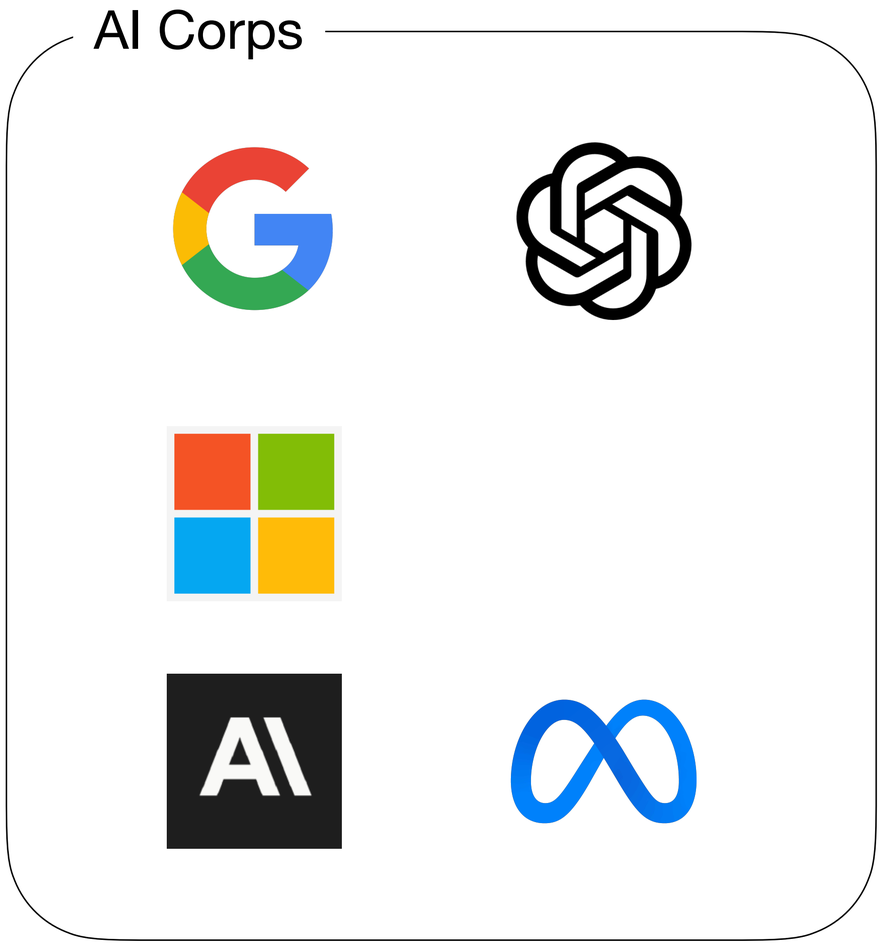
SIGMA: Secure GPT Inference with Function Secret Sharing

Kanav Gupta, Neha Jawalkar, Ananta Mukherjee, Nishanth Chandran, Divya Gupta, Ashish Panwar, Rahul Sharma

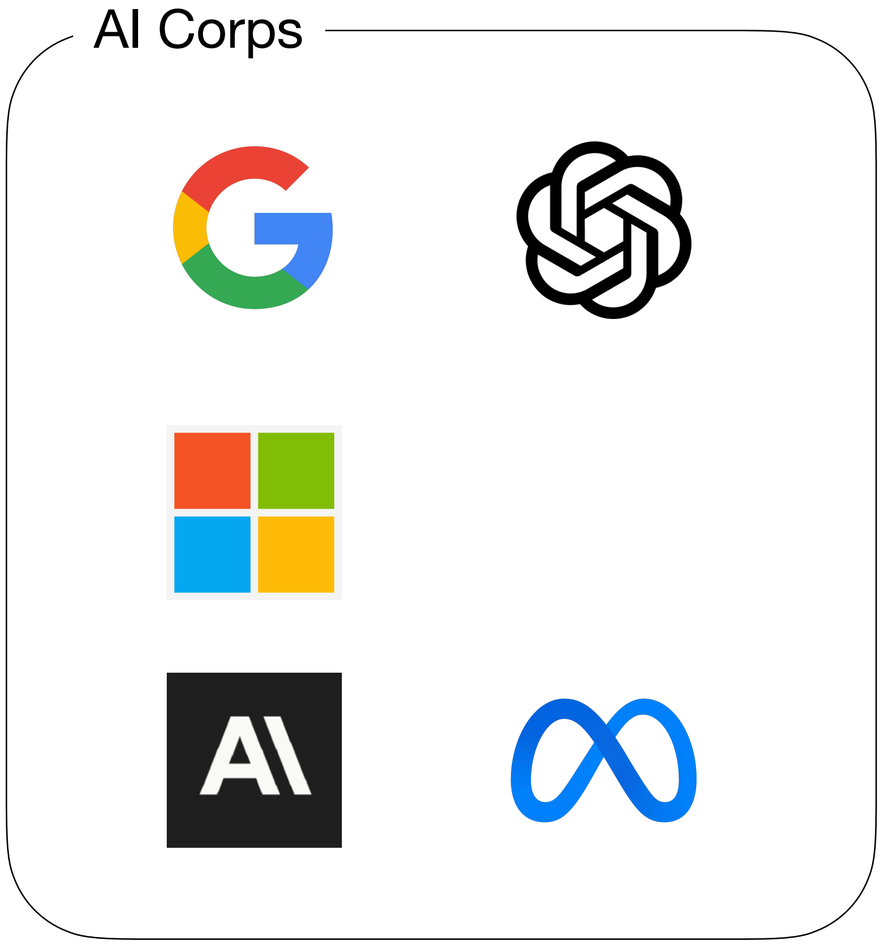
Real World Crypto 2024

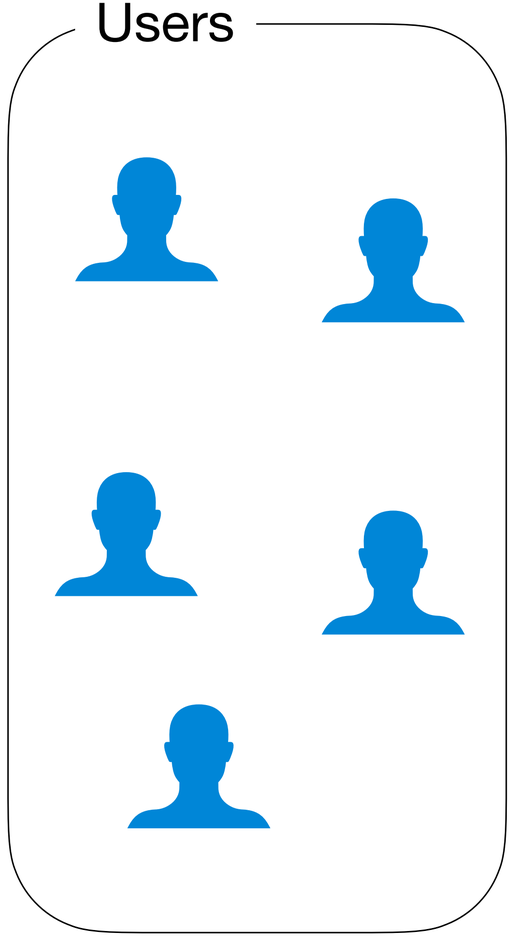
AI effects everyday life

AI effects everyday life



AI effects everyday life

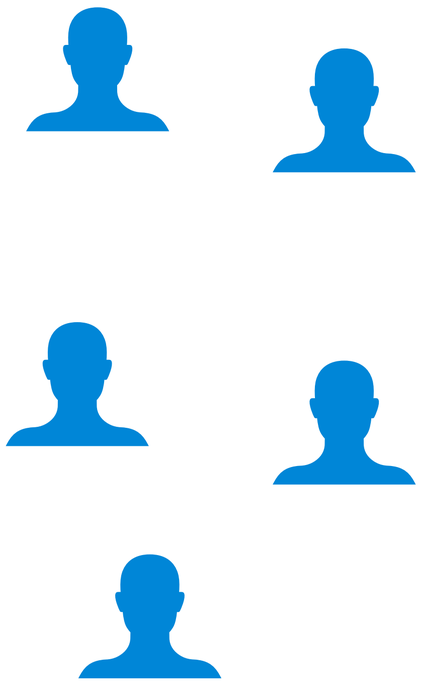




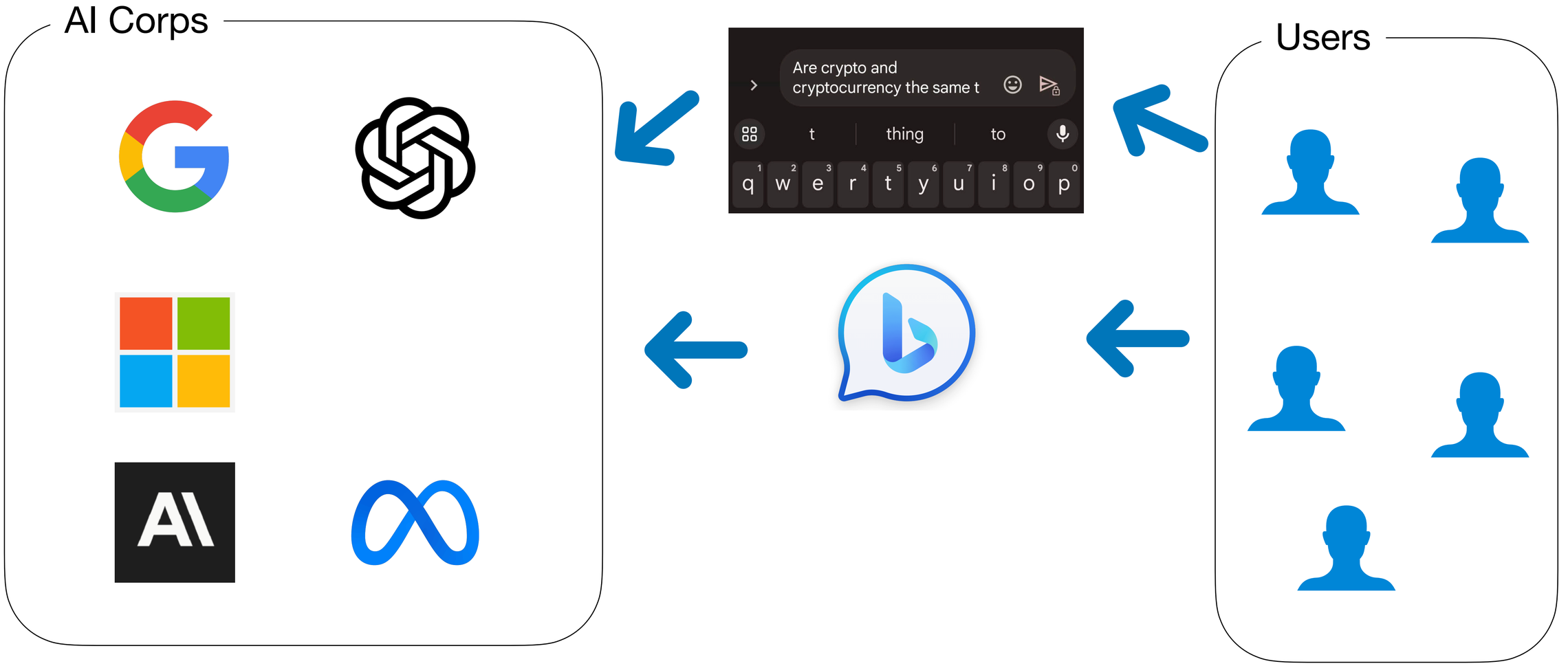
AI effects everyday life

AI Corps

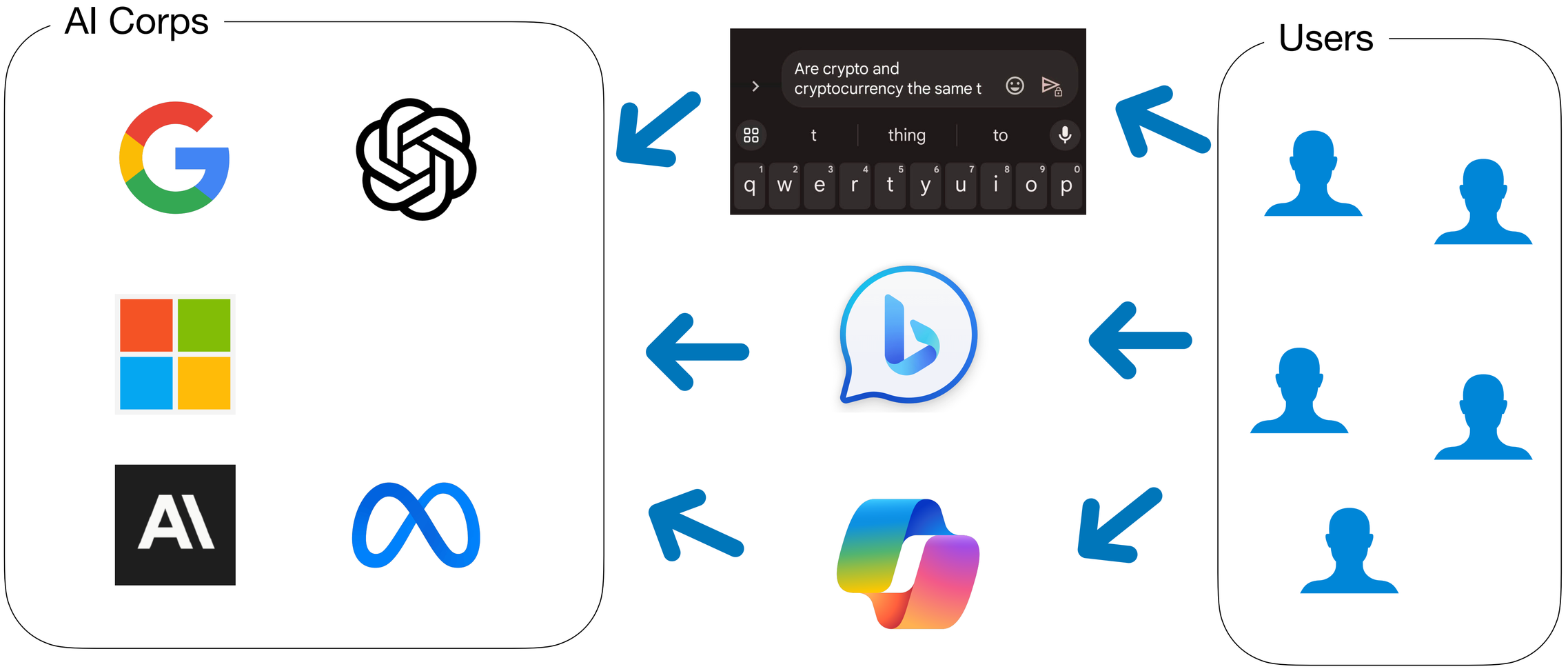
Users



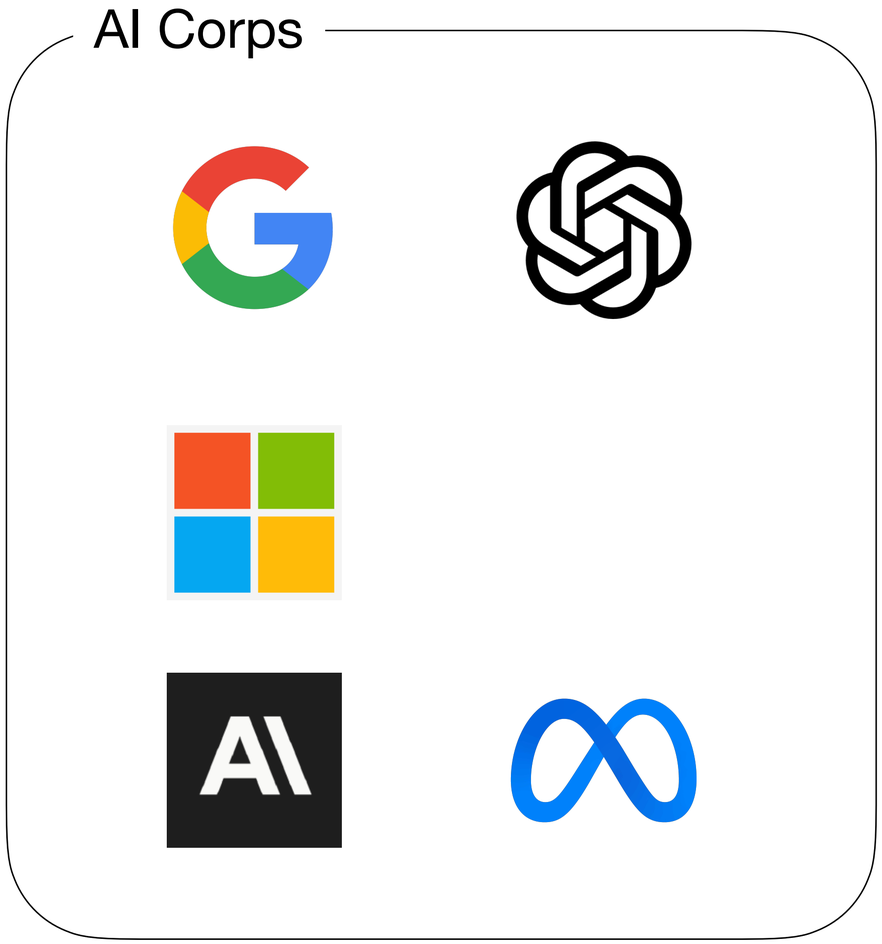
AI effects everyday life

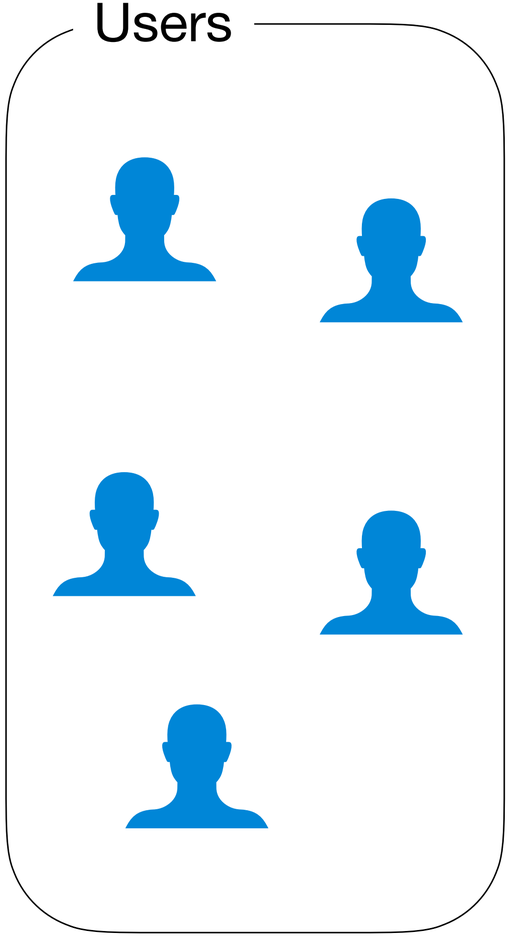


AI effects everyday life

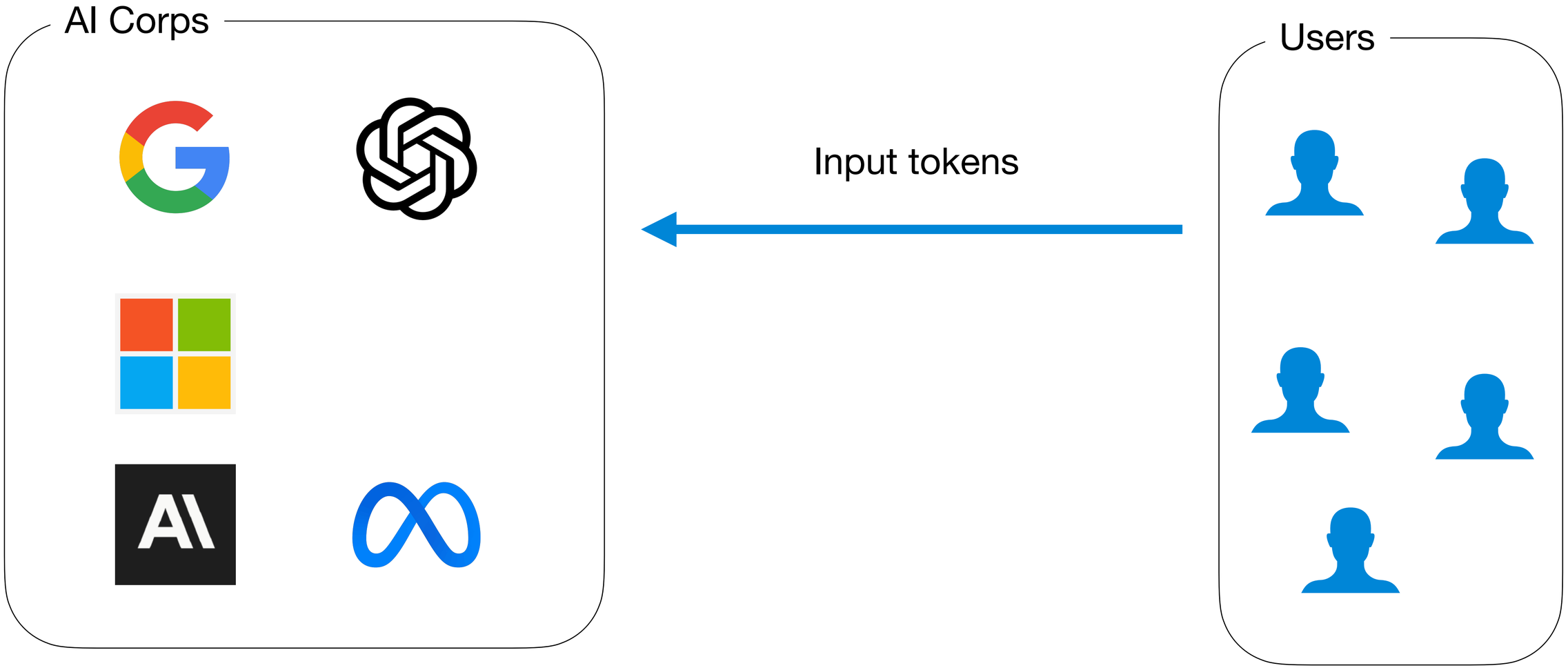


But privacy…

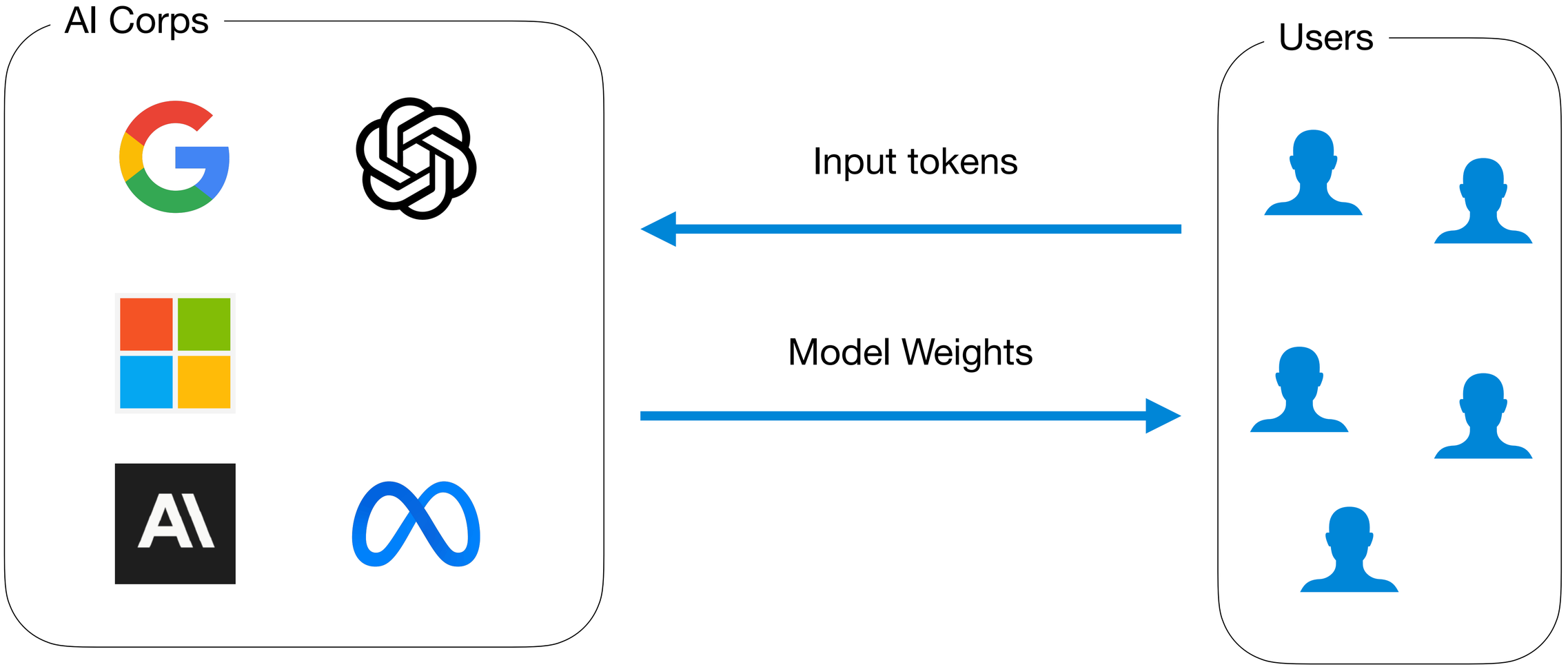




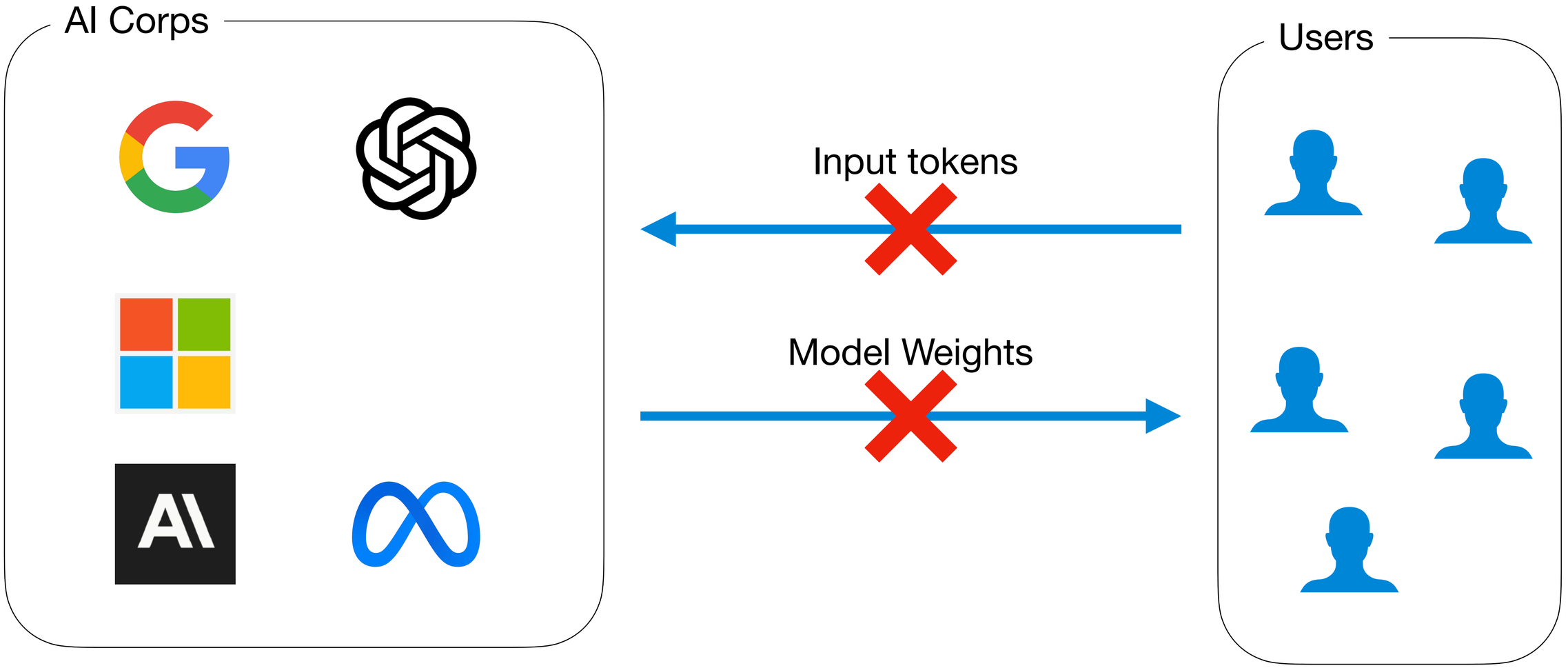
But privacy…



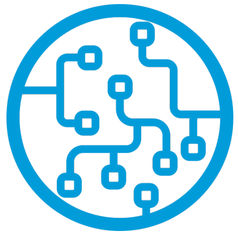
But privacy…



But privacy…



Secure Inference

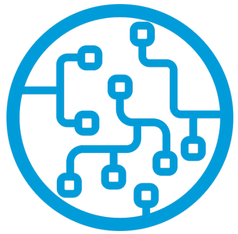


W

x

“What are some good places to visit in Toronto?”

Secure Inference



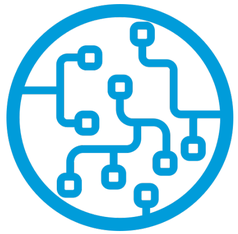
W

x

2PC

"What are some good places to visit in Toronto?"

Secure Inference



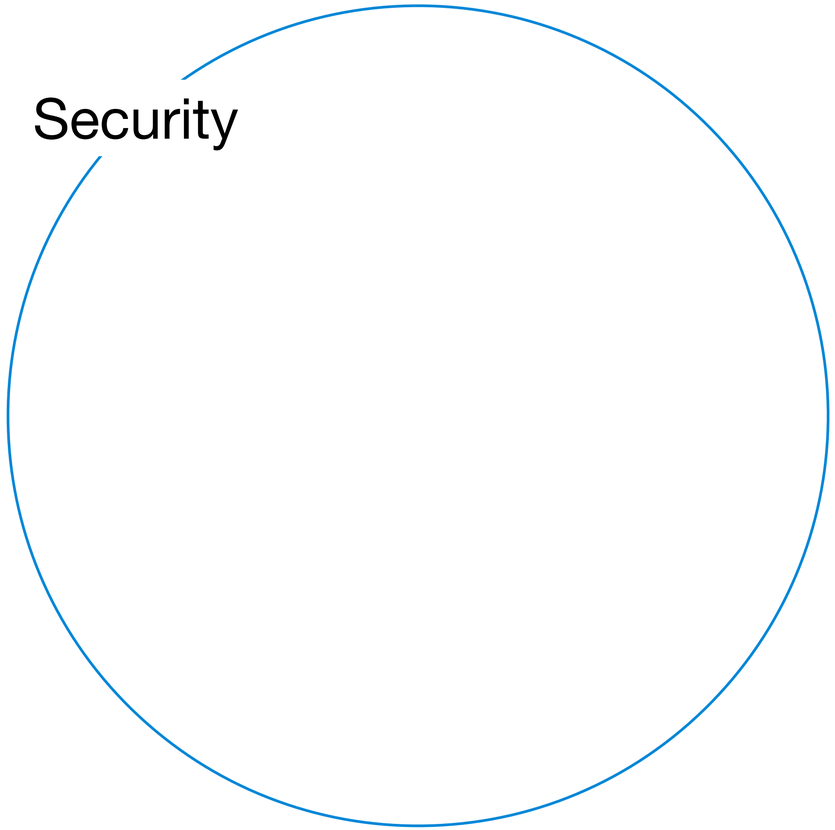
W

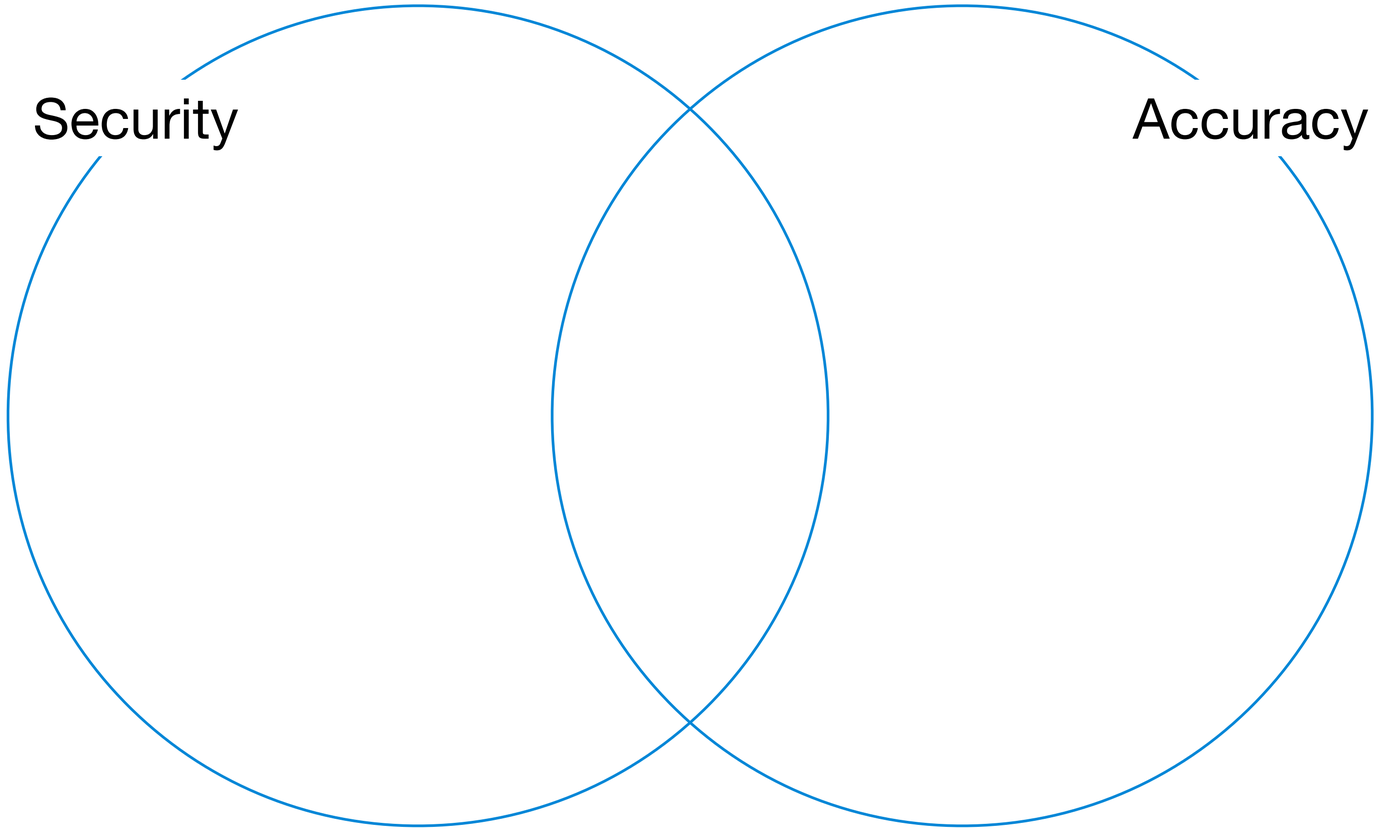
x

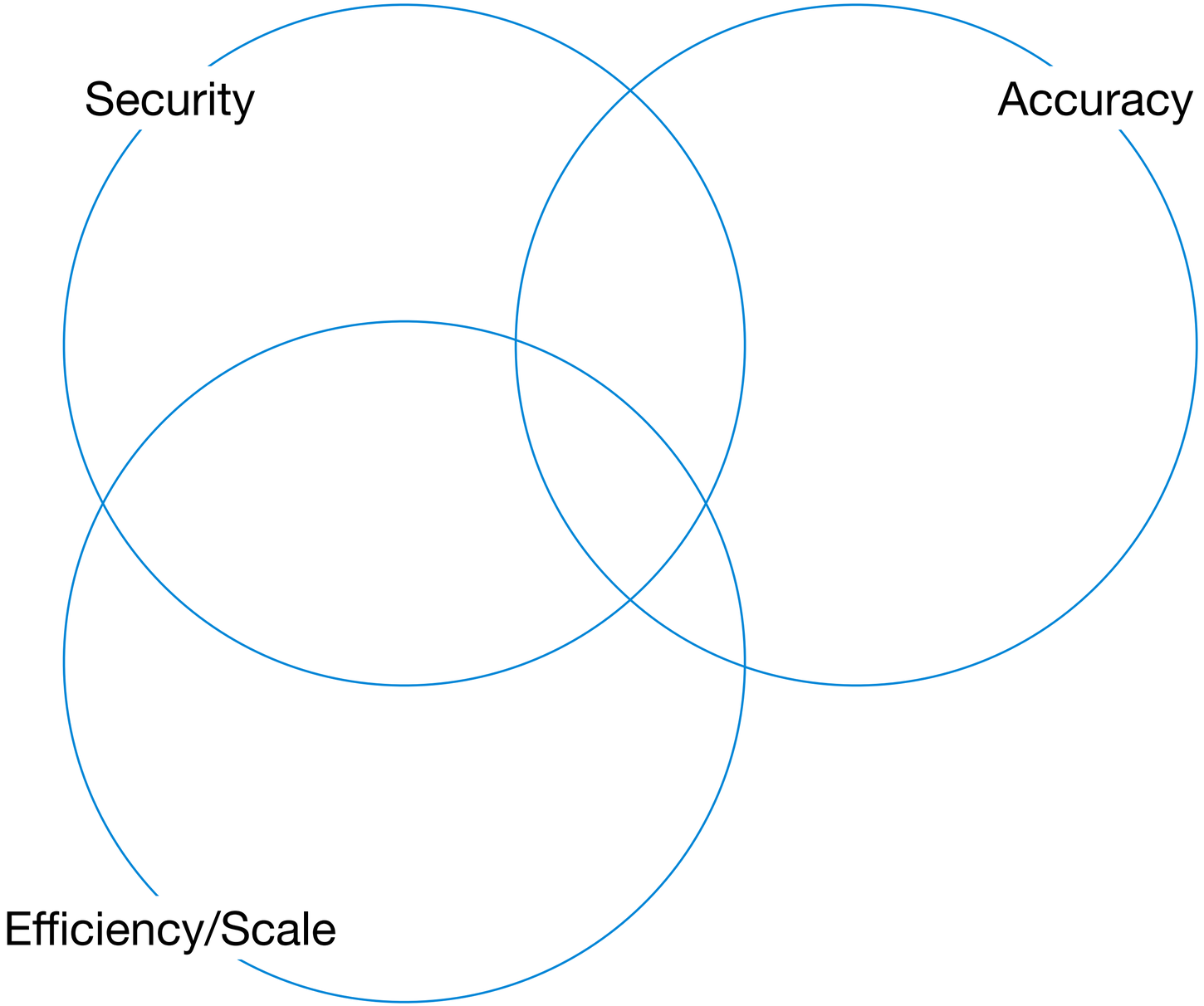
2PC

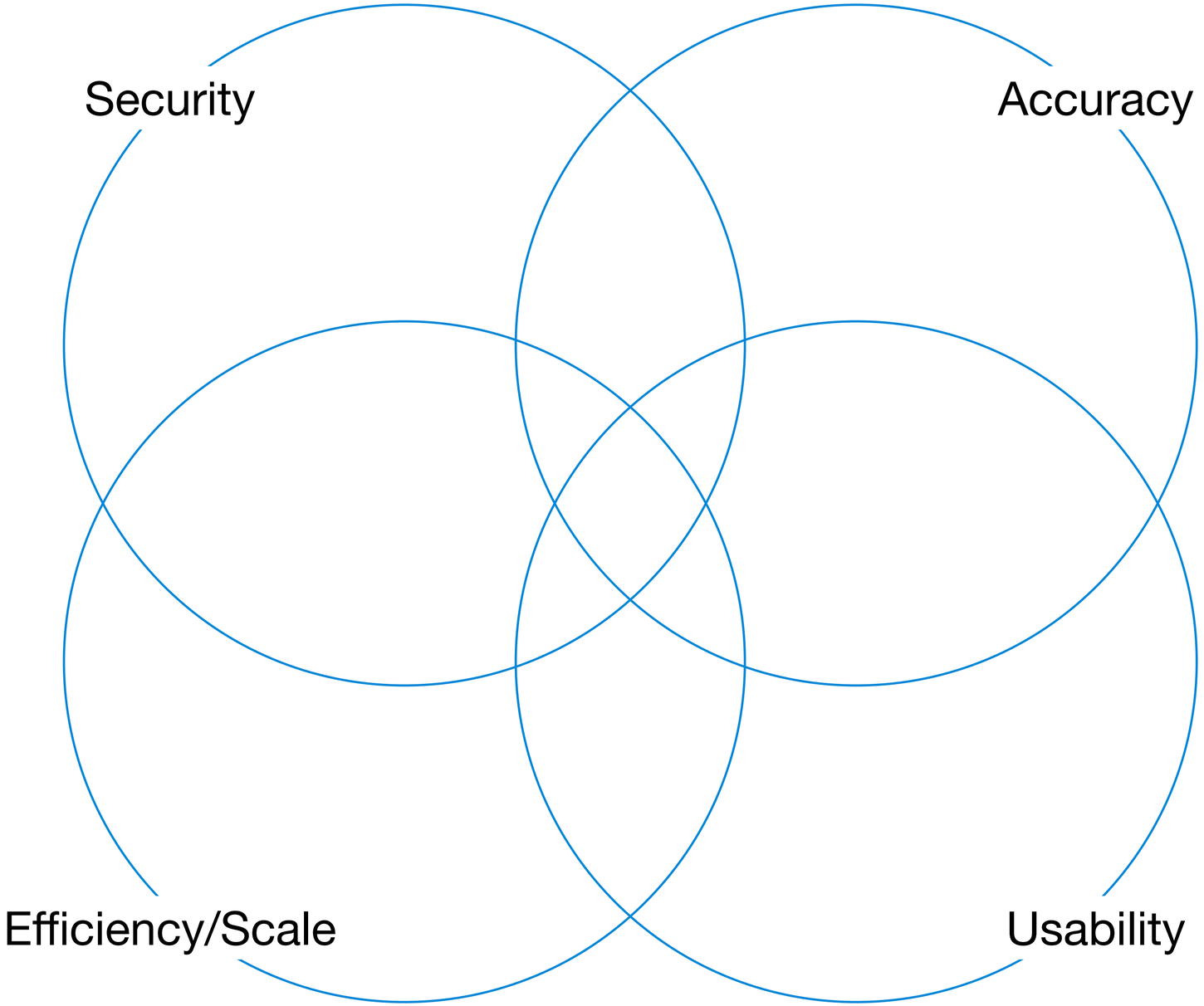
$y = M(W, x)$

"What are some good places to visit in Toronto?"


Security


Security
Accuracy


Security
Accuracy
Efficiency/Scale


Security
Accuracy
Efficiency/Scale
Usability

	Accuracy	Security	Efficiency/ Scalability	Usability

	Accuracy	Security	Efficiency/ Scalability	Usability
Iron	✓	✓		✓

	Accuracy	Security	Efficiency/ Scalability	Usability
Iron	✓	✓		✓
CrypTen			⋱	✓

	Accuracy	Security	Efficiency/ Scalability	Usability
Iron	✓	✓		✓
CrypTen			⋰	✓
MPCFormer	✓		✓	

	Accuracy	Security	Efficiency/ Scalability	Usability
Iron	✓	✓		✓
CrypTen			⋰	✓
MPCFormer	✓		✓	
SIGMA	✓	✓	✓	✓

Results

	PyTorch	SIGMA
GPT2		
GPT Neo		
LLaMA2-7B		
LLaMA2-13B		

Accuracy on LAMBADA dataset

Results

	PyTorch	SIGMA
GPT2	32.46	33.28
GPT Neo	57.46	57.81
LLaMA2-7B	70.17	70.01
LLaMA2-13B	73.14	72.98

Accuracy on LAMBADA dataset

Results

	CrypTen	MPCFormer	SIGMA
BERT Base			
BERT Large			
GPT2			
GPT Neo			
LLAMA-7B			
LLAMA-13B			

Performance comparison for next-token generation

Results

	CrypTen	MPCFormer	SIGMA
BERT Base	21.55	11.06	1.84
BERT Large	54.53	29.21	4.73
GPT2			
GPT Neo			
LLAMA-7B			
LLAMA-13B			

Performance comparison for next-token generation

Results

	CrypTen	MPCFormer	SIGMA
BERT Base	21.55	11.06	1.84
BERT Large	54.53	29.21	4.73
GPT2	20.45	-	1.61
GPT Neo		-	
LLAMA-7B		-	
LLAMA-13B		-	

Performance comparison for next-token generation

Results

	CrypTen	MPCFormer	SIGMA
BERT Base	21.55	11.06	1.84
BERT Large	54.53	29.21	4.73
GPT2	20.45	-	1.61
GPT Neo	108.30	-	7.43
LLAMA-7B		-	
LLAMA-13B		-	

Performance comparison for next-token generation

Results

	CrypTen	MPCFormer	SIGMA
BERT Base	21.55	11.06	1.84
BERT Large	54.53	29.21	4.73
GPT2	20.45	-	1.61
GPT Neo	108.30	-	7.43
LLAMA-7B	overflow	-	27.01
LLAMA-13B	overflow	-	44.13

Performance comparison for next-token generation

Results

	PyTorch w/ Google Colab	SIGMA
GPT2		
GPT Neo		

Comparison with insecure execution

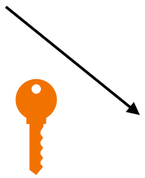
Results

	PyTorch w/ Google Colab	SIGMA
GPT2	0.04	1.61
GPT Neo	0.14	7.43

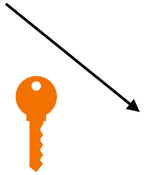
Comparison with insecure execution

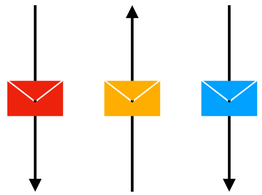
Threat Model

Threat Model



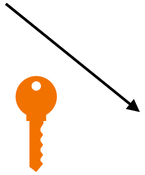
Threat Model

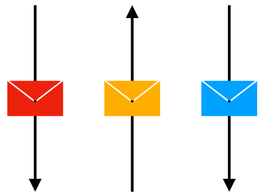




“What are some good places to visit in Toronto?”

Threat Model





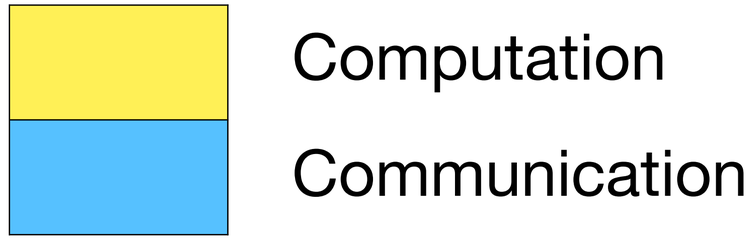
“What are some good places to visit in Toronto?”

$A($  $)$

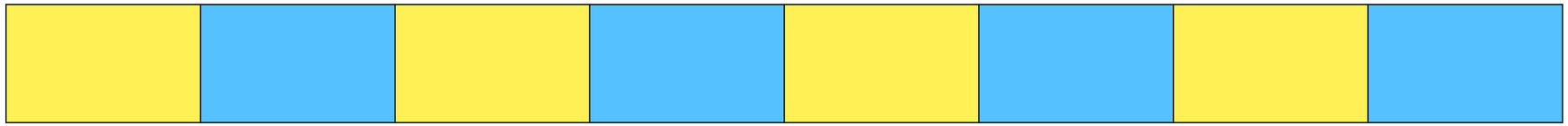
Leak()

Function Secret Sharing (FSS) based 2PC

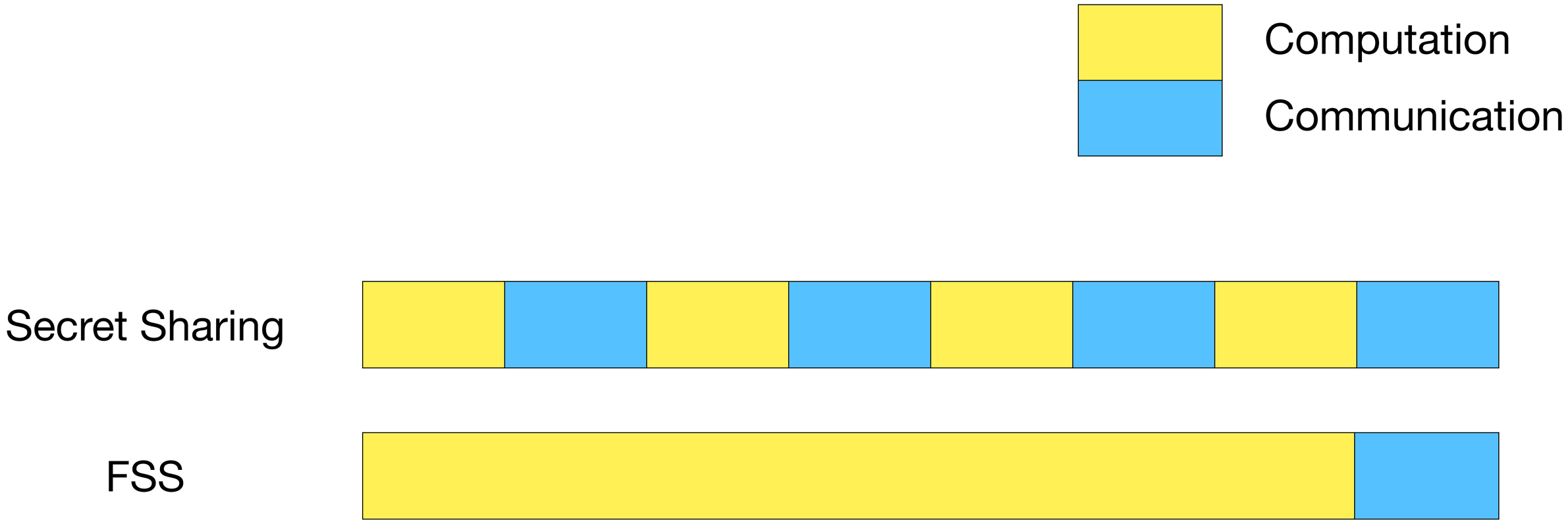
Function Secret Sharing (FSS) based 2PC



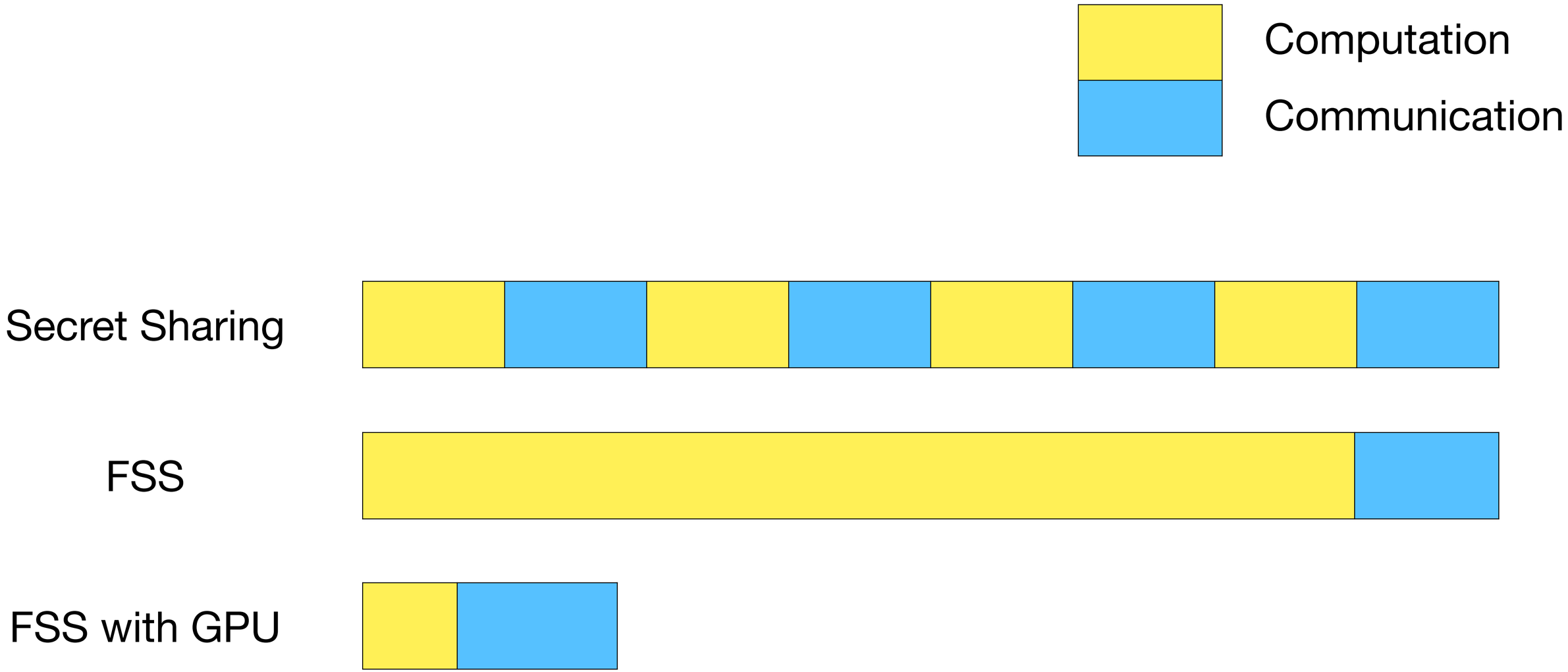
Secret Sharing



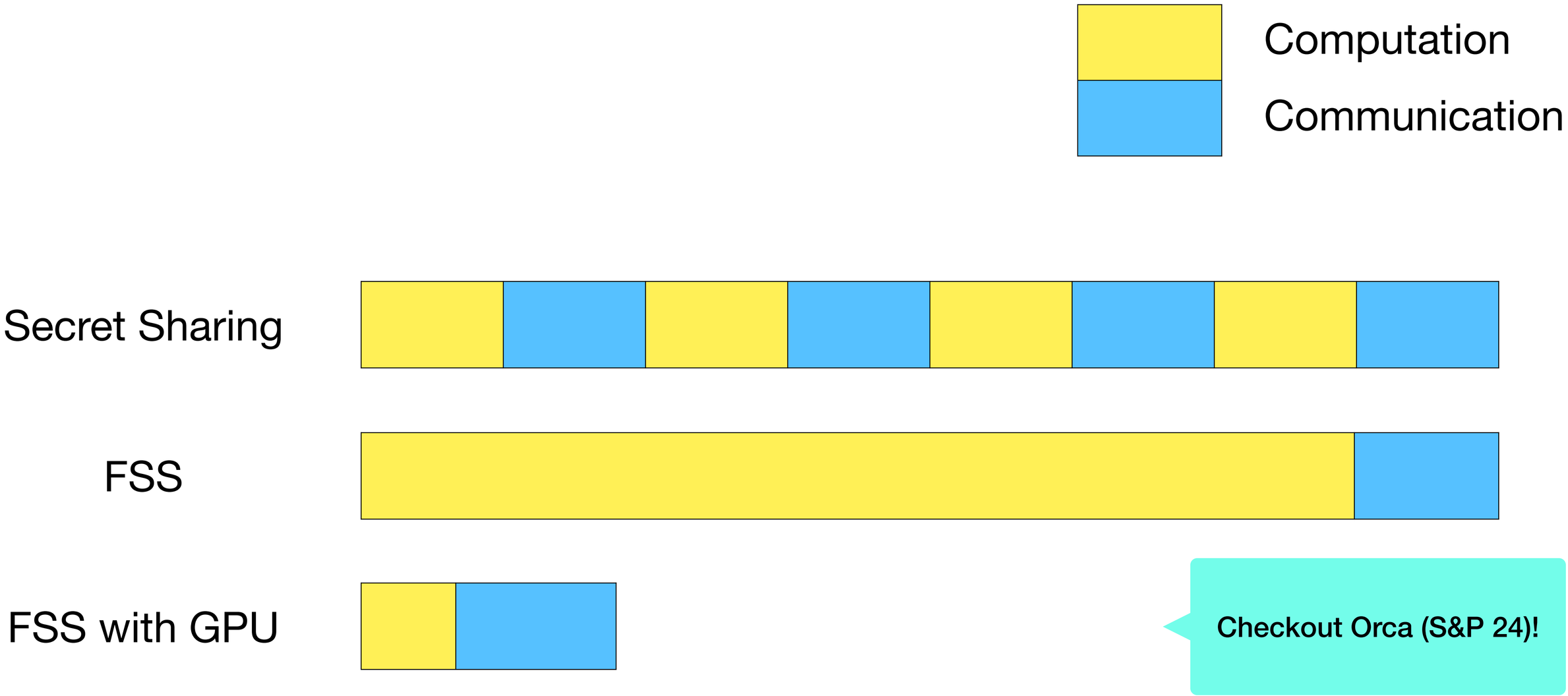
Function Secret Sharing (FSS) based 2PC



Function Secret Sharing (FSS) based 2PC



Function Secret Sharing (FSS) based 2PC



Existing Protocols

Existing Protocols

Integer Multiplication

Key size: $3n$

Comm.: $2n$

Existing Protocols

Integer Multiplication

Key size: $3n$
Comm.: $2n$

Truncation

Key size: $\sim\lambda n$
Comm.: $4n$

Existing Protocols

Integer Multiplication

Key size: $3n$
Comm.: $2n$

Truncation

Key size: $\sim\lambda n$
Comm.: $4n$

Comparison

Key size: $\sim\lambda n$
Comm.: 2

Fixed-Point Arithmetic

n bit width, s scale

$$N = 2^n$$

Fixed-Point Arithmetic

n bit width, s scale

$$N = 2^n$$

$$x \in \mathbb{R}$$

Encode

$$y = \lfloor x \cdot 2^s \rfloor \in \mathbb{U}_N$$

Fixed-Point Arithmetic

n bit width, s scale

$$N = 2^n$$

$x \in \mathbb{R}$

Encode

$y = \lfloor x \cdot 2^s \rfloor \in \mathbb{U}_N$

$y \in \mathbb{U}_N$

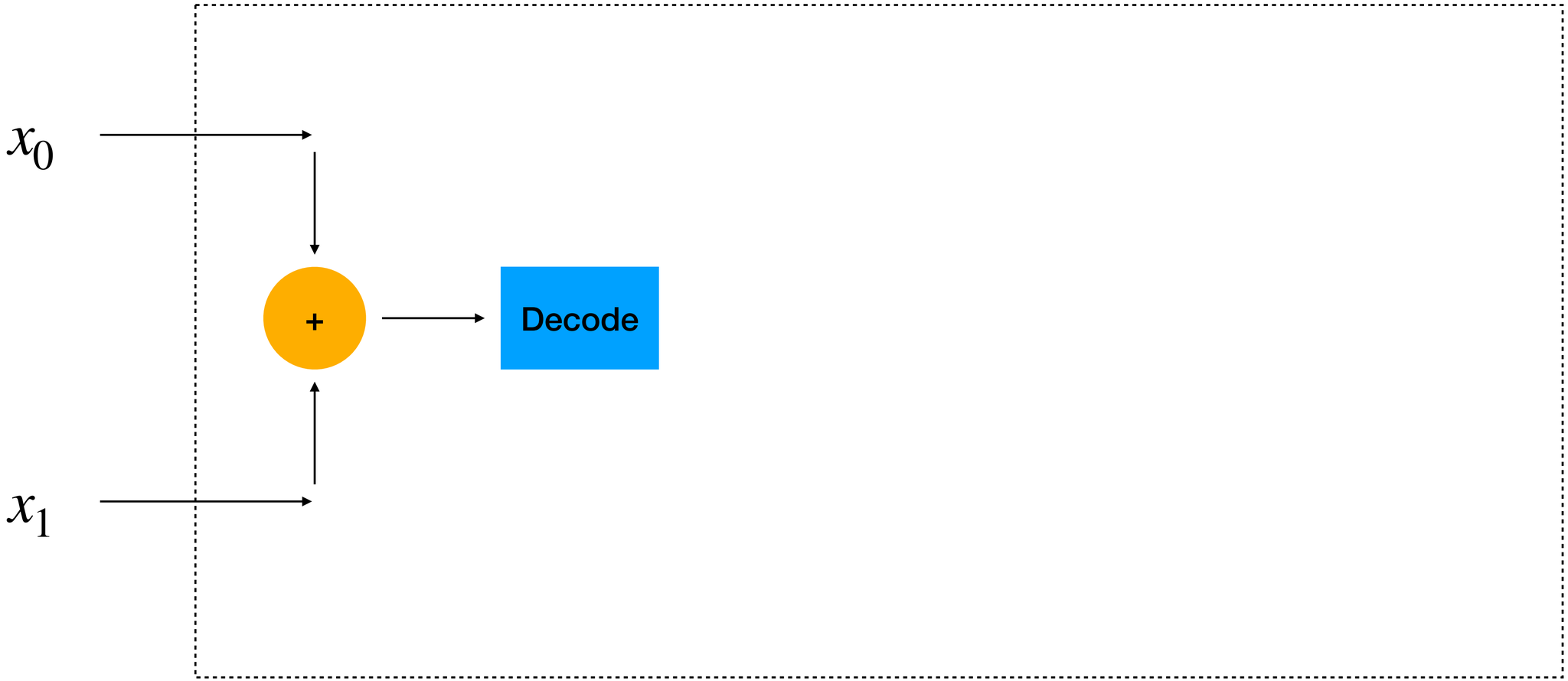
Decode

$x = \frac{y}{2^s} \in \mathbb{R}$

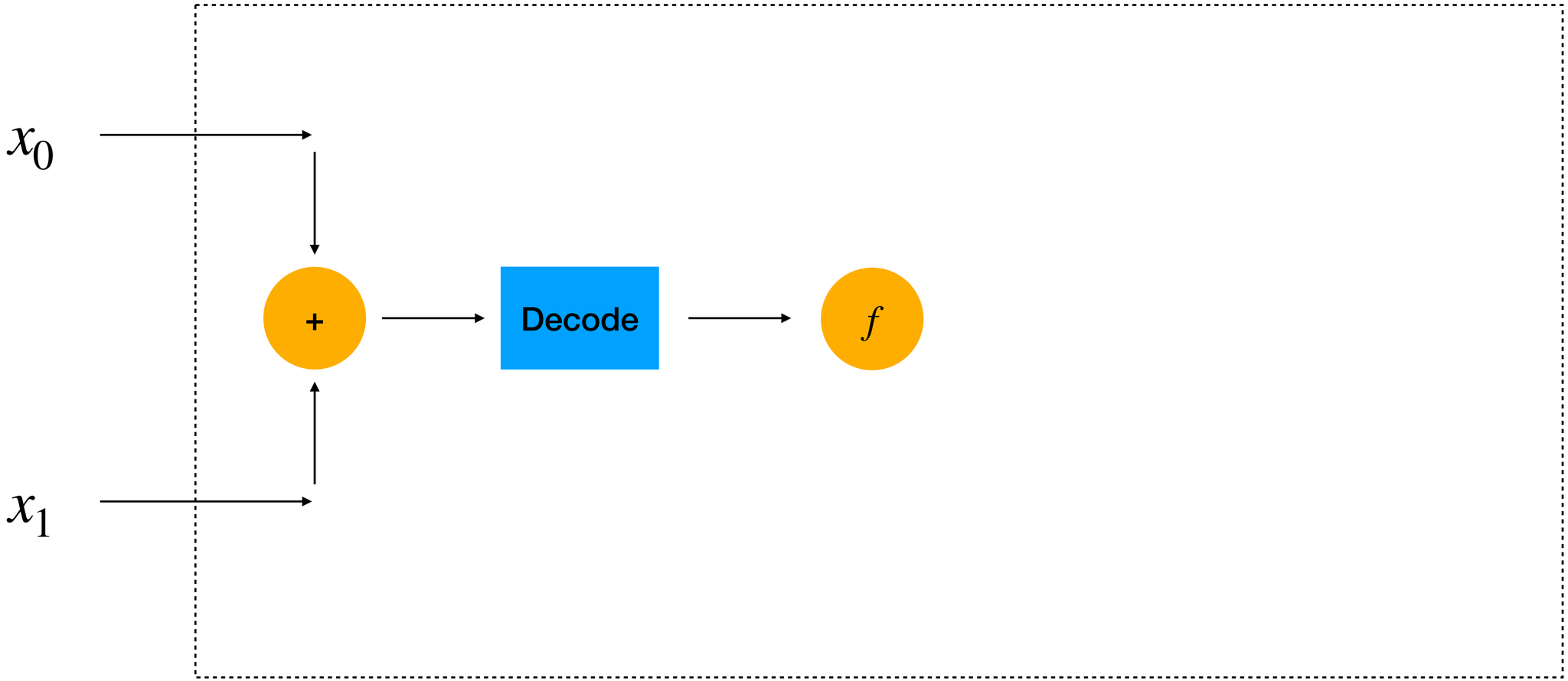
Fixed-Point Arithmetic

Fixed-Point Arithmetic

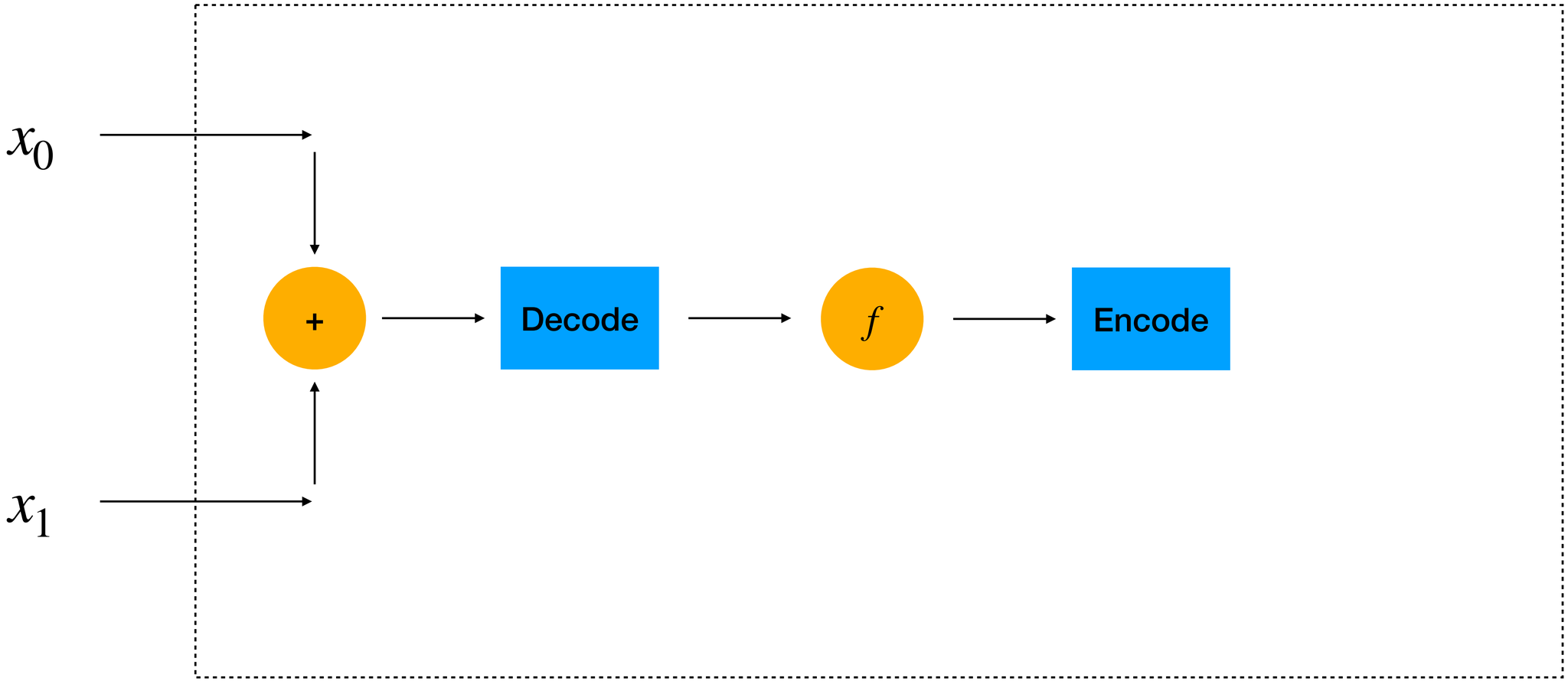
Fixed-Point Arithmetic



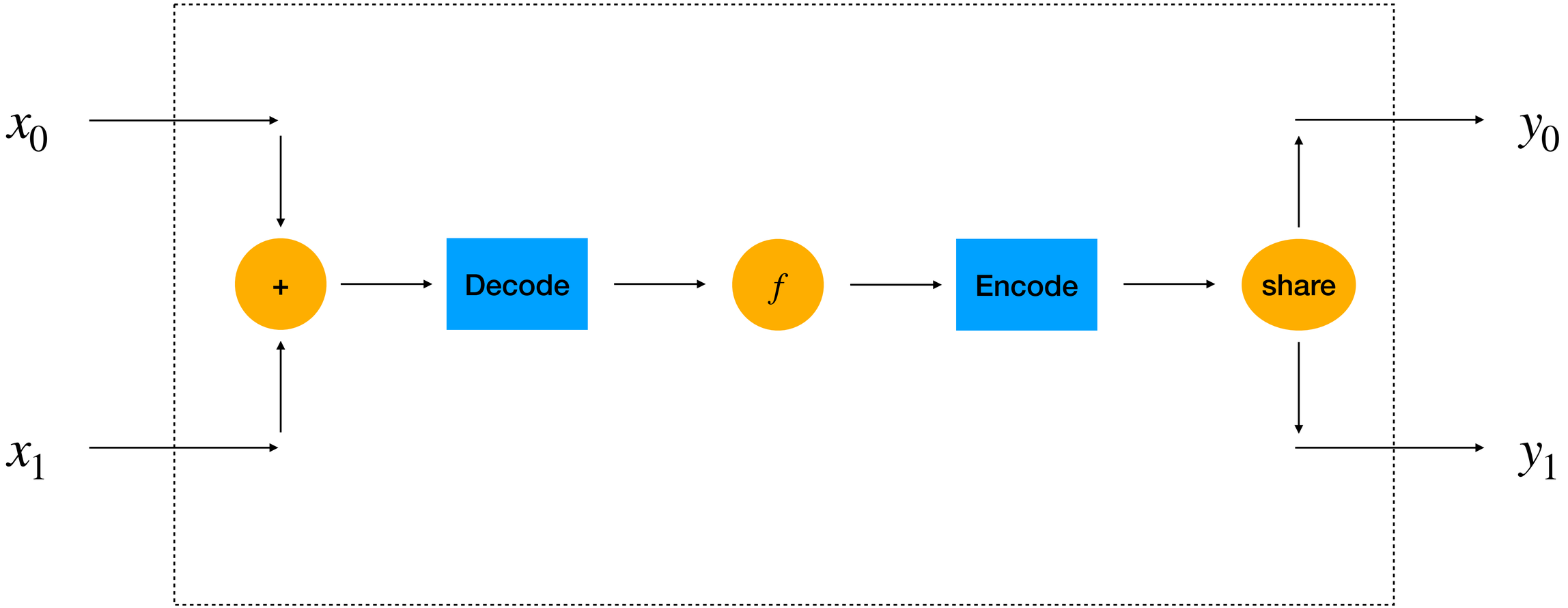
Fixed-Point Arithmetic



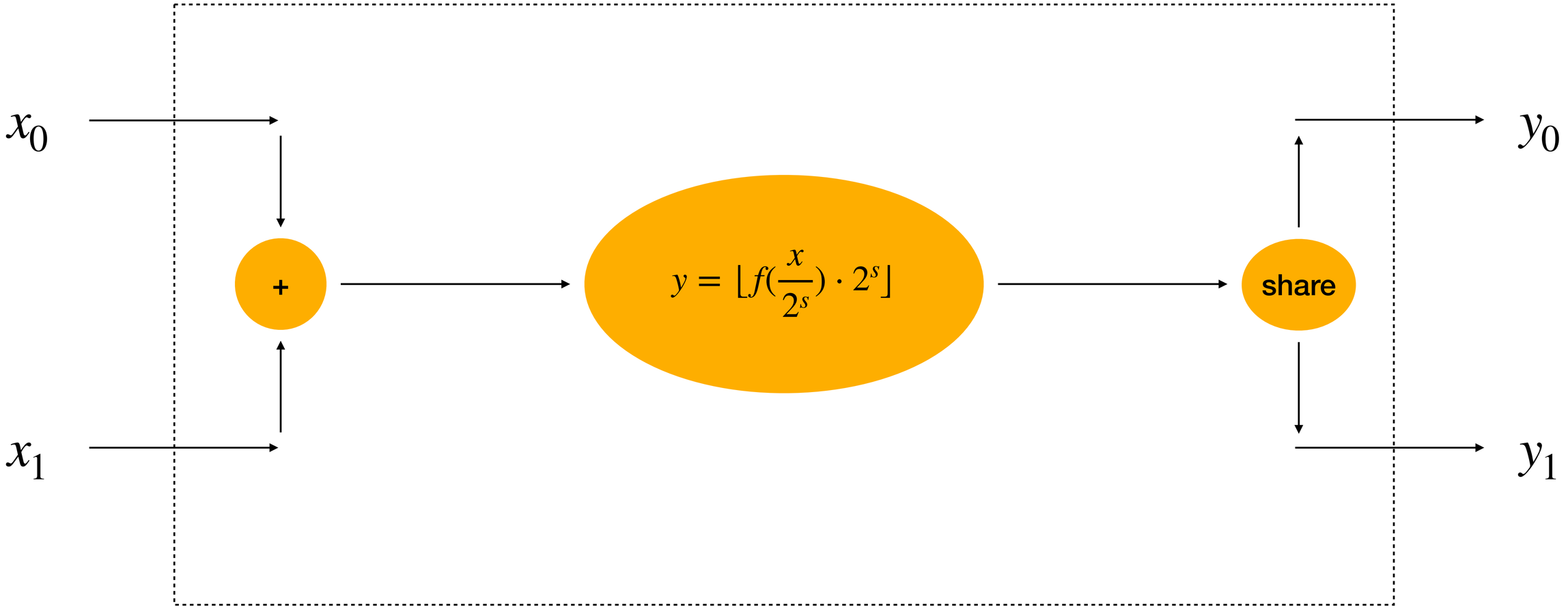
Fixed-Point Arithmetic



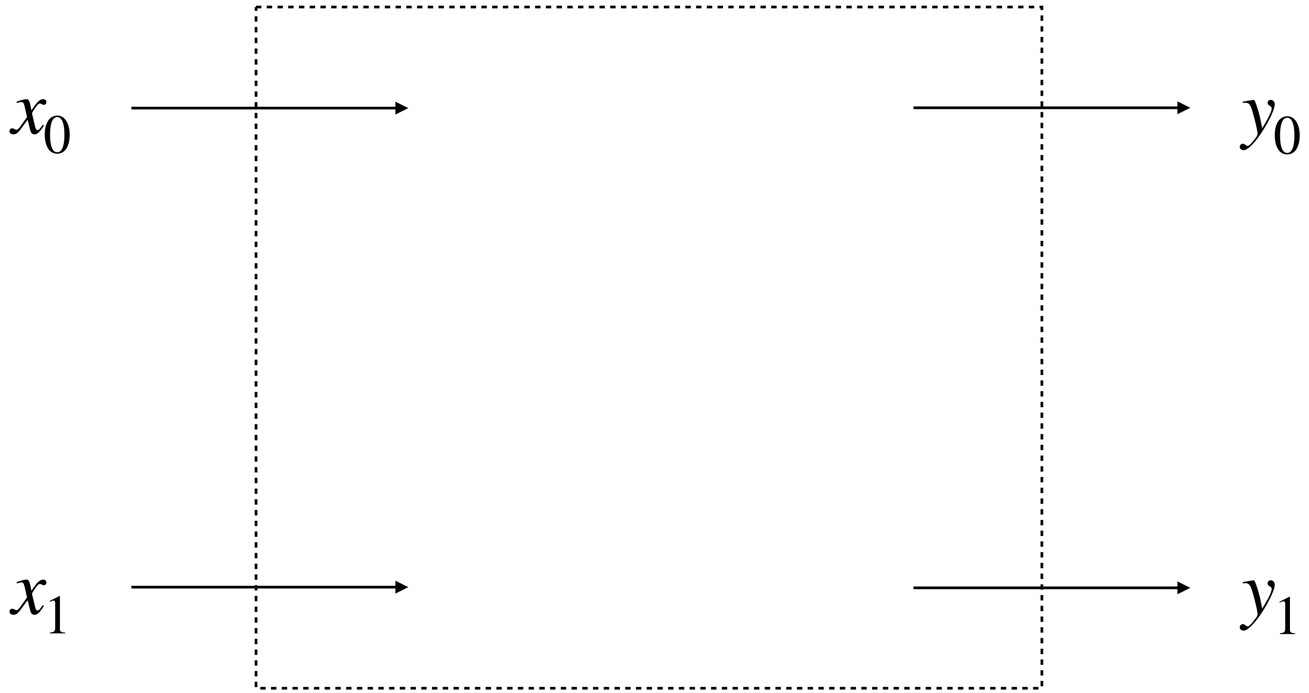
Fixed-Point Arithmetic



Fixed-Point Arithmetic

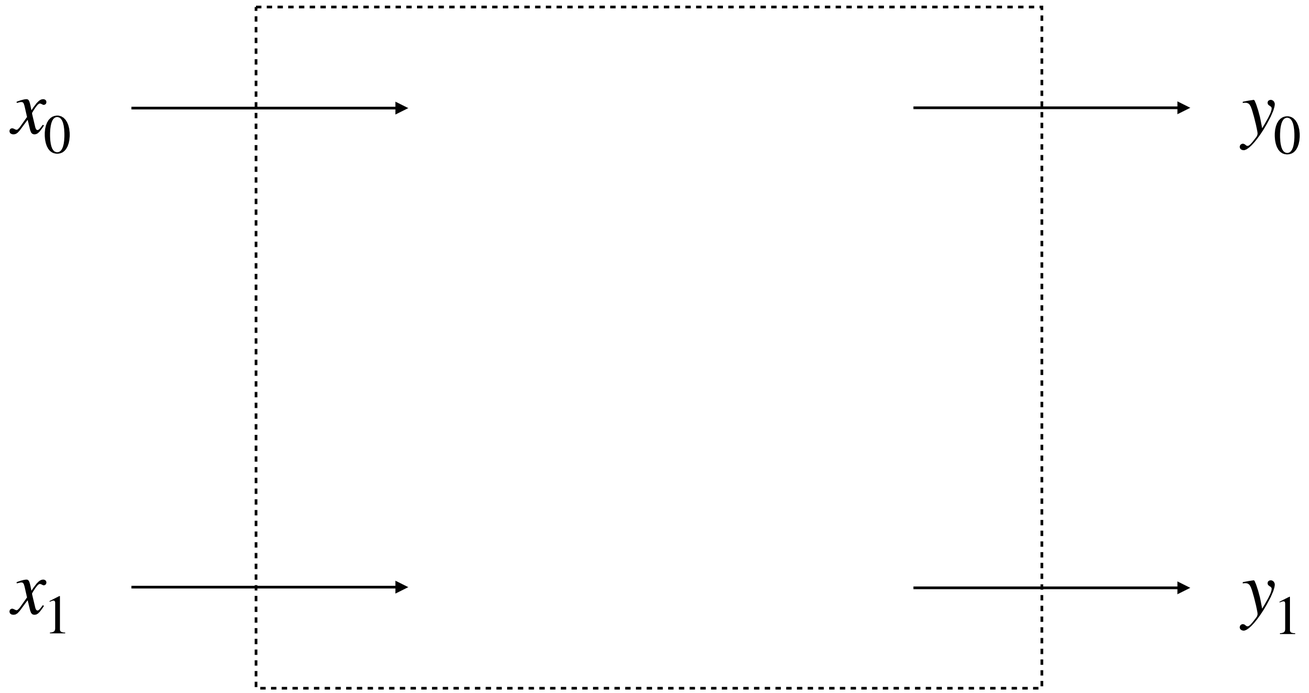


Look-Up Table based Protocol



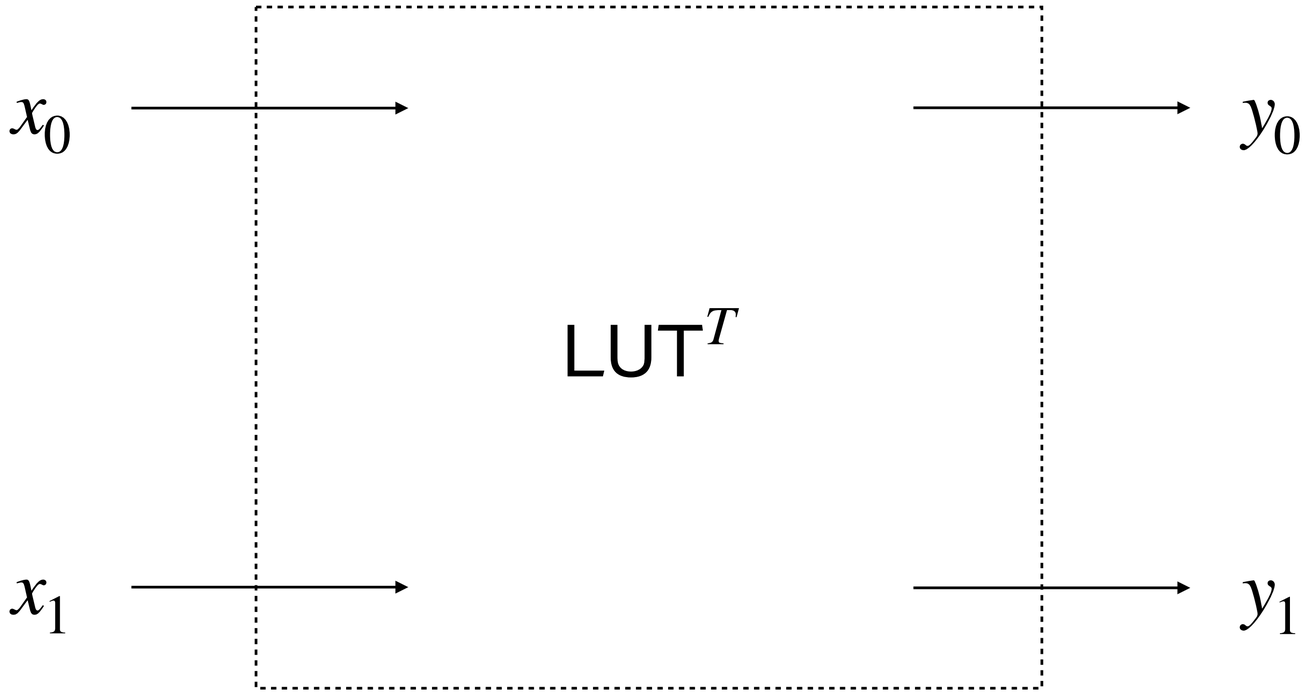
Look-Up Table based Protocol

$$T[i] = \lfloor f(\frac{i}{2^s}) \cdot 2^s \rfloor$$



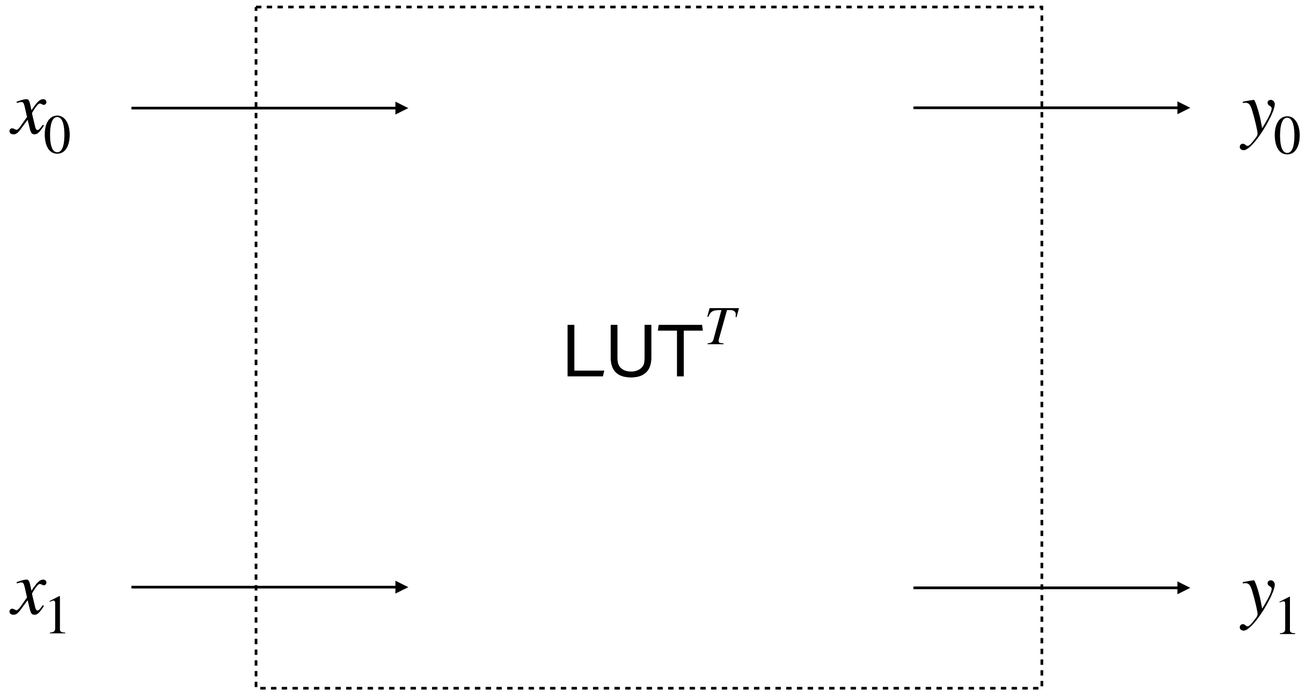
Look-Up Table based Protocol

$$T[i] = \lfloor f(\frac{i}{2^s}) \cdot 2^s \rfloor$$



Look-Up Table based Protocol

$$T[i] = \lfloor f(\frac{i}{2^s}) \cdot 2^s \rfloor$$



Pika (Wagh, 22)

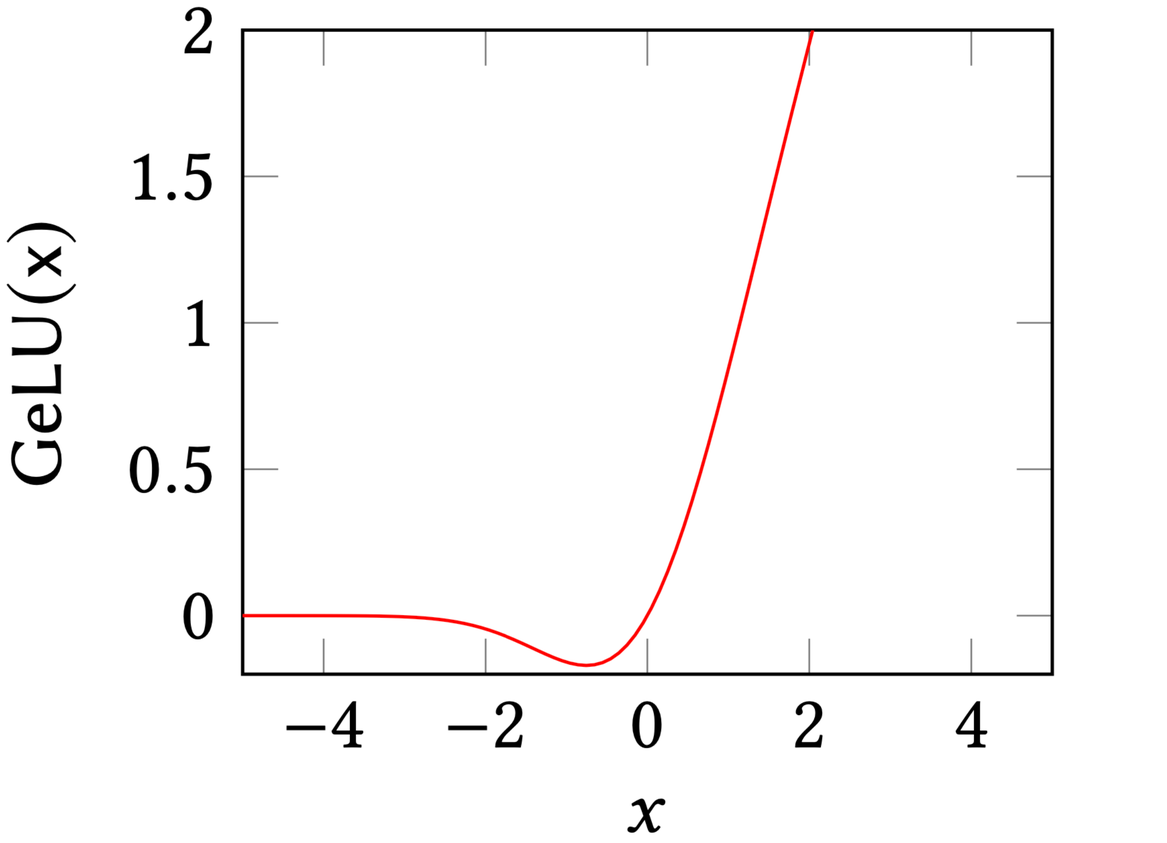
Key Size.: $O(n\lambda)$

Comm.: $O(n)$

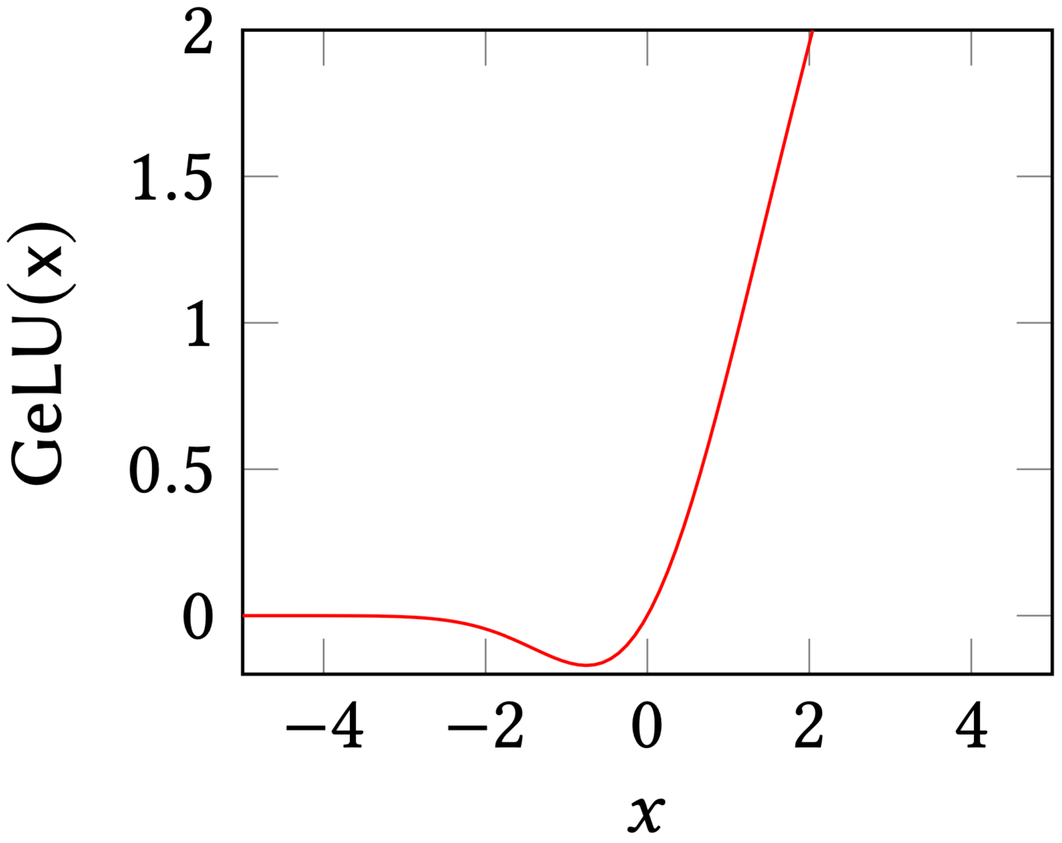
Comp.: $O(2^n)$

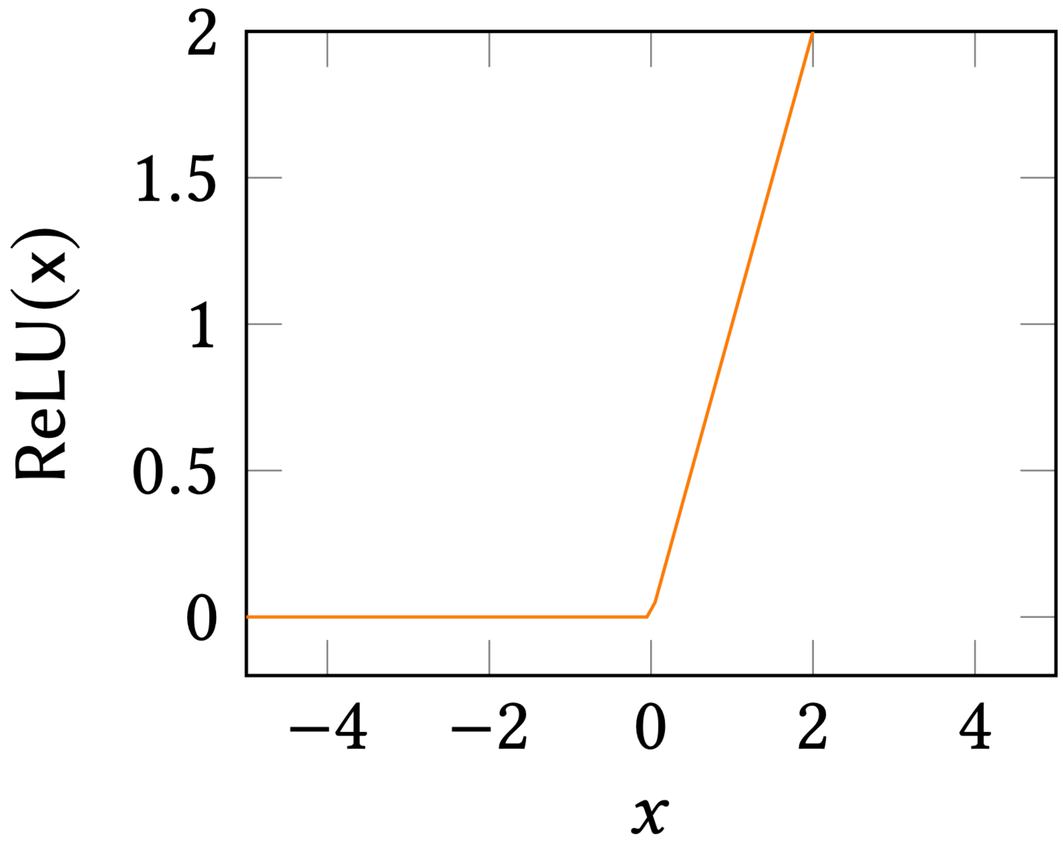
GeLU

GeLU



GeLU

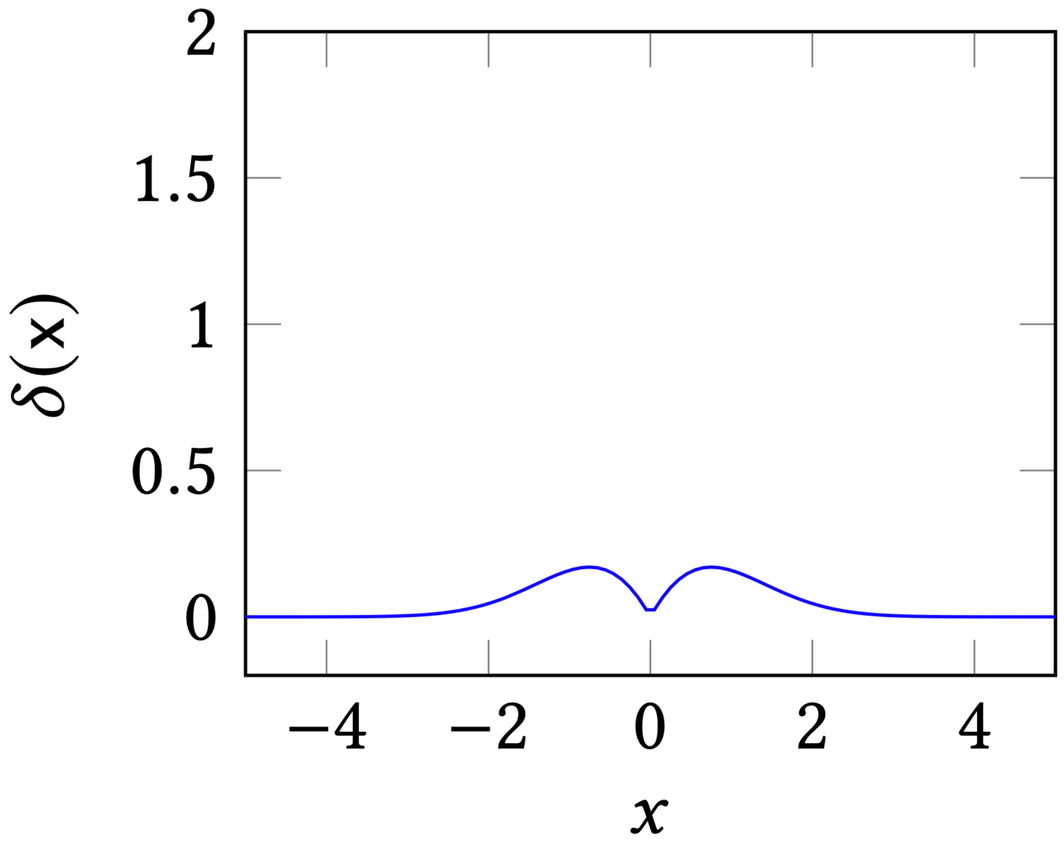




$$\mathrm{ReLU}(x) = \max(x, 0)$$

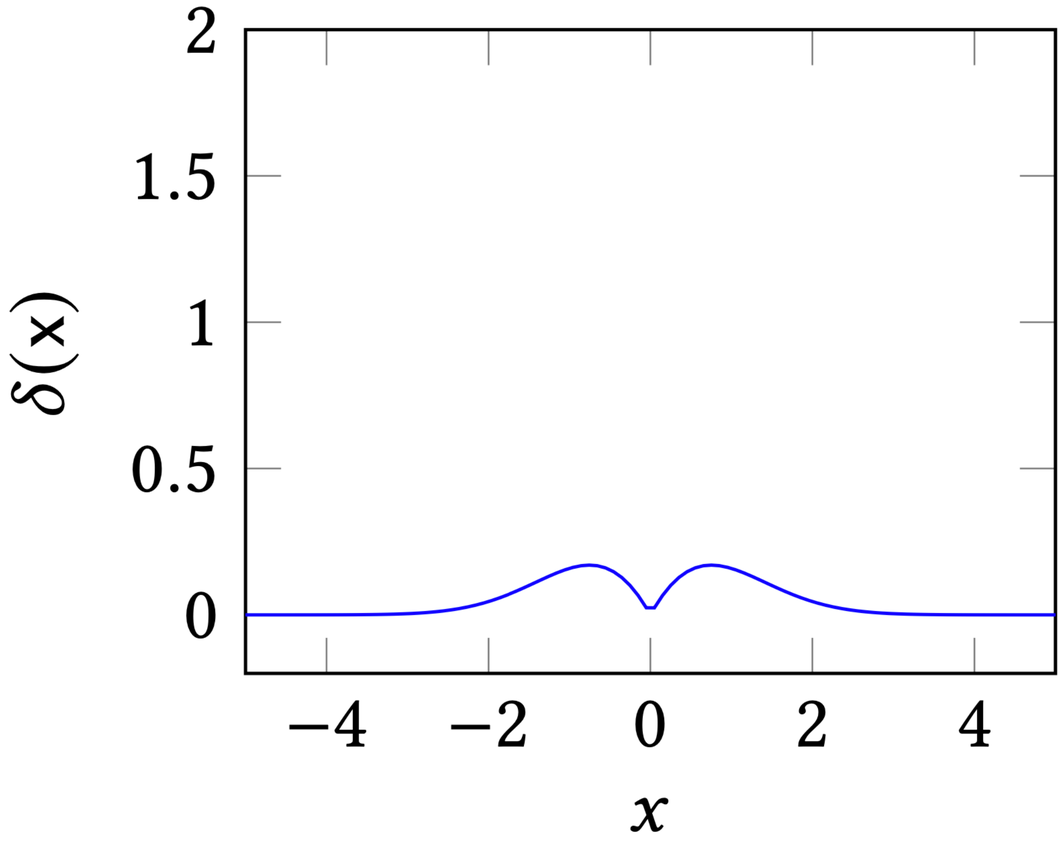
GeLU

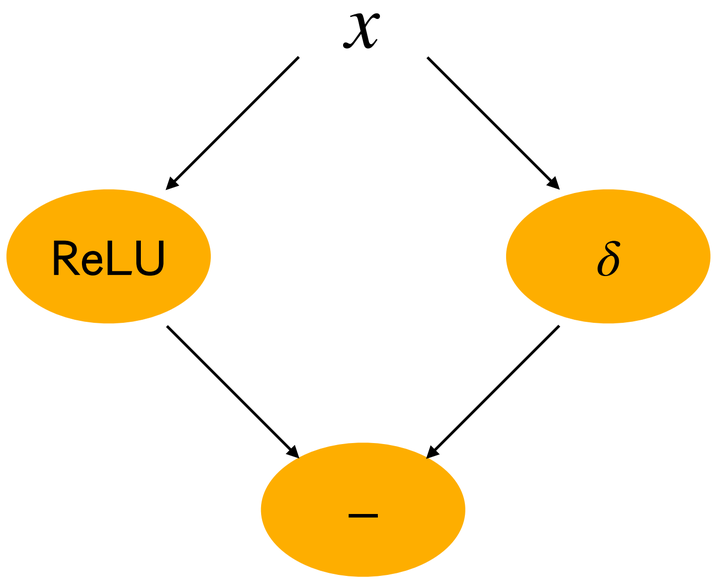
GeLU



$$\delta(x) = \mathrm{ReLU}(x) - \mathrm{GeLU}(x)$$

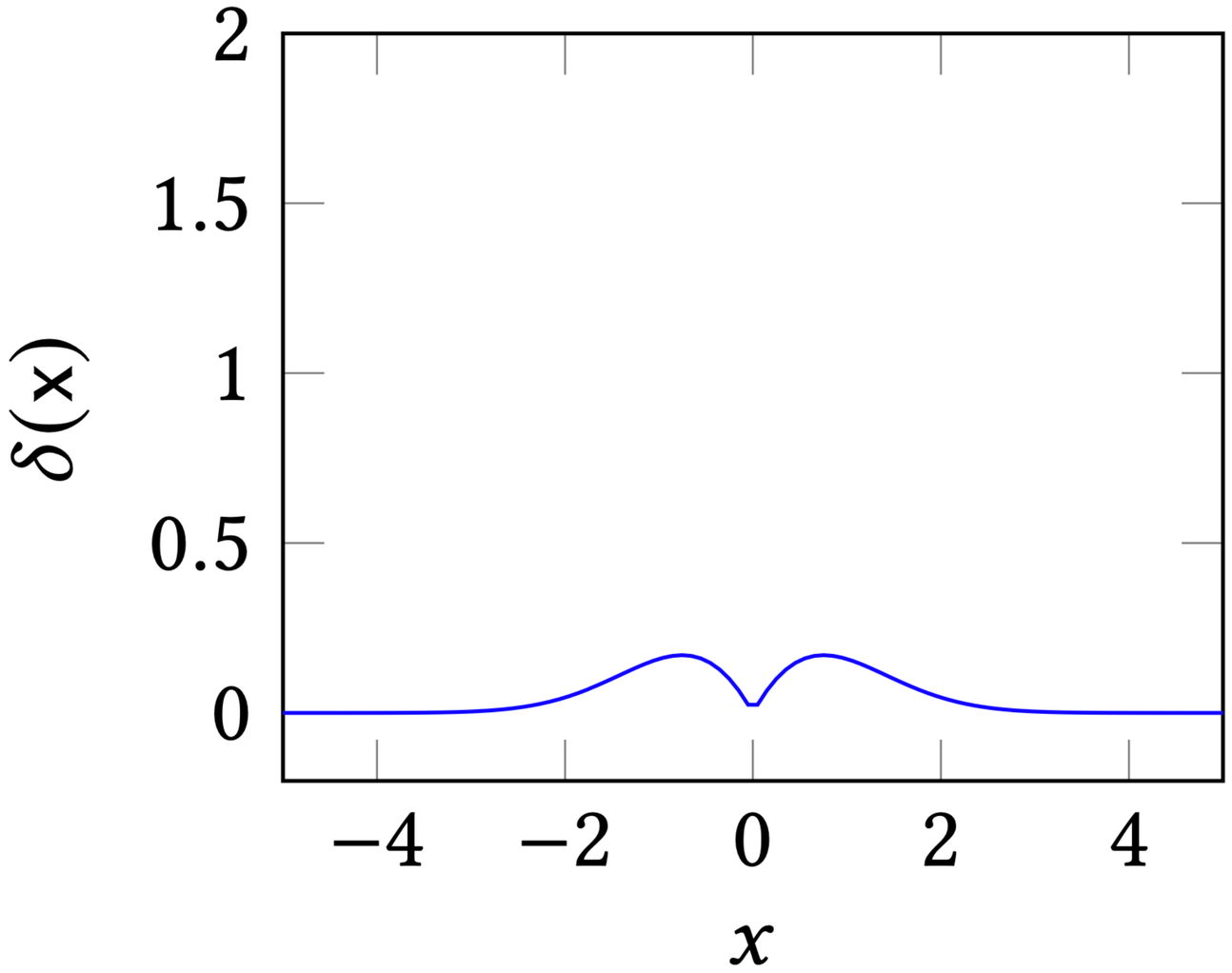
GeLU



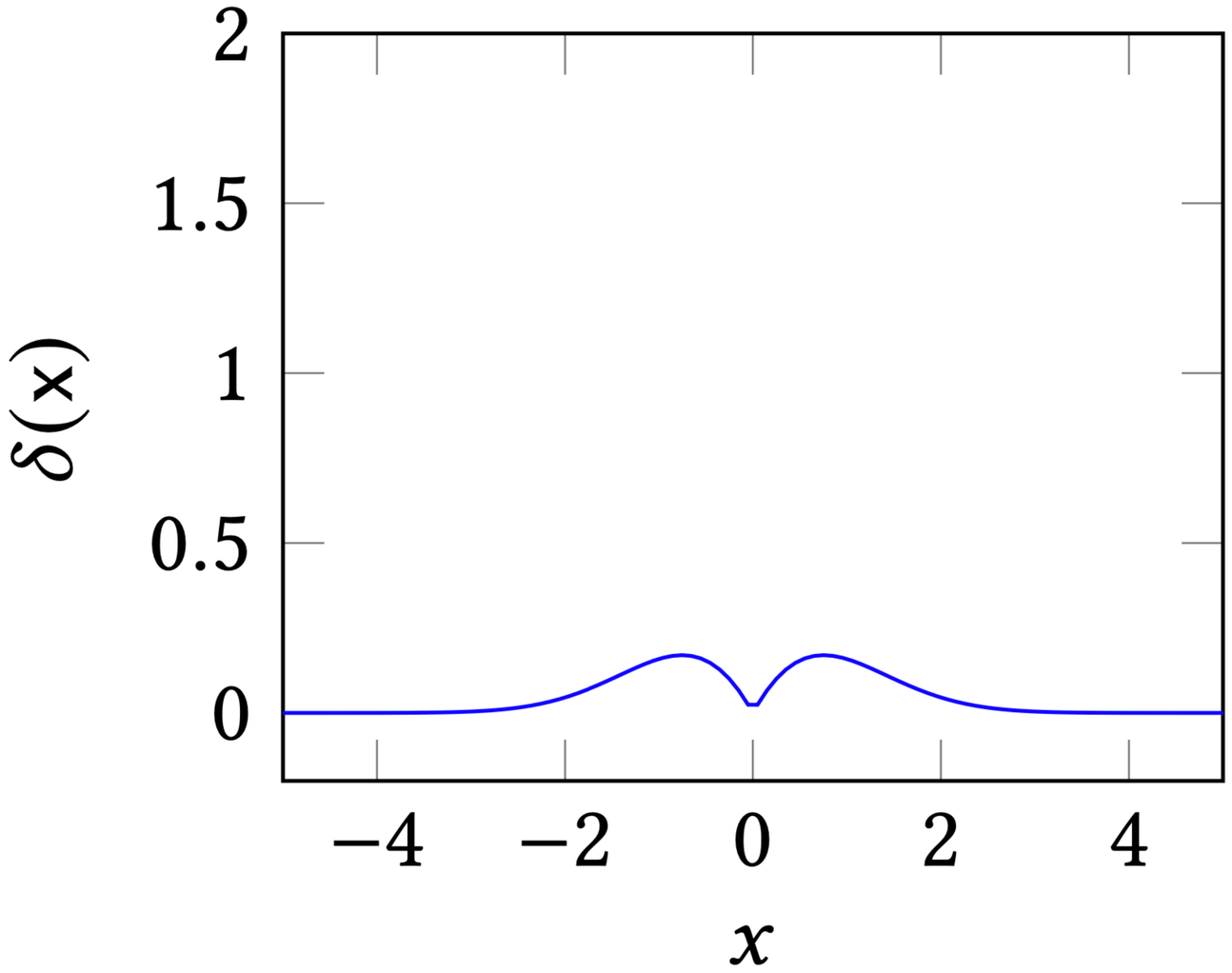


$$\delta(x) = \mathrm{ReLU}(x) - \mathrm{GeLU}(x)$$

GeLU



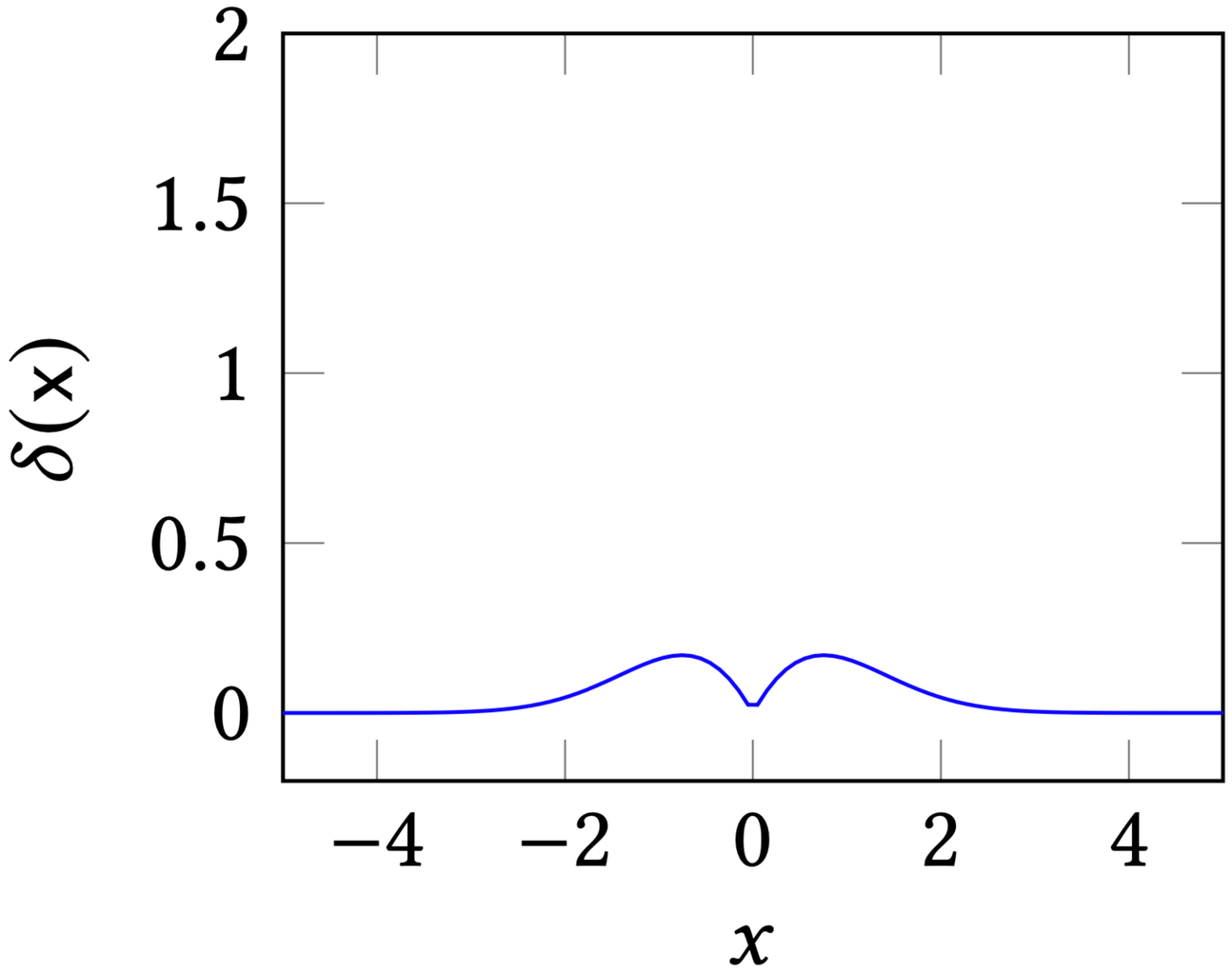
GeLU



1. “Looks” symmetric

$$\delta(x) = \delta(-x)$$

GeLU



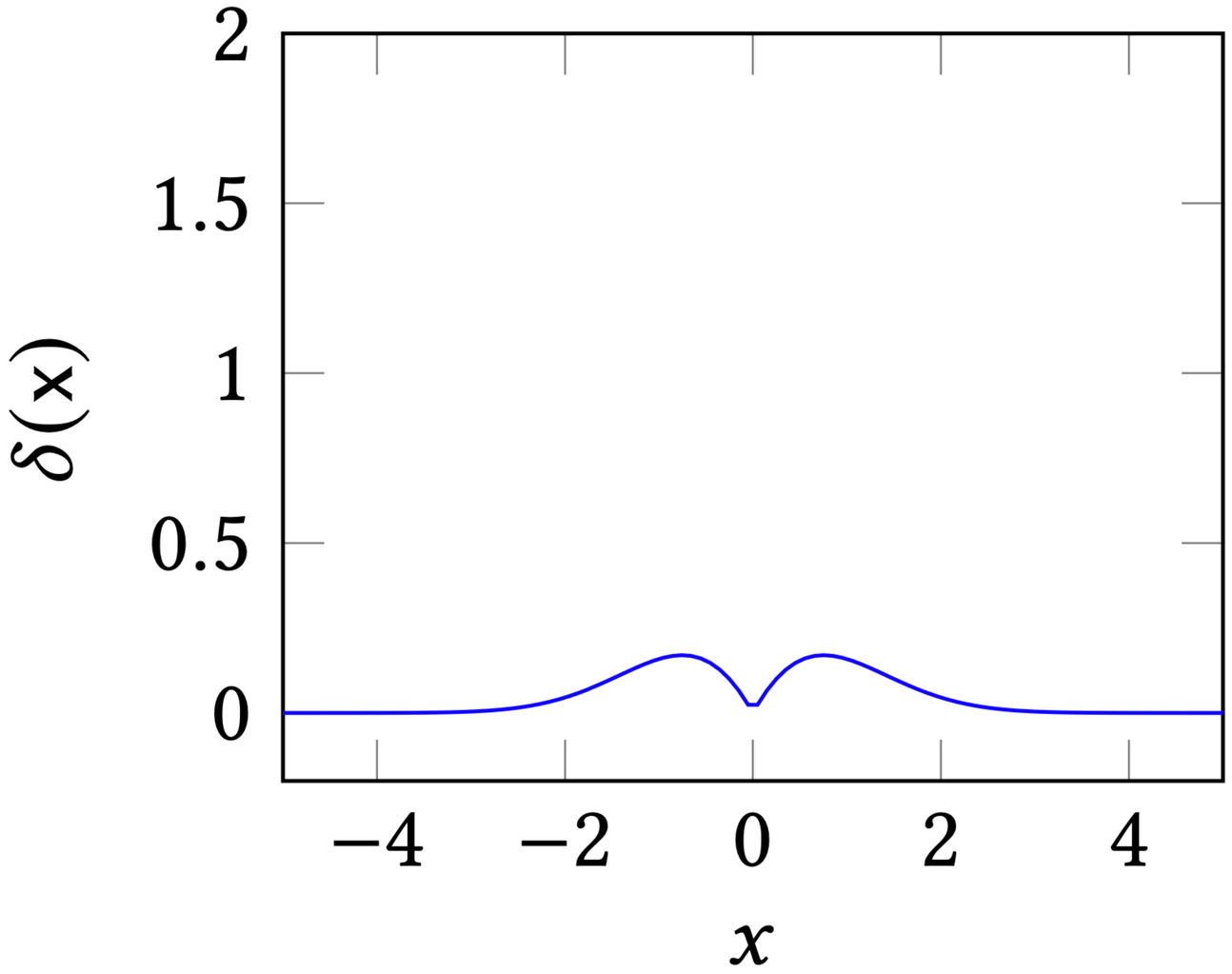
1. “Looks” symmetric

$$\delta(x) = \delta(-x)$$

2. Damps away after 4

$$\delta(4) \approx 0$$

GeLU



1. “Looks” symmetric

$$\delta(x) = \delta(-x)$$

2. Damps away after 4

$$\delta(4) \approx 0$$

3. Fixed point scale 6 suffices

GeLU - Protocol

GeLU - Protocol

1. $y = x \gg (s - 6)$ ▷ Reduce to scale of 6

GeLU - Protocol

1. $y = x \gg (s - 6)$ ▷ Reduce to scale of 6
2. $d = (y > 0)$ ▷ Calculate if > 0

GeLU - Protocol

1. $y = x \gg (s - 6)$ ▷ Reduce to scale of 6
2. $d = (y > 0)$ ▷ Calculate if > 0
3. $p = d \cdot y$

GeLU - Protocol

1. $y = x \gg (s - 6)$ ▷ Reduce to scale of 6
2. $d = (y > 0)$ ▷ Calculate if > 0
3. $p = d \cdot y$
4. $a = 2p - y$ ▷ Calculate abs value of y

GeLU - Protocol

1. $y = x \gg (s - 6)$ ▷ Reduce to scale of 6
2. $d = (y > 0)$ ▷ Calculate if > 0
3. $p = d \cdot y$
4. $a = 2p - y$ ▷ Calculate abs value of y
5. $c = a$ if $(a < 256)$ else 255 ▷ Clip up below 255

GeLU - Protocol

1. $y = x \gg (s - 6)$ ▷ Reduce to scale of 6
2. $d = (y > 0)$ ▷ Calculate if > 0
3. $p = d \cdot y$
4. $a = 2p - y$ ▷ Calculate abs value of y
5. $c = a$ if $(a < 256)$ else 255 ▷ Clip up below 255
6. **return** $d \cdot x - T^{\delta}[c]$ ▷ Calculate $\text{ReLU}(x) - \delta(x)$

Other Optimisations

Better Truncation Protocol

Better ReLU Protocol

Protocol tailored for GPUs

Effective Bitwidth Optimisation

Attention Mask Optimisation

SyTorch

SyTorch

```cpp
template <typename T>
class FFN : public SytorchModule<T>
{
    using SytorchModule<T>::gelu;

    u64 in;
    u64 hidden;
public:
    FC<T> *up;
    FC<T> *down;

    FFN(u64 in, u64 hidden) : in(in), hidden(hidden)
    {
        up = new FC<T>(in, hidden, true);
        down = new FC<T>(hidden, in, true);
    }

    Tensor<T> &_forward(Tensor<T> &input)
    {
        return down->forward(gelu(up->forward(input)));
    }
};
```

SyTorch

```
template <typename T>
class FFN : public SytorchModule<T>
{
    using SytorchModule<T>::gelu;

    u64 in;
    u64 hidden;
public:
    FC<T> *up;
    FC<T> *down;

    FFN(u64 in, u64 hidden) : in(in), hidden(hidden)
    {
        up = new FC<T>(in, hidden, true);
        down = new FC<T>(hidden, in, true);
    }

    Tensor<T> &_forward(Tensor<T> &input)
    {
        return down->forward(gelu(up->forward(input)));
    }
};
```

```
    GPT2SequenceClassification<T> bert(n_layer, n_head, n_embd);
    bert.init(scale);
    bert.setBackend(new SigmaCPU<T>());
    // bert.setBackend(new Piranha<T>);
    // bert.setBackend(new SecureML<T>);
    // bert.setBackend(new Crypten<T>);
    bert.optimize();
    bert.load("weights.dat");
```

SyTorch

```
template <typename T>
class FFN : public SytorchModule<T>
{
    using SytorchModule<T>::gelu;

    u64 in;
    u64 hidden;
public:
    FC<T> *up;
    FC<T> *down;

    FFN(u64 in, u64 hidden) : in(in), hidden(hidden)
    {
        up = new FC<T>(in, hidden, true);
        down = new FC<T>(hidden, in, true);
    }

    Tensor<T> &_forward(Tensor<T> &input)
    {
        return down->forward(gelu(up->forward(input)));
    }
};
```

```
GPT2SequenceClassification<T> bert(n_layer, n_head, n_embd);
bert.init(scale);
bert.setBackend(new SigmaCPU<T>());
// bert.setBackend(new Piranha<T>);
// bert.setBackend(new SecureML<T>);
// bert.setBackend(new Crypten<T>);
bert.optimize();
bert.load("weights.dat");
```

- GPU support
- Easy backend integration
- Training (CNNs only)
- Fast plaintext emulation

SyTorch

```
template <typename T>
class FFN : public SytorchModule<T>
{
    using SytorchModule<T>::gelu;

    u64 in;
    u64 hidden;
public:
    FC<T> *up;
    FC<T> *down;

    FFN(u64 in, u64 hidden) : in(in), hidden(hidden)
    {
        up = new FC<T>(in, hidden, true);
        down = new FC<T>(hidden, in, true);
    }

    Tensor<T> &_forward(Tensor<T> &input)
    {
        return down->forward(gelu(up->forward(input)));
    }
};
```

```
GPT2SequenceClassification<T> bert(n_layer, n_head, n_embd);
bert.init(scale);
bert.setBackend(new SigmaCPU<T>());
// bert.setBackend(new Piranha<T>);
// bert.setBackend(new SecureML<T>);
// bert.setBackend(new Crypten<T>);
bert.optimize();
bert.load("weights.dat");
```

- GPU support
- Easy backend integration
- Training (CNNs only)
- Fast plaintext emulation

Releases soon!™

Thank you!

https://eprint.iacr.org/2023/1269