# Curl: Private LLMs through Wavelet-Encoded Look-Up Tables

**Manuel B. Santos**[1], Dimitris Mouris[1], Mehmet Ugurbil[1], Stanislaw Jarecki[1,2], José Reis[1], Shubho Sengupta[3] and Miguel de Vega[1]

{ manuel.santos, dimitris, memo, stanislaw.jarecki, jose.reis, miguel }@nillion.com

ssengupta@meta.com

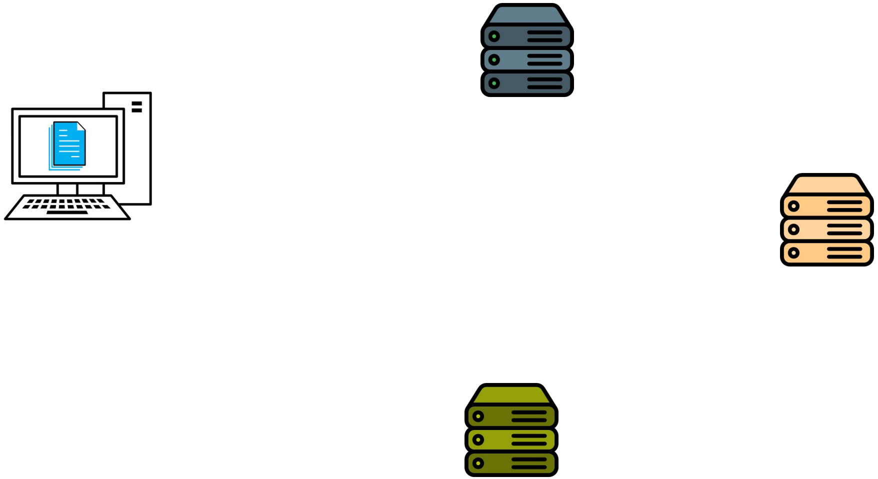
*https://ia.cr/2024/1127*

*https://github.com/jimouris/curl*

1 nillion 2 The University of California Irvine 3 Meta

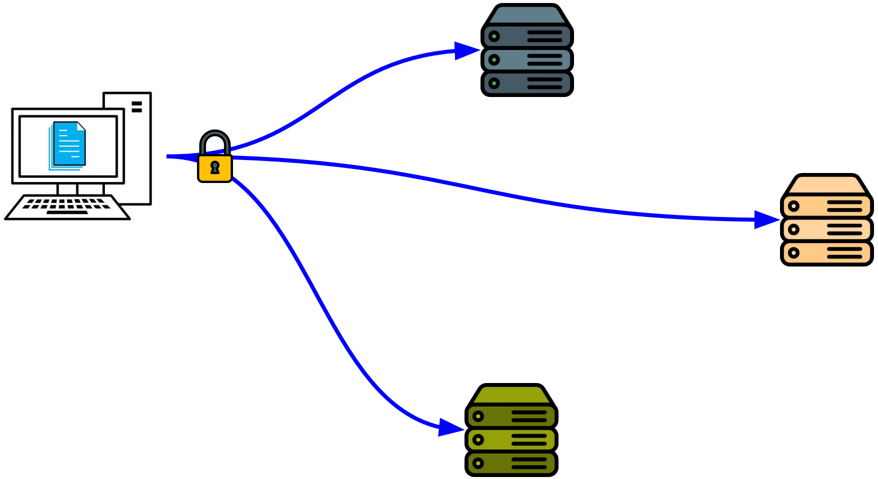
# Secure Multiparty Computation (MPC)

**MPC enables computing directly on private data!**



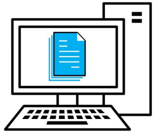
# Secure Multiparty Computation (MPC)

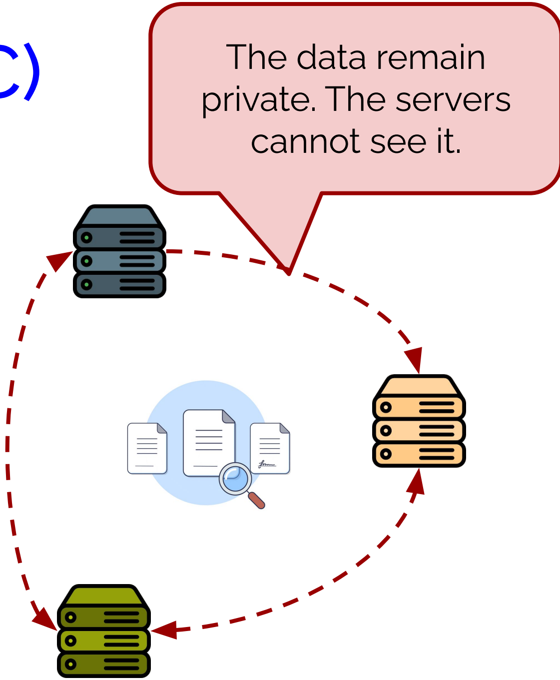
**MPC enables computing directly on private data!**



# Secure Multiparty Computation (MPC)

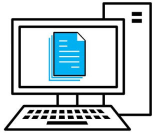
**MPC enables computing directly on private data!**



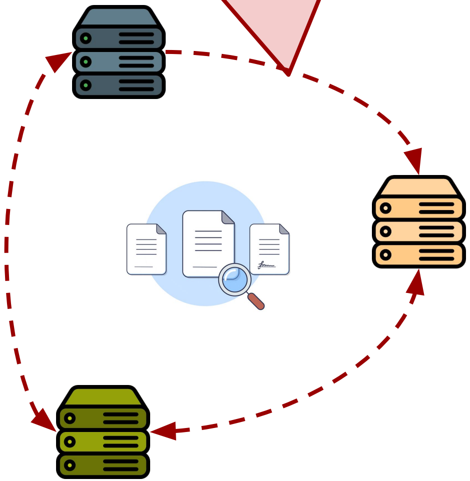


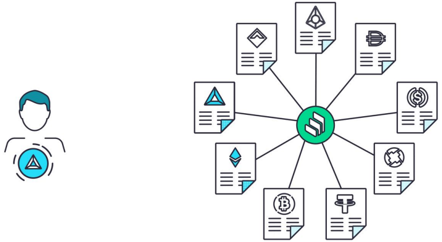
# Secure Multiparty Computation (MPC)

**MPC enables computing directly on private data!**



The data remain private. The servers cannot see it.



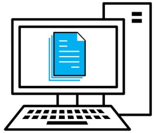


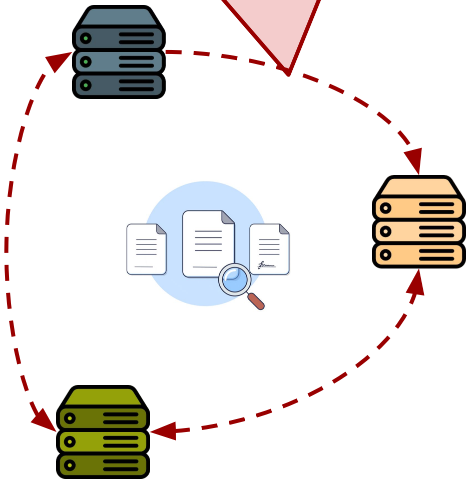
Threshold Signatures

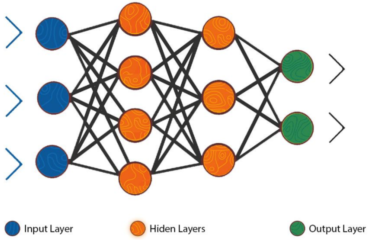
# Secure Multiparty Computation (MPC)

The data remain private. The servers cannot see it.

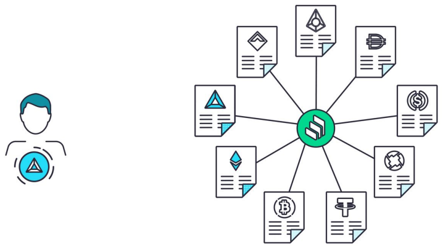
**MPC enables computing directly on private data!**







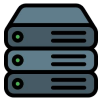

Privacy-preserving ML

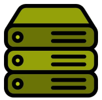


Threshold Signatures

# MPC

Three users want to compute the **sum** of their **private inputs**.





# MPC

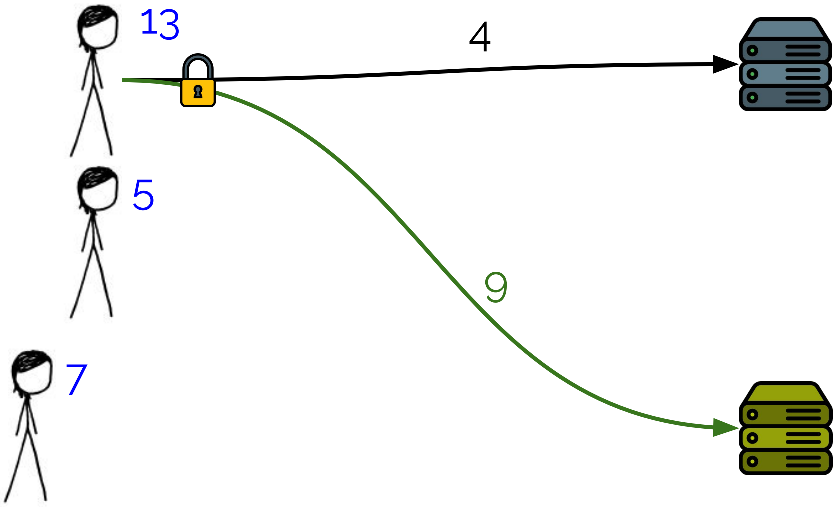
Three users want to compute the **sum** of their **private inputs**.

# MPC

Three users want to compute the **sum** of their **private inputs**.

Each user **secret shares** their input into random looking numbers.

(E.g.,: 4 and 9 reveal nothing about 13)

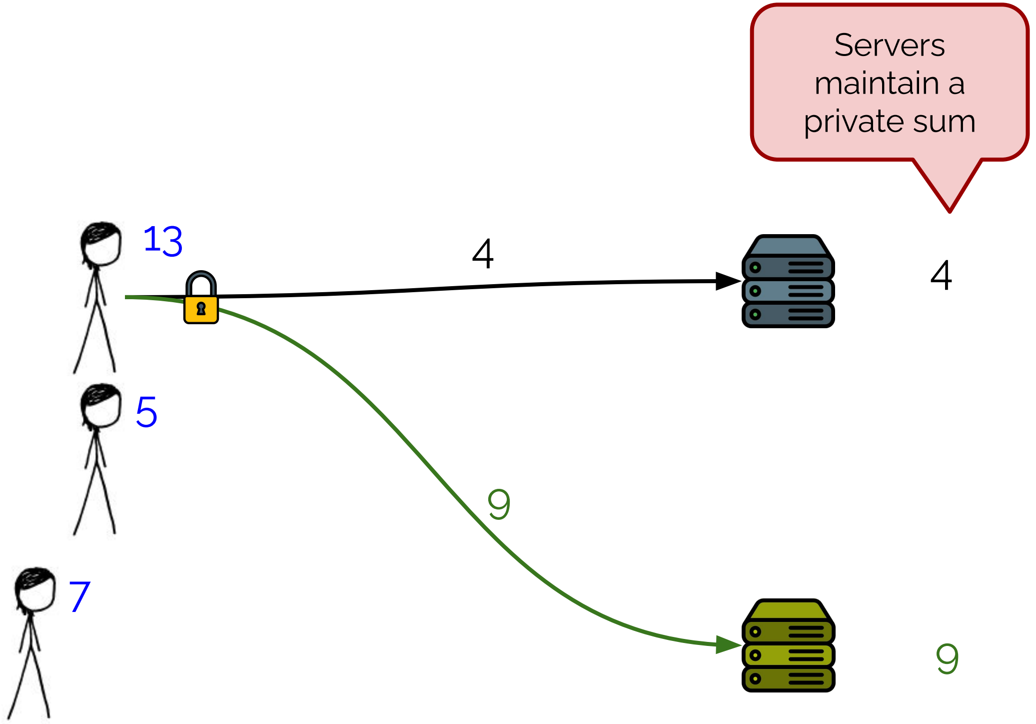


# MPC

Three users want to compute the **sum** of their **private inputs**.

Each user **secret shares** their input into random looking numbers.

(E.g.,: 4 and 9 reveal nothing about 13)

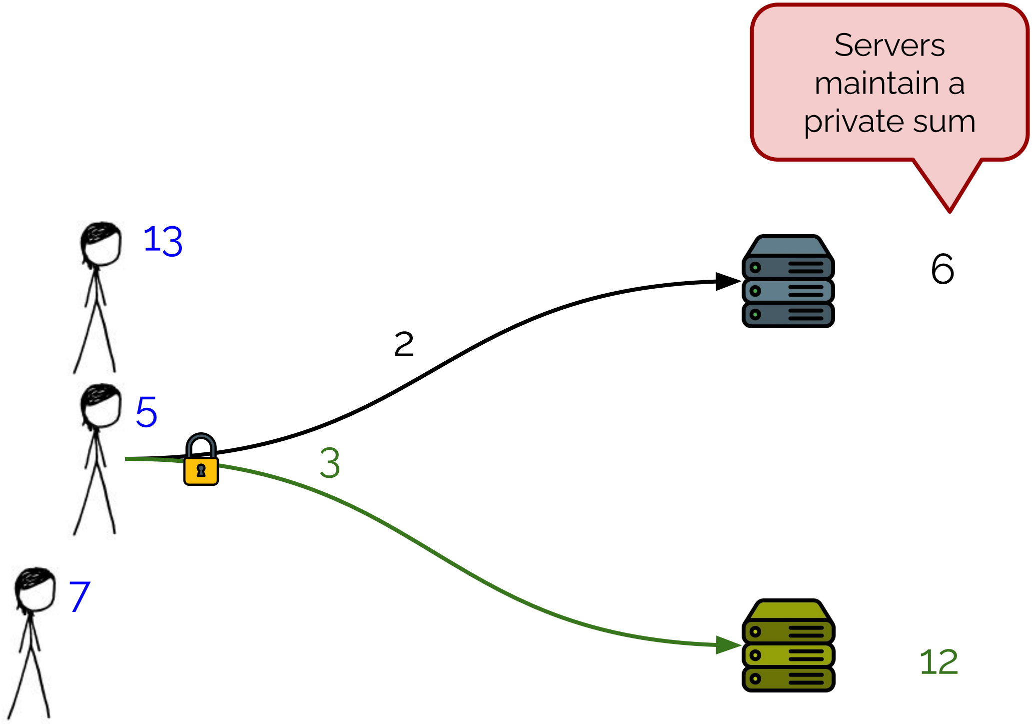


# MPC

Three users want to compute the **sum** of their **private inputs**.

Each user **secret shares** their input into random looking numbers.

(E.g.,: 4 and 9 reveal nothing about 13)

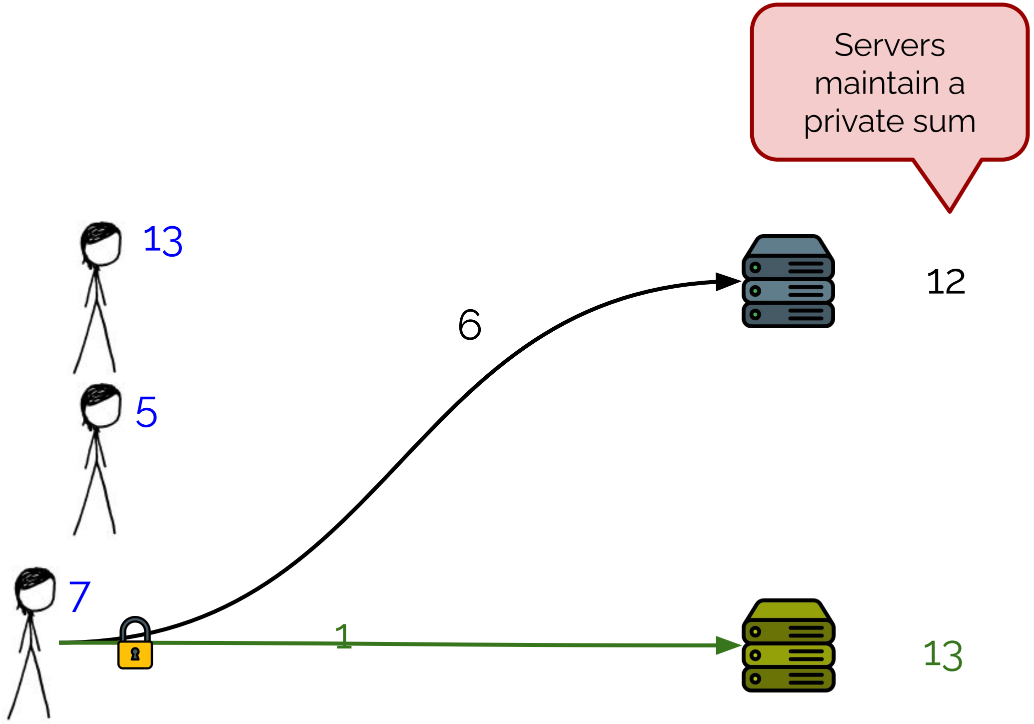


# MPC

Three users want to compute the **sum** of their **private inputs**.

Each user **secret shares** their input into random looking numbers.

(E.g.,: 4 and 9 reveal nothing about 13)

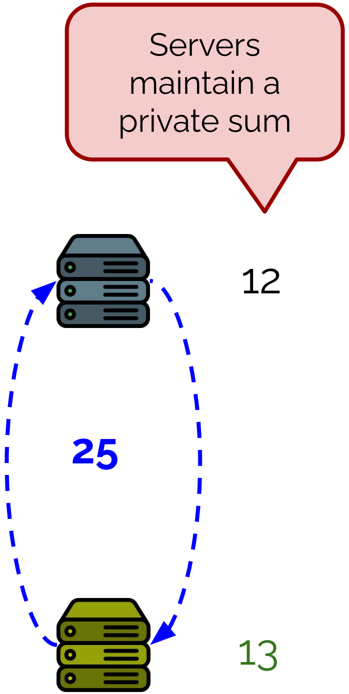


# MPC

Three users want to compute the **sum** of their **private inputs**.

Each user **secret shares** their input into random looking numbers.

(E.g.,: 4 and 9 reveal nothing about 13)

# MPC

Three users want to compute the **sum** of their **private inputs**.

Each user **secret shares** their input into random looking numbers.

(E.g.,: 4 and 9 reveal nothing about 13)

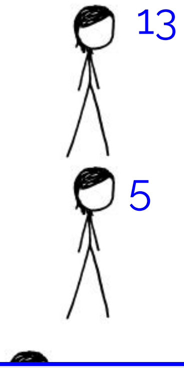

***Multiplication** can be computed similarly!*

# MPC for Machine Learning

Three users want to compute the **sum** of their **private inputs**.

Each user **secret shares** their input into random looking numbers.

(E.g.,: 4 and 9 reveal nothing about 13)




***Multiplication** can be computed similarly!*

*Using **Addition** and **Multiplication** we can do ML inference!*

# MPC for Machine Learning

Three users want to compute the **sum** of their **private inputs**.

Each user **secret shares** their input into random looking numbers.

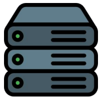
(E.g.,: 4 and 9 reveal nothing about 13)

13

5

***Multiplication** can be computed similarly!*

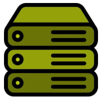
*Using **Addition** and **Multiplication** we can do ML inference!*



**Server 1**


Private model

Private inputs



**Server 2**

# MPC for Machine Learning

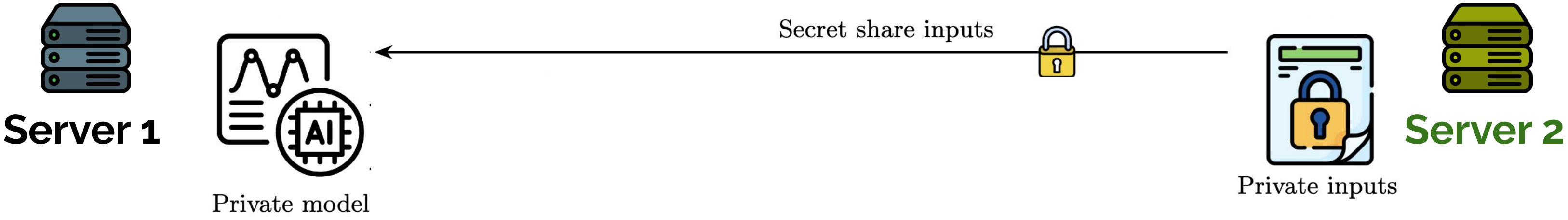
Three users want to compute the **sum** of their **private inputs**.

Each user **secret shares** their input into random looking numbers.

(E.g.,: 4 and 9 reveal nothing about 13)

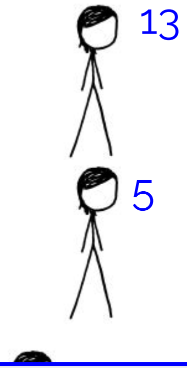

***Multiplication** can be computed similarly!*

*Using **Addition** and **Multiplication** we can do ML inference!*



# MPC for Machine Learning

Three users want to compute the **sum** of their **private inputs**.

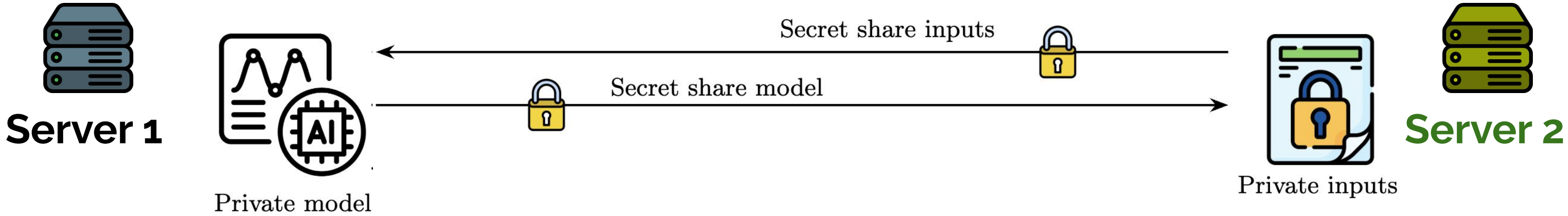



Each user **secret shares** their input into random looking numbers.

(E.g.,: 4 and 9 reveal nothing about 13)

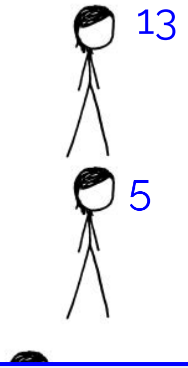
*__Multiplication__ can be computed similarly!*

*Using __Addition__ and __Multiplication__ we can do ML inference!*



# MPC for Machine Learning

Three users want to compute the **sum** of their **private inputs**.

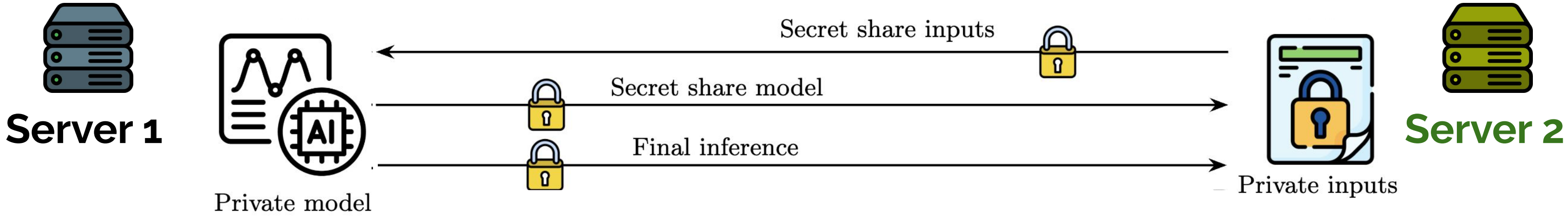



Each user **secret shares** their input into random looking numbers.

(E.g.,: 4 and 9 reveal nothing about 13)

*__Multiplication__ can be computed similarly!*

*Using __Addition__ and __Multiplication__ we can do ML inference!*



# MPC for Machine Learning

Three users want to compute the **sum** of their **private inputs**.

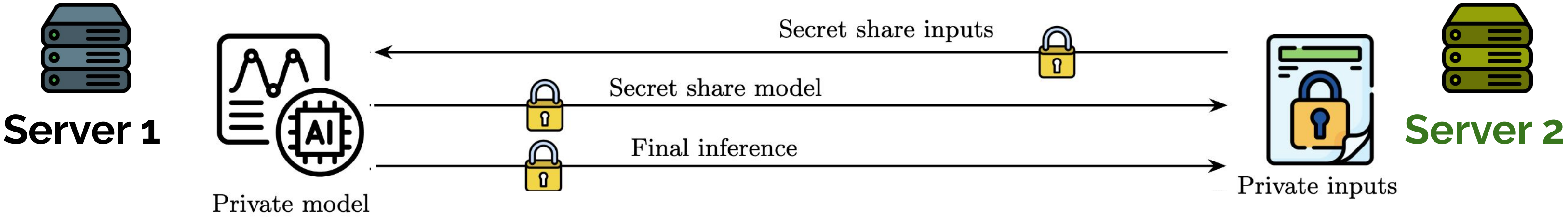
Each user **secret shares** their input into random looking numbers.

(E.g.,: 4 and 9 reveal nothing about 13)

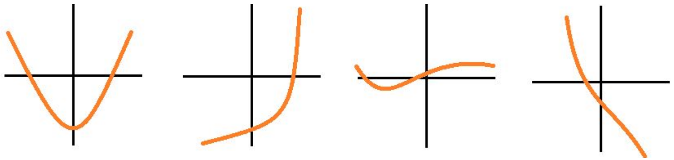

*__Multiplication__ can be computed similarly!*

*Using __Addition__ and __Multiplication__ we can do ML inference!*

**Almost**

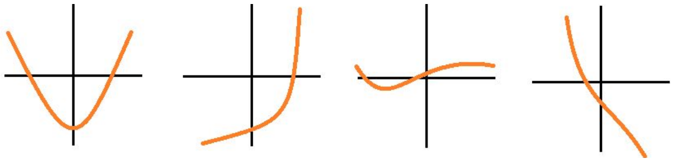


# Non-Linear Functions in MPC

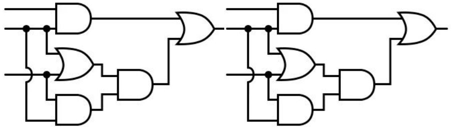


MPC protocols cannot evaluate **non-linearities** directly!

# Non-Linear Functions in MPC

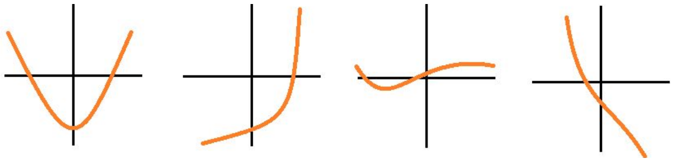


MPC protocols cannot evaluate **non-linearities** directly!

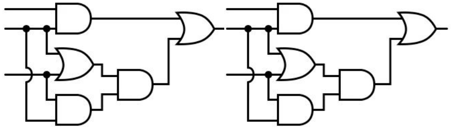


→**Boolean (aka garbled) circuits** can be used but are big and **expensive**.

# Non-Linear Functions in MPC



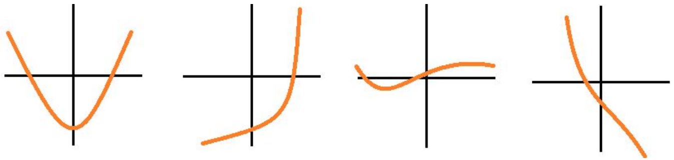
MPC protocols cannot evaluate **non-linearities** directly!



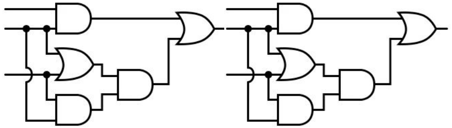
→**Boolean (aka garbled) circuits** can be used but are big and **expensive**.

→**Polynomial Approximations** can be used but are **slow** (**high communication**) and introduce big approximation **errors**.

# Non-Linear Functions in MPC



MPC protocols cannot evaluate **non-linearities** directly!



→**Boolean (aka garbled) circuits** can be used but are big and **expensive**.

→**Polynomial Approximations** can be used but are **slow** (**high communication**) and introduce big approximation **errors**.

SOTA MPC protocols evaluate non-linearities as lookup tables (LUTs), but

LUTs **scale poorly** for high precision → **very high communication**

# The **Curl** Framework

- Construct smaller LUTs **without sacrificing accuracy**
    - Using Discrete Wavelet Transforms (DWT)

# The **Curl** Framework

Curl

- Construct smaller LUTs **without sacrificing accuracy**
    - Using Discrete Wavelet Transforms (DWT)

- MPC-tailored protocols for evaluating DWT LUTs:
    - **Haar DWT:** faster, higher errors
    - **Biorthogonal DWT:** slower, lower errors

# The **Curl** Framework

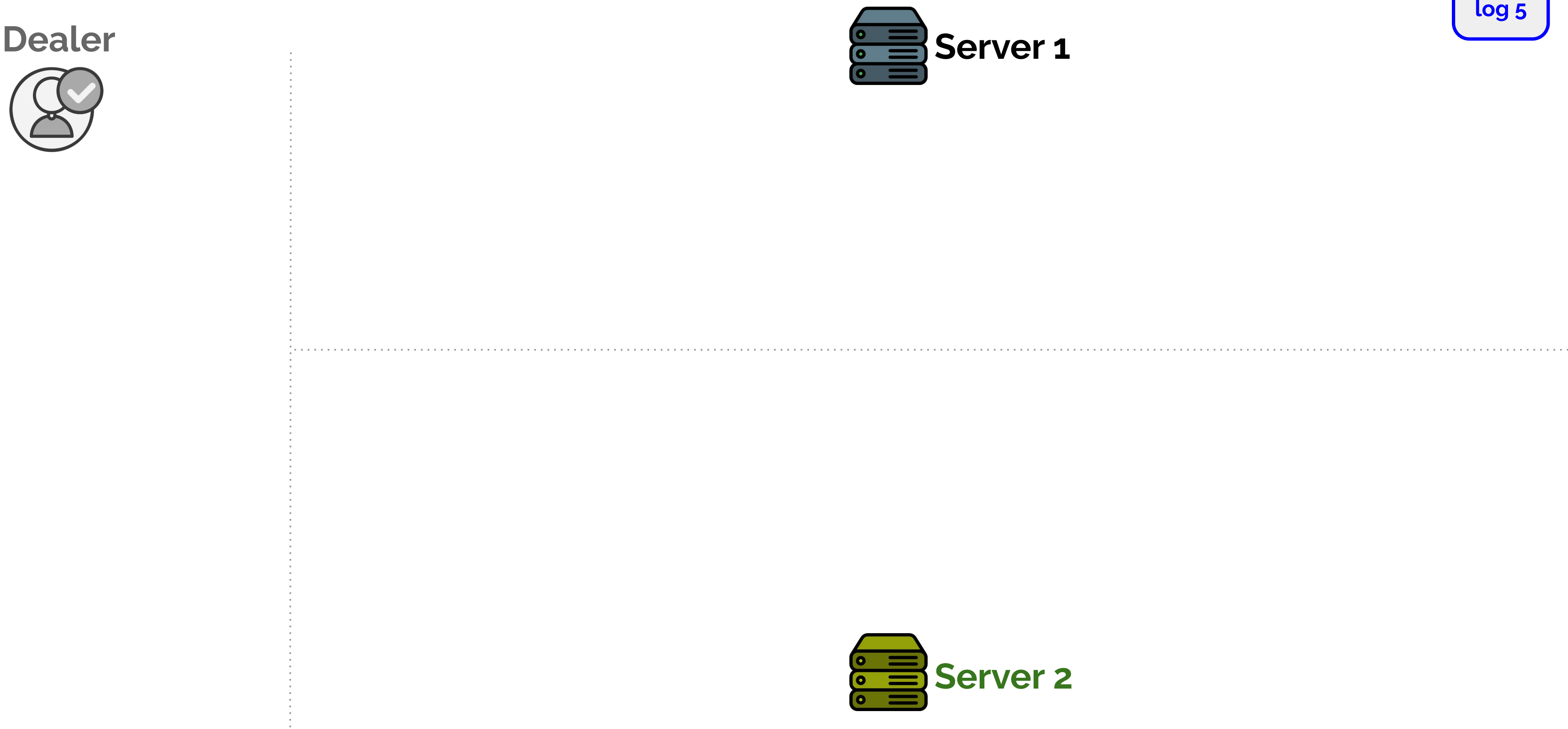
- Construct smaller LUTs **without sacrificing accuracy**
    - Using Discrete Wavelet Transforms (DWT)

- MPC-tailored protocols for evaluating DWT LUTs:
    - **Haar DWT:** faster, higher errors
    - **Biorthogonal DWT:** slower, lower errors

- Experiments over a suite of commonly used non-linear functions + LLMs.

# Secure Look-Up Table

log 1
log 2
log 3
log 4
log 5



# Secure Look-Up Table

Public LUT for log

| |
|---|
| 0 |
| 1 |
| 1.6 |
| 2 |
| 2.3 |

Dealer

Server 1

Server 2

# Secure Look-Up Table

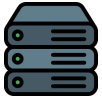
Secret Input

$x = 4$

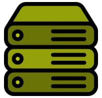
Public LUT for log

| |
|---|
| 0 |
| 1 |
| 1.6 |
| 2 |
| 2.3 |

Dealer



Server 1



Server 2

# Secure Look-Up Table

Secret Input

x = 4

Public LUT for log

| |
|---|
| 0 |
| 1 |
| 1.6 |
| 2 |
| 2.3 |

**Dealer**

**Server 1**

**Server 2**

# Secure Look-Up Table

Secret Input
x = 4

Public LUT for log

| |
|---|
| 0 |
| 1 |
| 1.6 |
| 2 |
| 2.3 |

**Dealer**

Input
[x] = 3
**Server 1**

Input
[x] = 1
**Server 2**

# Secure Look-Up Table

Secret Input
x = 4

Public LUT for log

| |
|---|
| 0 |
| 1 |
| 1.6 |
| 2 |
| 2.3 |

Dealer

Random
r ← 2

Input
[x] = 3

Server 1

Input
[x] = 1

Server 2

# Secure Look-Up Table

# Secure Look-Up Table

Secret Input
x = 4

Public LUT for log

| |
|---|
| 0 |
| 1 |
| 1.6 |
| 2 |
| 2.3 |

**Dealer**

Random
r ← 2

1-hot vector encoding **r**

| |
|---|
| 0 |
| **1** |
| 0 |
| 0 |
| 0 |

**Server 1**

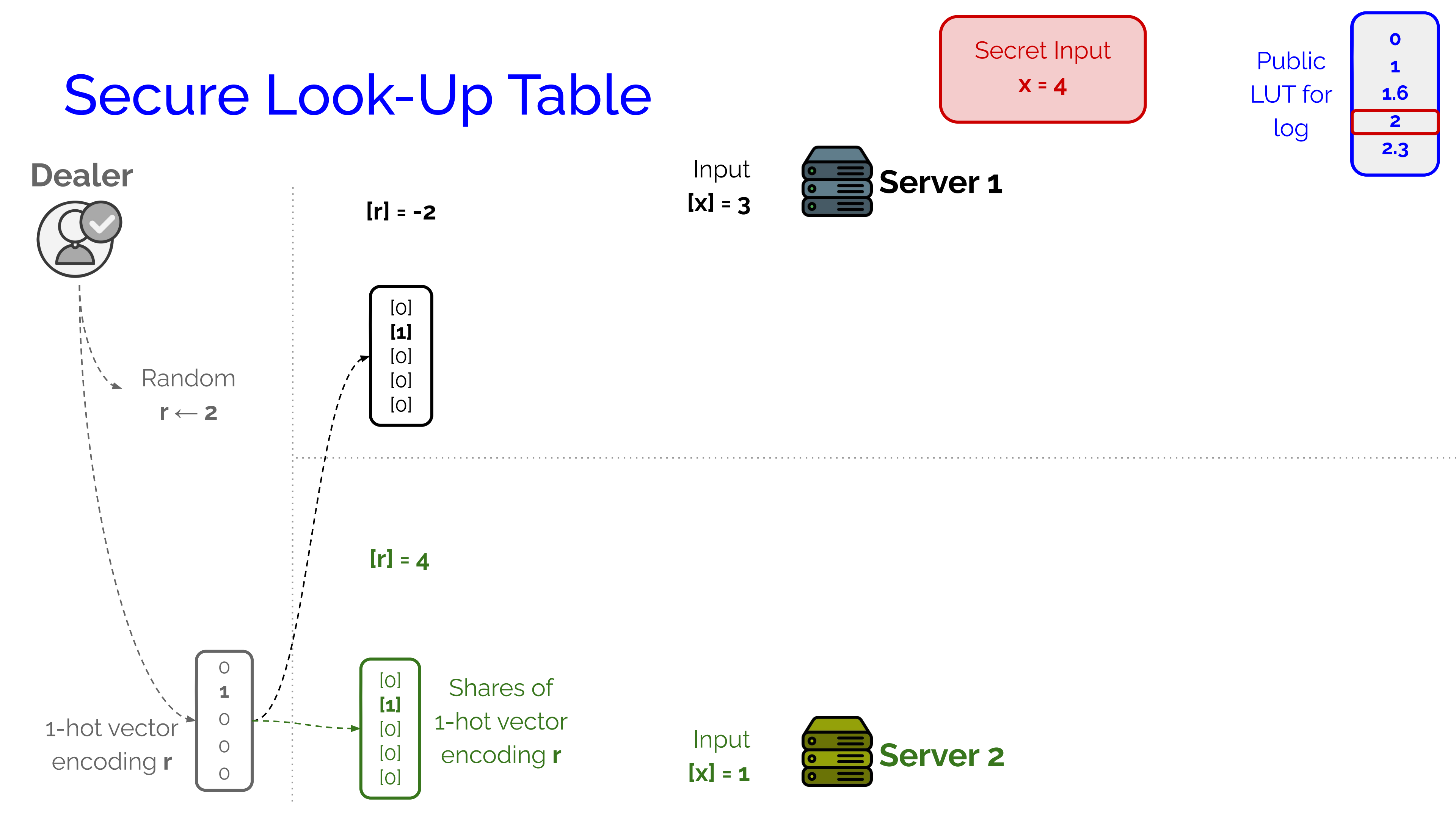
Input
[x] = 3

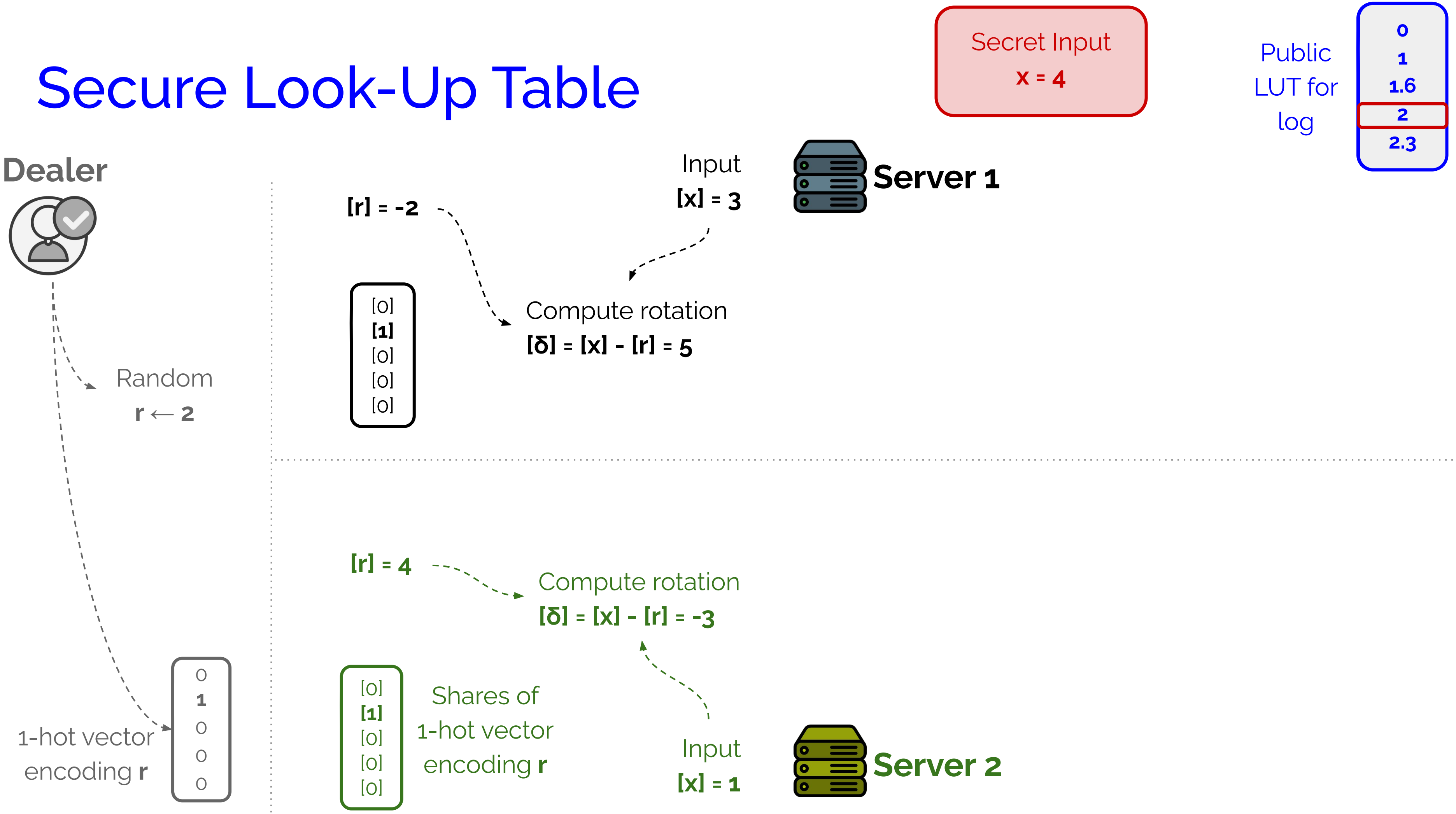
[r] = -2

**Server 2**

Input
[x] = 1

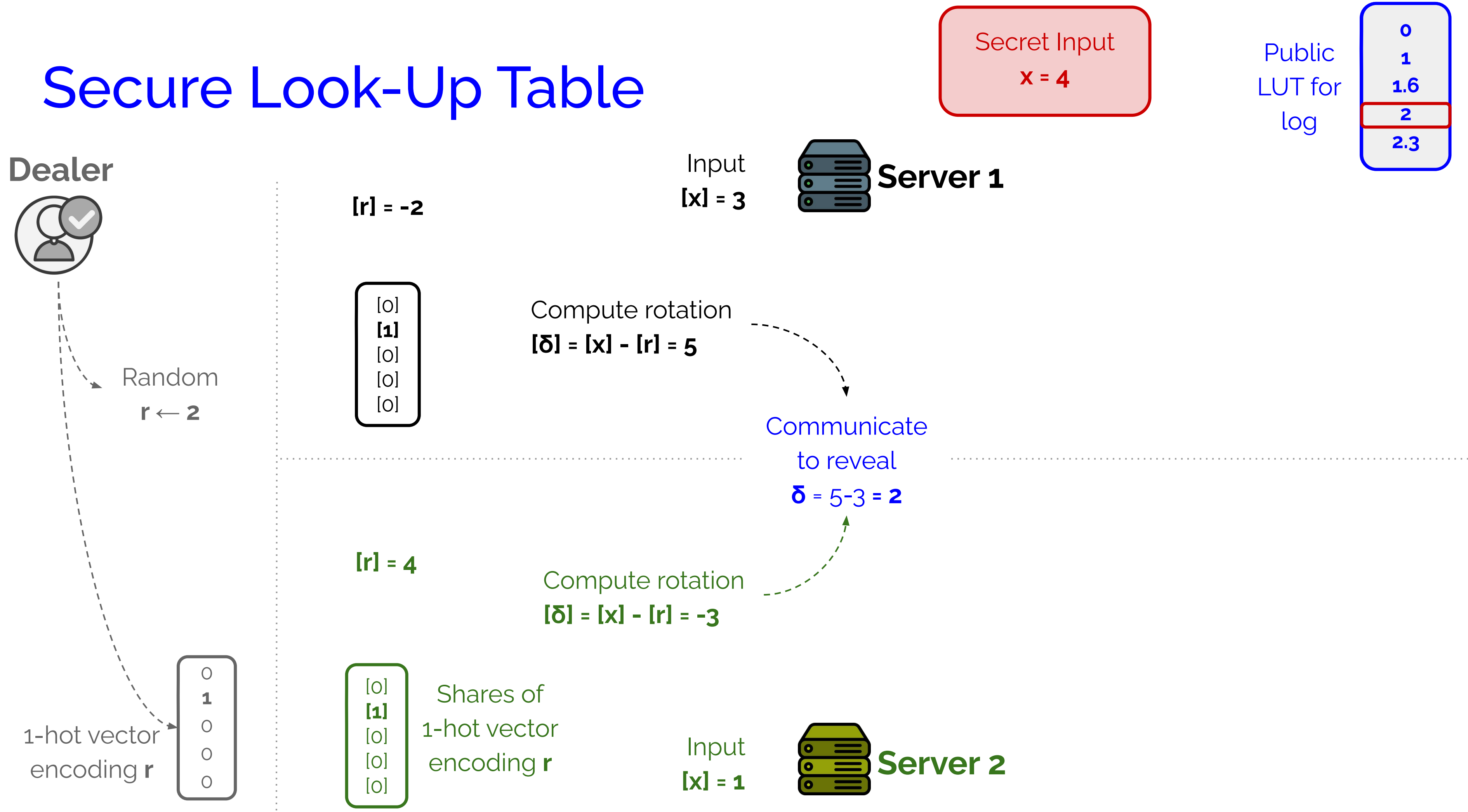
[r] = 4


Secure Look-Up Table
Secret Input
x = 4
Public
LUT for
log
0
1
1.6
2
2.3
Dealer
Random
r ← 2
1-hot vector
encoding r
0
1
0
0
0
[r] = -2
[0]
[1]
[0]
[0]
[0]
Input
[x] = 3
Server 1
[r] = 4
[0]
[1]
[0]
[0]
[0]
Shares of
1-hot vector
encoding r
Input
[x] = 1
Server 2

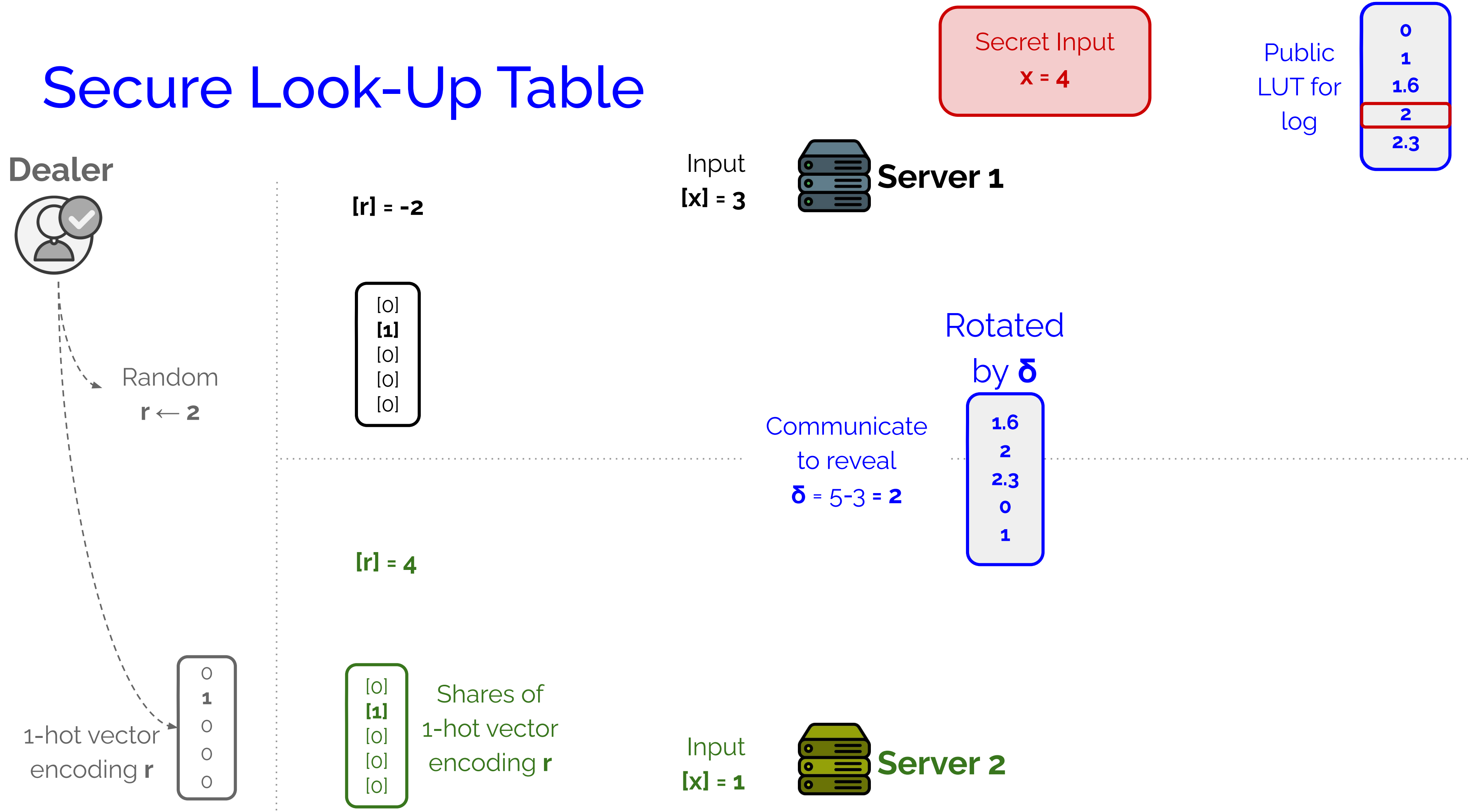


Secure Look-Up Table
Secret Input
x = 4
Public
LUT for
log
0
1
1.6
2
2.3
Dealer
Random
r ← 2
1-hot vector
encoding r
0
1
0
0
0
Server 1
Input
[x] = 3
[r] = -2
Compute rotation
[δ] = [x] - [r] = 5
[0]
[1]
[0]
[0]
[0]
[r] = 4
Compute rotation
[δ] = [x] - [r] = -3
[0]
[1]
[0]
[0]
[0]
Shares of
1-hot vector
encoding r
Input
[x] = 1
Server 2

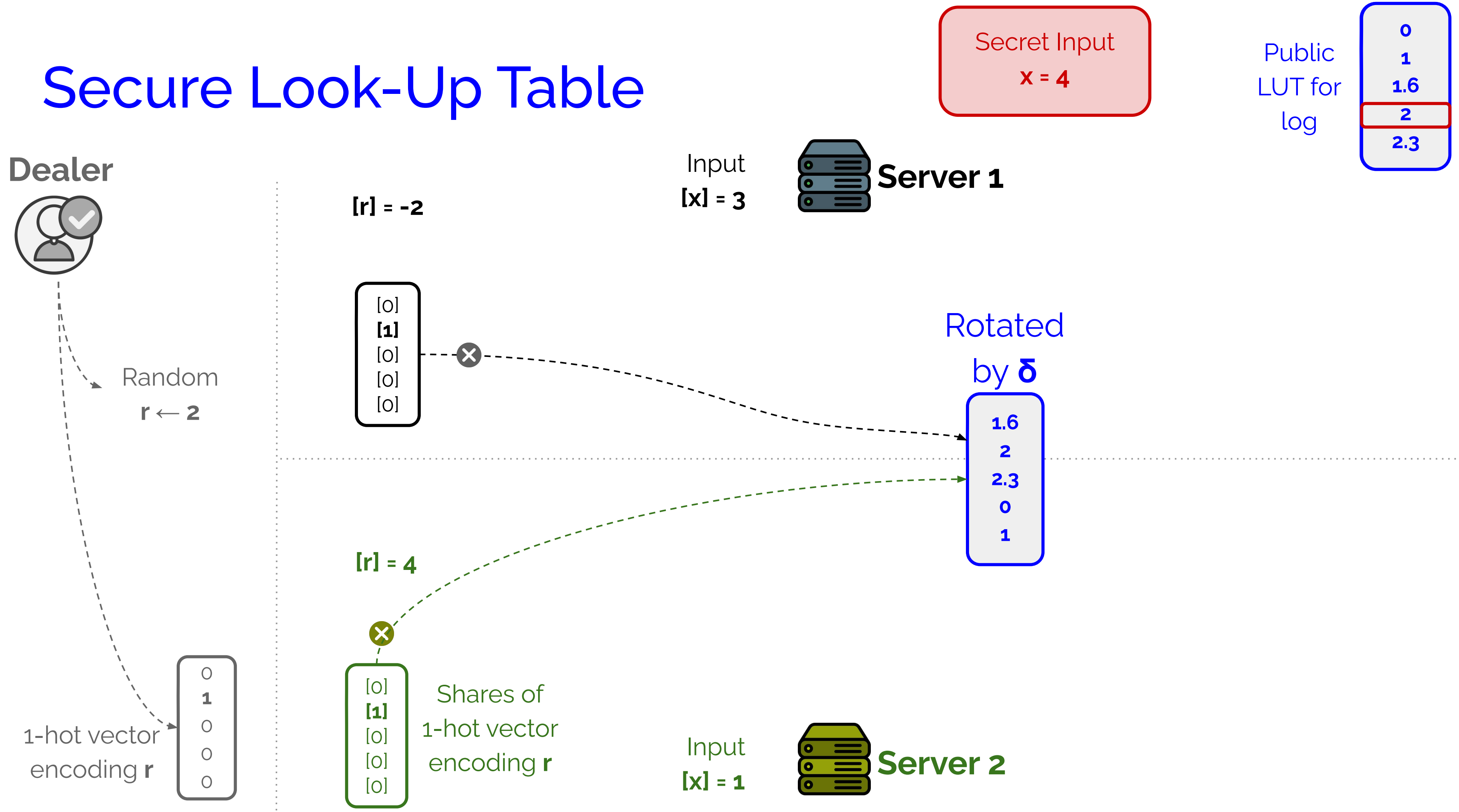
# Secure Look-Up Table



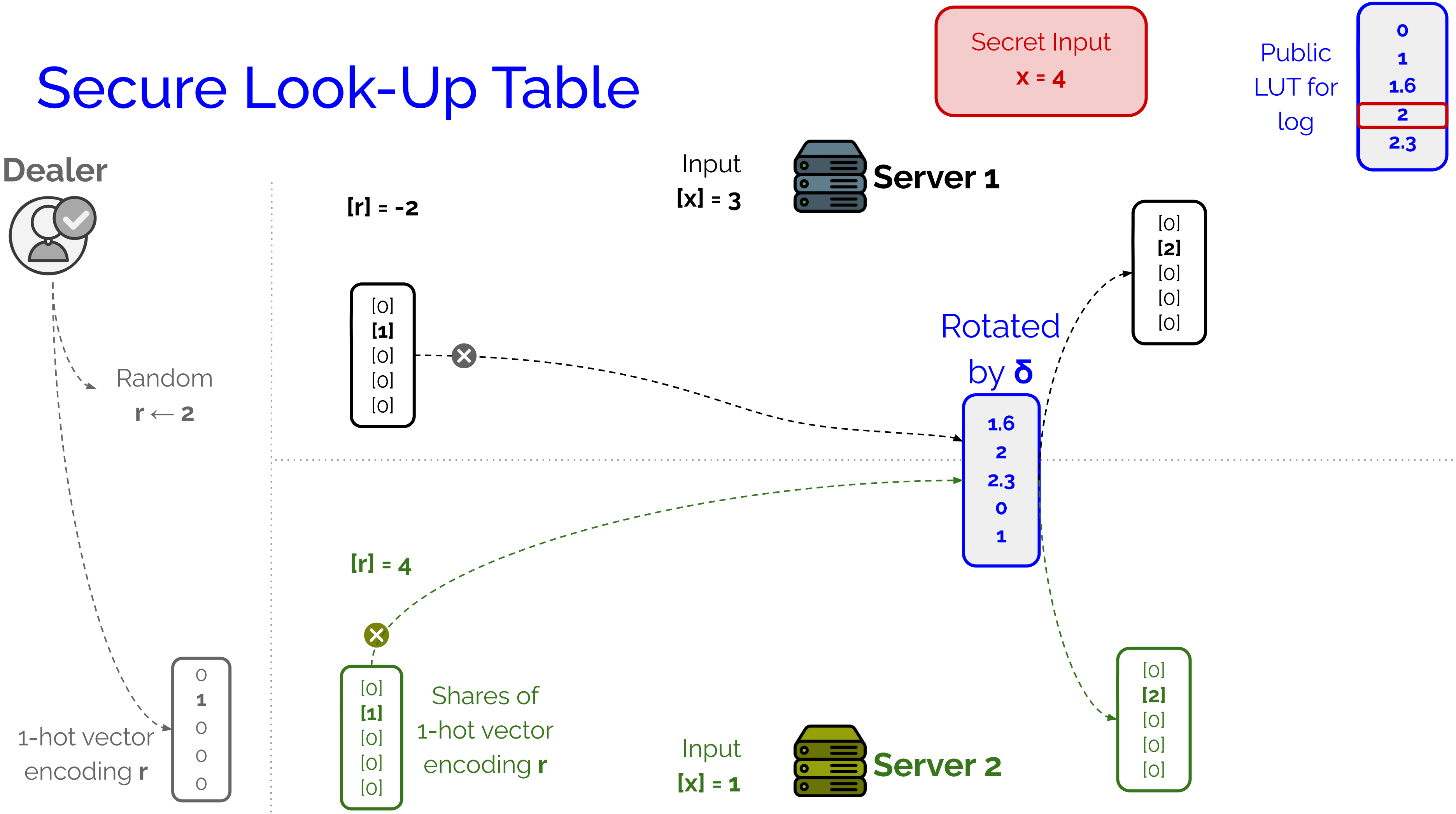
# Secure Look-Up Table



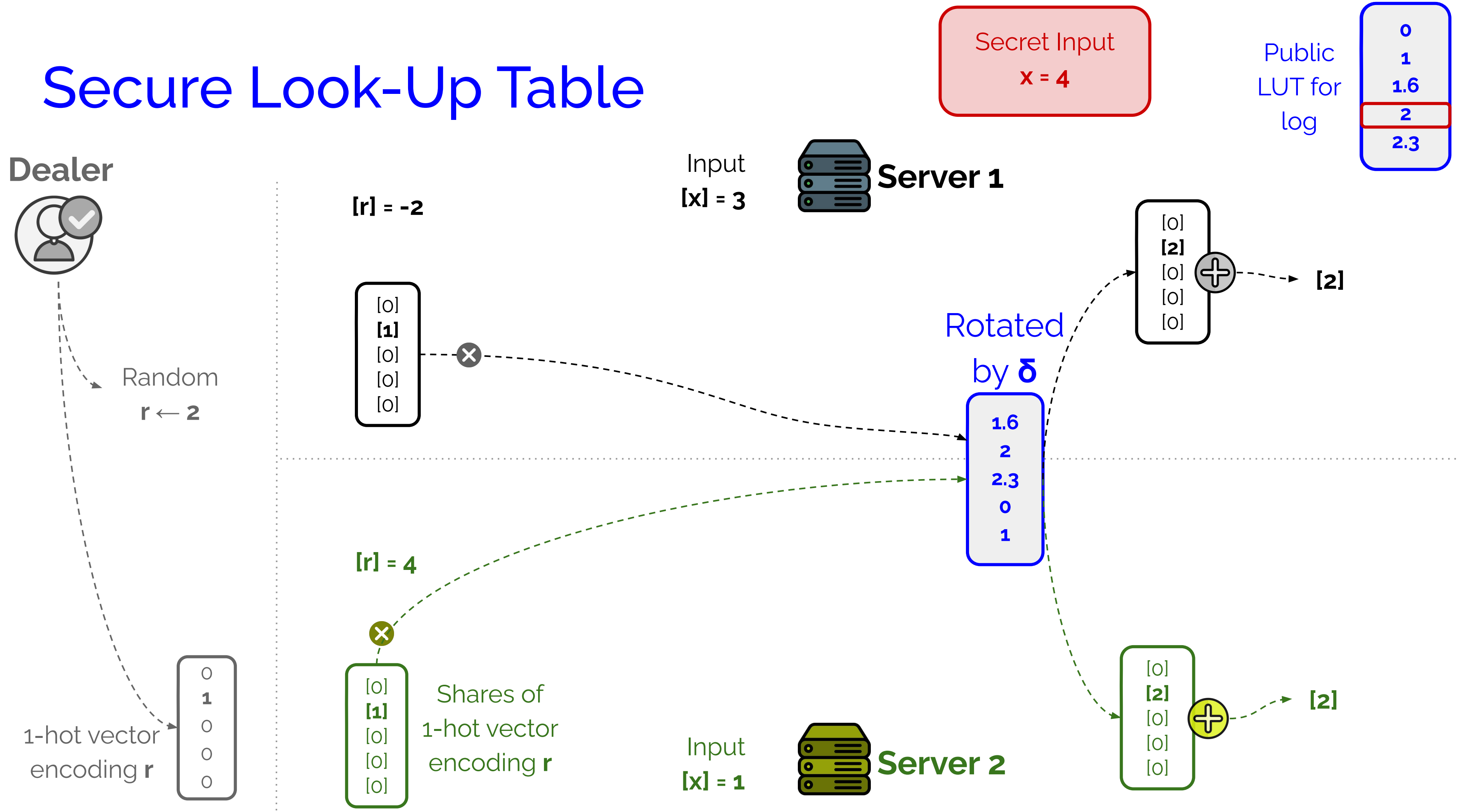
# Secure Look-Up Table



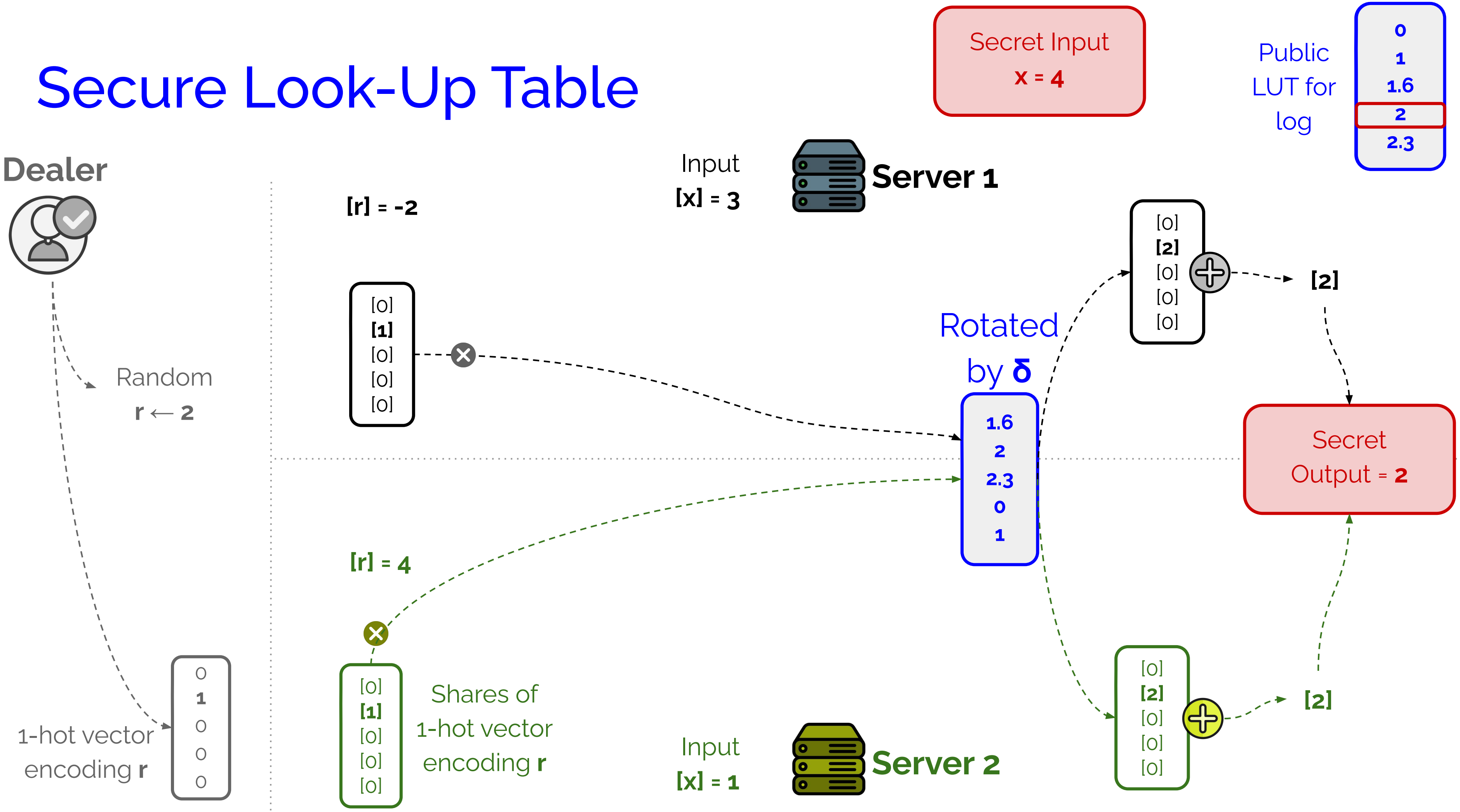
# Secure Look-Up Table



# Secure Look-Up Table

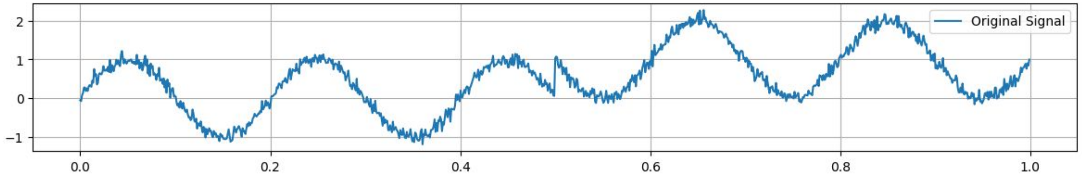


# Secure Look-Up Table



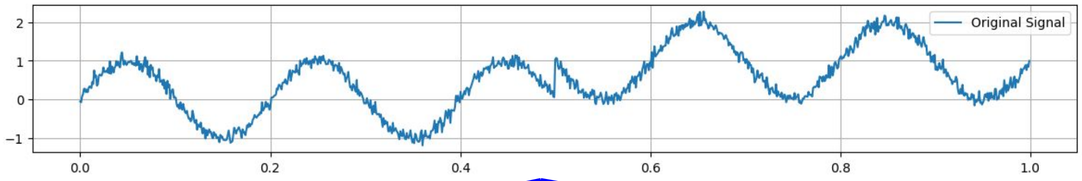
# Discrete Wavelet Transform (DWT)

Initial signal **s**



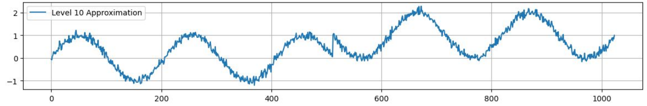
# Discrete Wavelet Transform (DWT)

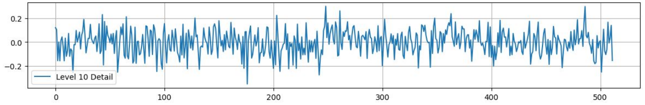
Initial signal **s**




**Approximations** $\boldsymbol{a}$

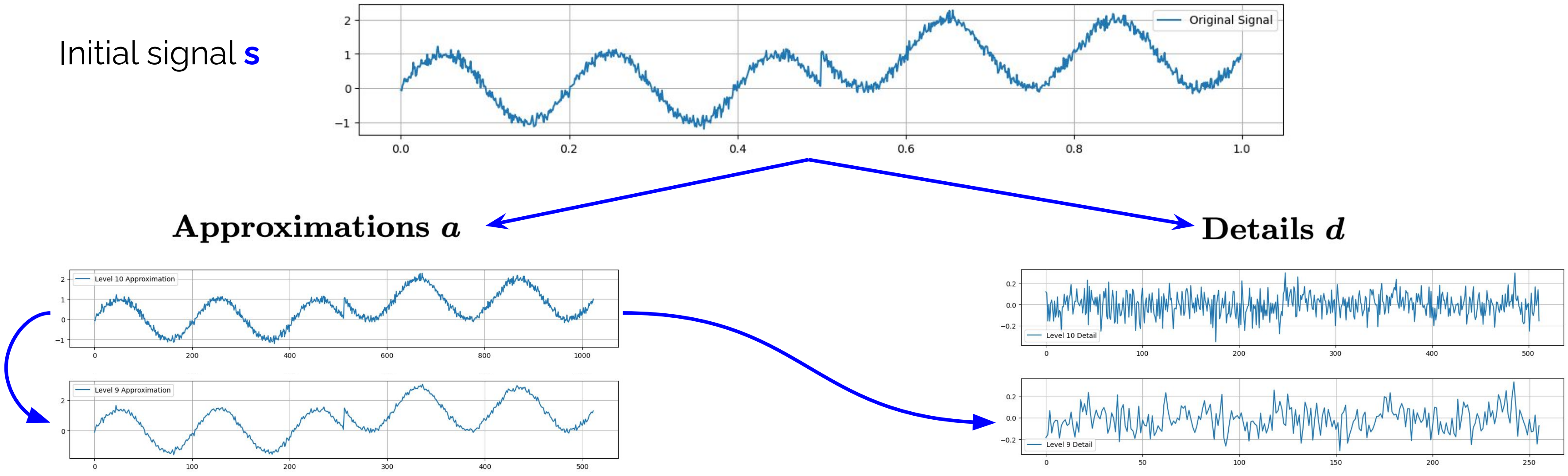
**Details** $\boldsymbol{d}$



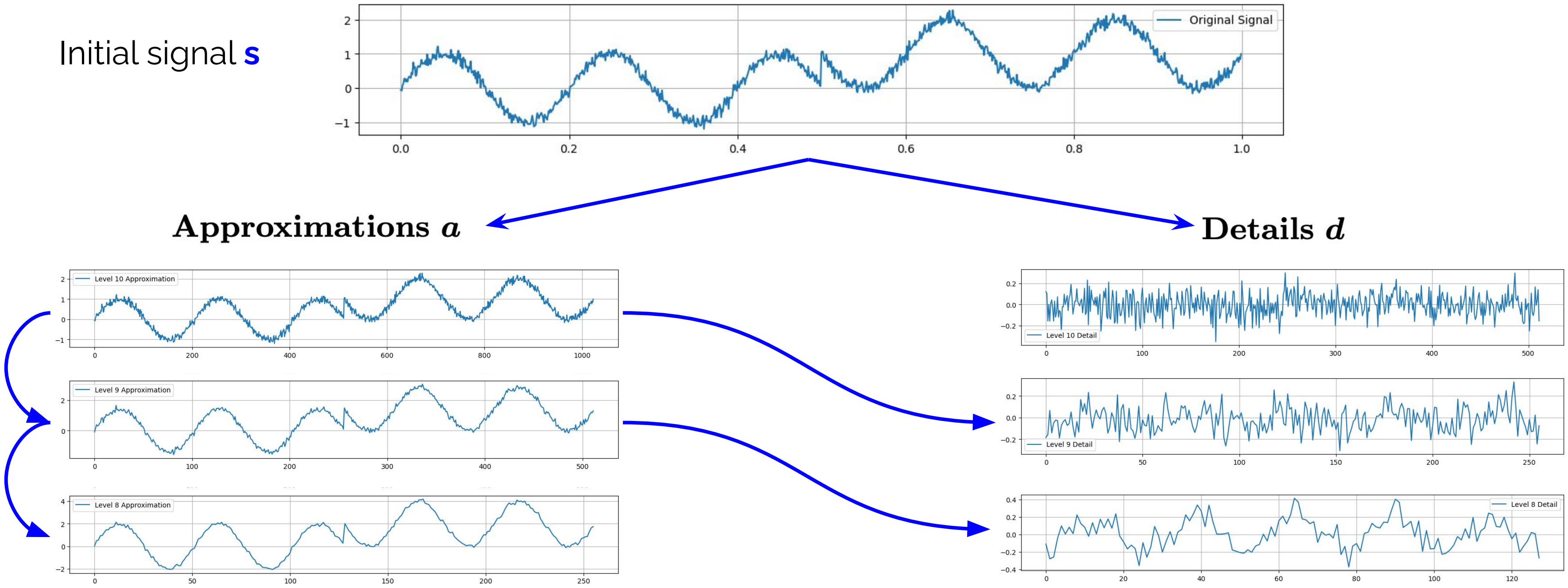
# Discrete Wavelet Transform (DWT)

Initial signal **s**

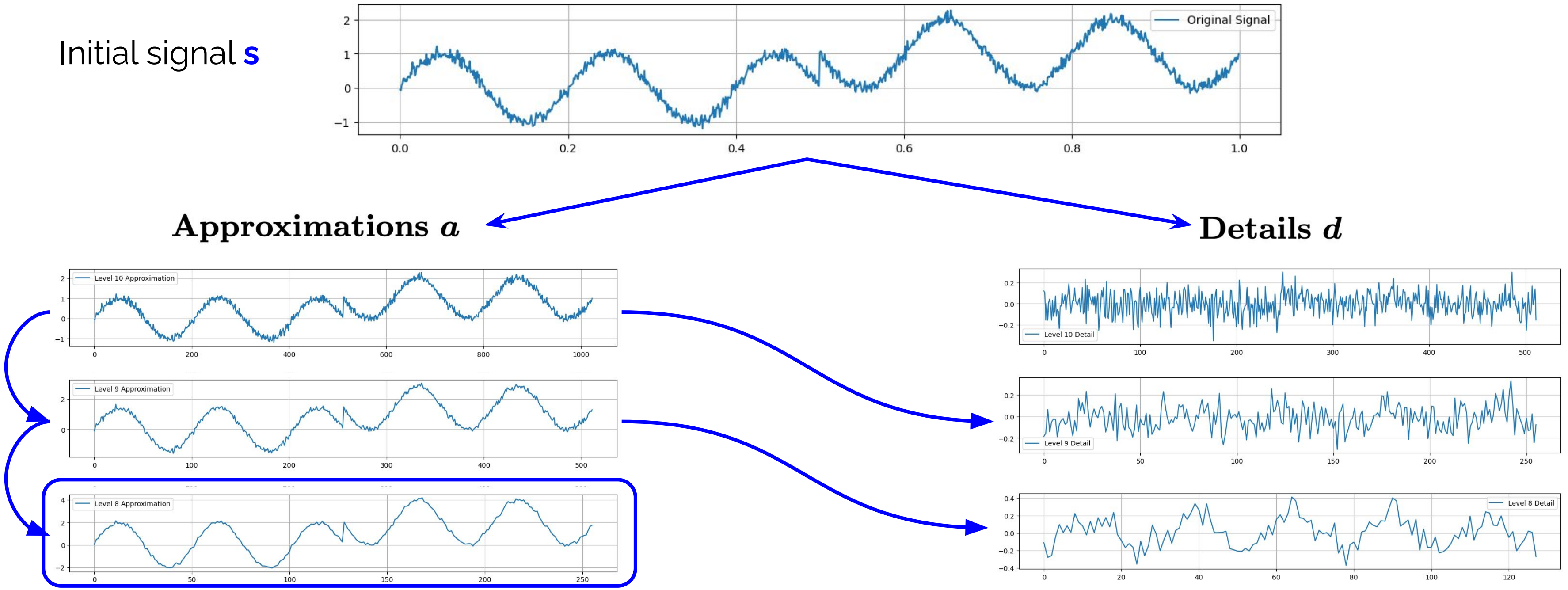


# Discrete Wavelet Transform (DWT)

Initial signal **s**

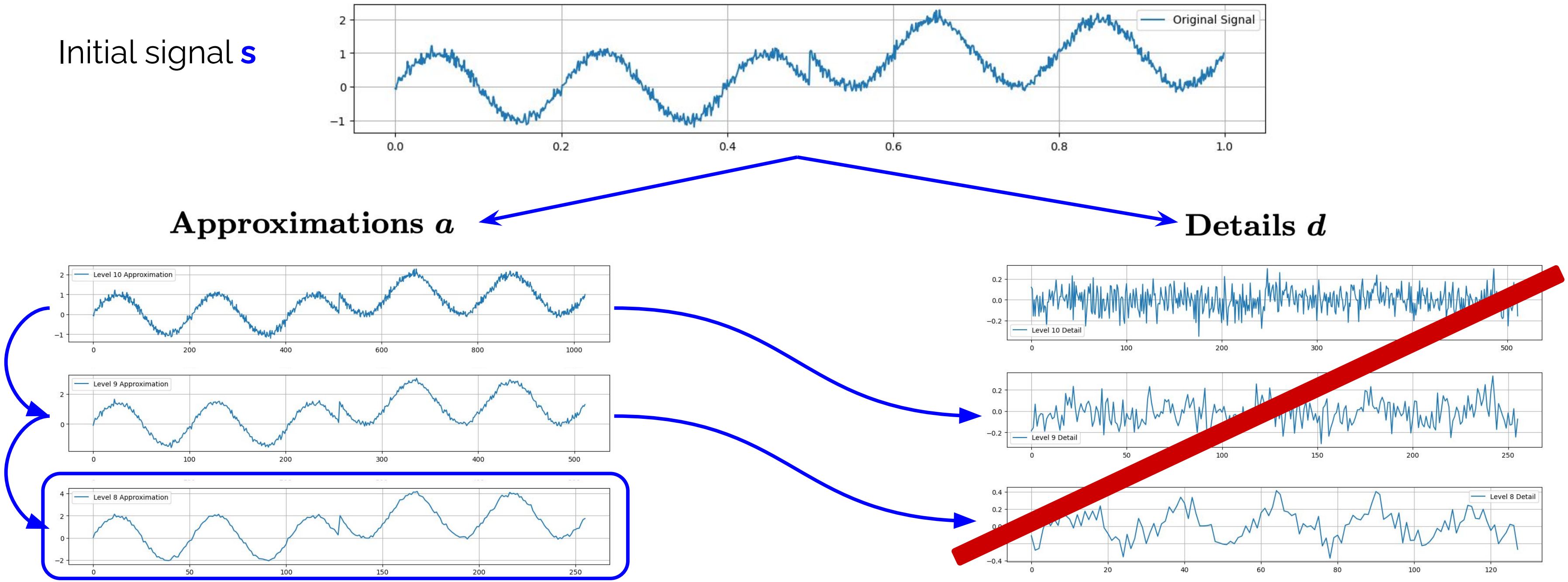


# Discrete Wavelet Transform (DWT)




Smooth part of **s** remains unchanged!

# Discrete Wavelet Transform (DWT)

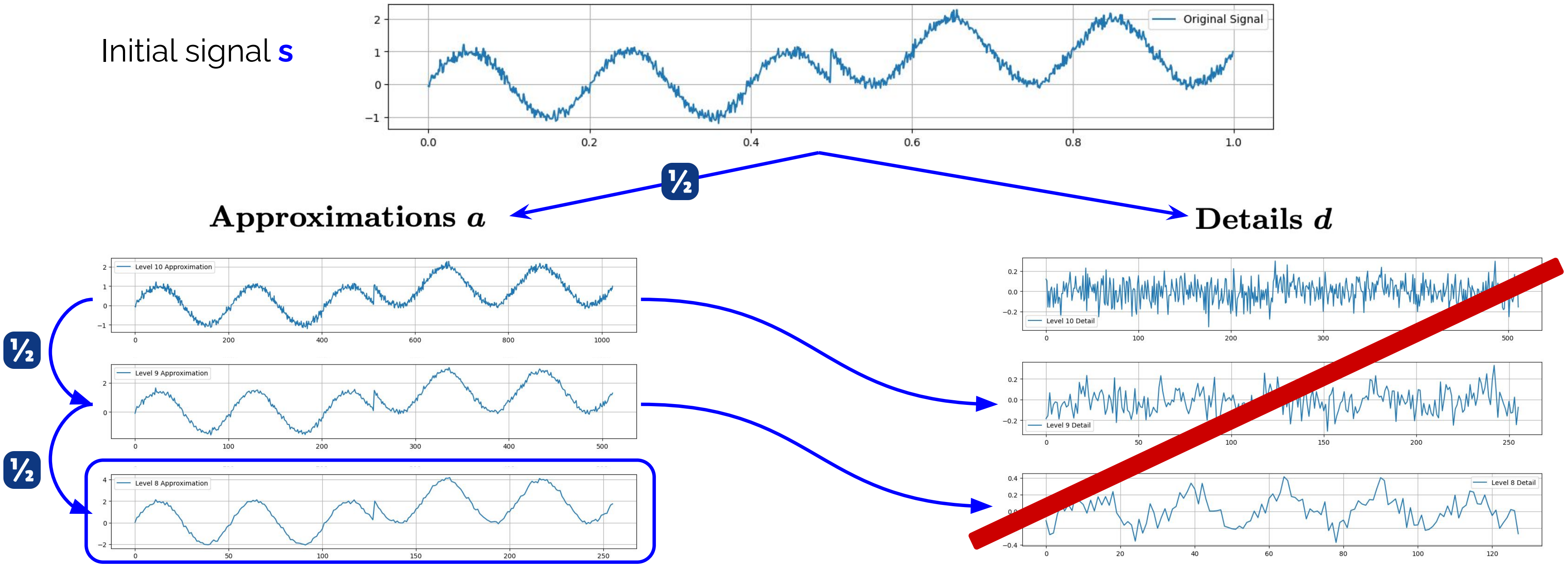



Smooth part of **s** remains unchanged!

**Details can be set to zero!**

# Discrete Wavelet Transform (DWT)

Initial signal **s**




Smooth part of **s** remains unchanged!

**Details can be set to zero!**

# Approximation Strategies

**Goal:** Evaluate  y = LUT(x) for W bits (e.g. 32)

# Approximation Strategies

Approximate

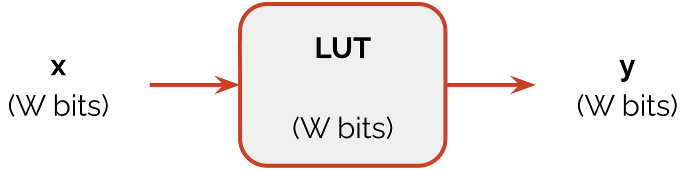
**Goal:** ~~Evaluate~~ y = LUT(x) for W bits (e.g. 32)

# Approximation Strategies

Approximate

**Goal:** ~~Evaluate~~ y = LUT(x) for W bits (e.g. 32)

## 0) Direct Evaluation



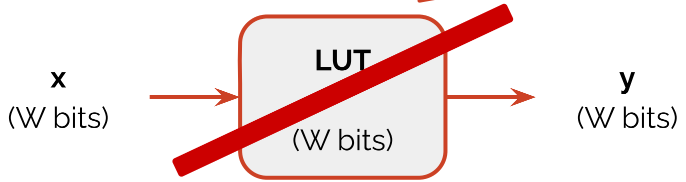
# Approximation Strategies

Approximate

**Goal:** ~~Evaluate~~ y = LUT(x) for W bits (e.g. 32)

High runtime ($2^{32}$)

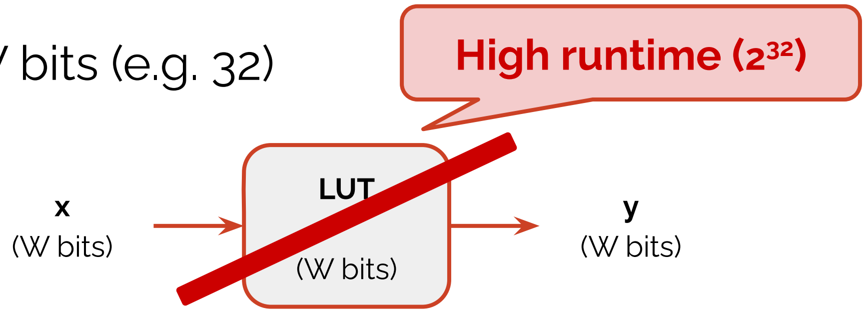
**0) Direct Evaluation**



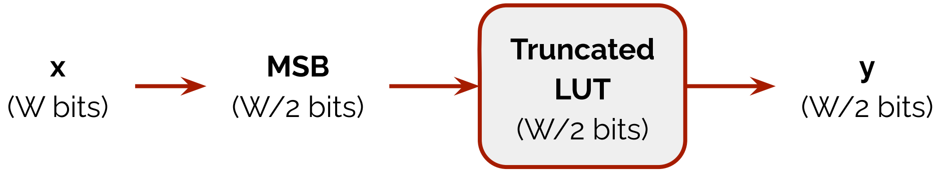
# Approximation Strategies

**Goal:** ~~Evaluate~~ Approximate y = LUT(x) for W bits (e.g. 32)

**0) Direct Evaluation**



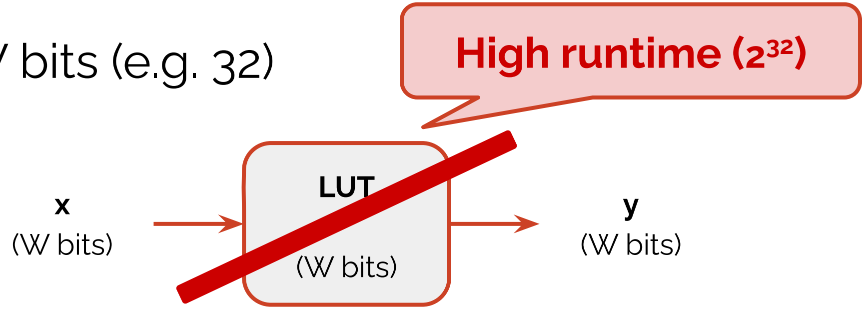

**1) Quantization/Truncation**



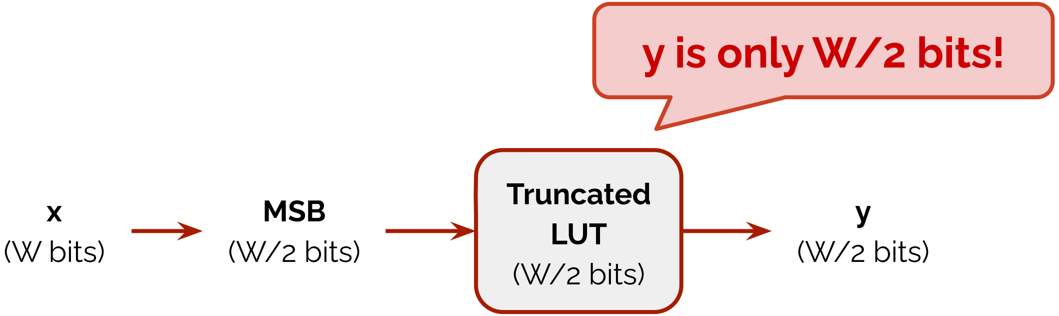
# Approximation Strategies

**Goal:** ~~Evaluate~~ Approximate y = LUT(x) for W bits (e.g. 32)

**0) Direct Evaluation**



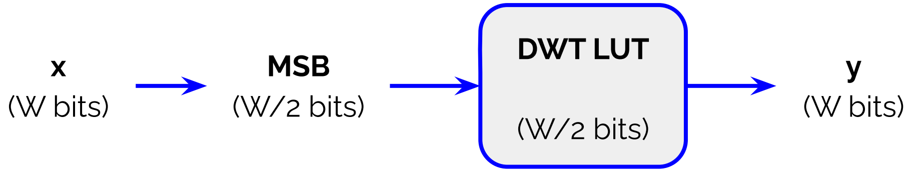

**1) Quantization/Truncation**



# Approximation Strategies

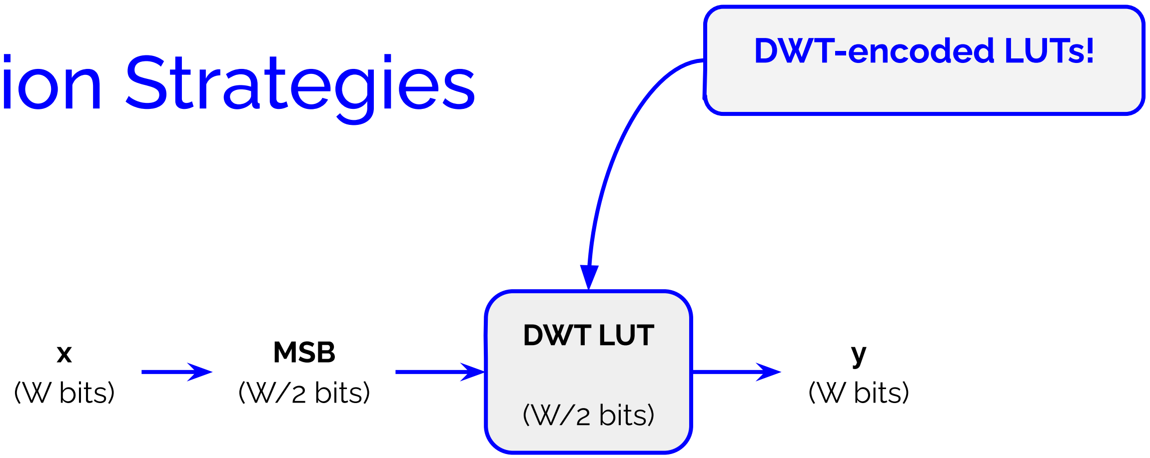
# Approximation Strategies

## 2) Haar DWT



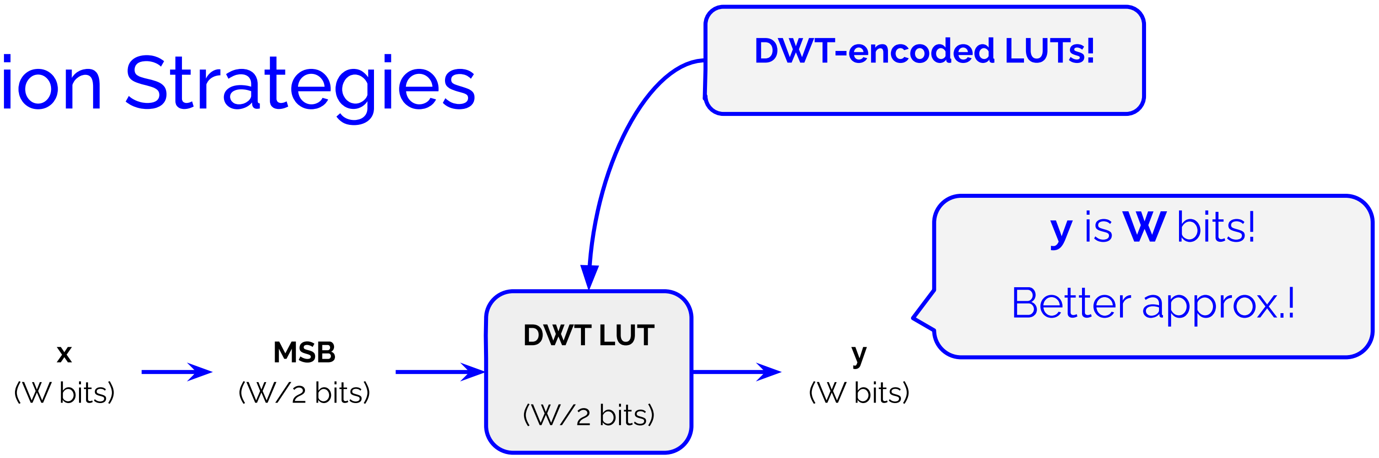
# Approximation Strategies

**2) Haar DWT**



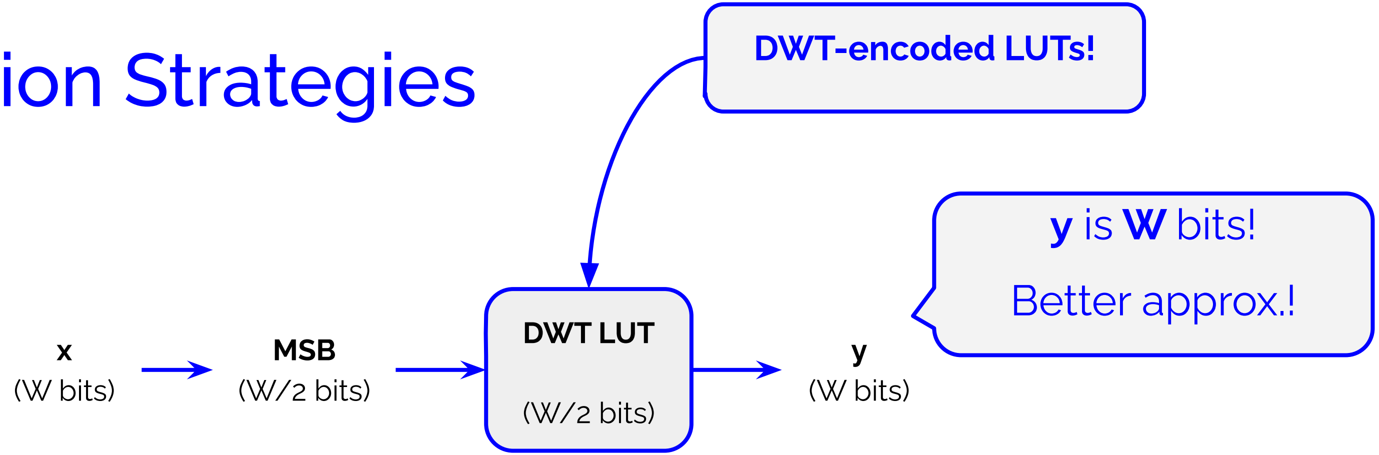
# Approximation Strategies

**2) Haar DWT**



# Approximation Strategies

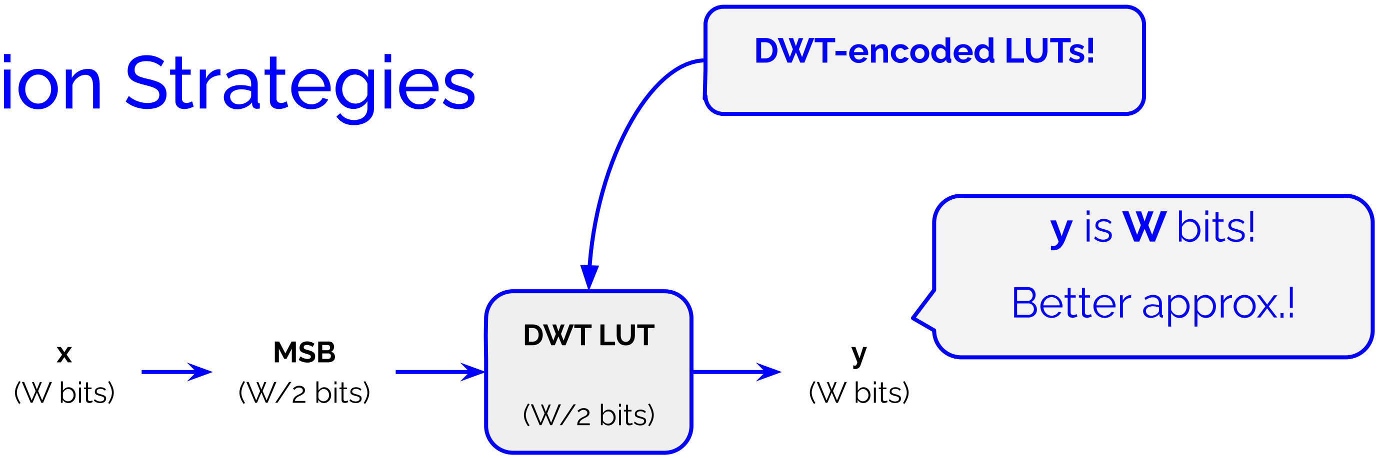
**2) Haar DWT**



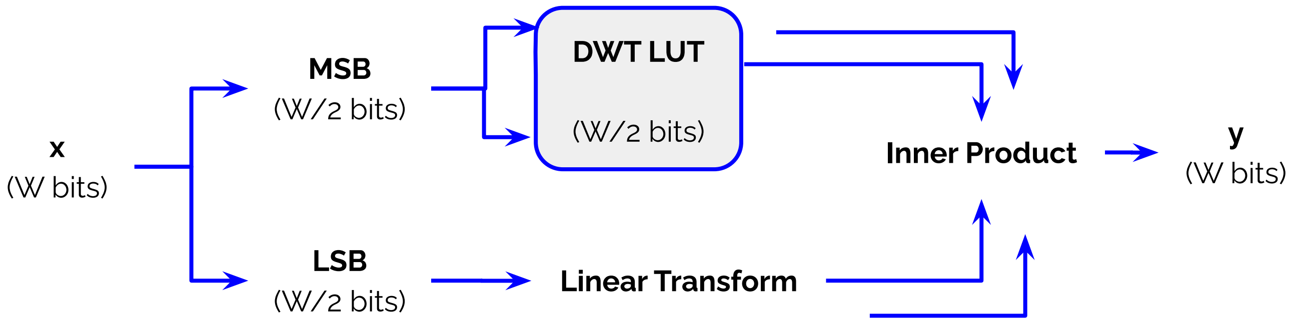

**3) Biorthogonal DWT**

# Approximation Strategies

## 2) Haar DWT

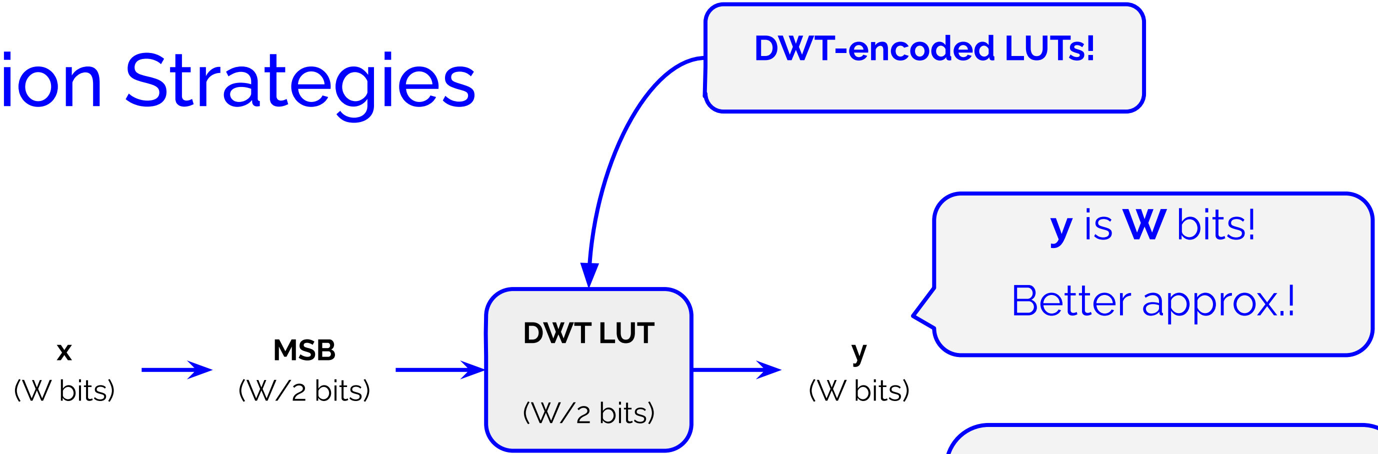



## 3) Biorthogonal DWT

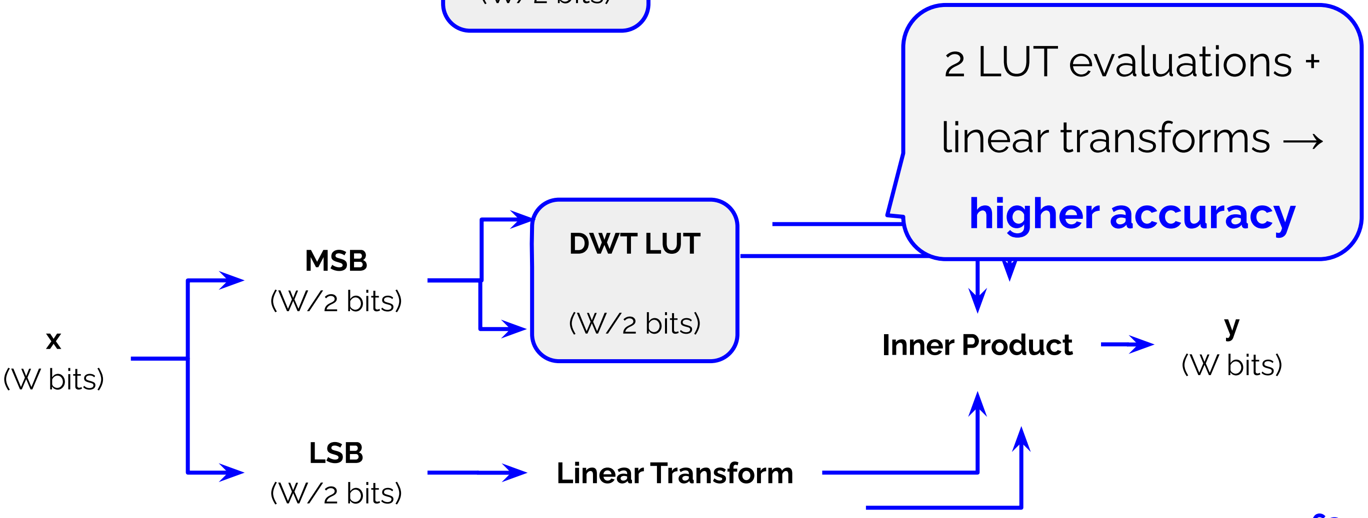


# Approximation Strategies

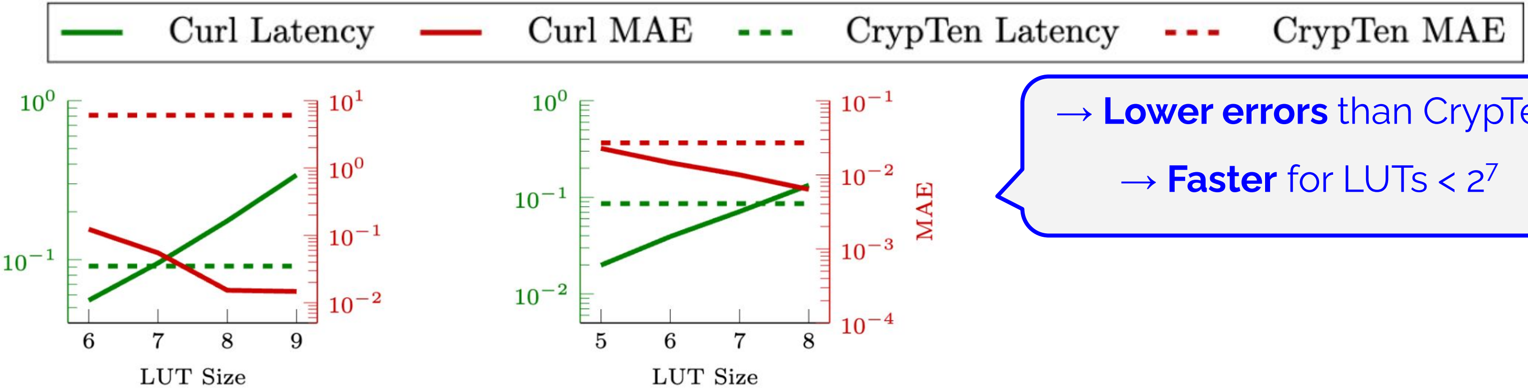
## 2) Haar DWT




## 3) Biorthogonal DWT



# **Evaluations:** Approximations



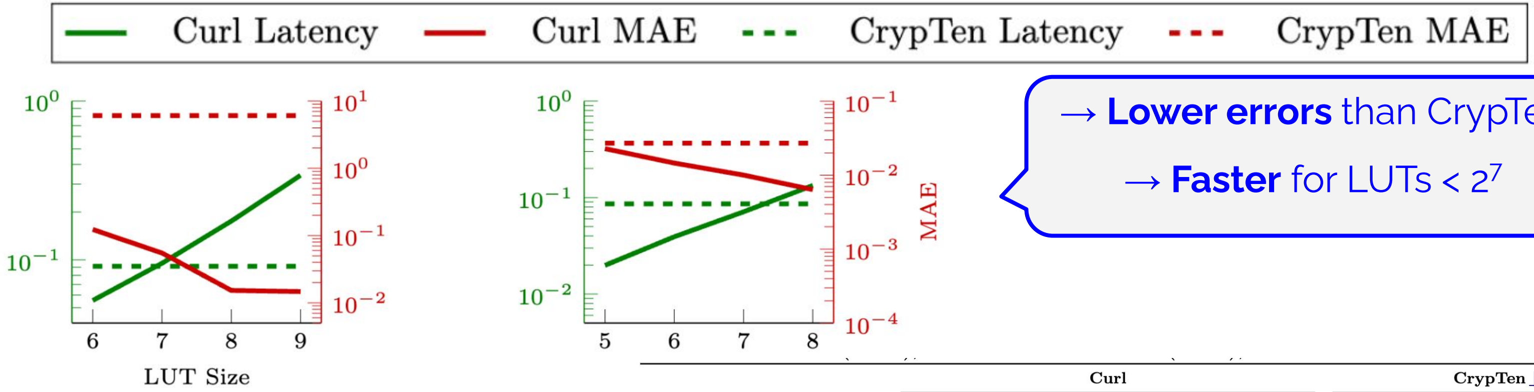

Square Root

Inverse Square Root

→ **Lower errors** than CrypTen

→ **Faster** for LUTs < $2^7$

# **Evaluations:** Approximations




Square Root

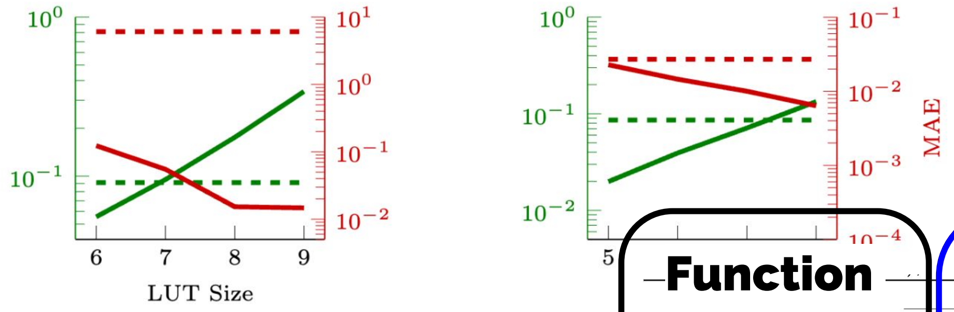
Inv

→ **Lower errors** than CrypTen

→ **Faster** for LUTs < $2^7$

| Op. | Protocol | Domain | LUT | Curl Latency (sec.)[†] | Curl Com.[‡] Rounds | Curl Com. MB | Curl Error[§] MAE | Curl Error MRE | CrypTen [45] Latency (sec.) | CrypTen Com. Rounds | CrypTen Com. MB | CrypTen Error MAE | CrypTen Error MRE |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| log | Fig. 7 | (0, 64) | $2^8$ | 0.17 | 4 | 2.6 | 2.09e-2 | 5.48e-2 | 0.17 | 40 | 39.8 | 2.14e-2 | 6.36e-3 |
| reciprocal | Fig. 7 | (1, 64) | $2^7$ | 0.09 | 4 | 2.6 | 7.18e-4 | 1.43e-3 | 0.11 | 59 | 38.3 | 1.7e-4 | 7.05e-3 |
| sqrt | Fig. 7 | (0, 256) | $2^6$ | 0.06 | 4 | 2.6 | 1.23-1 | 1.11e-2 | 0.09 | 26 | 17.3 | 6.09e+0 | 4.04e-1 |
| invsqrt | Fig. 6 | (0, 256) | $2^6$ | 0.04 | 2 | 1.0 | 1.45e-2 | 1.14e-1 | 0.09 | 24 | 15.7 | 2.69e-2 | 0.405e-1 |
| sin | App. B.3 | (−64, 64) | $2^5$ | 0.08 | 16 | 20.4 | 4.55e-3 | 1.14e-2 | 0.11 | 37 | 24.1 | 8.52e-1 | 1.58e+0 |
| cos | App. B.3 | (−64, 64) | $2^5$ | 0.08 | 16 | 20.4 | 4.77e-3 | 9.85e-2 | 0.10 | 37 | 24.1 | 8.86e-1 | 1.45e+0 |
| sigmoid | Fig. 11<br>Fig. 7 | (−64, 64)<br>(−64, 64) | $2^6$<br>$2^6$ | 0.10<br>0.10 | 22<br>4 | 33.6<br>2.6 | 4.70e-5<br>1.11e-2 | 7.83e-2<br>6.59e-2 | 0.11<br>0.11 | 26<br>26 | 26.2<br>26.2 | 7.00e-5<br>7.00e-5 | 3.49e+0<br>3.49e+0 |
| tanh | Fig. 11 | (−64, 64) | $2^5$ | 0.09 | 22 | 33.6 | 2.31e-4 | 3.96e-4 | 0.13 | 26 | 26.2 | 8.60e-5 | 1.19e-4 |
| erf | Fig. 11 | (−64, 64) | $2^3$ | 0.09 | 22 | 33.6 | 8.98e-4 | 1.83e-3 | 0.21 | 56 | 36.2 | 3.39e+7 | 3.40e+7 |
| GeLU | App. B.2<br>Fig. 7 | (−64, 64)<br>(−4, 4) | $2^4$<br>$2^4$ | 0.10<br>0.11 | 30<br>4 | 47.7<br>2.6 | 5.95e-3<br>2.60e-3 | 2.79e+0<br>5.02e-2 | N/A<br>N/A | N/A<br>N/A | N/A<br>N/A | N/A<br>N/A | N/A<br>N/A |
| SiLU | App. B.2<br>Fig. 7 | (−64, 64)<br>(−64, 64) | $2^6$<br>$2^6$ | 0.14<br>0.09 | 30<br>4 | 47.7<br>2.6 | 2.61e-3<br>1.54e-1 | 5.48e-3<br>1.18e-1 | N/A<br>N/A | N/A<br>N/A | N/A<br>N/A | N/A<br>N/A | N/A<br>N/A |

# **Evaluations:** Approximations


Square Root

In

→ **Lower errors** than CrypTen

→ **Faster** for LUTs < $2^7$

| **Functions** Op. | Protocol | Domain | LUT | **Curl** Latency (sec.)† | Com.‡ Rounds | MB | Error§ MAE | MRE | **CrypTen** Latency (sec.) | Com. Rounds | MB | Error MAE | MRE |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| log | Fig. 7 | (0, 64) | 2 | 0.17 | 4 | 2.6 | 2.09e-2 | 5.48e-2 | 0.17 | 40 | 39.8 | 2.14e-2 | 6.36e-3 |
| reciprocal | Fig. 7 | (1, 64) | 2 | 0.09 | 4 | 2.6 | 7.18e-4 | 1.43e-3 | 0.11 | 59 | 38.3 | 1.7e-4 | 7.05e-3 |
| sqrt | Fig. 7 | (0, 256) | 2 | 0.06 | 4 | 2.6 | 1.23-1 | 1.11e-2 | 0.09 | 26 | 17.3 | 6.09e+0 | 4.04e-1 |
| invsqrt | Fig. 6 | (0, 256) | 2 | 0.04 | 2 | 1.0 | 1.45e-2 | 1.14e-1 | 0.09 | 24 | 15.7 | 2.69e-2 | 0.405e-1 |
| sin | App. B.3 | (−64, 64) | 2 | 0.08 | 16 | 20.4 | 4.55e-3 | 1.14e-2 | 0.11 | 37 | 24.1 | 8.52e-1 | 1.58e+0 |
| cos | App. B.3 | (−64, 64) | 2 | 0.08 | 16 | 20.4 | 4.77e-3 | 9.85e-2 | 0.10 | 37 | 24.1 | 8.86e-1 | 1.45e+0 |
| sigmoid | Fig. 11 | (−64, 64) | 2 | 0.10 | 22 | 33.6 | 4.70e-5 | 7.83e-2 | 0.11 | 26 | 26.2 | 7.00e-5 | 3.49e+0 |
| | Fig. 7 | (−64, 64) | 2 | 0.10 | 4 | 2.6 | 1.11e-2 | 6.59e-2 | 0.11 | 26 | 26.2 | 7.00e-5 | 3.49e+0 |
| tanh | Fig. 11 | (−64, 64) | 2 | 0.09 | 22 | 33.6 | 2.31e-4 | 3.96e-4 | 0.13 | 26 | 26.2 | 8.60e-5 | 1.19e-4 |
| erf | Fig. 11 | (−64, 64) | 2 | 0.09 | 22 | 33.6 | 8.98e-4 | 1.83e-3 | 0.21 | 56 | 36.2 | 3.39e+7 | 3.40e+7 |
| GeLU | App. B.2 | (−64, 64) | 2 | 0.10 | 30 | 47.7 | 5.95e-3 | 2.79e+0 | N/A | N/A | N/A | N/A | N/A |
| | Fig. 7 | (−4, 4) | 2 | 0.11 | 4 | 2.6 | 2.60e-3 | 5.02e-2 | N/A | N/A | N/A | N/A | N/A |
| SiLU | App. B.2 | (−64, 64) | 2 | 0.14 | 30 | 47.7 | 2.61e-3 | 5.48e-3 | N/A | N/A | N/A | N/A | N/A |
| | Fig. 7 | (−64, 64) | 2 | 0.09 | 4 | 2.6 | 1.54e-1 | 1.18e-1 | N/A | N/A | N/A | N/A | N/A |

# **Evaluations:** Running LLMs in seconds

# **Evaluations:** Running LLMs in seconds

**Sequence length = 64**

| Model | Latency (s) | Rounds | Com. (GB) |
|---|---|---|---|
| BERT Tiny | 3.55 | 409 | 1.34 |
| BERT Base | 13.63 | 1,629 | 2.8 |
| BERT Large | 33.93 | 3,093 | 5.66 |
| GPT-2 | 16.61 | 1,630 | 3.77 |
| GPT-Neo | 103.4 | 3,118 | 14.9 |

# **Evaluations:** Running LLMs in seconds

**Sequence length = 64**

| Model | Latency (s) | Rounds | Com. (GB) |
|---|---|---|---|
| BERT Tiny | 3.55 | 409 | 1.34 |
| BERT Base | 13.63 | 1,629 | 2.8 |
| BERT Large | 33.93 | 3,093 | 5.66 |
| GPT-2 | 16.61 | 1,630 | 3.77 |
| GPT-Neo | 103.4 | 3,118 | 14.9 |

**BERT Base**
**(seq. len = 128)**

| Framework | Latency (s) | Rounds | Com. (GB) |
|---|---|---|---|
| Iron [35] | 475 | 13,663 | 281 |
| MPCFormer [47] | 55.3 | – | 12.1 |
| Puma [21] | 33.9 | – | 10.8 |
| Bolt [57] | 185 | 10,509 | 59.6 |
| Bolt (WE) [57]† | 91 | 10,901 | 25.7 |
| **Curl** | 22.5 | 1,629 | 5.7 |

† In Bolt, WE stands for word elimination.

# **Evaluations:** Running LLMs in seconds

**Sequence length = 64**

| Model | Latency (s) | Rounds | Com. (GB) |
|---|---|---|---|
| BERT Tiny | 3.55 | 409 | 1.34 |
| BERT Base | 13.63 | 1,629 | 2.8 |
| BERT Large | 33.93 | 3,093 | 5.66 |
| GPT-2 | 16.61 | 1,630 | 3.77 |
| GPT-Neo | 103.4 | 3,118 | 14.9 |

**BERT Base (seq. len = 128)**

| Framework | Latency (s) | Rounds | Com. (GB) |
|---|---|---|---|
| Iron [35] | 475 | 13,663 | 281 |
| MPCFormer [47] | 55.3 | – | 12.1 |
| Puma [21] | 33.9 | – | 10.8 |
| Bolt [57] | 185 | 10,509 | 59.6 |
| Bolt (WE) [57]† | 91 | 10,901 | 25.7 |
| **Curl** | 22.5 | 1,629 | 5.7 |

† In Bolt, WE stands for word elimination.

**Fastest runtime**

# **Evaluations:** Running LLMs in seconds

**Sequence length = 64**

| Model | Latency (s) | Rounds | Com. (GB) |
|---|---|---|---|
| BERT Tiny | 3.55 | 409 | 1.34 |
| BERT Base | 13.63 | 1,629 | 2.8 |
| BERT Large | 33.93 | 3,093 | 5.66 |
| GPT-2 | 16.61 | 1,630 | 3.77 |
| GPT-Neo | 103.4 | 3,118 | 14.9 |

**BERT Base (seq. len = 128)**

| Framework | Latency (s) | Rounds | Com. (GB) |
|---|---|---|---|
| Iron [35] | 475 | 13,663 | 281 |
| MPCFormer [47] | 55.3 | – | 12.1 |
| Puma [21] | 33.9 | – | 10.8 |
| Bolt [57] | 185 | 10,509 | 59.6 |
| Bolt (WE) [57]† | 91 | 10,901 | 25.7 |
| **Curl** | 22.5 | 1,629 | 5.7 |

† In Bolt, WE stands for word elimination.

Fastest runtime

Fewer Rounds

# **Evaluations:** Running LLMs in seconds

**Sequence length = 64**

| Model | Latency (s) | Rounds | Com. (GB) |
|---|---|---|---|
| BERT Tiny | 3.55 | 409 | 1.34 |
| BERT Base | 13.63 | 1,629 | 2.8 |
| BERT Large | 33.93 | 3,093 | 5.66 |
| GPT-2 | 16.61 | 1,630 | 3.77 |
| GPT-Neo | 103.4 | 3,118 | 14.9 |

**BERT Base (seq. len = 128)**

| Framework | Latency (s) | Rounds | Com. (GB) |
|---|---|---|---|
| Iron [35] | 475 | 13,663 | 281 |
| MPCFormer [47] | 55.3 | – | 12.1 |
| Puma [21] | 33.9 | – | 10.8 |
| Bolt [57] | 185 | 10,509 | 59.6 |
| Bolt (WE) [57]† | 91 | 10,901 | 25.7 |
| **Curl** | 22.5 | 1,629 | 5.7 |

† In Bolt, WE stands for word elimination.

Fastest runtime

Fewer Rounds

Lowest Communication

# Conclusions

# Conclusions

- Lookup Tables (LUTs) can be used to evaluate non-linear functions in MPC
  - LUTs **scale poorly** for high precision → enormous communication.
  - **Polynomial approximations** and **quantization** yield **low accuracy**!

# Conclusions

- Lookup Tables (LUTs) can be used to evaluate non-linear functions in MPC
  - LUTs **scale poorly** for high precision → enormous communication.
  - **Polynomial approximations** and **quantization** yield **low accuracy**!

- **Curl:** smaller LUTs without sacrificing accuracy
  - Using Discrete Wavelet Transforms (DWT) → **low communication**
  - Reduced LUT sizes → **high accuracy**
  - Run LLMs (BERT Tiny/Base/Large, GPT-2, GPT Neo) → **in seconds!**

# Conclusions

- Lookup Tables (LUTs) can be used to evaluate non-linear functions in MPC
  - LUTs **scale poorly** for high precision → enormous communication.
  - **Polynomial approximations** and **quantization** yield **low accuracy**!

- **Curl:** smaller LUTs without sacrificing accuracy
  - Using Discrete Wavelet Transforms (DWT) → **low communication**
  - Reduced LUT sizes → **high accuracy**
  - Run LLMs (BERT Tiny/Base/Large, GPT-2, GPT Neo) → **in seconds!**

- Curl's technique can enhance related works:
  - **FHE,** Ripple [1]
  - **FSS,** Wave Hello to Privacy [2]

# Conclusions

- Lookup Tables (LUTs) can be used to evaluate non-linear functions in MPC
  - LUTs **scale poorly** for high precision → enormous communication.
  - **Polynomial approximations** and **quantization** yield **low accuracy**!

- **Curl:** smaller LUTs without sacrificing accuracy
  - Using Discrete Wavelet Transforms (DWT) → **low communication**
  - Reduced LUT sizes → **high accuracy**
  - Run LLMs (BERT Tiny/Base/Large, GPT-2, GPT Neo) → **in seconds!**

- Curl's technique can enhance related works:
  - **FHE,** Ripple [1]
  - **FSS,** Wave Hello to Privacy [2]

[1] C. Gouert, M. Ugurbil, D. Mouris, M. de Vega, and N. G. Tsoutsos. **Ripple: Accelerating Programmable Bootstraps for FHE with Wavelet Approximations.** In International Conference on Information Security (ISC), 2024.

[2] J. Reis, M. Ugurbil, S. Wagh, R. Henry, M. de Vega. **Wave Hello to Privacy: Efficient Mixed-Mode MPC using Wavelet Transforms**, accepted to PoPETs 2025.

Portugal Crypto Day

December 13, 2024, Lisbon

# Curl: Private LLMs through Wavelet-Encoded Look-Up Tables

**Manuel B. Santos**[1], Dimitris Mouris[1], Mehmet Ugurbil[1], Stanislaw Jarecki[1,2], José Reis[1], Shubho Sengupta[3] and Miguel de Vega[1]

{ manuel.santos, dimitris, memo, stanislaw.jarecki, jose.reis, miguel }@nillion.com

ssengupta@meta.com

*https://ia.cr/2024/1127*

*https://github.com/jimouris/curl*

1 nillion

2 THE UNIVERSITY OF CALIFORNIA · IRVINE

3 Meta