TinyKeys:
A new approach to efficient multi-party computation

Carmit Hazay, Emmanuela Orsini, Peter Scholl
and **Eduardo Soria-Vazquez**

University of BRISTOL

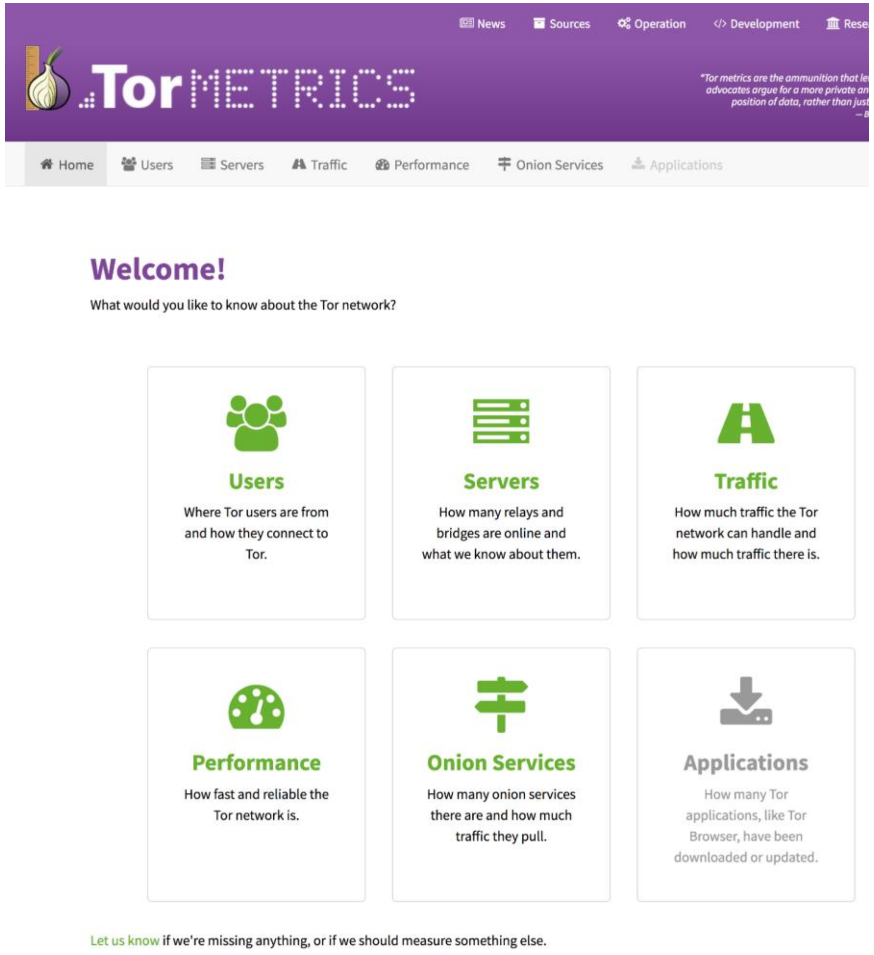
ECRYPT

Motivation: Large Scale, Dishonest Majority

Large number of users want to conduct surveys, auctions, statistical analysis, measure network activity, etc.

MPC between all users

Outsource to a committee



Dishonest Majority:

More parties ⇒ More trustworthy

MPC setting in this talk

Main focus:

- Concrete efficiency for large numbers of parties

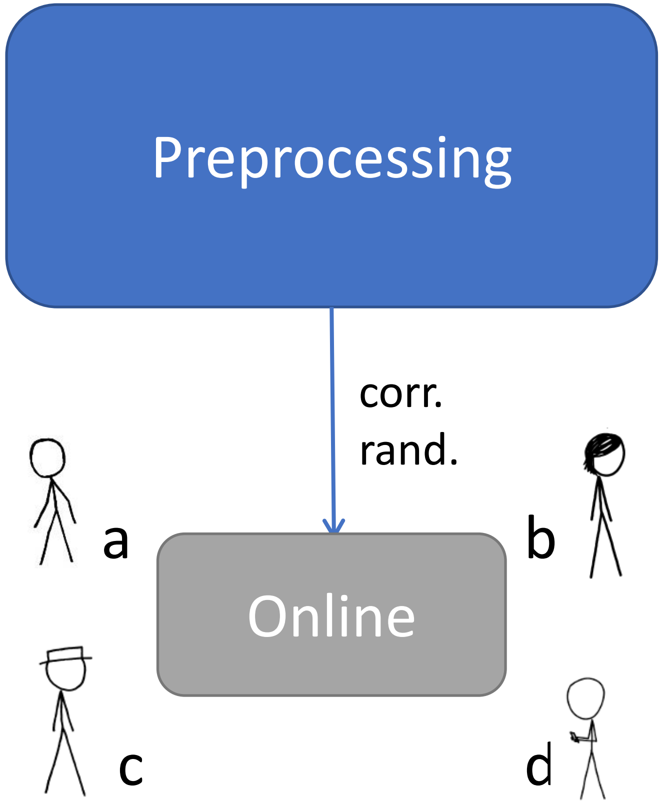
(e.g. n in 10s, 100s)

Adversary:

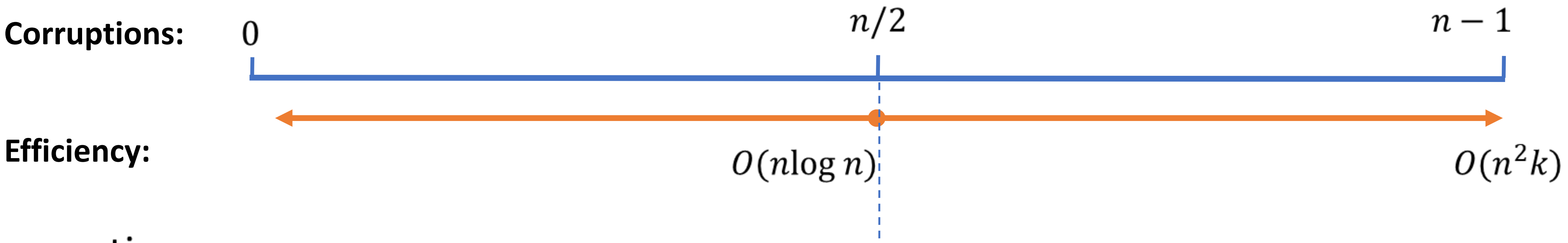
- Static, passive
- Dishonest majority $(t > n/2)$

Model of Computation:

- Boolean circuits
- Preprocessing phase



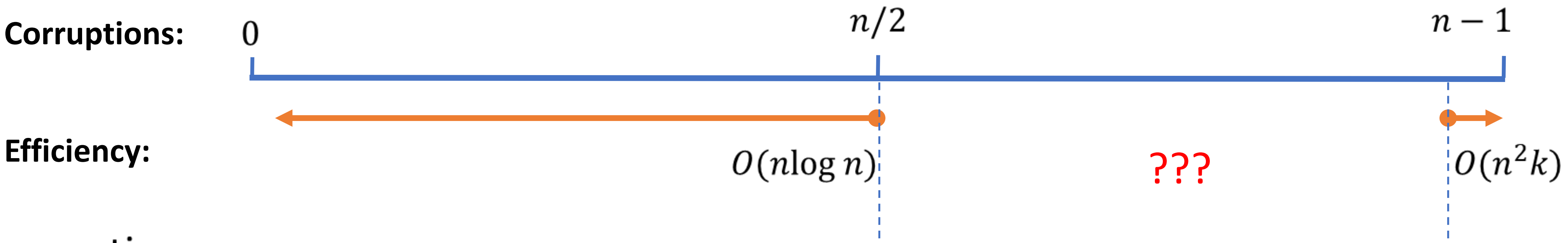
Corruption thresholds vs communication complexity of *practical* MPC



n parties

k-bit security

Corruption thresholds vs communication complexity of practical MPC



n parties
k-bit security

Can we design concretely efficient MPC protocols where each honest party can be leveraged to increase efficiency?

Our results

New dishonest majority protocols exploiting more honest parties:

1. Passive GMW-style MPC based on Oblivious Transfer.

- Up to **25x less communication** compared with $n-1$ corruptions.

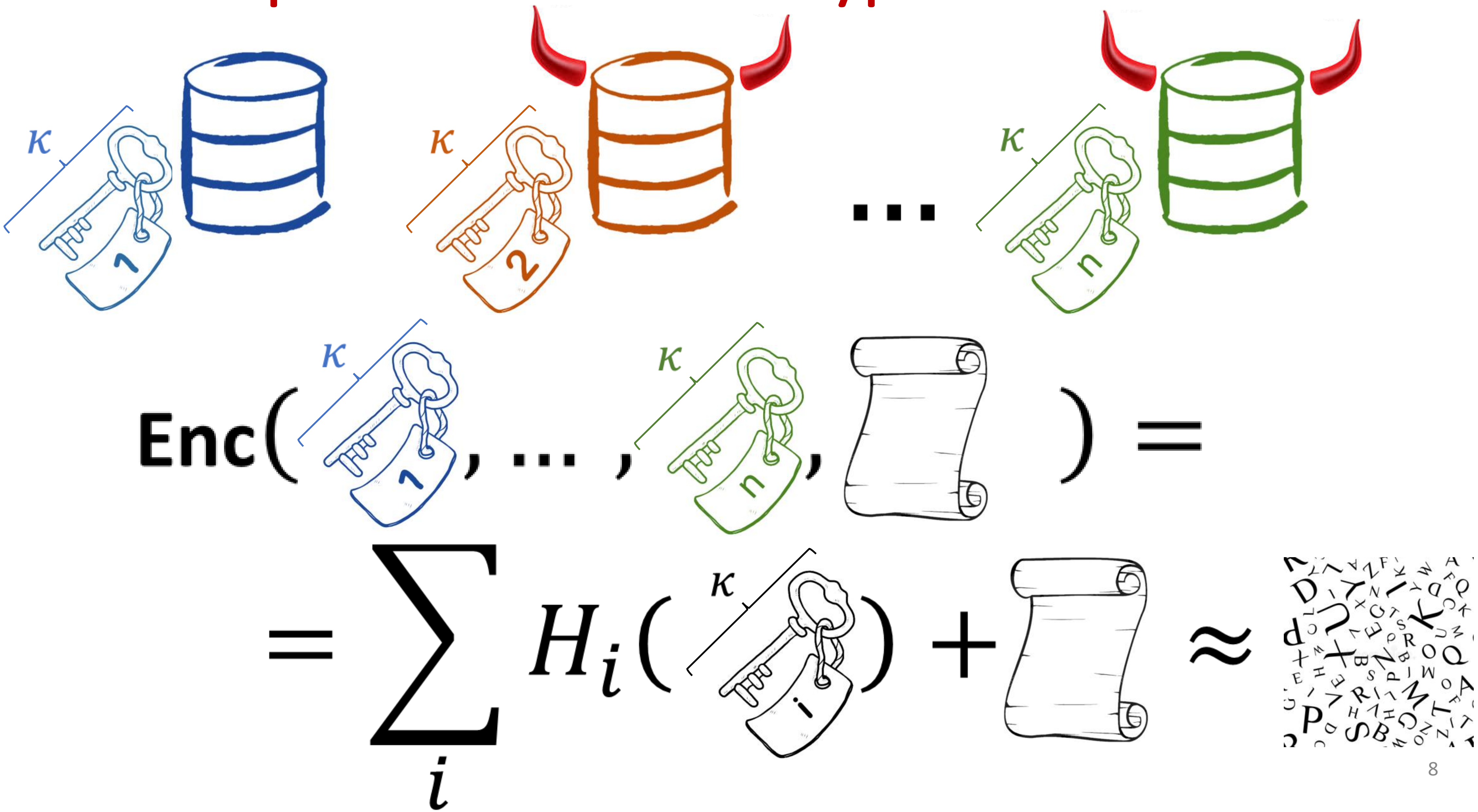
2. Passive constant-round MPC from Garbled Circuits.

- Up to **7x reduction** in GC size and communication cost.
- More efficient **online** phase: Up to **3x faster implementation** (circuit-dependent).

Best improvements with 20+ parties when 10-30% are honest.

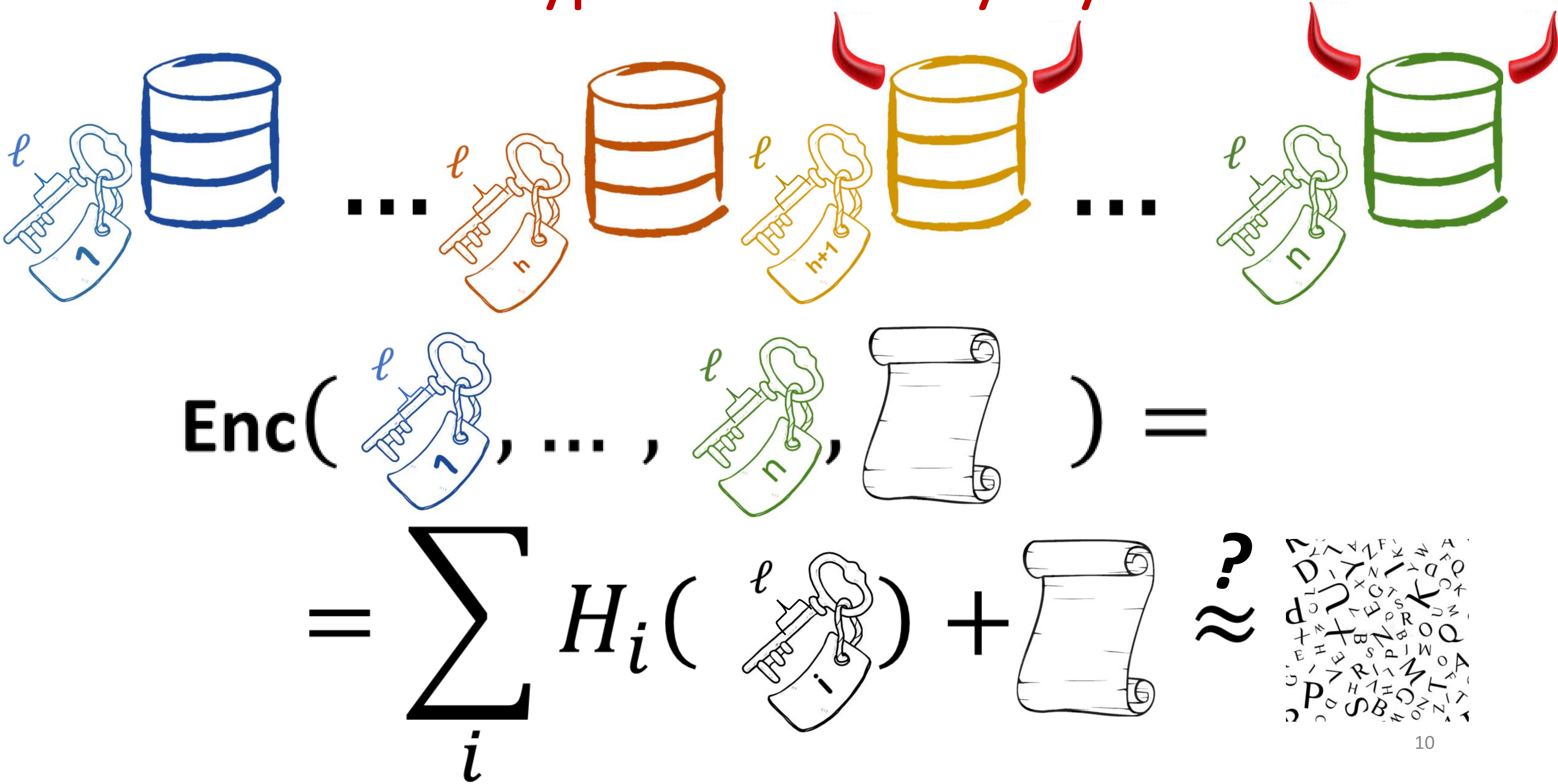
Introducing the TinyKeys technique

Warm-up: Distributed Encryption

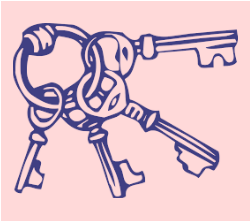


Distributed Encryption with TinyKeys

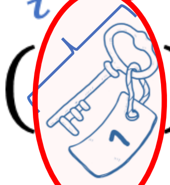
Distributed Encryption with TinyKeys



Breaking security



2^{ℓ} keys

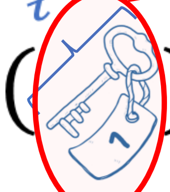
 $\overset{?}{\approx}$ $H_1($ 

$) + \cdots + H_h($ 

$)$

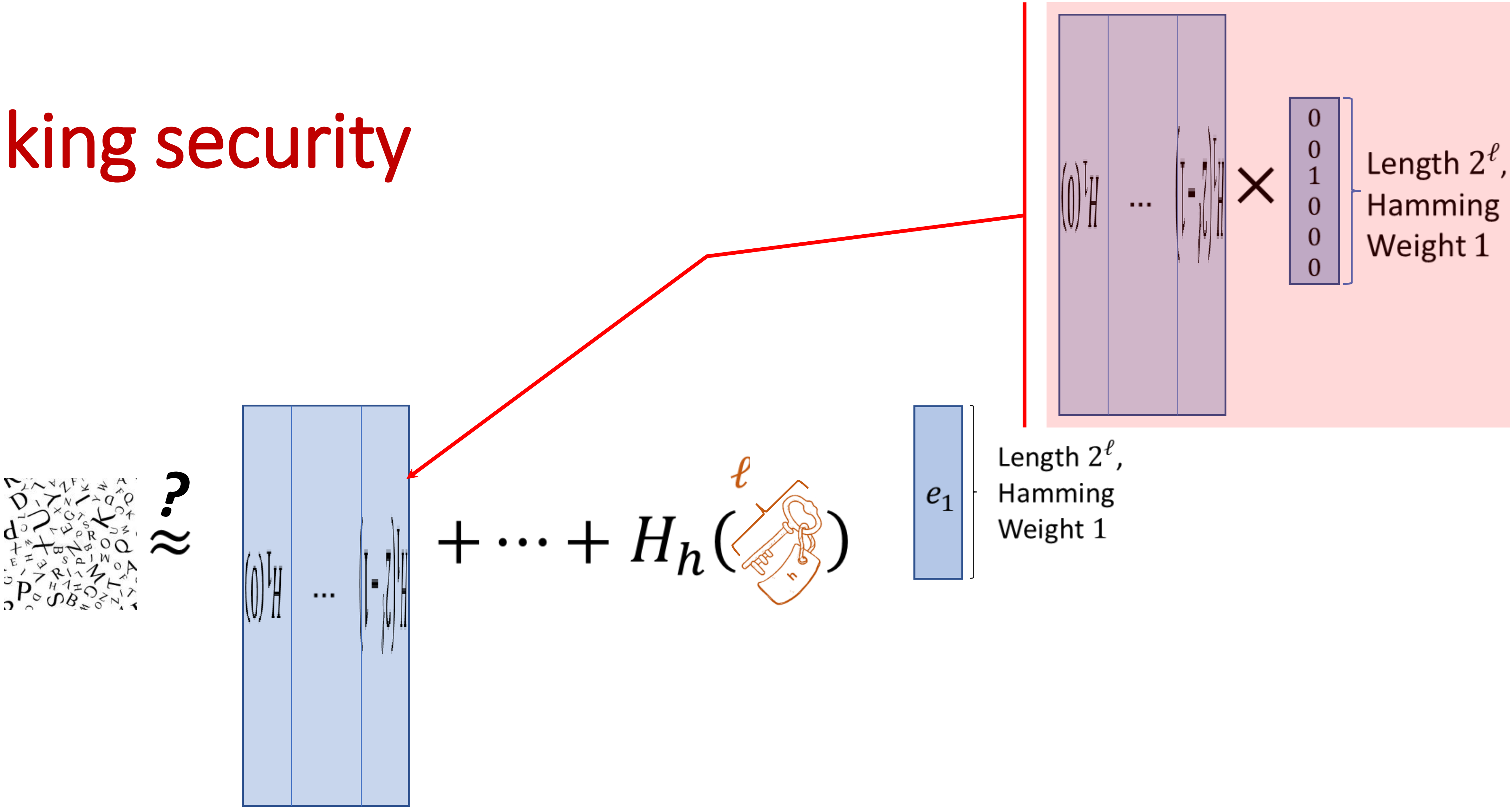
Breaking security

$$\begin{bmatrix} H_\ell(0) & \cdots & H_\ell(2^\ell-1) \end{bmatrix} \times \underbrace{\begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}}_{\text{Length } 2^\ell,\ \text{Hamming Weight } 1}$$

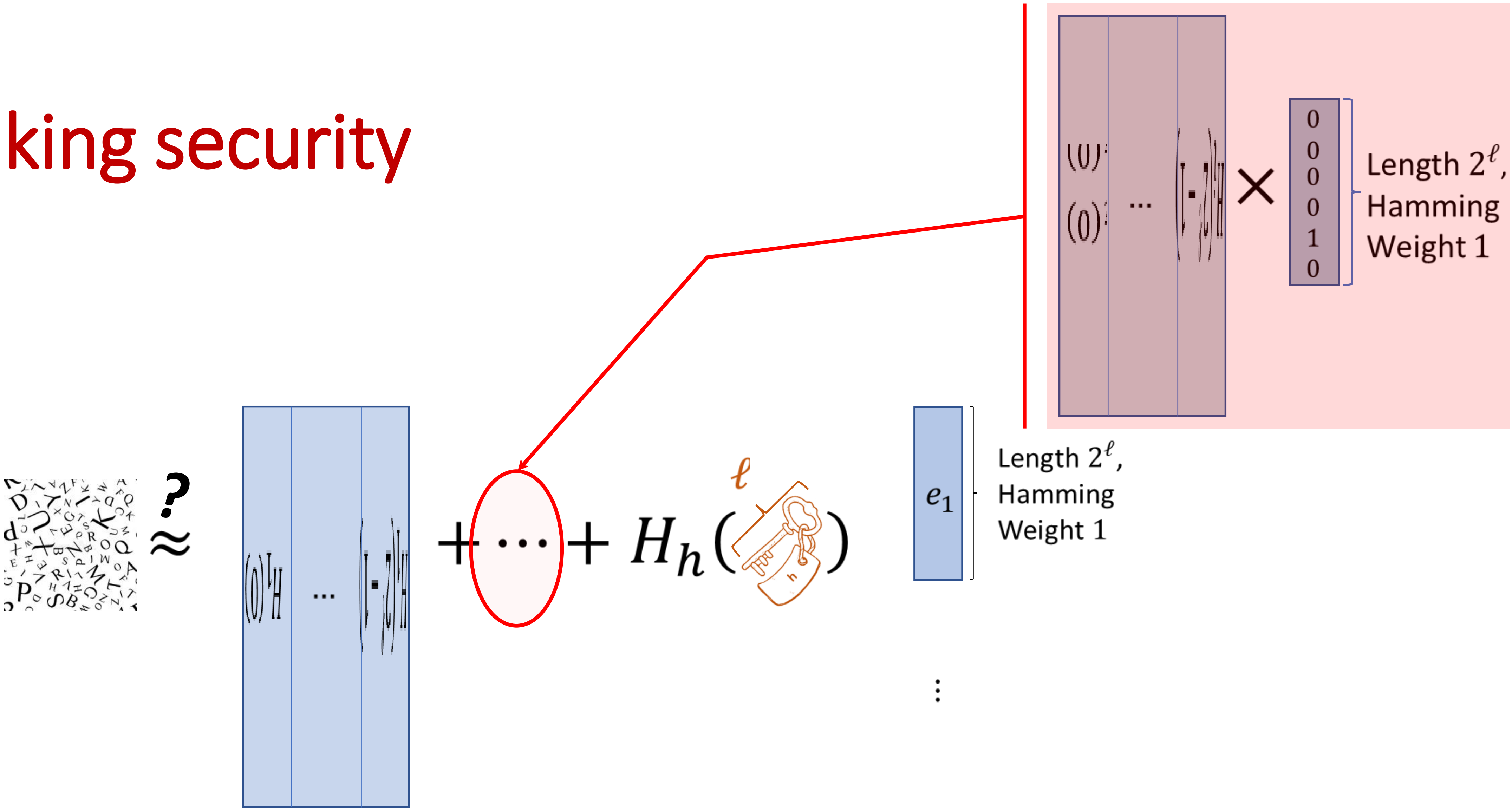
 $\stackrel{?}{\approx} H_1($  $) + \cdots + H_h($  $)$

ℓ ℓ

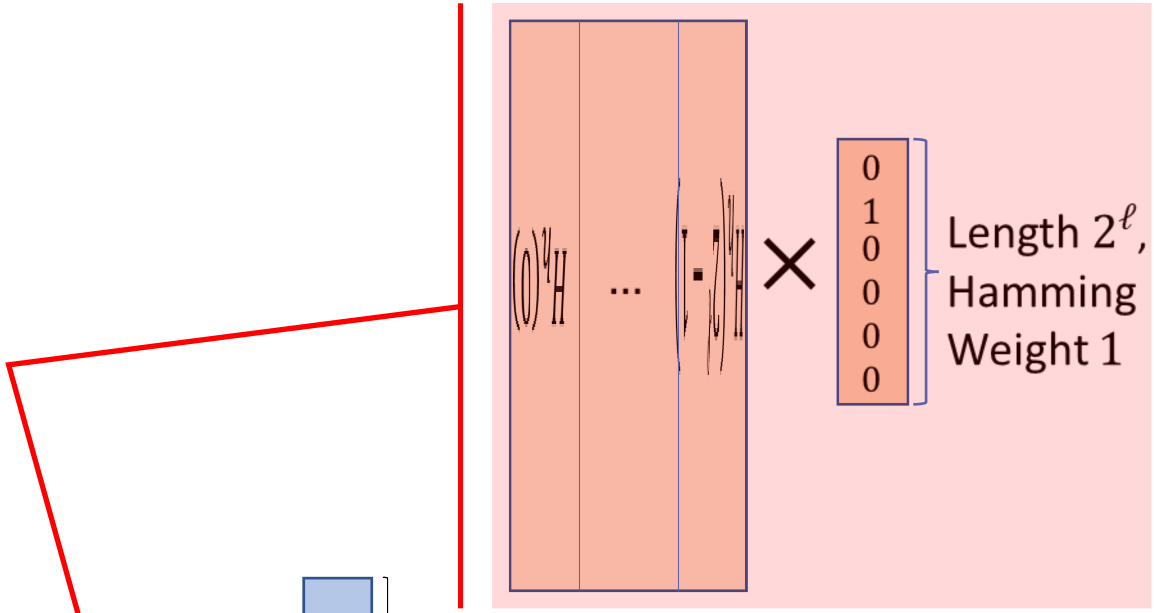
Breaking security

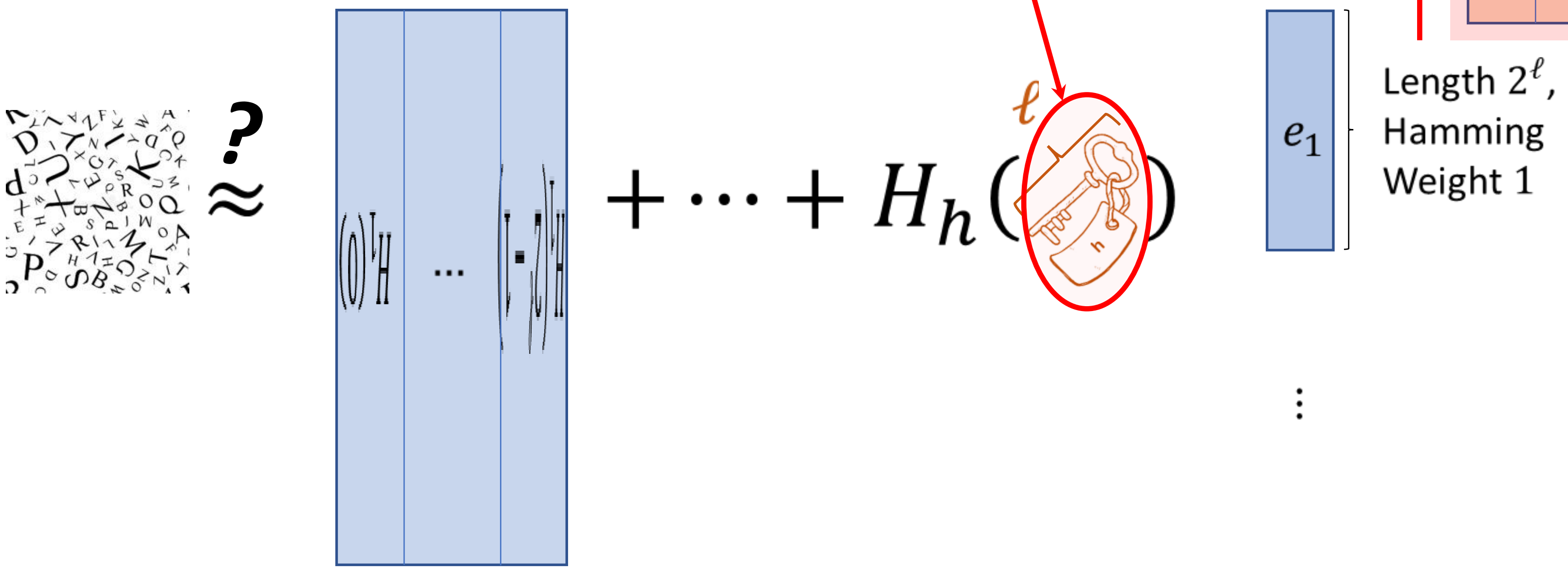


Breaking security

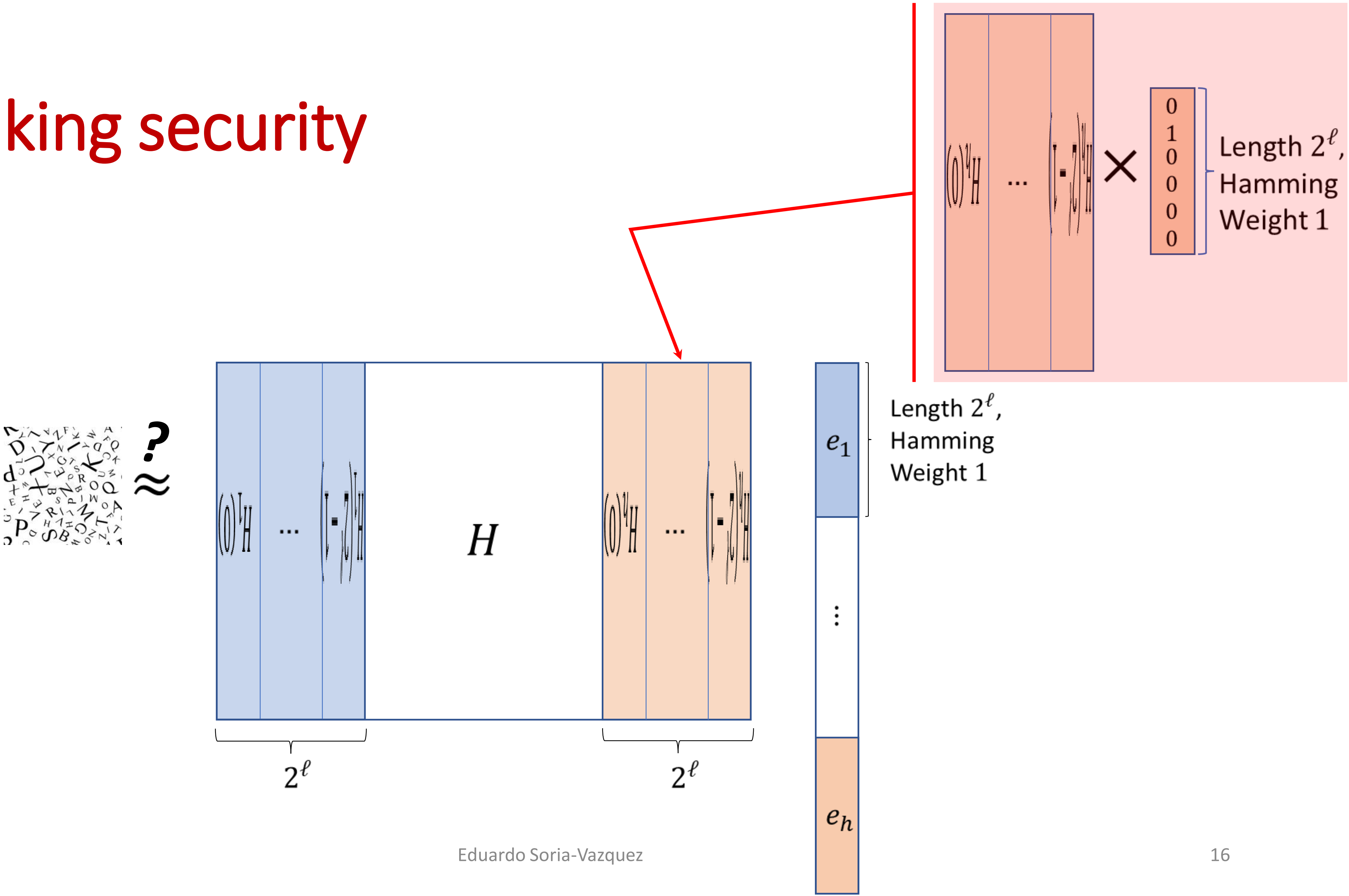


Breaking security



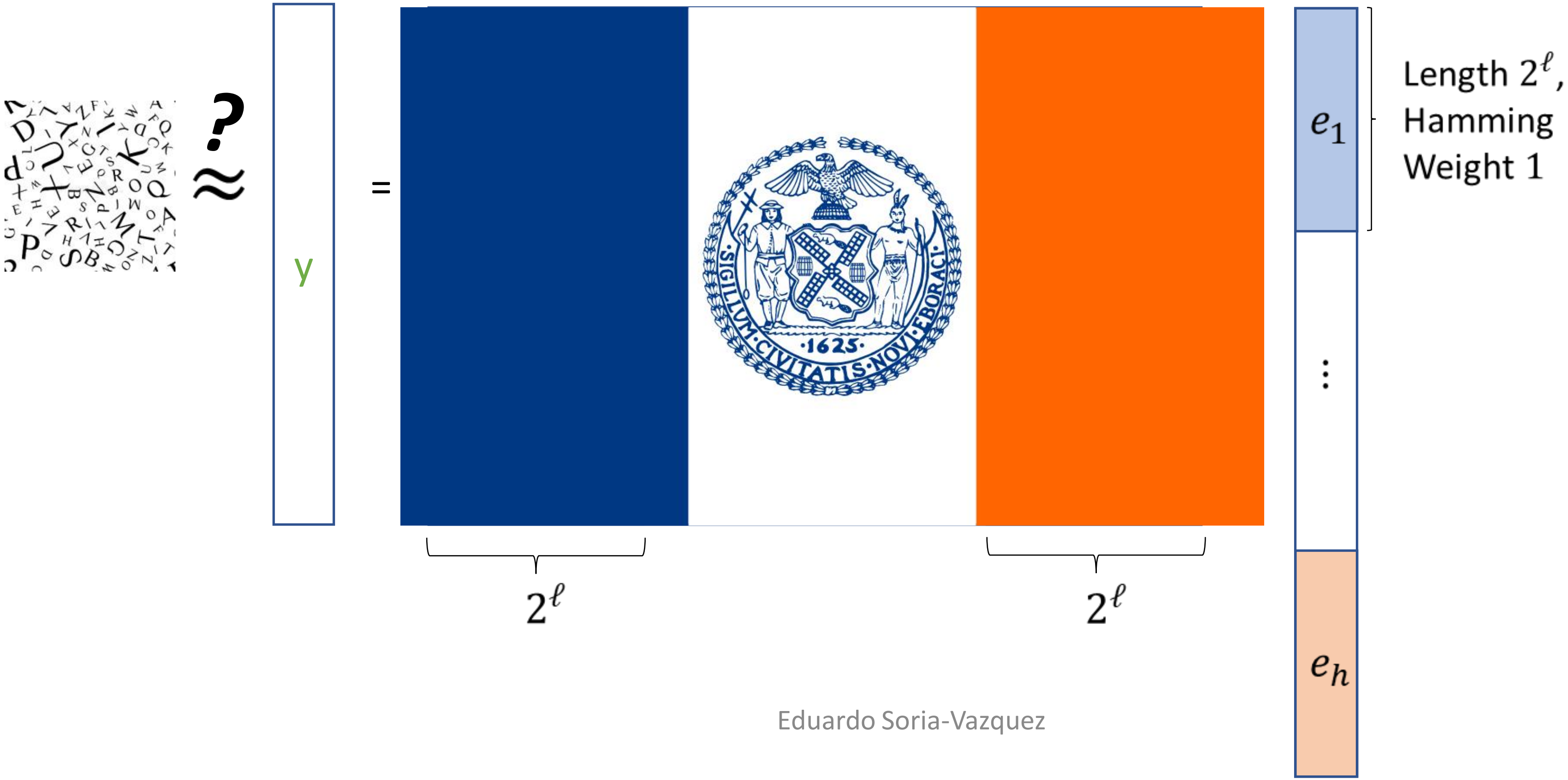


Breaking security



Breaking security

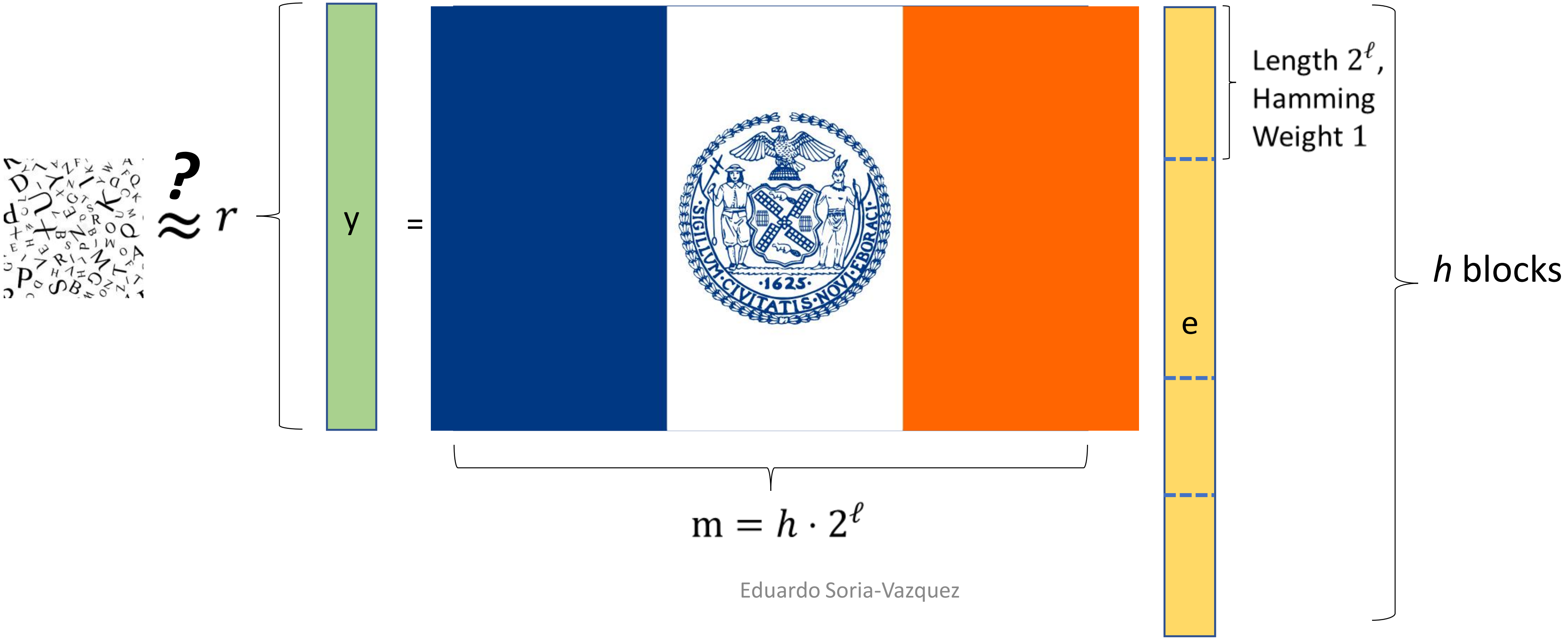
Adv wins: Given H and $y = He$, distinguish y from random



Breaking security: Regular Syndrome Decoding

Sample random $H \in \{0,1\}^{r \times m}$, and regular $e \in \{0,1\}^m$ of weight h

Adv wins: Given H and $y = He$, find e $\Leftrightarrow$ distinguish y from random.



Hardness of Regular Syndrome Decoding

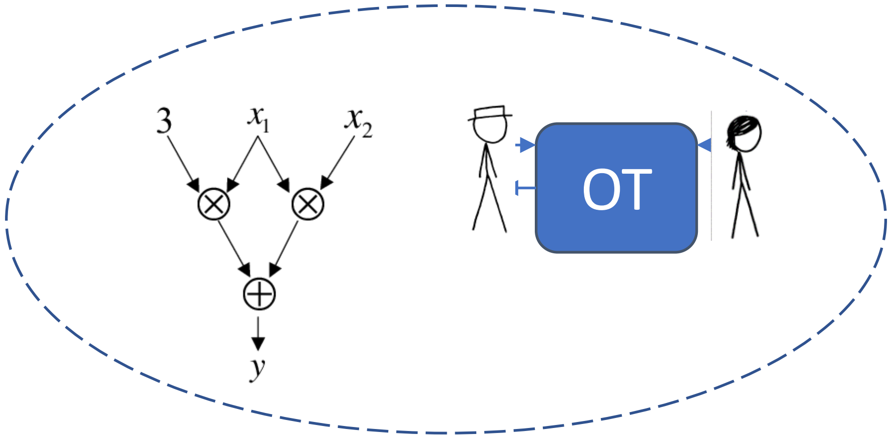
- Used for SHA-3 candidate FSB [Augot Finiasz Sendrier 03]
  - Not much easier than syndrome decoding ⇔ LPN
- Params: Message length r, key length ℓ, #honest h.
- **Statistically hard** for small r/large h.

[FS09]

[Saa07] [MO15] [NCB11] [Kir11] [BM17] [BJMM12] [CJ04] [BLN+09] [BLP08] [MS09] [MMT11] [BLP11]

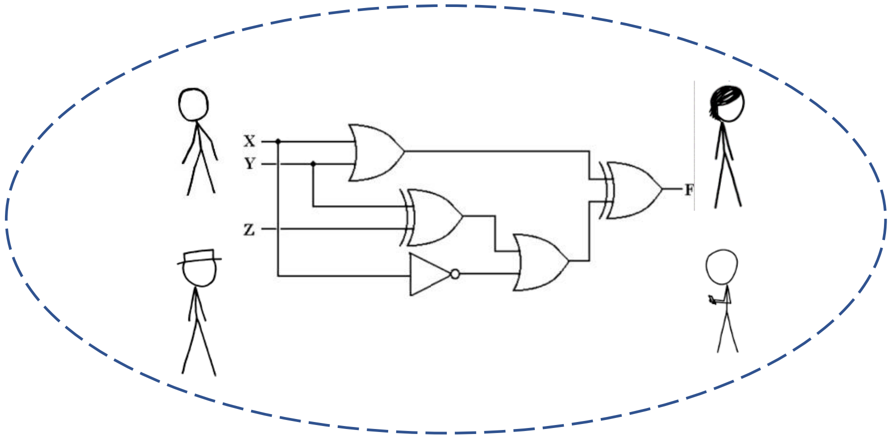
TinyKeys: A little honesty goes a long way

(Tiny)GMW



- Key length: $\ell \geq 1$
- Rest of the talk

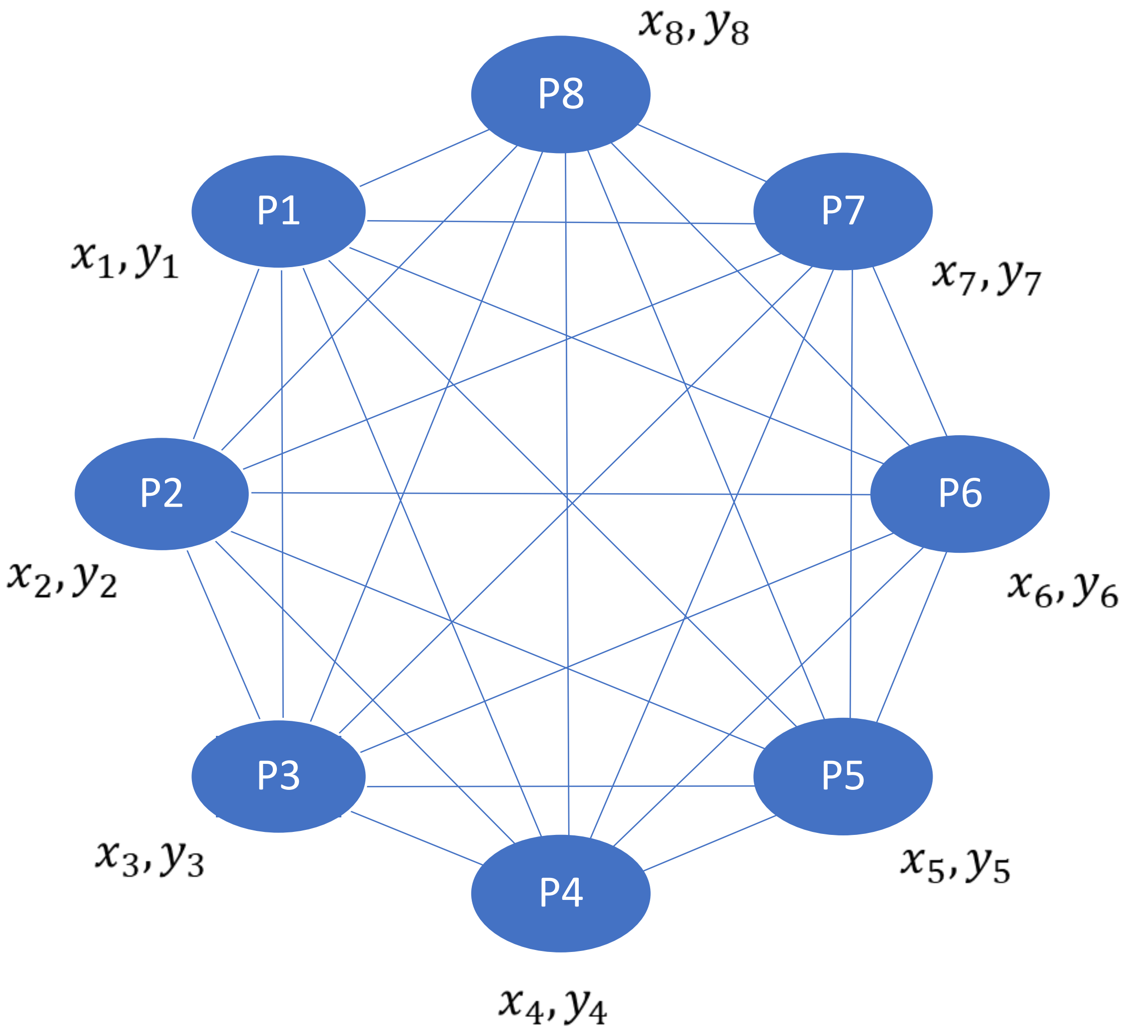
(Tiny)BMR



- Key length: $\ell \geq 5$
- Many challenges:
  - High Fan-Out
  - Enabling FreeXOR

(Tiny)GMW

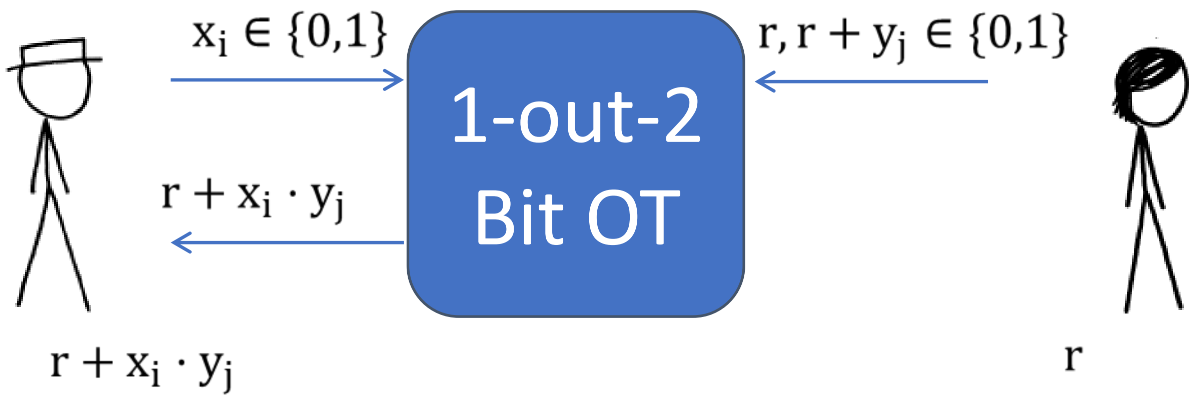
Quick recap of GMW



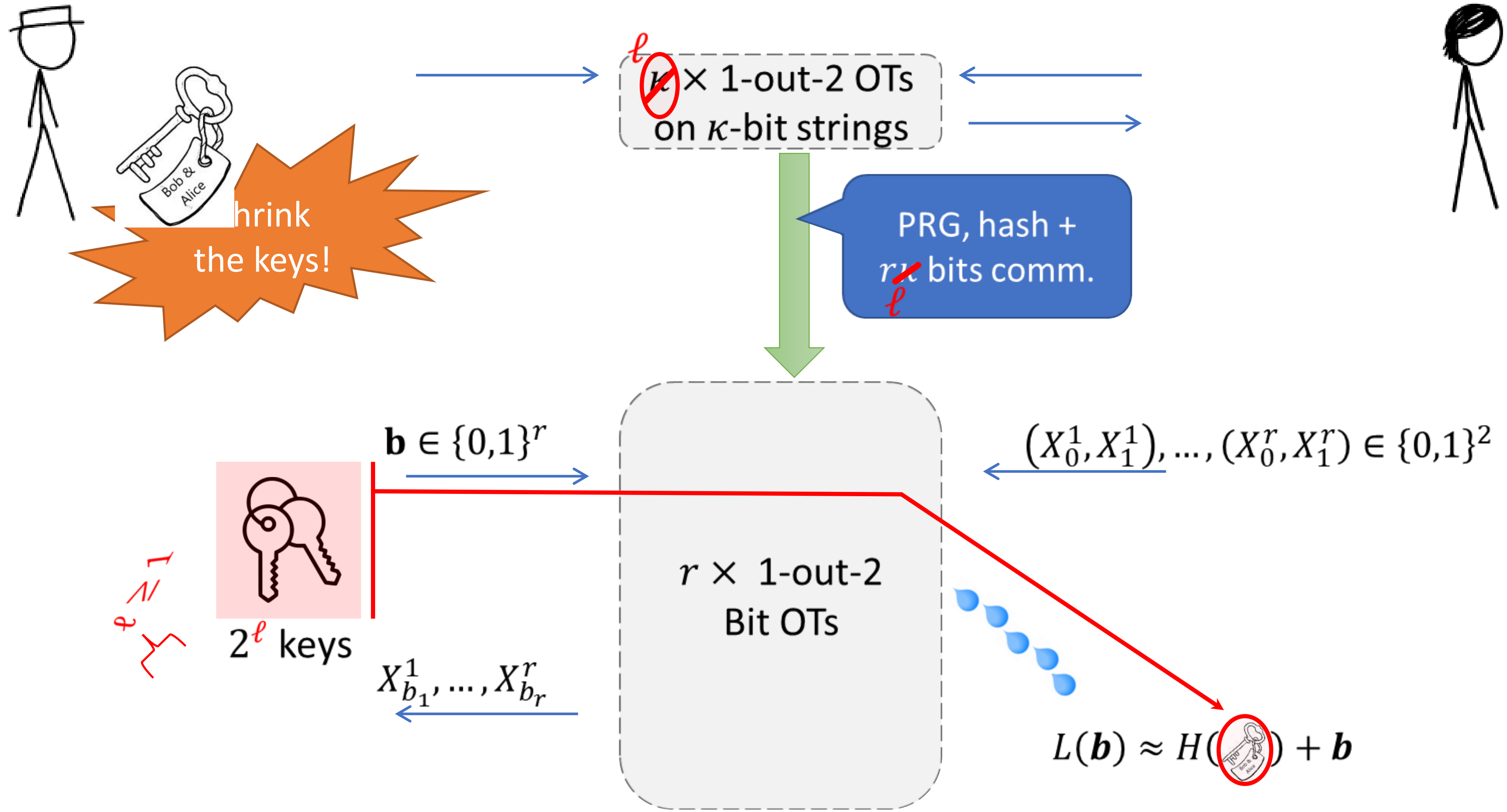
$$
\begin{array}{rcccccl}
x & = & x_1 & + \;\ldots\; + & x_n & \in \{0,1\} \\
+ \quad y & = & y_1 & + \;\ldots\; + & y_n & \in \{0,1\} \\
\hline
\end{array}
$$

$$x + y = (x_1 + y_1) + \ \ldots\ + (x_n + y_n)$$

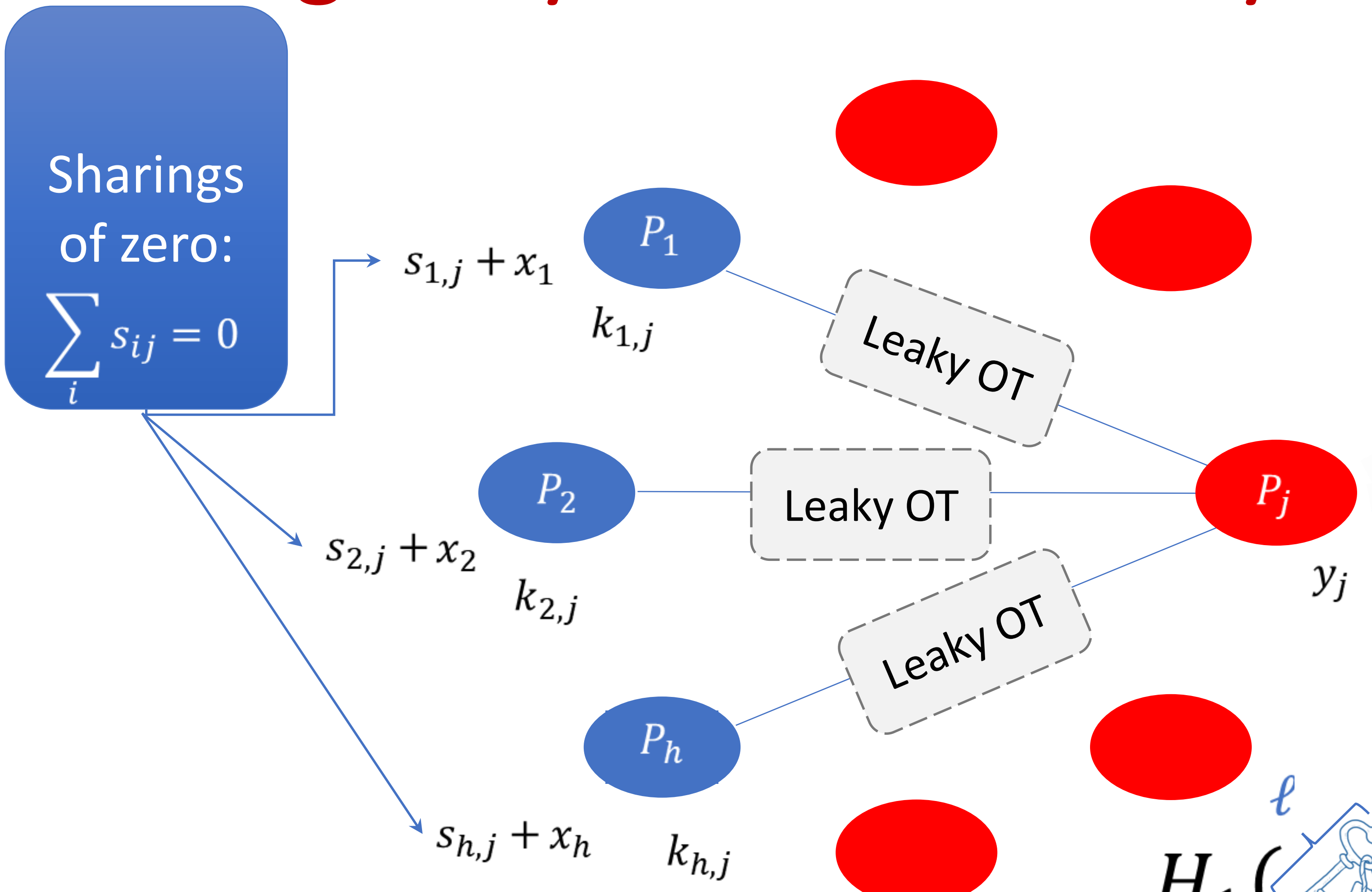
$$x \wedge y = (x_1 + \cdots + x_n) \cdot (y_1 + \cdots + y_n)$$



"IKNP" OT extension with short keys!



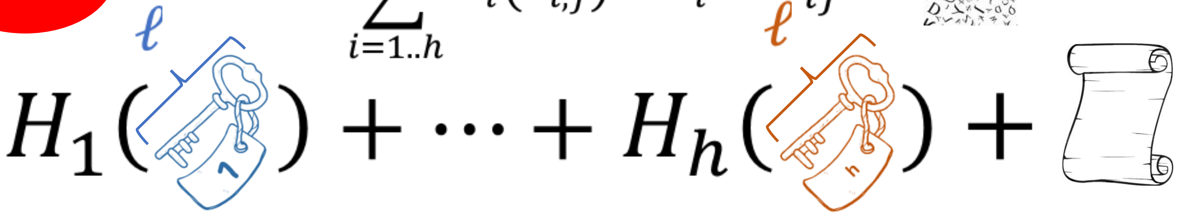
Using leaky OT for GMW-style MPC



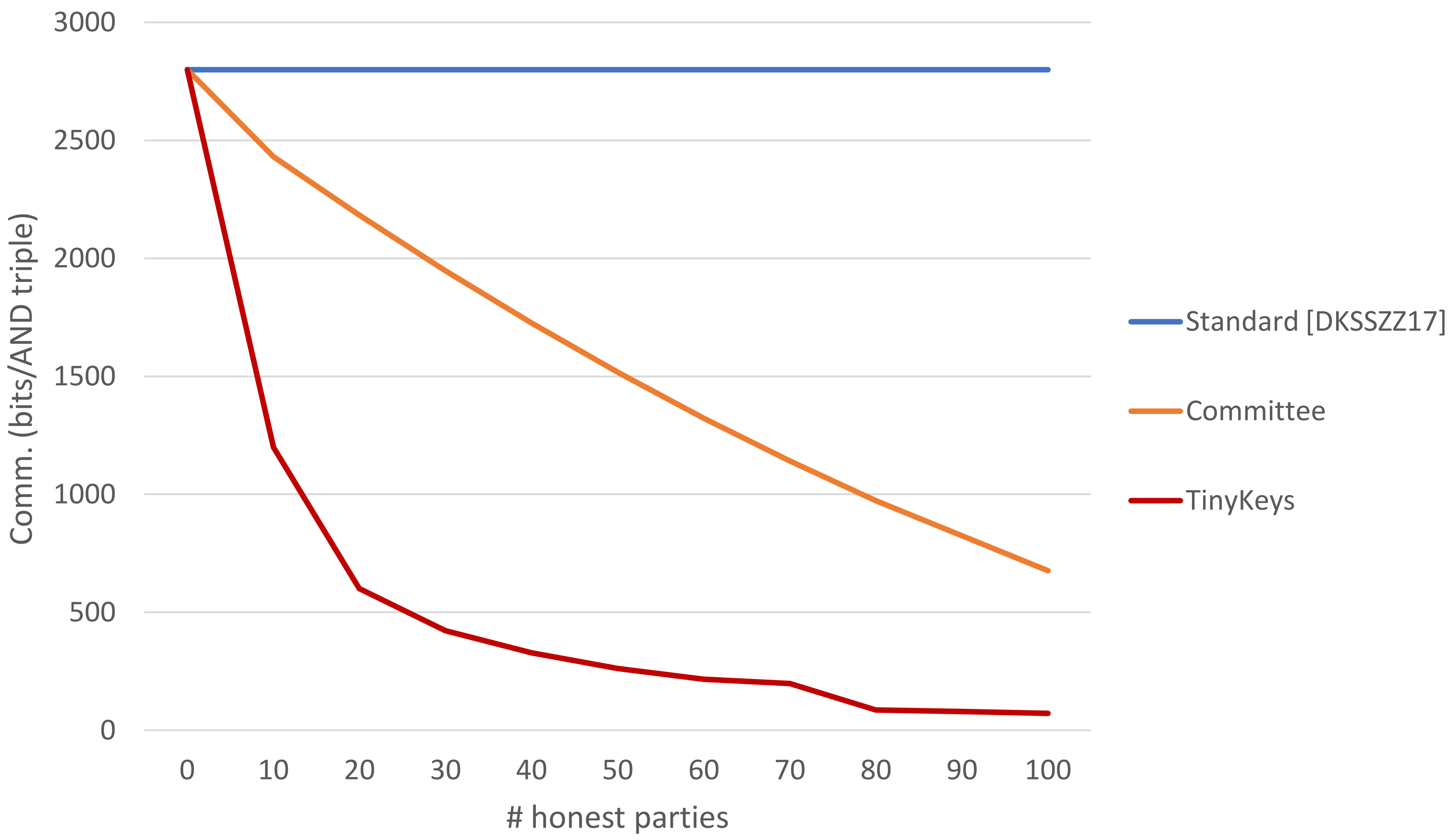
$$x \wedge y = (x_1 + \cdots + x_n) \cdot (y_1 + \cdots + y_n)$$
$$= \sum_{j=1..n} (x_1 + \cdots + x_n) \cdot y_j$$
$$+ \sum_{j=1..n} (s_{1,j} + \cdots + s_{n,j}) \cdot y_j$$

$$H_1(\mathrm{k}_{1,j}) + x_1 + s_{1,j} \approx$$
$$H_2(\mathrm{k}_{2,j}) + x_2 + s_{2,j} \approx$$
$$\ldots$$
$$+ \quad H_h(\mathrm{k}_{h,j}) + x_h + s_{h,j} \approx$$

$$\sum_{i=1..h} H_i(\mathrm{k}_{i,j}) + x_i + s_{ij} \overset{?}{\approx}$$



GMW: Communication compl. (200 parties)



Conclusion and future directions

- New technique for distributing trust (more honesty ⇒ shorter keys).
- Improved protocols with 20+ parties.
  - GMW: Up to 25x in communication (vs multiparty [DKSSZZ17]).
  - BMR: Up to 7x in communication (vs [BLO16]). Online phase up to 3x faster.

Follow-up work: Active Security – TinyKeys for TinyOT (Asiacrypt ’18).

Future challenges:

- Optimizations, more cryptanalysis (conservative parameters at the moment).
- More applications,

Thank you! Questions?

Paper: https://ia.cr/2017/214 [Full version]

TinyKeys: A New Approach to Efficient Multi-Party Computation
Carmit Hazay, Emmanuela Orsini, Peter Scholl and ***Eduardo Soria-Vázquez***

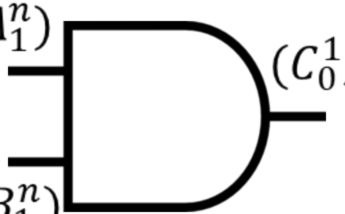
Mail: eduardo.soria-vazquez@bristol.ac.uk

[Beaver Micali Rogaway 90]

BMR: Multi-party garbled circuits

$(A_0^1, \dots, A_0^n), (A_1^1, \dots, A_1^n)$

$(B_0^1, \dots, B_0^n), (B_1^1, \dots, B_1^n)$



$(C_0^1, \dots, C_0^n), (C_1^1, \dots, C_1^n)$

$E_{A_0,B_0}(C_0)$

$E_{A_0,B_1}(C_0)$

$E_{A_1,B_0}(C_0)$

$E_{A_1,B_1}(C_1)$

Shrink
the keys!

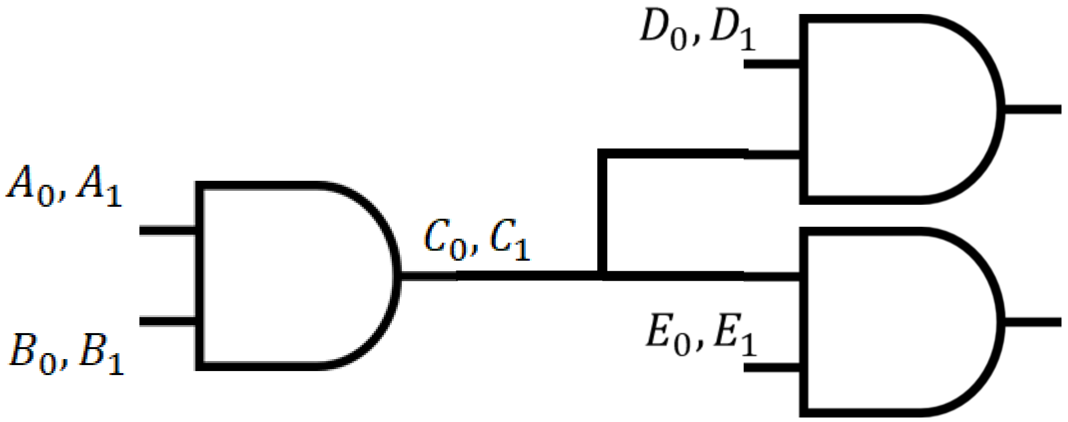
Each P_i gets $A_0^i, A_1^i \in \{0,1\}^{\cancel{\kappa}\,\ell}$ etc

Use distributed encryption: $E_{A,B}(C) = $

$$\begin{gathered} H(1 \,||\, A^1 || B^1) \\ + \\ \dots \\ + \\ H(n \,|| A^n || B^n) \\ + \\ (C^1, \dots, C^n) \end{gathered}$$

For hash function $H : \{0,1\}^* \to \{0,1\}^{\cancel{n\kappa}\, n\ell}$

(Tiny)BMR: Some technical challenges



- Reusing keys reduces security in regular syndrome decoding
- Problem for:
  - High fan-out
  - Free-XOR
- Solution:
  - Splitter gates [Tate Xu 03] – can be garbled for free
  - Free-XOR enabled using different offsets (FleXOR style [CITE])

Thank you! Questions?

Paper: https://ia.cr/2017/214 [Full version]

TinyKeys: A New Approach to Efficient Multi-Party Computation
Carmit Hazay, Emmanuela Orsini, Peter Scholl and ***Eduardo Soria-Vázquez***

Mail: eduardo.soria-vazquez@bristol.ac.uk