Evaluating Boolean circuits over ciphertexts using Fully Homomorphic Encryption over the Torus

Kotaro Matsuoka

Novembe/8/2022

Self Introduction

- Affiliation: 2nd grade Master's student at Kyoto University
- Certified as a "super creator" at IPA's MITOU program 2019.
- Gave lectures about TFHE at IPA's Security Camp 2020-2022.
  - Today's talk based on this lecture.
- Won the NHK Robot Contest 2019.

Figure 1: One of the robots we made in NHK Robot Contest.

Introduction

Classification of HEs

- Homomorphic Encryption (HE) = A form of encryption that permits encrypted data to be evaluated by arbitrary functions without decryption.
- Partially Homomorphic Encryption (PHE)
  - Support addition OR multiplication. (ex.: RSA)
- Somewhat Homomorphic Encryption (SHE)
  - PHE + a scheme-dependent number of additions or multiplications. (ex.: Lifted ElGamal)
- Leveled Homomorphic Encryption (LHE)
  - PHE + a security parameter-depended number of additions or multiplications.
  - Sometimes referred to as FHE. (The theoretical upper limit is not known.)
- Fully Homomorphic Encryption (FHE)
  - Supports any operations. (ex.: TFHE)

Lineage of HEs

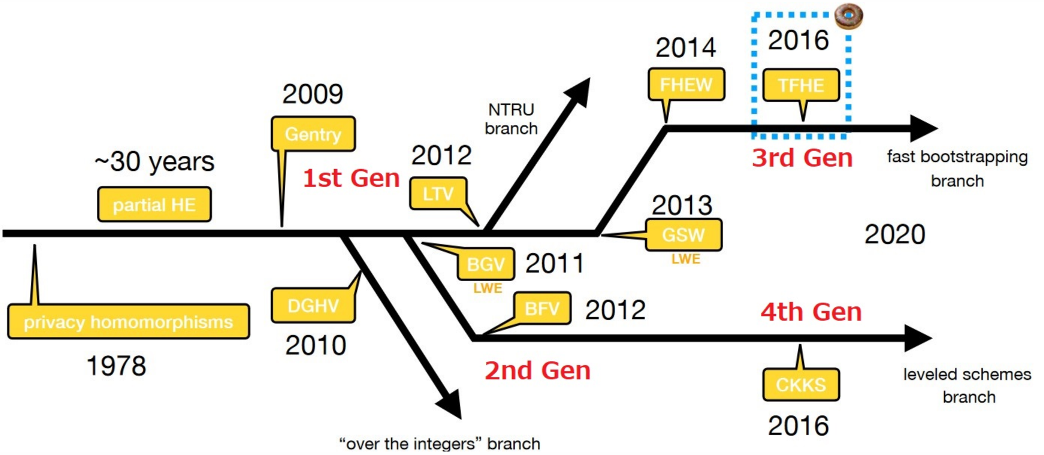


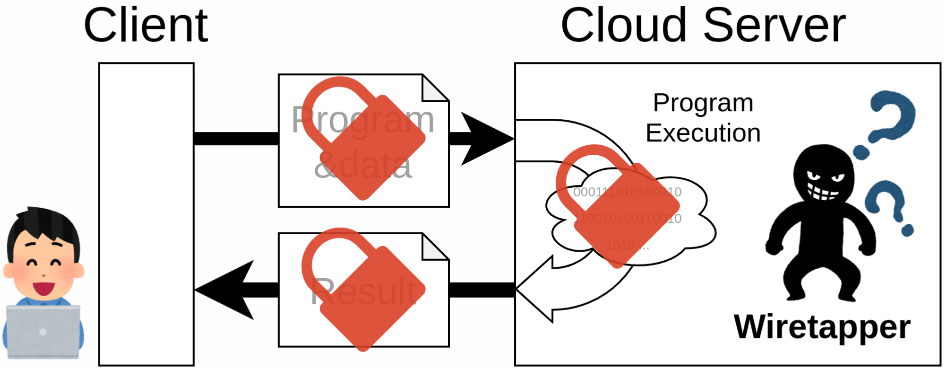
Figure 2:
https://drive.google.com/file/d/1aJCfhIpAk8unQ8BKof3C3cHlVzse1qpD/view,
Added red texts

What is TFHE?

- TFHE = Fully Homomorphic Encryption over the Torus
  - Pros:
    - Suitable for logic circuit evaluations. (Enable to use of existing synthesis tools.)
    - Fast Bootstrapping operations. ($<$ 10ms on the latest consumer grade CPUs)
  - Cons:
    - Relatively slow for linear operations. (Vector additions and multiplications.)

Virtual Secure Platform (USENIX Security 2021)

- One of the possible application of TFHE is evaluating a general processor.
  - Since the general processor represents a program as data, we can encrypt the program.
  - Theoretically highest tamper resistant capability.



Inside TFHE

The overview of HomNAND

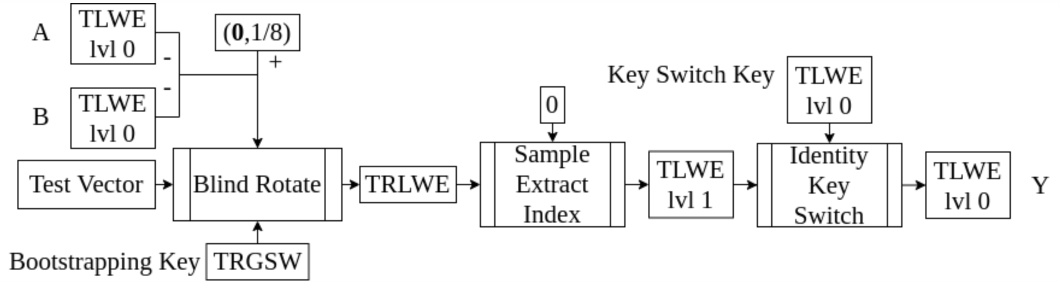


Figure 4: Block diagram of HomNAND

Torus($\mathbb{T}$, Circle group, 円周群)

- Def.: The group of angles. $\mathbb{T} = \mathbb{R} \bmod 1 \in [-0.5, 0.5)$
- The addition is defined but the multiplication is not.
  - ex.: $0.8 + 0.6 = 1.4 \equiv 0.4 \bmod 1, 0.3 - 0.9 = -0.6 \equiv 0.4 \bmod 1$
  - ex.: $1.2 \equiv 0.2 \bmod 1, 2.4 \equiv 0.4 \bmod 1$ but, $1.2 \cdot 2.4 = 2.88 \not\equiv 0.2 \cdot 0.4 = 0.08 \bmod 1$

TLWE

- TLWE = Torus Learning With Error
  - Most Post Quantum Cryptography uses Integer LWE.
  - Discretizing Torus by fixed point integer gives the same implementation and security.
- Notations:
  - The set of Boolean: $\mathbb{B} = \{0,1\} \in \mathbb{Z}$
  - Security parameters: $n \in \mathbb{Z}^+, \alpha \in \mathbb{R}^+$
  - Modular Gaussian distribution: $\mathcal{D}_{\mathbb{T},\alpha} = N(0,\alpha^2) \bmod 1$, Gaussian dist. over Torus.
  - $\mathbf{a} \in \mathbb{T}^n, e(\text{error, noise}), b \in \mathbb{T}, \mathbf{s} \leftarrow U(\mathbb{B}^n)(\text{Secret Key}), m(\text{plaintext}) \in \mathbb{T}$
  - $\leftarrow$: $x \leftarrow D$ means x itself or its entries or coefficients are drawn from D.
- TLWE ciphertext: $(\mathbf{a}, b) \in \mathbb{T}^{n+1}$
  - $\mathbf{a} \leftarrow U(\mathbb{T}^n), e \leftarrow \mathcal{D}_{\mathbb{T},\alpha}, \mathbf{s} \leftarrow U(\mathbb{B}^n), b = a \cdot \mathbf{s} + m + e$

The idea of 2-input NAND gate evaluation

- Since NAND takes Boolean inputs, the plaintext of TLWE should be Boolean.
  - Encode Boolean into Torus by $-\frac{1}{8}$(representing 0) and $\frac{1}{8}(1)$.'
  - Decryption under this encoded message space uses a sign function.
- c_0, c_1: TLWE input ciphertexts. $c_y = (\mathbf{0}, \frac{1}{8}) - c_0 - c_1$: The output.
  - The Torus plaintext of c_y is in $\{-\frac{1}{8}, \frac{1}{8}, \frac{3}{8}\}$.
  - Iff both ciphertexts encrypting $\frac{1}{8}$, $c_y = -\frac{1}{8} < 0$.
    - The decryption result of c_y becomes 0, iff both inputs are 1.

Bootstrapping

- Evaluating decryption function over ciphertexts.
  - Proposed by Gentry's Ph.D thesis in 2009.
  - By BOOTSTRAPPING, we can remove errors in ciphertexts.
    - We can evaluate more homomorphic operations after BOOTSTRAPPING.
  - All widely known FHE uses this idea.

The idea of constructing Bootstrapping in TFHE

- $\mathbb{T}_N[X]$: The ring of Torus coefficient polynomials $\bmod X^N + 1$.
- Test Vector (of the sign function): $TV[X] = \sum_{i=0}^{N-1} \frac{1}{8} X^i \in \mathbb{T}_N[X]$
- Figure 5 shows the negacyclic behavior on $\mathbb{T}_N[X]$.
- The constant term of $X^{\lceil 2N \cdot (b - \mathbf{a} \cdot \mathbf{s}) \rfloor} \cdot TV[X]$ is the plaintext of $(\mathbf{a}, b)$.
  - Homomorphic evaluation of $X^{\lceil 2N \cdot (b - \mathbf{a} \cdot \mathbf{s}) \rfloor}$ is the key idea.

$X^{-3} \cdot TV[X]$

$\frac{1}{8}$	$\frac{1}{8}$	$\frac{1}{8}$	$\frac{1}{8}$	$\frac{1}{8}$	$-\frac{1}{8}$	$-\frac{1}{8}$	$-\frac{1}{8}$

$X^{-N} \cdot TV[X]$

$-\frac{1}{8}$	$-\frac{1}{8}$	$-\frac{1}{8}$	$-\frac{1}{8}$	$-\frac{1}{8}$	$-\frac{1}{8}$	$-\frac{1}{8}$	$-\frac{1}{8}$

$X^{-(3+N)} \cdot TV[X]$

$-\frac{1}{8}$	$-\frac{1}{8}$	$-\frac{1}{8}$	$-\frac{1}{8}$	$-\frac{1}{8}$	$\frac{1}{8}$	$\frac{1}{8}$	$\frac{1}{8}$

Figure 5: Negacyclic Rotation ($N = 8$)

The idea of homomorphic exponent evaluation (Blind Rotate)

- $\lceil 2N \cdot (b - \mathbf{a} \cdot \mathbf{s}) \rfloor \approx \lfloor 2N \cdot b \rceil - \sum_{i=0}^{n-1} \lceil 2N \cdot a_i \rfloor \cdot s_i = \rho$
  - Now we can evaluate rounding without $\mathbf{s}$.
- Because $\mathbf{s} \in \mathbb{B}^n$, we can evaluate $X^{\sum_{i=0}^{n-1} \lceil 2N \cdot a_i \rfloor \cdot s_i}$ by multiplexers.
  - Multiplying $X^{\lceil 2N \cdot a_i \rfloor \cdot si}$ =Select multiplying $X^{\lceil 2N \cdot a_i \rfloor}$ or not.
  - The multiplexer can be constructed by multiplication with s_i.

Algorithm 1 The idea of BLINDROTATE

Require: $(\mathbf{a}, b)$: TLWE, $\mathbf{s}$: the secret key of TLWE, $TV[X] \in \mathbb{T}_N[X]$

Ensure: $crot$: $X^{-\rho} \cdot TV[X]$

```
crot ← X^{-⌊2N·cin.b⌉} · TV[X]
for i = 0 to n − 1 do
    //crot ← s_i ? X^{⌈2N·cin.a_i⌋} · crot : crot
    crot ← ((X^{⌈2N·cin.a_i⌋} · crot) − crot) · s_i + crot
```

Necessary components for Bootstrapping

1. The ciphertext for a Torus coefficient polynomial.
   - $crot$ in algorithm 1 must be encrypted.
2. Homomorphic multiplication between a Boolean and a polynomial.
   - s_i have to be encrypted.
3. Homomorphic extraction of the constant term.
   - What we need as a result of Bootstrapping is TLWE.

TRLWE

- TRLWE: Torus Ring Learning With Error
  - Security parameters: $N \in \mathbb{Z}^+, \alpha_{bk} \in \mathbb{R}^+$
  - Secret Key for TRLWE: $S[X] \leftarrow U(\mathbb{B}_N[X])$
- TRLWE ciphertext:
  $(a[X], b[X]) \in (\mathbf{T}_N[X])^2, a[X] \leftarrow U(\mathbb{T}_N[X]), b[X] = a[X] \cdot S[X] + e[X]$

TRGSW

- TRGSW: Torus Ring Gentry-Sahai-Waters
- Supports multiplication with TRLWE as a Leveled HE.
  - TFHE uses this ciphertext to encrypt $\mathbf{s}$.
  - In general, it can encrypt an integer polynomial.

Scaling

- All ciphertexts in TFHE use Torus-based message spaces.
  - We want to encode s_i into Torus.
- Parameter (not security related): $Bg \in \mathbb{Z}^+$
- Scaling TRLWE by Bg and rounding into integer polynomials.

$$(\lceil Bg \cdot a[X] \rfloor, \lceil Bg \cdot b[X] \rfloor) \cdot \begin{pmatrix} \frac{s_i}{Bg} & 0 \\ 0 & \frac{s_i}{Bg} \end{pmatrix} = (a^r[X], b^r[X])$$

$$\approx s_i \cdot (a[X], b[X])$$

$$b^r[X] - a^r[X] \cdot S[X] = s_i(b[X] - a[X] \cdot S[X] + e_r[X])$$

Masking by zero ciphertexts

- Adding a vector of TRLWE ciphertexts to encrypt.
- $(a_1[X], b_1[X]), (a_2[X], b_2[X])$ are encryption of 0.
- The result of the inner product with ciphertexts of 0 is a ciphertext of 0.
  - Adding the vector introduces more errors but not changes the result plaintext.

$$(\lceil Bg \cdot a[X] \rfloor, \lceil Bg \cdot b[X] \rfloor) \cdot \left(\begin{array}{cc} \frac{s_i}{Bg} & 0 \\ 0 & \frac{s_i}{Bg} \end{array} \right) + \left(\begin{array}{cc} a_1[X] & b_1[X] \\ a_2[X] & b_2[X] \end{array} \right)] \approx$$
$$s_i \cdot (a[X], b[X]) + \lceil Bg \cdot a[X] \rfloor \cdot (a_1[X], b_1[X]) + \lceil Bg \cdot b[X] \rfloor \cdot (a_2[X], b_2[X])$$

Trade-off in the selection of Bg

- The max coefficient value of $\lceil Bg \cdot a[X] \rfloor, \lceil Bg \cdot b[X] \rfloor$ is Bg.
    - The bigger Bg means more errors in the resulting ciphertext.
- However, decreasing Bg means bigger rounding errors.
- $\textsc{Decomposition}$ is the idea to avoid this trade-off.

Decomposition

- Parameter (not security related): $l \in \mathbb{Z}^+$
- Decomposition takes a TRLWE ciphertext as the input.
  - Returns $(\bar{a}_i[X], \bar{b}_i[X]) \in ((\mathbb{Z}/Bg)[X])^{2l}, i \in (1, l)$
  - The coefficients of $(\bar{a}_i[X], \bar{b}_i[X])$ are ith digits of $(a[X], b[X])$ in the base Bg.

$$(a[X], b[X]) \approx (\bar{a}_1[X], ..., \bar{a}_l[X], \bar{b}_1[X], ..., \bar{b}_l[X]) \begin{pmatrix} \frac{1}{Bg} & 0 \\ \frac{1}{Bg^2} & 0 \\ \vdots & \vdots \\ \frac{1}{Bg^l} & 0 \\ 0 & \frac{1}{Bg} \\ 0 & \frac{1}{Bg^2} \\ \vdots & \vdots \\ 0 & \frac{1}{Bg^l} \end{pmatrix}$$

- We assume applying DECOMPOSITION to the TRLWE ciphertext to be multiplied.

$$
\begin{pmatrix}
\frac{s_i}{Bg} & 0 \\
\frac{s_i}{Bg^2} & 0 \\
\vdots & \vdots \\
\frac{s_i}{Bg^l} & 0 \\
0 & \frac{s_i}{Bg} \\
0 & \frac{s_i}{Bg^2} \\
\vdots & \vdots \\
0 & \frac{s_i}{Bg^l}
\end{pmatrix}
+
\begin{pmatrix}
a_1[X] & b_1[X] \\
a_2[X] & b_2[X] \\
\vdots & \vdots \\
a_l[X] & b_l[X] \\
a_{l+1}[X] & b_{l+1}[X] \\
a_{l+2}[X] & b_{l+2}[X] \\
\vdots & \vdots \\
a_{2l}[X] & b_{2l}[X]
\end{pmatrix}
$$

Trade-off in the selection of l

- Increasing l exponentially reduces rounding errors but linearly amplifies 0 ciphertexts errors.
- Increasing l means more polynomial multiplications.
    - The heaviest computation part in TFHE.

Breaking down homomorphic constant term extraction

- The homomorphic constant term extraction can be divided into 2 parts.
- Notation:
  - TLWElvl0: The $n+1$ degree TLWE ciphertext. (Already introduced one.)
  - TLWElvl1: The $N+1$ degree TLWE ciphertext. (The secret key is different.)

1. Sample Extract Index: Convert TRLWE into TLWElvl1.
2. Identity Key Switching: Convert TLWElvl1 into TLWElvl0.

Sample Extract Index

- Careful look for the decryption of TRLWE gives the idea.
- The minus sign of the third term comes from the negacyclic property of $\mathbb{T}_N[X]$.

$$b[X]-a[X]\cdot S[X]=\sum_{k=0}^{N-1}[b_k-(\sum_{i+j=k,0\leq i,j\leq N-1}a_i\cdot S_j)-(\sum_{i+j=N+k,0\leq i,j\leq N-1}-a_i\cdot S_j)]X^k$$

- The idea is to fix k in the above formula.
  - Regard $\mathbf{S}$ (the vector of coefficients) as the key.
  - By setting $k=0$, we can get the constant term.
  - $(\mathbf{a}',b')$: A TLWElvl1 ciphertext encrypting the kth coefficient.

$$b'=b_k$$

$$a'_i=\begin{cases}a_{k-i} \text{ if } i\leq k\\ -a_{N+k-i} \text{ otherwise}\end{cases}$$

IdentityKeySwitching

- "Key Switching" means changing the secret key without decryption.
  - In general, "Key Switching" can evaluate linear function at the same time.
  - "Identity" means we evaluate the identity function in this case.
- The idea is to compute $b - \mathbf{a} \cdot \mathbf{S}$ directly.
  - We can reuse the idea of Scaling and $\textsc{Decomposition}$.
- Notations:
  - Parameters: $base, t \in \mathbb{Z}^+$(similar to Bg, l respectively.)
  - $IKSK_{ij}$: The TLWElvl1 ciphertext encrypting $\frac{S_i}{base^j}$
  - $\bar{a}_{ij}$: jth digit of a_i in base $base$.

$$(0, b) - \sum_{i=0}^{N-1} \sum_{j=1}^{t} \bar{a}_{ij} \cdot IKSK_{ij}$$

The overview of HomNAND (again)

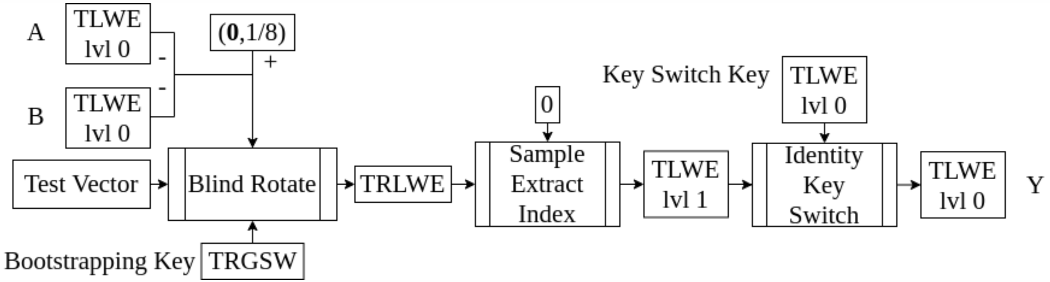


Figure 6: Block diagram of HomNAND

Parameter Selection

- The security parameter of TFHE affects the performance.
  - Unlike conventional ciphers, there is relatively high motivation to cut off the security margin.
- Current de facto tool for security estimation is "lwe-estimator".
  - `https://github.com/malb/lattice-estimator/`
  - Supporting dual/hybrid attack (`https://ia.cr/2020/515`).
  - Missing RLWE specific attacks. (AFAIK)
- Estimating error rate of decryption after computations is also necessary.
  - To reduce n, N, we have to increase α, α_{bk}.
  - concrete-npe seems to be best one for TFHE.
    `https://github.com/zama-ai/concrete-core/tree/main/concrete-npe`

Further Optimizations

- Using MLWE (Increasing the dimension of TRLWE.) or NTRU.
    - Reducing the degree of polynomials. (Ease polynomial multiplications.)
- Programmable Bootstrapping.
    - Supports more sophisticated non-linear functions than NAND. (ex.: Full Adder, ReLU)
    - My paper about compound gates. (WAHC2021) `https://doi.org/10.1145/3474366.3486927`
- TLWE (or TRGSW) to TRGSW bootstrapping (called Circuit Bootstrapping)
    - We can use TRGSW's multiplication as a base operation.
- Scheme switching. (Switching between TFHE and other HEs without decryption.)
    - We can use a suitable scheme for different operations.

The list of open-source TFHE implementations

- TFHE: The original implementation. Deprecated.
  - `https://github.com/tfhe/tfhe`
- Concrete: The authors' Rust implementation. Supporting integer operations.
  - `https://github.com/zama-ai/concrete.git`
- OpenFHE: Supporting multiple HEs. Aims to support scheme switching.
  - `https://github.com/openfheorg/openfhe-development`
- FINAL: NTRU based TFHE.`https://github.com/KULeuven-COSIC/FINAL`
- MOSFHET: Supports key compression and automorphism Blind Rotate.
  - `https://github.com/antoniocgj/MOSFHET`
- TFHEpp: My implementation. Supporting Circuit Bootstrapping.
  - `https://github.com/virtualsecureplatform/TFHEpp.git`
- cuFHE: CUDA implementation.
  - Forked ver.: `https://github.com/virtualsecureplatform/cuFHE.git`

Boolean Circuit Evaluations

Boolean circuit evaluations

- Combinational circuit is the Directed Acyclic Graph(DAG) of logic gates.
  - We can use DAG-based job scheduling for evaluations.
  - ex.: Taskflow(`https://github.com/taskflow/taskflow`), StarPU(`https://starpu.gitlabpages.inria.fr/`)
- Sequential circuits can be divided by registers into combinational circuits.
  - Just copy the inputs of registers to the outputs at the end of the cycle.
- The netlist (DAG) of the circuit can be obtained by conventional synthesis tools.
  - ex.: Yosys(`https://github.com/YosysHQ/yosys.git`)

Speed on real environments

- As an example, here is the evaluation time for VSP.
- Equipped with 512 bytes ROM and 512 bytes RAM.
- Pipelining degrades performance if the number of worker is not enough.
- At the best case, we achieve around 1.25 Hz evaluation.

Table 1: Performance Evaluation Using Hamming

Machine	Pipelining?	# of cycles	Runtime [s]	sec./cycle
AWS c5.metal	No	936	2342.0	2.502
	Yes	1216	2773.0	2.280
AWS p3.16xlarge	No	936	1440.0	1.538
	Yes	1216	**965.9**	**0.794**

The list of open-source Boolean circuit evaluation frameworks

- HDL-based ones:
  - Iyokan: `https://github.com/virtualsecureplatform/Iyokan`
    - Supports both CPU and GPU. CMUX Memory is integrated.
  - Sudachi: `https://github.com/virtualsecureplatform/Sudachi`
    - Taskflow based. Compound gates are supported.
- HLS-based ones:
  - Cingulta: `https://github.com/CEA-LIST/Cingulata`
    - Deprecated.
  - FHE Transpiler: `https://github.com/google/fully-homomorphic-encryption`
    - Actively developed. Using XLS as a HLS language.

Conclusion (Open questions)

- FHE is the holy grail but not the silver bullet.
  - The security only depends on the secret key.
  - Efficiency is generally lower than other privacy-preserving computing methods.
- Who is the winner of FHE?
  - Currently, CKKS is the mainstream because it is suitable for the private AI.
- How we can extend HE for multi-party settings?
  - There are few works about multi-key or threashold HEs.
- How we can resolve malleability?
  - Restricting possible computations is difficult.
  - Merging with verifiable computation? (Zero-knowledge proof)
- How about the hardware acceleration?
  - DARAP DPRIVE project