阿里安全
ALIBABA SECURITY

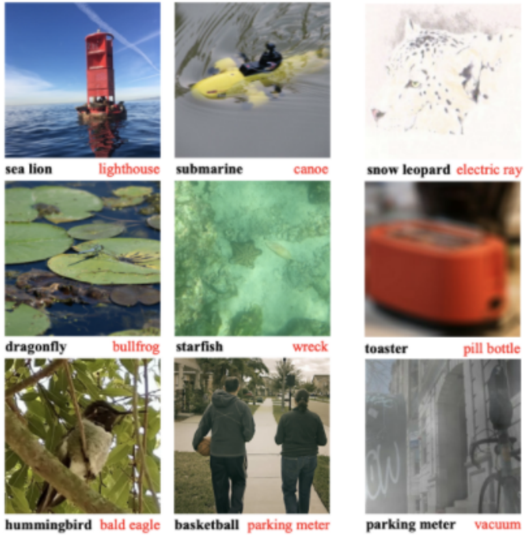
Cheetah

Lean and Fast Secure Two-Party Deep Neural Network Inference

Zhicong (Zico) Huang, **Wen-jie Lu**, Cheng Hong, and Jiansheng Ding
Alibaba Group

Secure Neural Network Inference

- Simple images and models
  - ➢ MNIST (28x28 black/white): 3~5 layers
- Complex images and models
  - ➢ CIFAR-10 (32x32 rgb): 10+ layers
  - ➢ IMAGENET (224x224 rgb): ResNet50
- ResNet50: one of the most popular DNN models
- Secure two-party ResNet50 inference
  - ➢ Prior best work: CryptFLOW2
  - ➢ 10 mins for one image inference (LAN, 3Gbps)
  - ➢ 20 mins for one image inference (WAN, 300Mbps)



Design Challenges in 2PC Frameworks

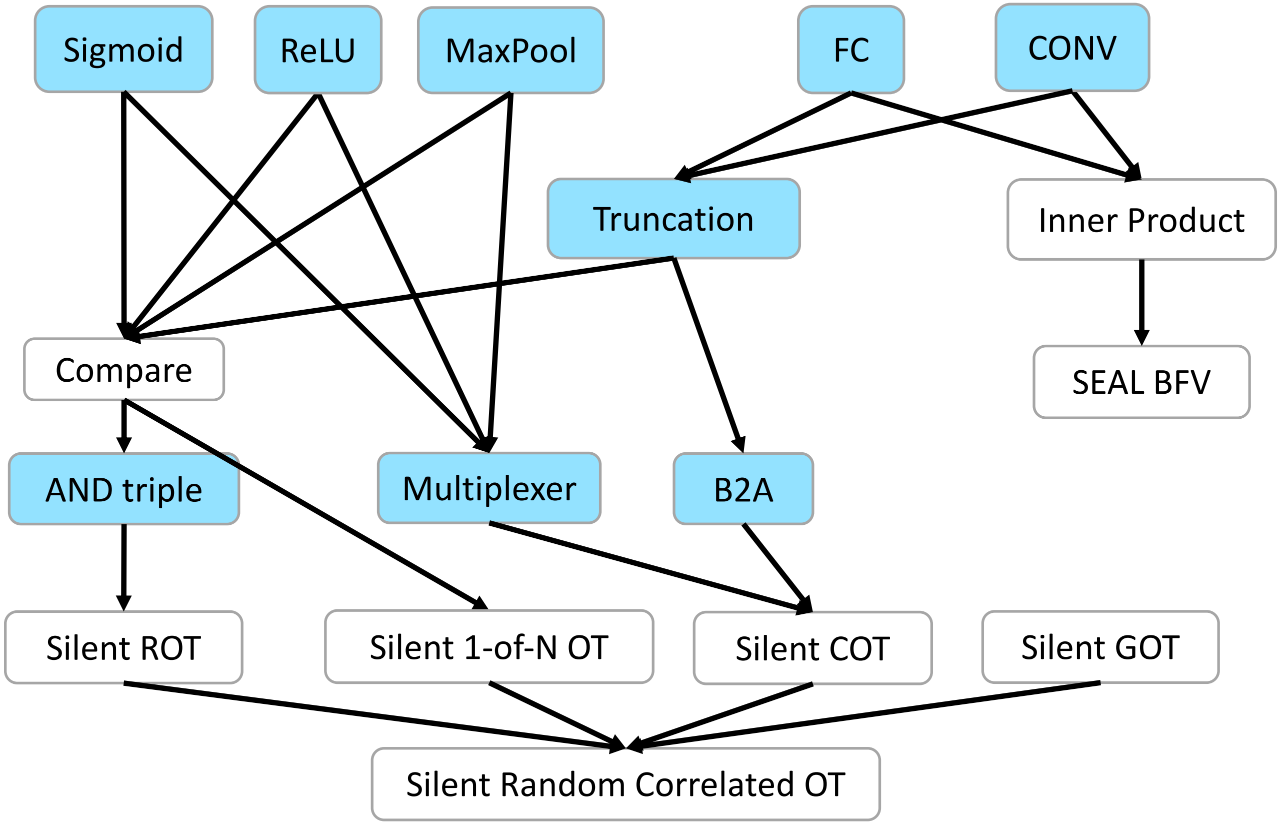
阿里安全 ALIBABA SECURITY

- Optimize trade-offs among different primitives
- Adapt to concrete application cases

Framework Type	Computation Cost	Communication Amount	Communication Round	Existing Works
GC（Y）	☆	☆☆☆	☆	EMP
SS（A、B）	☆	☆☆	☆☆☆	SPDZ、CryptFlow2
FHE	☆☆☆	☆	☆	Pegasus
A + B + Y	☆	☆☆☆	☆☆	ABY、SecureML
SS（A、B）	☆	☆	☆☆	**Cheetah**

☆ Low ☆☆ Medium ☆☆☆ High

Cheetah Protocol Architecture



Additive Secret Sharing

- Integer $a \in [0, P)$ is split into shares a_1, a_2
  - Computation party P_i has share a_i
  - Satisfy $a_1 + a_2 \bmod P = a$
- Local Add/Sub computation
- Two types of sharings depending on modulus P
  - P = 2 → **B**oolean Share
  - P > 2 → **A**rithmetic Share。 P is usually a prime or a power of 2 (e.g., 2^64)

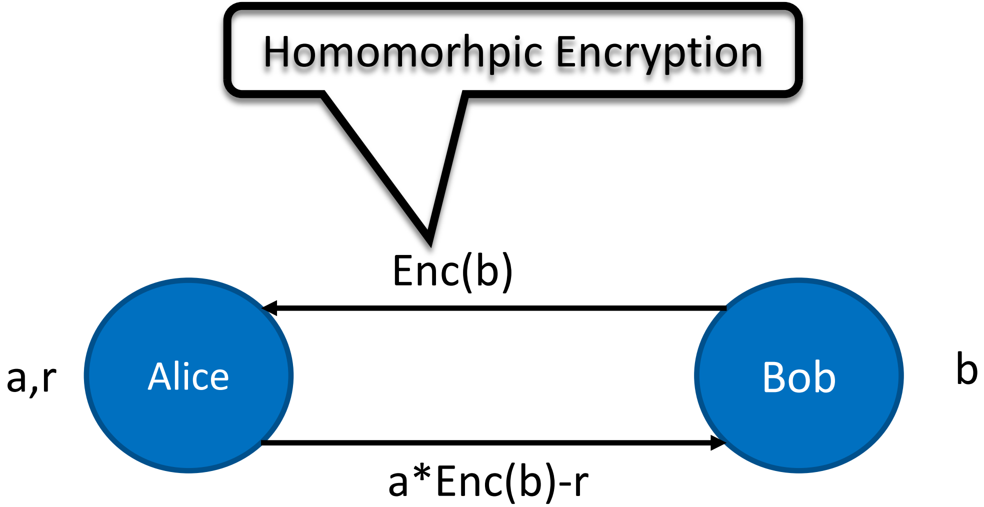
阿里安全
ALIBABA SECURITY

02

Linear Primitives

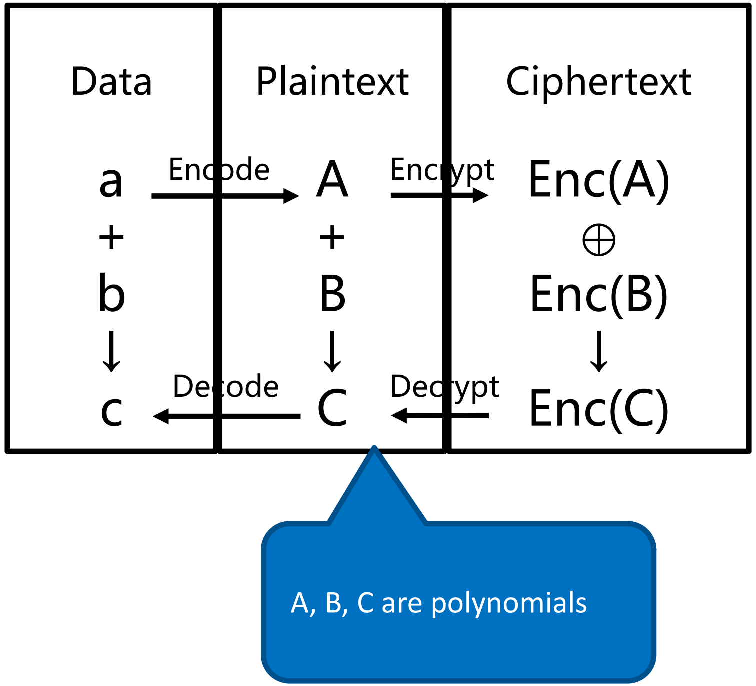
Linear layers: CONV, FC

- CONV/FC： Matrix Mult → Inner Product
- Input:
  - ➢ Alice (model owner)： vector $\vec{a}$
  - ➢ Bob (data owner): vector $\vec{b}$
- Output:
  - ➢ Alice: r
  - ➢ Bob: $\vec{a} \cdot \vec{b} - r \bmod k$



Computation based on Polynomials

- Plaintext space for BFV: Polynomial Ring
  - Polynomial $Z_t(x)/X^N+1$
  - Degree of N-1. Each integer coeff in [0, t-1]
  - Ciphertext add/mult <-> Polynomial add/mult
  - E.g.: $N = 2,\ t = 7$ → $\bmod x^2 + 1$
    Enc(x+2) * Enc(x+3)
    = Enc(x^2+5x+6)
    = Enc(5x+5)



Packing: CRT Batching

2PC Computation Systems
Gazelle、Delphi、CryptFlow2

阿里安全 ALIBABA SECURITY

- How to encode data into polynomials?
  - ➢ x^n+1 can be broken into the product of n polynomials： $x^n+1=(x+a_1)(x+a_2)\ldots(x+a_n)$
    - E.g.： t=17,n=2 → $x^2+1=(x-4)(x-13)$ // $x^2-17x+52$ mod 17
  - ➢ f(x) mod (x^n+1) can represent n integers： $x_i = f(x) \bmod (x+a_i)$
    - E.g.： x mod (x^2+1) → x mod (x-4) 和 x mod (x-13) → **x mod (x^2+1) "pack" 4 and 13**

- Given n integers， find corresponding f(x) to encode them by CRT
  - E.g.： **2x-7 "pack" 1 and 2** : // 2x-7 mod (x-4) = 1 , 2x-7 mod (x-13) = 2 mod 17

- Packing keeps homomorphism modulo t
  - Add: **x+(2x-7) packs 5 and 15** : // 3x-7 mod (x-4) = 5 , 3x-7 mod (x-13) = 15 mod 17
  - **Mult**: **x*(2x-7) packs 4 and 9** :

    // $2x^2-7x$ mod (x^2+1) = -7x-2 ； -7x-2 mod (x-4) = 4 , -7x-2 mod (x-13) = 9 mod 17

- **SIMD**: One polynomial calculation completes n integer calculations

Precondition of SIMD Packing in BFV

- Almost all efficient BFV applications use SIMD Packing
  - One poly mult → 1000+ plain integer mults
- SIMD requires plain modulus t to be a prime →
  Secret sharing has to work in prime field in a mixed protocol
  - Performance degrades significantly（60% more overhead [CrypTFlow2]）

Inner Product 1st Try : SIMD Packing + Ciphertext Rotation

- A has a vector $a=(a_0,a_1, \ldots a_n)$, B has a vector $b=(b_0,b_1, \ldots b_n)$
- A SIMD packs a as a polynomial $A(x)/X^N+1$; B SIMD packs b as a polynomial $B(x)/X^N+1$
- B uses its public key to encrypt B(x) , and send to A
- A performs homomorhic mult on Enc(B(x)) and A(x) → Obtains Enc(C(x)) $/X^N+1$
    - C(x) packs ($\mathbf{a_0b_0}$, $a_1b_1, \ldots a_nb_n$)
    - **! One step away from inner product: BIG SUM**
- A **rotates** the ciphertext Enc(C(x)) , obtaining

$$\left.\begin{array}{l} (\mathbf{a_1b_1}, \ldots a_{n-1}b_{n-1}, a_nb_n, a_0b_0) \\ (\mathbf{a_2b_2}, \ldots a_nb_n, \quad a_0b_0, a_1b_1) \\ \qquad \ldots \\ (\mathbf{a_nb_n}, a_0b_0, a_1b_1, \ldots a_{n-1}b_{n-1}) \end{array}\right\} \text{n ciphertexts}$$

    - A performs homomorphic add to get $(a\cdot b, \ldots a\cdot b)$, sends to B , and B decrypts to get $a\cdot b$
- **! Needs log(n) rotates and n adds. Performance not better than Paillier.**

Inner Product 2nd Try : Polynomial Coefficient Encoding

- A has a vector $a=(a_0,a_1, \dots a_n)$, B has a vector $b=(b_0,b_1, \dots b_n)$
  - A encodes a into a polynomial

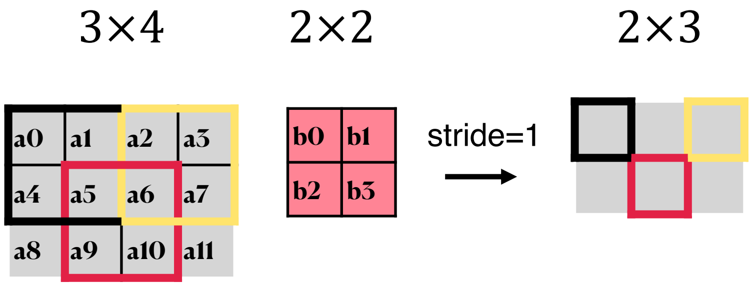
$$P_a = a_0 + a_1X + a_2X^2 + \cdots + a_nX^n$$

  - B encodes b into a polynomial

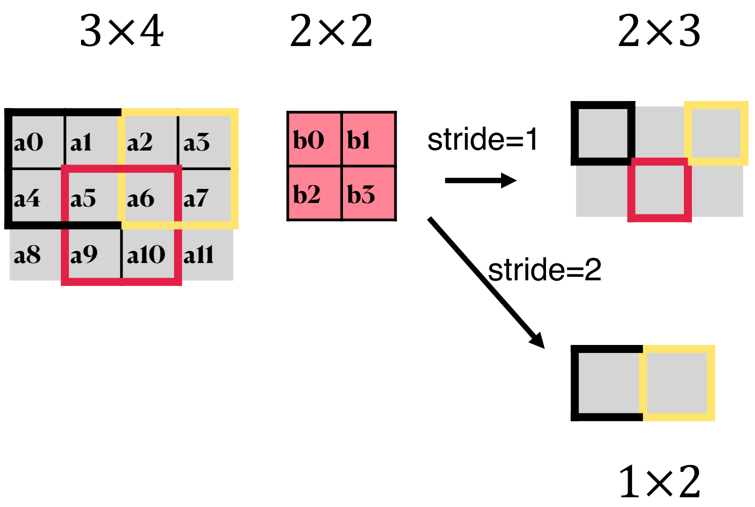
$$P_b = b_0 - b_1X^{N-1} - b_2X^{N-2} - \cdots - b_nX^{N-n}$$

  - where $X^N = -1 \bmod (X^N + 1)$
  - Hence the constant term of $P_a * P_b$ is the inner product $a \cdot b$
  - √ Only one homomorphic mult.
    - N=4096 costs only 1 millisecond

2D Convolution



2D Convolution



2D Convolution

Encoding for Tensor

$$a(X) = a_0 + a_1X + a_2X^2 + a_3X^3 + a_4X^4 + a_5X^5 + a_6X^6 + a_7X^7 + a_8X^8 + a_9X^9 + a_{10}X^{10} + a_{11}X^{11}$$

a0	a1	a2	a3
a4	a5	a6	a7
a8	a9	a10	a11

2D Convolution

Encoding for Kernel

$$a(X) = a_0 + a_1X + a_2X^2 + a_3X^3 + a_4X^4 + a_5X^5 + a_6X^6 + a_7X^7 + a_8X^8 + a_9X^9 + a_{10}X^{10} + a_{11}X^{11}$$

$$b(X) = b_3 + b_2X + 0X^2 + 0X^3 + b_1X^4 + b_0X^5$$

b0	b1
b2	b3

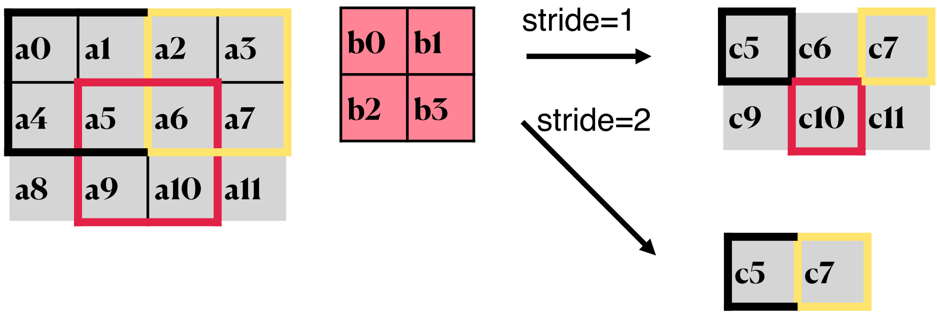
$$a(X) \cdot b(X) = \sum_{i=0}^{15} c_iX^i$$

2D Convolution

Multiplication between a long polynomial and a short polynomial → Convolution

$$a(X) = a_0 + a_1X + a_2X^2 + a_3X^3 + a_4X^4 + a_5X^5 + a_6X^6 + a_7X^7 + a_8X^8 + a_9X^9 + a_{10}X^{10} + a_{11}X^{11}$$

$$b(X) = b_3 + b_2X + 0X^2 + 0X^3 + b_1X^4 + b_0X^5$$



$$a(X) \cdot b(X) = \sum_{i=0}^{15} c_i X^i$$

$$c_5 = a_0b_0 + a_1b_1 + a_4b_2 + a_5b_3$$

$$c_7 = a_2b_0 + a_3b_1 + a_6b_2 + a_7b_3$$

$$c_{10} = a_5b_0 + a_6b_1 + a_9b_2 + a_{10}b_3$$

$$c_6 = a_1b_0 + a_2b_1 + a_5b_2 + a_6b_3$$

$$c_9 = a_4b_0 + a_5b_1 + a_8b_2 + a_9b_3$$

$$c_{11} = a_6b_0 + a_7b_1 + a_9b_2 + a_{11}b_3$$

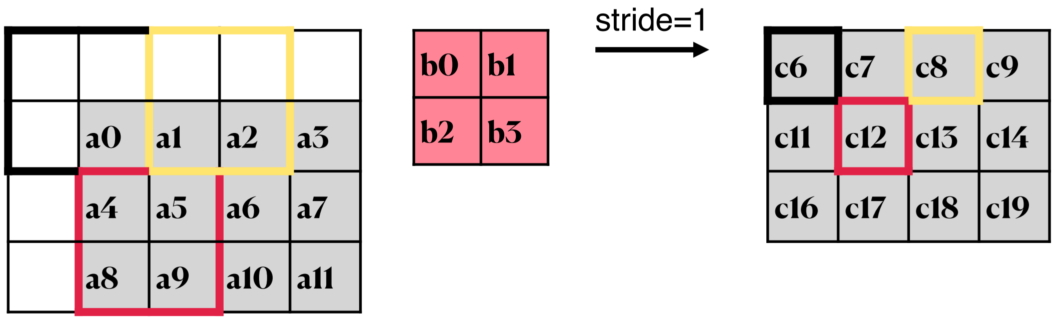
Convolution

High flexibility: stride >= 1 & Same/Valid Padding & 3D Convolution

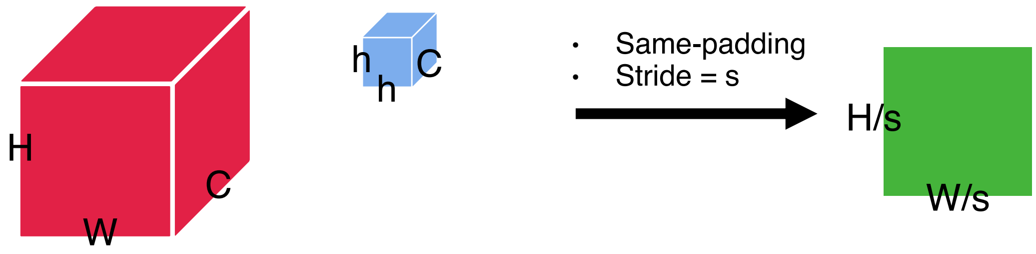
$$a(X) = a_0X^6 + a_1X^7 + \cdots + a_{11}X^{19}$$

$$b(X) = b_3 + b_2X + 0X^2 + 0X^3 + 0X^4 + b_1X^5 + b_0X^6$$

$$a(X) \cdot b(X) = \sum_{i=0}^{25} c_iX^i$$

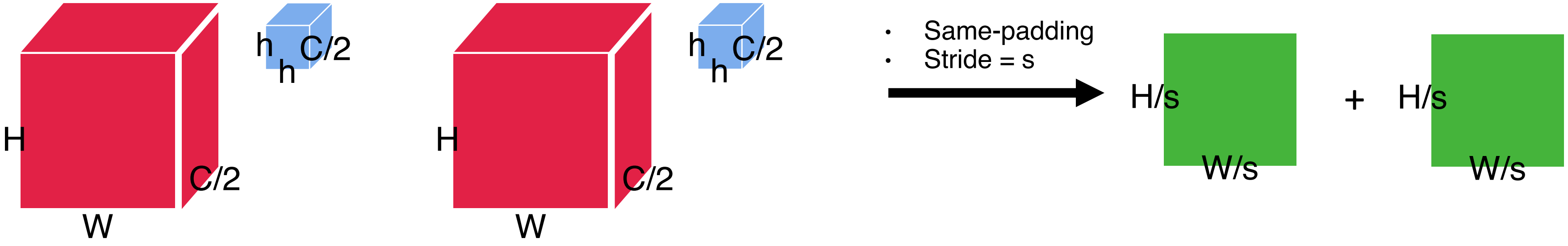


Big Tensor



- The whole Tensor needs to be Encoded into a polynomial of degree N
  - $HWC \leq N$ (valid padding)
  - $(H-h+1)(W-h+1)C \leq N$ (same padding)
  - (rare case) when stride s >= h, we can skip some computation

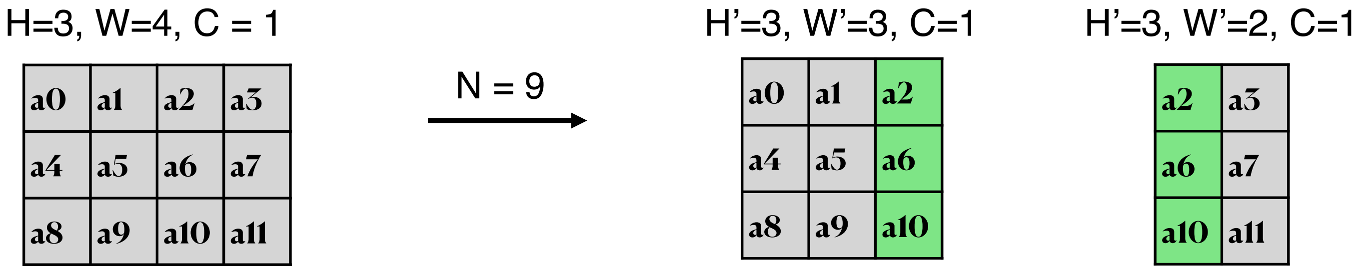
Split along Channels



- The whole Tensor needs to be Encoded into a polynomial of degree N
  - $HWC \leq N$ (valid padding)
  - $(H-h+1)(W-h+1)C \leq N$ (same padding)
  - (rare case) when stride s >= h, we can skip some computation
- Big Tensor (e.g., HWC > N) can be split into small tensors
  - Along Channels: Just a simple addition in the end

Big Tensor

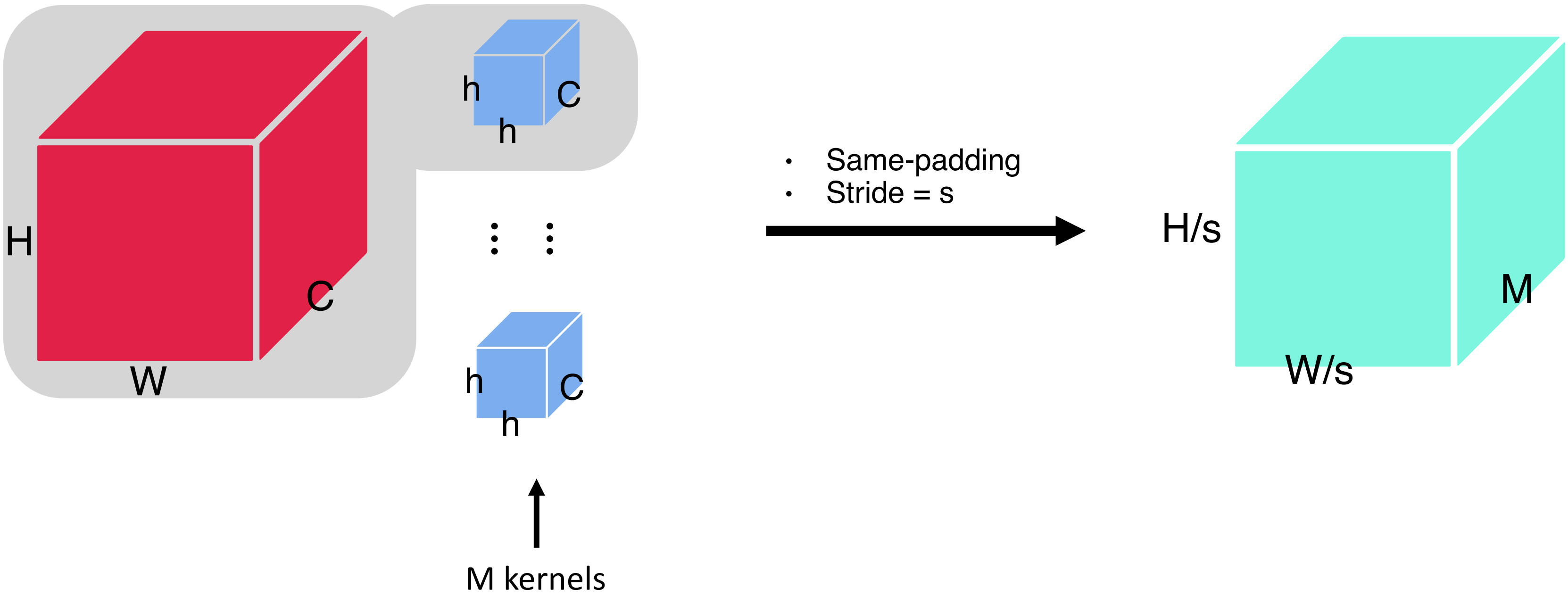
Split along Height/Width



- The whole Tensor needs to be Encoded into a polynomial of degree N
  - $HWC \leq N$ (valid padding)
  - $(H - h + 1)(W - h + 1)C \leq N$ (same padding)
  - (rare case) when stride s >= h, we can skip some computation
- Big Tensor (e.g., HWC > N) can be split into small tensors
  - Along Channels: Just a simple addition in the end
  - Along Height/Width: Might contain overlaps

Multiple Kernels

Compute independently for each kernel

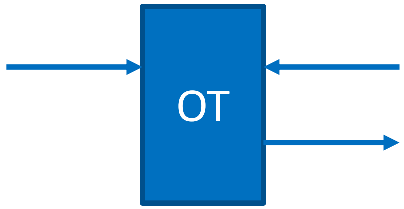


03

Non-Linear Primitives

Oblivious Transfer (Primitive)

- Sender has ℓ-bit integers a_0, a_1
- Receiver chooses one of them with a choice bit $b \in \{0, 1\}$
- OT result：
  - Receiver gets a_b, but does not know a_{1-b}
  - Sender does not know b
- Other variants：
  - 1-of-m OT: Sender has m >=2 messages
  - Random OT: Sender obtains random messages a_0, a_1
  - Correlated OT: Sender's inputs a_0, a_1 satisfy some correlation (e.g., $a_1 = \Delta \oplus a_0$)



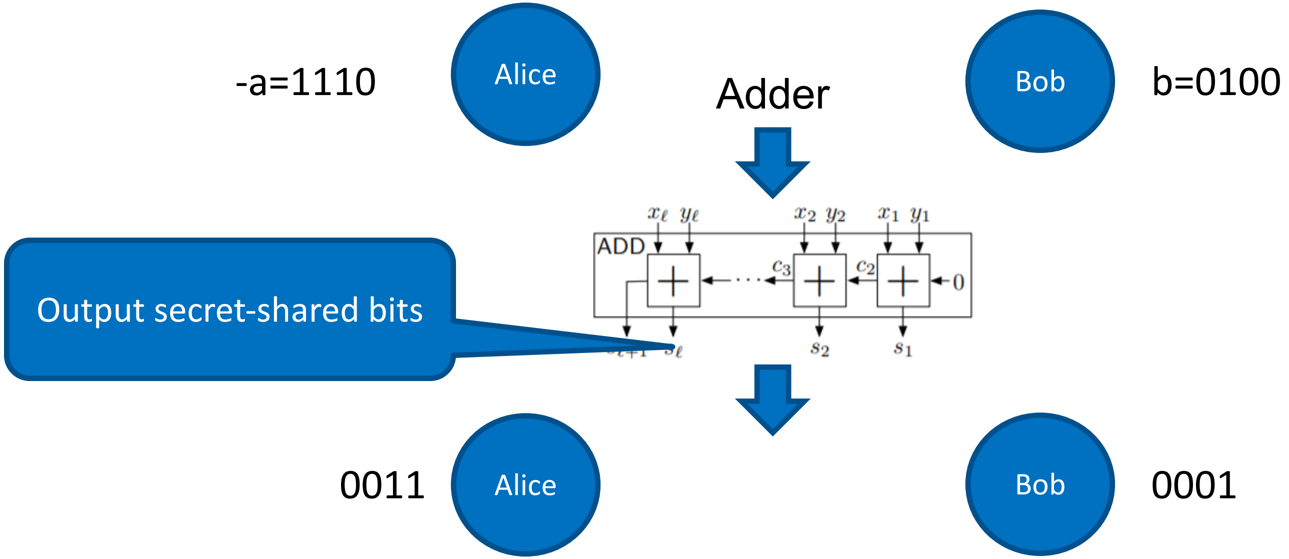
Non-linear layers（ReLU，MaxPool）

- ReLU：ReLU(x) := $\max(x, 0)$
- Input:
  - Alice、Bob：Secret-shared x
- Output:
  - Alice、Bob: Secret-shared $\text{Compare}(x, 0) * x$

- $\text{Compare}(x, 0)$
  - $= 0, \text{if } x < 0$
  - $= 1, \text{if } x >= 0$

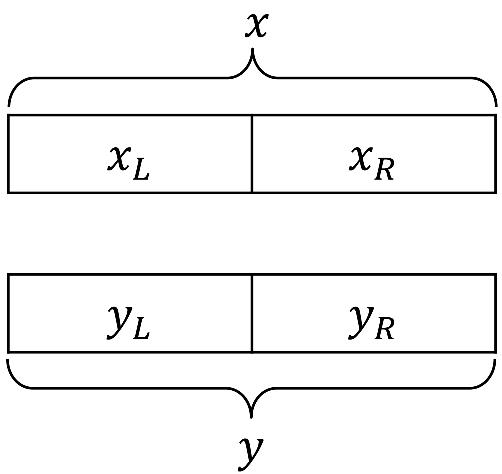
Compare

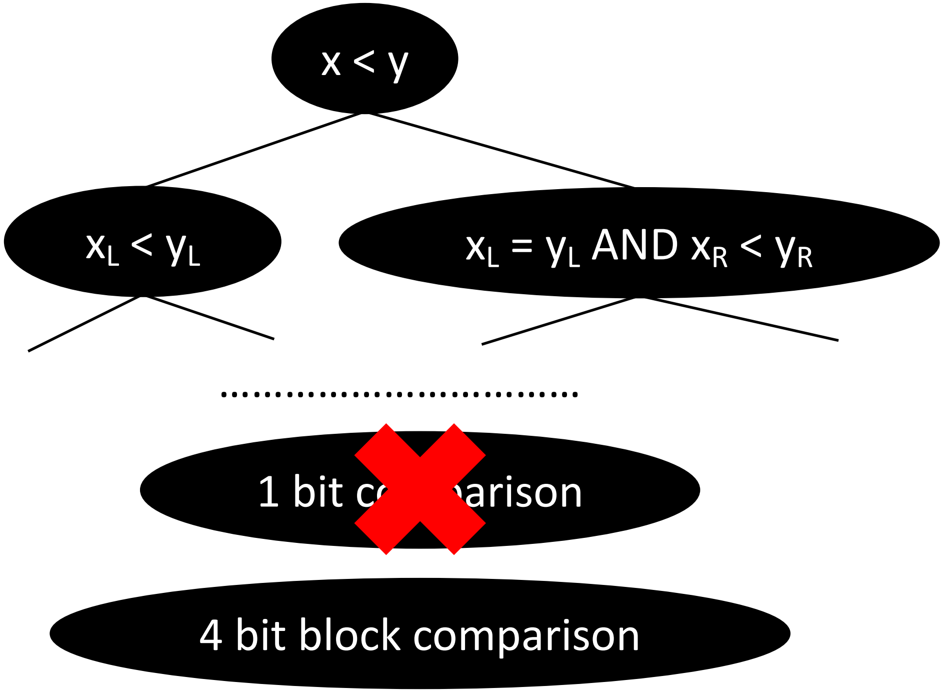
- $\mathrm{Compare}(b, a)$ solution 1： execute boolean adder to obtain $\mathrm{MSB}(b-a)$



Compare

- Solution 2: comparison tree [CrypTFlow2]





Compare

- Solution 2: comparison tree [CryptFlow2]
  - Minimize comm. rounds and AND gates

4 bit block comparison

Assume $x = a$

x < 0
x < 1
…
x < a
} 0

x < a+1
…
x < 15
} 1

1-of-16 OT

Alice inputs: $r \oplus \{x < i\}$, 0 <= i <= 15
Bob inputs: y
➔
Alice obtains: r
Bob obtains: $r \oplus \{x < y\}$

Primitives in Compare

- 1-of-2^m OT
- AND Gate
  - Beaver triple
  - 1-of-2 Random OT

- CryptFlow2 uses classic IKNP-OT
- Recent years, we have seen a series of Silent OT schemes based on VOLE
  - [CCS19], [Crypto21], **[Ferret]**
  - Generate massive amount of Random Correlated OT with little communication：

$$c_i = b_i + a_i \cdot x$$

  where $(b_i, b_i + x)$ are Sender's random correlated messages， $a_i \in \{0,1\}$ is Receiver's choice bit

  - Random Correlated OT → Any other OT variant

Primitives in Compare

阿里安全 ALIBABA SECURITY

- 1-of-2^m OT
  - IKNP-OT scheme：[KKOT 2013]
  - Silent OT scheme：m instances of 1-of-2 Random OT [NaorPinkas1999]

Sender	Receiver
$\mathbf{2^m}$ **messges：** $\boldsymbol{x_0, x_1, \ldots, x_{2^m-1}}$	**choice** $\boldsymbol{c \in [0, 2^m)}$

Invoke m ROTs

Obtain: $(p_0^0, p_1^0), (p_0^1, p_1^1), \ldots, (p_0^{m-1}, p_1^{m-1})$

For all k $\in [0, 2^m)$， compute and send：

$$y_k = x_k \oplus p_{k[0]}^0 \oplus p_{k[1]}^1 \oplus \cdots \oplus p_{k[m-1]}^{m-1} \longrightarrow$$

Obtain： $p_{c[0]}^0, p_{c[1]}^1, \ldots, p_{c[m-1]}^1$

Compute：

$$x_c = y_c \oplus p_{c[0]}^0 \oplus p_{c[1]}^1 \oplus \cdots \oplus p_{c[m-1]}^{m-1}$$

Primitives in Compare

Primitives	Communication (bits)	
	IKNP (CF2)	Silent (Cheetah)
$\binom{2}{1}-\mathrm{ROT}_\ell$	λ	0 or 1
$\binom{2}{1}-\mathrm{COT}_\ell$	$\ell+\lambda$	$\ell+1$
$\binom{2}{1}-\mathrm{OT}_\ell$	$2\ell+\lambda$	$2\ell+1$
$\binom{n}{1}-\mathrm{OT}_\ell\ (\mathrm{n}\geq 3)$	$n\ell+2\lambda$	$n\ell+\log_2 n$

E.g.: $\ell=64,\ \lambda=128$

Truncation

- Fixed point (FP) numbers for MPC
  - ➢ Value is 0.5， scale is 2^{15} → FP representation: $0.5 \times 2^{15} = 16384$
- Problem： multiplication increases the scale
  - ➢ 0.5×0.5 → $16384 \times 16384 = 268435456 = 0.25 \times 2^{30}$
  - ➢ Several mults would lead to an overflow
- Need a method to truncate secret-shared values to maintain the scale
  - ➢ Plain truncation: x >> 15
  - ➢ but local truncation leads to BIG error on secret sharings [SecureML]：

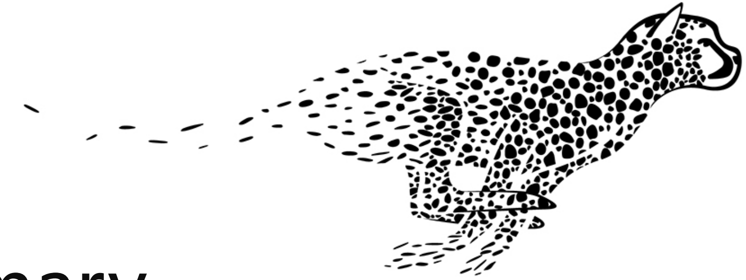
    x = x1 + x2 mod 2^k

    (x >> 15) != (x1 >> 15) + (x2 >> 15)

➢ **Cheetah： Efficient silent OT-based truncation protocol**

(1/2 probability with tiny one-bit LSB error)

04



Performance and Summary

Performance

Benchmark	System	End2End Time LAN	End2End Time WAN	Commu.
SqNet	SCI_{HE} [50]	41.1s	147.2s	5.9GB
	SecureQ8 [16]	4.4s	134.1s	0.8GB
	Cheetah	16.0s	39.1s	0.5GB
RN50	SCI_{HE} [50]	295.7s	759.1s	29.2GB
	SecureQ8 [16]	32.6s	379.2s	3.8GB
	Cheetah	80.3s	134.7s	2.3GB
DNet	SCI_{HE} [50]	296.2s	929.0s	35.4GB
	SecureQ8 [16]	22.5s	342.6s	4.6GB
	Cheetah	79.3s	177.7s	2.4GB

SqNet = SqueezeNet; RN50 = ResNet50; DNet = DenseNet121

Computation： 3x

Communication： 10x

SCI_{HE}: CryptFlow2

SecureQ8: State-of-the-Art 3PC framework

Takeaways

☆ Low　☆☆ Medium　☆☆☆ High

Framework Type	Computation cost	Communication Amount	Communication Round	Work
SS（A、B）	☆	☆	☆☆	Cheetah

- With RLWE and Silent OT， 2PC systems can be implemented in very efficient ways
- The most optimized design need to consider computation tasks, primitives and parameters
  - Mod 2^k OR Mod p
  - Data encoding： SIMD OR Coefficient Encoding
  - Comparison： Adder circuit、Pure AND triple OR 1-of-N OT
  - …
- Available:
  - https://github.com/Alibaba-Gemini-Lab/OpenCheetah

THANKS

阿里安全
ALIBABA SECURITY