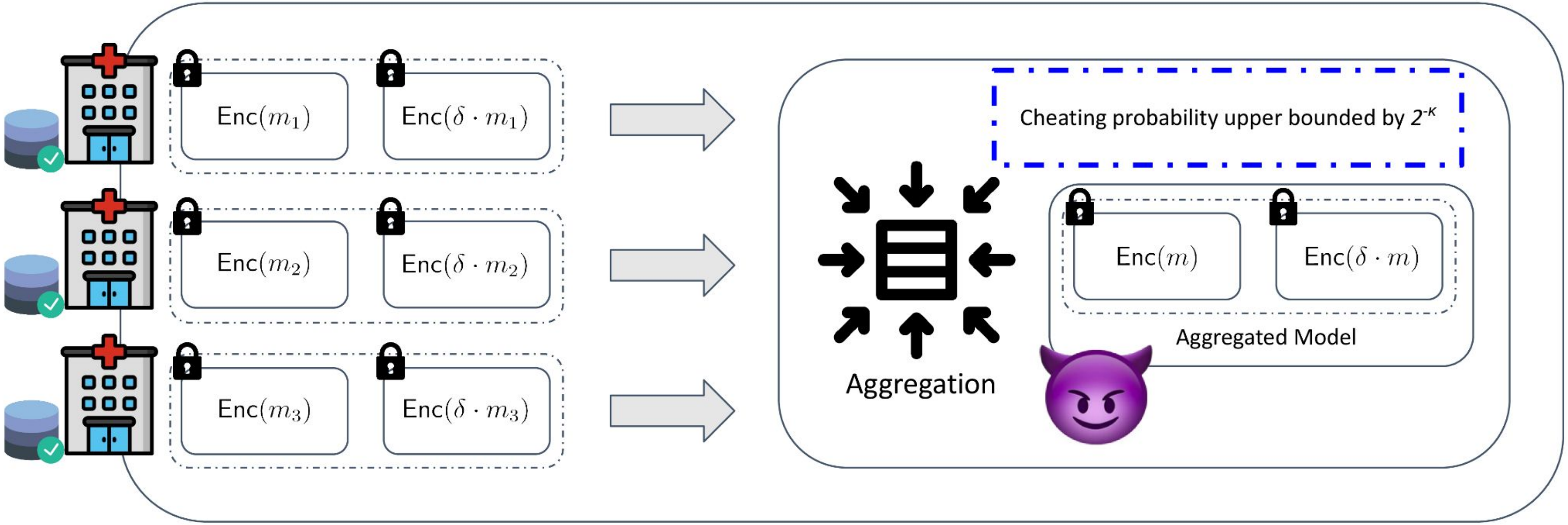
- **LImiting ciphertexts' malleability**
    - By assuming the Common Reference String (*CRS*) model, a different "*a*" term is fixed per each aggregation round.
- **Upgrade to malicious aggregators**
    - The Aggregator can only apply additive transformations without being detected.
    - An extra condition check can be embedded into ciphertexts to verify honest behavior.
- **Stronger semi-honest DOs:**
    - As there is no public key, DOs cannot generate encryptions of the global secret key.

Some nice properties

- **LImiting ciphertexts' malleability**
  - By assuming the Common Reference String (*CRS*) model, a different "*a*" term is fixed per each aggregation round.
- **Upgrade to malicious aggregators**
  - The Aggregator can only apply additive transformations without being detected.
  - An extra condition check can be embedded into ciphertexts to verify honest behavior.
- **Stronger semi-honest DOs:**
  - As there is no public key, DOs cannot generate encryptions of the global secret key.

An upgrade to malicious aggregators

- An extra condition check can be embedded into Secret-Key ciphertexts (e.g., $\delta \cdot m$ with δ unknown to aggregator). This verifies the honest behavior during aggregation.



Some outfit comparisons

Comparing with others HE-based solutions

Comparison with other solutions

M: Model Size N: Number of DOs n: lattice dimension M ≈ constant · n	Ours [2]	[5]	[3]	[4]	[6]
Agg. Comp. Cost	$O(MN)$ add.	$O(MN)$ mult.	$O(MN)$ add.	$O(MN)$ add.	$O(MN^2)$
DO Comp. Cost	LWE: $O(Mn)$ mult. RLWE: $O(M \log M)$ mult.	$O(M)$ exp.	$O(M \log M)$ mult.	$O(M \log M)$ mult.	$O(MN + N^2)$
Total Com. Cost	$O(MN)$	$O(MN)$	$O(MN)$	$O(MN)$	$O(MN + N^2)$
Multiple Keys	✅	🚫	🚫	✅	✅
Passive parties	✅	✅	✅	✅	✅
Malicious Agg.	✅ Verify Agg.	✅ Verify Agg.	🚫	🚫	✅ only DOs input privacy if $T > N/2$
Assumptions	LWE/RLWE	Paillier	RLWE	RLWE	T non-colluding DOs
Flexible Dec.	✅ only DOs contributing to aggregated model	🚫	🚫	🚫	✅ required T out of N DOs

Conclusions

When you go to the beach, all you truly need is a bathing suit!

Conclusions

- We *tailor and optimize HE* constructions *for secure average aggregation*.
- Multi-key homomorphic encryption mitigates c*ollusion attacks between aggregator and data owners*.
- We propose a lightweight communication-efficient multi-key approach suitable for the Federated Averaging rule.
    - Communication cost per party is reduced approximately
        - by a half with RLWE.
        - from quadratic to linear in terms of lattice dimension if considering LWE.
    - Easy to update to be secure against malicious aggregators.

TRUMPET

Thanks for your attention!

References:

[1] Mohamad Mansouri, Melek Önen, Wafa Ben Jaballah, and Mauro Conti, "Sok: Secure aggregation based on cryptographic schemes for federated learning," Proc. Priv. Enhancing Technol., vol. 2023, no. 1, pp. 140–157, 2023.

[2] Alberto Pedrouzo-Ulloa, Aymen Boudguiga, Olive Chakraborty, Renaud Sirdey, Oana Stan, and Martin Zuber, "Practical multi-key homomorphic encryption for more flexible and efficient secure federated aggregation (preliminary work)," IACR Cryptol. ePrint Arch., p. 1674, 2022. Published in IEEE CSR 2023: 612-617.

[3] Arnaud Grivet Sébert, Renaud Sirdey, Oana Stan, and Cédric Gouy-Pailler, "Protecting data from all parties: Combining FHE and DP in federated learning," CoRR, vol. abs/2205.04330, 2022.

[4] Christian Mouchet, Juan Ramón Troncoso-Pastoriza, Jean-Philippe Bossuat, and Jean-Pierre Hubaux, "Multiparty homomorphic encryption from ring-learning-with-errors," Proc. Priv. Enhancing Technol., vol. 2021, no. 4, pp. 291–311, 2021.

[5] Abbass Madi, Oana Stan, Aurélien Mayoue, Arnaud Grivet-Sébert, Cédric Gouy-Pailler, and Renaud Sirdey, "A secure federated learning framework using homomorphic encryption and verifiable computing," 2021, pp. 1–8.

[6] Kallista A. Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, H. Brendan McMahan, Sarvar Patel, Daniel Ramage, Aaron Segal, and Karn Seth, "Practical secure aggregation for privacy-preserving machine learning," in ACM SIGSAC CCS. 2017, pp. 1175–1191, ACM.

Bonus slides: An example of a simple BGV-type HE

- ***Consider two encryptions:***

$$\mathsf{Enc}(m_1) = (a_1, b_1 = -a_1 s + te_1 + m_1)$$
$$\mathsf{Enc}(m_2) = (a_2, b_2 = -a_2 s + te_2 + m_2)$$

- ***Decryption:***

$$(b_1 + a_1 s \bmod 1 + z^n) \bmod q = m_1 + te_1$$

- ***Homomorphic Addition:***

$$\mathsf{Enc}(m_1 + m_2) = (a_{\mathsf{add}} = a_1 + a_2, b_{\mathsf{add}} = b_1 + b_2)$$
$$\mathsf{Enc}(m_1 + m_2) = (a_{\mathsf{add}}, b_{\mathsf{add}} = -a_{\mathsf{add}} s + t(e_1 + e_2) + (m_1 + m_2))$$

Bonus slides: An example of a simple BGV-type HE

- ***Consider two encryptions:***

$$\mathsf{Enc}(m_1) = (a_1, b_1 = -a_1 s + te_1 + m_1)$$
$$\mathsf{Enc}(m_2) = (a_2, b_2 = -a_2 s + te_2 + m_2)$$

- ***Decryption:***

$$(b_1 + a_1 s \bmod 1 + z^n) \bmod q = m_1 + te_1$$

- ***Homomorphic Addition:***

$$\mathsf{Enc}(m_1 + m_2) = (a_{\mathsf{add}} = a_1 + a_2, b_{\mathsf{add}} = b_1 + b_2)$$
$$\mathsf{Enc}(m_1 + m_2) = (a_{\mathsf{add}}, b_{\mathsf{add}} = -a_{\mathsf{add}} s + t(e_1 + e_2) + (m_1 + m_2))$$

Bonus slides: An example of a simple BGV-type HE

- ***Consider two encryptions:***

$$\mathsf{Enc}(m_1) = (a_1, b_1 = -a_1 s + te_1 + m_1)$$
$$\mathsf{Enc}(m_2) = (a_2, b_2 = -a_2 s + te_2 + m_2)$$

- ***Decryption:***

$$(b_1 + a_1 s \bmod 1 + z^n) \bmod q = m_1 + te_1$$

- ***Homomorphic Addition:***

$$\mathsf{Enc}(m_1 + m_2) = (a_{\mathsf{add}} = a_1 + a_2, b_{\mathsf{add}} = b_1 + b_2)$$
$$\mathsf{Enc}(m_1 + m_2) = (a_{\mathsf{add}}, b_{\mathsf{add}} = -a_{\mathsf{add}} s + t(e_1 + e_2) + (m_1 + m_2))$$

Bonus slides: An example of a simple BGV-type HE

- ***Consider two encryptions:***

$$\mathsf{Enc}(m_1) = (a_1, b_1 = -a_1 s + te_1 + m_1)$$
$$\mathsf{Enc}(m_2) = (a_2, b_2 = -a_2 s + te_2 + m_2)$$

- ***Decryption:***

$$(b_1 + a_1 s \bmod 1 + z^n) \bmod q = m_1 + te_1$$

- ***Homomorphic Multiplication:***
  - It is slightly more complicated

$$\mathsf{Enc}(m_1 m_2) = (a_{\mathsf{mult}}, b_{\mathsf{mult}}, c_{\mathsf{mult}}) = (a_1 a_2, a_1 b_2 + a_2 b_1, b_1 b_2)$$

  - The number of polynomial elements increases. Decryption is now:

$$(c_{\mathsf{mult}} + b_{\mathsf{mult}} s + a_{\mathsf{mult}} s^2 \bmod 1 + z^n) \bmod q = m_1 m_2 + te_{\mathsf{mult}}$$

Bonus slides: An example of a simple BGV-type HE

- ***Consider two encryptions:***

$$\mathsf{Enc}(m_1) = (a_1, b_1 = -a_1 s + te_1 + m_1)$$
$$\mathsf{Enc}(m_2) = (a_2, b_2 = -a_2 s + te_2 + m_2)$$

- ***Decryption:***

$$(b_1 + a_1 s \bmod 1 + z^n) \bmod q = m_1 + te_1$$

- ***Homomorphic Multiplication:***
  - It is slightly more complicated

$$\mathsf{Enc}(m_1 m_2) = (a_{\mathsf{mult}}, b_{\mathsf{mult}}, c_{\mathsf{mult}}) = (a_1 a_2, a_1 b_2 + a_2 b_1, b_1 b_2)$$

  - The number of polynomial elements increases. Decryption is now:

$$(c_{\mathsf{mult}} + b_{\mathsf{mult}} s + a_{\mathsf{mult}} s^2 \bmod 1 + z^n) \bmod q = m_1 m_2 + te_{\mathsf{mult}}$$

Bonus slides: An example of a simple BGV-type HE

- ***Homomorphic Multiplication:***
  - It is slightly more complicated

$$\mathsf{Enc}(m_1 m_2) = (a_{\mathsf{mult}}, b_{\mathsf{mult}}, c_{\mathsf{mult}}) = (a_1 a_2, a_1 b_2 + a_2 b_1, b_1 b_2)$$

  - The number of polynomial elements increases. Decryption is now:

$$(c_{\mathsf{mult}} + b_{\mathsf{mult}} s + a_{\mathsf{mult}} s^2 \bmod 1 + z^n) \bmod q = m_1 m_2 + t e_{\mathsf{mult}}$$

  - “Relinearization step” is used to relinearize the “decryption circuit”:

$$\mathsf{Relinearization}\,(a_{\mathsf{mult}}, b_{\mathsf{mult}}, c_{\mathsf{mult}}) = (a_{\mathsf{relin}}, b_{\mathsf{relin}})$$

$$(b_{\mathsf{relin}} + a_{\mathsf{relin}} s \bmod 1 + z^n) \bmod q = m_1 m_2 + t(e_{\mathsf{mult}} + e_{\mathsf{relin}})$$

Proposed solution: some extra details

- The distributed decryption introduces an extra error component

$$e_{\text{distributed}} = \lfloor as \rceil_p - \sum_i \lfloor as_i \rceil_p$$

- It can be removed with an additional rounding phase ($q > p' > p$)

$$\Pr(\text{Ev}) \leq \frac{2 \cdot n \cdot N_{\text{AggRounds}} \cdot N_{\text{Ctxts.PerRound}} \cdot p' \cdot B_{\text{Agg}}}{q}$$

$$q \geq 4 \cdot n^2 \cdot N_{\text{AggRounds}} \cdot N_{\text{Ctxts.PerRound}} \cdot p \cdot L^2 \cdot B_{\text{Init}}^2 \cdot 2^\kappa$$

Input per DO	Decryption share per DO	Aggregator output	Decrypted result
$N_{\text{ModelParam}} \cdot \log_2 q$	$N_{\text{ModelParam}} \cdot \log_2 p'$	$N_{\text{ModelParam}} \cdot \log_2 p'$	$N_{\text{ModelParam}} \cdot \log_2 p$

Table 2. Communication costs per party in each aggregation round.

Proposed solution: some extra details

- The distributed decryption introduces an extra error component

$$e_{\text{distributed}} = \lfloor as \rceil_p - \sum_i \lfloor as_i \rceil_p$$

- It can be removed with an additional rounding phase ($q > p' > p$)

$$\Pr(\text{Ev}) \leq \frac{2 \cdot n \cdot N_{\text{AggRounds}} \cdot N_{\text{Ctxts.PerRound}} \cdot p' \cdot B_{\text{Agg}}}{q}$$

$$q \geq 4 \cdot n^2 \cdot N_{\text{AggRounds}} \cdot N_{\text{Ctxts.PerRound}} \cdot p \cdot L^2 \cdot B_{\text{Init}}^2 \cdot 2^{\kappa}$$

Input per DO	Decryption share per DO	Aggregator output	Decrypted result
$N_{\text{ModelParam}} \cdot \log_2 q$	$N_{\text{ModelParam}} \cdot \log_2 p'$	$N_{\text{ModelParam}} \cdot \log_2 p'$	$N_{\text{ModelParam}} \cdot \log_2 p$

Table 2. Communication costs per party in each aggregation round.

Proposed solution: some extra details

- The distributed decryption introduces an extra error component

$$e_{\text{distributed}} = \lfloor as \rceil_p - \sum_i \lfloor as_i \rceil_p$$

- It can be removed with an additional rounding phase (*q* > *p'* > *p*)

$$\Pr(\text{Ev}) \leq \frac{2 \cdot n \cdot N_{\text{AggRounds}} \cdot N_{\text{Ctxts.PerRound}} \cdot p' \cdot B_{\text{Agg}}}{q}$$

$$q \geq 4 \cdot n^2 \cdot N_{\text{AggRounds}} \cdot N_{\text{Ctxts.PerRound}} \cdot p \cdot L^2 \cdot B_{\text{Init}}^2 \cdot 2^{\kappa}$$

Input per DO	Decryption share per DO	Aggregator output	Decrypted result
$N_{\text{ModelParam}} \cdot \log_2 q$	$N_{\text{ModelParam}} \cdot \log_2 p'$	$N_{\text{ModelParam}} \cdot \log_2 p'$	$N_{\text{ModelParam}} \cdot \log_2 p$

Table 2. Communication costs per party in each aggregation round.