Cloud Encryption and Blockchain:

A primer on TFHE and Fhevm

Nigel Smart

Dec 2025

About Zama

Zama is a cryptography company building open source homomorphic encryption tools for developers.

Founded in 2020 by Pascal Paillier, a pioneer in FHE, and Rand Hindi, a serial privacy entrepreneur, Zama continues to lead the way in the development and implementation of homomorphic encryption technology.

Over

140

Million USD

in funding raised

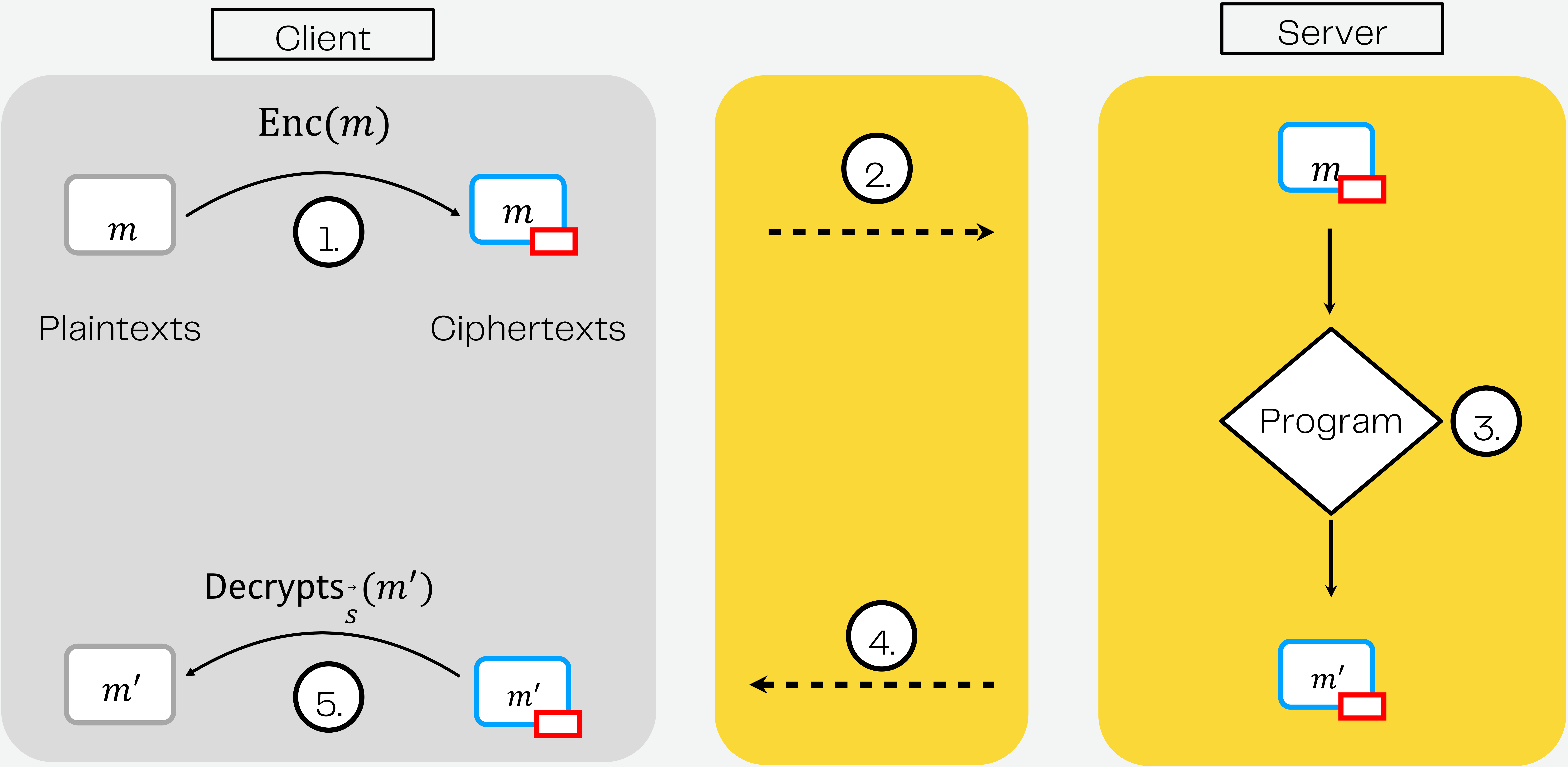
~100 Team members

42% PHDs

25 Nationalities

Introduction to **TFHE**

Fully Homomorphic Encryption (FHE)



Data stays encrypted!

4

Learning With Errors (LWE)

LWE Sample:

Secret key	$\boldsymbol{s} \leftarrow \mathcal{D}_s^n$	Mask	$\boldsymbol{a} \leftarrow \mathcal{U}(\mathbb{Z}_q)^n$	Error (a.k.a. noise)	$e \leftarrow \mathcal{D}_e$

$$(\boldsymbol{a}, b) = \underbrace{\boldsymbol{a}}_{n} \cdot \boldsymbol{s} + e \mod q$$

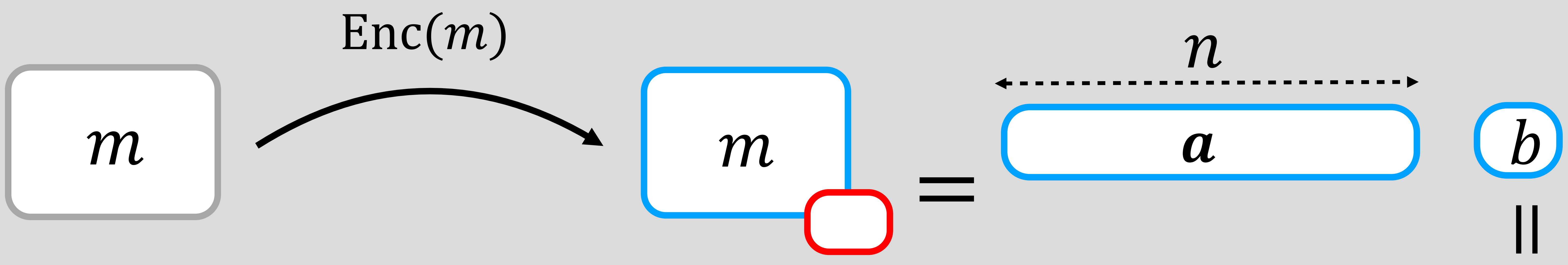
Decision LWE: Given d samples $(\vec{a}_i, b_i)$ find if $b_i = \langle a_i, s \rangle + e_i$ (LWE samples) or if $b \leftarrow \mathcal{U}(\mathbb{Z}_q)$ (Random samples)

Search LWE: Given d LWE samples $(\vec{a}_i, b_i)$, with $b_i = \langle a_i, s \rangle + e_i$, **find the secret key**

5

LWE-based Ciphertexts

Learning With Errors



$$b = \langle \boldsymbol{a}, \boldsymbol{s} \rangle + \Delta m + e \bmod q$$

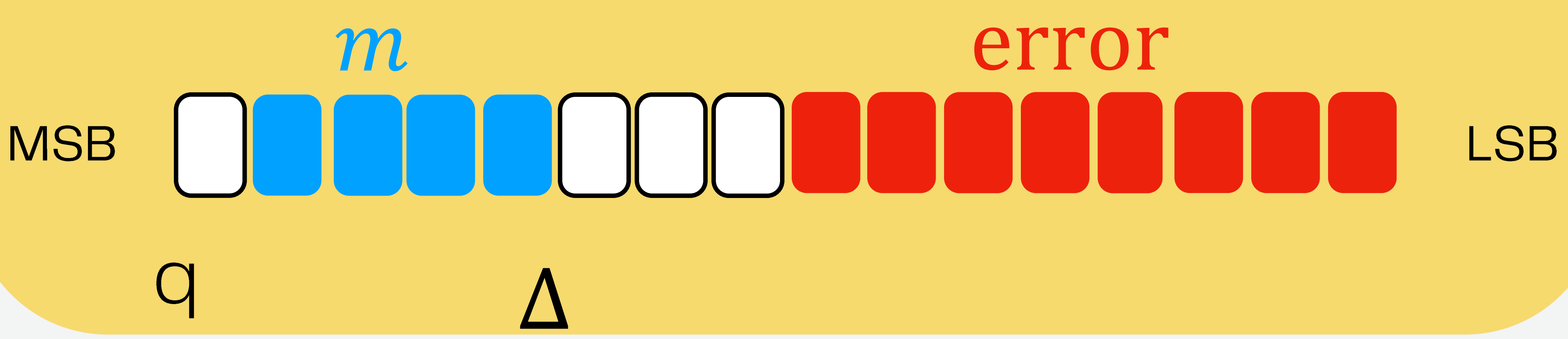
Secret key $\boldsymbol{s} \leftarrow \mathcal{U}(\{0,1\})^n$

Mask $\boldsymbol{a} \leftarrow \mathcal{U}(\mathbb{Z}_q)^n$

Error $e \leftarrow \mathcal{N}_\sigma$

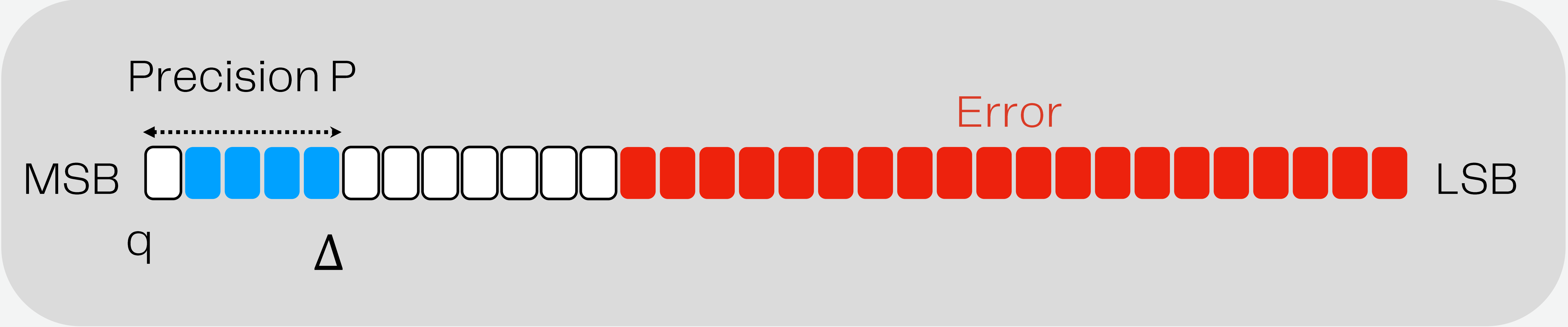
(a.k.a. noise)

Plaintext Representation



Plaintext Representation

$$\Delta m + e \; mod \; \mathrm{q}$$

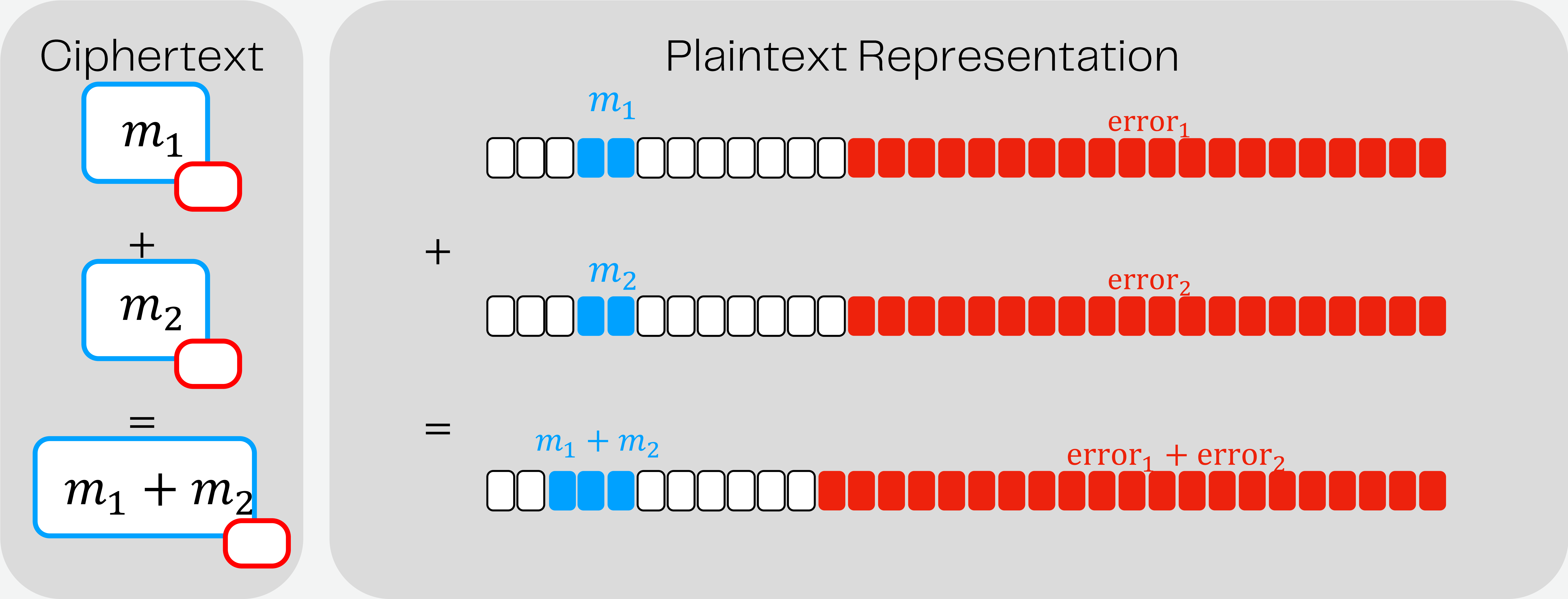


Example of parameters

$n \approx 900$ | $\mathrm{q} = 2^{64}$ | $2 \leq \mathrm{P} \leq 10$

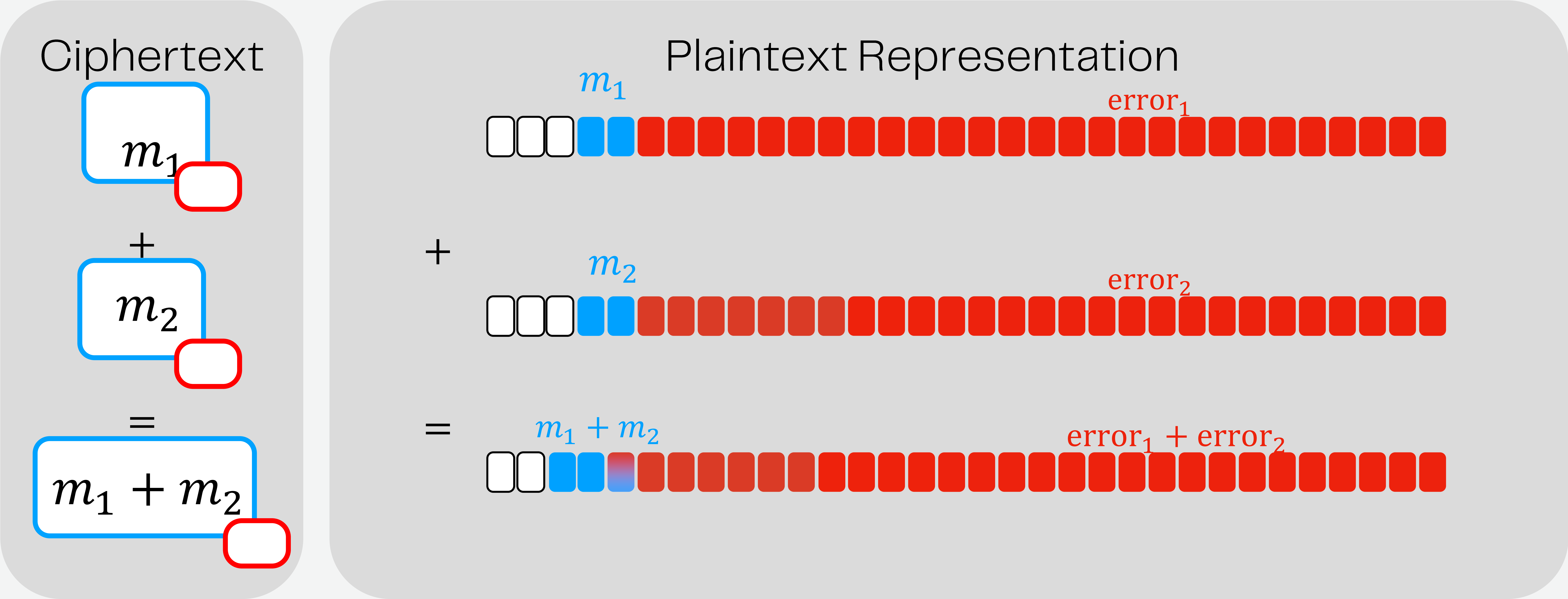
7

Homomorphic Addition



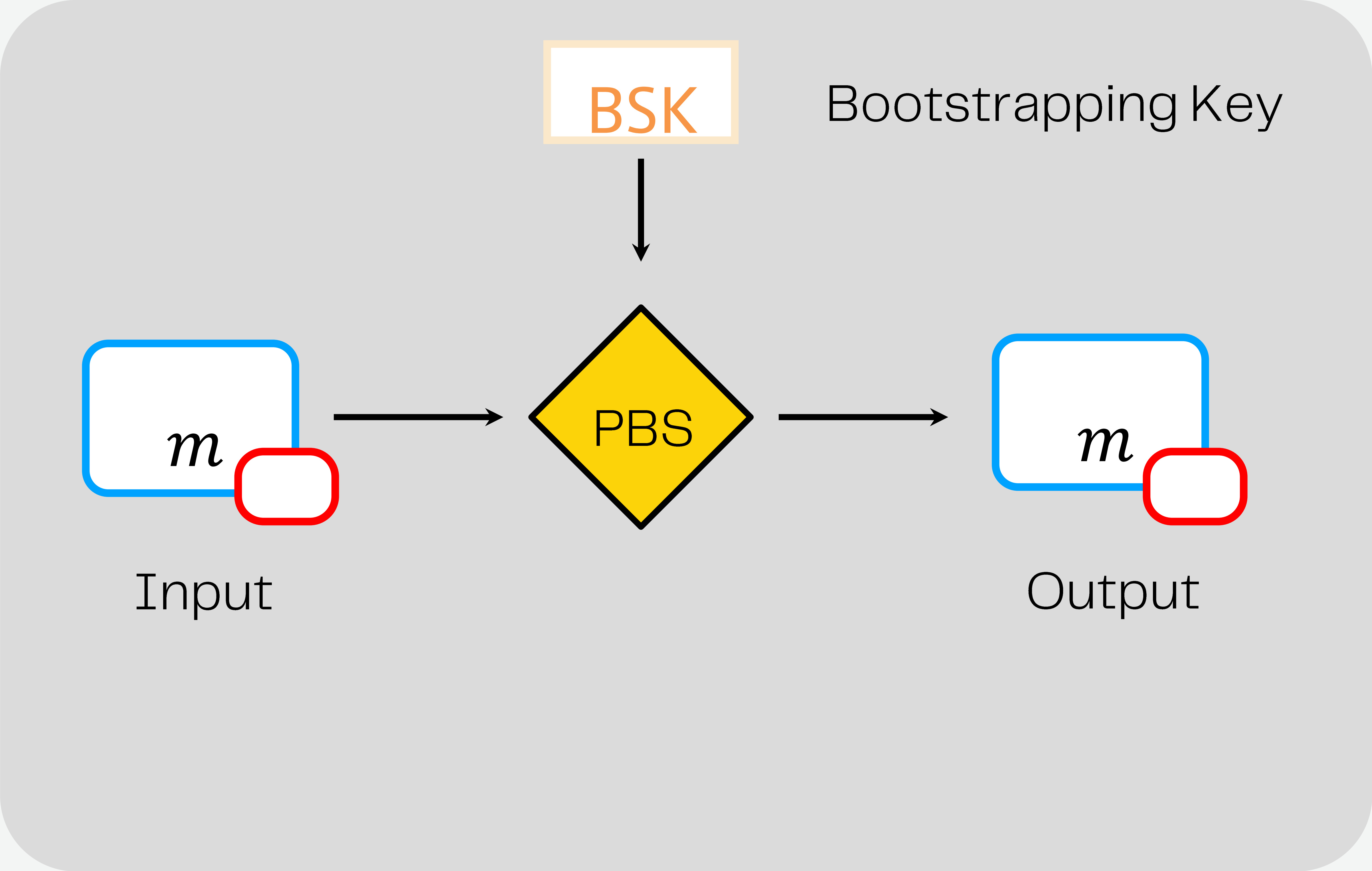
Message & Error sizes increase

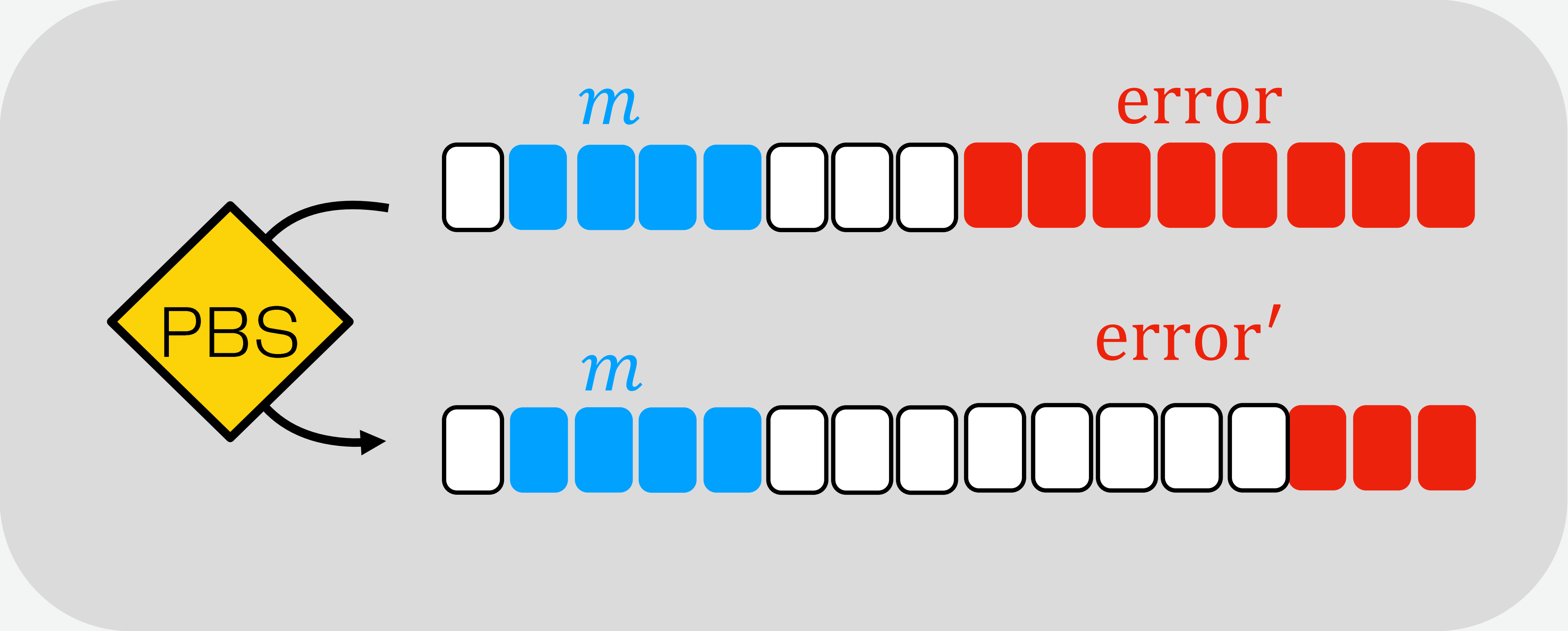
Homomorphic Addition



Noise management needed

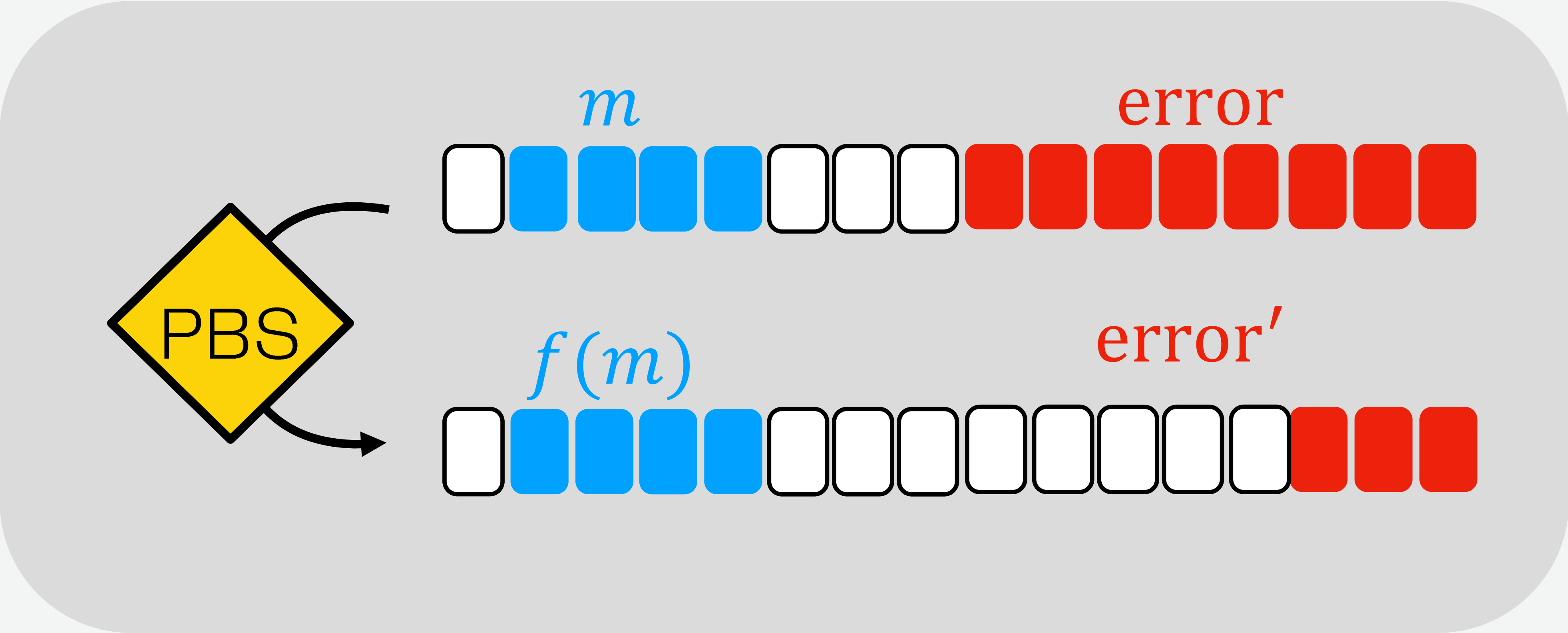
TFHE Programmable Bootstrapping (PBS)





10

TFHE Programmable Bootstrapping (PBS)



Low latency
e.g. ≈ 945μs for 4 bits

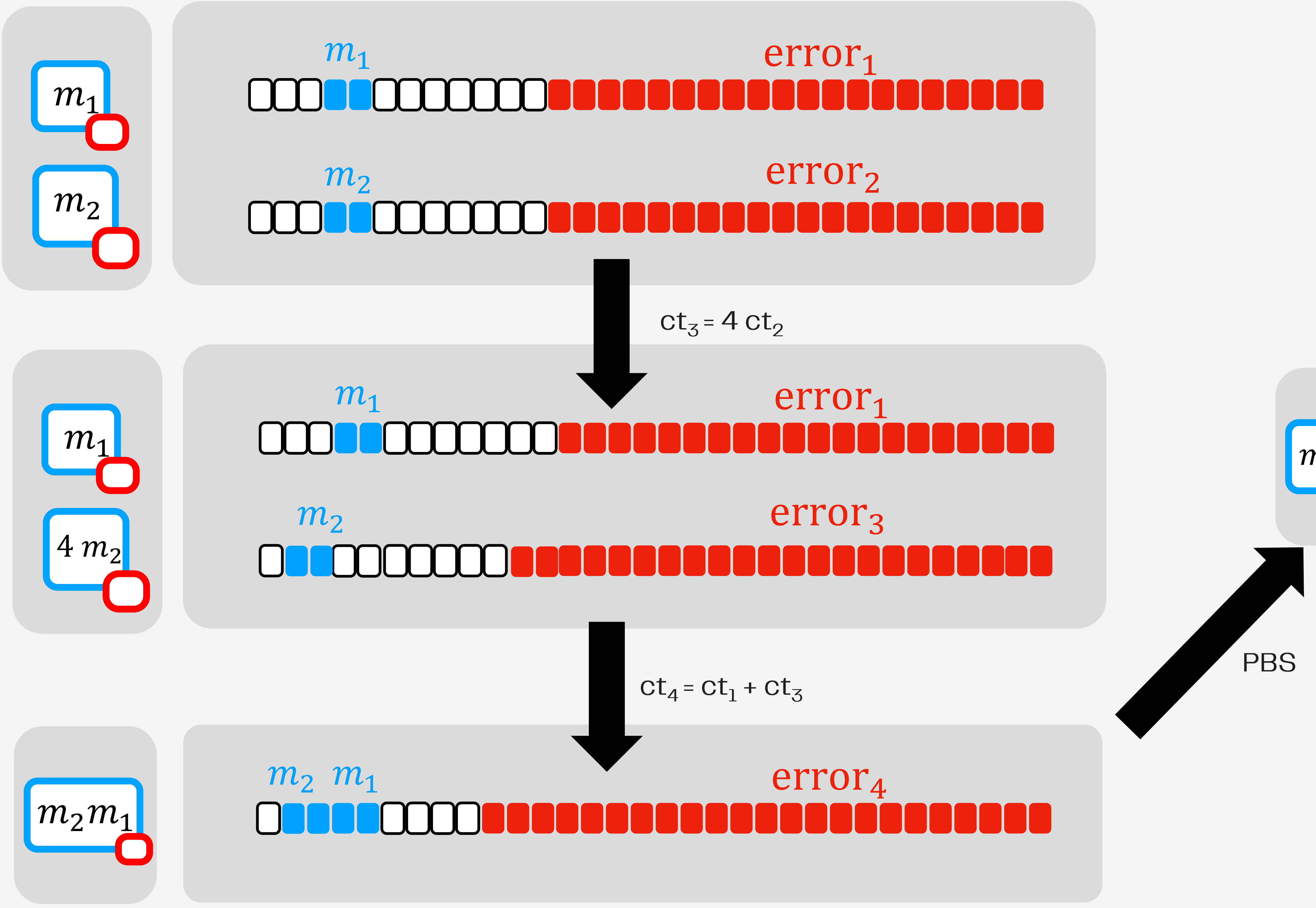
Small messages only:

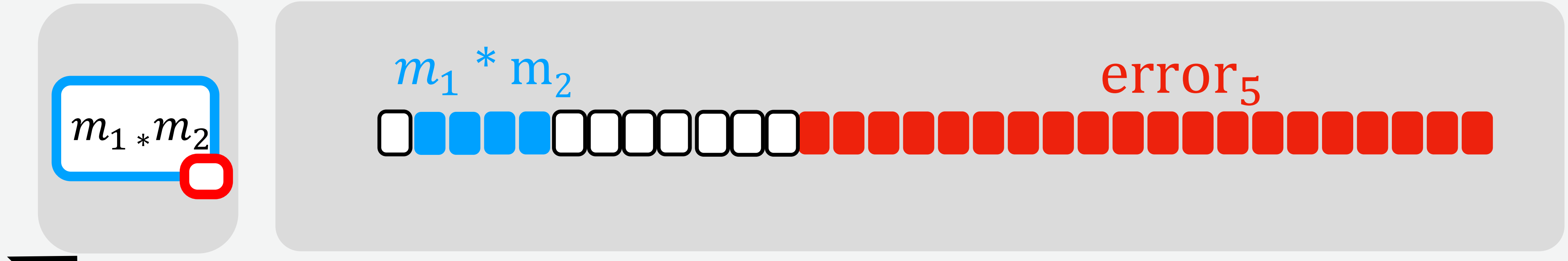
$|m| \leq 10 bits$

Noise reduction &
Homomorphic evaluation of any $f(\cdot)$

11

Homomorphic Multiplication



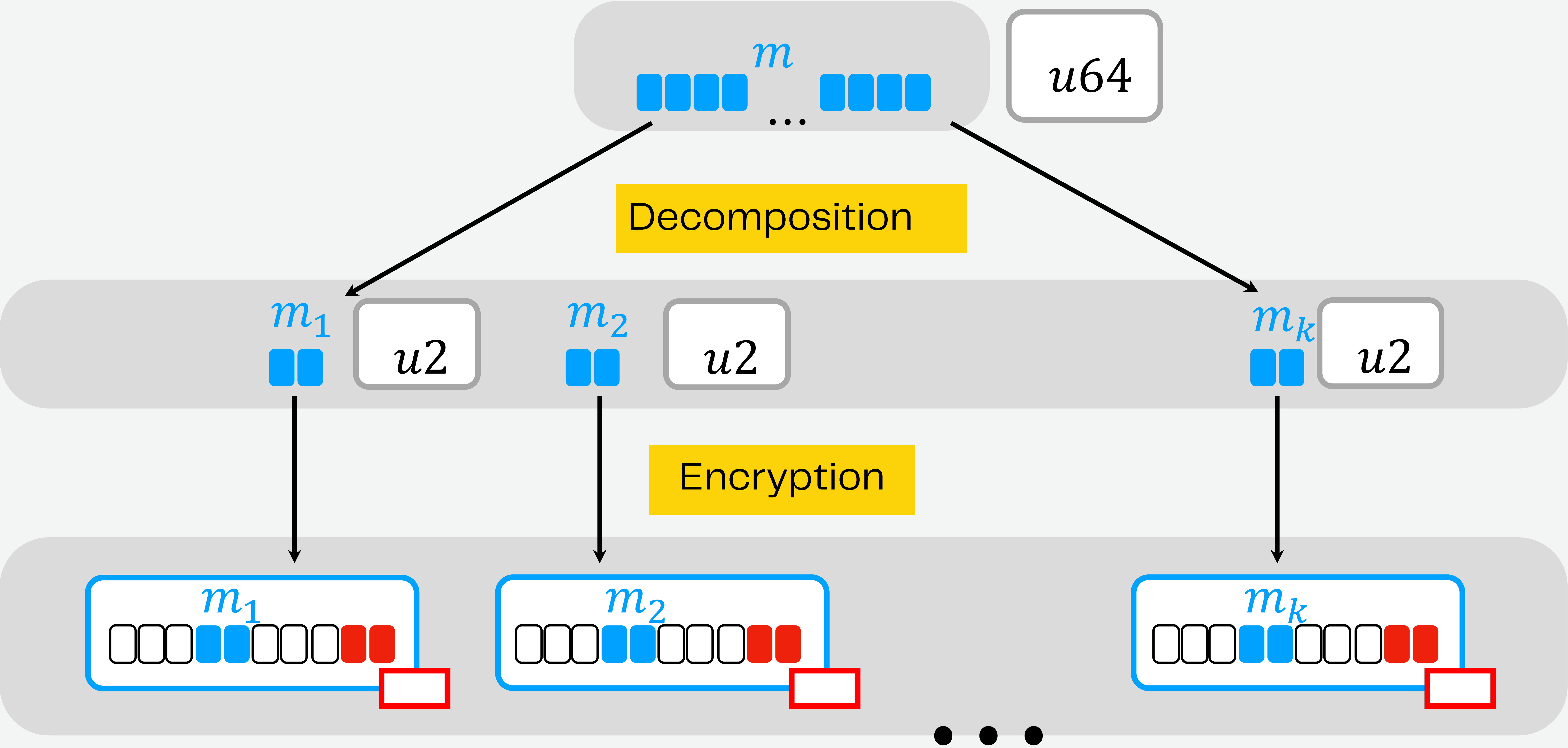


x	F(x)
0000	0000
0001	0000
0010	0000
0011	0000
0100	0000
0101	0001
0110	0010
0111	0011

x	F(x)
1000	0000
1001	0010
1010	0100
1011	0110
1100	0000
1101	0011
1110	0110
1111	1001

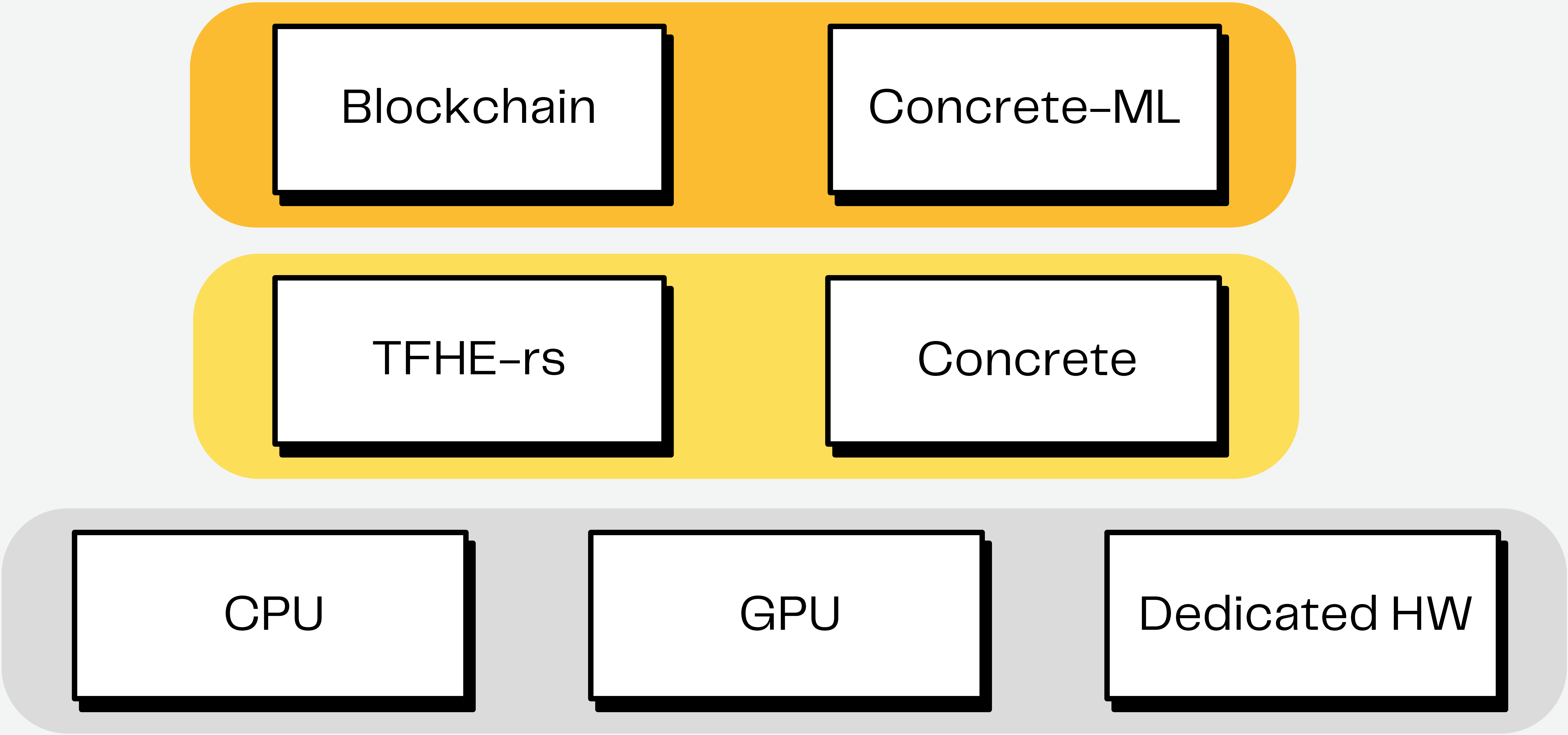
12

Homomorphic Integers



64 bit integers ≈ 32 LWE Ciphertexts

Zama's Product Stack



14

Zama's Product Stack

Blockchain

TFHE-rs

How to make TFHE work better in a Blockchain scenario?

1. Encrypt

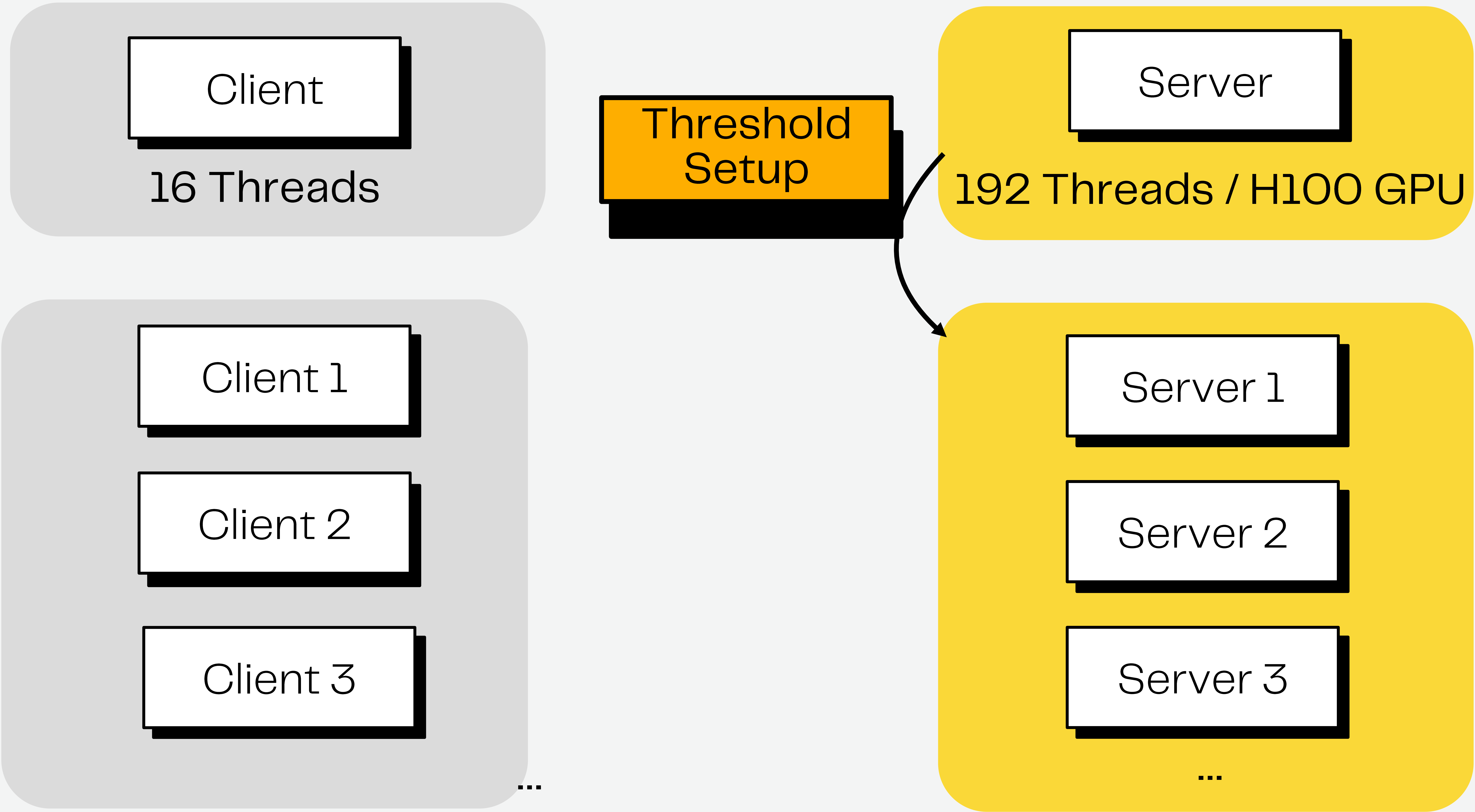
2. Compute

3. Decrypt

15

Making **TFHE** Blockchain Friendly

Machine Assumptions



Compact Public Key Encryption

LWE-based

$pk_{Server} = \{0, 0, \ldots, 0\}$ ≈ 130,000 w/ usual parameters

$\{ct\} = m_0, m_1, \ldots, m_h$

Large pk_{Server} & noisy ciphertexts

RLWE-based

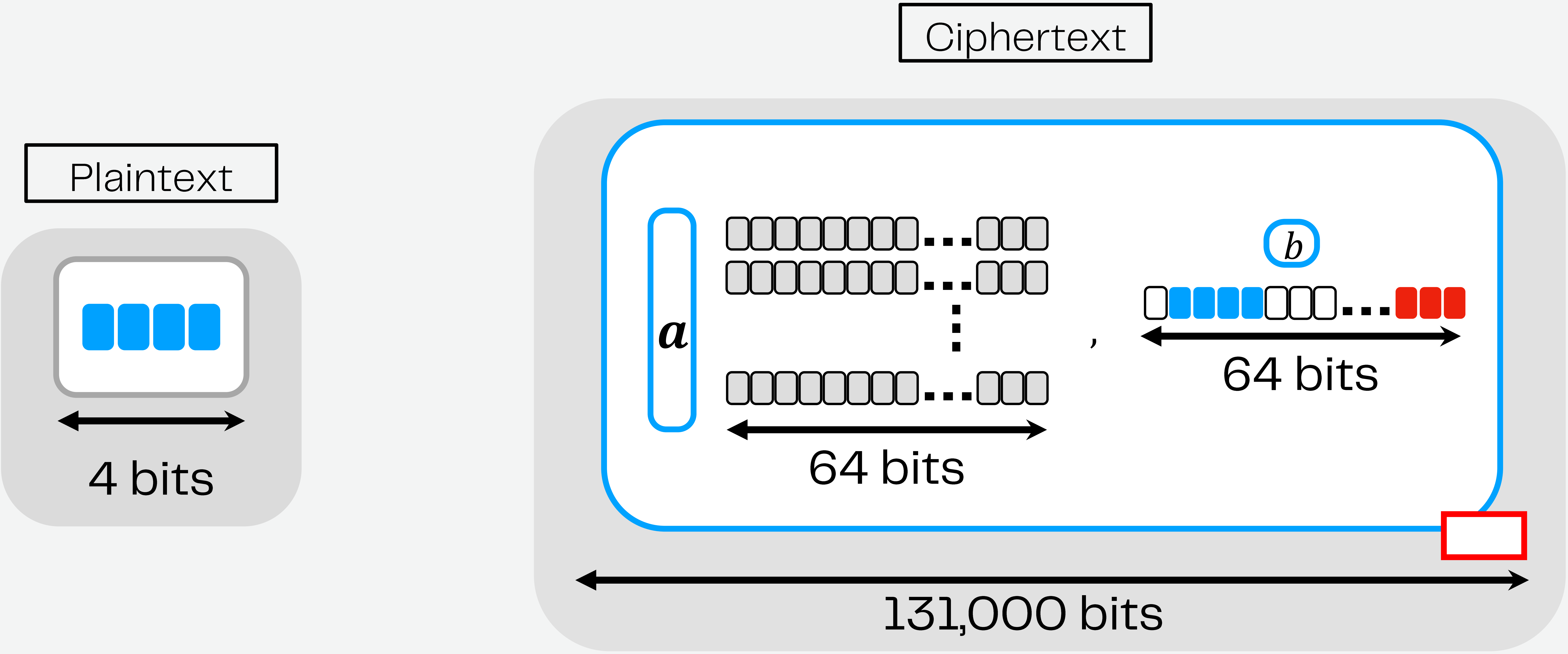
$pk_{Server} = 0$

$\{ct\} = m_0 \; m_1 \; m_h$

Smaller pk_{Server} & less noisy cts

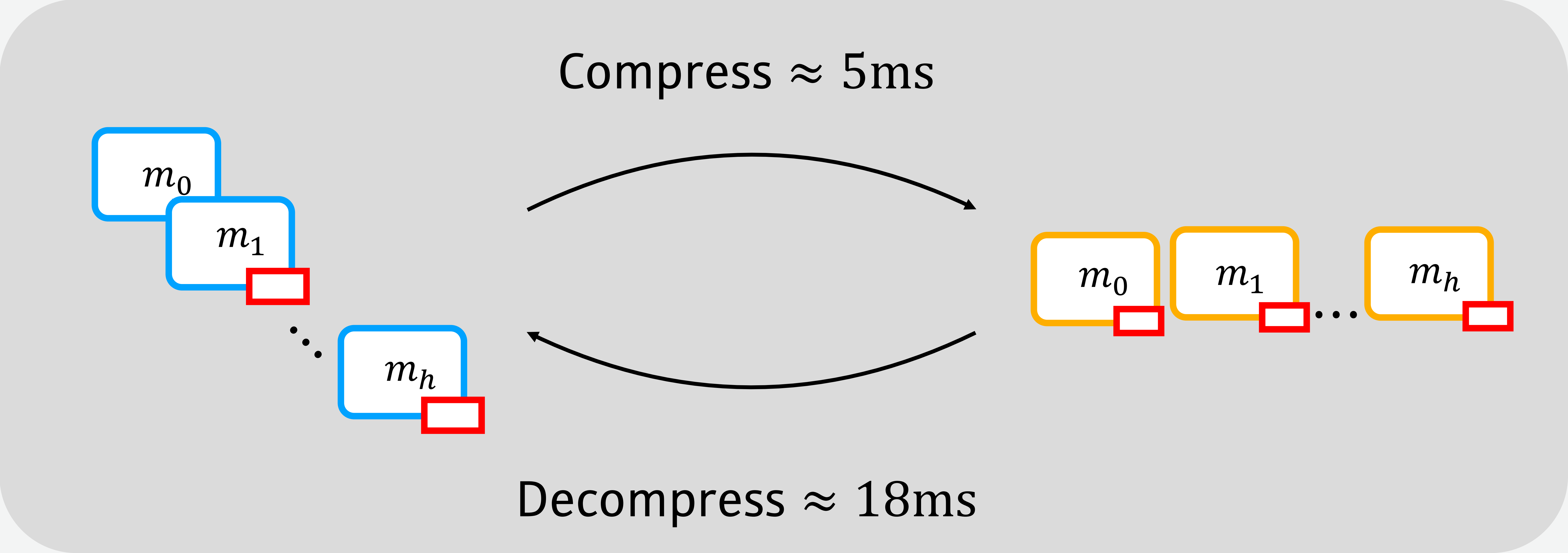
18

Standard Ciphertext Size



1 clear bit encrypted as 32,750 bits

Ciphertext Compression



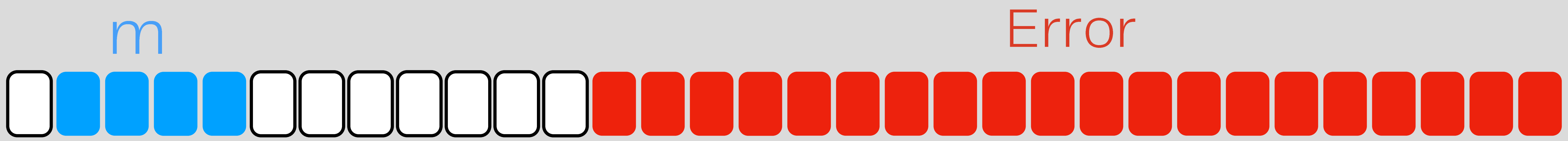
Type	Size (kB)	Compressed Size (kB)
FheUint2	16	1.70
FheUint64	527	1.74
FheUint512	4214	2.08

Nominal vs compressed sizes

20

Provably Good Encryption

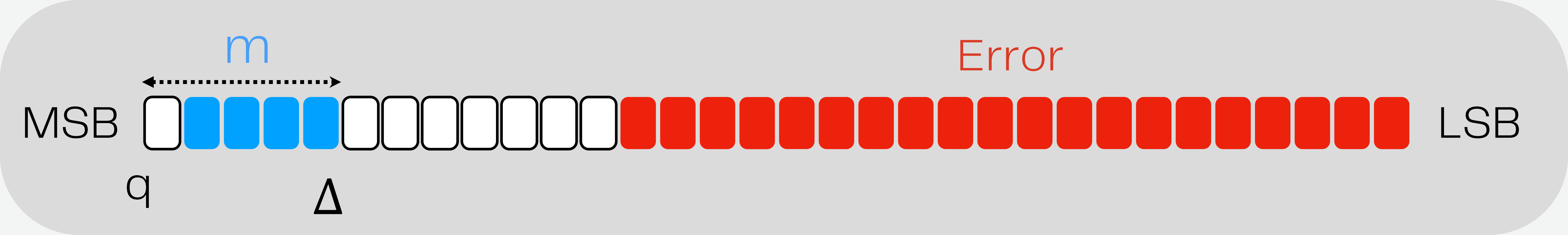
Bad encryption pattern: Message & Error mixed



Good encryption pattern: Distinct Error & Message

21

Provably Good Encryption



Good encryption: Error & message bounded

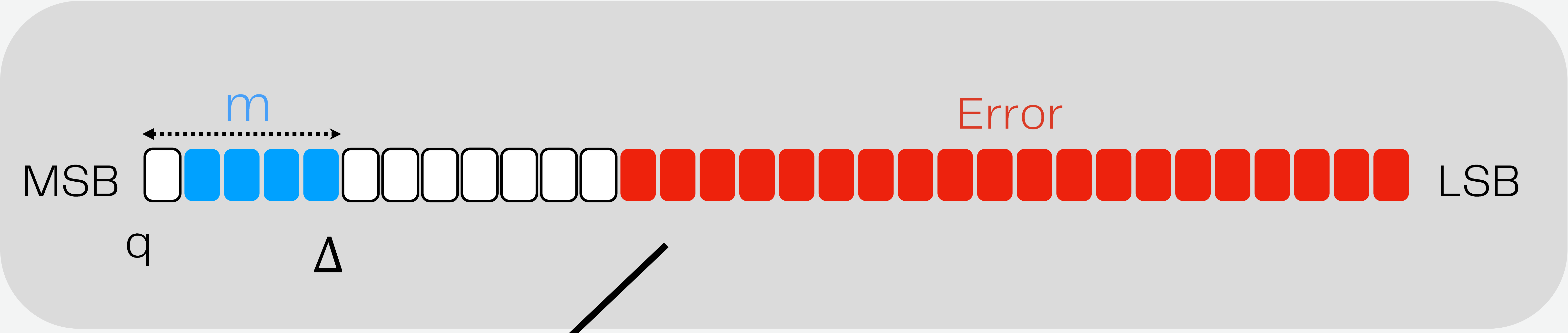
Client

Server

$$\mathrm{ct} = \mathrm{Enc}(m)$$
$$\pi = \mathrm{Prove}(\mathrm{ct})$$

$$b = \mathrm{Verify}(\pi)$$

Tweaking Noise Distribution



~~Gaussian~~ → TUniform:

Centered Tweaked Uniform

TUniform: well suited for FHE and better fit w/ MPC generation

23

Security Models

Usual Setup	Threshold Setup
IND-CPA	IND-CPA-D
$\Downarrow$	$\Downarrow$
pfail < 2^{-64}	pfail < 2^{-128}

Reducing the bootstrapping failure probability!

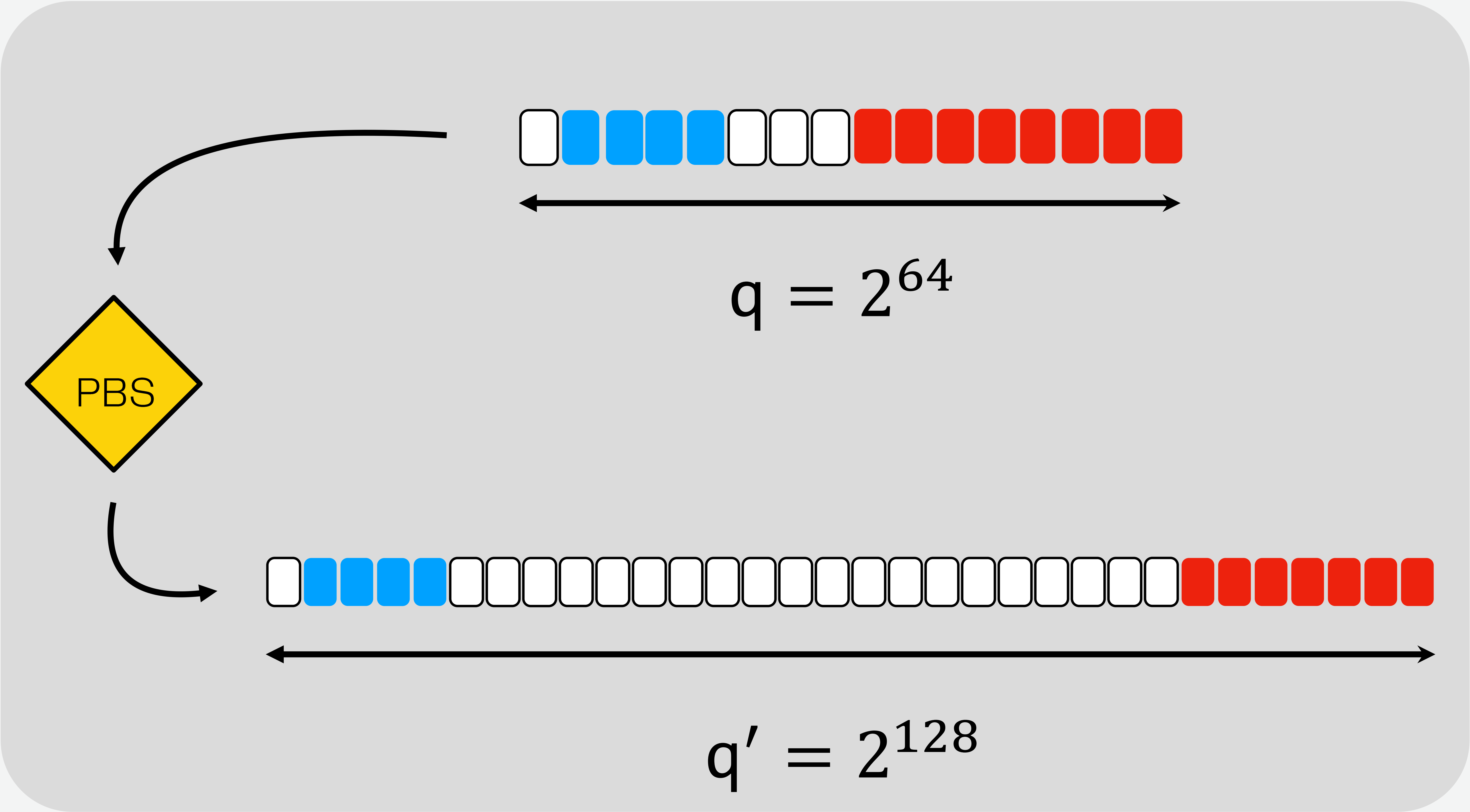
$\Rightarrow$ No impact on performance

24

Secure Threshold-Decryption

Enlarging q

Reducing the error



Now compliant w/ Noise Flooding!

25

Threshold-Friendly TFHE

Bandwidth Efficient

Storage Efficient

IND-CPA-D secure

MPC-Friendly Key Generation

The **TFHE-rs** Library

TFHE-rs

Open-source

Rust

User-friendly

```rust
use tfhe::{ConfigBuilder, generate_keys, set_server_key, FheUint8};
use tfhe::prelude::*;

fn main() {
    let config = ConfigBuilder::default().build();

    // Client-side
    let (client_key, server_key) = generate_keys(config);

    let clear_a = 27u8;
    let clear_b = 128u8;

    let a = FheUint8::encrypt(clear_a, &client_key);
    let b = FheUint8::encrypt(clear_b, &client_key);

    //Server-side
    set_server_key(server_key);
    let result = a + b;

    //Client-side
    let decrypted_result: u8 = result.decrypt(&client_key)

    let clear_result = clear_a + clear_b;

    assert_eq!(decrypted_result, clear_result);
}
```

TFHE-rs

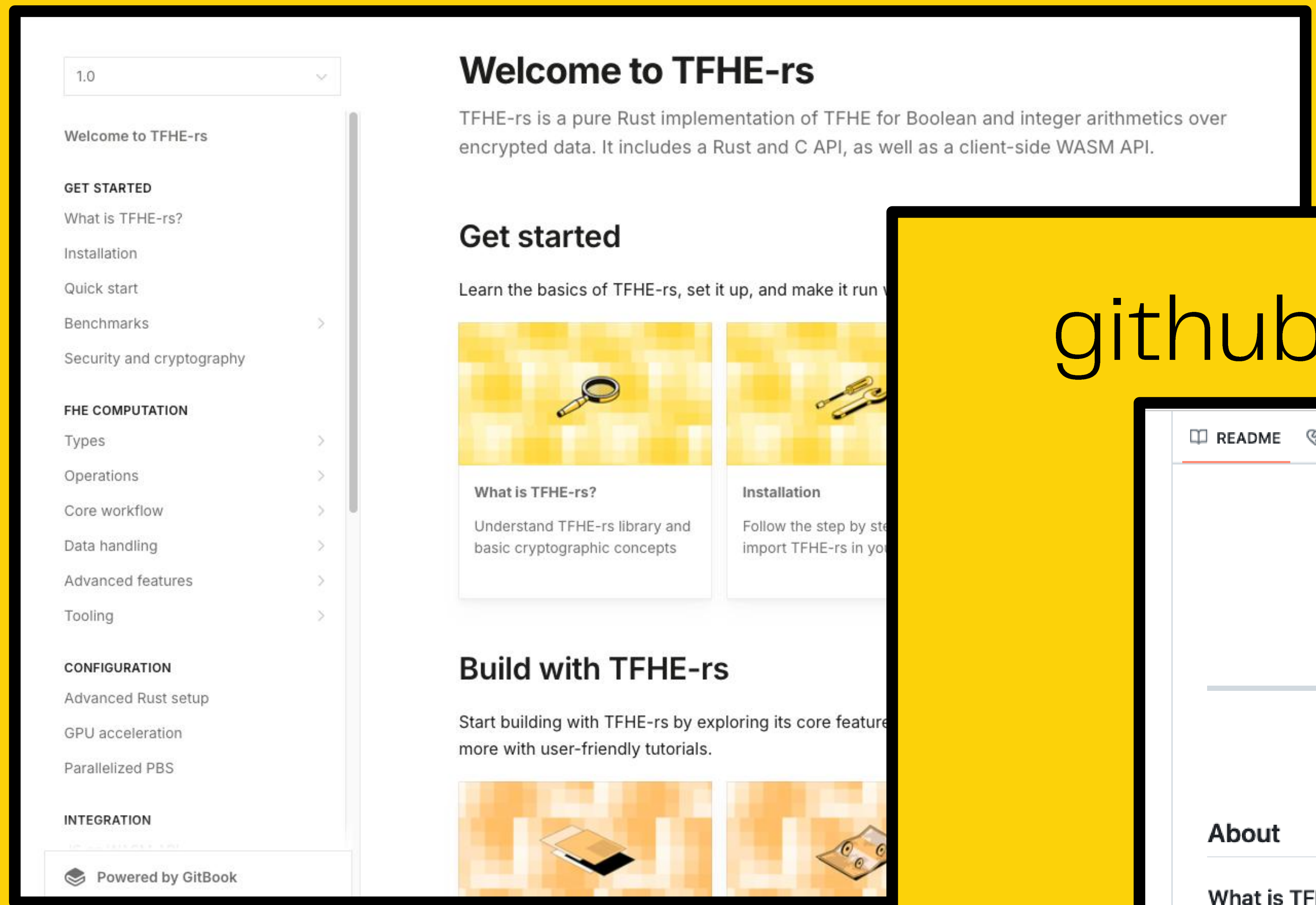
High-Level API → User Friendly API a la Rust

Homomorphic Types → Homomorphic Arithmetic & Specific Crypto Params

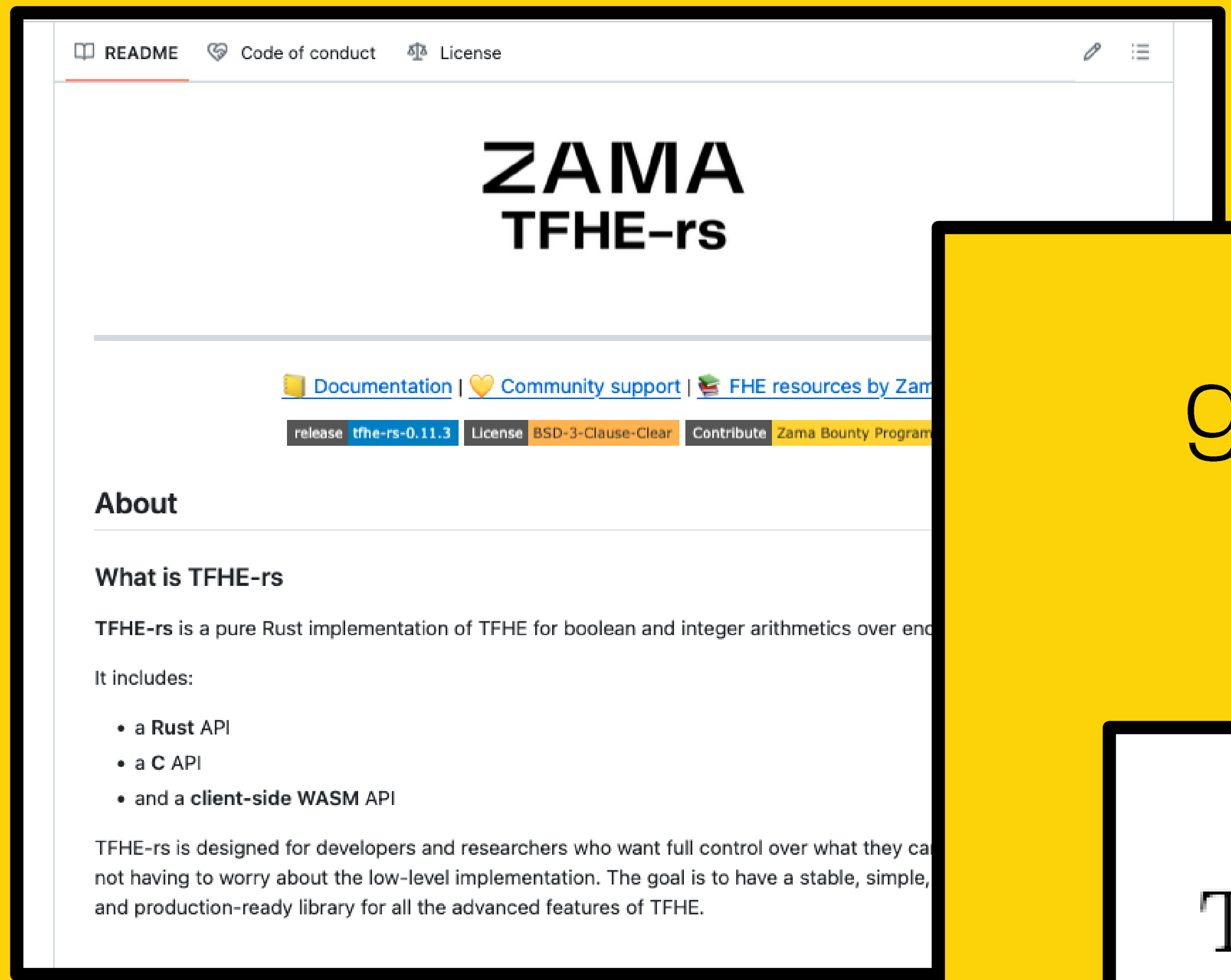
Core → Cryptographic Primitives

TFHE-rs

docs.zama.ai/tfhe-rs



github.com/zama-ai/tfhe-rs



github.com/zama-ai/tfhe-rs-handbook

TFHE-rs: A (Practical) Handbook

First Edition

Beyond FHE Computation

Public Key Encryption

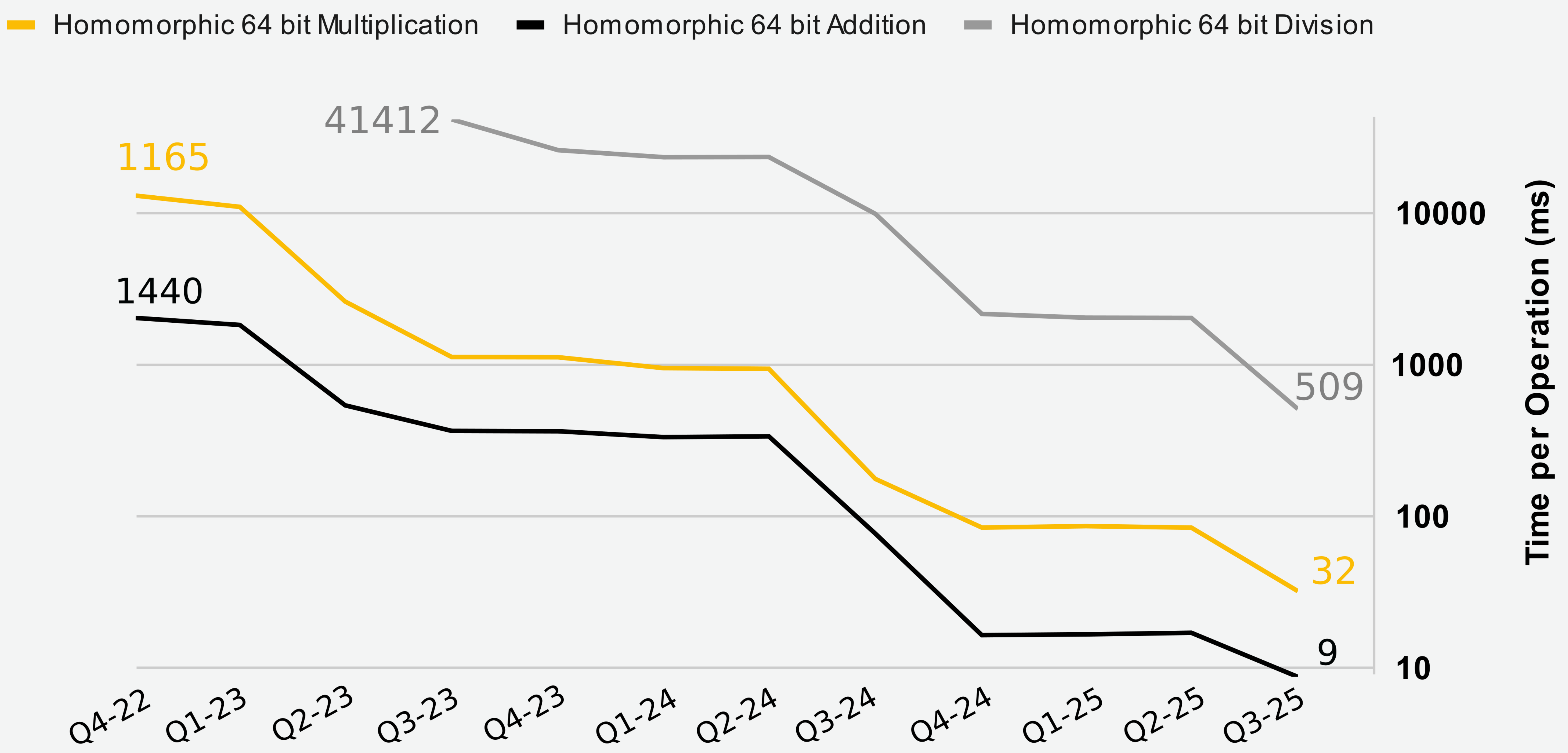
Zero-Knowledge Proofs

Stable User API

Compliance with Distributed Protocols

Ciphertext Compression

FHE Computation Timings Over Time



Zama benchmarks on FHE performance (64-bit Operations)

2022 to 2025

32

FHE Computation Latency

Operation \ Size	FheUint 8	FheUint 16	FheUint 32	FheUint 64	FheUint 128
Add / Sub (+,-)	4.74 ms	5.27 ms	6.91 ms	8.97 ms	13.4 ms
Mul (x)	9.26 ms	12.2 ms	18.3 ms	31.9 ms	79.0 ms
Equal / Not Equal (eq, ne)	3.6 ms	5.04 ms	5.42 ms	7.29 ms	8.3 ms
Comparisons (ge, gt, le, lt)	5.17 ms	6.63 ms	8.5 ms	10.6 ms	13.3 ms
Max / Min (max, min)	8.2 ms	10.0 ms	12.3 ms	14.9 ms	18.3 ms
Bitwise operations (&, |, ^)	1.44 ms	1.61 ms	1.73 ms	2.0 ms	2.28 ms
Div / Rem (/, %)	50.0 ms	93.9 ms	205 ms	502 ms	1.36 s
Left / Right Shifts (<<, >>)	9.01 ms	11.3 ms	14.6 ms	17.9 ms	22.8 ms
Left / Right Rotations (left_rotate, right_rotate)	9.0 ms	11.3 ms	14.6 ms	17.8 ms	22.8 ms
Leading / Trailing zeros/ones	10.1 ms	13.3 ms	15.6 ms	20.0 ms	24.8 ms
Log2	9.63 ms	14.0 ms	16.4 ms	21.9 ms	26.6 ms
Select	3.13 ms	3.47 ms	3.93 ms	4.63 ms	5.66 ms

33

FHE Computation Throughput

Operation \ Size	FheUint 8	FheUint 16	FheUint 32	FheUint 64	FheUint 128
Negation (-)	713 ops/s	703 ops/s	685 ops/s	648 ops/s	419 ops/s
Add / Sub (+,-)	679 ops/s	693 ops/s	655 ops/s	635 ops/s	415 ops/s
Mul (x)	542 ops/s	501 ops/s	248 ops/s	64.7 ops/s	13.1 ops/s
Equal / Not Equal (eq, ne)	564 ops/s	538 ops/s	546 ops/s	525 ops/s	376 ops/s
Comparisons (ge, gt, le, lt)	392 ops/s	382 ops/s	379 ops/s	347 ops/s	236 ops/s
Max / Min (max, min)	308 ops/s	305 ops/s	291 ops/s	247 ops/s	157 ops/s
Bitwise operations (&, |, ^)	2.35 k.ops/s	2.73 k.ops/s	2.67 k.ops/s	2.23 k.ops/s	1.55 k.ops/s
Div / Rem (/, %)	61.4 ops/s	38.4 ops/s	21.8 ops/s	9.6 ops/s	2.79 ops/s
Left / Right Shifts (<<, >>)	623 ops/s	581 ops/s	418 ops/s	213 ops/s	97.3 ops/s
Left / Right Rotations (left_rotate, right_rotate)	634 ops/s	586 ops/s	418 ops/s	213 ops/s	97.4 ops/s
Leading / Trailing zeros/ones	368 ops/s	355 ops/s	347 ops/s	308 ops/s	198 ops/s
Log2	409 ops/s	374 ops/s	364 ops/s	306 ops/s	199 ops/s
Select	863 ops/s	1.15 k.ops/s	1.06 k.ops/s	819 ops/s	392 ops/s

34

Upcoming in TFHE-rs

Improving Performance

More features & new types

Stable GPU backend

35

Zama's **Fhevm**

Onchain data is public by design, making it hard to build dapps that require confidentiality

Transactions | Internal Txns | Erc20 Token Txns | Erc721 Token Txns | Erc1155 Token Txns | Analytics | Comments

Latest 25 ERC-20 Token Transfer Events

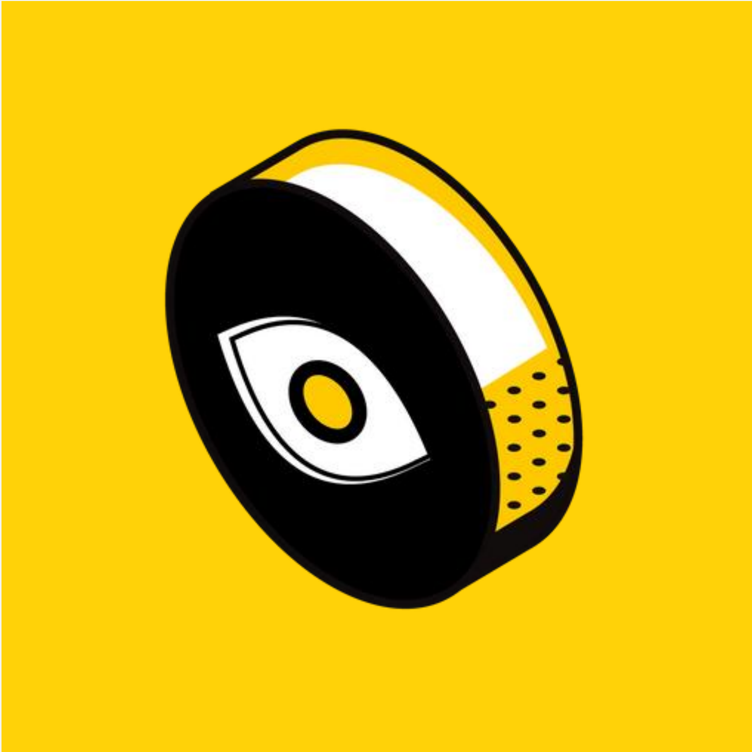
View All

Txn Hash	Age	From		To	Value	Token
0x61bac8ed64cf49ff537…	1 hr 8 mins ago	Uniswap V2: KCAL 2	IN	vitalik.eth	2,500	Step.app (KCAL)
0xd9f47a344e278579cb…	1 hr 15 mins ago	Justin Sun	IN	vitalik.eth	25,143,213.150843308745475521	Step.app (KCAL)
0xdea02c32d141997aaa…	12 hrs 57 mins ago	plamer.eth	IN	vitalik.eth	1	AssangeDAO (JUSTIC...)
0x74205c19a313ba8865…	1 day 11 hrs ago	Uniswap V2: SEGA 3	IN	vitalik.eth	227,158,544.808096280091774569	SEGA (SEGA)
0xad5c19e1af6de6508e…	2 days 20 hrs ago	0xad29c28a868c945caf9…	IN	vitalik.eth	21,420	ERC-20 (BASTAR...)
0x1014024546d2e94f39…	3 days 4 mins ago	Uniswap V2: ALIS 2	IN	vitalik.eth	153,473.76198500365822856	Acropolis DA... (ALIS)
0xbffdb2fcd52e96f136c7…	3 days 24 mins ago	vitalik.eth	OUT	OlympusDAO: DAO Funds	40,323.284453294043855726	Acropolis DA... (ALIS)
0x6ac57444413cd7bbef…	3 days 31 mins ago	Uniswap V2: ALIS 2	IN	vitalik.eth	40,323.284453294043855726	Acropolis DA... (ALIS)
0xb15136c85e15dd81b3…	3 days 1 hr ago	OlympusDAO: DAO Funds	IN	vitalik.eth	8,633.511805120159396357	Acropolis DA... (ALIS)
0xa9749c78f8ed9da996…	3 days 14 hrs ago	Uniswap V2: Bvlgari	IN	vitalik.eth	3,853,058,515,307.2989734036202684...	ERC-20 (Bvlgar...)
0x27fe35a36a42bbed75…	3 days 15 hrs ago	Uniswap V2: Bvlgari	IN	vitalik.eth	3,652,123,857,386.0501562459646840...	ERC-20 (Bvlgar...)
0x3880e037ae4833638f…	3 days 15 hrs ago	Uniswap V2: Bvlgari	IN	vitalik.eth	3,176,588,279,214.80176999868555856	ERC-20 (Bvlgar...)
0x7d18a354e603c74f3a…	3 days 16 hrs ago	Uniswap V2: Bvlgari	IN	vitalik.eth	2,784,937,897,903.6328859385267450...	ERC-20 (Bvlgar...)

This Leads to Many Privacy Issues

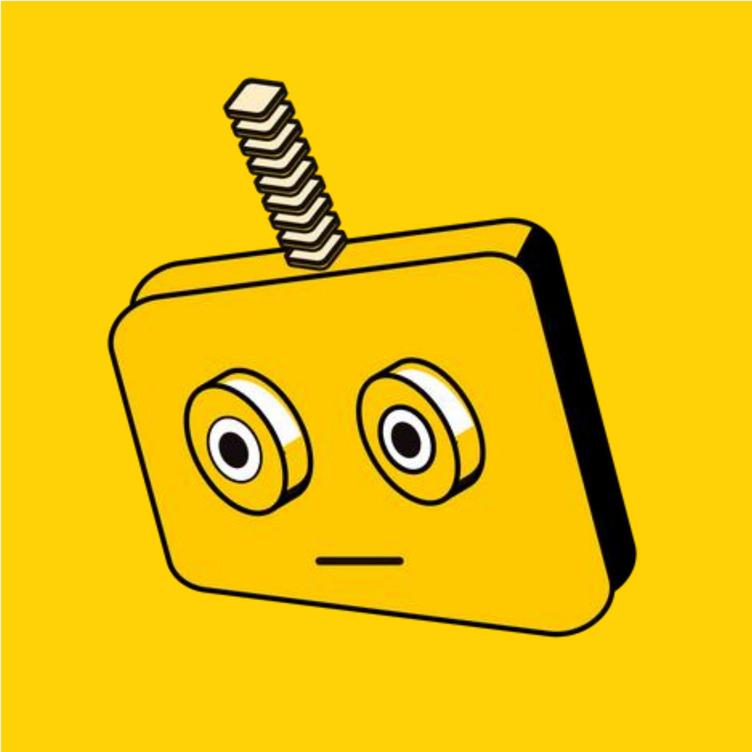
Theft

Criminals know what you own, so they can easily target you and steal your crypto.



Surveillance

Governments can surveil you, even if you use multiple addresses.



MEV

Bots can front-run you, creating a hidden tax on every transaction.

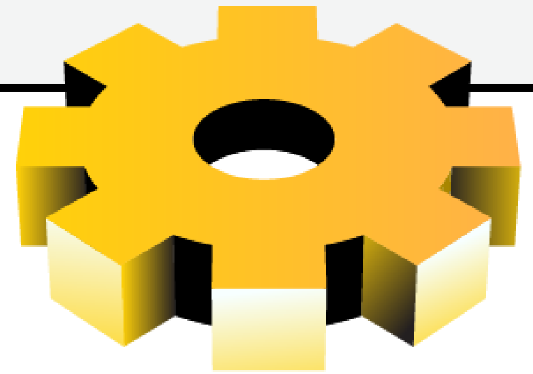
Privacy Preserving Computation

	FHE	ZK	MPC
Private	✓	✓	✓
Decentralized			✓
Composable	✓		✓
Non-interactive	✓	✓	

Zama's fhevm enables confidential smart contracts using FHE


E2E encryption of transactions and state


On-chain data availability


Compatible with existing dapps and state

Without Compromising Transparancy and Utility

Computation

Users can still know what contracts are doing.

Access Control

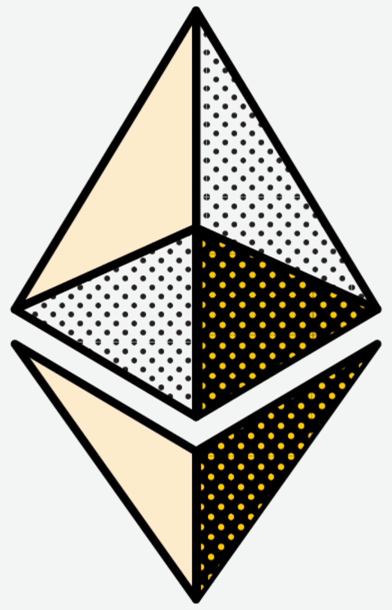
Contracts are free to implement their own access control logic.

Composability

It is easy to mix data from multiple users and compose smart contracts.

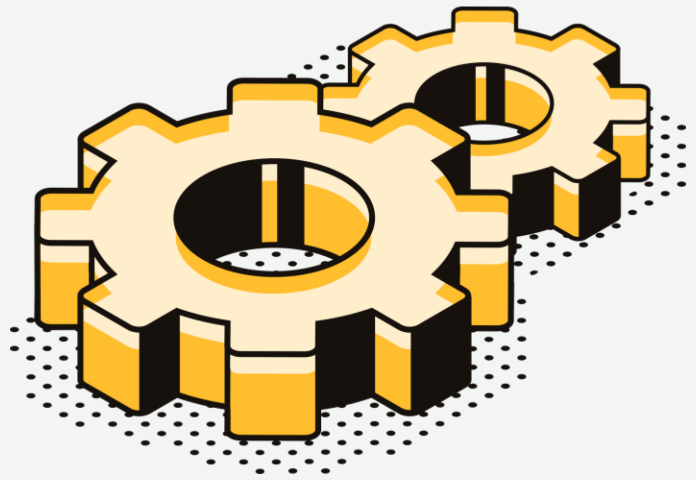
The Architecture

The Zama Protocol uses FHEVM to add confidentiality to smart contracts using a modular, offchain/onchain design, realizing MPC-as-a-Service (MaaS)



Host Chains

Where smart contracts run (e.g., Ethereum).



Coprocessors

Perform FHE computations offchain.

Gateway

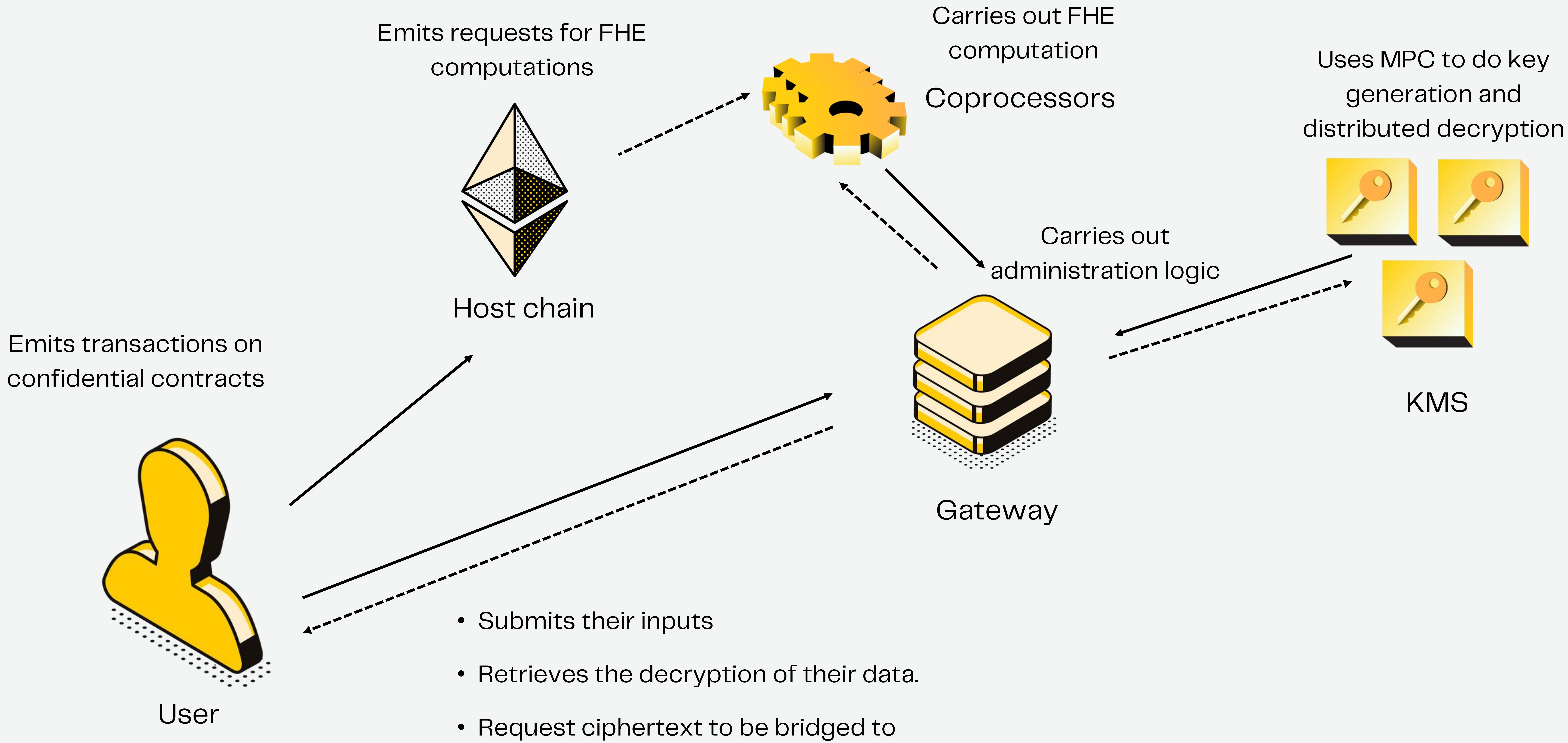
Coordinates all protocol activity.

KMS

Manages secret keys using MPC.

42

Gateway, the API of the Protocol



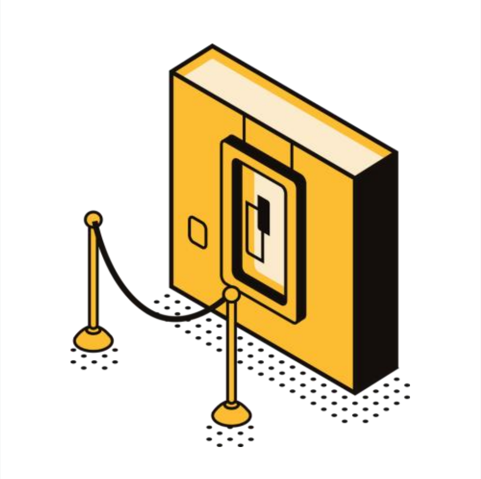
- Submits their inputs
- Retrieves the decryption of their data.
- Request ciphertext to be bridged to another chain.

43

Zama's fhevm unlocks new use cases

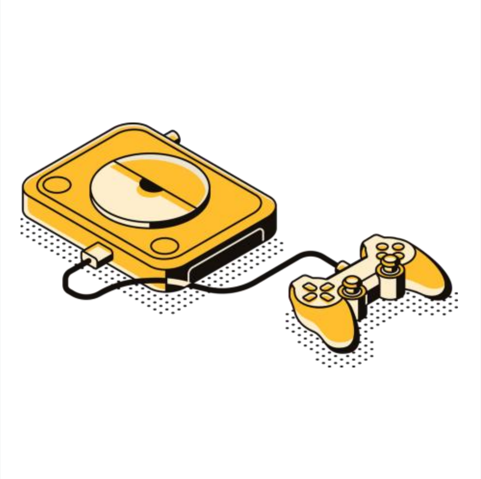
Tokenization

Manage and swap tokenized assets without other seeing it



Blind Auctions

Bid on items without revealing the amount or the winner

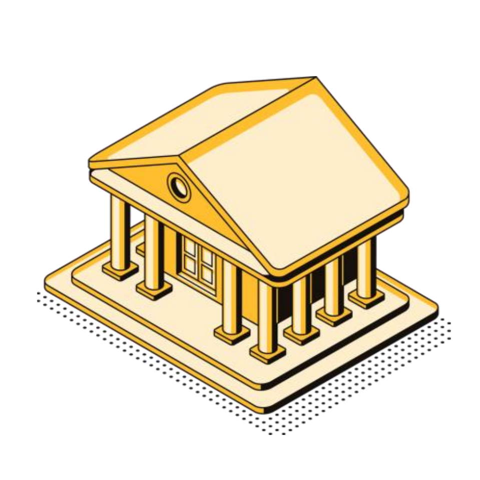


Onchain Games

Hide cards and moves until reveal (e.g. poker, blackjack, ..)

Encrypted DIDs

Store identities on-chain and generate attestations without ZK



Private Transfers

Keep balances and amounts private, without using mixers

44

Technical Overview

Zama's fhevm combines state of the art cryptography in a provably secure way

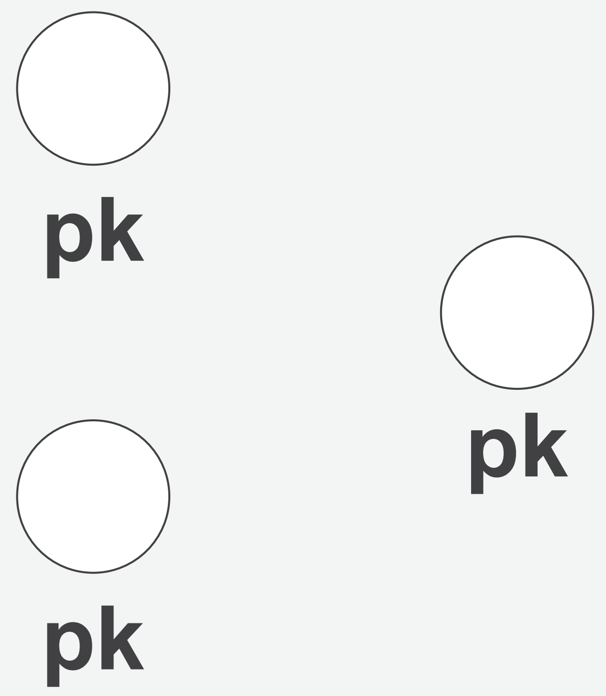
FHE	MPC	ZK
Homomorphic encryption is used to compute on private state, directly on-chain	Multi-party computation is used for threshold key generation and decryption of FHE ciphertexts	Zero-Knowledge Proofs of Knowledge are used to ensure encryption and decryption integrity

46

Everything is encrypted under a single global FHE public key

fhevm Blockchain

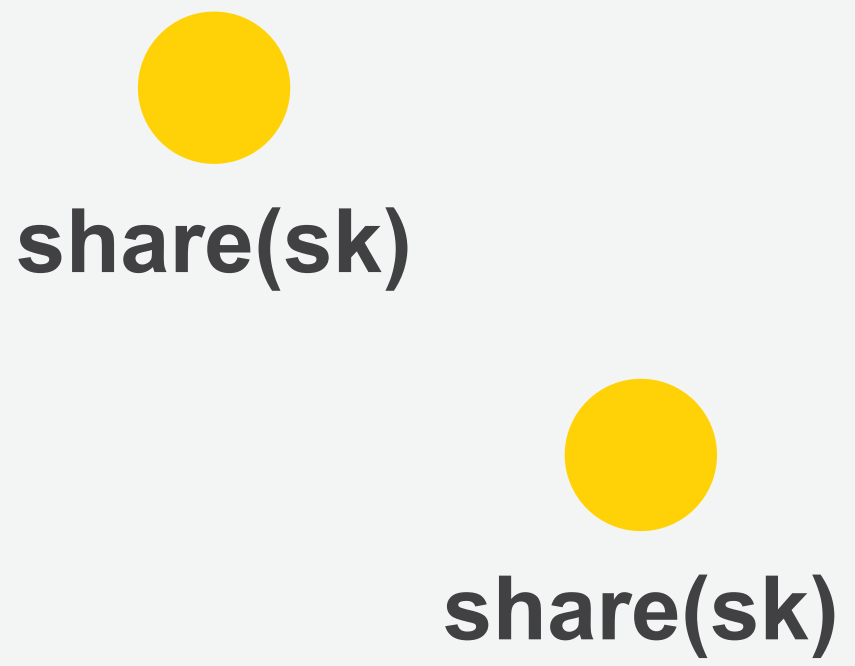
pk



PoS consensus

KMS

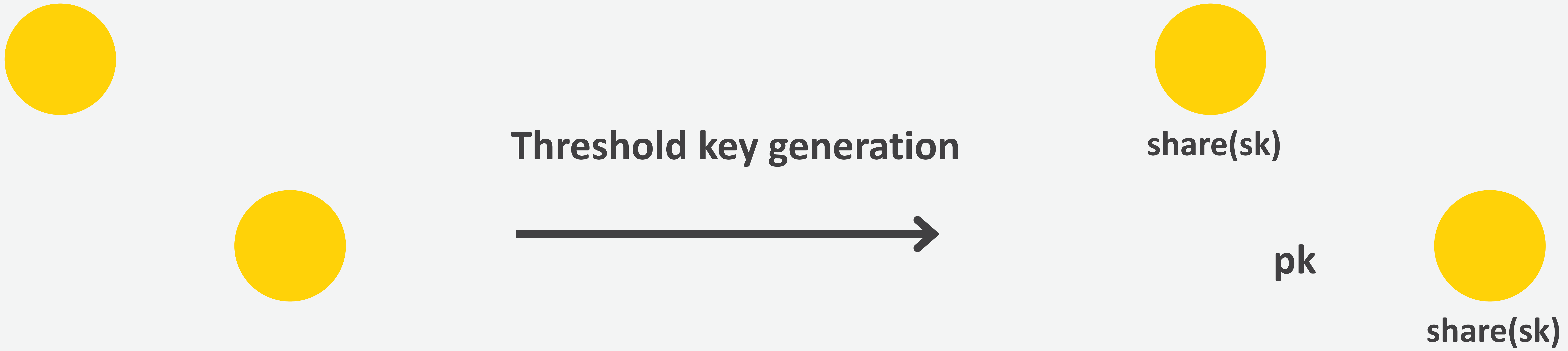
sk



MPC threshold

47

The global key is generated securely using a threshold protocol

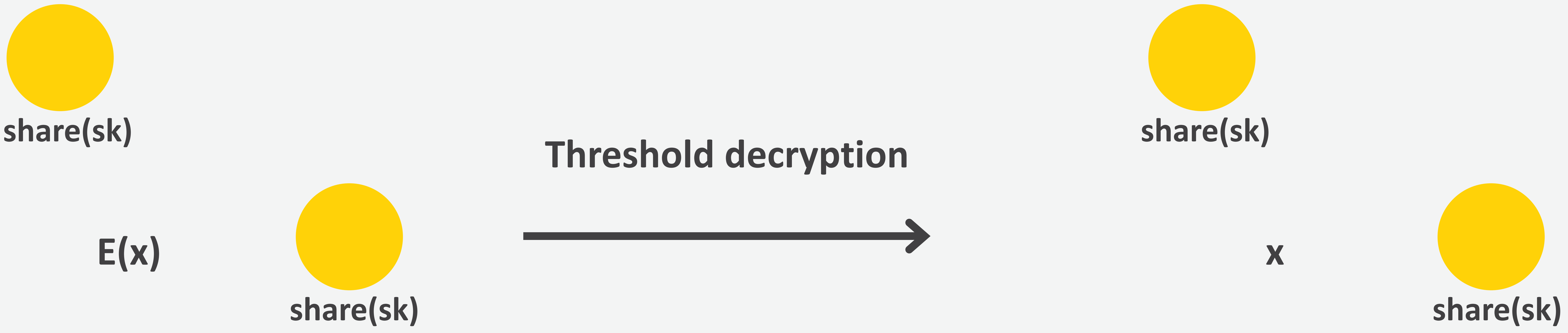


48

The inputs are simply encrypted using the global public FHE key

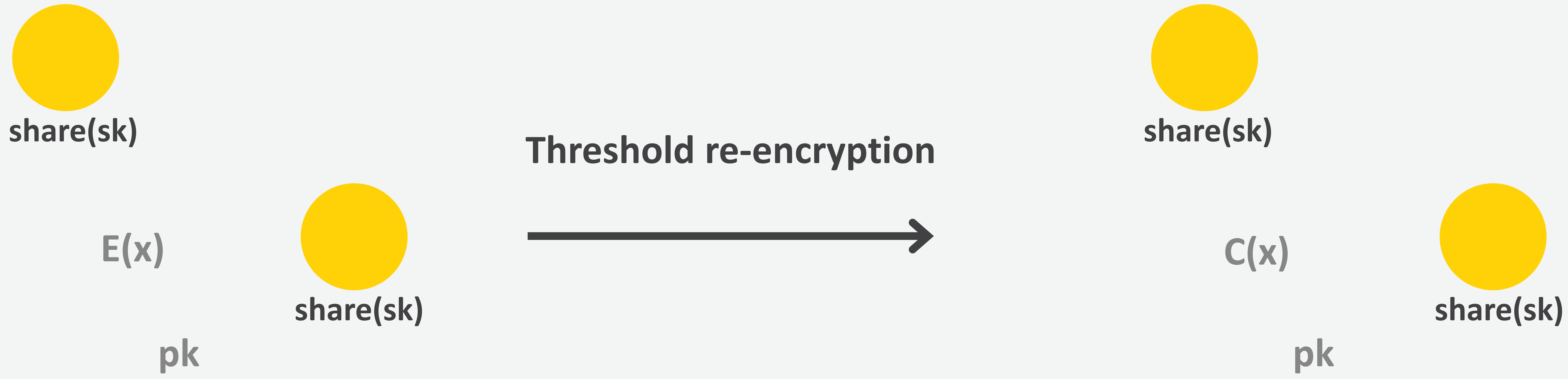
49

Values can be publicly decrypted by validators using a threshold protocol



50

Values can also be re-encrypted to a user public key using a threshold protocol



51

Re-encrypted values can be read and decrypted only by the user owning the key

52

Solidity API

Developers can write confidential smart contracts without learning cryptography

```solidity
contract ConfidentialERC20 {
    // A mapping from address to an encrypted balance
    mapping(address => euint64) internal balances;

    // transfer an encrypted amount
    function transfer(
        address from, address to, externalEuint64 encryptedAmount, bytes calldata inputProof
    ) public virtual returns (bool) {
        // Verify the input is correct
        euint64 amount = FHE.fromExternal(encryptedAmount, inputProof);

        // Check if sender has enough funds and determine transferValue, i.e. the amount to send
        ebool hasFunds = FHE.le(amount, balances[from]);
        euint64 transferValue = FHE.select(hasFunds, amount, FHE.asEuint64(0));

        // Add to transferValue to `to` and subtract from `from`.
        euint64 newBalanceTo = FHE.add(balances[to], transferValue);
        balances[to] = newBalanceTo;
        euint64 newBalanceFrom = FHE.sub(balances[from], transferValue);
        balances[from] = newBalanceFrom;

        // Allow users to see their balances
        FHE.allowThis(newBalanceTo);
        FHE.allow(newBalanceTo, to);
        FHE.allowThis(newBalanceFrom);
        FHE.allow(newBalanceFrom, from);

        return true;
    }
}
```

Solidity Integration

fhevm contracts are simple solidity contracts that are built using traditional solidity toolchains.

Simple DevX

Developers can use the euint data types to mark which part of their contracts should be private.

Programmable Privacy

All the logic for access control of encrypted states is defined by developers in their smart contracts.

54

FHE::euint

```
// A mapping from address to an encrypted balance
mapping(address => euint64) internal balances;
```

- Represents an encrypted value.
- Can be used for computation, storage, composition, etc.
- Efficient since they are small (only handles to ciphertexts).
- euint8, euint16, euint32, euint64, … add, sub, mul, eq, le, gt, …

55

FHE.fromExternal

```
// Verify the input is correct
euint64 amount = FHE.fromExternal(encryptedAmount, inputProof);
```

- Well-formed to not leak anything about global FHE secret key.
- Prevent user from decrypting arbitrary ciphertexts.
- Ciphertexts include ZK proof of plaintext knowledge that must be checked.

56

FHE.requestDecryption

```
function decryptUint64(euint64 value) public {
    bytes32[] memory cts = new bytes32[](1);
    // cast to bytes
    cts[0] = FHE.toBytes32(value);
    // request decryption
    FHE.requestDecryption(cts, this.callbackUint64.selector);
}

function callbackUint64(
    uint256 requestID,
    uint64 decryptedInput,
    bytes[] memory signatures
) public returns (uint64) {
    // validate plaintext signatures
    FHE.checkSignatures(requestID, signatures);
    // if valid, return decryption
    return decryptedInput;
}
```

- Public decryption
- Can be used in require statements
- Leaks value, even when just used for checking requirements.
- Alternative is FHE.select(eCondition, eTrueValue, eFalseValue).

57

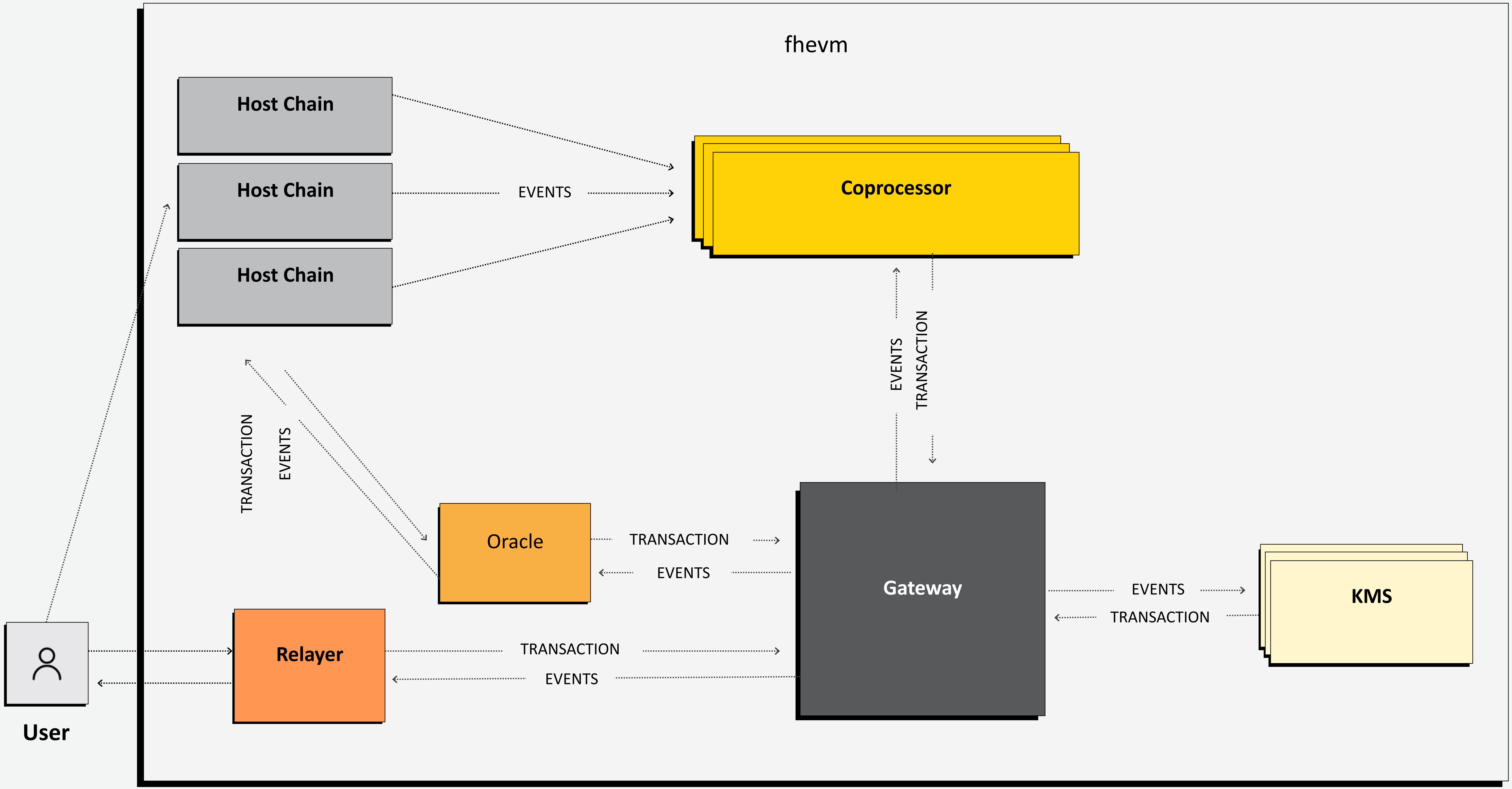
Getting Things Done

The fhevm protocol will enable the support for writing smart contracts on SolanaVM, CosmWasm, and more.

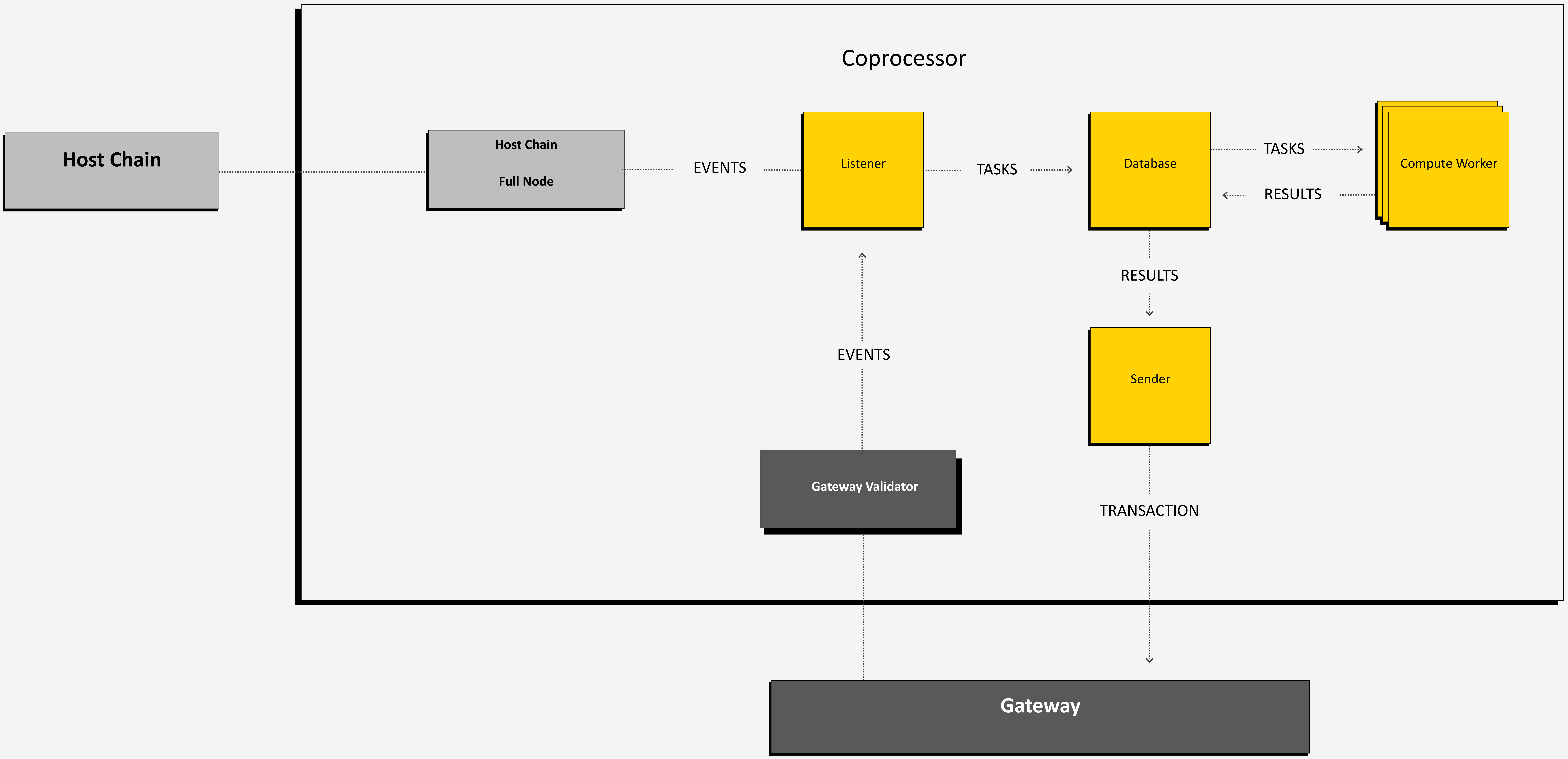
Writing smart contracts in Solidity will be just one way to do it.

58

Inside the **Fhevm**

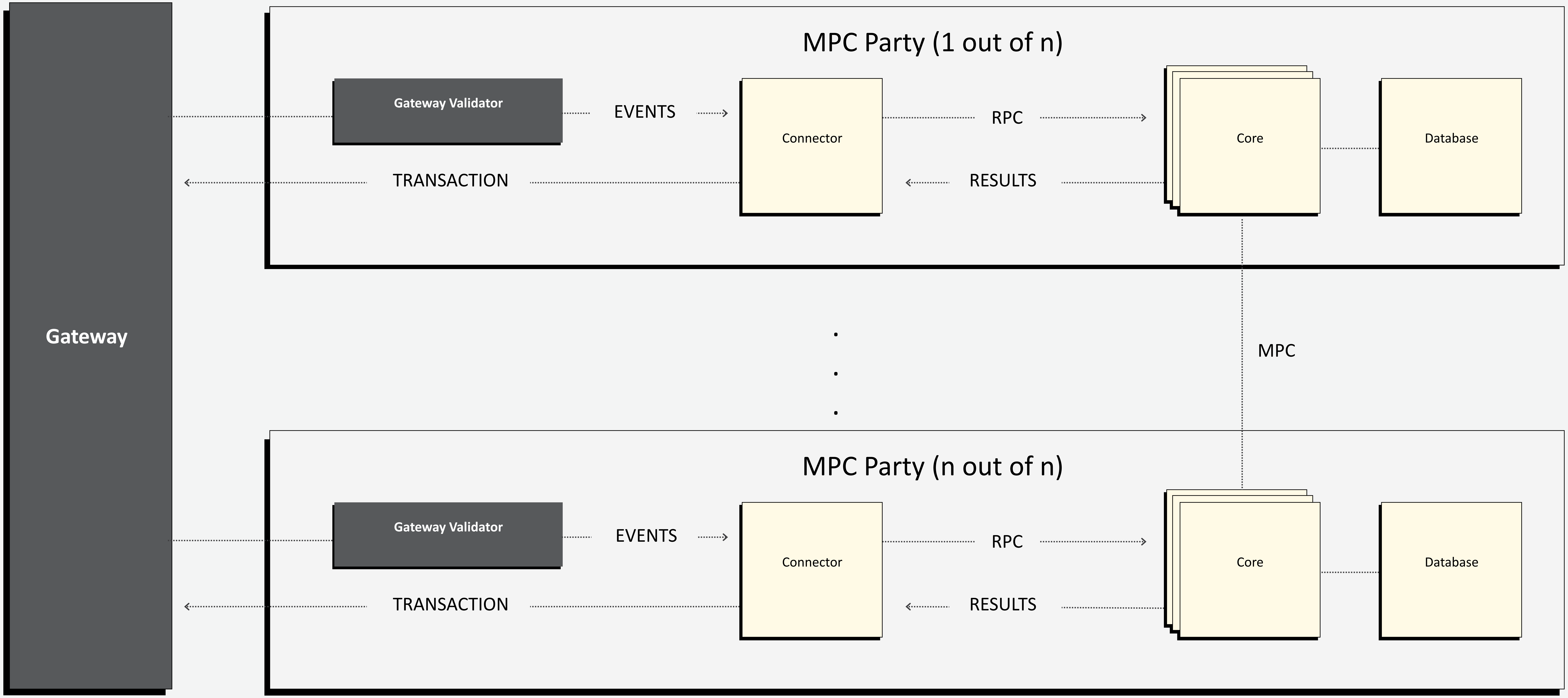

fhevm
Host Chain
Host Chain
Host Chain
EVENTS
Coprocessor
EVENTS
TRANSACTION
TRANSACTION
EVENTS
Oracle
TRANSACTION
EVENTS
Gateway
EVENTS
TRANSACTION
KMS
Relayer
TRANSACTION
EVENTS
User

60


Coprocessor
Host Chain
Host Chain
Full Node
EVENTS
Listener
TASKS
Database
TASKS
RESULTS
Compute Worker
RESULTS
EVENTS
Sender
Gateway Validator
TRANSACTION
Gateway

61

MPC Enabled **KMS** Backend


Gateway
MPC Party (1 out of n)
Gateway Validator
EVENTS
Connector
RPC
Core
Database
TRANSACTION
RESULTS
MPC
MPC Party (n out of n)
Gateway Validator
EVENTS
Connector
RPC
Core
Database
TRANSACTION
RESULTS

MPC Technique Overview

Secret Sharing over Rings

TFHE modulus q is 2^64.

Need for secret sharing mod q.

Finite field arithmetic replaced by a Galois Ring.

Switch-n-Squash

Switch from $(q, l) \rightarrow (Q, L)$, including a bootstrap to squash the noise.

q=2^64 → 2^128

Noise Flooding

Flooding with improved statistical distance argument using two uniform random flooding values and robust open.

Security Proofs

Simulation-based proofs.

128-bit security.

Details in the Noah's Ark paper.

Performance

Threshold decryption in <500ms, even for 40 parties in WAN.

64

Thank you

ZAMA

CONTACT

nigel@zama.org

zama.org

github.com/zama-ai

zama.org/community