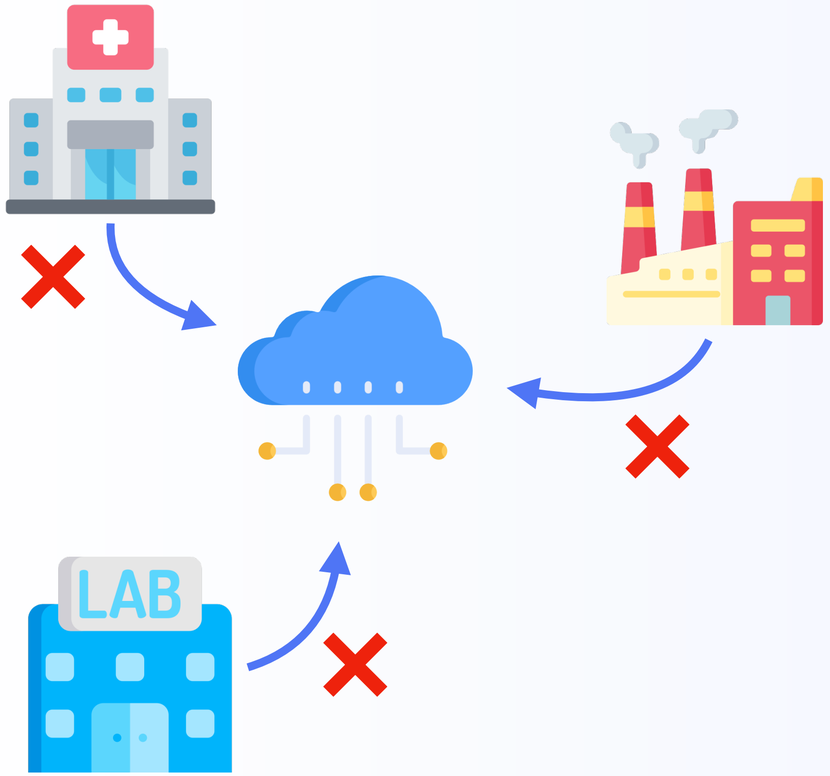
iSingLab HKUST

CLUSTAR 星云

FLASH: Towards High-performance Hardware Acceleration Architecture for Cross-silo Federated Learning

Junxue ZHANG, Xiaodian CHENG, Wei WANG, Liu YANG, Jinbin HU, Kai CHEN

Data Silos & Islands of Data



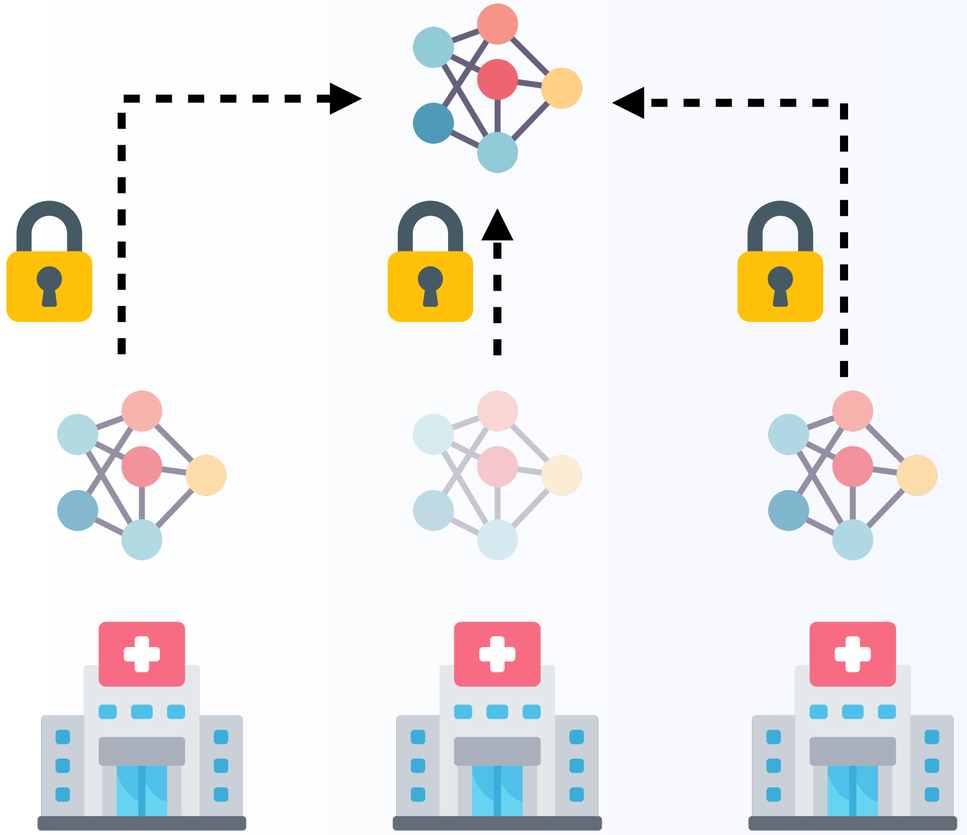
Emerging lawsuits and regulations **restrict** us to collect data into one central place for processing. Data from different entities become **island** and are isolated from each other.

Cross-silo Federated Learning

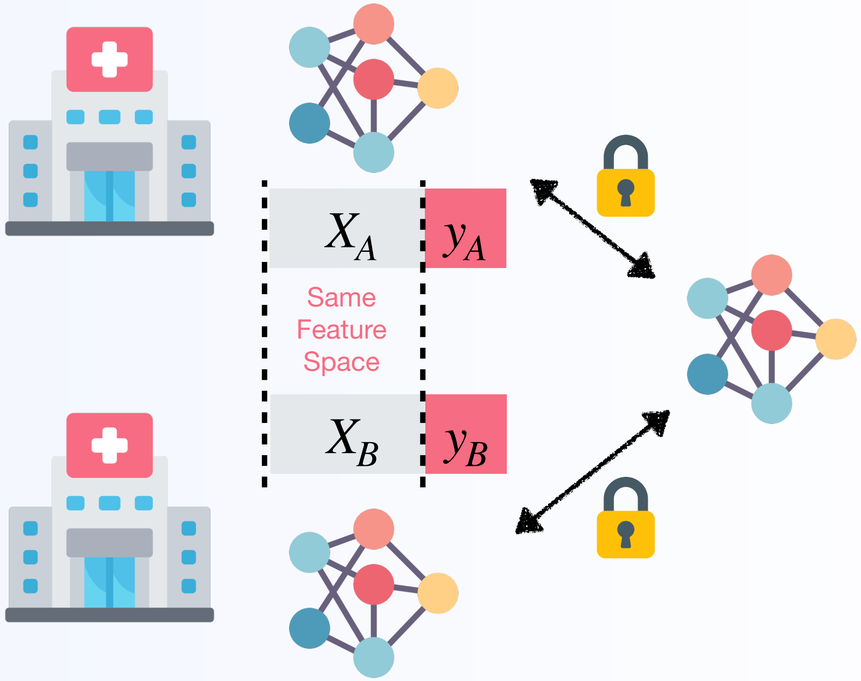
1. Each Participant trains a local model

2. Instead of sharing original data, all participants exchange their trained model

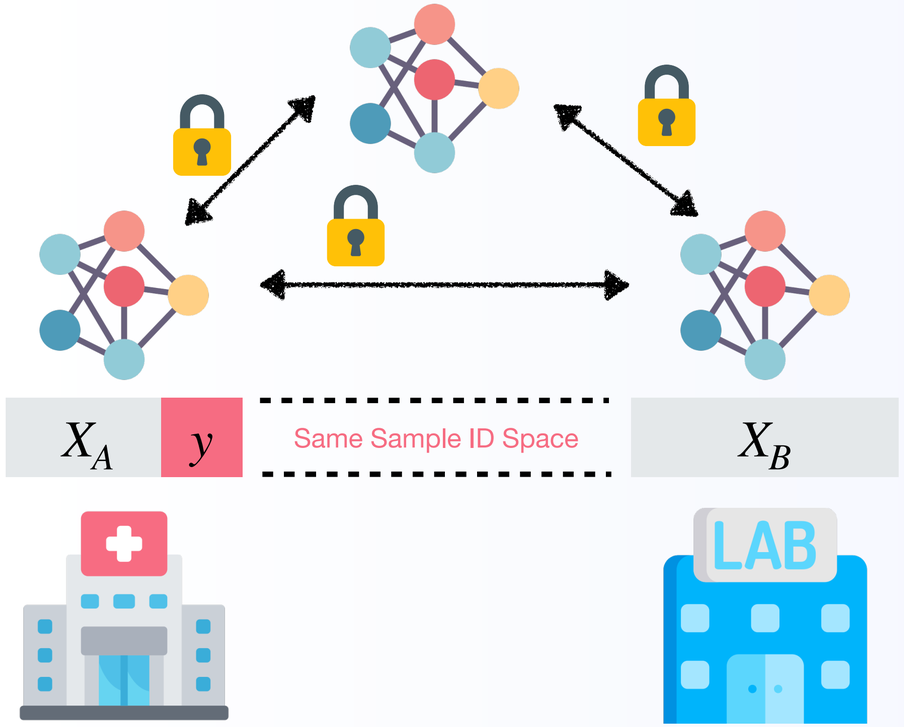
3. The model/intermediate results are protected via cryptographic techniques



Horizontal & Vertical Federated Learning



Horizontal Federated Learning



Vertical Federated Learning

Cryptographic Techniques

Examples of some used cryptographic techniques:

1. Additive HE, Paillier

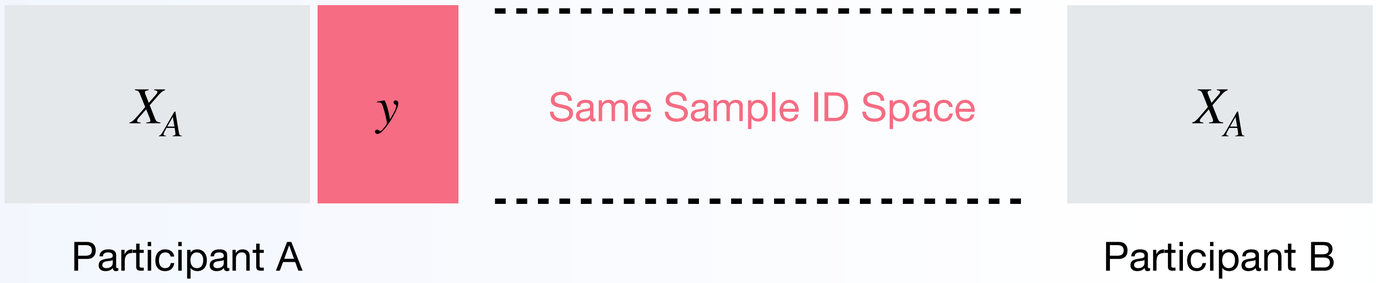
   Encryption Function $E(p)$

   If A, B, C are plaintext, then

   $A + B = C \rightarrow E(A) + E(B) = E(C)$

→ Horizontal FL
Vertical FL

2. RSA blind signature-based PSI



→ Vertical FL

Double-edged Sword — Cryptographic Operations

We identify 9 commonly used cryptographic operations in cross-silo FL

ID	Crytographic Operations
O1	Paillier Encryption
O2	Paillier Decryption
O3	Ciphertext Matrix Addition
O4	Ciphertext & Cleartext Matrix Element-wise Multiplication.
O5	Ciphertext & Cleartext Matrix Multiplication
O6	Ciphertext Histogram Building
O7	RSA Encryption
O8	RSA Blind
O9	RSA Unblind

→ Preserving Privacy

→ Performance Penalty

Our previous work *Quantifying the Performance of Federated Transfer Learning* has observed this problem and delivers a brief analysis

https://arxiv.org/abs/1912.12795

A Fine-grained Analysis

Applications & Their Sub-tasks		Involved Operations	w/o CO (s)	w CO (s)	Degradation
RSA-PSI	Computing intersection	**O7, O8, O9**	18.91	203.88	10.78× ↓
VLR (One Epoch) Total: 17.4× ↓	Encrypting logits	**O1**	0	242.09	-
	Aggregating logits	**O3**	6.67	9.81	1.47× ↓
	Computing fore gradients[a]	**O3, O4**	7.88	25.71	3.26 × ↓
	Computing gradients	**O3, O4, O5**	32.68	1550.02	47.43× ↓
	Decrypting gradients	**O2**	0	0.06	-
	Computing loss	**O1, O3, O4**	24.20	37.74	1.56 × ↓
SBT (One Epoch) Total: 2.59× ↓[b]	Encrypting gradients	**O1**	0	486.73	-
	Aggregating gradients	**O3, O6**	83.13	2125.50	25.57 × ↓
	Finding split	**O2**	0.78	24.71	31.51 × ↓

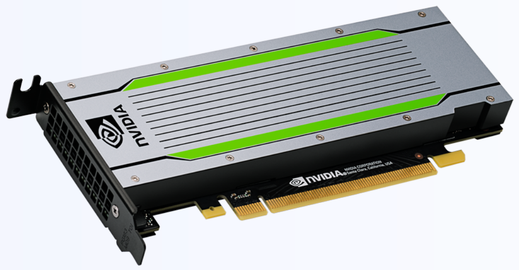
[a] According to Federated Logistic Regression [39], the gradient computation takes two steps: fore gradients computation and gradients computation.

[b] The overall performance degradation of SBT is smaller than the sum of those sub-tasks because we do not include SBT's pure cleartext computation or networking communication sub-tasks in the table.

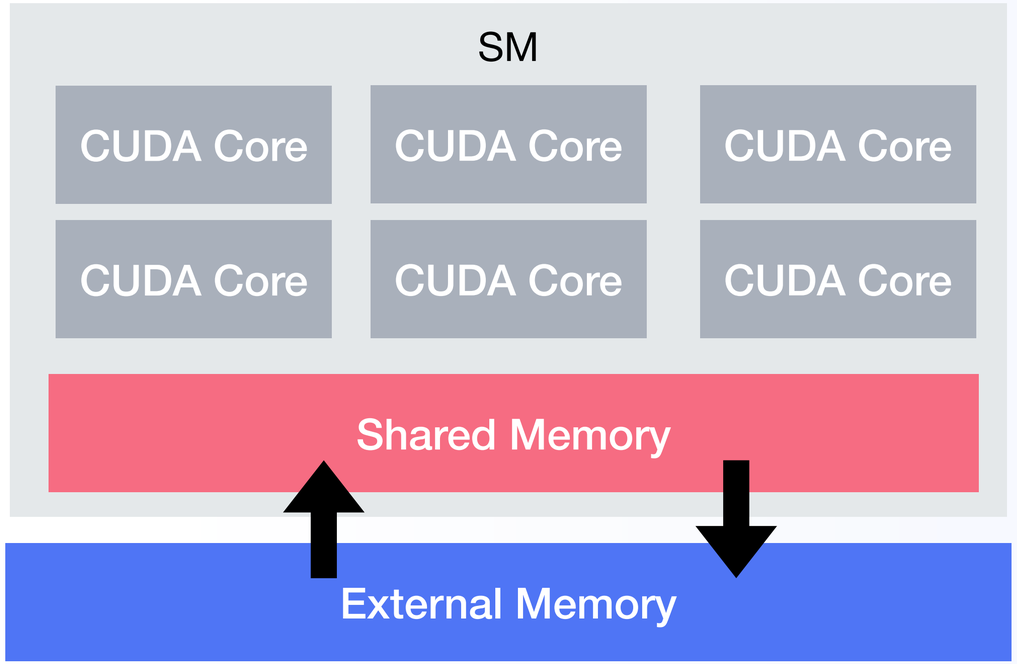
- All cryptographic operations do cause much performance penalty for cross-silo federated learning applications
- Different applications may use different cryptographic operations
- Even within one application, different cryptographic operations are used at different time

Hardware Offloading

Our research path:



GPU



Vendor-proprietary hardware GPU:

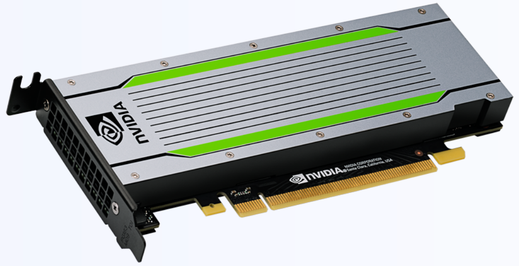
The hardware is designed for data parallel with small numbers, such as FP16, FP32.
Cross-silo FL requires pipeline parallelism with large integer numbers of 2048 bit and more.

Small amount of integers can be stored in shared memory of a SM

Pipeline execution pauses due to data exchange between shared and external memory

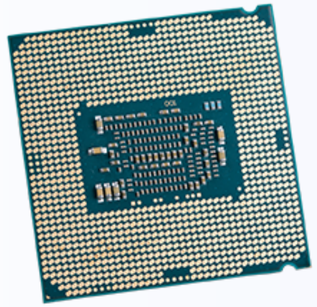
Hardware Offloading

Our research path:



GPU

FPGA as a prototype



ASIC as an ultimate form

We seek a more efficient hardware acceleration architecture beyond the existing GPU architecture

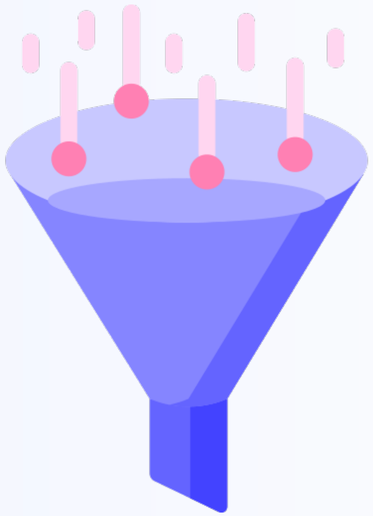
Pipeline Parallism

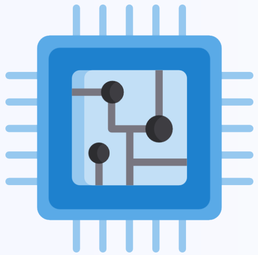
Inflated Data

Design a hardware pipeline that is suitable for efficiently executing these cryptographic operations

Challenges

Cryptographic Operations





Statically offloading all cryptographic operations to hardware leads to the following two problems:

Problem 1: The hardware chip has limited hardware resources to significantly accelerate all 9 operations

Problem 2: Hardware resource is wasted because not all operations are used at the same time

Our Observations

Paillier Encryption: given the public key (n, g), and data $m(0 \leq m < n)$, select a random $r, 0 < r < n, r \in \mathbb{Z}_n^*$, the ciphertext $c = g^m \cdot r^n \mod n^2$

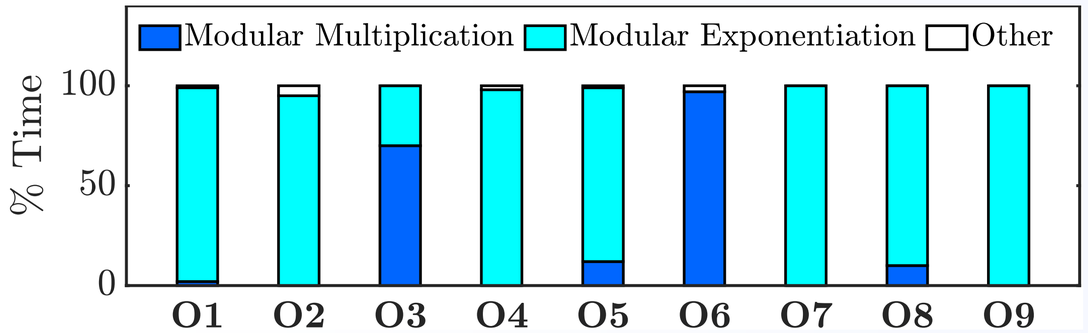
Addition: given ciphertext a and b, the addition is $a * b \mod n^2$

Ciphertext & plaintext multiplication: given ciphertext a and plaintext k, the multiplication is $a^k \mod n^2$

Similar operators are used in decryption and RSA-related operations

The core of these cryptographic operations is 2 basic operator: modular multiplication & exponentiation

Testbed experiments to breakdown the execution time of all cryptographic operations

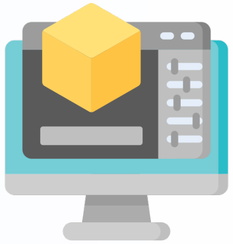


The performance of these operations mainly reply on the 2 basic operators

FLASH: a Cross-silo Federated Learning Acceleration Hardware Architecture

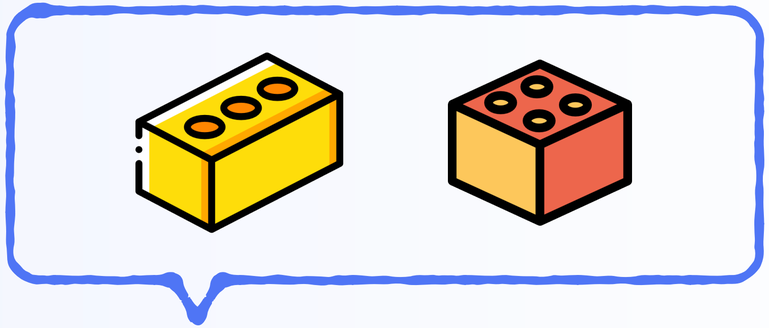
FLASH in One Slide

FATE



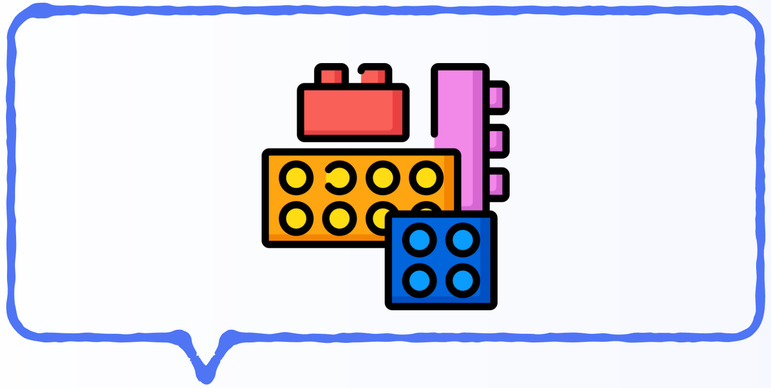
The software uses FLASH's API to offload cryptographic operations

Software Integration



FLASH uses majority of hardware resources to build 2 basic operators: modular multiplication & exponentiation

Modular Exponentiation & Multiplication Engine



FLASH dynamically composes the cryptographic operations with the basic operators.

Dataflow Scheduling

Modular Exponentiation & Multiplication Engine

$$P = m^e \mod N \qquad m, e, N \in \mathbb{Z}^+$$

$$P = ab \mod N \qquad a, b, N \in \mathbb{Z}^+$$

Binary Exponentiation & Montgomery Algorithm

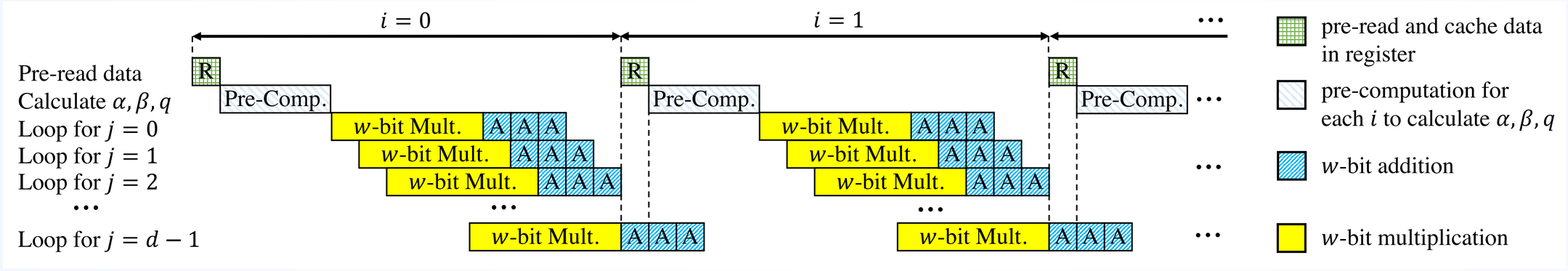
Algorithm 1: Montgomery Algorithm for Modular Multiplication with Radix 2^k

Input: $X = \sum_{j=0}^{l/k-1} X^j \cdot 2^{jk}$, $Y = \sum_{j=0}^{l/k-1} Y^j \cdot 2^{jk}$, $M = \sum_{j=0}^{l/k-1} M^j \cdot 2^{jk}$, $r = 2^k$

Output: $S = X \cdot Y/2^l \mod M$

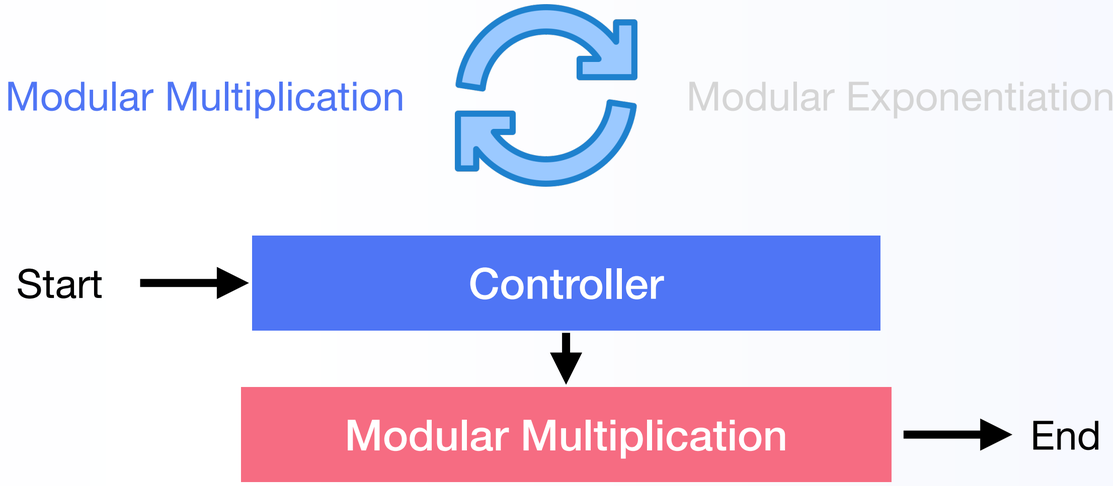
$S_0 \leftarrow 0$;
for $i = 0 \ldots l/k - 1$ **do**
  $q \leftarrow ((S_i + X * Y^i) \cdot (-M^{-1})) \mod r$;
  for $j = 0 \ldots l/k$ **do**
    $\bar{S}_{i+1}^j \leftarrow S_i^j + X^j * Y^i + q * M^j$;
  end
  $S_{i+1} \leftarrow \bar{S}_{i+1}/2^k$
end
if $S_{l/k} > M$ **then**
  $S_{l/k} \leftarrow S_{l/k} - M$;
end
return $S_{l/k}$

Construct pipeline



Dataflow Scheduling

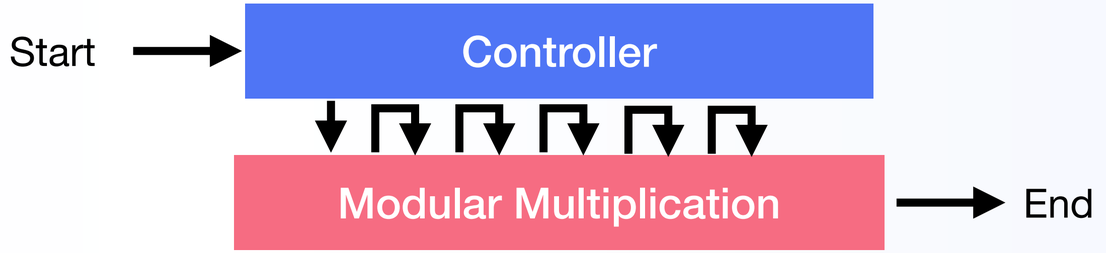
1. The engine can work in two modes. They can switch between modular multiplication and exponentiation



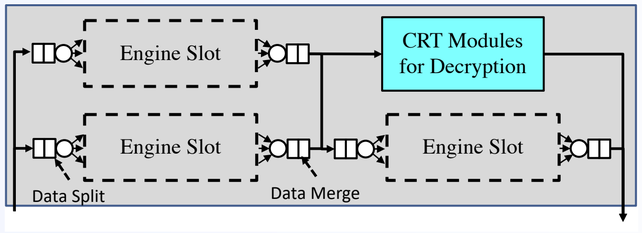
Dataflow Scheduling

Modular Multiplication  Modular Exponentiation

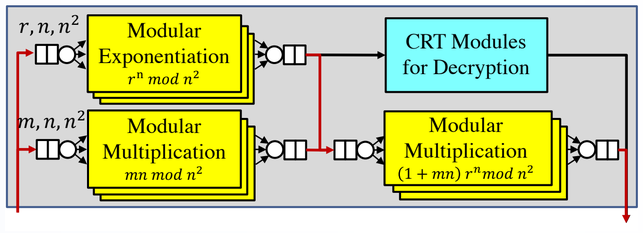
1. The engine can work in two modes. They can switch between modular multiplication and exponentiation



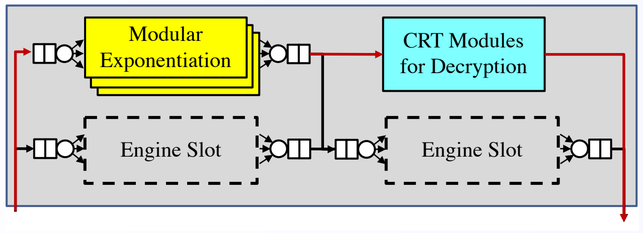
2. The core idea of dataflow scheduling is using an on-chip controller to determine which data paths should be active based on which operation is offloaded on-demand.



All available paths for dataflow scheduling

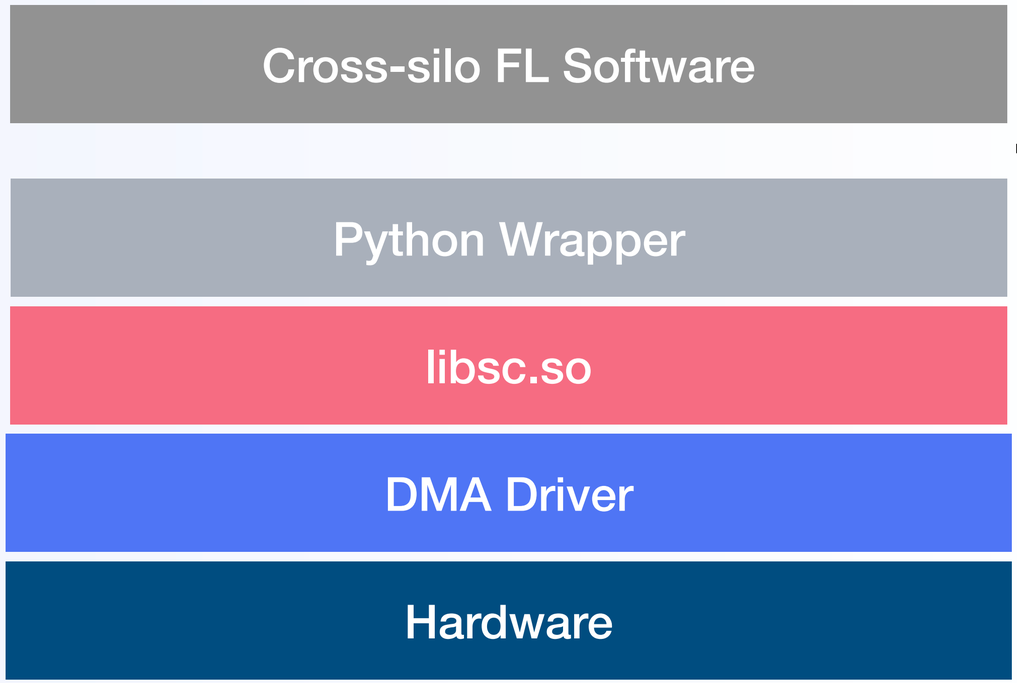


Dataflow for encryption



Dataflow for decryption

Software Integration



Listing 1: FLASH's NumPy-like APIs

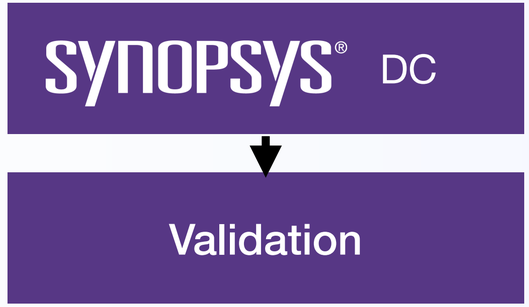
```
import flash_np as np
# Generating 2 Paillier-encrypted arrays accelerated by FLASH
x1 = np.array([1, 2, 3], encryption="paillier")
x2 = np.array([4, 5, 6], encryption="paillier")

x3 = x1 + x2 # Homomorphic addition

x4 = np.array([1, 2, 3], encryption=None)
x5 = x4 * x1 # Ciphertext & cleartext multiplication

x3.decrypt() # Decrypting the ciphertext
x5.decrypt()

# Transferring the data from accelerator to host
x3.get()
x5.get()
```

Implementation

~30,000 lines of Verilog
Prototyping with Xilinx VU13P

FATE

The most adopted cross-silo FL framework
~10,000 lines of C++ and Python code

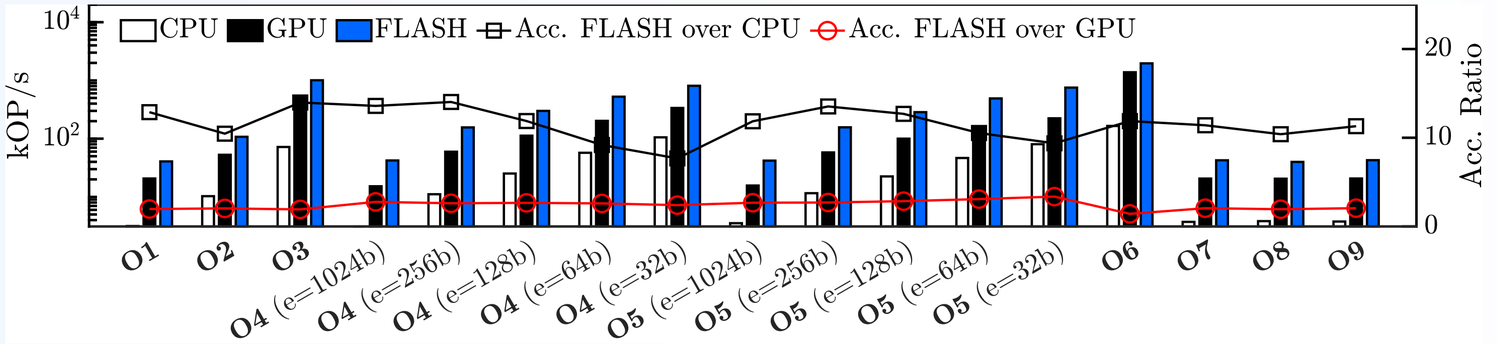


Synopsys DC for logical synthesis
Synopsys PT and VCS for validation

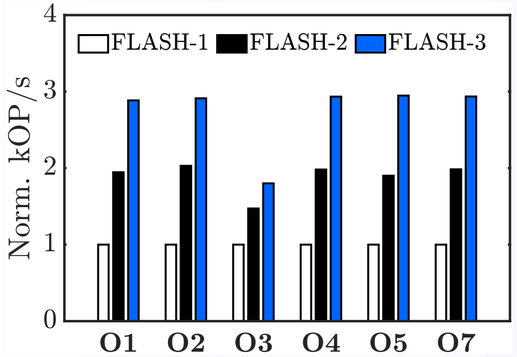
https://github.com/FederatedAI/FATE

Evaluation — Cryptographic Operations

Intel Xeon Silver 4114 CPU (10 core)
NVIDIA P4 (share the similar INT8 TOPS with FLASH)



(a) Cryptographic operation performance of all compared schemes. The left Y-axis is associated with the bar chart and is in log scale. The right Y-axis is associated with the line chart.



(b) Multi-accelerator performance with selected operations.

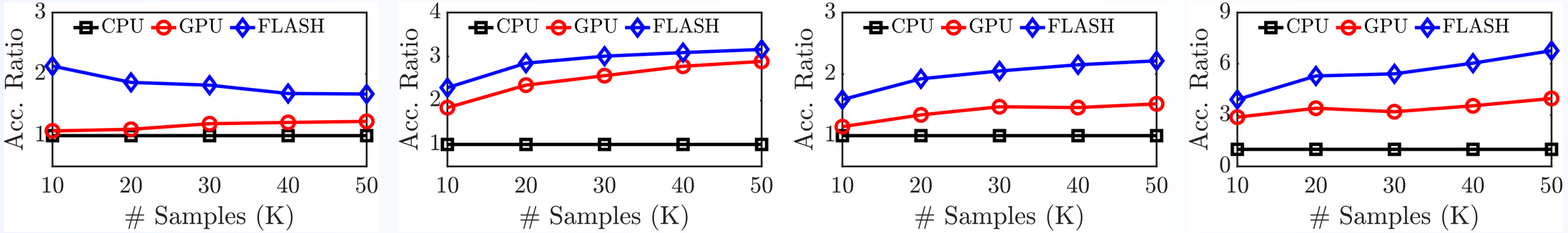
Figure 8: Performance of cryptographic operations.

For cryptographic operations:

1. FLASH outperforms CPU by achieving $7.7\times \sim 14.0\times$ speedup
2. FLASH outperforms GPU by achieving $1.4\times \sim 3.4\times$ speedup
3. The overall performance of FLASH is almost linear to the number of accelerators

Evaluation — Cross-silo FL Application

CPU: Intel Xeon Silver 4114 CPU (10 core)
GPU: NVIDIA P4 (share the similar INT8 TOPS with FLASH)
FPGA: Xilinx VU13P



(a) RSA-PSI (b) VLR (c) SBT (d) HLR

Figure 9: Performance of RSA-PSI, VLR, SBT, and HLR with changing data volumes.

For RSA-PSI, VLR, SBT, and HLR:

1. FLASH outperforms CPU by achieving $1.6\times \sim 6.8\times$ speedup
2. FLASH outperforms GPU by achieving $1.1\times \sim 2.0\times$ speedup

Evaluation — Cross-silo FL Application

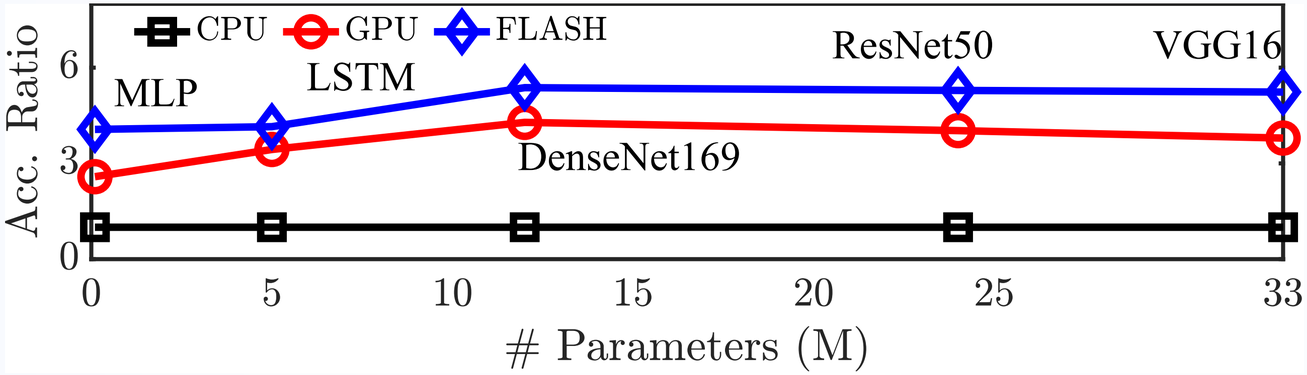


Figure 10: Performance of 5 deep neural networks.

For deep learning models:

1. FLASH outperforms CPU by achieving $4.1\times \sim 5.4\times$ speedup
2. FLASH outperforms GPU by achieving $1.2\times \sim 1.6\times$ speedup

Evaluation — ASIC Evaluation

	28nm Technology Library			12nm Technology Library		
	Area/Unit (mm^2)	**# Unit**	**Total Area (mm^2)**	**Area/Unit (mm^2)**	**# Unit**	**Total Area (mm^2)**
PCIe Gen3×16	8.46	1	8.460 (6.56%)	5.25	1	5.250 (4.04%)
DDR4	7.25	2	14.500 (11.24%)	4.43	2	8.860 (6.81%)
Engine Logic	0.093	800	74.480 (57.72%)	0.046	1900	87.499 (67.26%)
Engine Memory	0.033	800	26.200 (20.30%)	0.014	1900	25.927 (19.93%)
Dataflow Scheduling & Others	5.399	1	5.399 (4.18%)	2.561	1	2.561 (1.97%)
Total	-	-	129.04 (99.26%)	-	-	130.10 (100.08%)

Table 4: ASIC resource evaluation for both 28nm and 12nm technology libraries.

If implemented as an ASIC

1. FLASH achieves $7.11\times$ performance gain compared to FPGA prototype with 28nm technology library
2. FLASH achieves $23.64\times$ performance gain compared to FPGA prototype with 12nm technology library

	Frequency (MHz)	**# Engines**	**Performance[a]**
VU13P FPGA	300	300	1
28nm ASIC	800	800	$7.11\times\uparrow$
12nm ASIC	1120	1900	$23.64\times\uparrow$

[a] We use the performance achieved by VU13P FPGA as a baseline. All performance data achieved by other implementations are normalized to VU13P FPGA.

Table 5: ASIC performance estimation.

Conclusion

1. We **identified 9 cryptographic operations** that are widely used in cross-silo FL that cause dramatic performance degradation.

2. We proposed FLASH, a high-performance hardware acceleration architecture for cross-silo federated learning. FLASH leverages the observation that these 9 cryptographic operations are **built upon two basic operators**: modular multiplication & exponentiation to achieve high performance and resource utilization.

3. We provided a **fully-functional implementation of FLASH with FPGA** and integrated it with FATE. We also used **Synopsys tools to evaluate FLASH as an ASIC**. Testbed and evaluation results show that FLASH is a promising solution.

Thank You !

THE HONG KONG UNIVERSITY OF SCIENCE AND TECHNOLOGY

CLUSTAR 星云