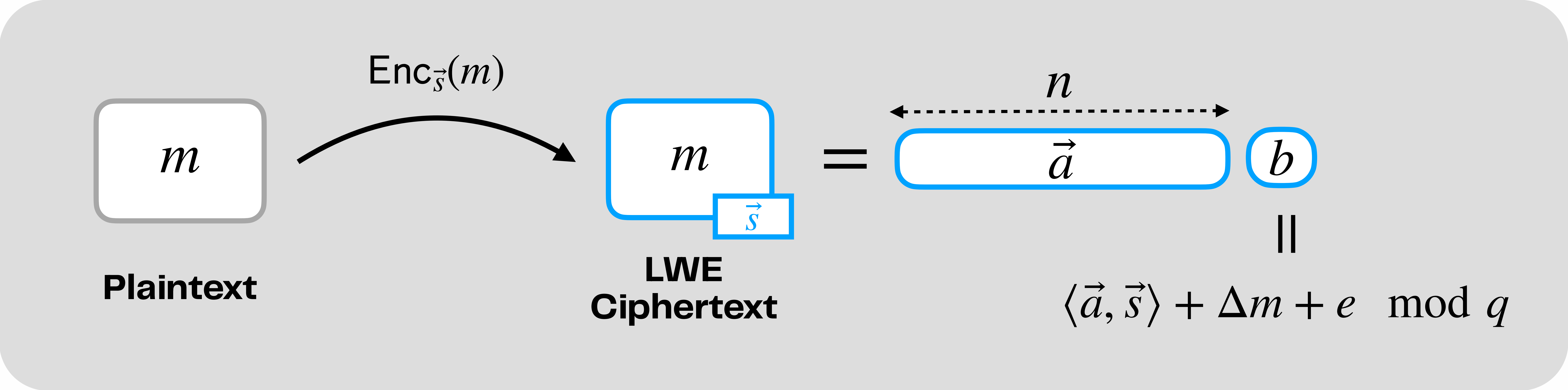
Chillotti I., El Kazadi S., Ligier D., Meyer A., Montaigu T., **Orfila J.-B.**, Tap S.

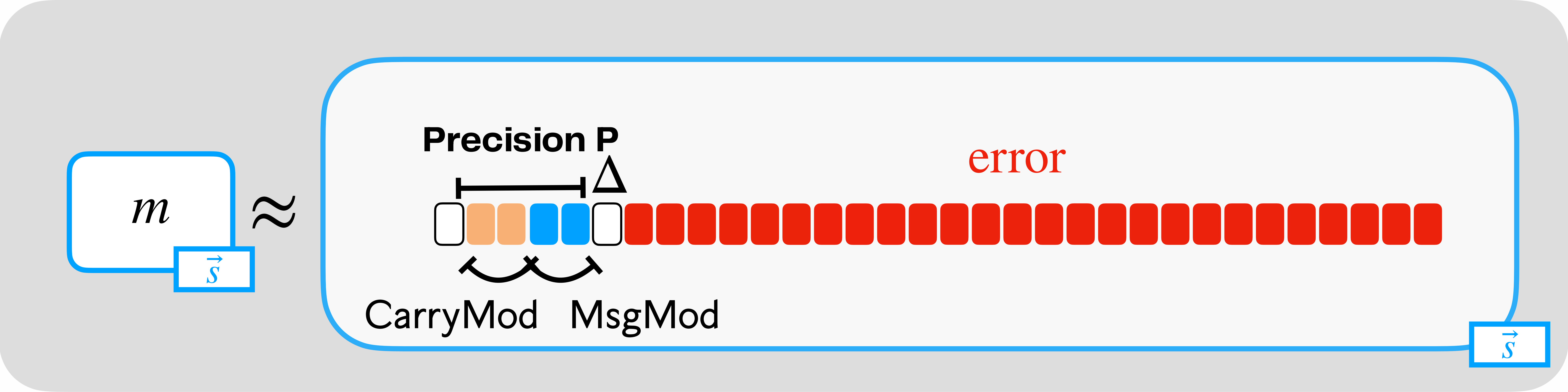
March 2024

TFHE Simplified: A practical Guide to Integer Arithmetic and Reliability

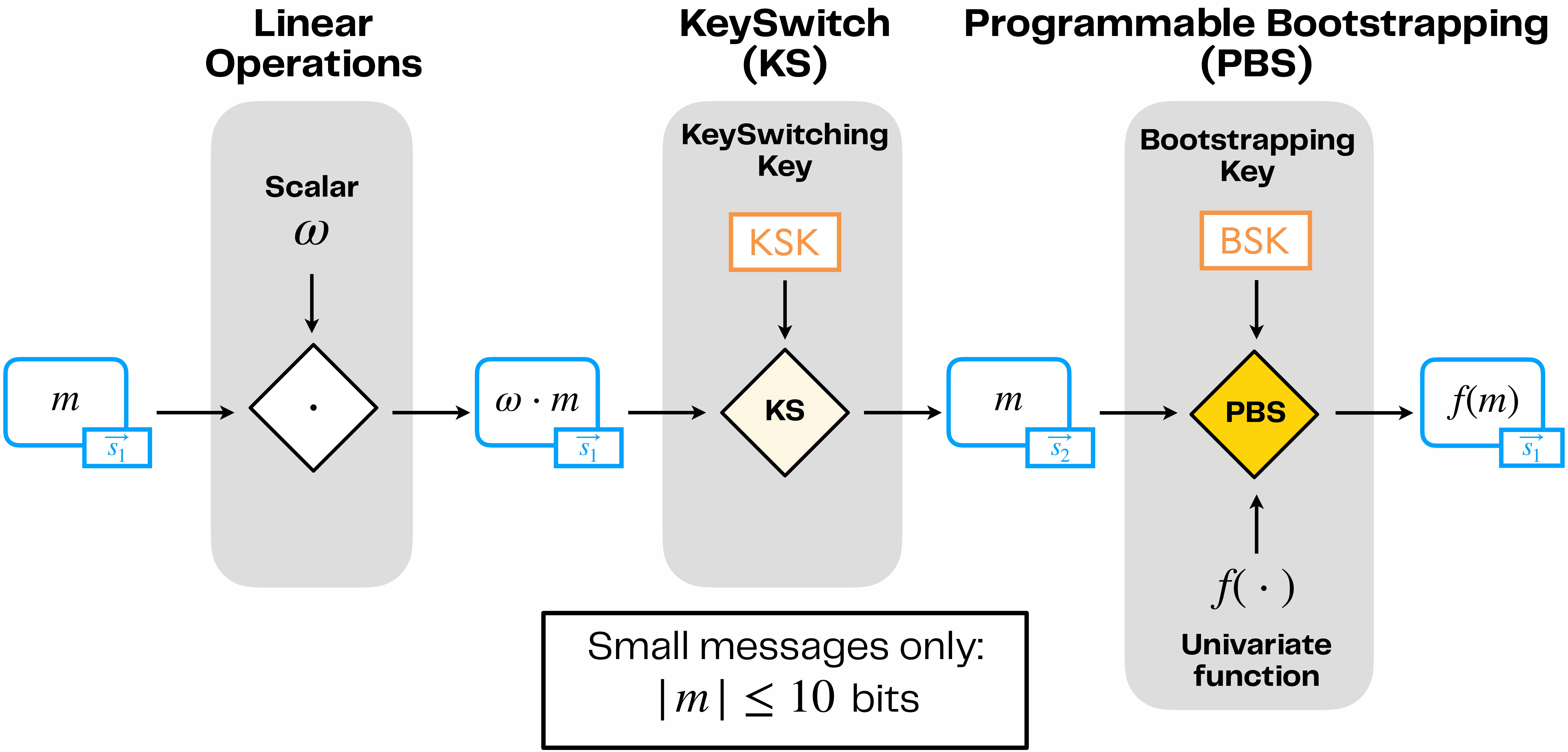
ZAMA

TFHE Ciphertexts



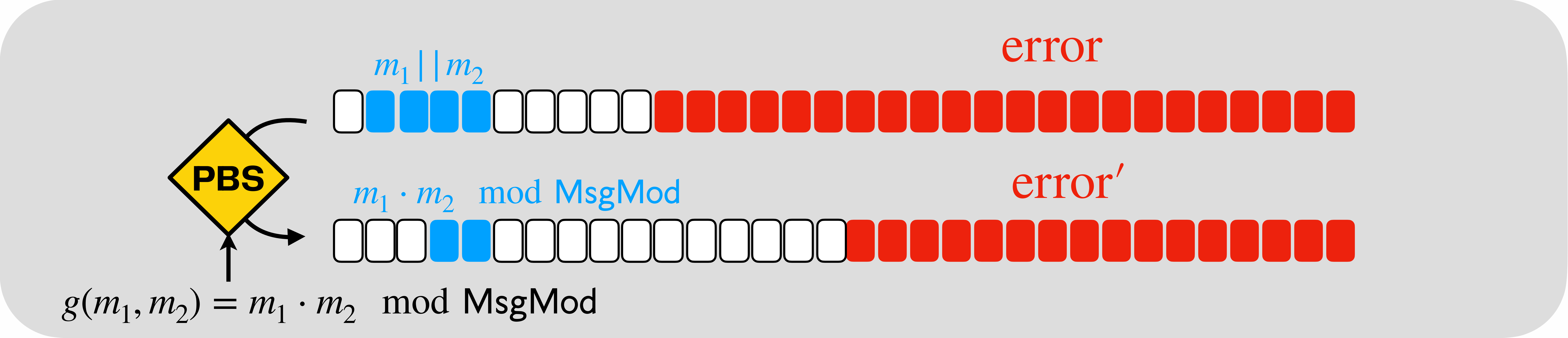


TFHE Toolbox 1/2



TFHE Toolbox 2/2

Bivariate PBS: Multiplication Example



Splitting cleartext space in two parts:

$$f(m) = g(m_1, m_2)$$

$\Rightarrow$

Constraint:

$$\mathsf{MsgMod} \leq \mathsf{CarryMod}$$

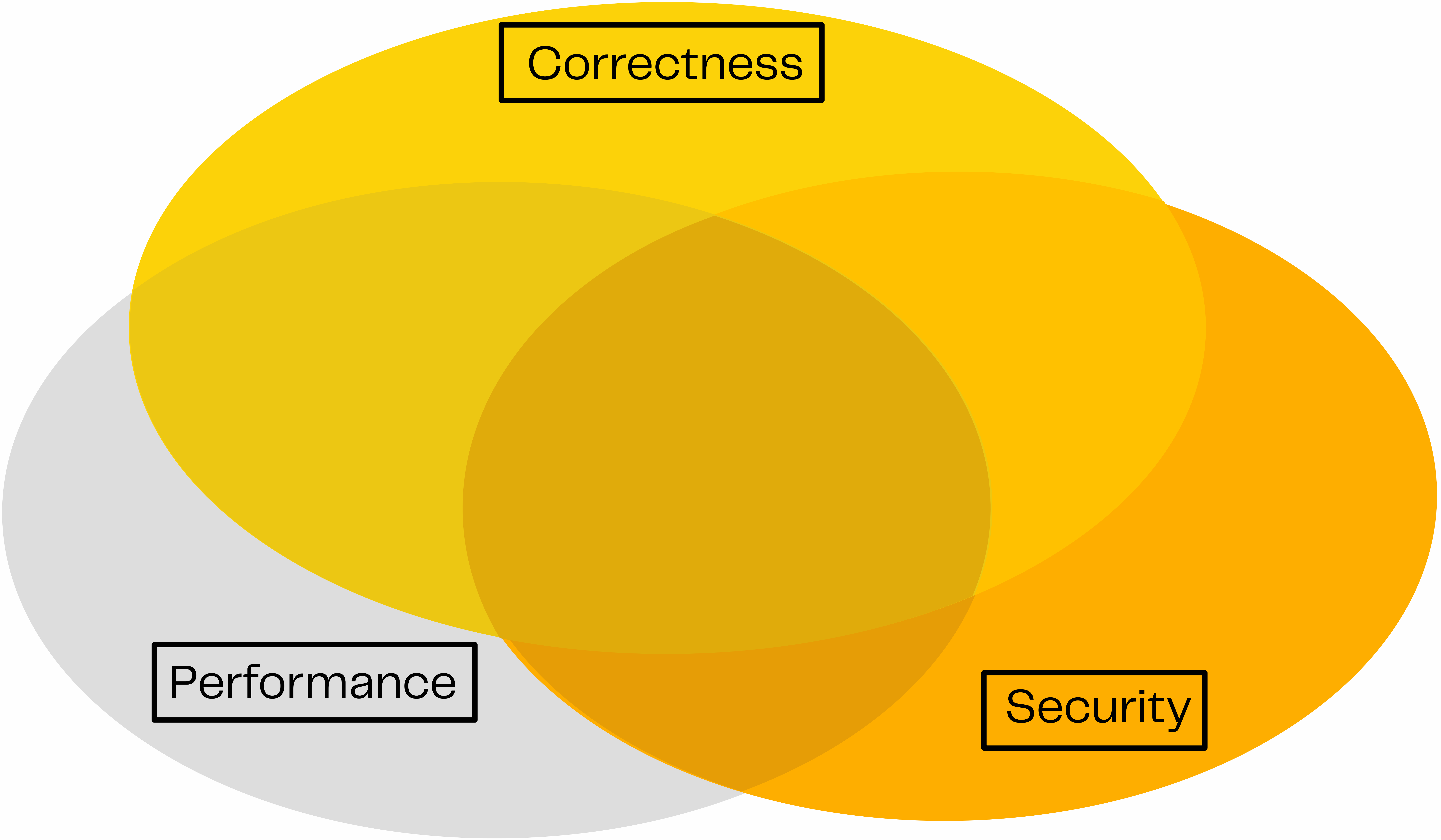
A few parameters...

Correctness

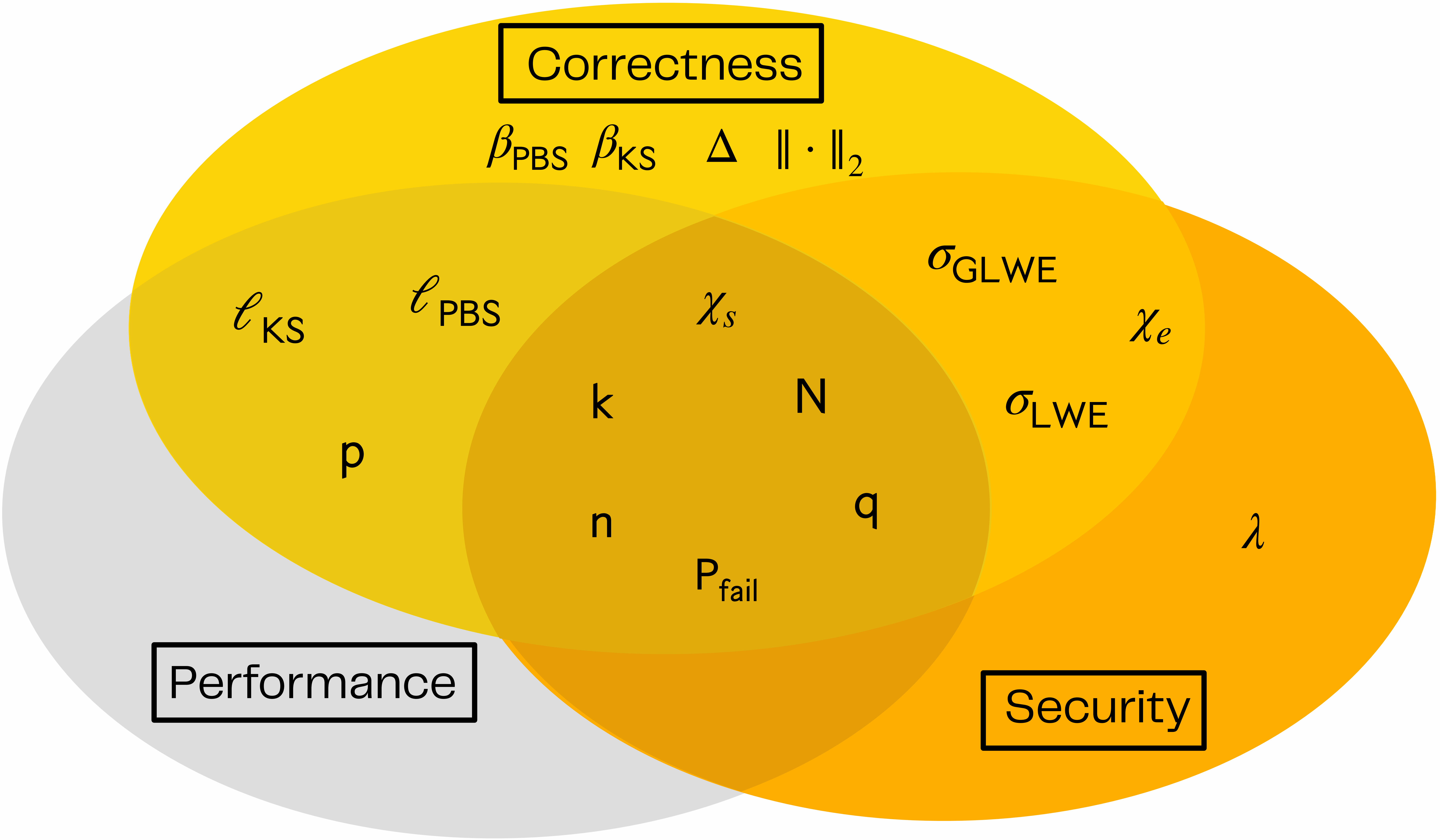
Performance

Security

A few parameters...



A few parameters...



Parameter Oracle

$\mathscr{C}$lient

Message precision P

Linear operation number $\|\cdot\|_2$

Error Probability P_{fail}

Security Level λ

1.

2.

$\mathscr{O}$racle*

β_{KS} ℓ_{KS} β_{PBS} ℓ_{PBS}

n k N

σ_{LWE} σ_{GLWE}

χ_s χ_e

Δ q

*Parameter optimization and larger precision for (T) FHE, Bergerat, Boudi, Bourgerie, Chillotti, Ligier, Orfila and Tap, Journal of Cryptology (2023)

FHE constraints

Security, Correctness and Performance should be **assured**

Cryptographic parameters must be **correctly** chosen for the users

By default, precision of messages ≤ 10 bits

Extension needed to support u16, u32, u64, ...

How can we abstract **FHE programming** from its cryptographic complexity to match the simplicity of **traditional coding**?

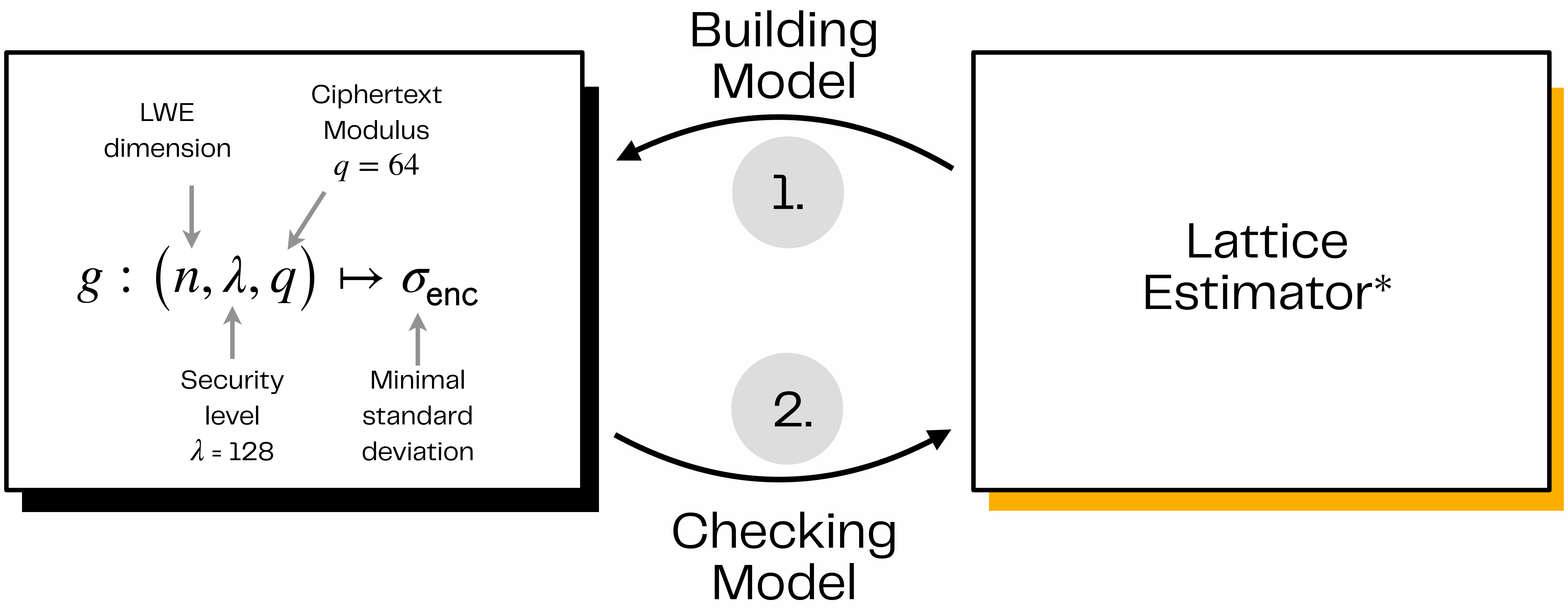
Summary

- A Foolproof Parameter Oracle
- Building Homomorphic Integers
- Benchmarking FHE

A Foolproof Parameter Oracle

FHE Requirements

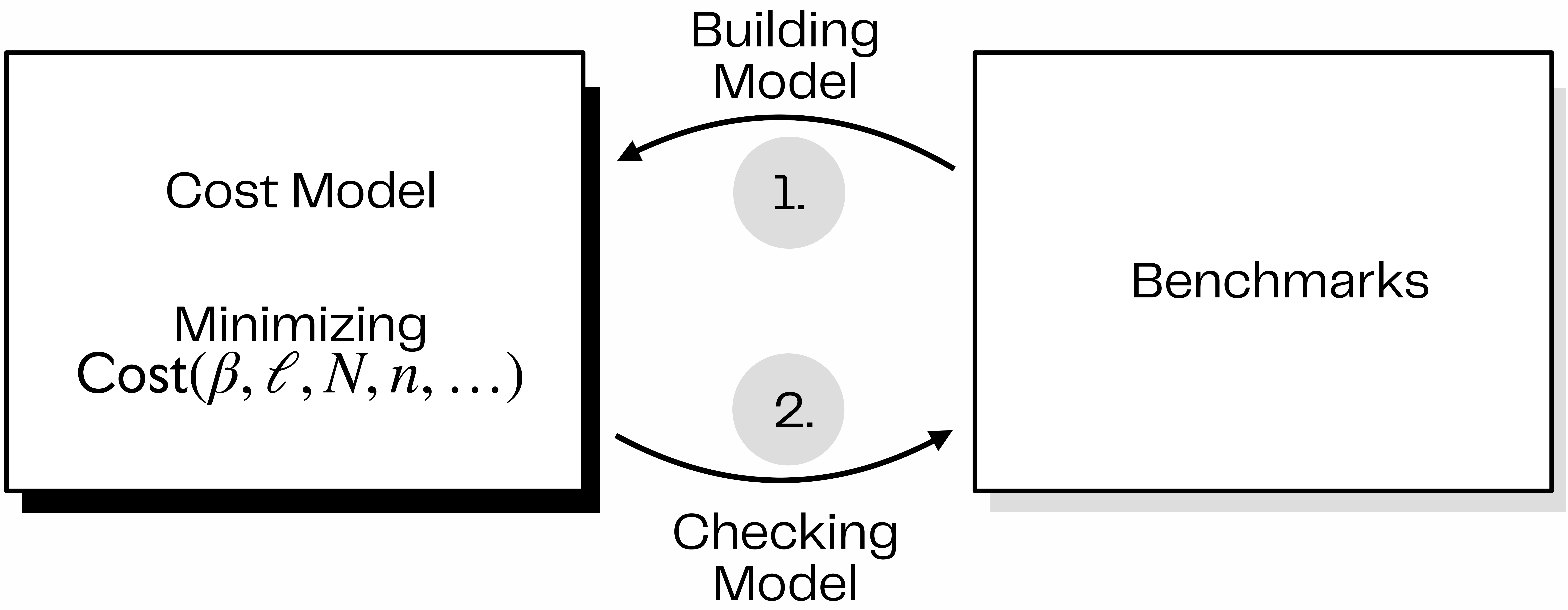
Security | Performance | Correctness



*https://github.com/malb/lattice-estimator

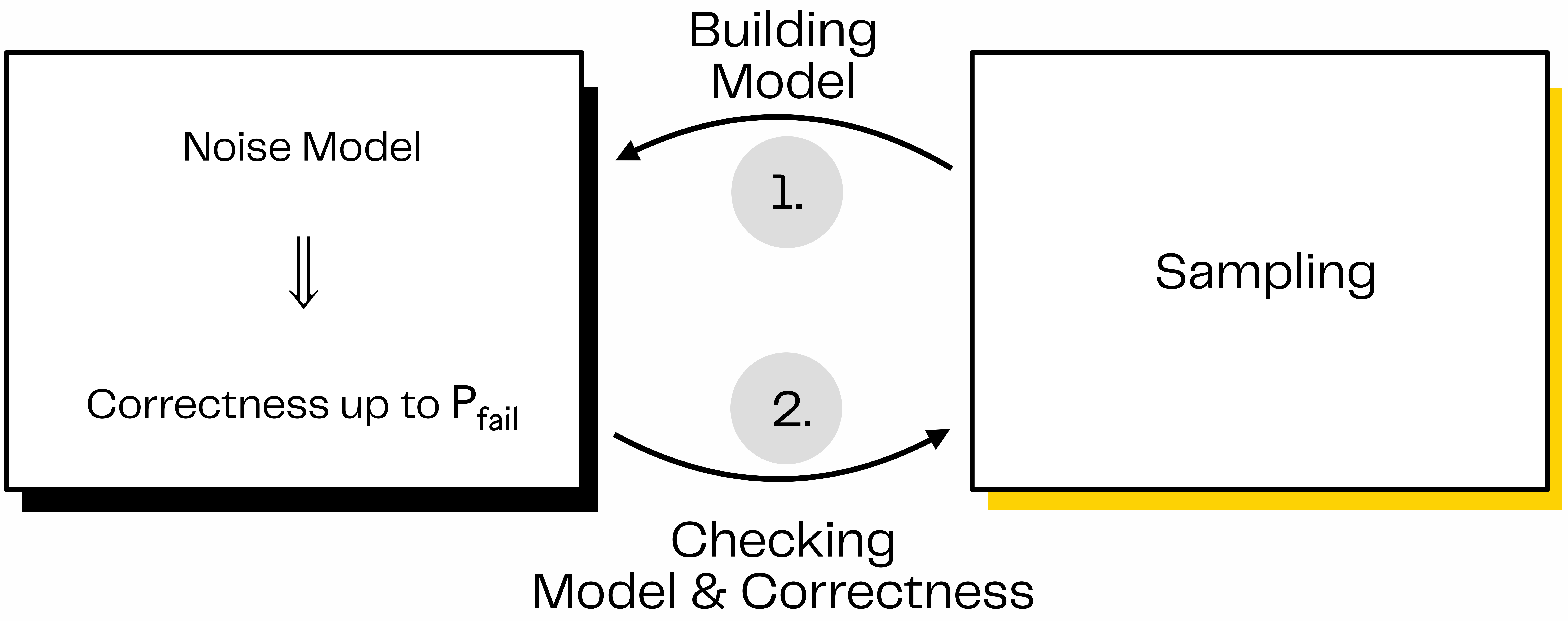
FHE Requirements

Security | **Performance** | Correctness



FHE Requirements

Security | Performance | Correctness



Checking Noise Model

<u>Hypothesis</u>: Gaussian noise distribution

<u>CLT</u>: Noise approximated by a Gaussian distribution

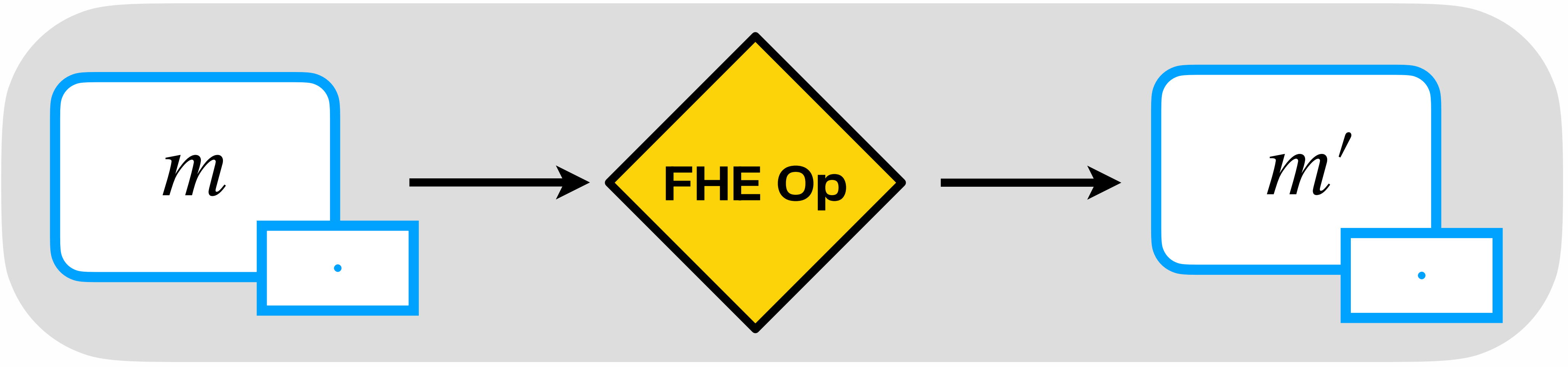
$\Longrightarrow$ Noise is fully described by its **variance** and **expectation**

1. **Normality check** Shapiro-Francia* test

2. **Variance check** Check that $\mathsf{Var}_{\mathsf{exp}} \in \left[\frac{1}{2} \cdot \mathsf{Var}_{\mathsf{th}}, \mathsf{Var}_{\mathsf{th}}\right]$

*An approximate analysis of variance test for normality, S. S. Shapiro and R. S. Francia, Journal of the American Statistical Association, 67(337):215–216, 197

Error Probability



Correct evaluation with probability $1 - \mathrm{P}_{\text{fail}}$

Ex: $\mathrm{P}_{\text{fail}} = 2^{-40}$ $\Longrightarrow$ On average, 2^{41} executions to observe **one** failure

Rare event

Conditional Probability Basic Property

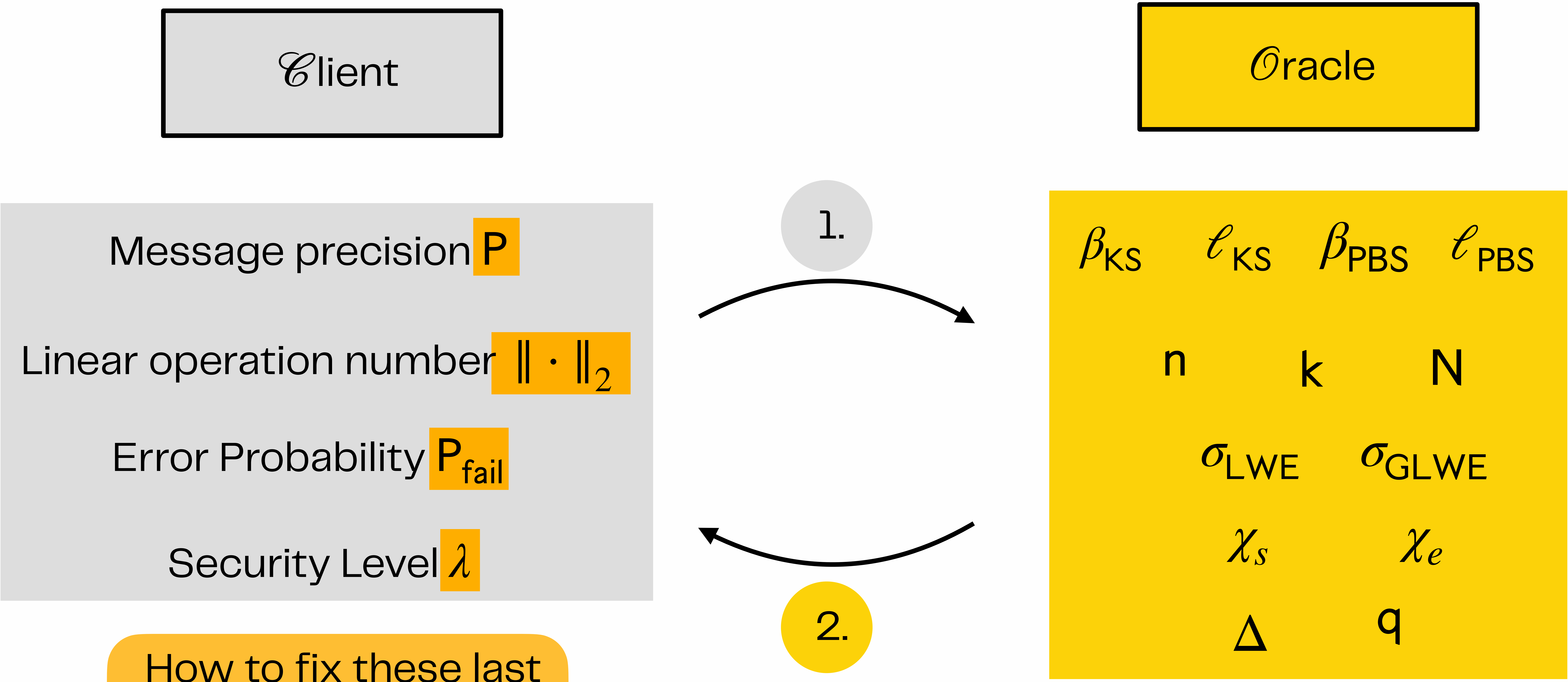
Importance Splitting*

For $\tau_0 < \tau_1 < \cdots < \tau_{J-1}$, $\Pr\left[|e| > \frac{\Delta}{2}\right] = \Pr\left[|e| > \frac{\Delta}{2} \mid |e| > \tau_{J-1}\right] \cdot \ldots \cdot \Pr\left[|e| > \tau_1 \mid |e| > \tau_0\right] \Pr\left[|e| > \tau_0\right]$

$\gg p_{\text{fail}}$ $\gg p_{\text{fail}}$ $\gg p_{\text{fail}}$

Estimation of small failure probabilities in high dimensions by subset simulation, Siu-Kui Au, James L. Beck, Probabilistic Engineering Mechanics 16 (2001) 263-277

Parameter Oracle Bis



How to fix these last parameters?

Fixing the last parameters

Failure Probability P_{fail}

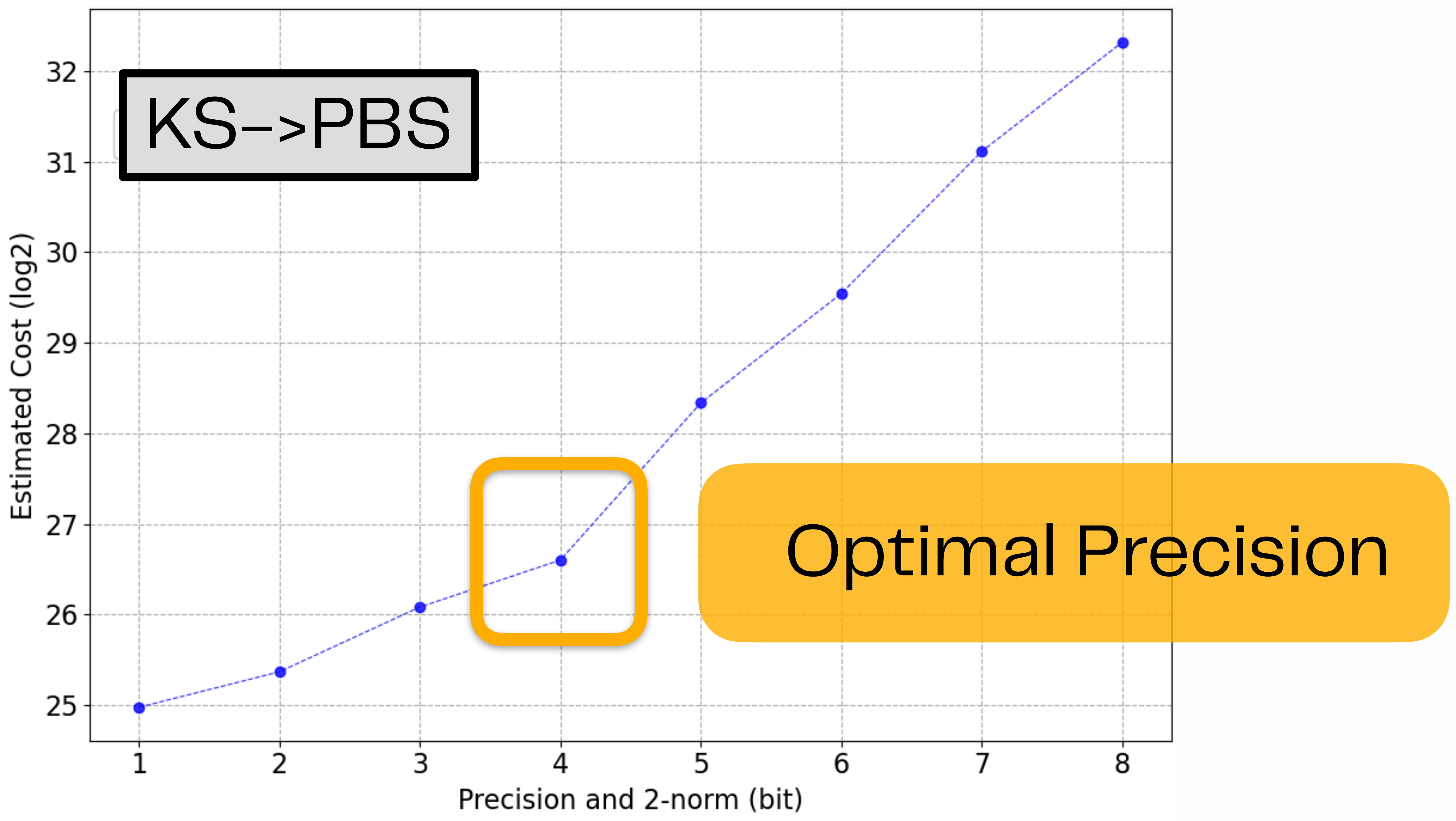
$P_{fail} \leq 2^{-40} \iff$ KS→PBS is correct 99.9999999991%

Linear Ops $\|\cdot\|_2$

$\text{MsgMod}, \text{CarryMod} \Rightarrow \|\cdot\|_2 = \left\lfloor \frac{\text{MsgMod} \cdot \text{CarryMod} - 1}{\text{MsgMod} - 1} \right\rfloor$

Security λ

Default value: $\lambda = 128$

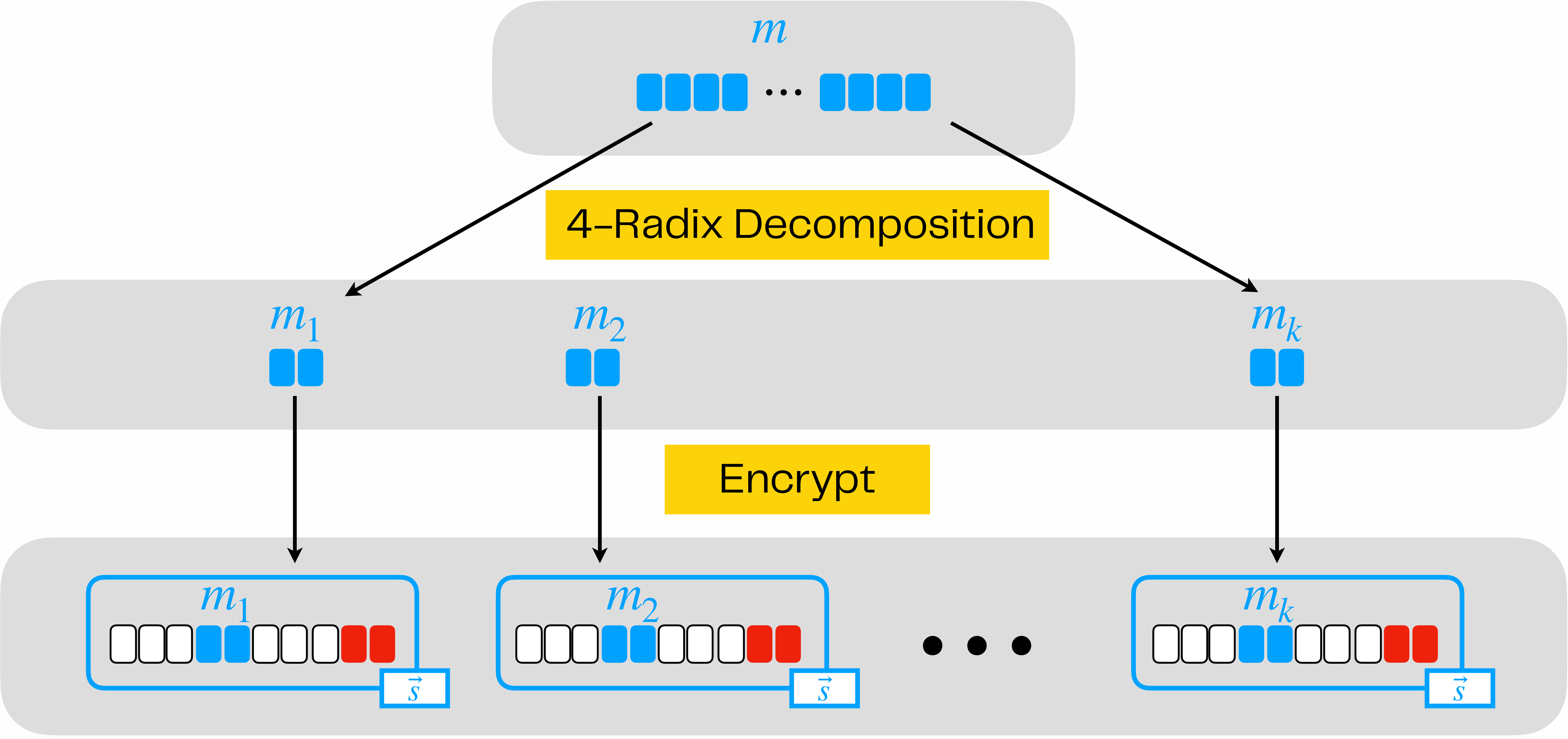


Precision P

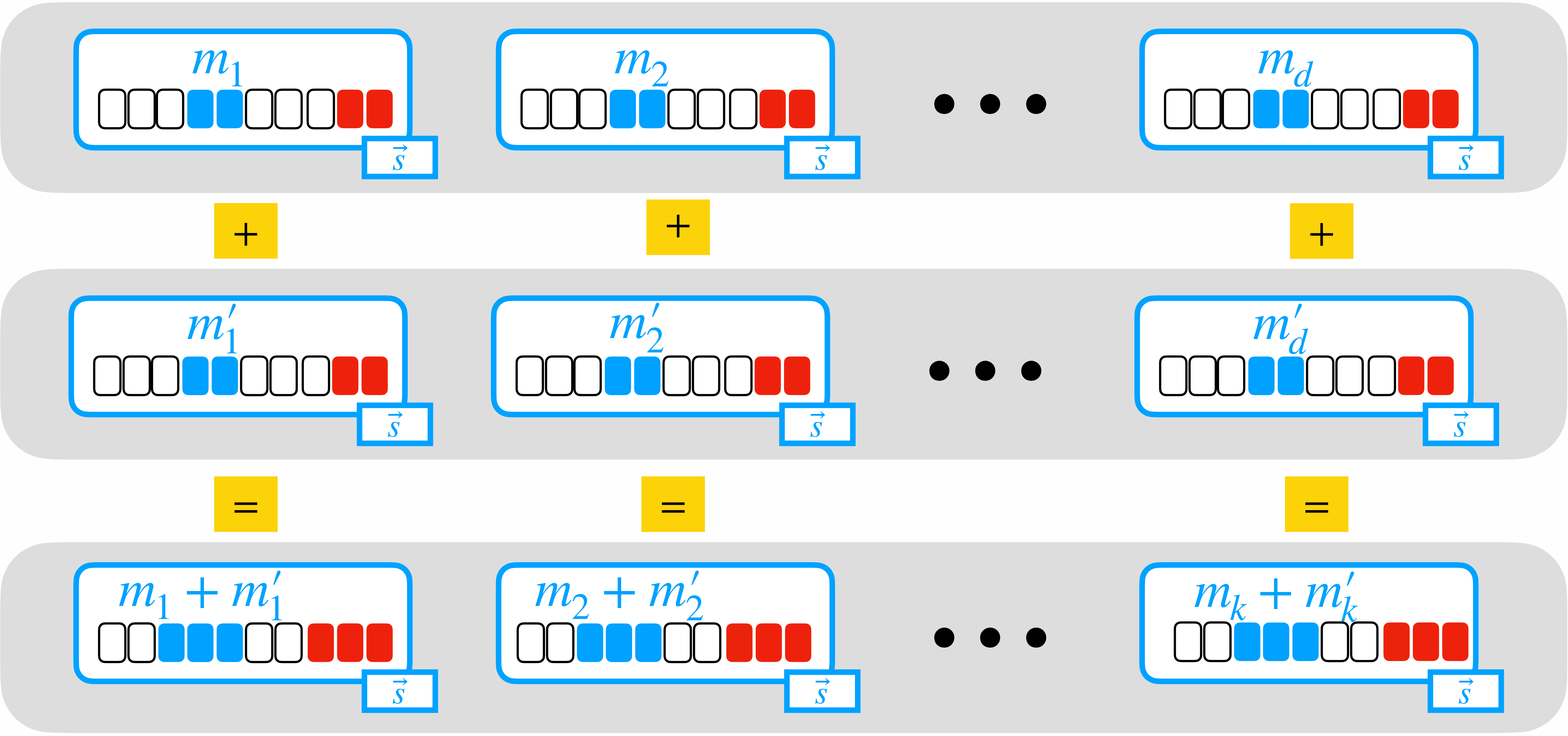
$\text{MsgMod} = 2^2$ & $\text{CarryMod} = 2^2$

Building Homomorphic Integers

Integers

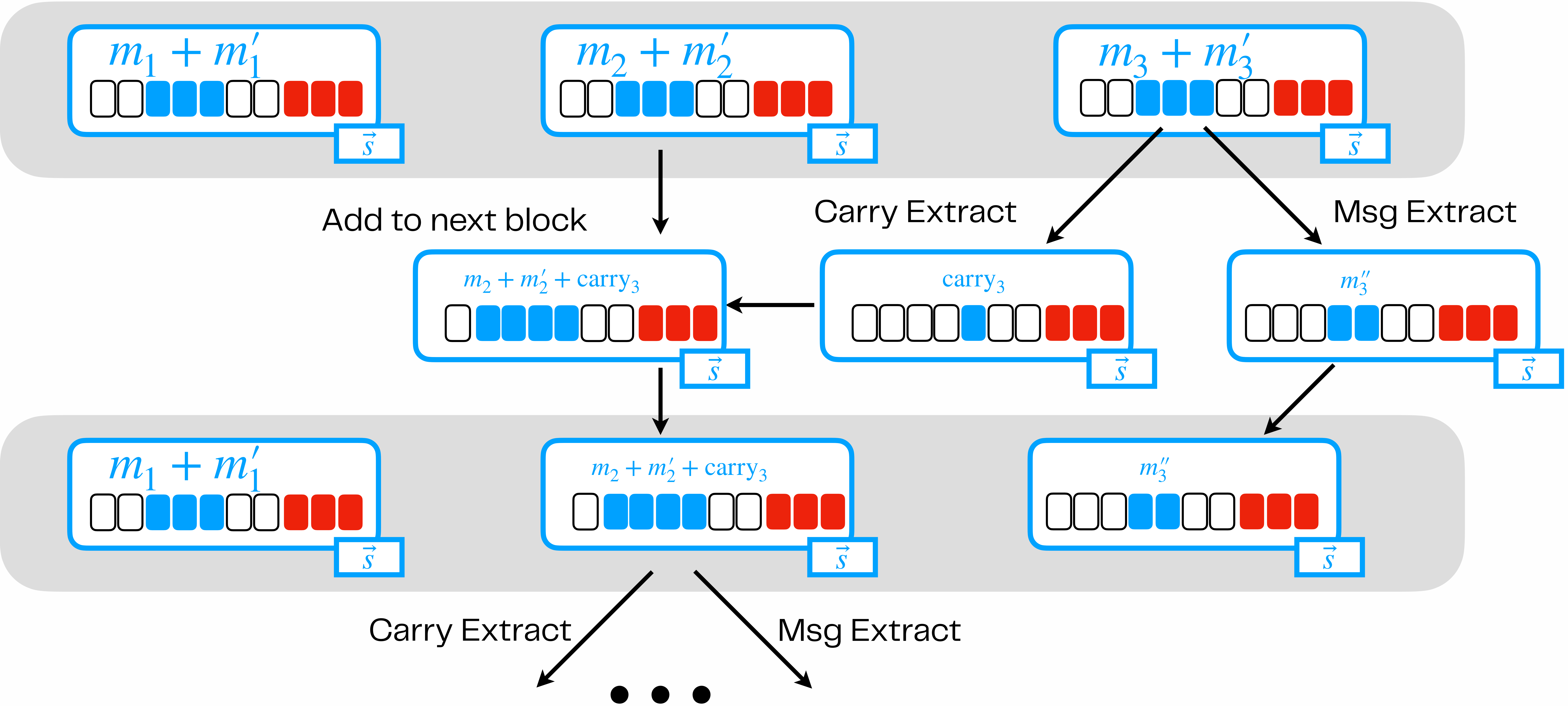


Addition



Output Encoding ≠ Input Encoding ⇒ Needs to propagate carries

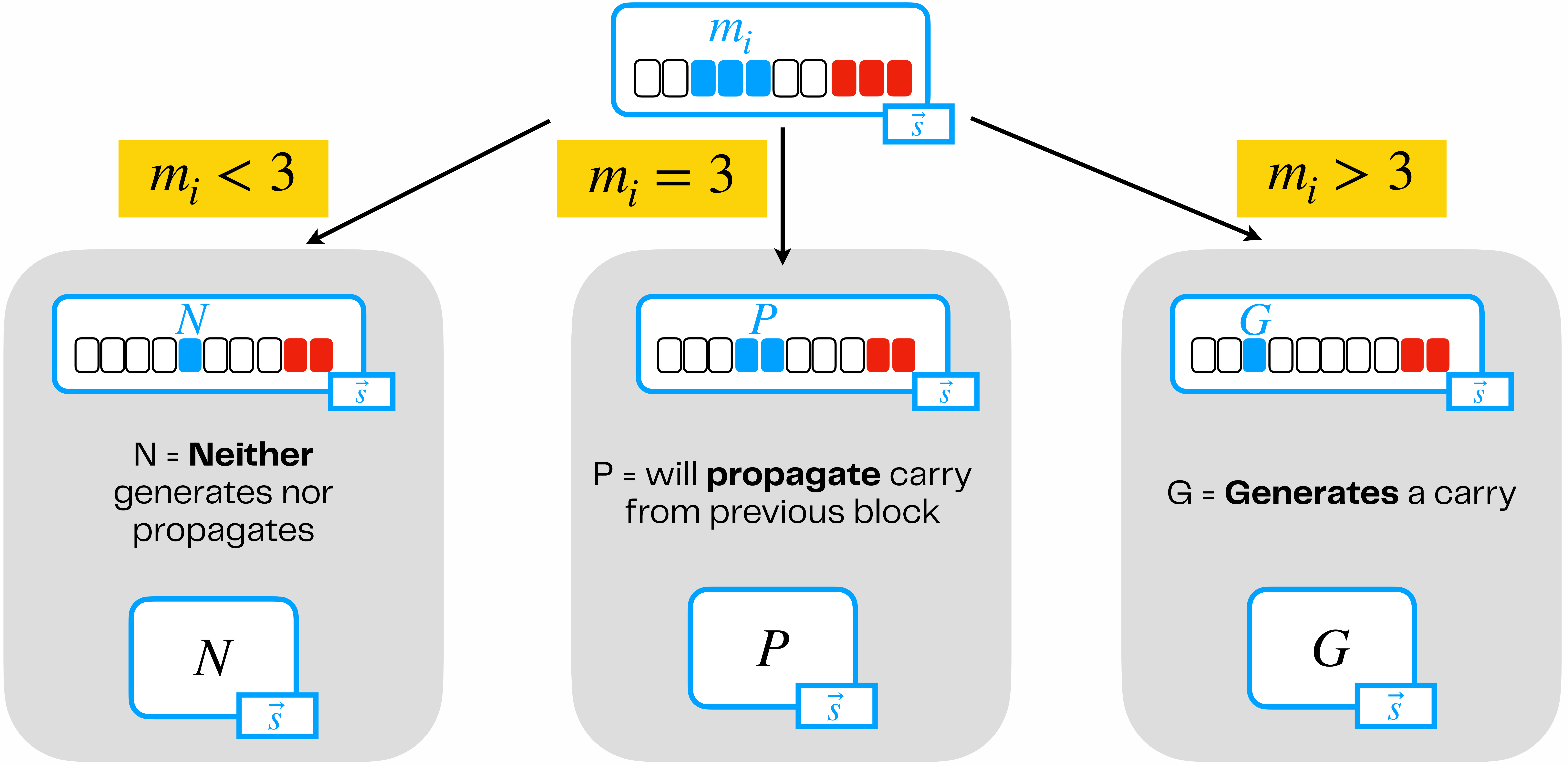
Propagating Carries



Highly sequential: $O(\text{BlockNumber})$

Faster Carry Propagation 1/2

1. Re-encode the state of each block

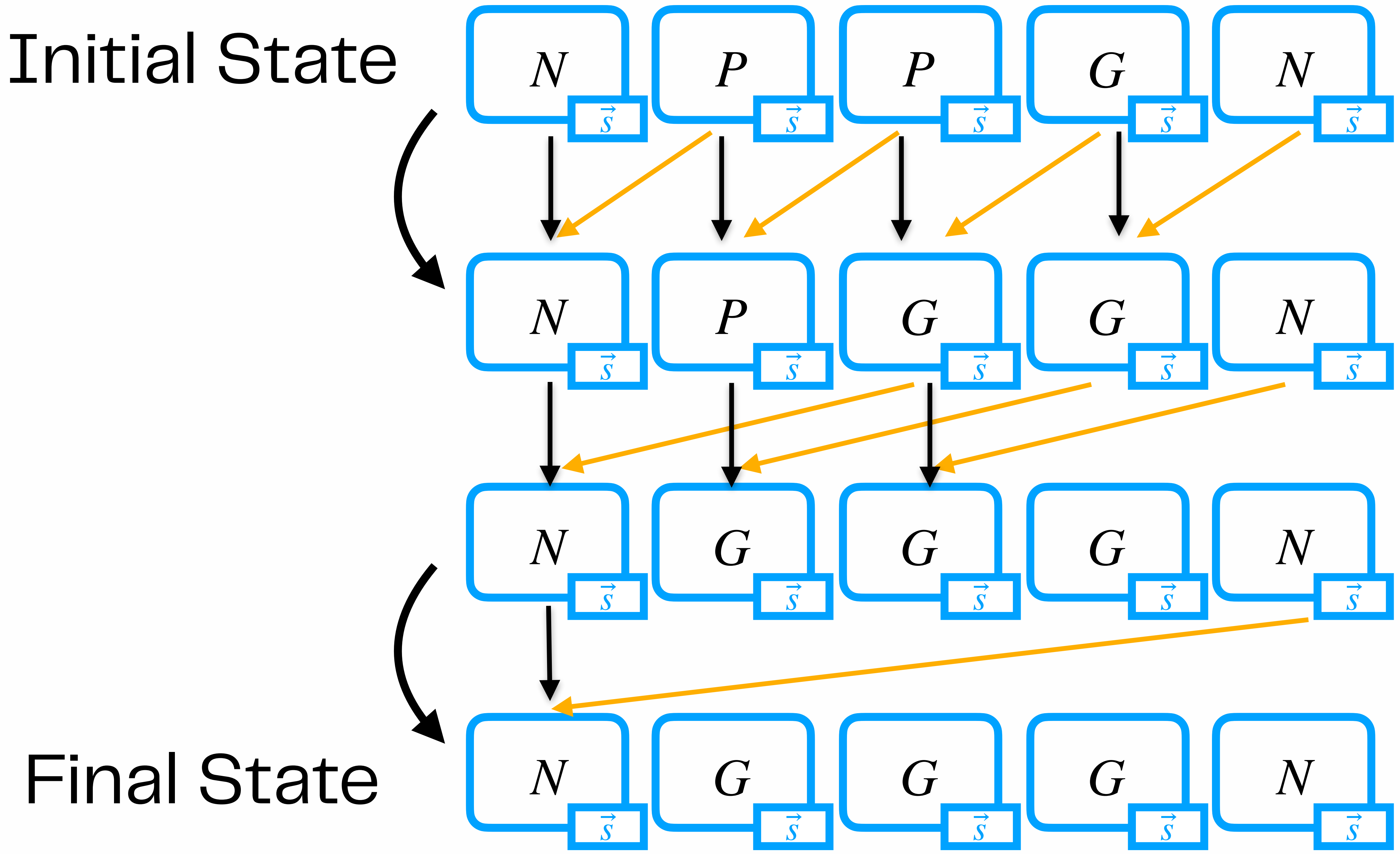


Faster Carry Propagation 1/2

2. Resolve the final state of the 'P' blocks [Hillis/Steele*]

↓	↙	**Output**
N	N/P/G	**N**
G	N/P/G	**G**
P	N	**N**
P	P	**P**
P	G	**G**

Truth Table



Leverage Parallelism: Depth $O(\log_2(\text{BlockNumber}))$

*Data Parallel Algorithms, Hillis, W Daniel and Steele Jr, Guy L, Communications of the ACM (1986)

Other operations

Operators:
$\& \& , \|, \ll , \gg , = , \leq , \geq , \ldots$

Signed Integers
(2's complement)

Complete Arithmetic
$+ \quad - \quad \times \quad /$

Operations with
overflow flag

Benchmarking FHE & Timings

How to benchmark FHE?

Security	$\lambda = 128$ bits
Error Probability	$P_{fail} = 2^{-40}$
A graph of FHE operators s.t. Input Encoding == Output Encoding	KS→PBS with Integer encoding
MsgMod, CarryMod, $\lVert \cdot \rVert_2$	MsgMod $= 2^2$, CarryMod $= 2^2 \Rightarrow \lVert \cdot \rVert_2 = 5$
Hardware	AMD EPYC 9R14 CPU @ 2.60GHz
Latency/Throughput	Latency

Some Benchmarks

KS→PBS

with $P_{fail} = 2^{-40}$

using TFHE-rs

Precision	2 bits	4 bits
Timings	6.09 ms	12.72 ms

Sequential Timings

Precision	2 bits	4 bits
Timings	3.94 ms	6.01 ms

Parallelized Timings (grouping factor 3)

AND Gate

with $P_{fail} = 2^{-80}$

Library	TFHE-rs v0.5	TFHElib (main)	OpenFHE v1.1.1 w/ Hexl
Timings	7.62 ms	19 ms	21.8 ms

Sequential Timings

TFHE-rs v0.5, https://github.com/zama-ai/tfhe-rs
TFHEllib v1.1, https://github.com/tfhe/tfhe
OpenFHE v1.1.1, https://github.com/openfheorg/openfhe-development, v1.1.1 w/ Hexl using the LMKCDEY variant

Integer Operations

Operation	FheUint32		FheUint64	
Add/Sub (+,-)	106 ms	81 PBS	124 ms	193 PBS
Mul (x)	**218 ms**	**481 PBS**	**371 ms**	**1856 PBS**
Equal/Not Equal (eq, ne)	48.2 ms	19 PBS	48.9 ms	36 PBS
Comparisons (ge, gt, le, lt)	83.8 ms	31 PBS	100 ms	63 PBS
Max/Min (max, min)	120 ms	79 PBS	134 ms	159 PBS
Bitwise operations (&, |, ^)	17.7 ms	16 PBS	17.6 ms	32 PBS
Div/Rem (/, %)	**3.82 s**	**2761 PBS**	**8.87 s**	**11504 PBS**
Left/Right Shifts/Rotations (<<, >>)	135 ms	213 PBS	154 ms	486 PBS

TFHE-rs timings with 192 cores

N

Conclusion

How can we abstract **FHE programming** from its cryptographic complexity to match the simplicity of **traditional coding**?

Trusted Automatic Parameter Generation
Efficient **Distribution** and **Failure** Tests

$\Longrightarrow$ Foolproof Automatic Parameter Selection

Static Parameters with the best precision-complexity ratio

$\Longrightarrow$ Efficient on a wide range of use cases

Thank you.

ZAMA

Contact and Links

jb.orfila@zama.ai

zama.ai

Github

Community links

ZAMA