Accelerate Encrypt/Decrypt Operation in Functional Encryption

YSS TEAM

Members : Yi-Chuan Liang, Shin-Wei Chiu, Shan-Jung Hou

Advisor : Shih-Hao Hung

Performance, Application and Security Laboratory (PAS Lab)

Mentors : CK Lee, Frank Lin, Jay, Tian Zheng (NVIDIA)

National Taiwan University

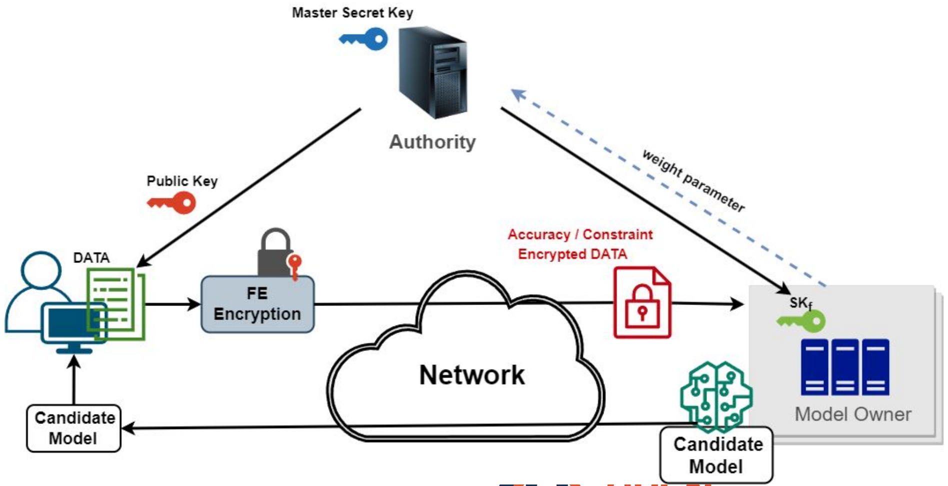
NARLabs 國家實驗研究院
國家高速網路與計算中心
National Center for High-performance Computing

OPEN HACKATHONS
OpenACC

Functional Encryption apply in Machine Learning Service

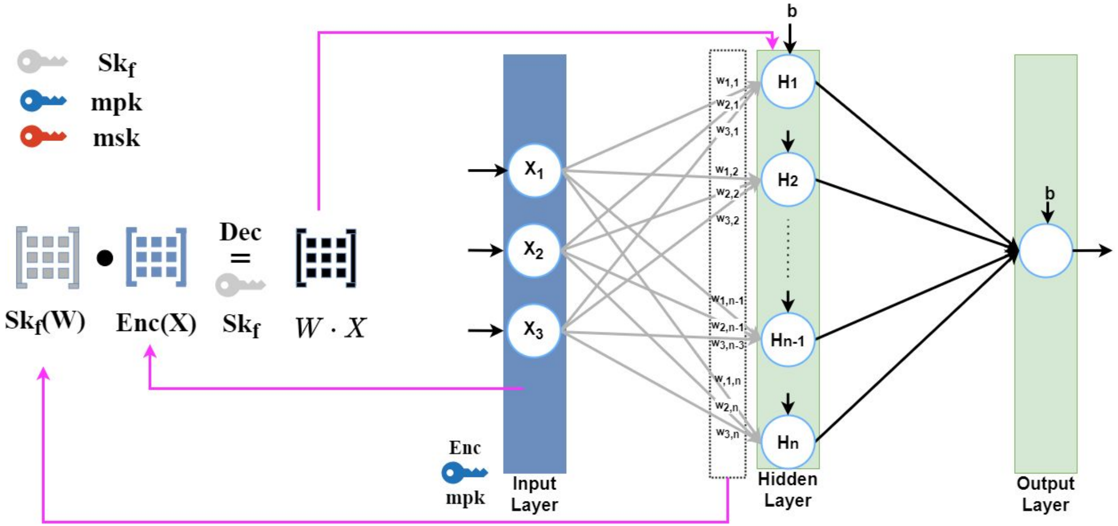
- Machine learning as a Service (MaaS) is apply for medical, financial, or other types of sensitive data, not only require accurate prediction but also careful to maintaining data privacy and security.
- It is important for server to provide fast and secure service
- Functional Encryption is one of the cryptosystem mechanisms used for data privacy In MaaS.

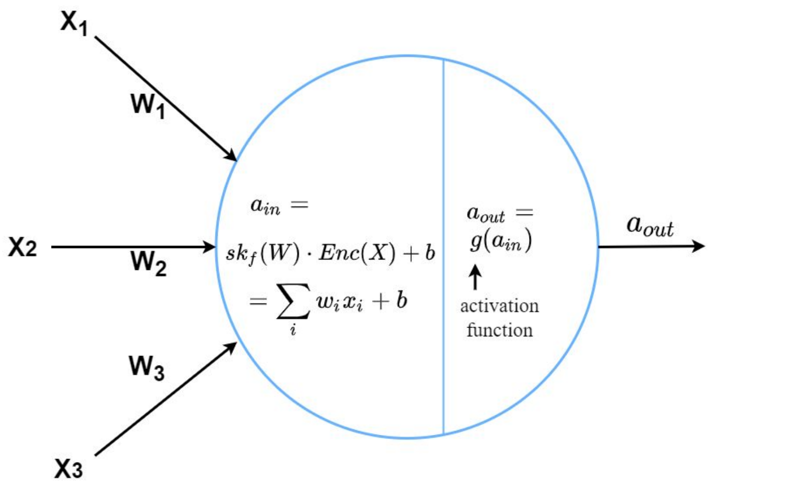
- What's the algorithmic motif?
- What parts are you focusing on?



Functional Encryption apply in Machine Learning Service

- Using FE on deep learning operation
- It won’t affect 1st layer’s activation function and other layers’ operations





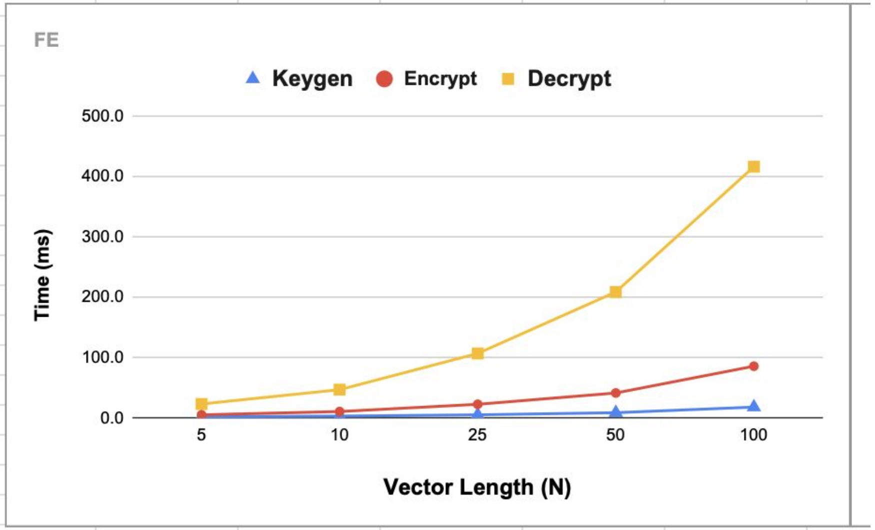
$$a = g(sk_f(W) \cdot enc(X) + b) = g(f(WX) + b)$$

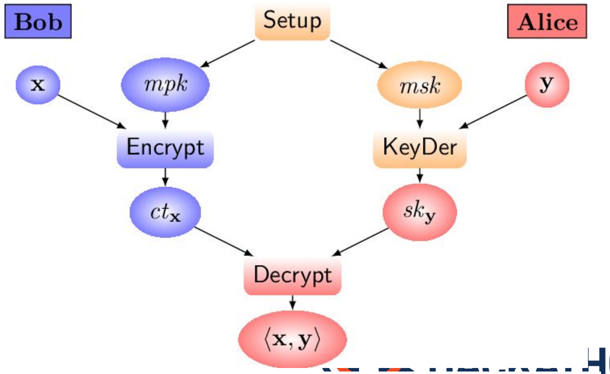
Profiling Functional Encryption Operation

- **Setup(1^λ,1^ℓ)**
  - (**G**,p,g)←GroupGen(1^ℓ), and **s**=(s_1, …, s_ℓ)←Z_p^{ℓ}
  - set *mpk*=$(h_i=g^{S_i})_{i \in [\ell]}$ and *msk*=**s**
- **Encrypt(*mpk*, x)**
  - choose a random r ←Z_p and compute $ct_0=g^r$,$ct_i=h_i^r \cdot g^{x_i}$, ∀i ∈ [ℓ]
  - return ct = (ct_0, ct_i) , ∀i ∈ [ℓ]
- **KeyDerive(*msk*, y)**
  - return secret key $sk_y=\langle y,s \rangle$
-
- **Decrypt(*mpk*, ct, sk_y)**

$$= \frac{\prod_{i\in[\ell]} ct_i^{y_i}}{ct_0^{sk_y}} = \frac{\prod_{i\in[\ell]}(g^{s_i r + x_i})^{y_i}}{g^{r(\sum_{i\in[\ell]} y_i s_i)}}$$
$$= g^{\sum_{i\in[\ell]} y_i s_i r + \sum_{i\in[\ell]} y_i x_i - r(\sum_{i\in[\ell]} y_i s_i)}$$
$$= g^{\sum_{i\in[\ell]} y_i x_i} = g^{\langle \mathbf{x},\mathbf{y}\rangle}.$$

- Need faster $\Pi_{i \in [\ell]} ct_i^{y_i}$ and logarithm operation





[12] Bourse, Florian. Functional encryption for inner-product evaluations. 2017.
[8] Abdalla, Michel, et al. "Simple functional encryption schemes for inner products." IACR International Workshop on Public Key Cryptography. Springer, Berlin, Heidelberg, 2015.

Evolution and Strategy

- What was your goal coming here ?
  - Speedup decryption operation in FE
- What was your initial strategy ?
  - Using GPU to parallel Baby-Step-Giant-Step Algorithm to solve discrete logarithm

Baby Step Giant Step Algorithm

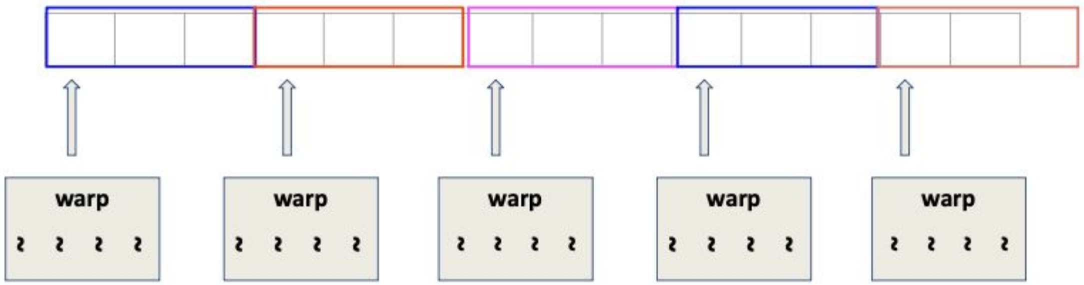
- Methodology
  - cuCollections (check)
  - XMP (not yet)

Algorithm 1: Baby Step Giant Step

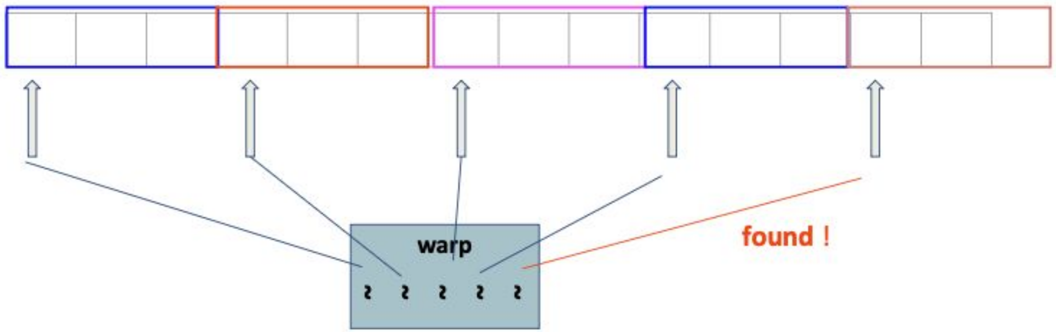
Input: g (generator), p (1024bit present prime),
Output: a

```
/* Know p, g and x=g^a mod p, find exponent a
/* Step1 Create a Giant step table in GPU share memory
m = √(p − 1), a = im + j where i, j ∈ {0 ... m − 1}
for i=0 to range √p do
    insert result of (xg^i mod p) in to Giant table
/* Step2 parallel compute baby step in threads
while unmatch in Giant table do
    compute result of( g^j mod p)
    j++
a = im+j
```

Giant step : Parallel Insert

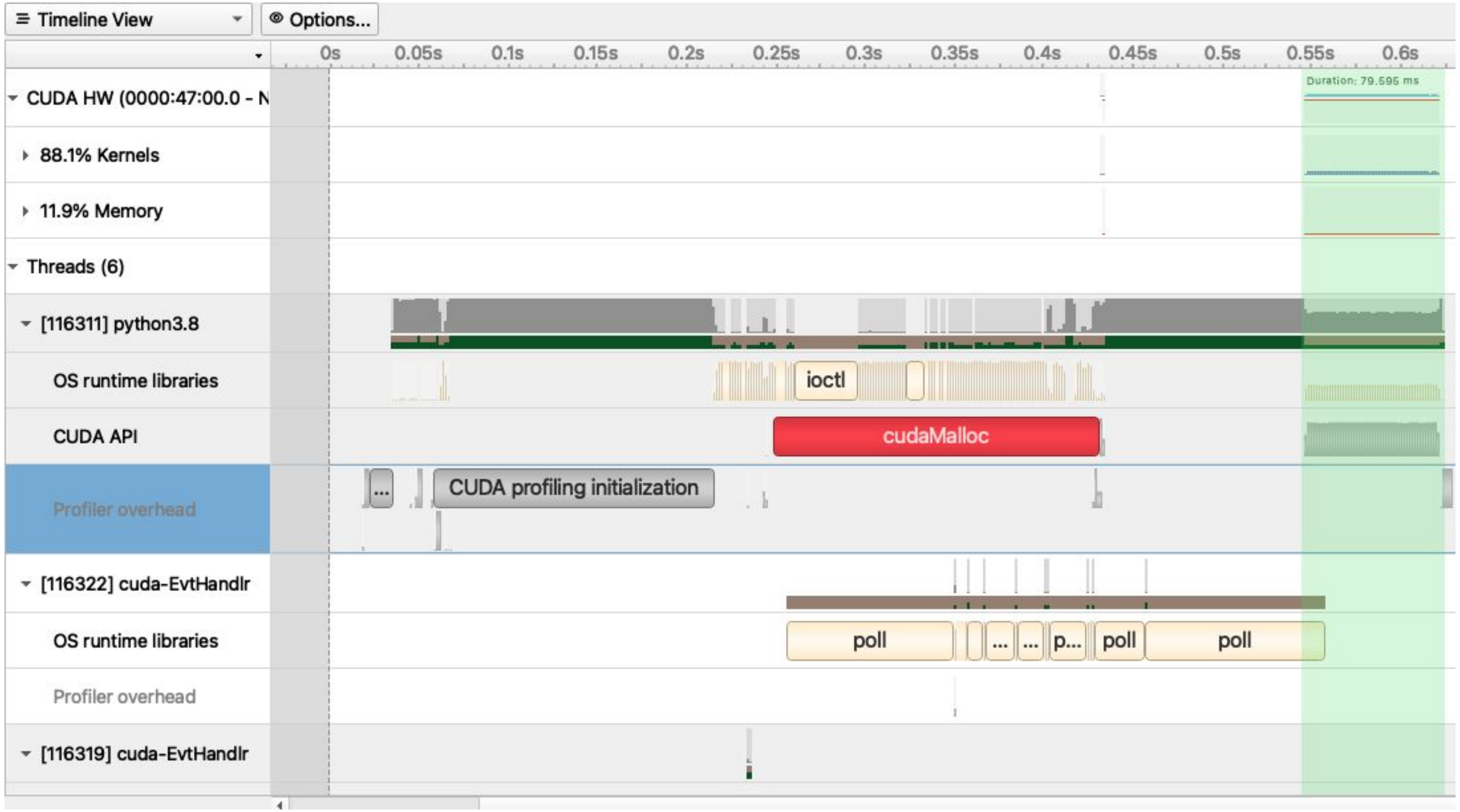


Baby step : Parallel Find



Nsight Profiling

- bit_length 32
- vector length 100
- Speedup 2.2

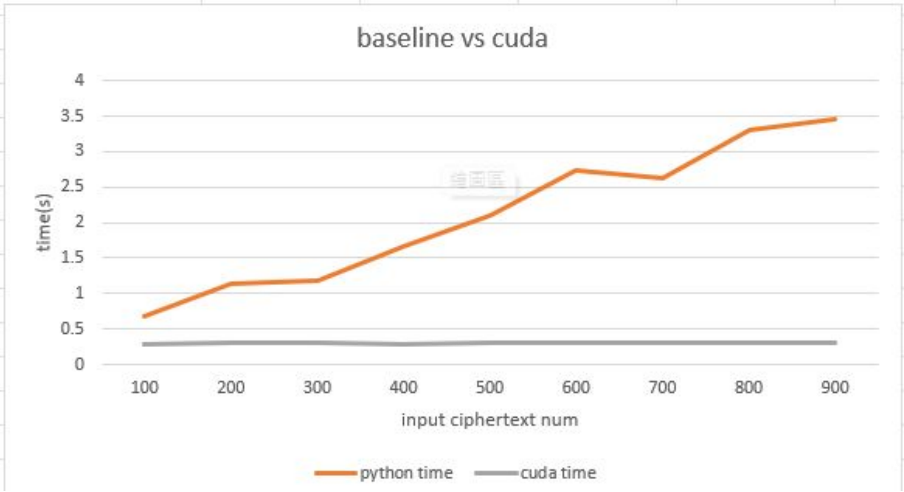


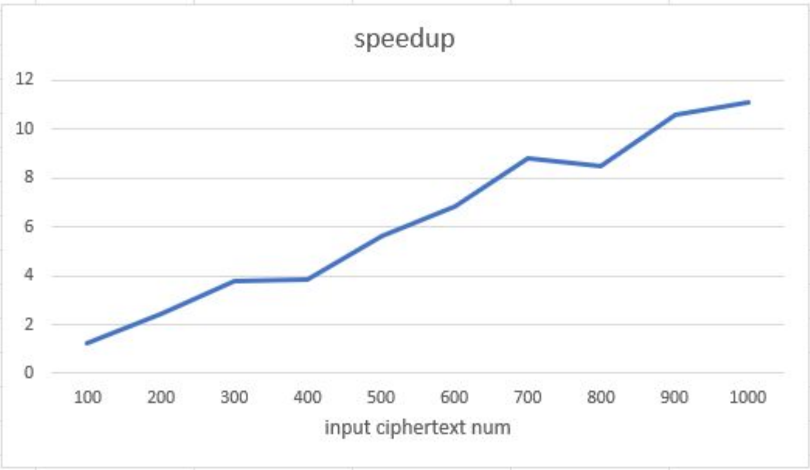
- Increase bit length and vector length will speed up more

Nsight Profiling

BSGS Experiment

- Increase input ciphertext length on Cuda kernel
- GPU A100





NARLabs 國家實驗研究院
國家高速網路與計算中心
National Center for High-performance Computing

OPEN HACKATHONS
OpenACC

FE+NN Experiment

- LeNet-5, Batch size 64
- Training time speedup 2.5x ~

CPU Training Time		
Framework	Enc	Training time
LeNet-5	-	4h
FE + LeNet (baseline)	3h	11h33m
parallel enc + parallel training	1h40m	6h6m
Parallel Enc + Opt Dec + Parallel training	1h40m	about 5h

Expect

Energy Efficiency

INPUTS	
# CPU Cores	128
# GPUs (A100)	1
Application Speedup	2.2x

Node Replacement	17.6x

GPU NODE POWER SAVINGS			
	AMD Dual Rome 7742	8x A100 80GB SXM4	Power Savings
Compute Power (W)	19,360	6,500	12,860
Networking Power (W)	817	93	724
Total Power (W)	**20,177**	**6,593**	13,584

Node Power efficiency	**3.1x**

ANNUAL ENERGY SAVINGS PER GPU NODE			
	AMD Dual Rome 7742	8x A100 80GB SXM4	Power Savings
Compute Power (kWh/year)	169,594	56,940	112,654
Networking Power (kWh/year)	7,159	814	6,346
Total Power (kWh/year)	**176,753**	**57,754**	**118,999**

$/kWh	$ 0.34
Annual Cost Savings	$ 40,459.82
3-year Cost Savings	$ 121,379.45

Metric Tons of CO_2	84
Gasoline Cars Driven for 1 year	18
Seedlings Trees grown for 10 years	1,395

(source: Link)

NARLabs 國家高速

National Center for High-performance Computing

OPEN HACKATHONS

OpenACC

Result and Final Profile

- What were you able to accomplish ?
  - In BSGS Algorithm, we got a two-time speedup methodology.
  - Increasing kernel utilization can increase the speed-up factor.

- What did you learn ?
  - Create a new algorithm?
    - Transform BSGS code to CUDA code.
    - Nsight and NVTX profiling tools

What problems have you encountered?

- Issues with algorithm
  - Increase kernel utilization
- Tool lack of features
  - cuCollections only support 64bit key not support XMP format as a key
- System setup
  - Co-compile with c, cuda, cuCollections, XMP and other libraries.

Future Work and Wishlist

- Will you continue development ?
  - Next Step : Replace GPM to XMP in CUDA kernel,
  - Future plans : Combine FE and MaaS
  - Compare Encryption operation with OpenMP version and OpenACC version.
- What sustained resources/support will be critical for your work after the event?
  - Support from CUDA programming technique knowledge from NVIDIA and computing resource from NCHC
- Wishlist
  - Automatic Tools (OpenMP -> OpenACC)
  - Lesson

Summary

- Bit_lengths bigger than 32bit are better for GPU parallel computing.
- In this Hackathon event, our team learned GPU parallel computing skills from mentors.
- We also use Nsight and NVTX Profiling tools to analyze programs for optimization
- Mentors give our team many helpful suggestions and research directions.
- We will keep going to Optimize our project.

Thank you