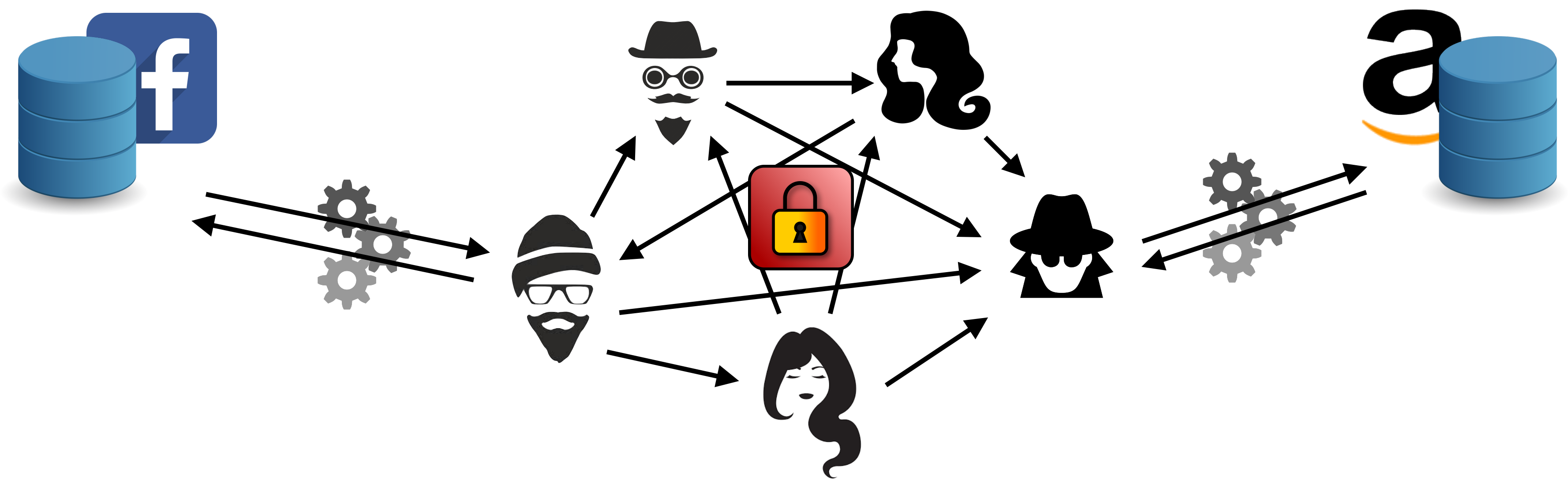
Pseudorandom Correlation Generators from Learning Parity with Noise

Elette Boyle, **Geoffroy Couteau**, Niv Gilboa, Yuval Ishai,
Lisa Kohl, Peter Rindal, Peter Scholl

Based on the results in [CCS:B**C**GIO17, CCS:B**C**GI18, CRYPTO:B**C**GIKS19,
CCS:B**C**GIKRS19, CRYPTO:B**C**GIKS20, FOCS:B**C**GIKS20]

Secure Computation

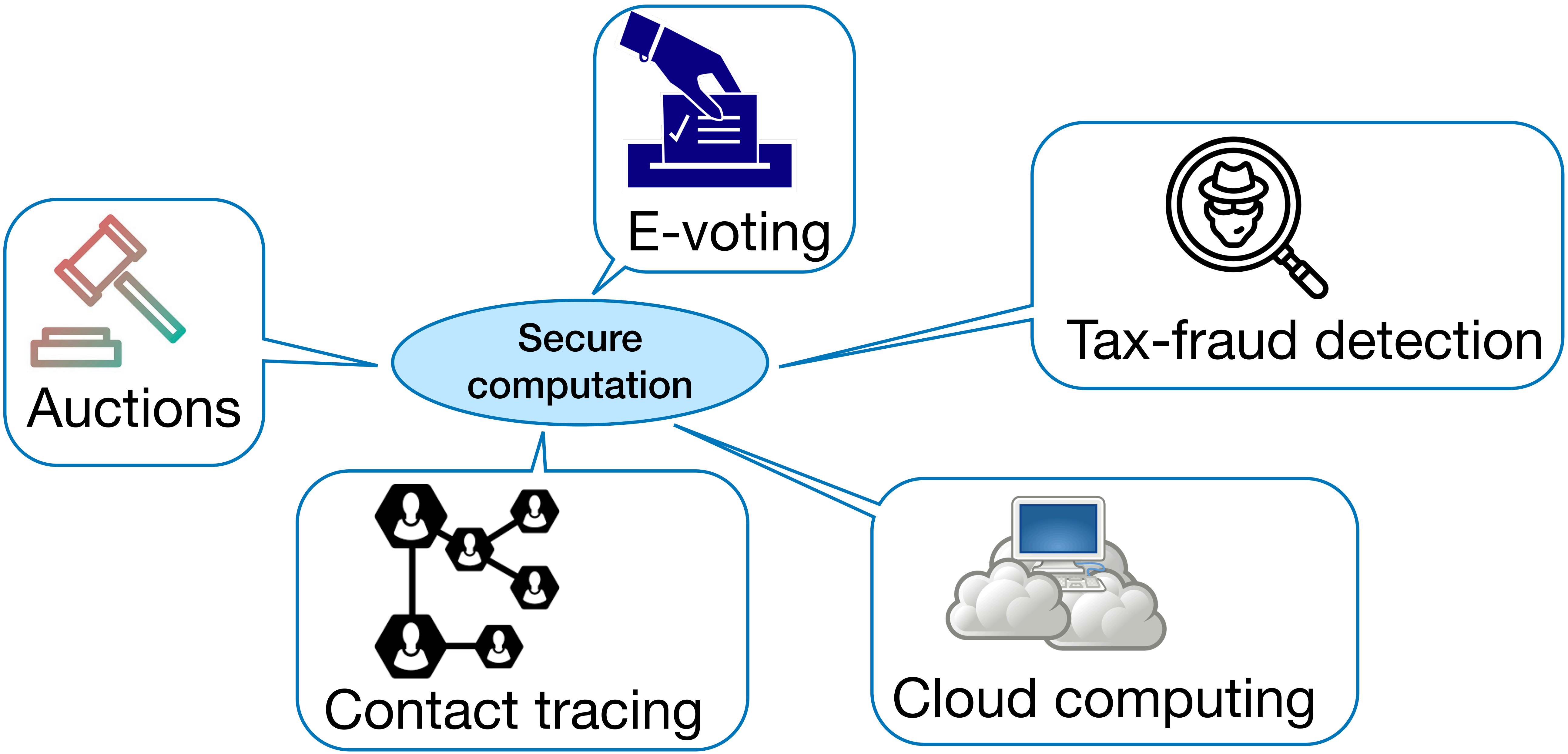


Classical cryptography: protecting communications. However, data are not only *exchanged*: they are often *used in computations.*

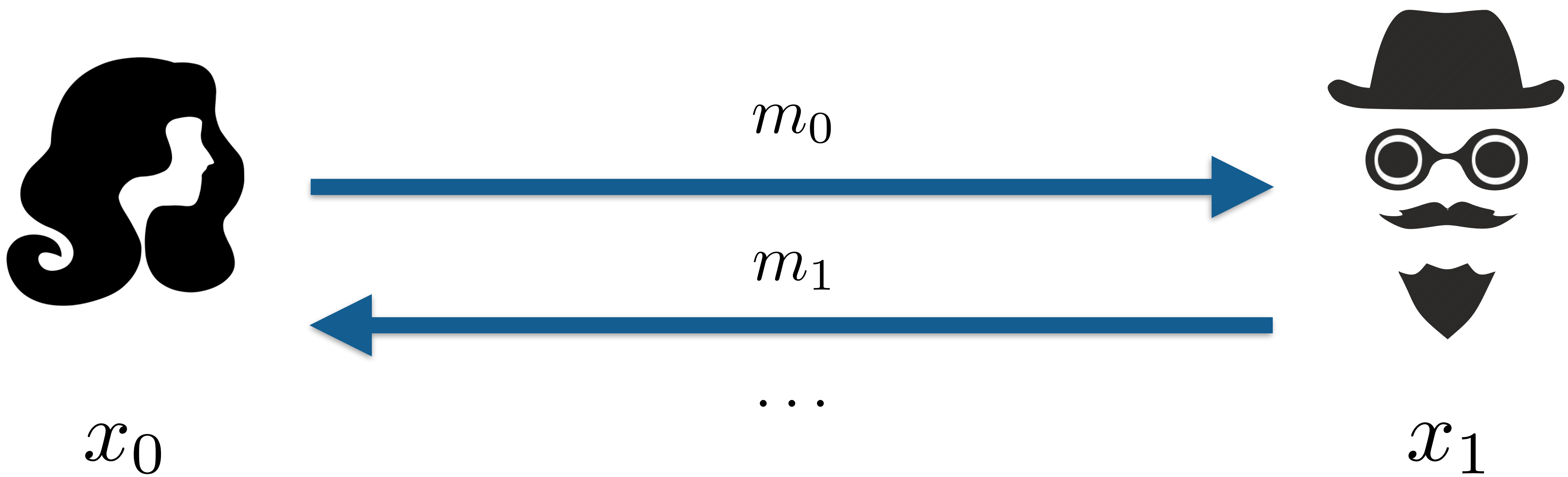
Is it possible to protect data privacy even when it's used in computations?

Secure Computation - Examples

Scenarios



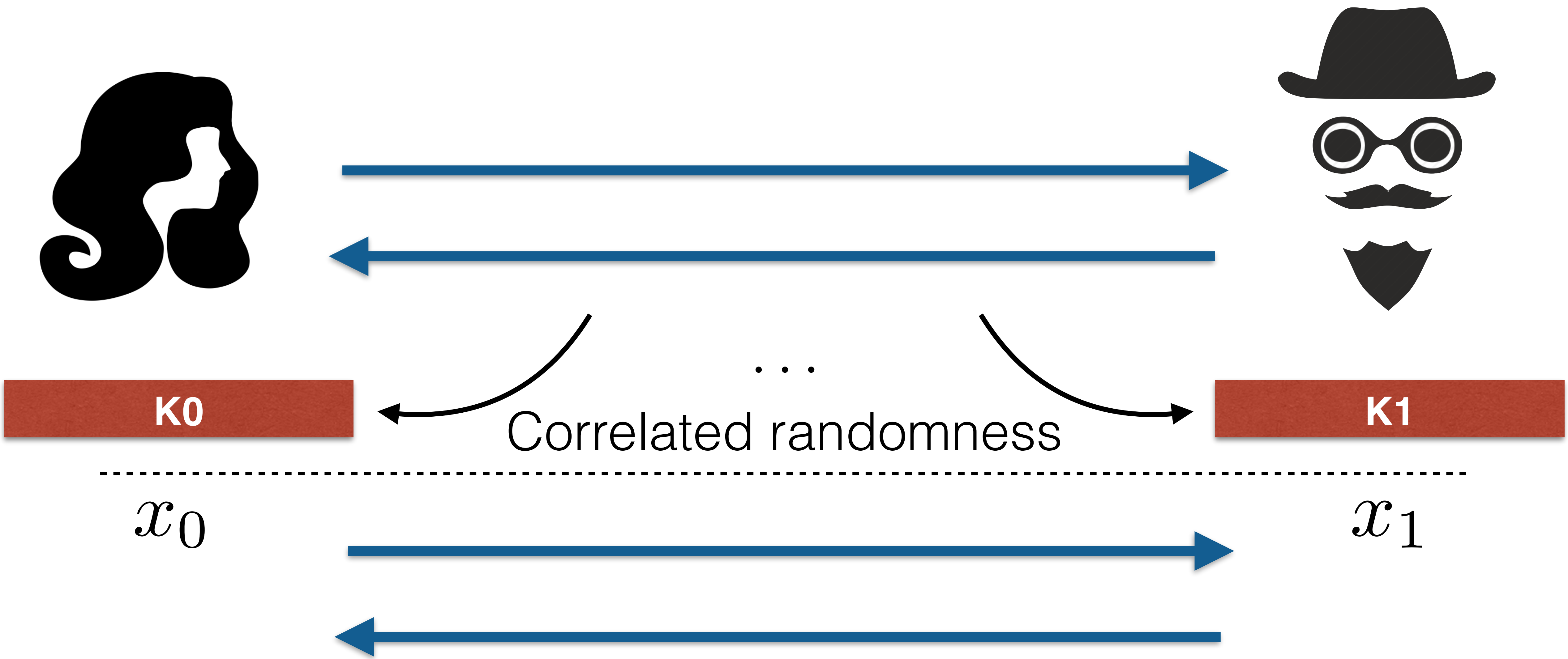
Secure 2-Party Computation



- both parties learn the output $f(x_0, x_1)$
- no party learns additional information

✓ (Yao, 1986) Can evaluate any poly time function
✗ Computationally expensive

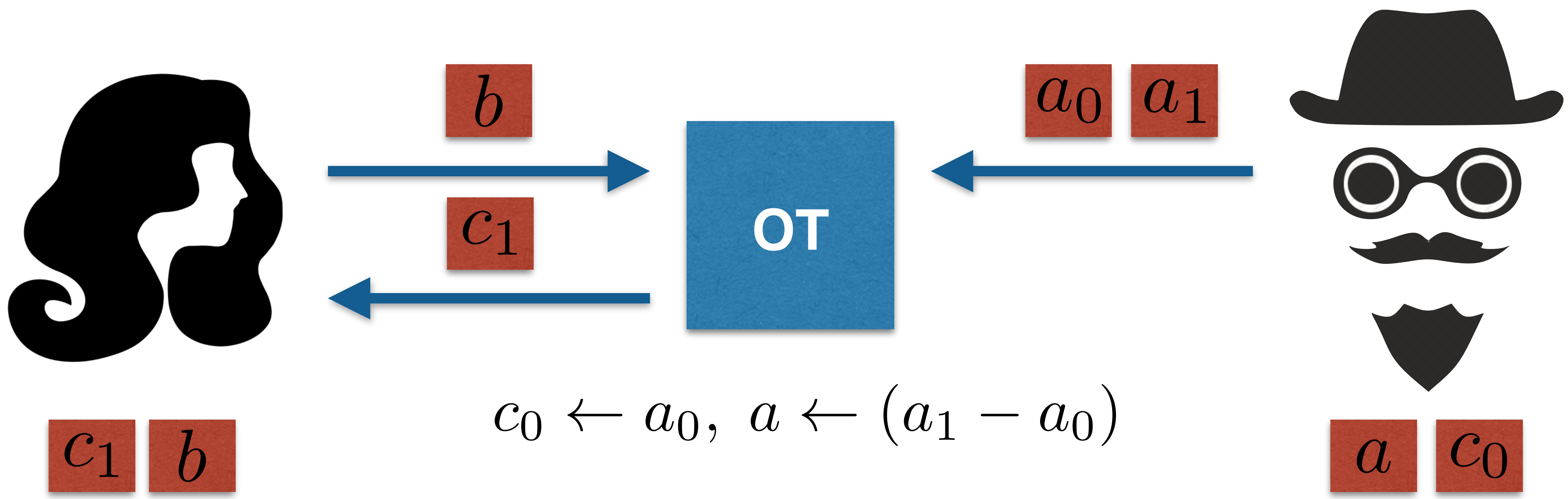
Secure 2-Party Computation in the Preprocessing Model



- ✓ Fast, information-theoretic online phase
- ✓ Security against dishonest majority
- ✗ Expensive preprocessing (storage and communication)

Example of a Useful Correlation: Oblivious Transfer

Security. Alice learns only c_1, Bob learns nothing



$$c_0 \leftarrow a_0, \; a \leftarrow (a_1 - a_0)$$

$$c_0 \oplus c_1 = a \cdot b$$

=> degree 2 correlation

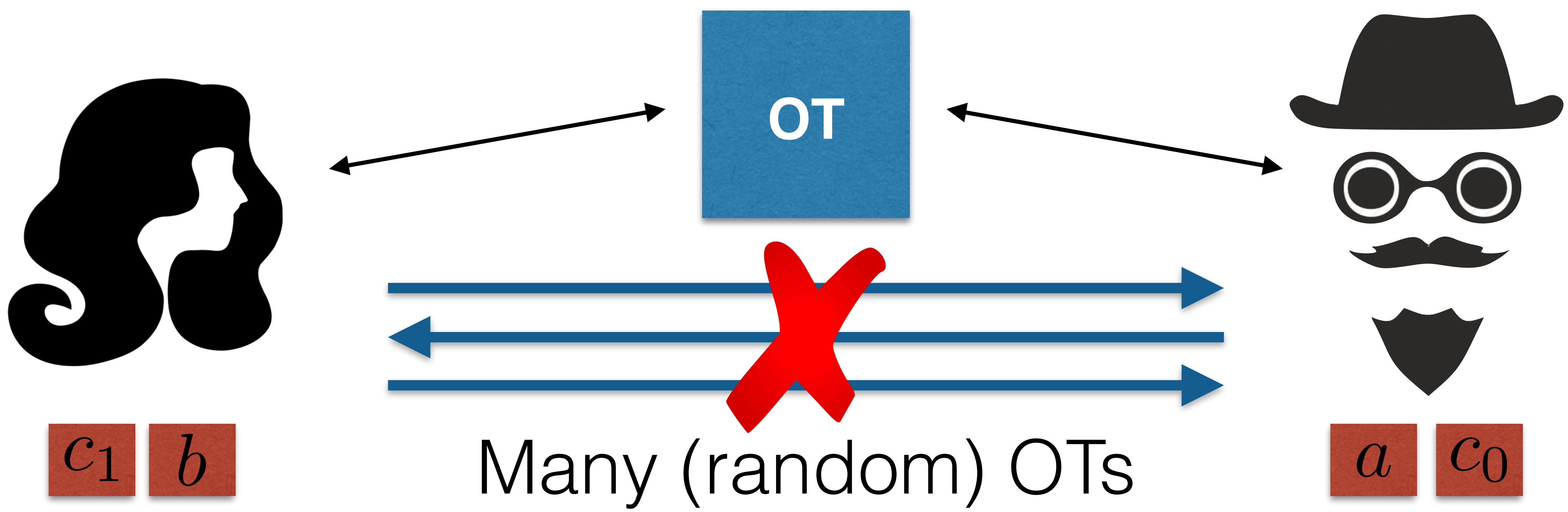
GMW Protocol. 2 OT per AND gate

Problem. OT is expensive (public-key primitive)

OT Extension

[Bea96, IKNP03]

Hybrid Approach. Few base OTs + symmetric crypto

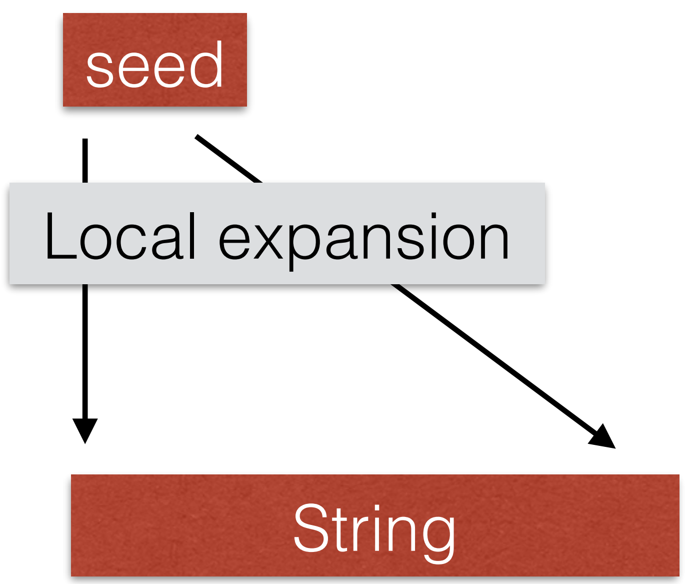


Problem. Communication & storage linear in #OTs

Silent OT Extension. Communication & storage *sublinear*

Pseudorandom Generator

$$\mathsf{PRG} : \{0,1\}^n \mapsto \{0,1\}^m \text{ with } m \gg n$$



Short seed

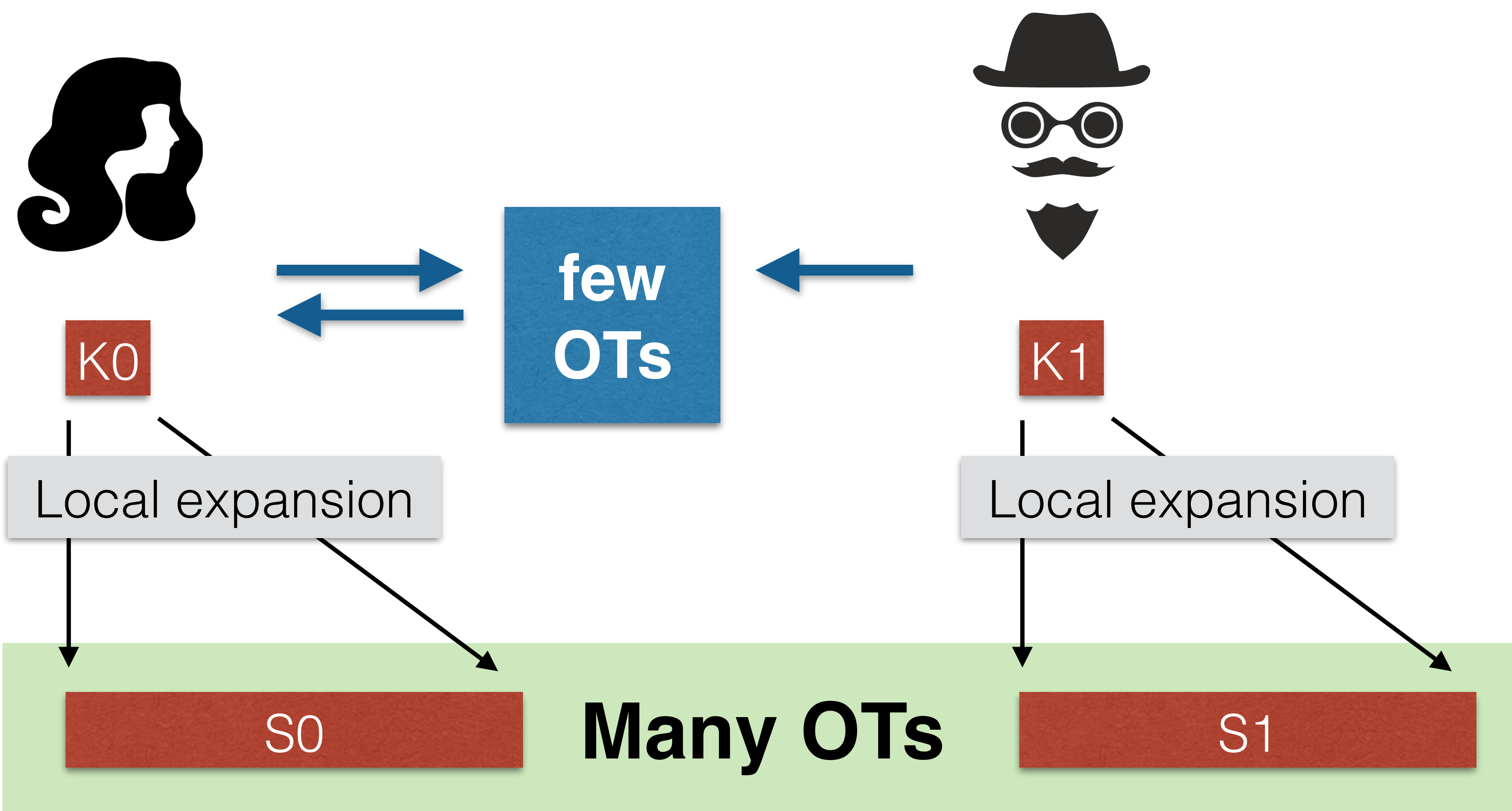
Long, pseudorandom string

Formally, $\forall$ PPT $\mathcal{A}$,

$$|\mathsf{Pr}[y \leftarrow_\$ \{0,1\}^m \; : \; \mathcal{A}(y) = 1]| - \mathsf{Pr}[x \leftarrow_\$ \{0,1\}^n, y \leftarrow \mathsf{PRG}(x) \; : \; \mathcal{A}(y) = 1]| \approx 0$$

Pseudorandom Correlation Generator (PCG)

[B**C**GI18,B**C**GIKS19]



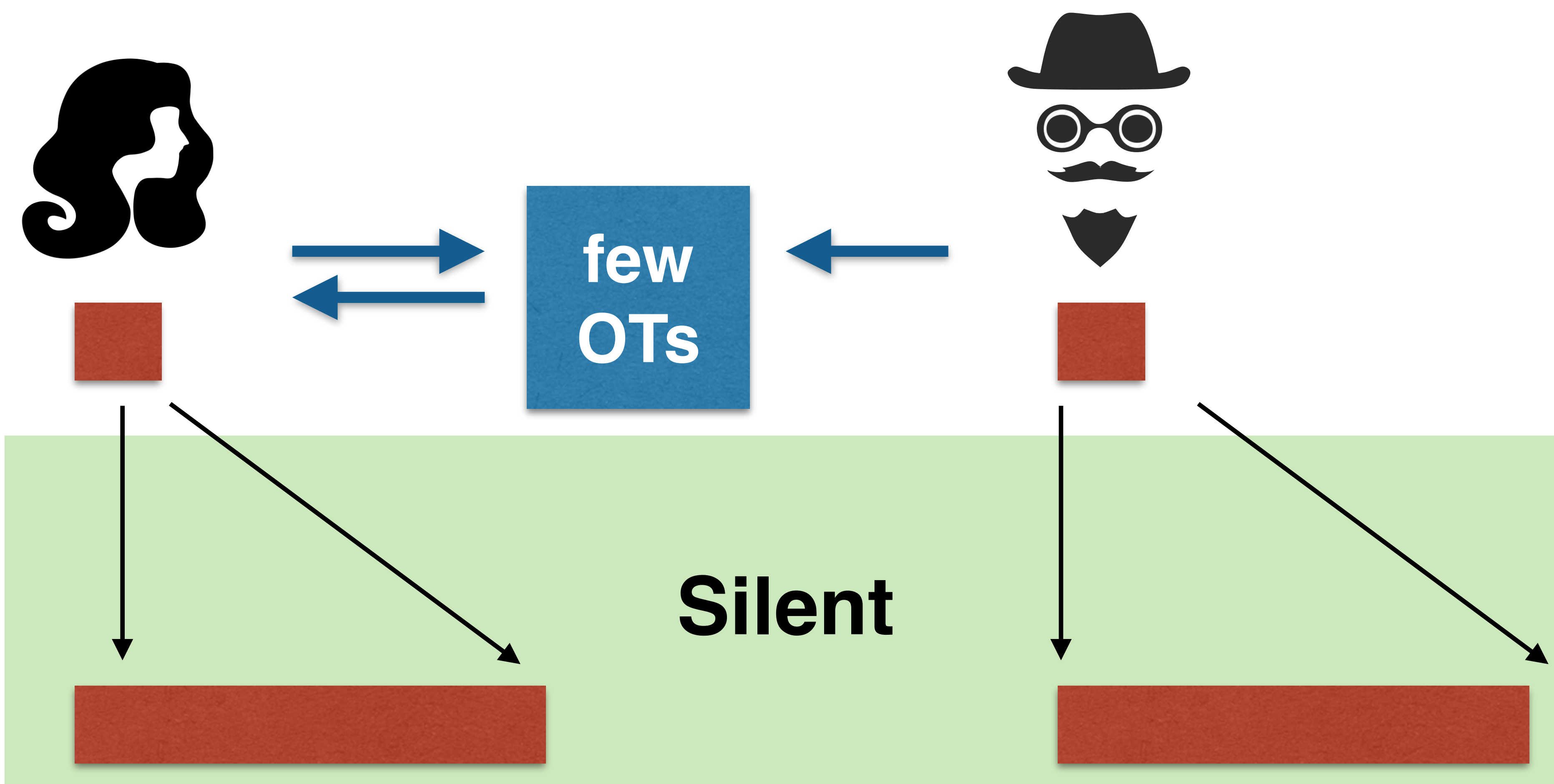
Short, correlated seeds

Long, correlated strings

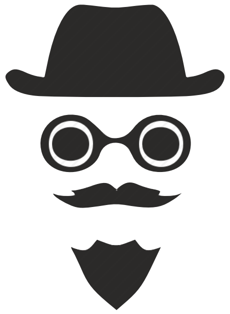
Pseudorandom Correlation Generator (PCG)

[BCGI18,BCGIKS19]

PCGs have the **silent** feature.



PCG Definition



$$(k_0, k_1) \leftarrow \mathsf{Gen}(1^\lambda)$$

$R_0 \leftarrow \mathsf{Expand}(k_0)$ $\quad$ $R_1 \leftarrow \mathsf{Expand}(k_1)$

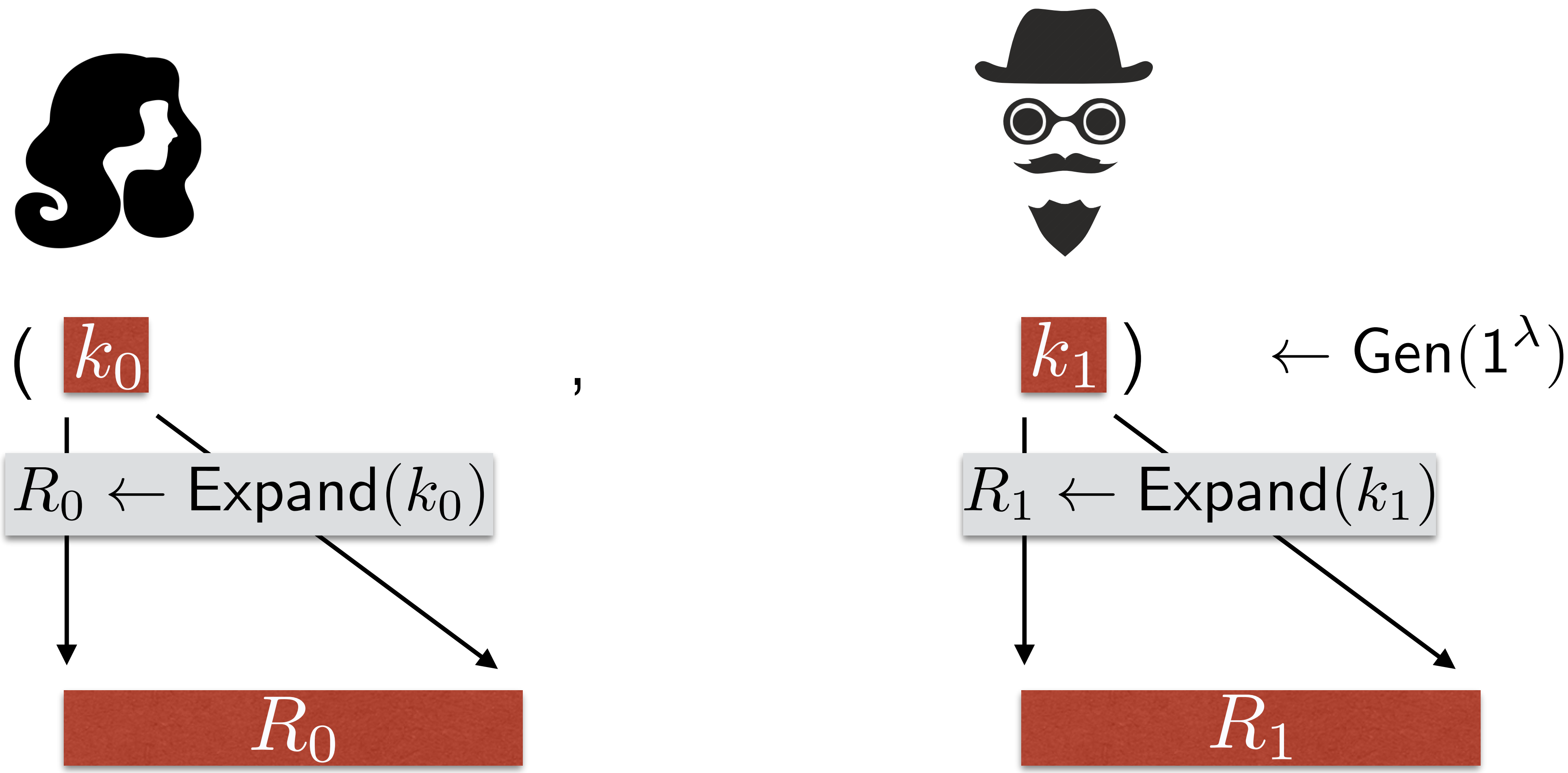
R_0 $\quad$ R_1

Correctness. $\mathsf{rel}(R_0, R_1) = 1$

Security. $(k_0, R_1) \approx (k_0, [R_1 \text{ random s.t. } \mathsf{rel}(R_0, R_1) = 1])$

+ Expand is a PRG

PCG Definition

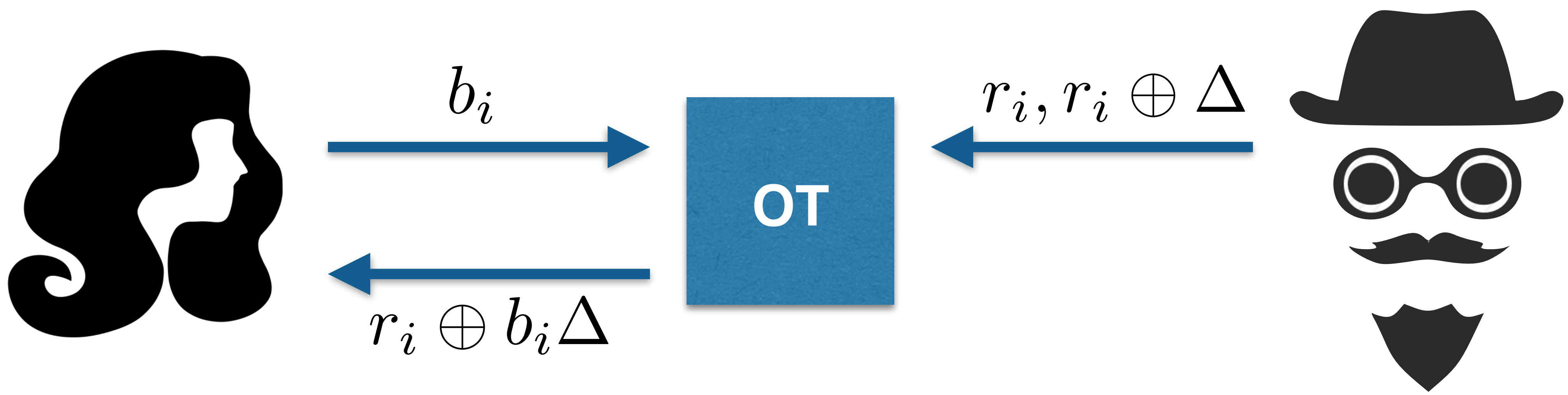


Plug-and-play: can we use PCG to generate preprocessing material?
We show several impossibility results (e.g. randomized functionalities) and some positive results (**corruptible** functionalities)

Towards Silent OT Extension

[CCS:B**C**GIKS18, CRYPTO:B**C**GIKS19, CCS:B**C**GIKRS19]

Correlated OT:



Correlated OT + correlation-robust hash functions => OT [IKNP03]

$H(r_i \oplus b_i\Delta)$ $\quad$ $H(r_i), H(r_i \oplus \Delta)$

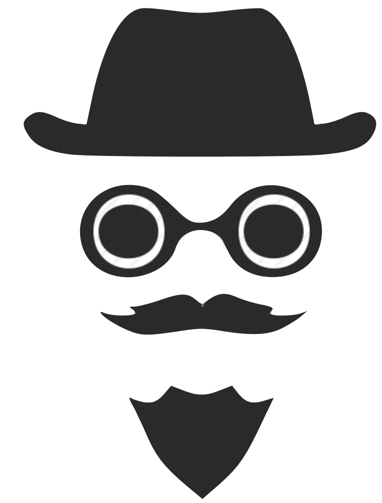
Rephrasing correlated OT:

$$
\begin{aligned}
(\vec{r} \oplus \vec{b} \cdot \Delta) \oplus \vec{r} &= \vec{b} \cdot \Delta \\
\Longrightarrow \quad \vec{q} \oplus \vec{r} &= \vec{b} \cdot \Delta
\end{aligned}
$$

PCG for Correlated OT - Strategy

Correlated OT:

$$\underbrace{\vec{q} \oplus \vec{r} = \vec{b} \cdot \Delta}_{\text{additive shares of } \vec{b} \cdot \Delta}$$



$\vec{q}, \vec{b}$ $\vec{r}, \Delta$

Goal: compressing $\vec{q}, \vec{b}$ and $\vec{r}, \Delta$

Roadmap:

PPRFs	Summation	Syndrome decoding
PCG for a unit vector $\vec{b} \cdot \Delta$	PCG for a sparse $\vec{b} \cdot \Delta$	PCG for a pseudorandom $\vec{b} \cdot \Delta$

First Tool: Puncturable PRFs

PRF:

A function sampled from $\mathcal{F} = \{F_k\}_k$ is indistinguishable from a truly random function (via black-box access)

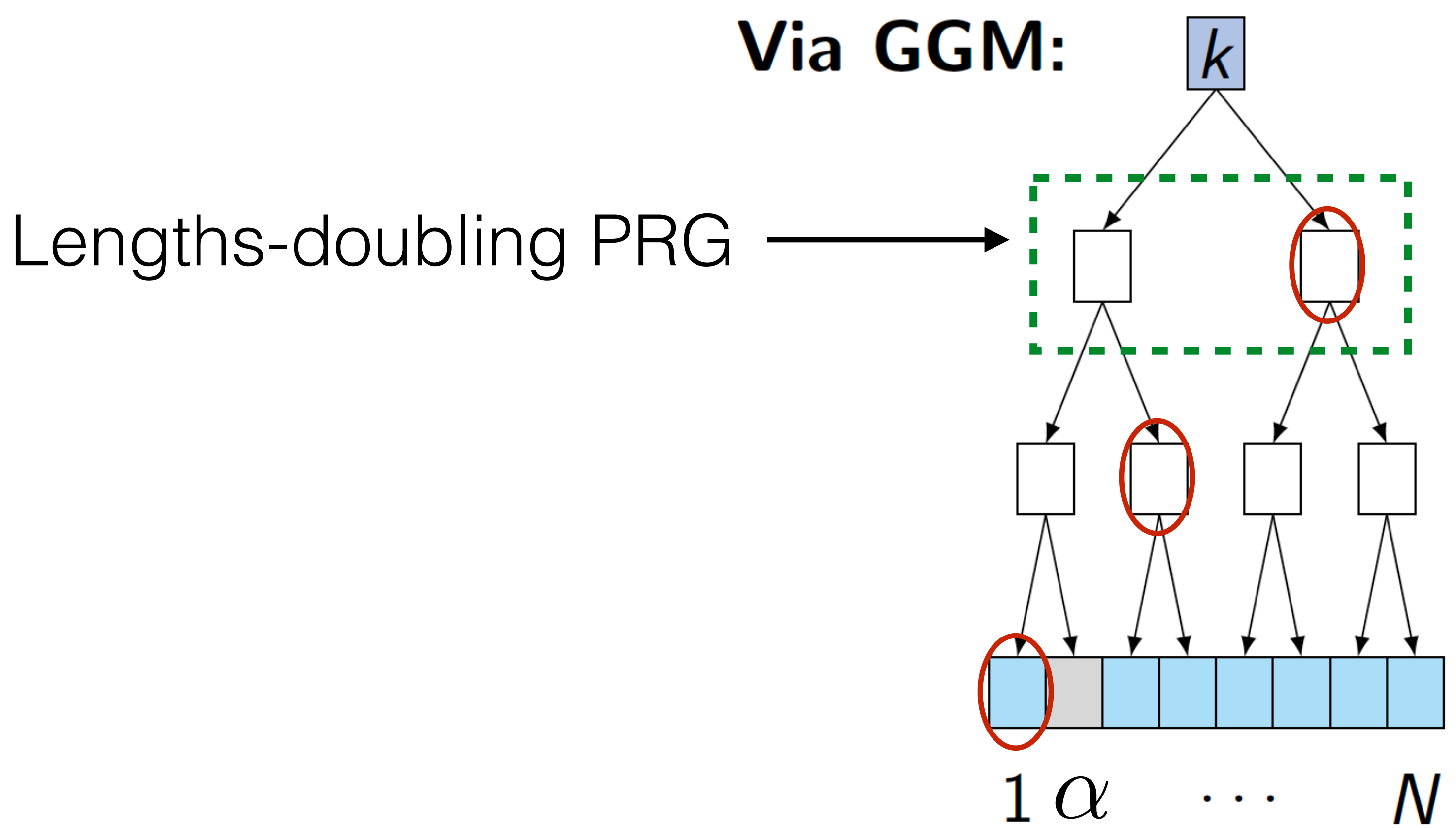
First Tool: Puncturable PRFs

Puncturable PRF (PPRF):

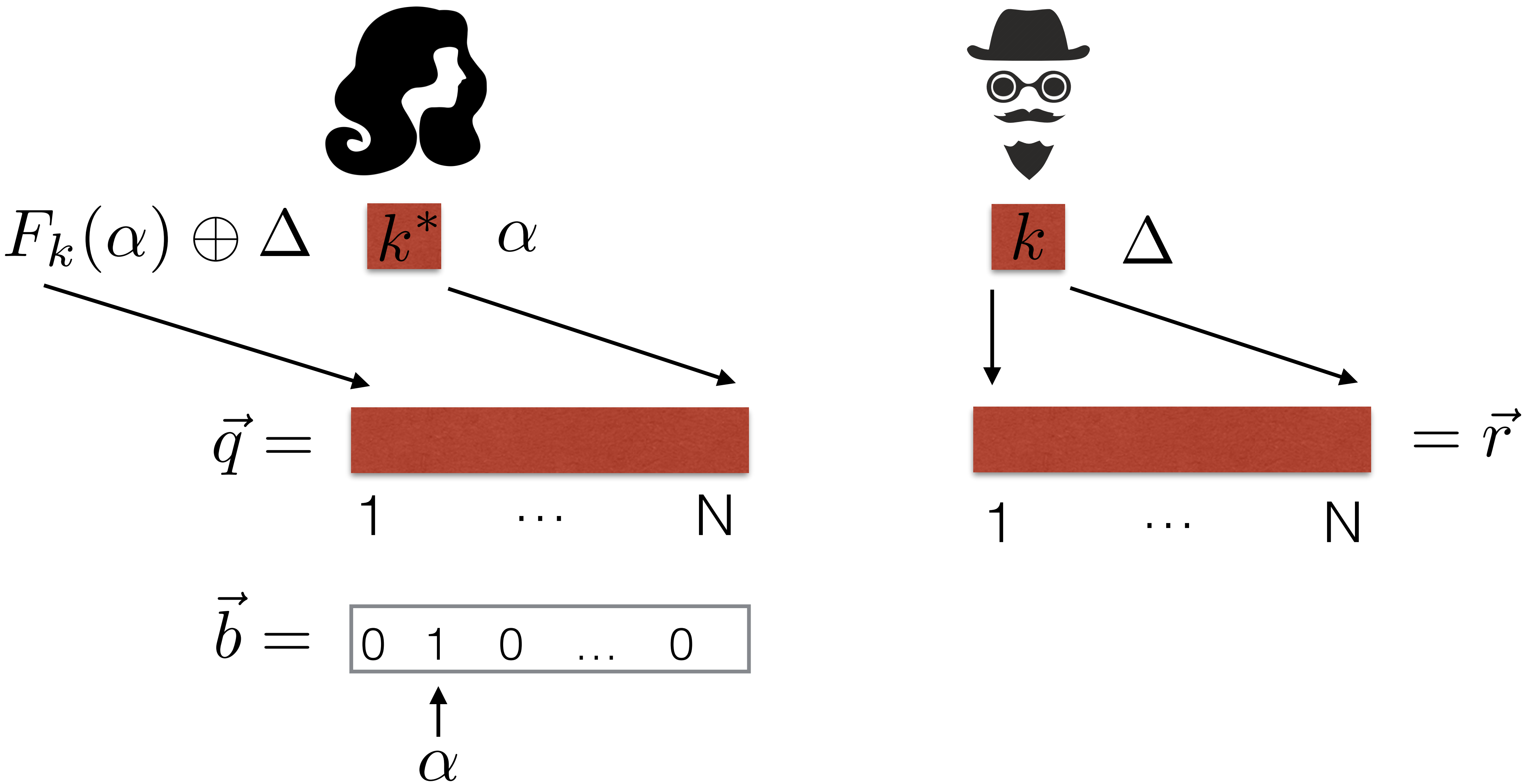
$F_k \colon \{1, \ldots, N\} \to \mathbb{F}_{2^\lambda}$

- $k \rightsquigarrow F_k(x)$ for all x
- $k^\star \rightsquigarrow F_k(x)$ for all $x \neq \alpha$

Via GGM:



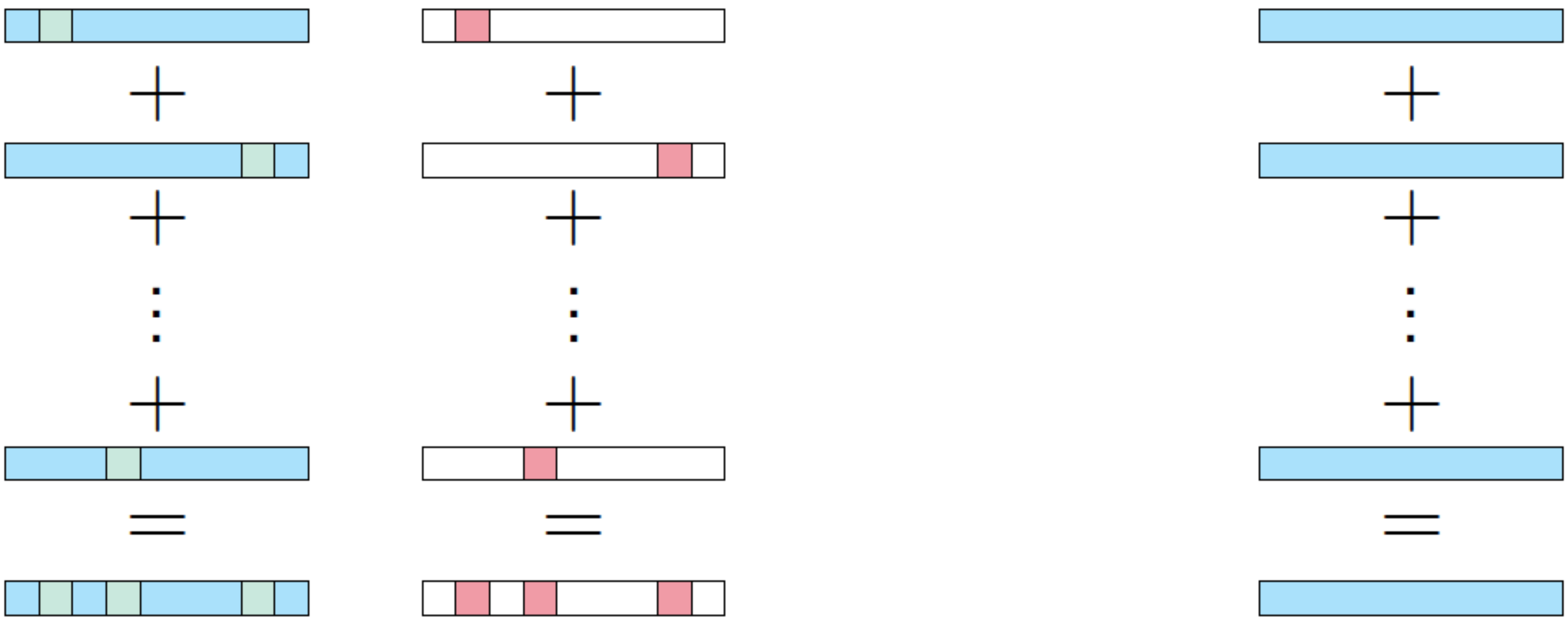
PCG for Unit Vectors via PPRFs



$$\vec{q} \oplus \vec{r} = \vec{b} \cdot \Delta$$

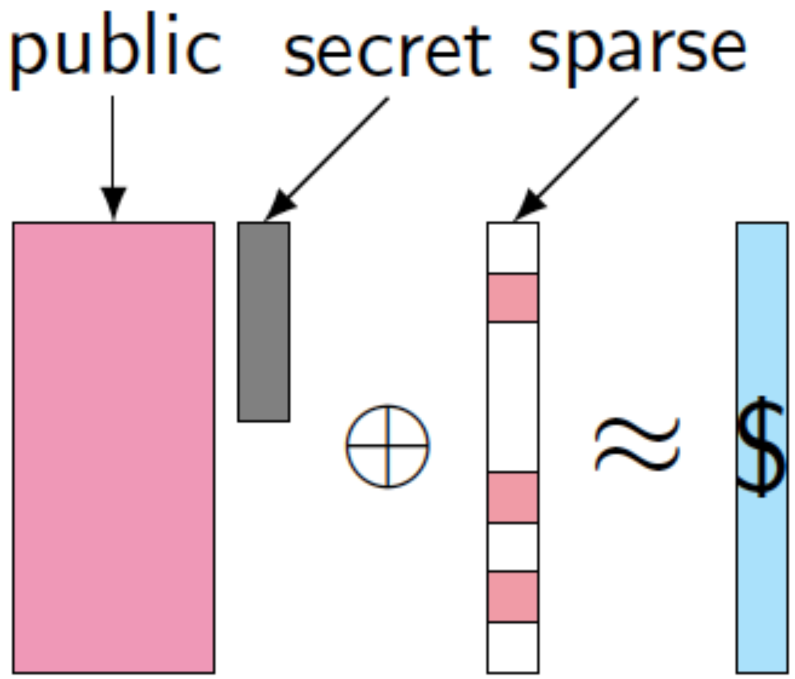
From Unit Vectors to Sparse Vectors via Addition

PCG for unit vectors => PCG for weight-t vectors
by t-fold repetition of the unit vector version:

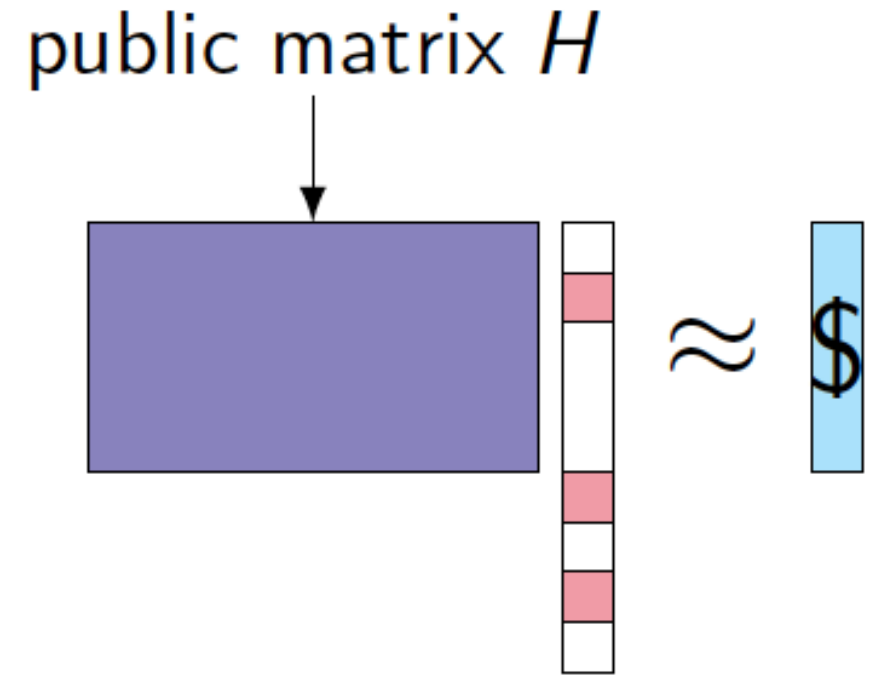


Syndrome Decoding (SD)

Primal:



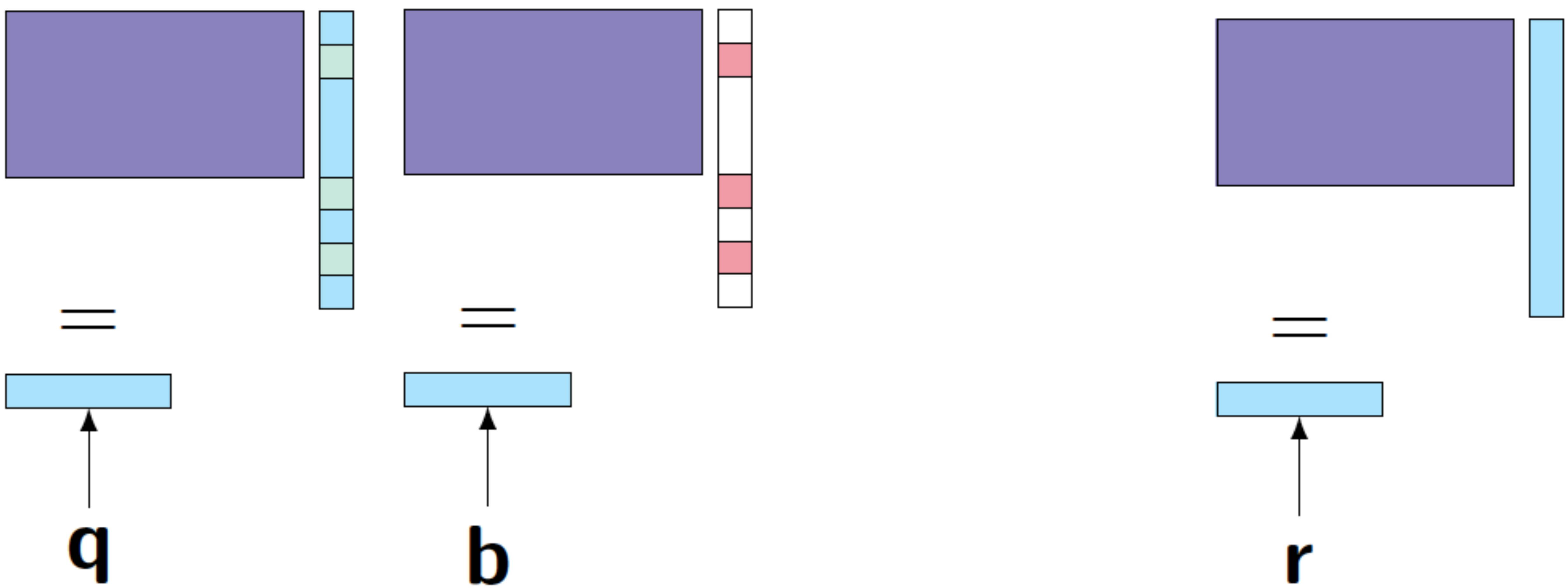
Dual:



Notes:

- Security is similar to PQ cryptosystems e.g. BIKE, HQC [AAB+19, ABB+19]
- Not known to imply PKE for certain noise rates

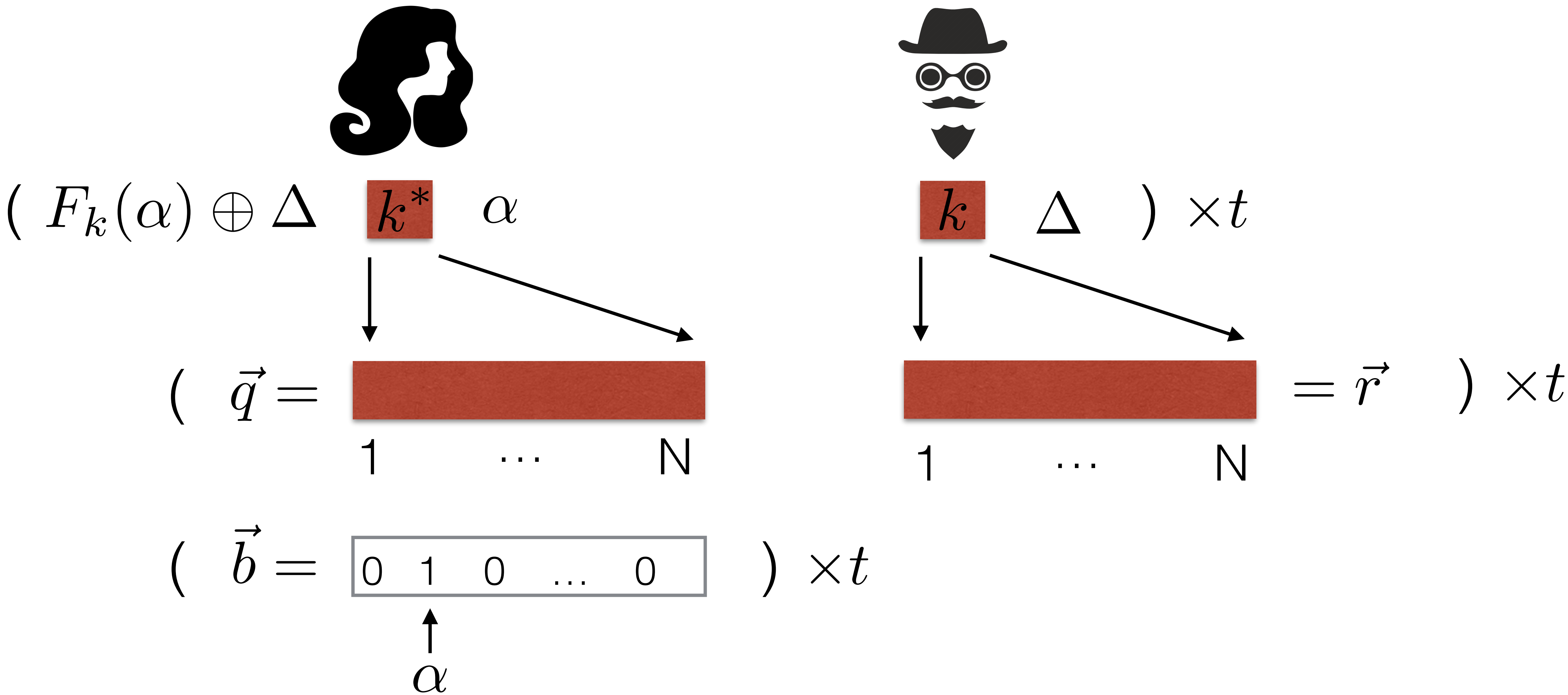
From Sparse to Pseudorandom via SD



By correctness of DPF + linearity of addition + linearity of SD:

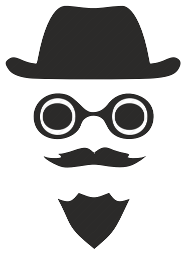
$$\vec{q} \oplus \vec{r} = \vec{b} \cdot \Delta$$

Wrapping Up - PCG for Correlated OT



Then sum and multiply by public matrices to get dense vectors
Security: provably reduces to syndrome decoding

Wrapping Up - PCG for Correlated OT



$(\; F_k(\alpha) \oplus \Delta \quad \boxed{k^*} \quad \alpha \qquad\qquad \boxed{k} \quad \Delta \;) \times t$

$(\; \vec{q} = \; \ldots \; = \vec{r} \;) \times t$

1 $\cdots$ N

1 $\cdots$ N

$(\; \vec{b} = \boxed{0 \quad 1 \quad 0 \quad \ldots \quad 0} \;) \times t$

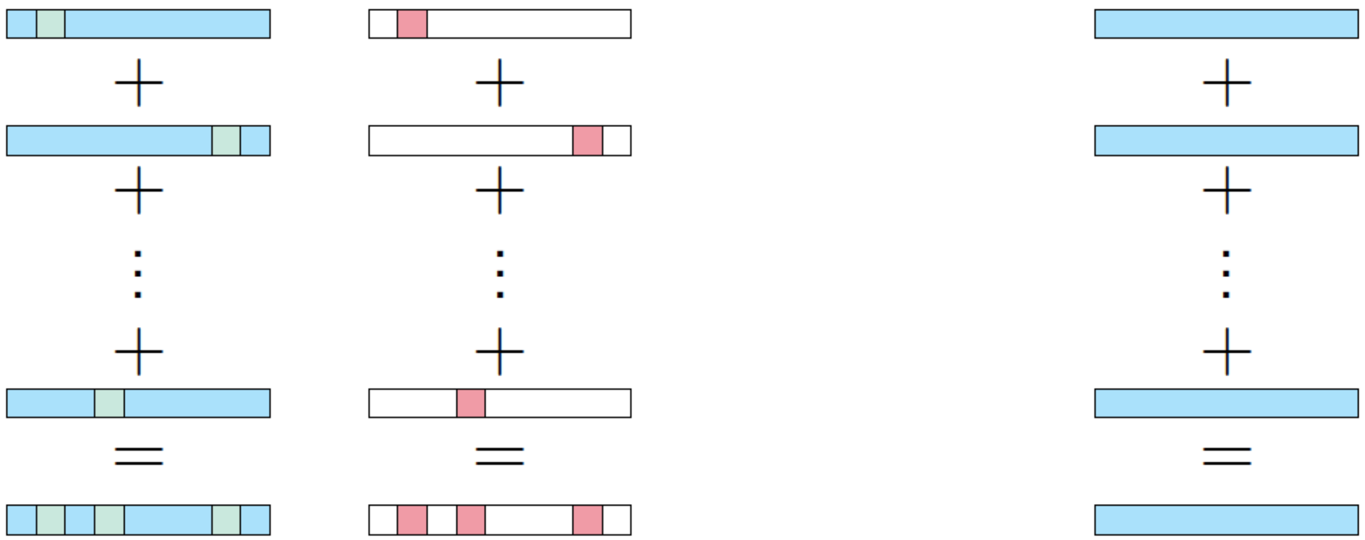
α

Correlated OT + correlation-robust hash functions => OT [IKNP03]

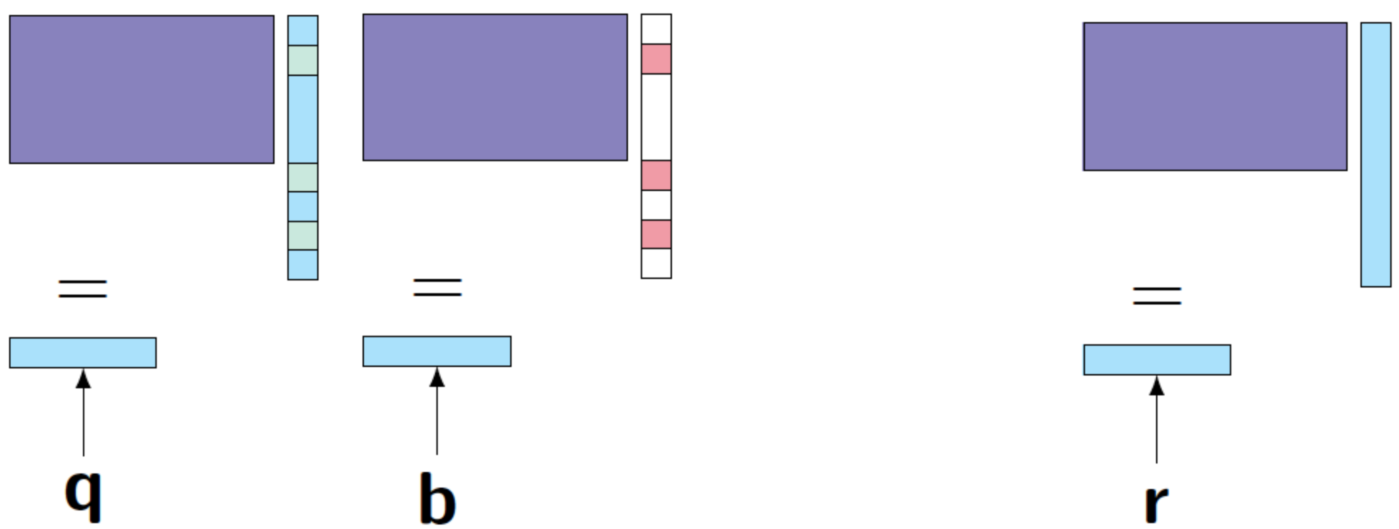
technicality: must use extension fields

$H(r_i \oplus b_i \Delta)$ $\qquad$ $H(r_i), H(r_i \oplus \Delta)$

Optimizing under Stronger Assumptions

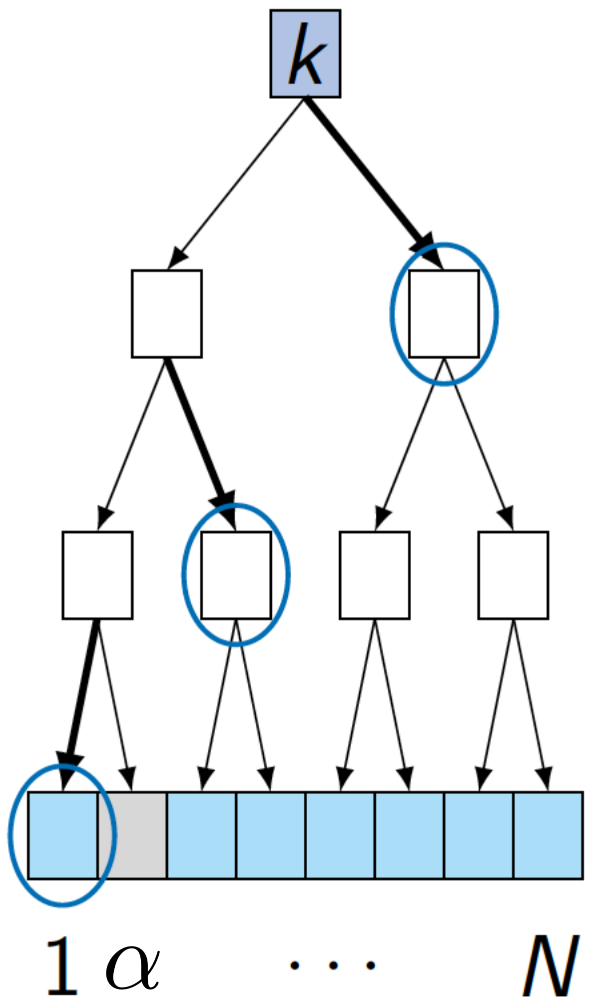


Idea 1:
Regular Syndrome Decoding



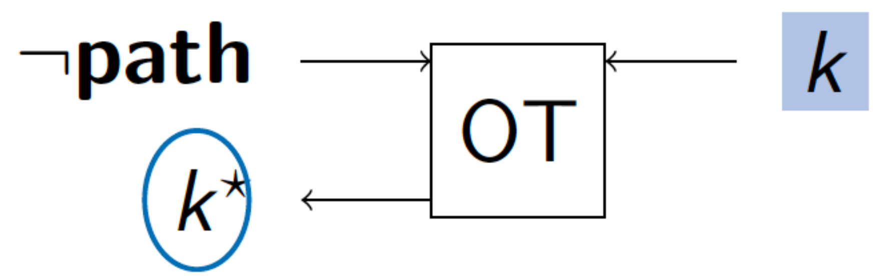
Idea 2:
Quasi-Cyclic Syndrome Decoding

Distributing the Seed Generation



Strategy: (based on [Ds17])

- Sender chooses k
- Receiver receives $k^\star$ via chosen OTs:



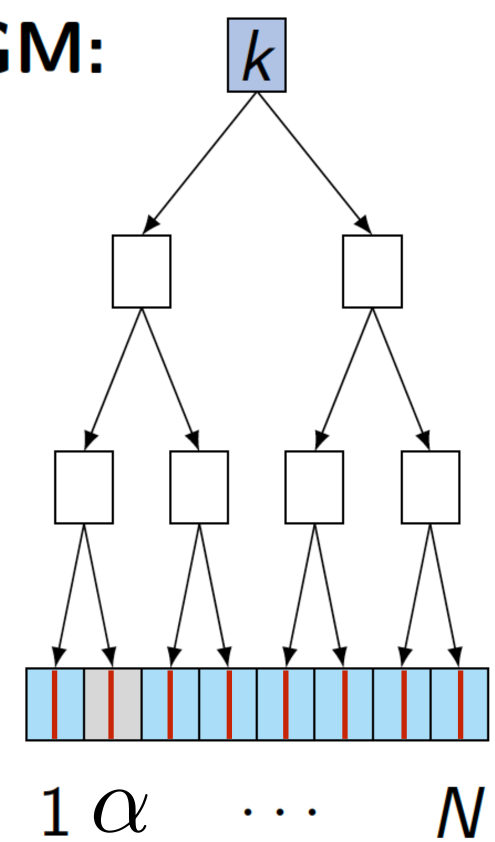
Main observation:

- Receiver knows α
- $\rightsquigarrow$ OTs can be executed *in parallel!*

Malicious Security

Core idea: add consistency check inside the PPRF
$\Longrightarrow$ extend the domain size from N to 2N,
use a hash of the odd values to check the punctured key

Via GGM:



Comparison - OT Extension, 128 bits Security

Reference	Rounds	Comm. per random OT	Silent	Active	Based on
[Bea96]	2	poly	✗	✗	OWF
[IKNP03; ALSZ13; KOS15]	$3^\star$	128	✗	✓	crh
[KK13] (short strings)	3	≈ 78	✗	✗	crh
[B**C**GIKS19]	$\log N$	$0 - 3$	✓	✗	LPN, crh
[B**C**GIKRS19]	$2^\star$	0.1	✓	✓	LPN, crh

*Fiat-Shamir for active security, crh = correlation robust hash function

- Semi-honest 2-PC w/ 4.2 bits per AND, $30\times$ less than [DKSSZZ17]
- Improves PSI, malicious MPC
- Useful for non-interactive secure comp. [IKOPS11; AMPR14; MR17]

Open Problems, Ongoing Works

- Multiparty setting [CRYPTO:B**C**GIKS20]
- Linear time computation *(ongoing work)*
- Pseudorandom correlation *functions* [FOCS:B**C**GIKS20]
- Large fields [CRYPTO:B**C**GIKS20]

…

Thank you for your attention

Questions?