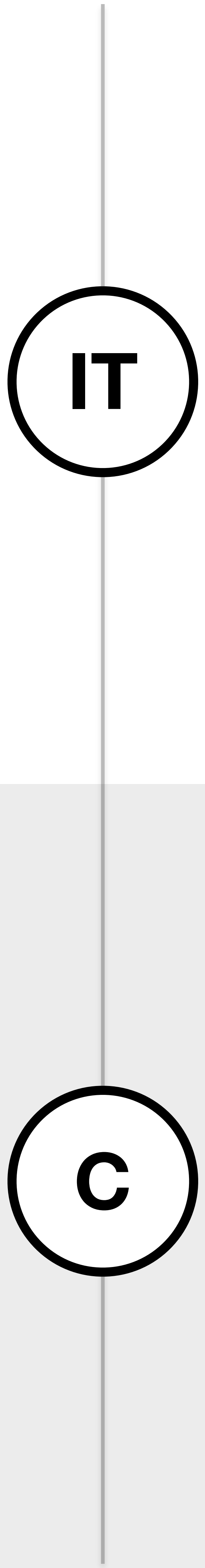
Pseudorandom Correlation Functions from Variable-Density LPN

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

Correlated Randomness in Cryptography

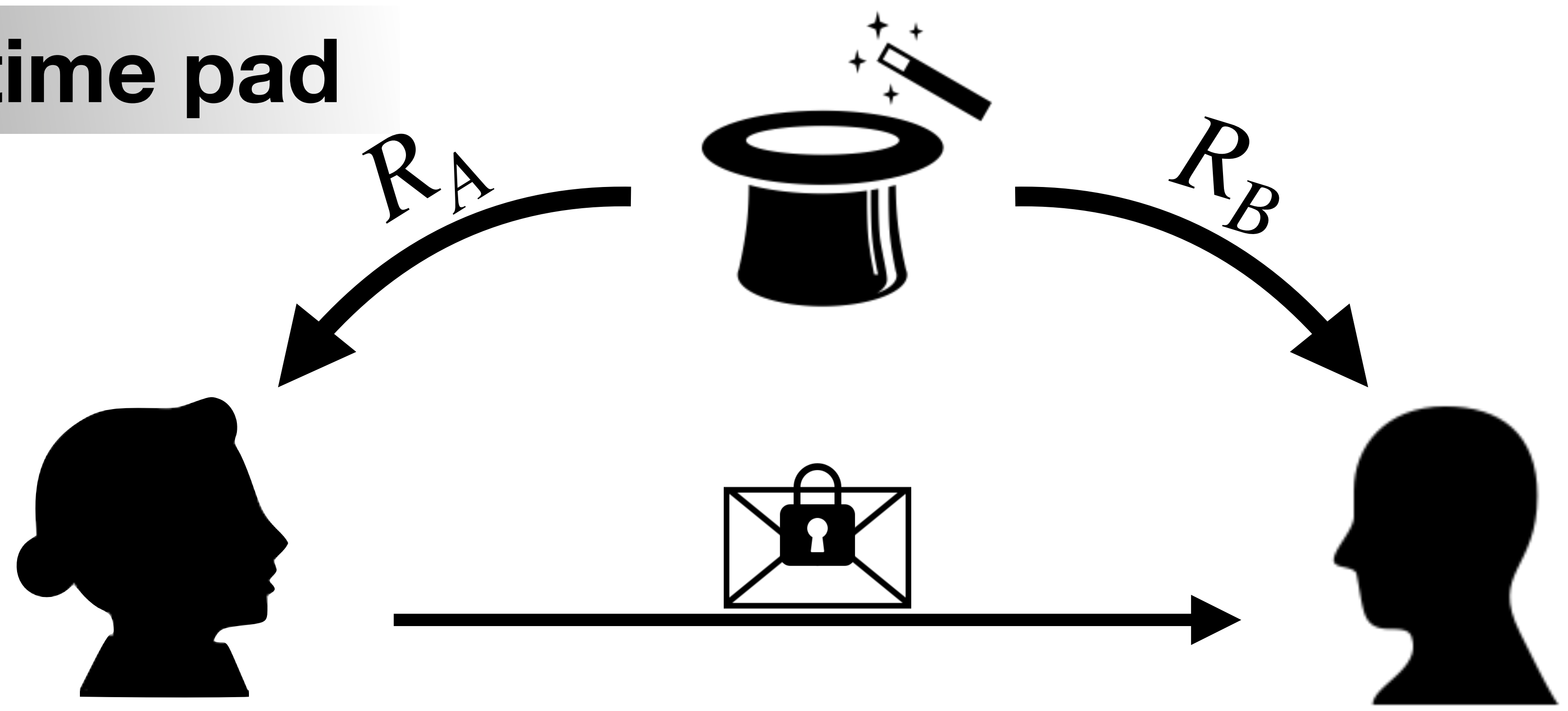
A source of secret *correlated* randomness is an extremely useful resource in secure protocols:



Correlated Randomness in Cryptography

A source of secret *correlated* randomness is an extremely useful resource in secure protocols:

One-time pad



Correlation: $R_A = R_B$

(Equality correlation)

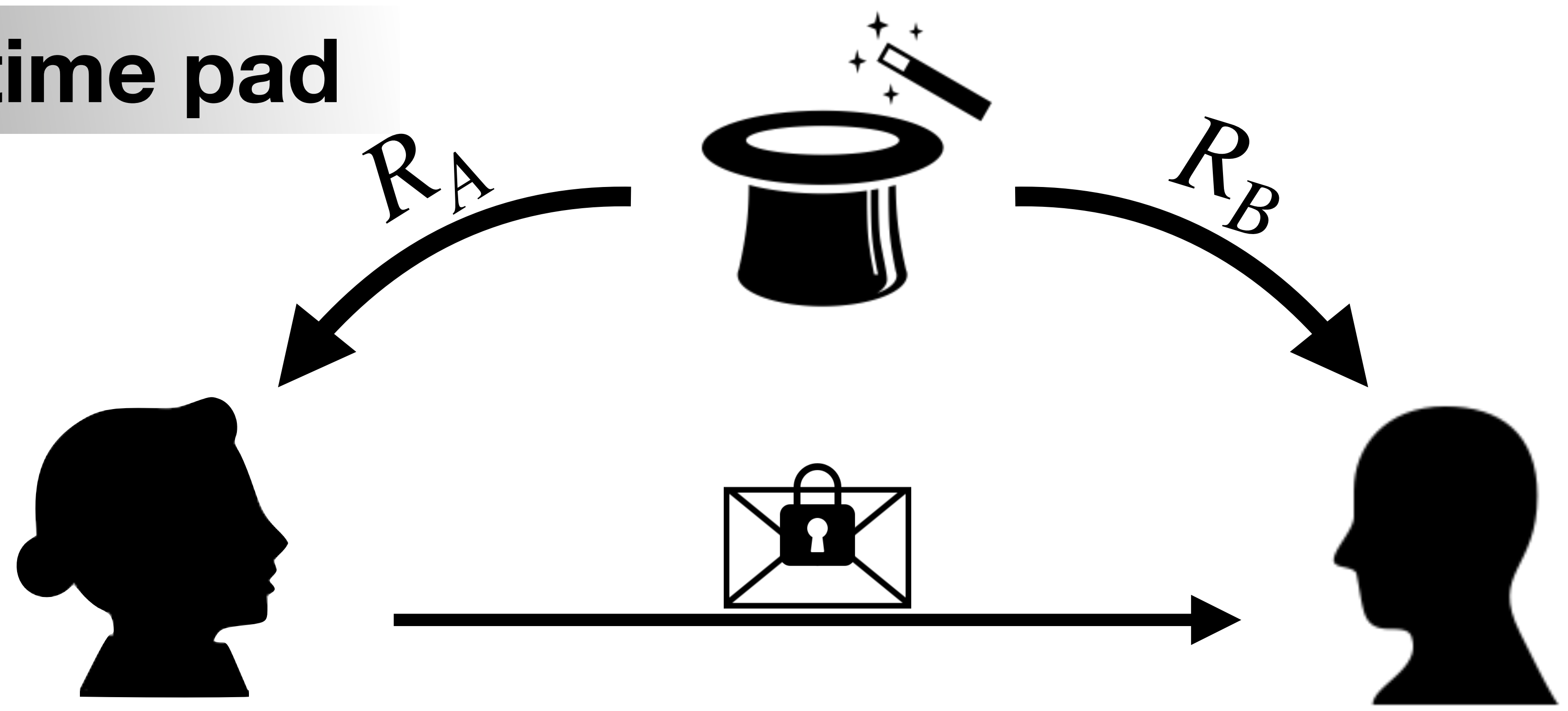
IT

C

Correlated Randomness in Cryptography

A source of secret *correlated* randomness is an extremely useful resource in secure protocols:

One-time pad

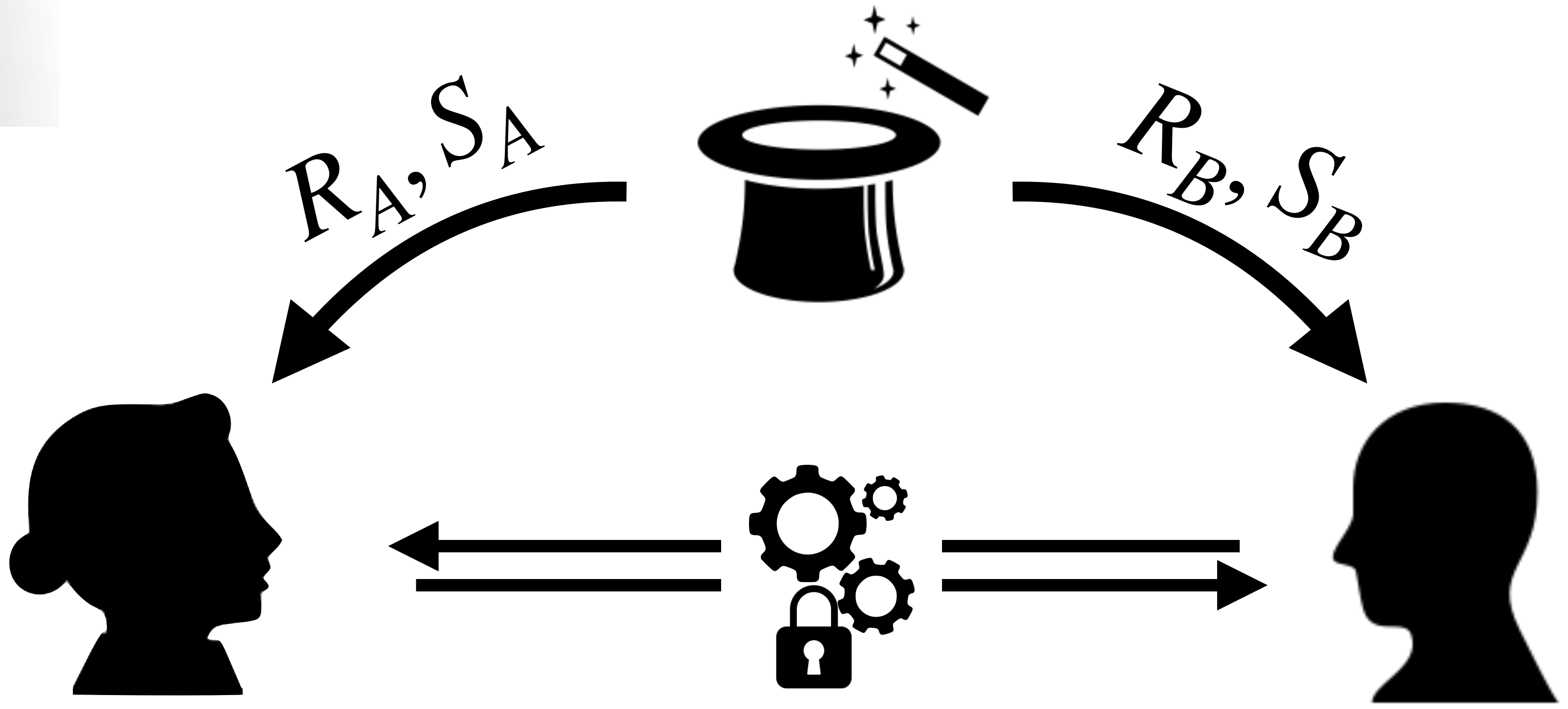


Correlation: $R_A = R_B$

(Equality correlation)

IT

GMW



Correlation: $R_A + R_B = S_A \wedge S_B$

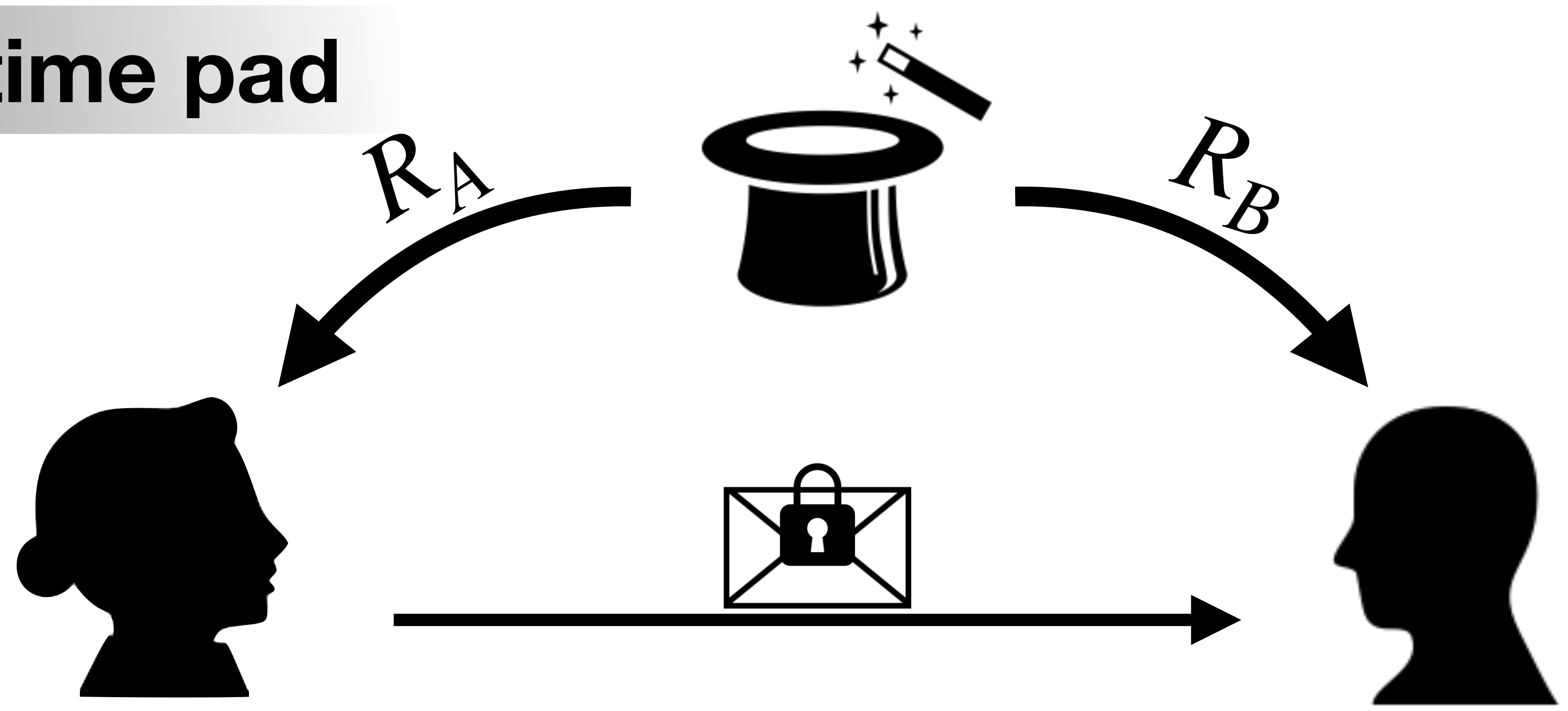
(Oblivious transfer correlation)

C

Correlated Randomness in Cryptography

A source of secret *correlated* randomness is an extremely useful resource in secure protocols:

One-time pad

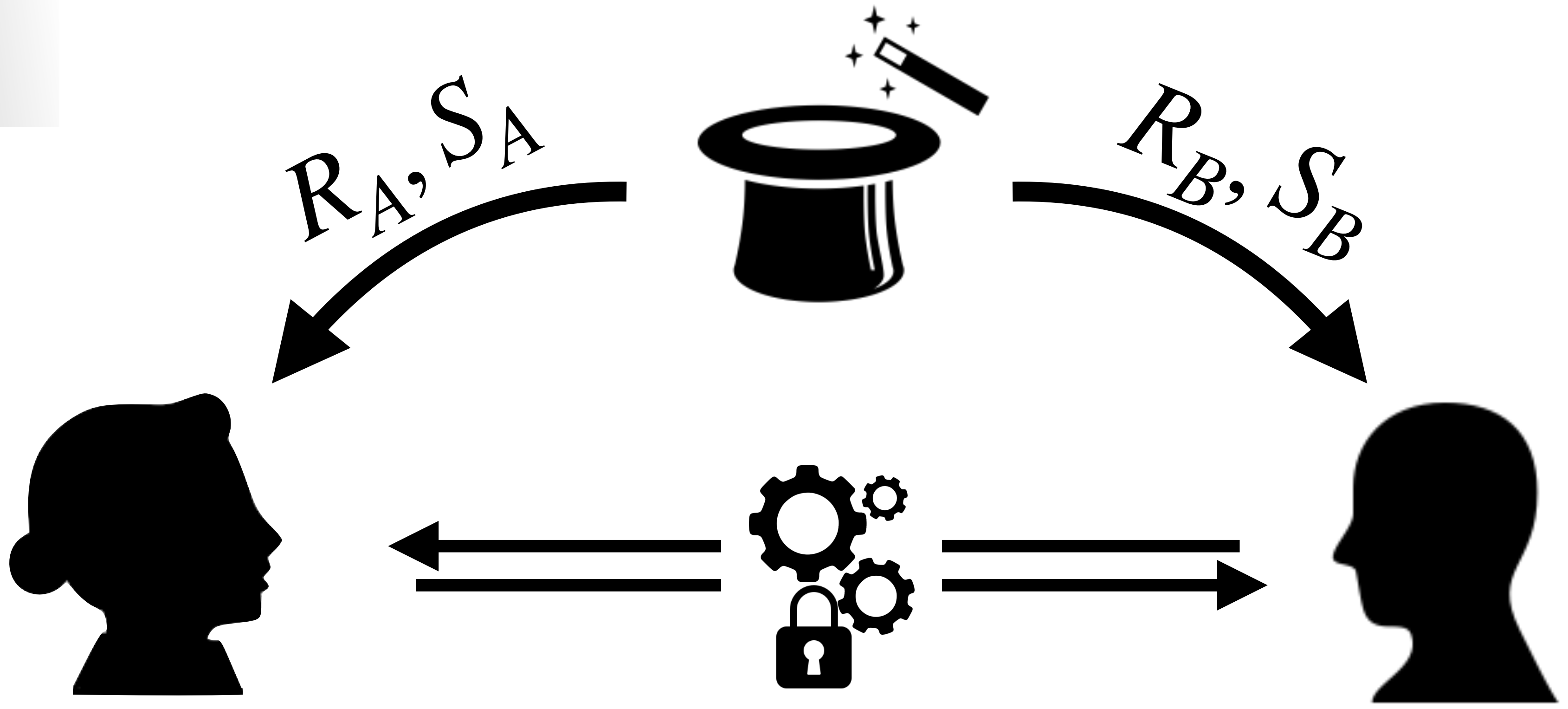


Correlation: $R_A = R_B$

(Equality correlation)

IT

GMW



Correlation: $R_A + R_B = S_A \wedge S_B$

(Oblivious transfer correlation)

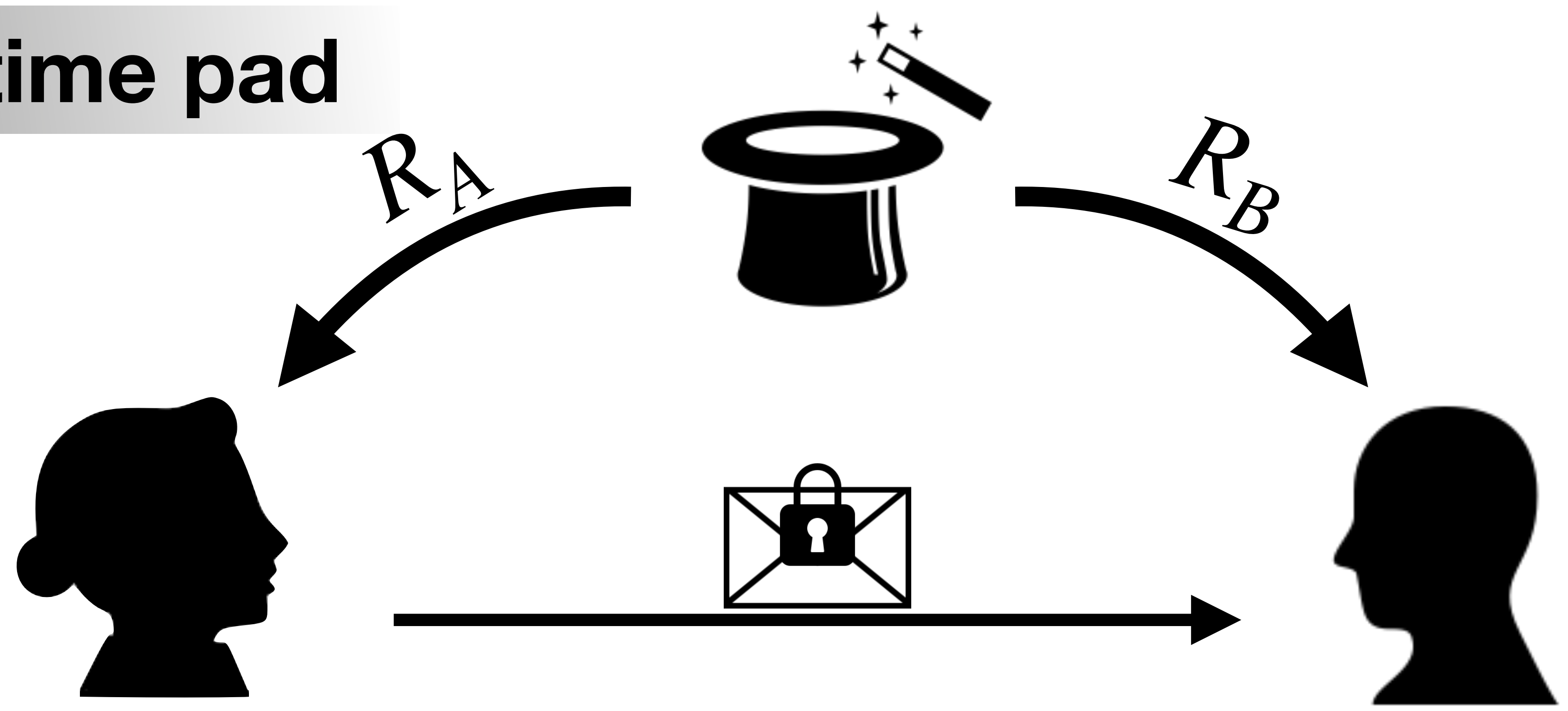
In the computational world, can we *compress* correlated randomness?

C

Correlated Randomness in Cryptography

A source of secret *correlated* randomness is an extremely useful resource in secure protocols:

One-time pad

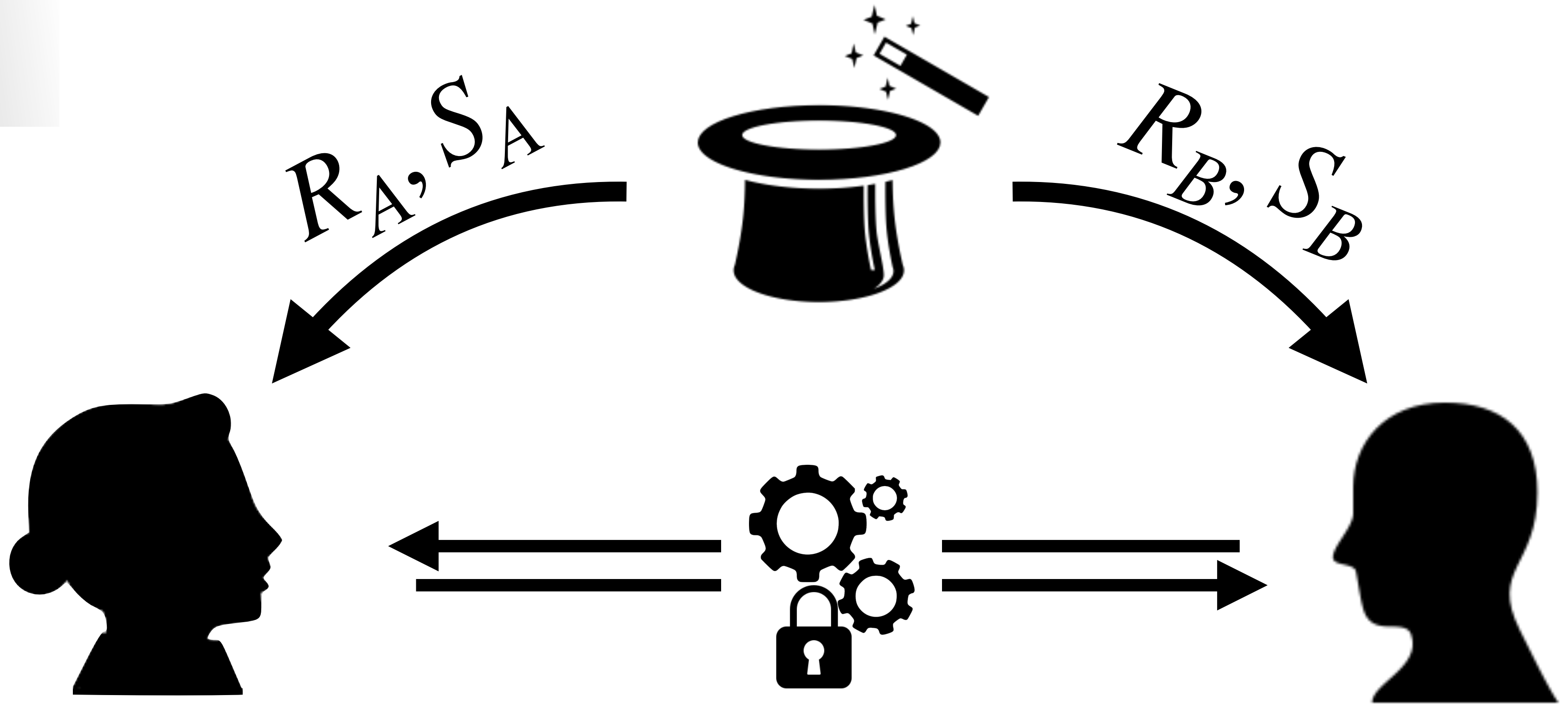


Correlation: $R_A = R_B$

(Equality correlation)

IT

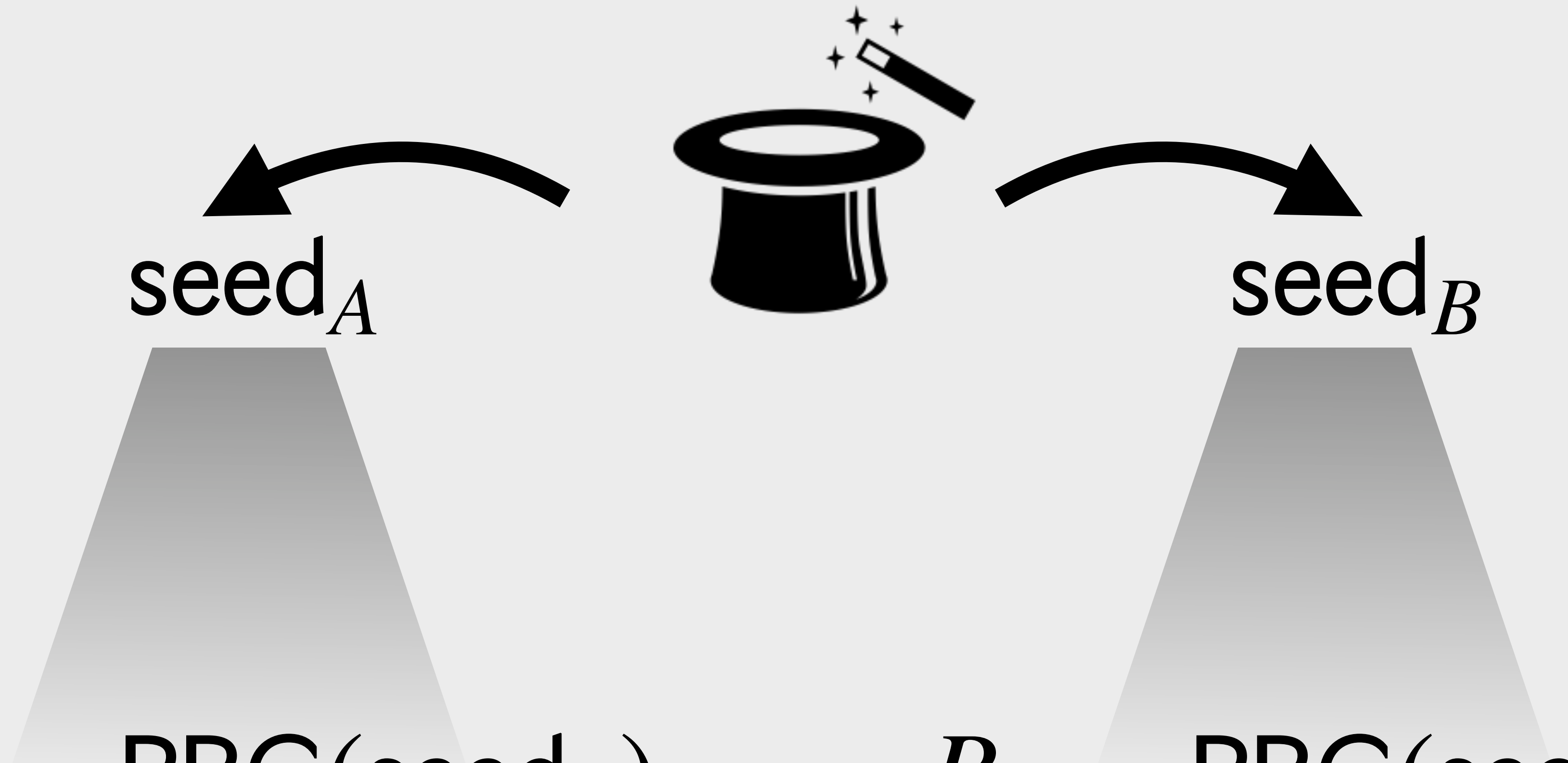
GMW



Correlation: $R_A + R_B = S_A \wedge S_B$

(Oblivious transfer correlation)

Equality correlations can be *compressed* using a PRG:



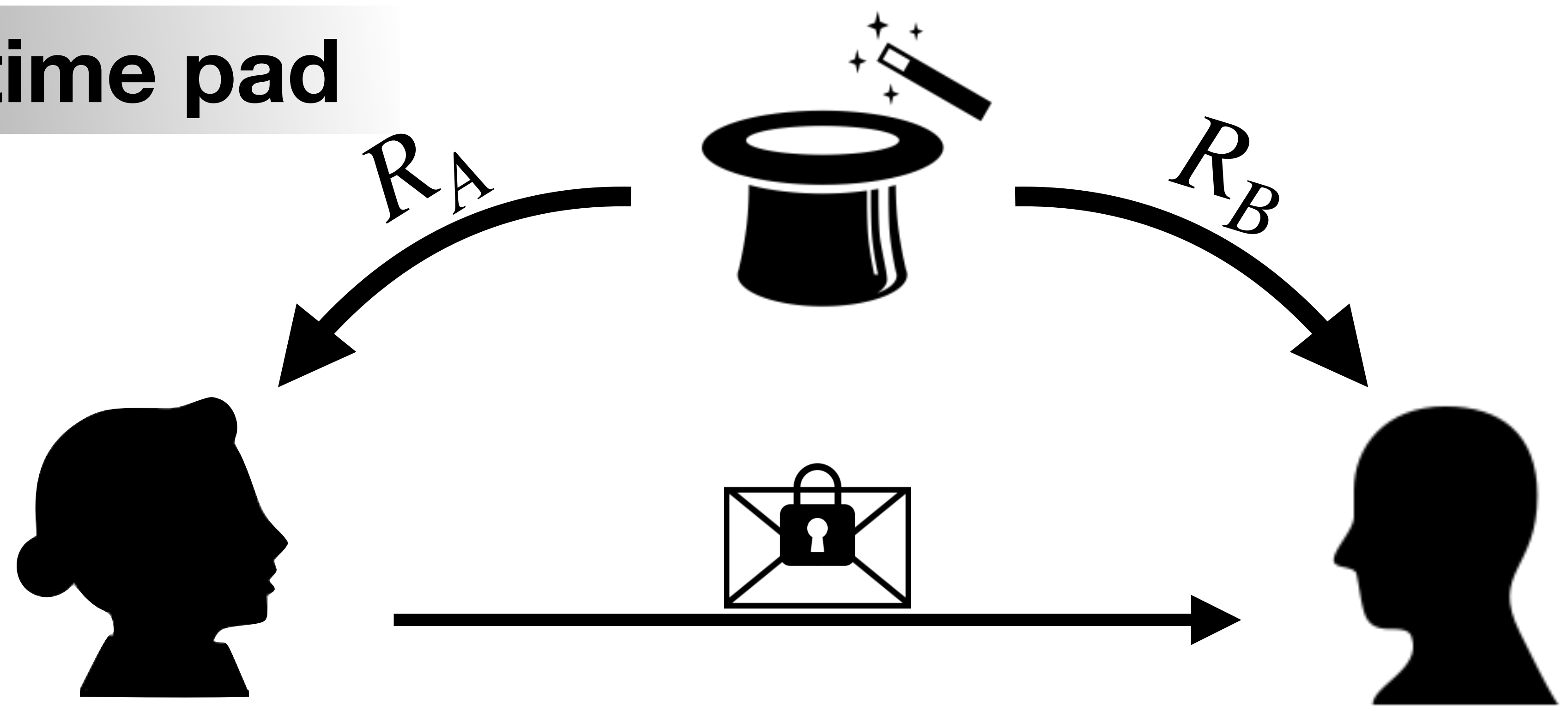
$R_A = \mathsf{PRG}(\mathsf{seed}_A)$ $\quad$ $R_B = \mathsf{PRG}(\mathsf{seed}_B)$

C

Correlated Randomness in Cryptography

A source of secret *correlated* randomness is an extremely useful resource in secure protocols:

One-time pad

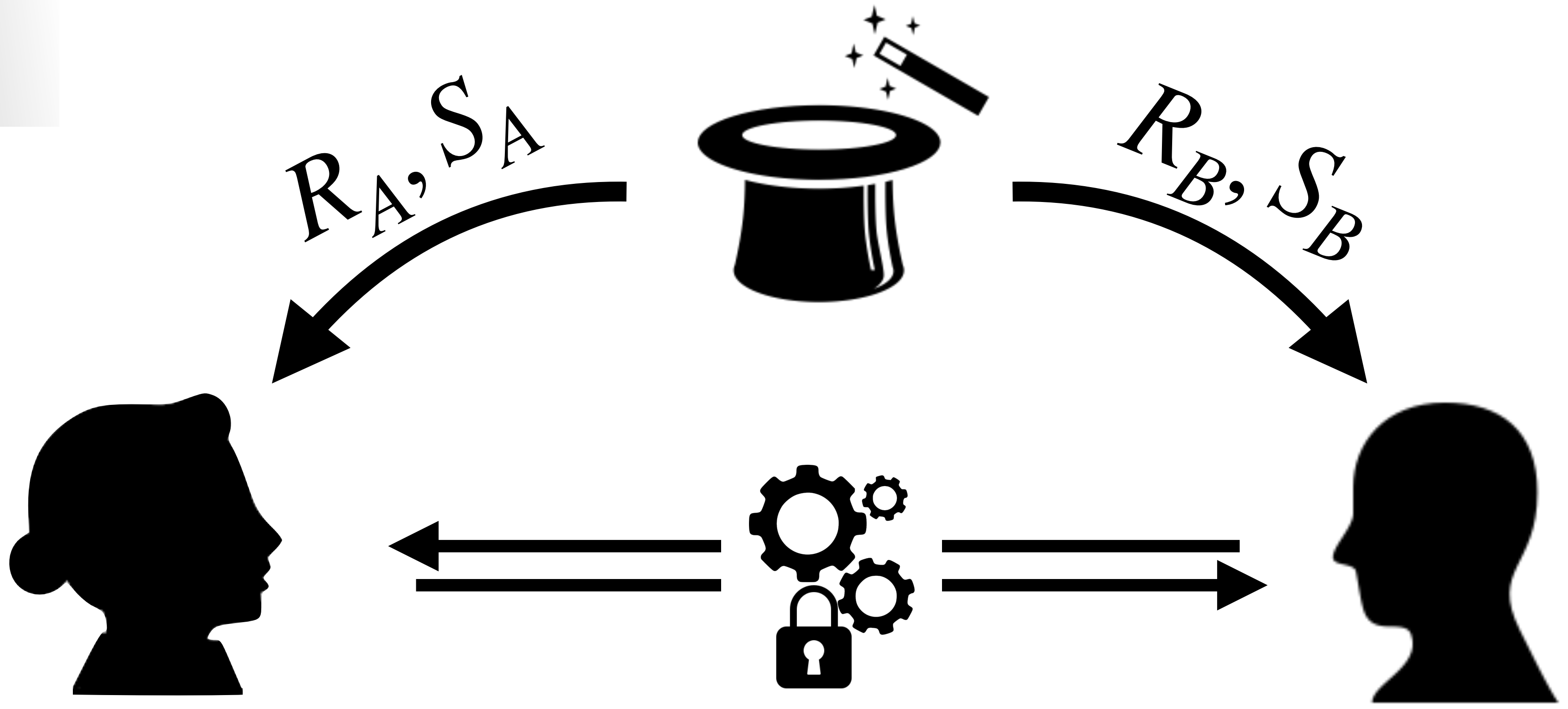


Correlation: $R_A = R_B$

(Equality correlation)

IT

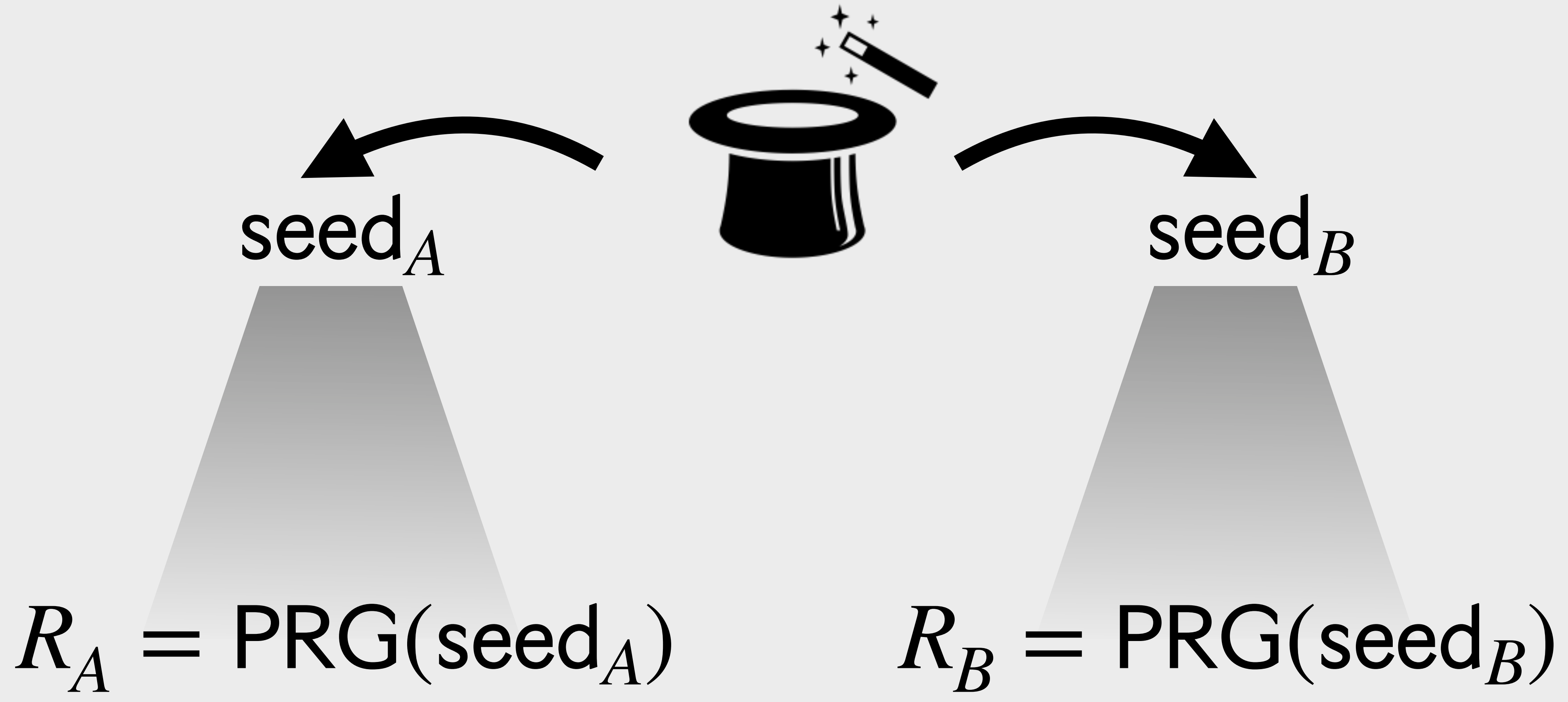
GMW



Correlation: $R_A + R_B = S_A \wedge S_B$

(Oblivious transfer correlation)

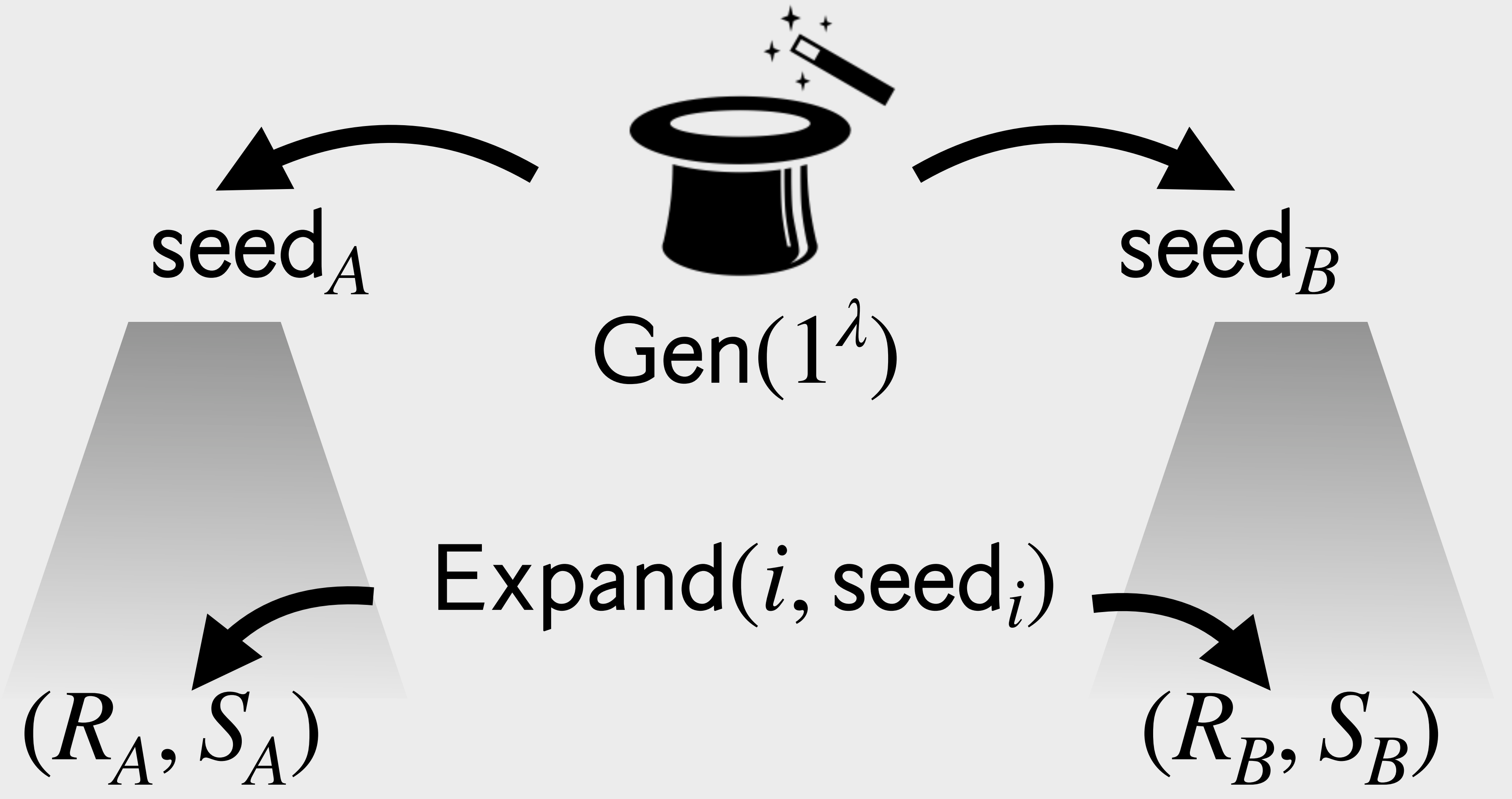
Equality correlations can be *compressed* using a PRG:



$R_A = \mathsf{PRG}(\mathsf{seed}_A)$ $\quad$ $R_B = \mathsf{PRG}(\mathsf{seed}_B)$

C

Can OT correlations be *compressed* using a PCG?


seed_A
$\mathsf{Gen}(1^\lambda)$
seed_B
$\mathsf{Expand}(i, \mathsf{seed}_i)$
(R_A, S_A)
(R_B, S_B)

Secure Computation with Silent Preprocessing

Pseudorandom correlation generator: $\mathsf{Gen}(1^\lambda) \rightarrow (\mathsf{seed}_A, \mathsf{seed}_B)$ such that **(1)** $(\mathsf{Expand}(A, \mathsf{seed}_A), \mathsf{Expand}(B, \mathsf{seed}_B))$ looks like n samples from the target correlation, and **(2)** $\mathsf{Expand}(A, \mathsf{seed}_A)$ looks ‘random conditioned on satisfying the correlation with $\mathsf{Expand}(B, \mathsf{seed}_B)$’ to Bob (similar property w.r.t. Alice).

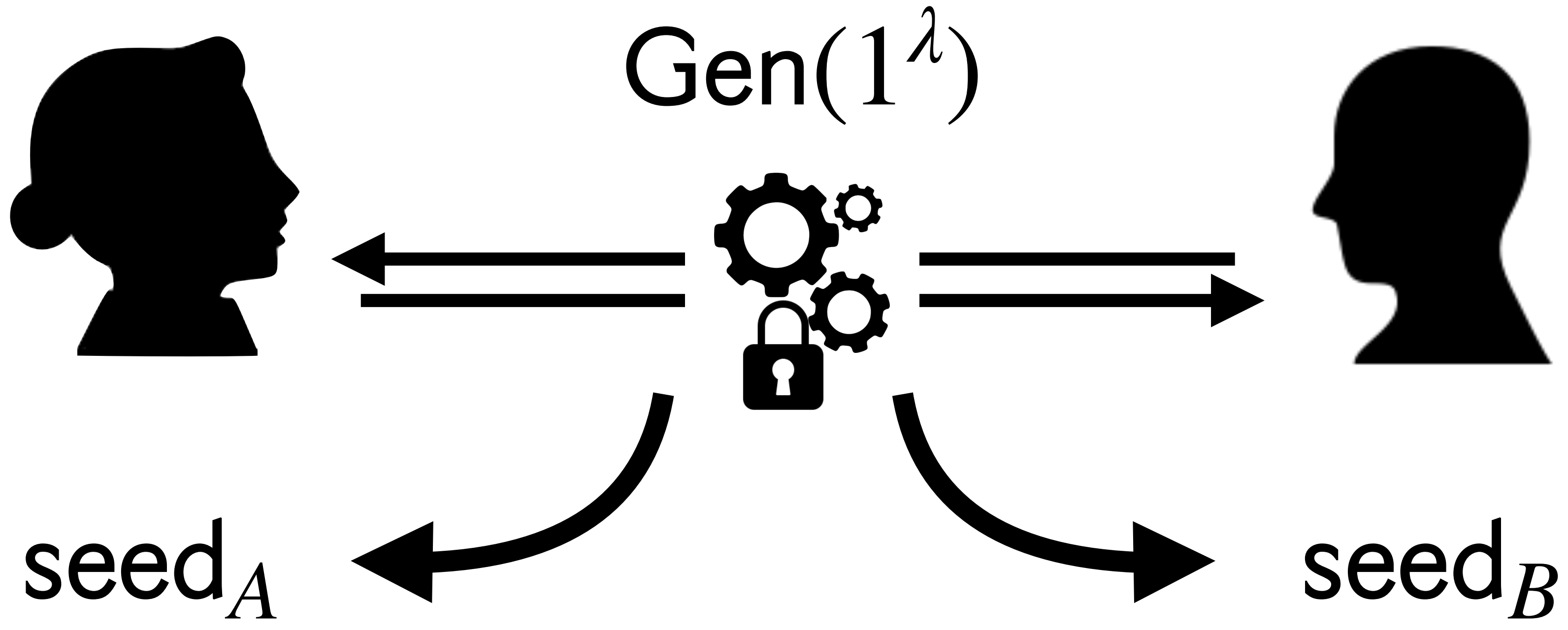
Preprocessing phase

Online phase

Secure Computation with Silent Preprocessing

Pseudorandom correlation generator: $\mathsf{Gen}(1^\lambda) \to (\mathsf{seed}_A, \mathsf{seed}_B)$ such that (1) $(\mathsf{Expand}(A, \mathsf{seed}_A), \mathsf{Expand}(B, \mathsf{seed}_B))$ looks like n samples from the target correlation, and (2) $\mathsf{Expand}(A, \mathsf{seed}_A)$ looks 'random conditioned on satisfying the correlation with $\mathsf{Expand}(B, \mathsf{seed}_B)$' to Bob (similar property w.r.t. Alice).

One-time short interaction



Interactive protocol with short communication and computation; Alice and Bob store a small seed afterwards.

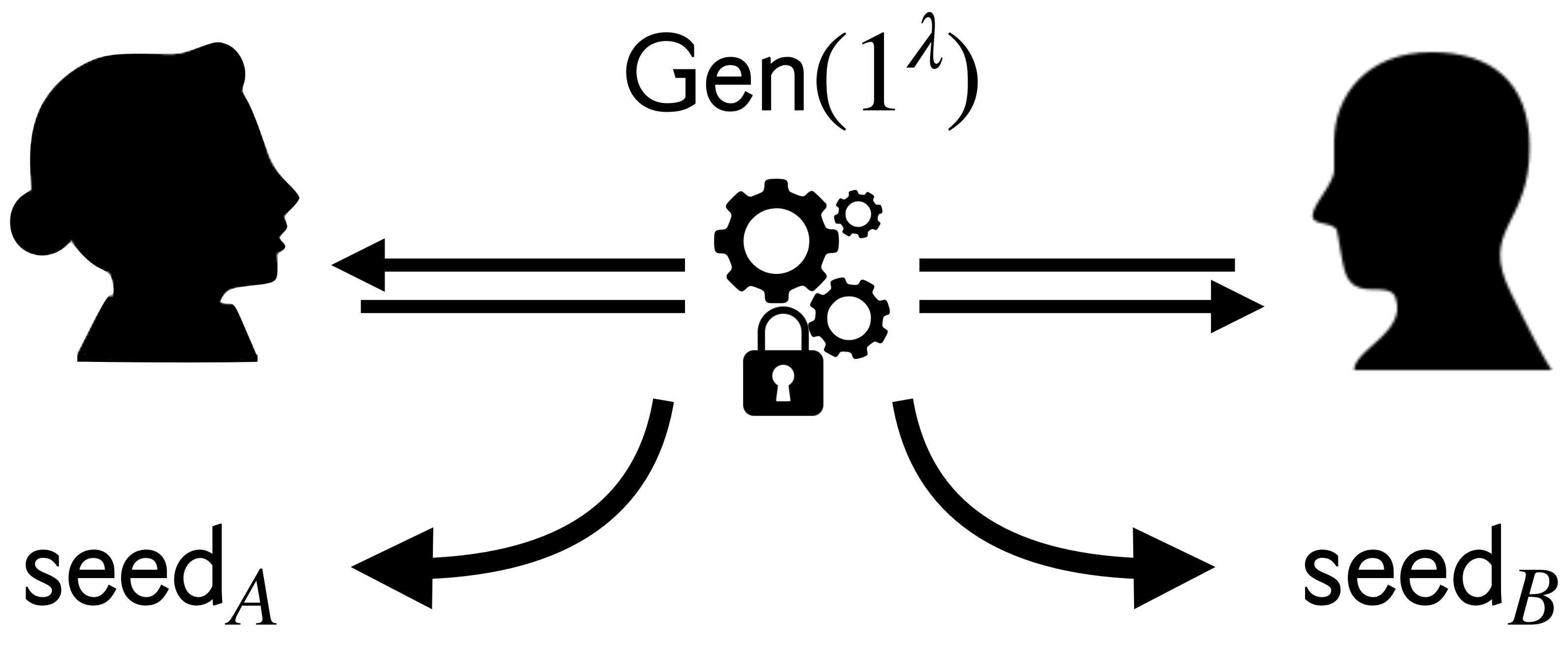
Preprocessing phase

Online phase

Secure Computation with Silent Preprocessing

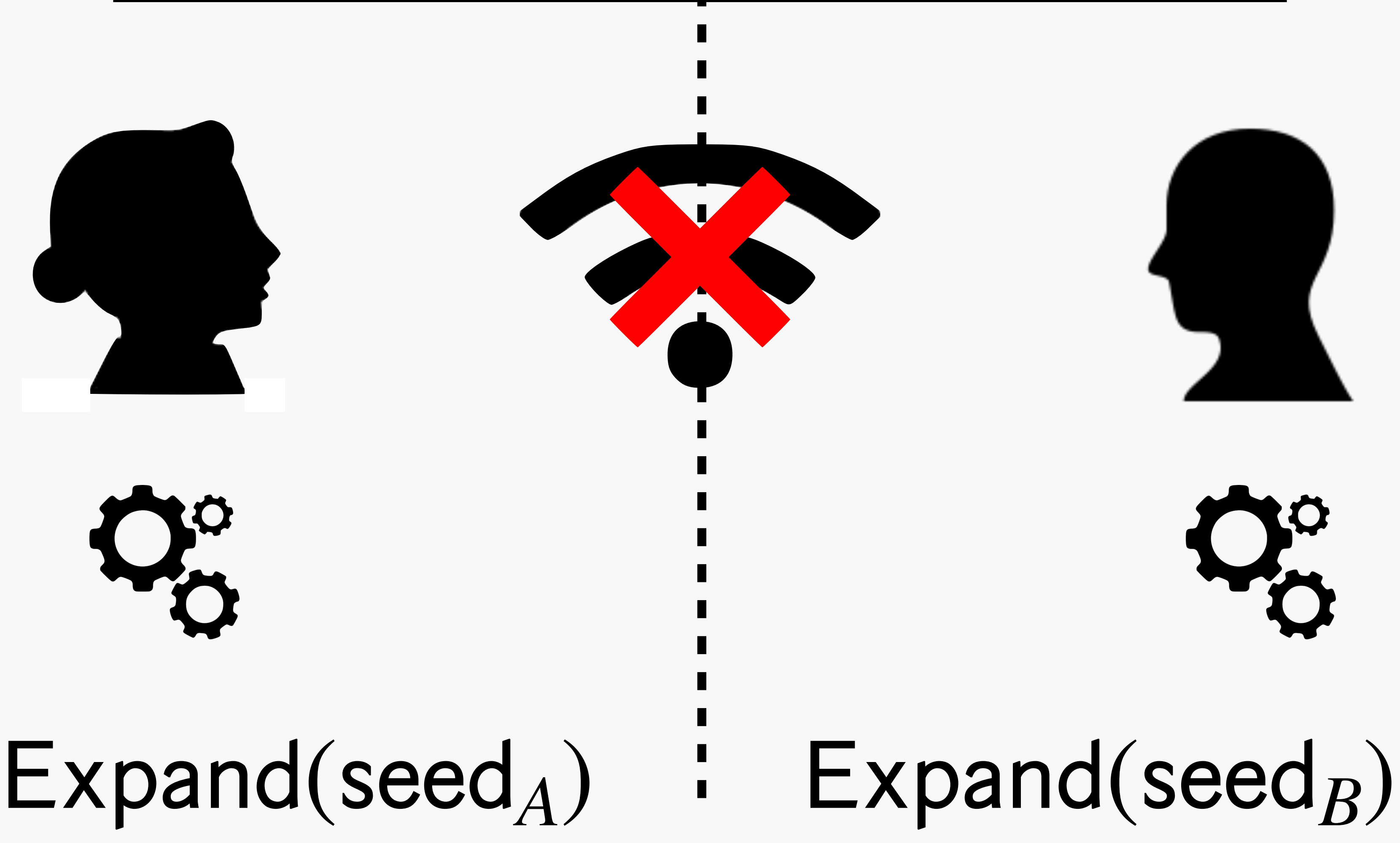
Pseudorandom correlation generator: $\mathsf{Gen}(1^\lambda) \rightarrow (\mathsf{seed}_A, \mathsf{seed}_B)$ **such that (1)** $(\mathsf{Expand}(A, \mathsf{seed}_A), \mathsf{Expand}(B, \mathsf{seed}_B))$ **looks like** n **samples from the target correlation, and (2)** $\mathsf{Expand}(A, \mathsf{seed}_A)$ **looks 'random conditioned on satisfying the correlation with** $\mathsf{Expand}(B, \mathsf{seed}_B)$**' to Bob (similar property w.r.t. Alice).**

One-time short interaction



Interactive protocol with short communication and computation; Alice and Bob store a small seed afterwards.

'Silent' computation



$\mathsf{Expand}(\mathsf{seed}_A)$
$\mathsf{Expand}(\mathsf{seed}_B)$

The bulk of the preprocessing phase is offline: Alice and Bob stretch their seeds into large pseudorandom correlated strings.

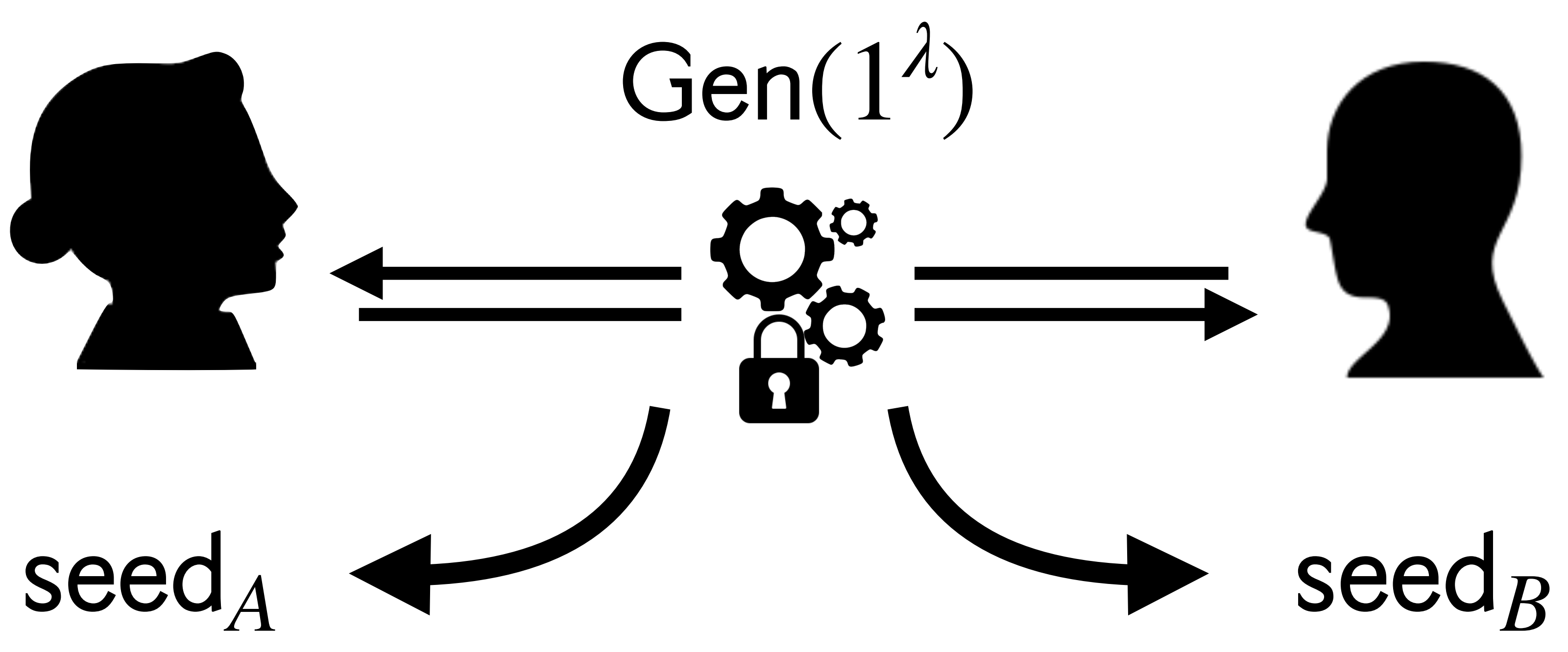
Preprocessing phase

Online phase

Secure Computation with Silent Preprocessing

Pseudorandom correlation generator: $\mathsf{Gen}(1^\lambda) \to (\mathsf{seed}_A, \mathsf{seed}_B)$ **such that (1)** $(\mathsf{Expand}(A, \mathsf{seed}_A), \mathsf{Expand}(B, \mathsf{seed}_B))$ **looks like** n **samples from the target correlation, and (2)** $\mathsf{Expand}(A, \mathsf{seed}_A)$ **looks 'random conditioned on satisfying the correlation with** $\mathsf{Expand}(B, \mathsf{seed}_B)$**' to Bob (similar property w.r.t. Alice).**

One-time short interaction



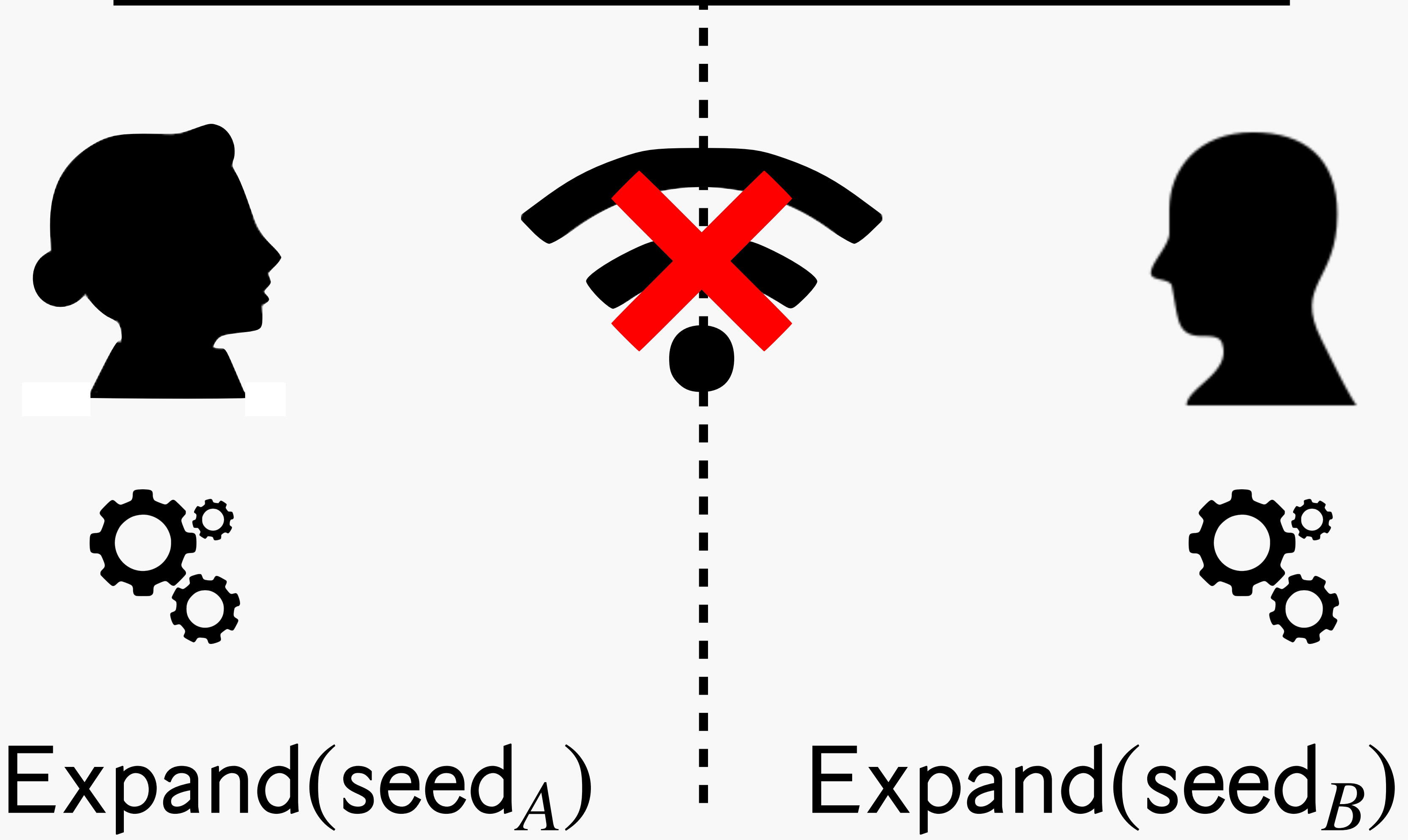
$\mathsf{Gen}(1^\lambda)$

seed_A

seed_B

Interactive protocol with short communication and computation; Alice and Bob store a small seed afterwards.

'Silent' computation

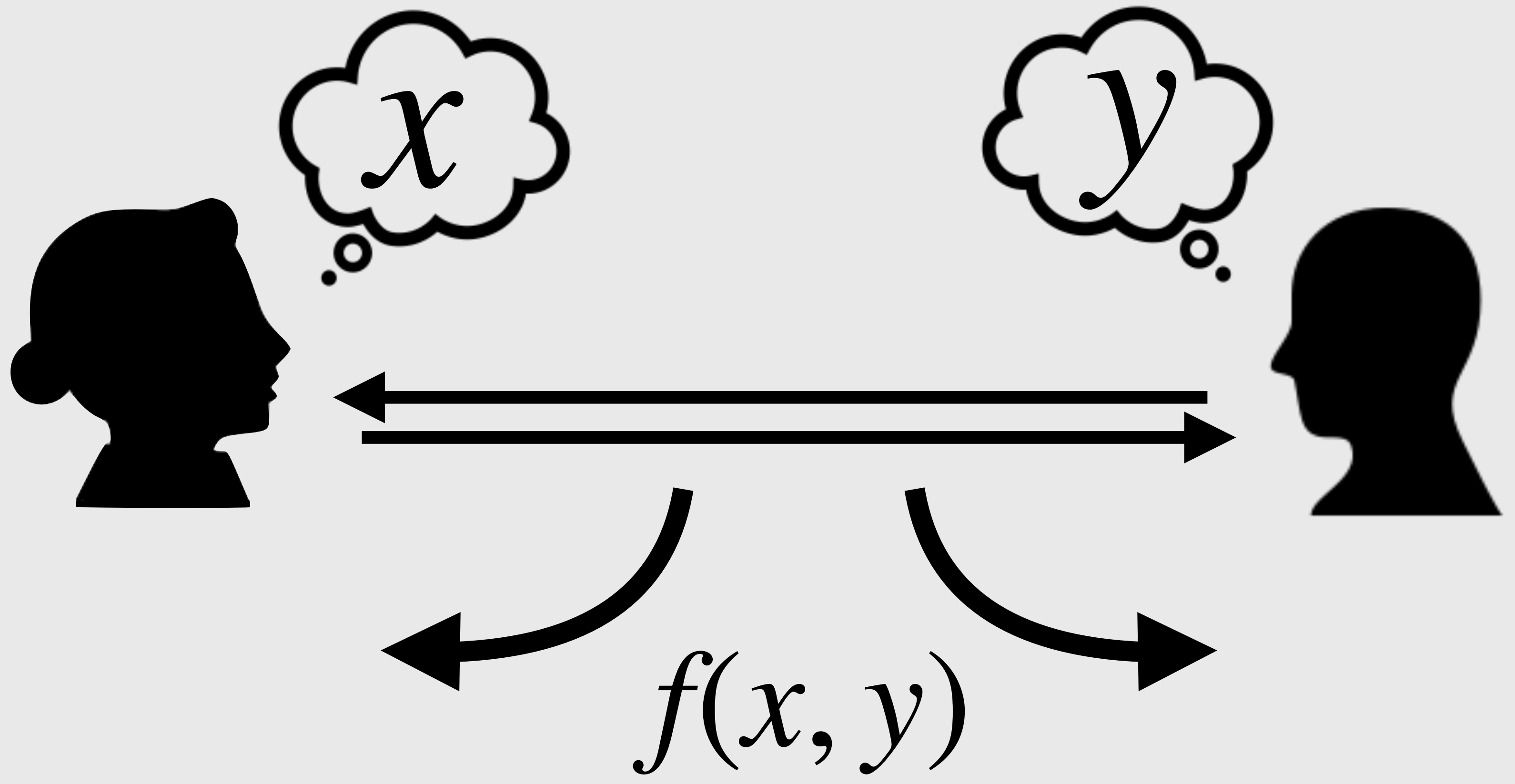


$\mathsf{Expand}(\mathsf{seed}_A)$

$\mathsf{Expand}(\mathsf{seed}_B)$

The bulk of the preprocessing phase is offline: Alice and Bob stretch their seeds into large pseudorandom correlated strings.

Non-cryptographic



x

y

$f(x, y)$

Alice and Bob consume the preprocessing material in a fast, non-cryptographic online phase.

Preprocessing phase

Online phase

Secure Computation with Silent Preprocessing

Pseudorandom correlation generator: $\mathsf{Gen}(1^\lambda) \rightarrow (\mathsf{seed}_A, \mathsf{seed}_B)$ **such that (1)** $(\mathsf{Expand}(A, \mathsf{seed}_A), \mathsf{Expand}(B, \mathsf{seed}_B))$ **looks like** n **samples from the target correlation, and (2)** $\mathsf{Expand}(A, \mathsf{seed}_A)$ **looks 'random conditioned on satisfying the correlation with** $\mathsf{Expand}(B, \mathsf{seed}_B)$**' to Bob (similar property w.r.t. Alice).**

History

This work

🟩 : efficient
🟨 : doable
🟥 : purely theoretical

🟦 : linear correlation
🟪 : non-linear correlation

Secure Computation with Silent Preprocessing

Pseudorandom correlation generator: $\mathsf{Gen}(1^\lambda) \to (\mathsf{seed}_A, \mathsf{seed}_B)$ **such that (1)** $(\mathsf{Expand}(A, \mathsf{seed}_A), \mathsf{Expand}(B, \mathsf{seed}_B))$ **looks like** n **samples from the target correlation, and (2)** $\mathsf{Expand}(A, \mathsf{seed}_A)$ **looks 'random conditioned on satisfying the correlation with** $\mathsf{Expand}(B, \mathsf{seed}_B)$**' to Bob (similar property w.r.t. Alice).**

History

- 2-party, linear correlation, from OWF (use a PRG)
- Multi-party linear correlations, from OWF [GI99, CDI05]

- All correlations, from iO [BCGIO17]
- All additive correlations, from LWE [BGI15, DHRW16]

🟩 : efficient
🟨 : doable
🟥 : purely theoretical

🟦 : linear correlation
🟪 : non-linear correlation

This work

Secure Computation with Silent Preprocessing

Pseudorandom correlation generator: $\mathsf{Gen}(1^\lambda) \to (\mathsf{seed}_A, \mathsf{seed}_B)$ **such that (1)** $(\mathsf{Expand}(A, \mathsf{seed}_A), \mathsf{Expand}(B, \mathsf{seed}_B))$ **looks like n samples from the target correlation, and (2)** $\mathsf{Expand}(A, \mathsf{seed}_A)$ **looks 'random conditioned on satisfying the correlation with** $\mathsf{Expand}(B, \mathsf{seed}_B)$**' to Bob (similar property w.r.t. Alice).**

History

- 2-party, linear correlation, from OWF (use a PRG)
- Multi-party linear correlations, from OWF [GI99, CDI05]

- All correlations, from iO [BCGIO17]
- All additive correlations, from LWE [BGI15, DHRW16]

- OT correlation, from DDH [BGI16, B**C**GIO17]

- Vector oblivious linear evaluation, from LPN [B**C**GI18]
- OT correlation, from LPN [B**C**GIKS19]

- Bilinear correlations, from LPN [B**C**GIKS19]

- OLE correlations, from ring-LPN [B**C**GIKS20]

🟩 : efficient
🟨 : doable
🟥 : purely theoretical

▭ : linear correlation
▭ : non-linear correlation

This work

Secure Computation with Silent Preprocessing

Pseudorandom correlation generator: $\mathsf{Gen}(1^\lambda) \to (\mathsf{seed}_A, \mathsf{seed}_B)$ **such that (1)** $(\mathsf{Expand}(A, \mathsf{seed}_A), \mathsf{Expand}(B, \mathsf{seed}_B))$ **looks like** n **samples from the target correlation, and (2)** $\mathsf{Expand}(A, \mathsf{seed}_A)$ **looks 'random conditioned on satisfying the correlation with** $\mathsf{Expand}(B, \mathsf{seed}_B)$**' to Bob (similar property w.r.t. Alice).**

History

- 2-party, linear correlation, from OWF (use a PRG)
- Multi-party linear correlations, from OWF [GI99, CDI05]

- All correlations, from iO [BCGIO17]
- All additive correlations, from LWE [BGI15, DHRW16]

- OT correlation, from DDH [BGI16, B**C**GIO17]

- Vector oblivious linear evaluation, from LPN [B**C**GI18]
- OT correlation, from LPN [B**C**GIKS19]

- Bilinear correlations, from LPN [B**C**GIKS19]

- OLE correlations, from ring-LPN [B**C**GIKS20]

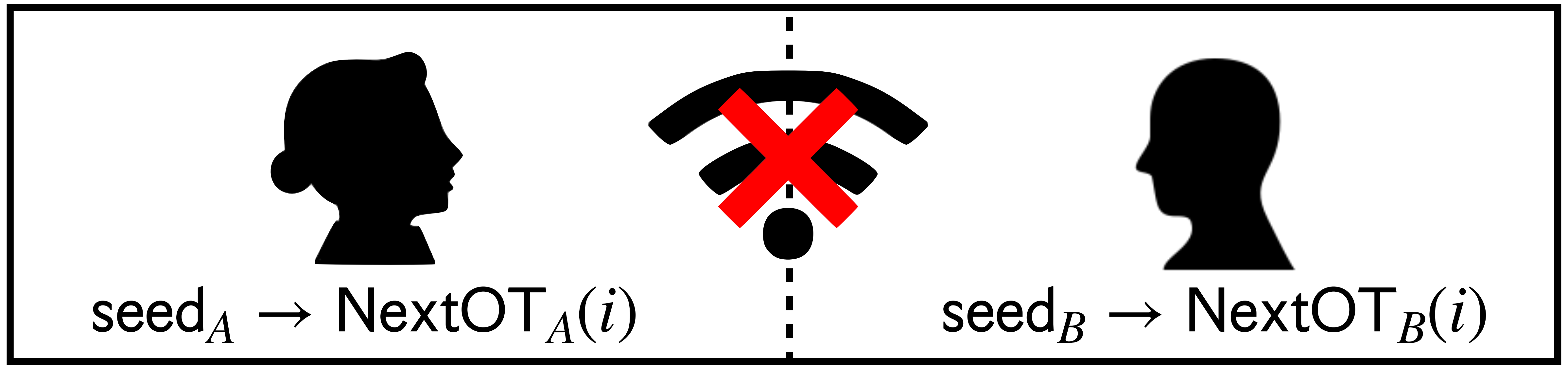
🟩 : efficient
🟧 : doable
🟥 : purely theoretical

🟦 : linear correlation
🟪 : non-linear correlation

This work

PCGs are limited to a *one-time* stretch of the seeds into a bounded polynomially-long pseudorandom correlation.

? Can we achieve the dream result of an *indefinitely reusable* source of correlated pseudorandomness? ?


$\mathsf{seed}_A \to \mathsf{NextOT}_A(i)$
$\mathsf{seed}_B \to \mathsf{NextOT}_B(i)$

Secure Computation with Silent Preprocessing

Pseudorandom correlation generator: $\mathsf{Gen}(1^\lambda) \rightarrow (\mathsf{seed}_A, \mathsf{seed}_B)$ **such that (1)** $(\mathsf{Expand}(A, \mathsf{seed}_A), \mathsf{Expand}(B, \mathsf{seed}_B))$ **looks like** n **samples from the target correlation, and (2)** $\mathsf{Expand}(A, \mathsf{seed}_A)$ **looks 'random conditioned on satisfying the correlation with** $\mathsf{Expand}(B, \mathsf{seed}_B)$**' to Bob (similar property w.r.t. Alice).**

History

- 2-party, linear correlation, from OWF (use a PRG)
- Multi-party linear correlations, from OWF [GI99, CDI05]

- All correlations, from iO [BCGIO17]
- All additive correlations, from LWE [BGI15, DHRW16]

- OT correlation, from DDH [BGI16, B**C**GIO17]

- Vector oblivious linear evaluation, from LPN [B**C**GI18]
- OT correlation, from LPN [B**C**GIKS19]

- Bilinear correlations, from LPN [B**C**GIKS19]

- OLE correlations, from ring-LPN [B**C**GIKS20]

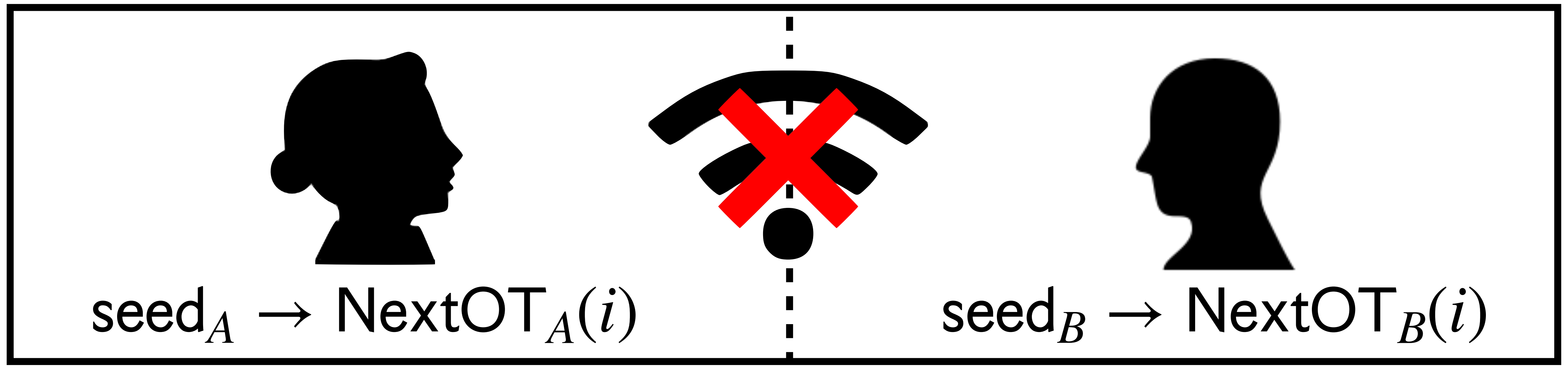
🟩 : efficient
🟨 : doable
🟥 : purely theoretical

🟦 : linear correlation
🟪 : non-linear correlation

This work

PCGs are limited to a *one-time* stretch of the seeds into a bounded polynomially-long pseudorandom correlation.

Can we achieve the dream result of an *indefinitely reusable* source of correlated pseudorandomness?


$\mathsf{seed}_A \rightarrow \mathsf{NextOT}_A(i)$
$\mathsf{seed}_B \rightarrow \mathsf{NextOT}_B(i)$

Idea: we could use the doubling trick… but it prevents accessing the correlations *incrementally*

Secure Computation with Silent Preprocessing

Pseudorandom correlation generator: $\mathsf{Gen}(1^\lambda) \to (\mathsf{seed}_A, \mathsf{seed}_B)$ **such that (1)** $(\mathsf{Expand}(A, \mathsf{seed}_A), \mathsf{Expand}(B, \mathsf{seed}_B))$ **looks like** n **samples from the target correlation, and (2)** $\mathsf{Expand}(A, \mathsf{seed}_A)$ **looks 'random conditioned on satisfying the correlation with** $\mathsf{Expand}(B, \mathsf{seed}_B)$**' to Bob (similar property w.r.t. Alice).**

History

- 2-party, linear correlation, from OWF (use a PRG)
- Multi-party linear correlations, from OWF [GI99, CDI05]

- All correlations, from iO [BCGIO17]
- All additive correlations, from LWE [BGI15, DHRW16]

- OT correlation, from DDH [BGI16, B**C**GIO17]

- Vector oblivious linear evaluation, from LPN [B**C**GI18]
- OT correlation, from LPN [B**C**GIKS19]

- Bilinear correlations, from LPN [B**C**GIKS19]

- OLE correlations, from ring-LPN [B**C**GIKS20]

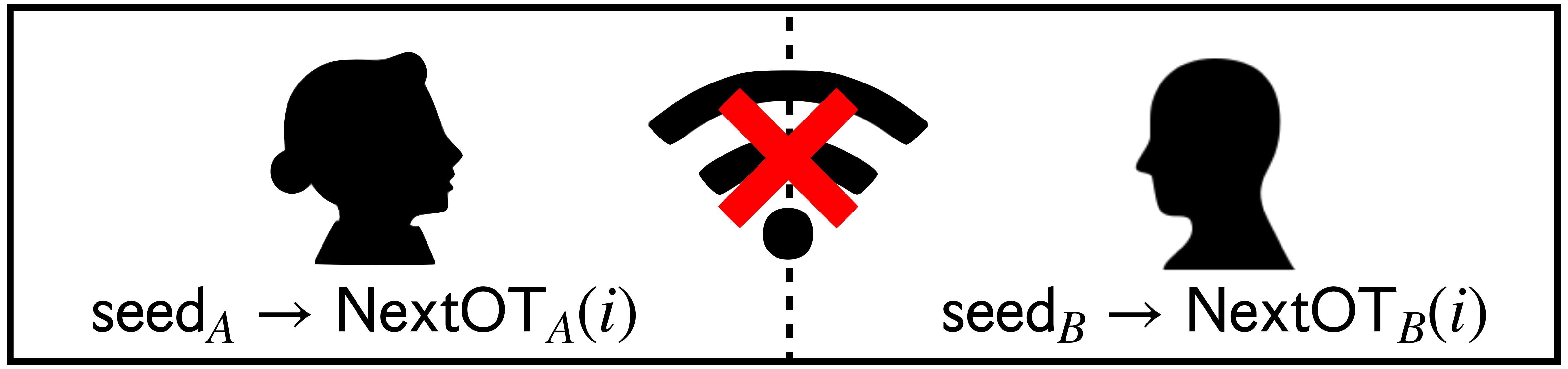
🟩 : efficient
🟨 : doable
🟥 : purely theoretical

□ : linear correlation
□ : non-linear correlation

This work

PCGs are limited to a *one-time* stretch of the seeds into a bounded polynomially-long pseudorandom correlation.

Can we achieve the dream result of an *indefinitely reusable* source of correlated pseudorandomness?



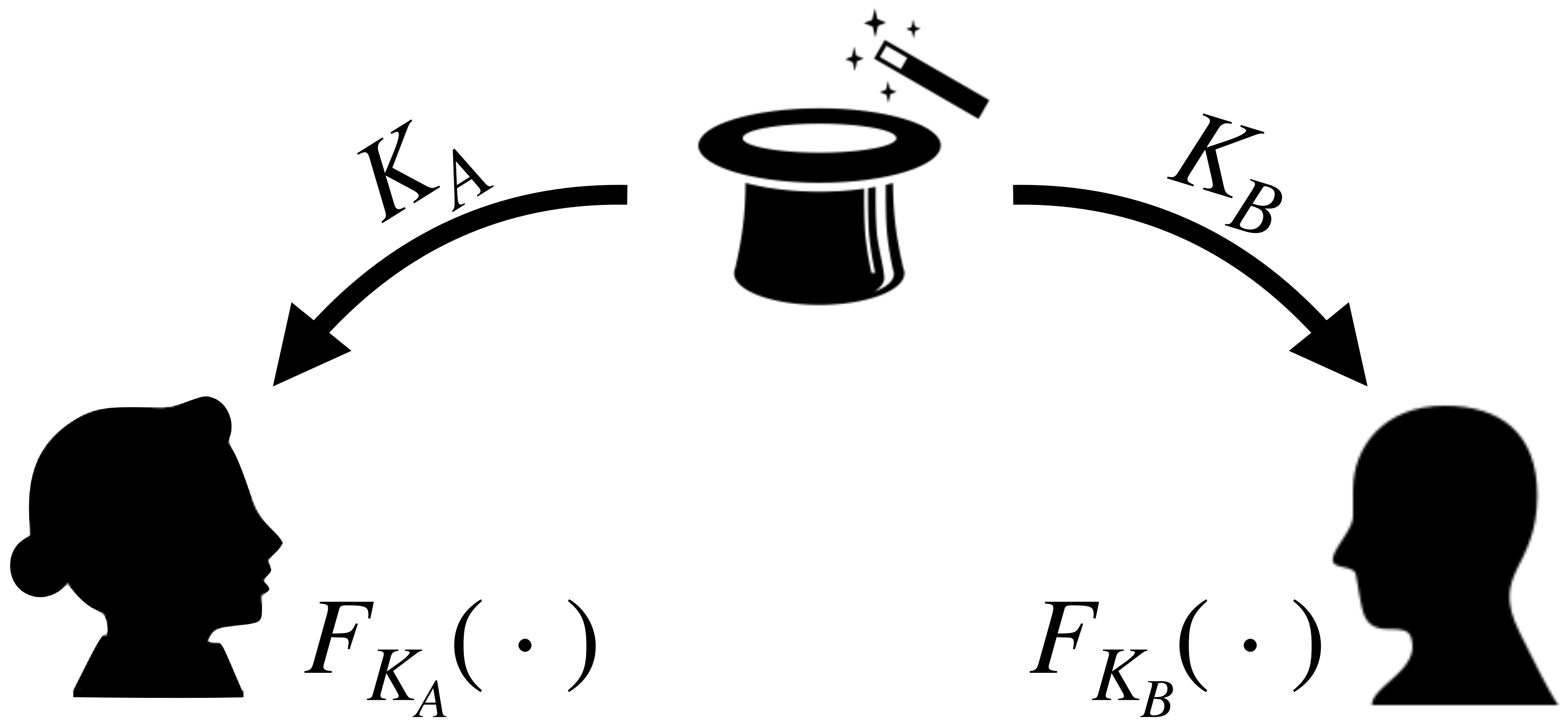
$\mathsf{seed}_A \to \mathsf{NextOT}_A(i)$

$\mathsf{seed}_B \to \mathsf{NextOT}_B(i)$

We want a pseudorandom correlation **function.**

Pseudorandom Correlation Functions and Low-Complexity WPRFs

Correlated pseudorandom functions

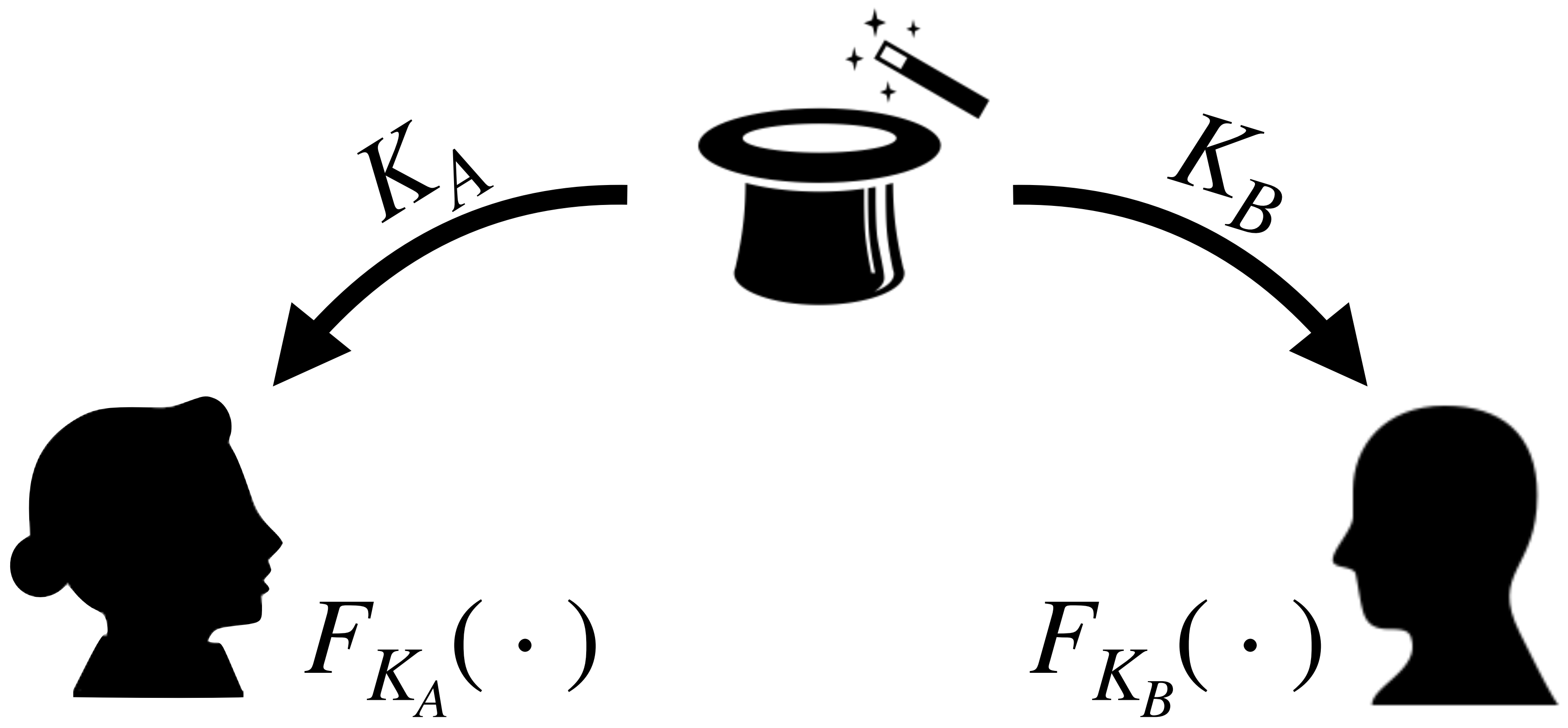


$\approx$

Low-Complexity Weak PRFs

Pseudorandom Correlation Functions and Low-Complexity WPRFs

Correlated pseudorandom functions



Correctness & security:

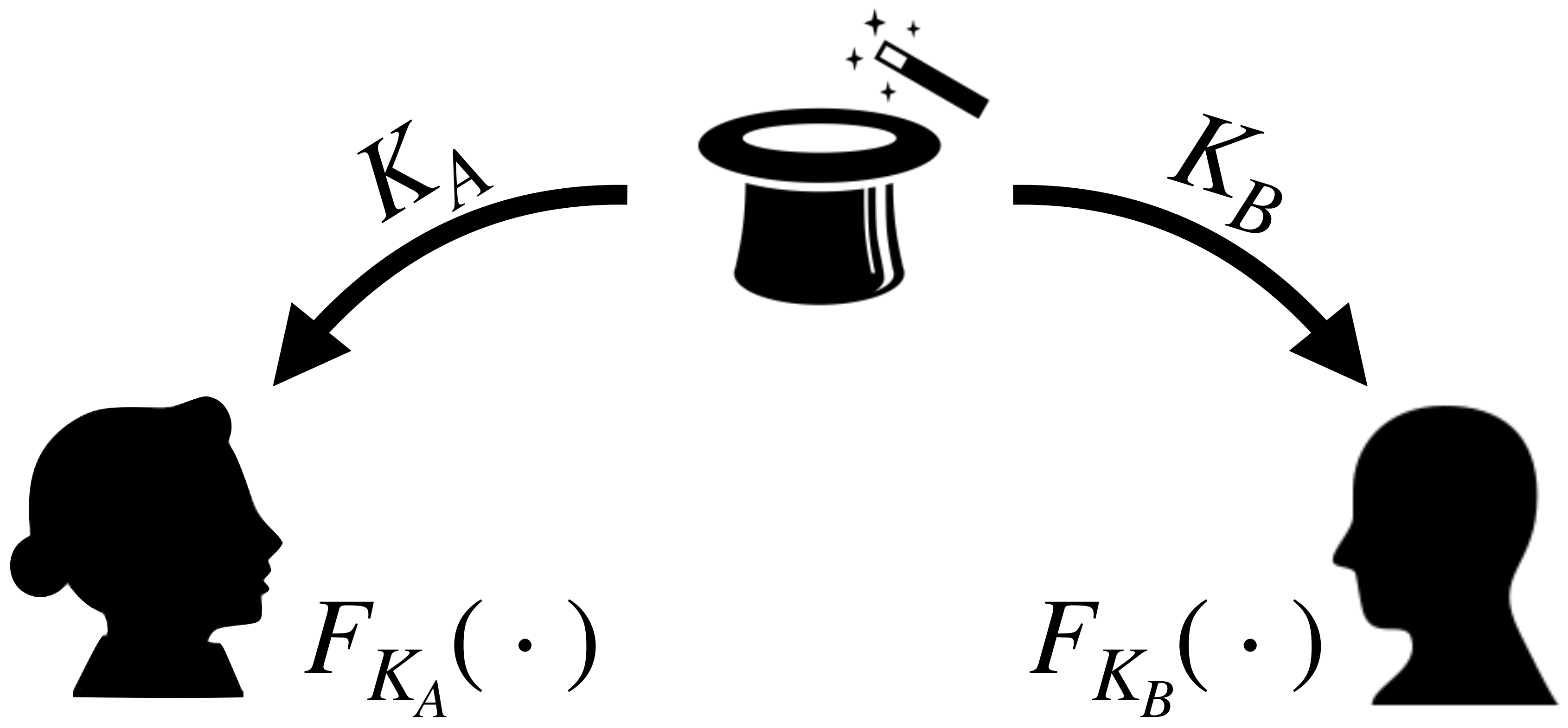
- Black-box access to samples of the form $(F_{K_A}(x), F_{K_B}(x))$ are indistinguishable from black-box access to random samples from a target correlation.
- From the viewpoint of Alice, each $F_{K_B}(x)$ is indistinguishable from a random value sampled *conditioned on satisfying the correlation with* $F_{K_A}(x)$*.*
- Same condition in the other direction.

$\approx$

Low-Complexity Weak PRFs

Pseudorandom Correlation Functions and Low-Complexity WPRFs

Correlated pseudorandom functions



Correctness & security:

- Black-box access to samples of the form $(F_{K_A}(x), F_{K_B}(x))$ are indistinguishable from black-box access to random samples from a target correlation.
- From the viewpoint of Alice, each $F_{K_B}(x)$ is indistinguishable from a random value sampled *conditioned on satisfying the correlation with $F_{K_A}(x)$.*
- Same condition in the other direction.

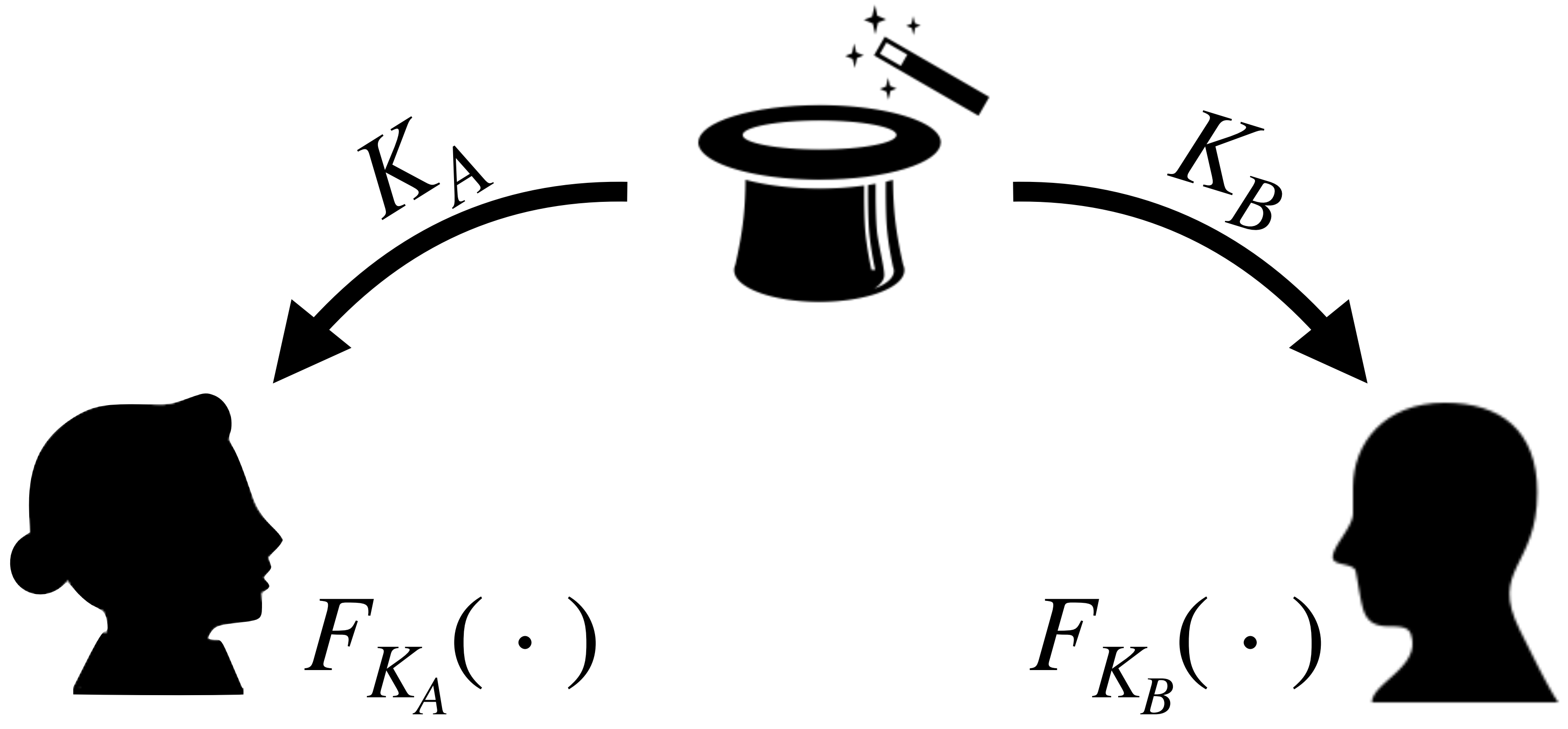
$\approx$

Low-Complexity Weak PRFs

WPRF: F_K is indistinguishable from a random function when evaluated on random inputs.

Pseudorandom Correlation Functions and Low-Complexity WPRFs

Correlated pseudorandom functions



Correctness & security:

- Black-box access to samples of the form $(F_{K_A}(x), F_{K_B}(x))$ are indistinguishable from black-box access to random samples from a target correlation.
- From the viewpoint of Alice, each $F_{K_B}(x)$ is indistinguishable from a random value sampled *conditioned on satisfying the correlation with* $F_{K_A}(x)$*.*
- Same condition in the other direction.

$\boxed{\approx}$

Low-Complexity Weak PRFs

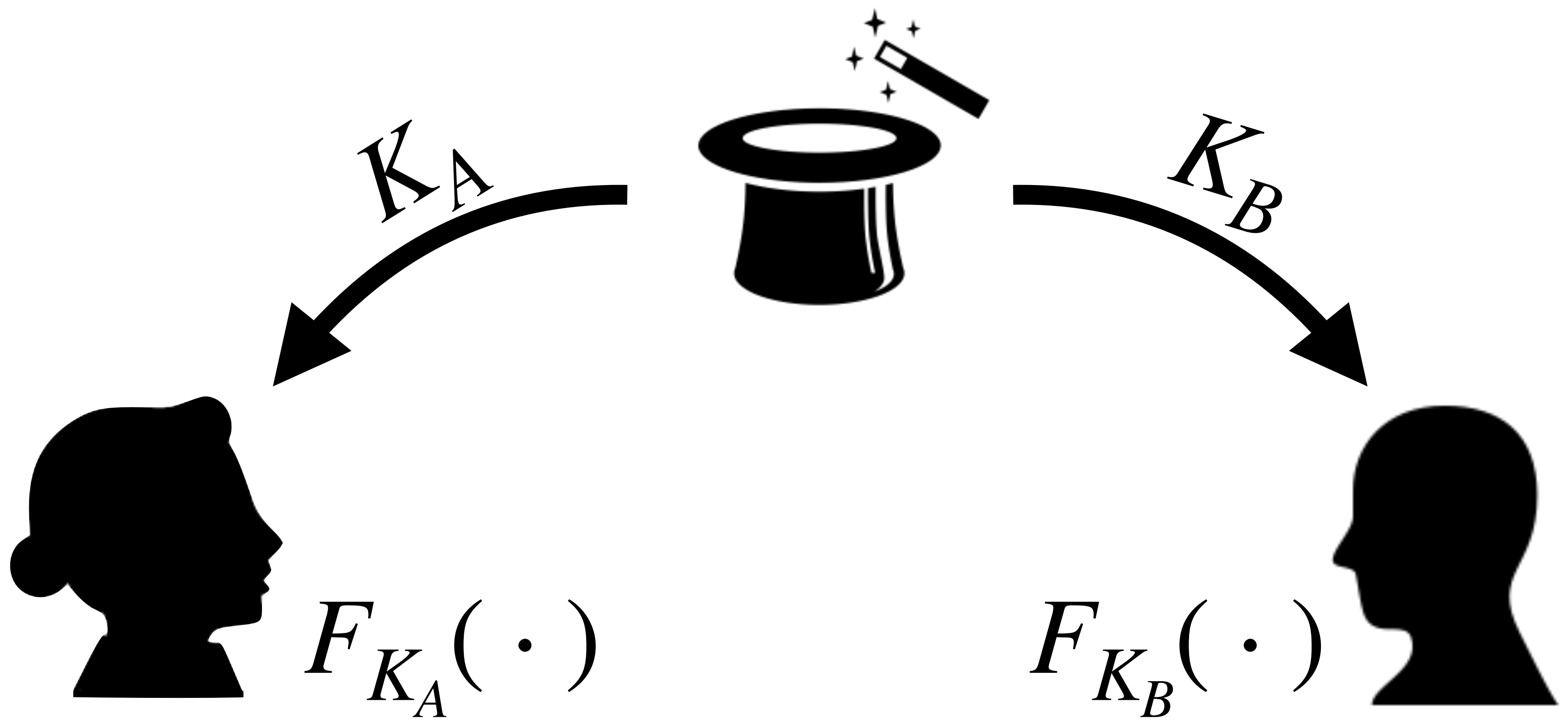
WPRF: F_K is indistinguishable from a random function when evaluated on random inputs.

How low can we go?

Copyright Yuval Ishai - 2020

Pseudorandom Correlation Functions and Low-Complexity WPRFs

Correlated pseudorandom functions



Correctness & security:

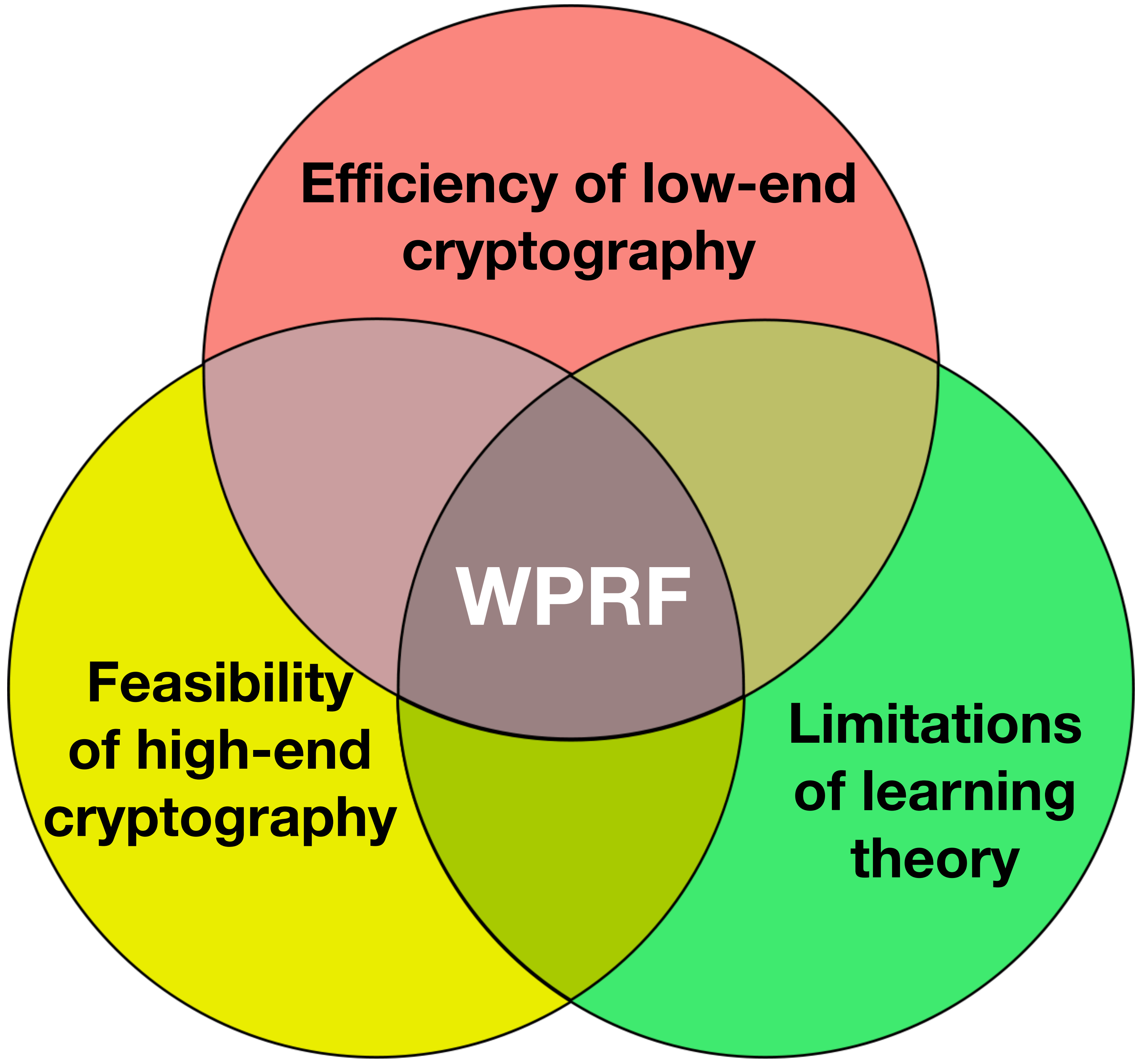
- Black-box access to samples of the form $(F_{K_A}(x), F_{K_B}(x))$ are indistinguishable from black-box access to random samples from a target correlation.
- From the viewpoint of Alice, each $F_{K_B}(x)$ is indistinguishable from a random value sampled *conditioned on satisfying the correlation with* $F_{K_A}(x)$.
- Same condition in the other direction.

$\approx$

Low-Complexity Weak PRFs

WPRF: F_K is indistinguishable from a random function when evaluated on random inputs.

How low can we go?



Efficiency of low-end cryptography

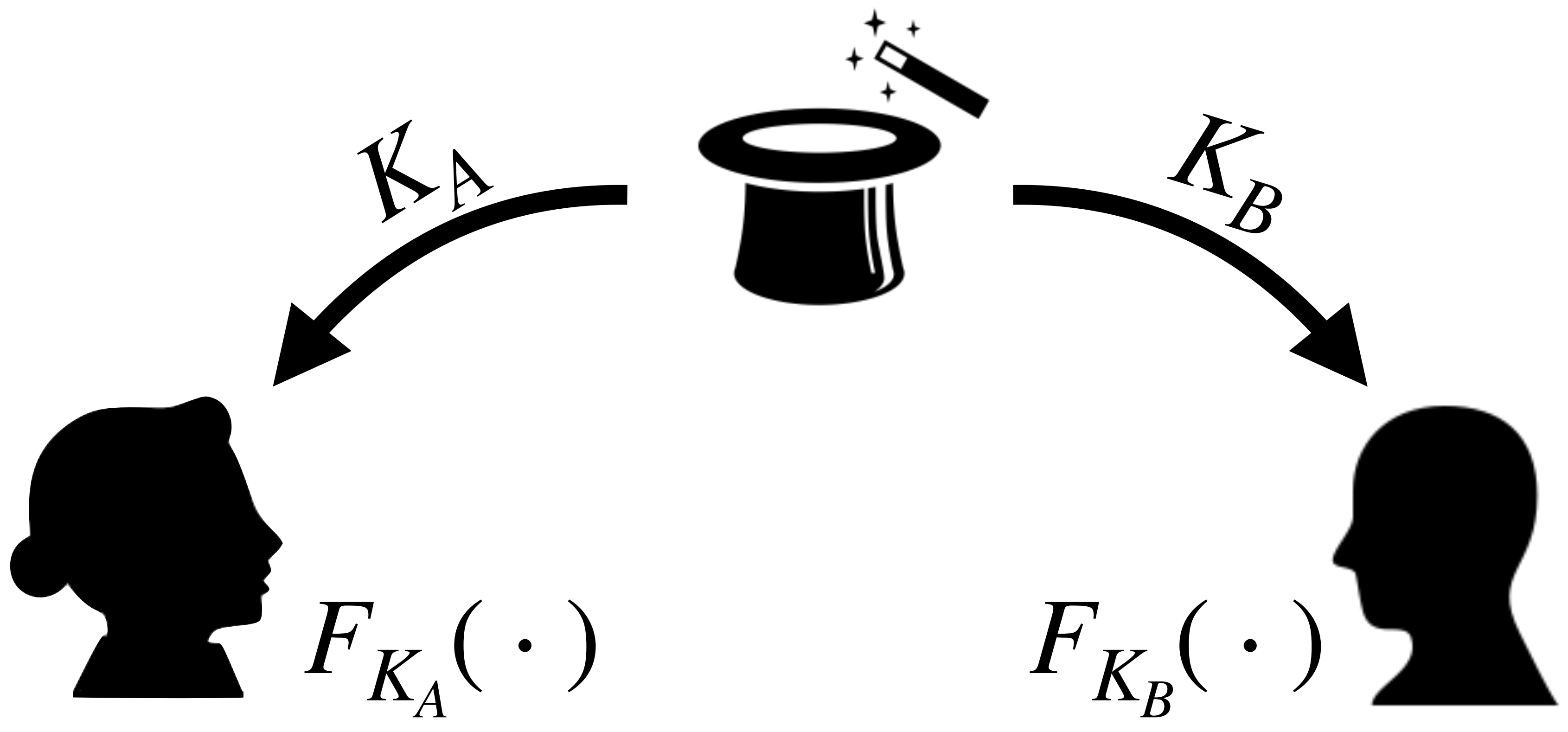
Feasibility of high-end cryptography

Limitations of learning theory

WPRF

Pseudorandom Correlation Functions and Low-Complexity WPRFs

Correlated pseudorandom functions



Correctness & security:

- Black-box access to samples of the form $(F_{K_A}(x), F_{K_B}(x))$ are indistinguishable from black-box access to random samples from a target correlation.
- From the viewpoint of Alice, each $F_{K_B}(x)$ is indistinguishable from a random value sampled *conditioned on satisfying the correlation with $F_{K_A}(x)$.*
- Same condition in the other direction.

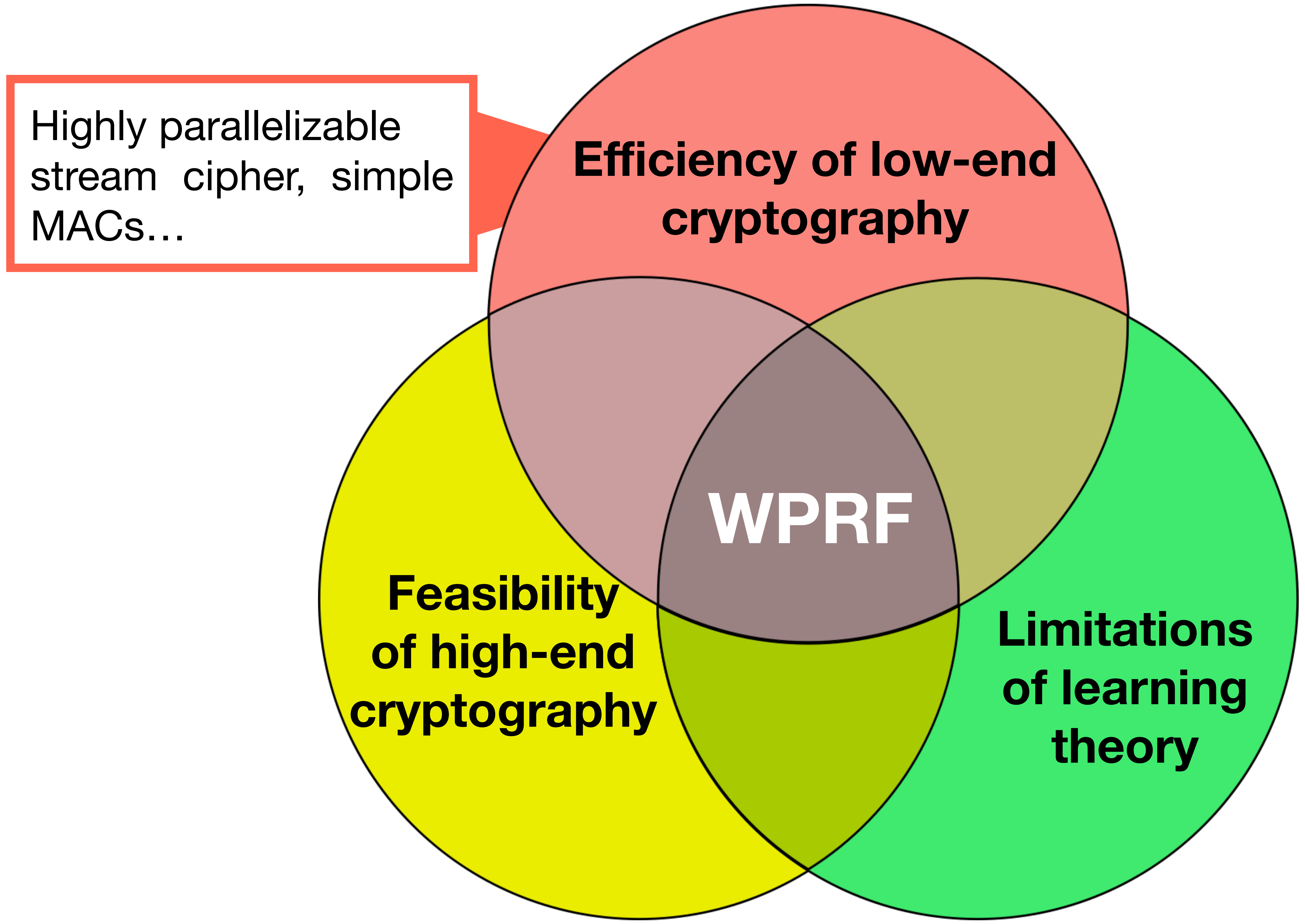
$\approx$

Low-Complexity Weak PRFs

WPRF: F_K is indistinguishable from a random function when evaluated on random inputs.

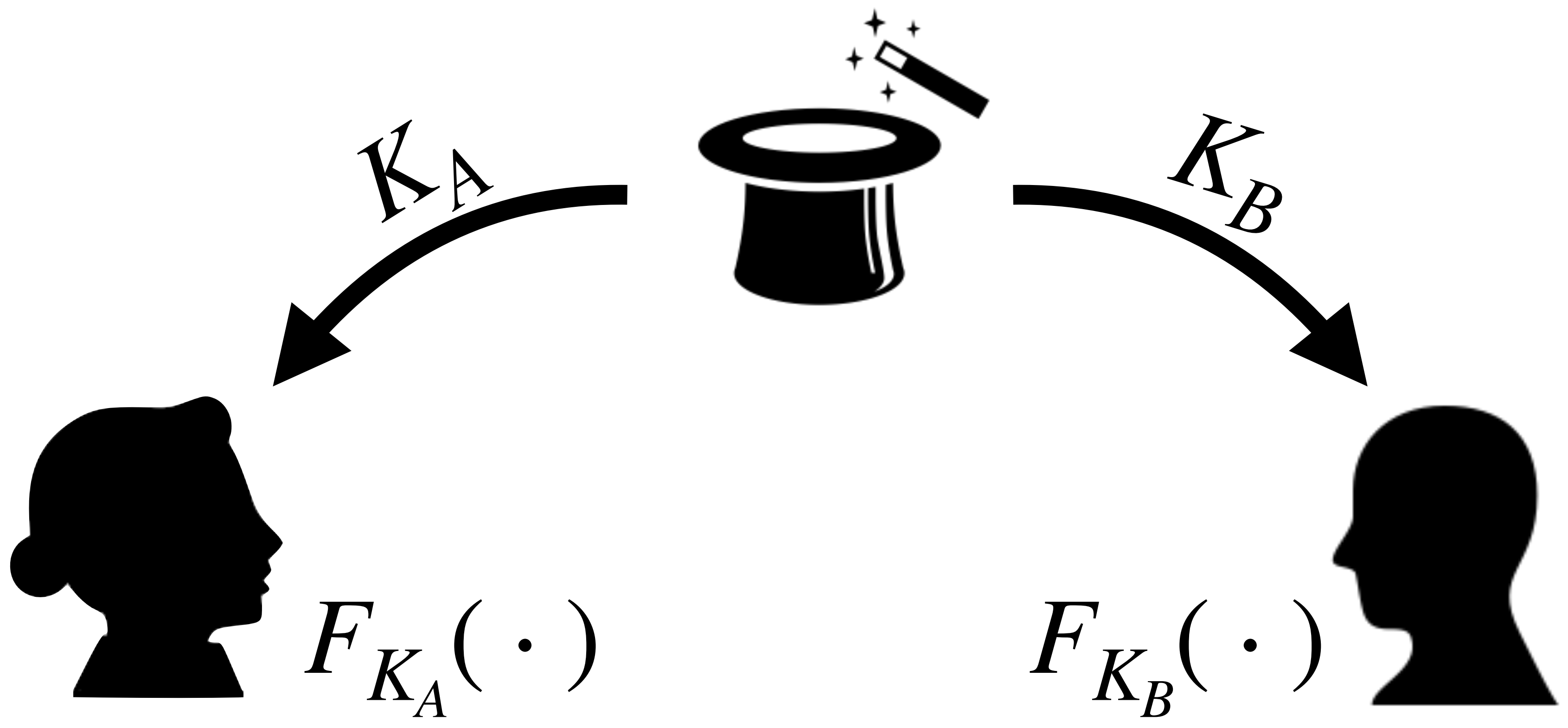
How low can we go?

Copyright Yuval Ishai - 2020


Highly parallelizable stream cipher, simple MACs…
Efficiency of low-end cryptography
WPRF
Feasibility of high-end cryptography
Limitations of learning theory

Pseudorandom Correlation Functions and Low-Complexity WPRFs

Correlated pseudorandom functions



Correctness & security:

- Black-box access to samples of the form $(F_{K_A}(x), F_{K_B}(x))$ are indistinguishable from black-box access to random samples from a target correlation.
- From the viewpoint of Alice, each $F_{K_B}(x)$ is indistinguishable from a random value sampled *conditioned on satisfying the correlation with* $F_{K_A}(x)$.
- Same condition in the other direction.

$\approx$

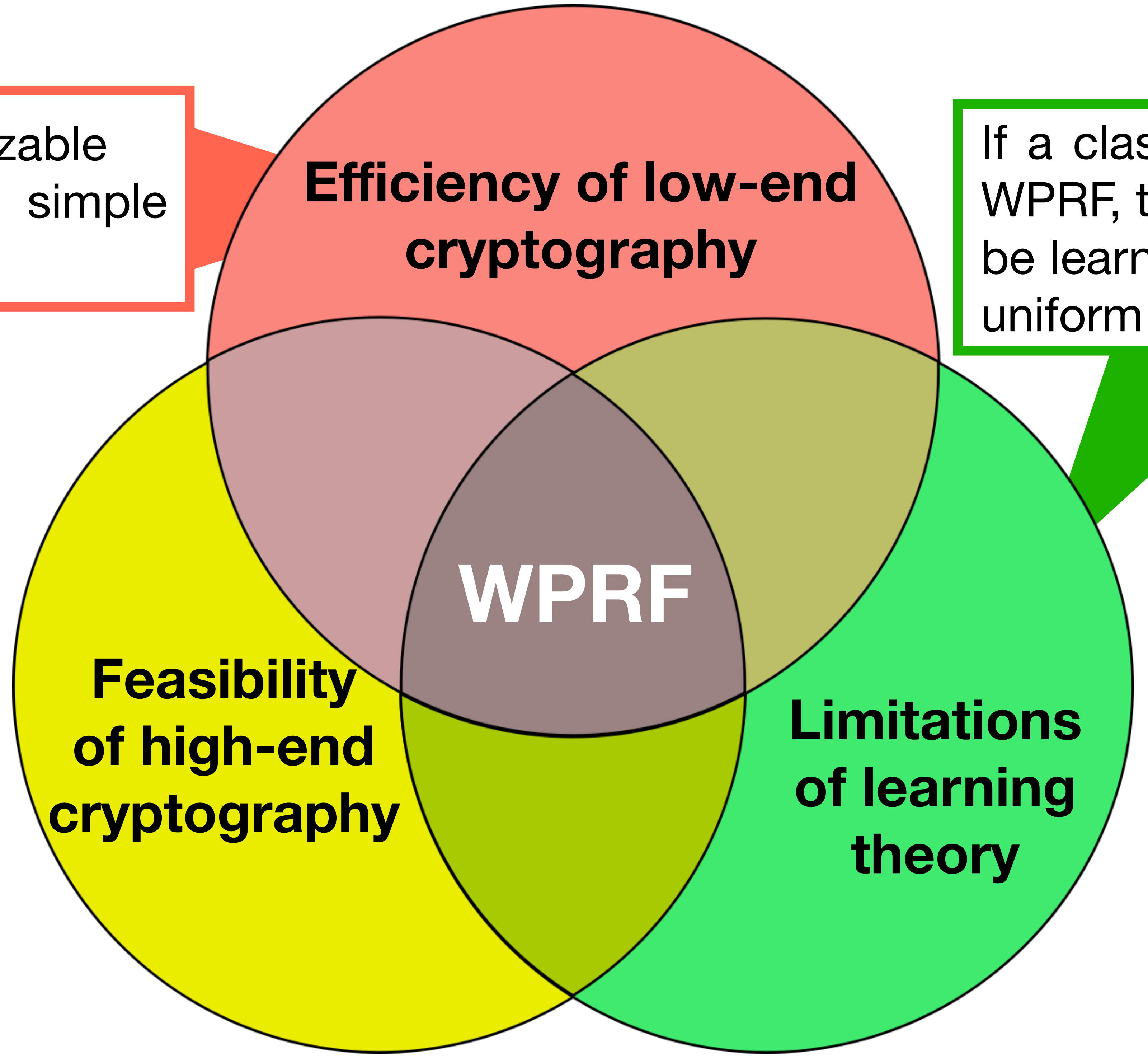
Low-Complexity Weak PRFs

WPRF: F_K is indistinguishable from a random function when evaluated on random inputs.

How low can we go?

Copyright Yuval Ishai - 2020

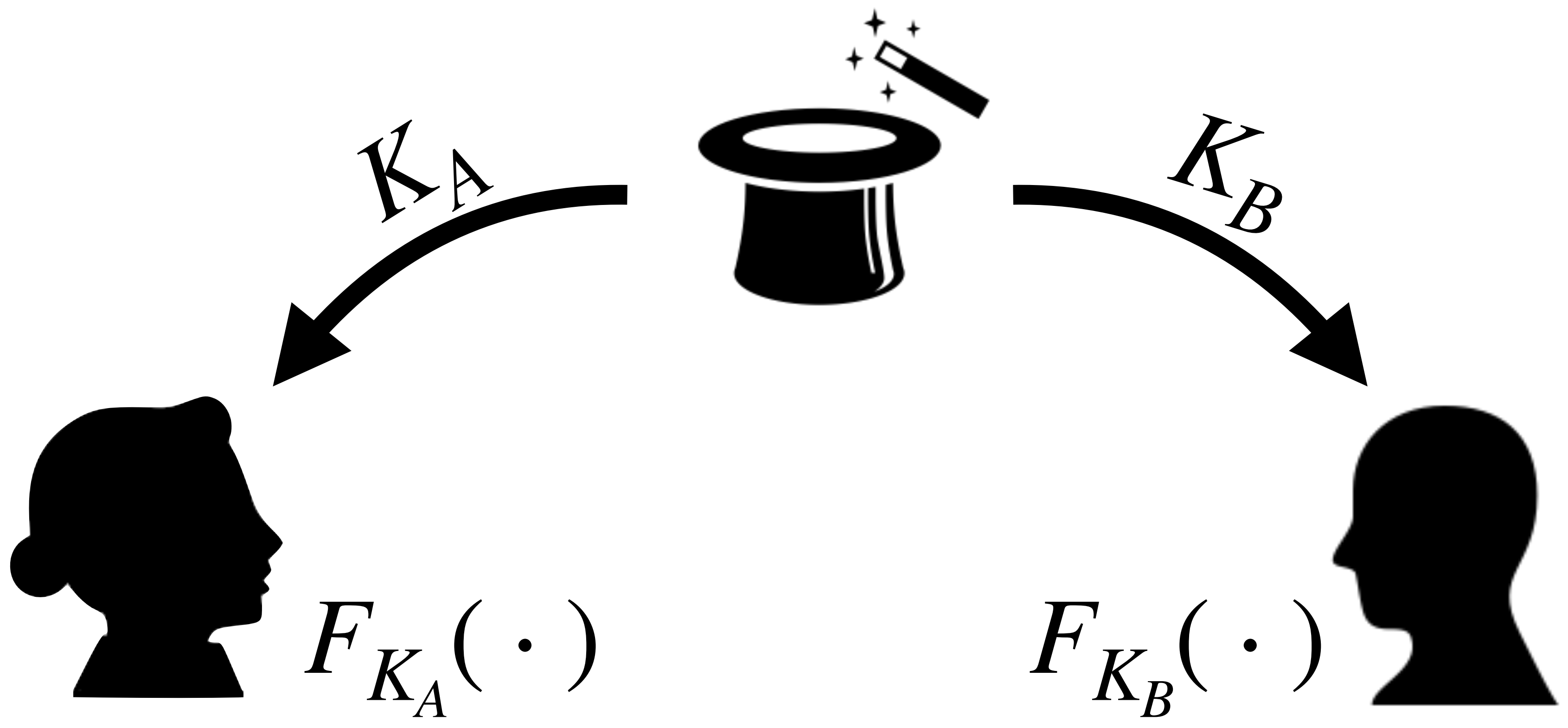
Highly parallelizable stream cipher, simple MACs…


Efficiency of low-end cryptography
WPRF
Feasibility of high-end cryptography
Limitations of learning theory

If a class contains a WPRF, then it cannot be learned under the uniform distribution

Pseudorandom Correlation Functions and Low-Complexity WPRFs

Correlated pseudorandom functions



Correctness & security:

- Black-box access to samples of the form $(F_{K_A}(x), F_{K_B}(x))$ are indistinguishable from black-box access to random samples from a target correlation.
- From the viewpoint of Alice, each $F_{K_B}(x)$ is indistinguishable from a random value sampled *conditioned on satisfying the correlation with* $F_{K_A}(x)$.
- Same condition in the other direction.

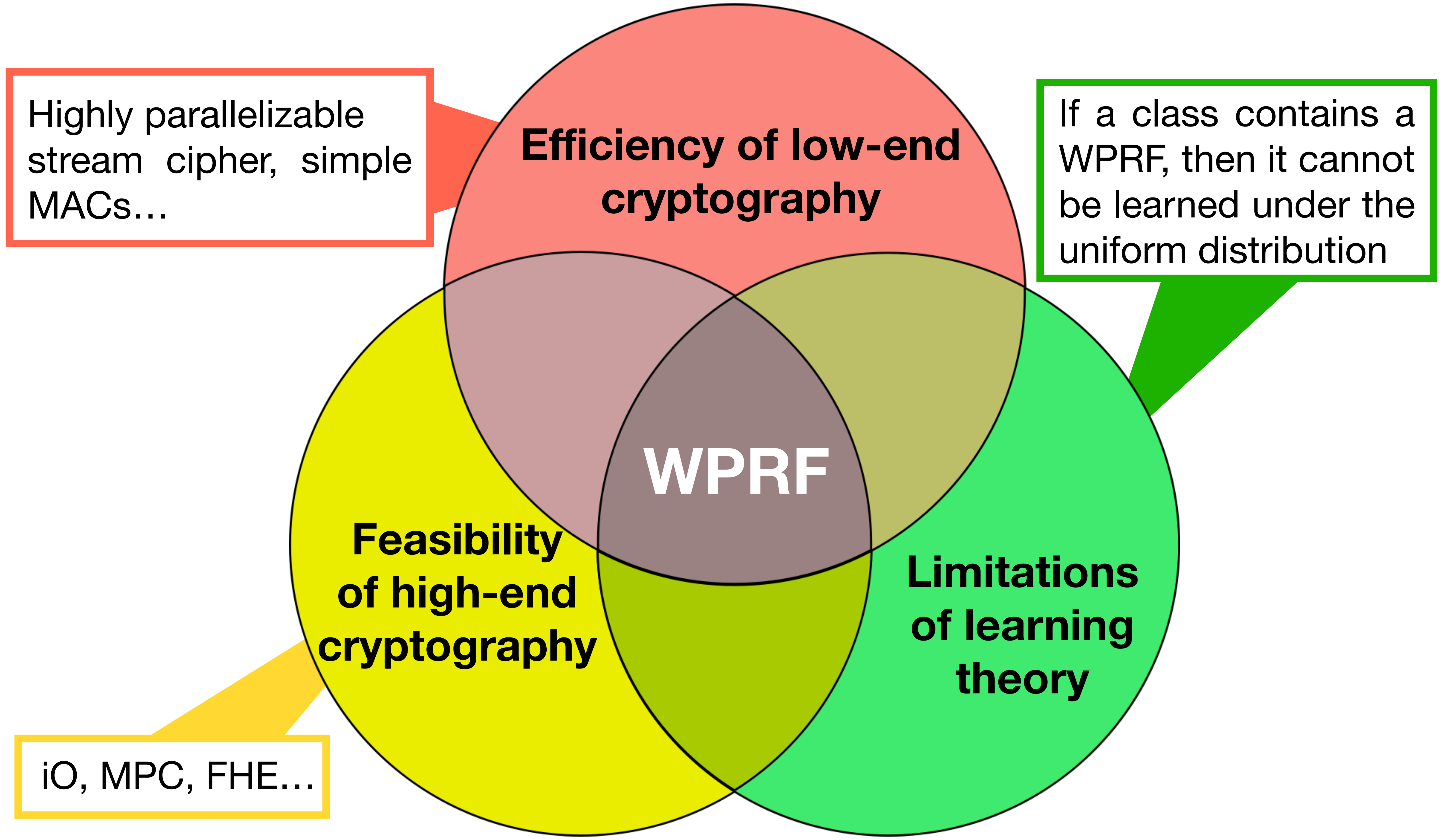
$\approx$

Low-Complexity Weak PRFs

WPRF: F_K is indistinguishable from a random function when evaluated on random inputs.

How low can we go?

Copyright Yuval Ishai - 2020



Highly parallelizable stream cipher, simple MACs…

Efficiency of low-end cryptography

If a class contains a WPRF, then it cannot be learned under the uniform distribution

WPRF

Feasibility of high-end cryptography

Limitations of learning theory

iO, MPC, FHE…

Pseudorandom Correlation Functions and Low-Complexity WPRFs

What we show in the paper

- If you have *Function Secret Sharing* (FSS) for a class C that contains a weak pseudorandom function, then there is a pseudorandom correlation function for the OT correlation.
- If you have *Function Secret Sharing* (FSS) for the class $C^{(2)} = \{f_1 f_2 \ : \ f_1, f_2 \in C\}$ where C contains a weak pseudorandom function, then there is a pseudorandom correlation function for any additive bilinear correlation (authenticated Beaver triples, OLE, inner products, etc).

Pseudorandom Correlation Functions and Low-Complexity WPRFs

What we show in the paper

- If you have *Function Secret Sharing* (FSS) for a class C that contains a weak pseudorandom function, then there is a pseudorandom correlation function for the OT correlation.
- If you have *Function Secret Sharing* (FSS) for the class $C^{(2)} = \{f_1 f_2 \ : \ f_1, f_2 \in C\}$ where C contains a weak pseudorandom function, then there is a pseudorandom correlation function for any additive bilinear correlation (authenticated Beaver triples, OLE, inner products, etc).

FSS for C: for any $f \in C$, share$(f) \to (k_1, k_2)$ such that $\forall x$, Eval(k_1, x) + Eval$(k_2, x) = f(x)$, yet k_i hides f.

Pseudorandom Correlation Functions and Low-Complexity WPRFs

What we show in the paper

- If you have *Function Secret Sharing* (FSS) for a class C that contains a weak pseudorandom function, then there is a pseudorandom correlation function for the OT correlation.
- If you have *Function Secret Sharing* (FSS) for the class $C^{(2)} = \{f_1 f_2 \ : \ f_1, f_2 \in C\}$ where C contains a weak pseudorandom function, then there is a pseudorandom correlation function for any additive bilinear correlation (authenticated Beaver triples, OLE, inner products, etc).

FSS for C: for any $f \in C$, share$(f) \to (k_1, k_2)$ such that $\forall x$, Eval(k_1, x) + Eval$(k_2, x) = f(x)$, yet k_i hides f.

High-End

Any WPRF + FSS for all circuits [BGI15, DHRW16]

- Many limitations: imperfect correctness, requires very powerful assumptions (FHE-style)
- Unlikely to lead to concretely efficient candidates

Pseudorandom Correlation Functions and Low-Complexity WPRFs

What we show in the paper

- If you have *Function Secret Sharing* (FSS) for a class C that contains a weak pseudorandom function, then there is a pseudorandom correlation function for the OT correlation.
- If you have *Function Secret Sharing* (FSS) for the class $C^{(2)} = \{f_1 f_2 \ : \ f_1, f_2 \in C\}$ where C contains a weak pseudorandom function, then there is a pseudorandom correlation function for any additive bilinear correlation (authenticated Beaver triples, OLE, inner products, etc).

FSS for C: for any $f \in C$, share$(f) \rightarrow (k_1, k_2)$ such that $\forall x$, Eval(k_1, x) + Eval$(k_2, x) = f(x)$, yet k_i hides f.

High-End

Any WPRF + FSS for all circuits [BGI15, DHRW16]

- Many limitations: imperfect correctness, requires very powerful assumptions (FHE-style)
- Unlikely to lead to concretely efficient candidates

Low-End

Using efficient FSS?

- FSS for sums of point functions exists from OWFs [BGI16]
- Existing constructions are very efficient
- $\Longrightarrow$ Can we have a WPRF in this low complexity class?

Pseudorandom Correlation Functions and Low-Complexity WPRFs

What we show in the paper

- If you have *Function Secret Sharing* (FSS) for a class C that contains a weak pseudorandom function, then there is a pseudorandom correlation function for the OT correlation.
- If you have *Function Secret Sharing* (FSS) for the class $C^{(2)} = \{f_1 f_2 \ : \ f_1, f_2 \in C\}$ where C contains a weak pseudorandom function, then there is a pseudorandom correlation function for any additive bilinear correlation (authenticated Beaver triples, OLE, inner products, etc).

FSS for C: for any $f \in C$, share$(f) \rightarrow (k_1, k_2)$ such that $\forall x$, Eval(k_1, x) + Eval$(k_2, x) = f(x)$, yet k_i hides f.

High-End

Any WPRF + FSS for all circuits [BGI15, DHRW16]

- Many limitations: imperfect correctness, requires very powerful assumptions (FHE-style)
- Unlikely to lead to concretely efficient candidates

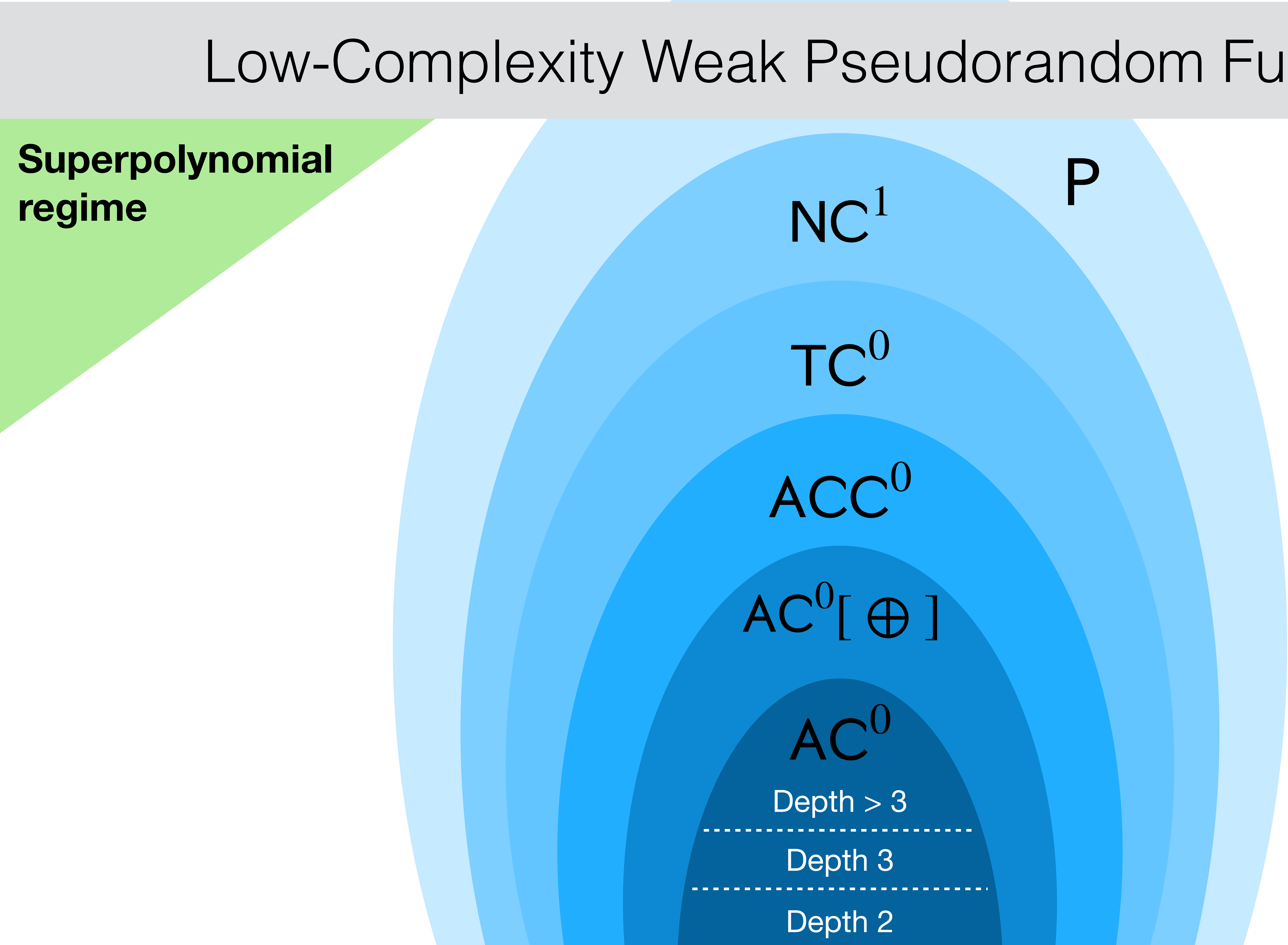
Low-End

Using efficient FSS?

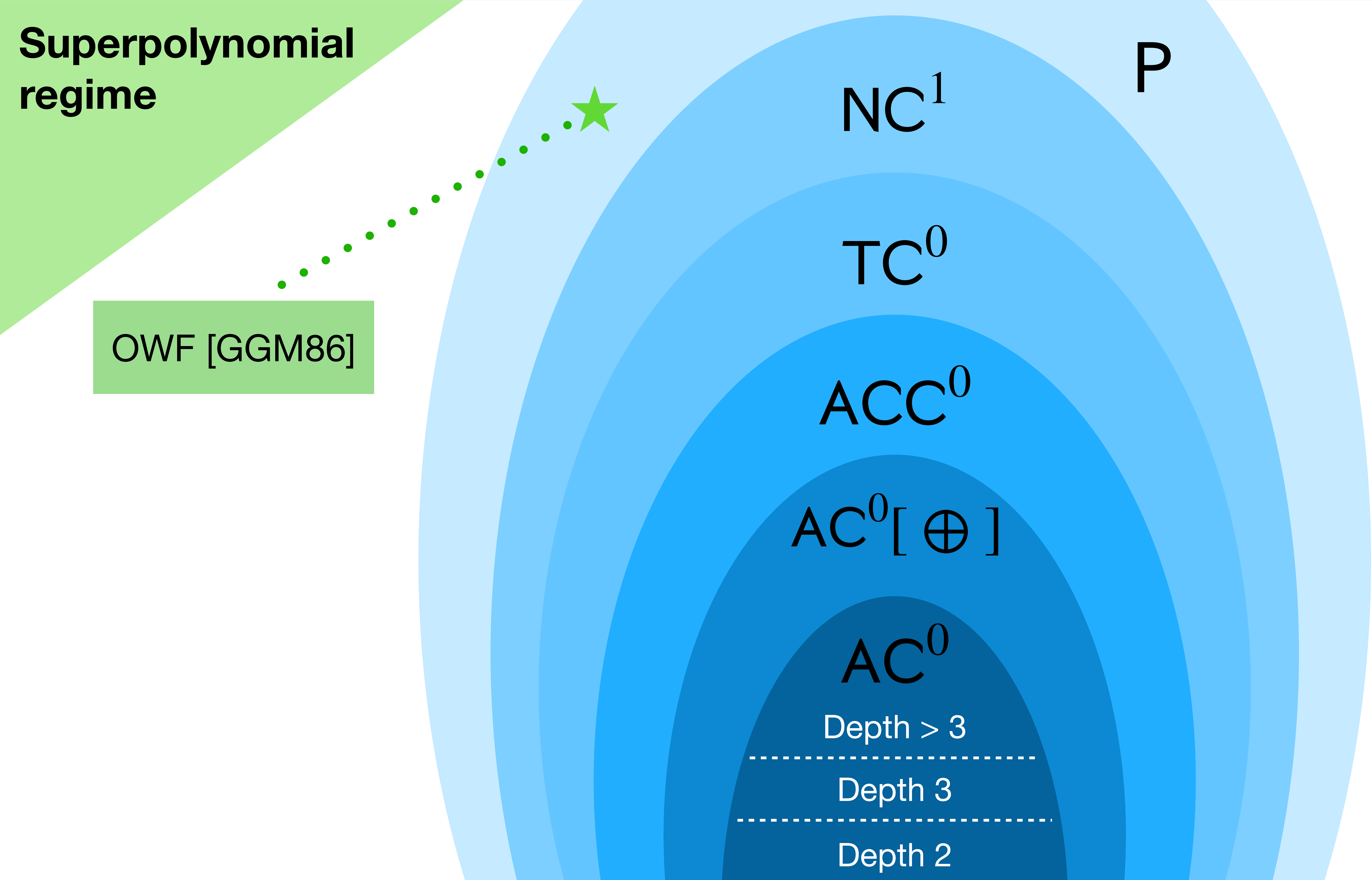
- FSS for sums of point functions exists from OWFs [BGI16]
- Existing constructions are very efficient
- $\Longrightarrow$ Can we have a WPRF in this low complexity class?

This talk: I will present a step-by-step construction of a PCF for OT, from which the new WPRF candidate emerges naturally. The construction does not go through FSS since for the specific case of OT, puncturable PRFs suffice.

Low-Complexity Weak Pseudorandom Functions



Low-Complexity Weak Pseudorandom Functions



Low-Complexity Weak Pseudorandom Functions

Superpolynomial regime
P
Factoring [Kha93]
NC^1
OWF [GGM86]
TC^0
ACC^0
$AC^0[\oplus]$
DDH [NR97]
AC^0
Depth > 3
Depth 3
Depth 2

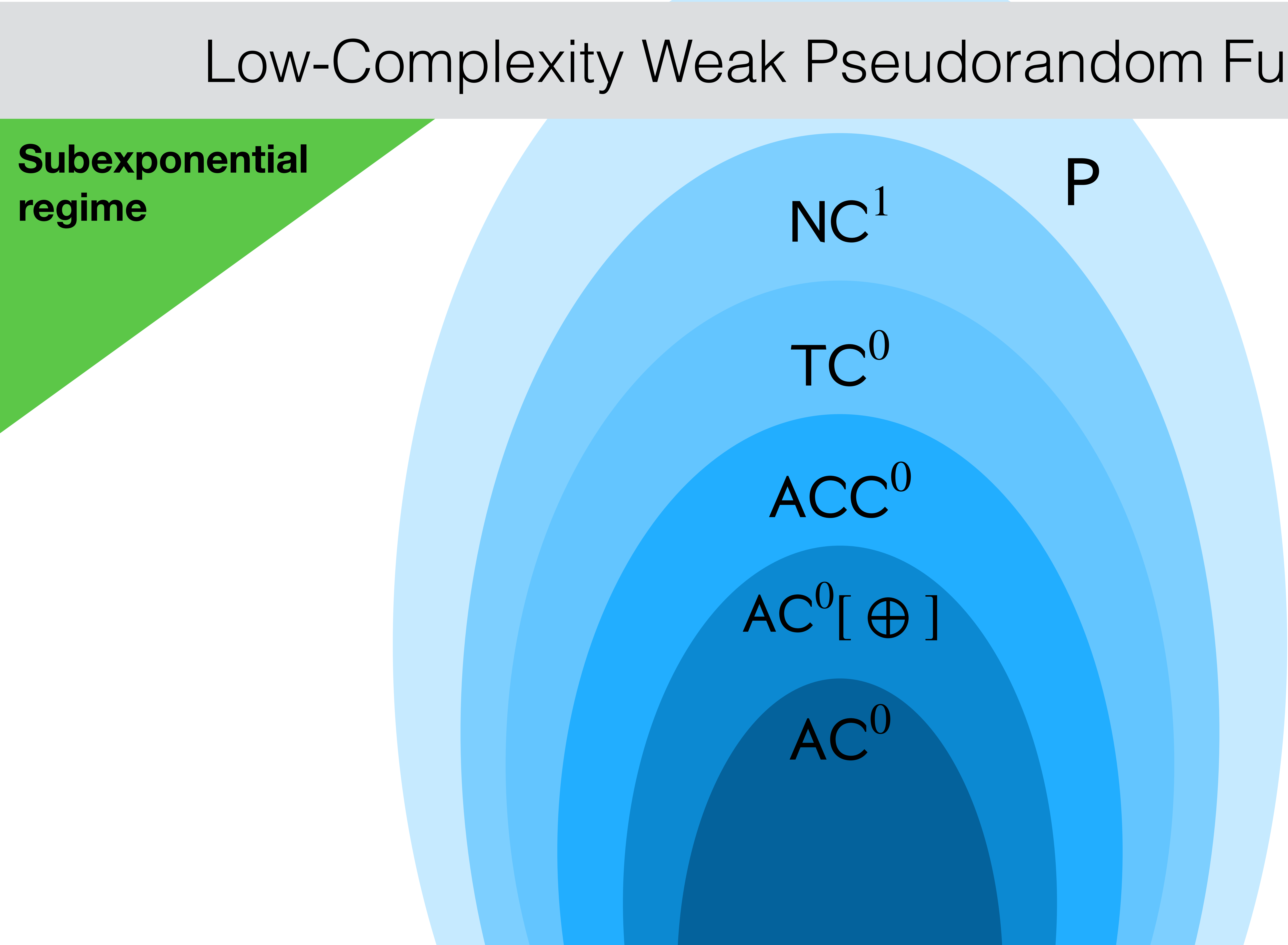
Low-Complexity Weak Pseudorandom Functions

Superpolynomial regime
P
NC^1
TC^0
ACC^0
$AC^0[\oplus]$
AC^0
Depth > 3
Depth 3
Depth 2
Factoring [Kha93]
RLF [AR16]
OWF [GGM86]
DDH [NR97]

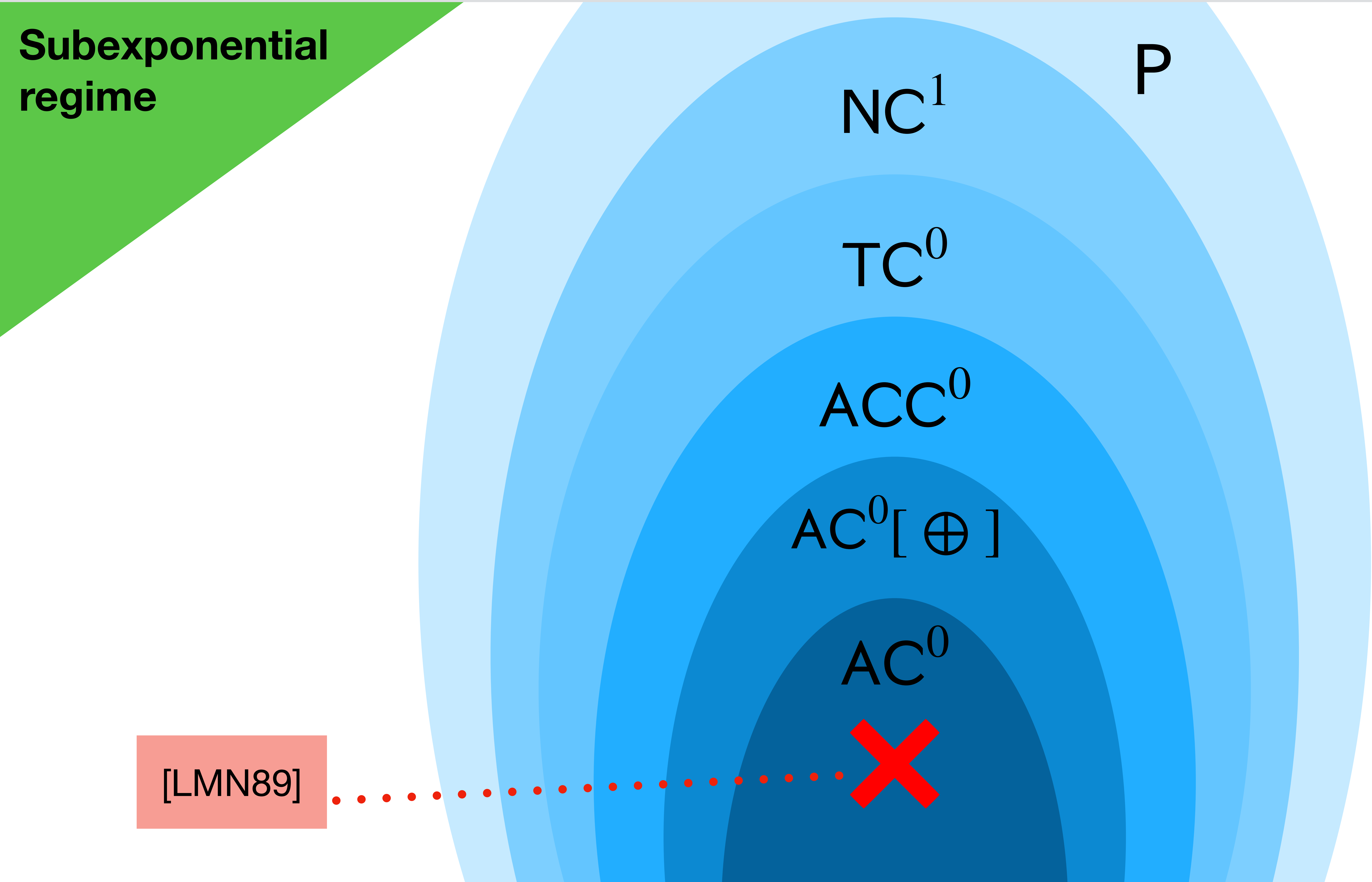
Low-Complexity Weak Pseudorandom Functions

Superpolynomial regime
P
NC^1
TC^0
ACC^0
AC^0[⊕]
AC^0
Depth > 3
Depth 3
Depth 2
OWF [GGM86]
DDH [NR97]
Factoring [Kha93]
RLF [AR16]
Heuristic [BKFL94]

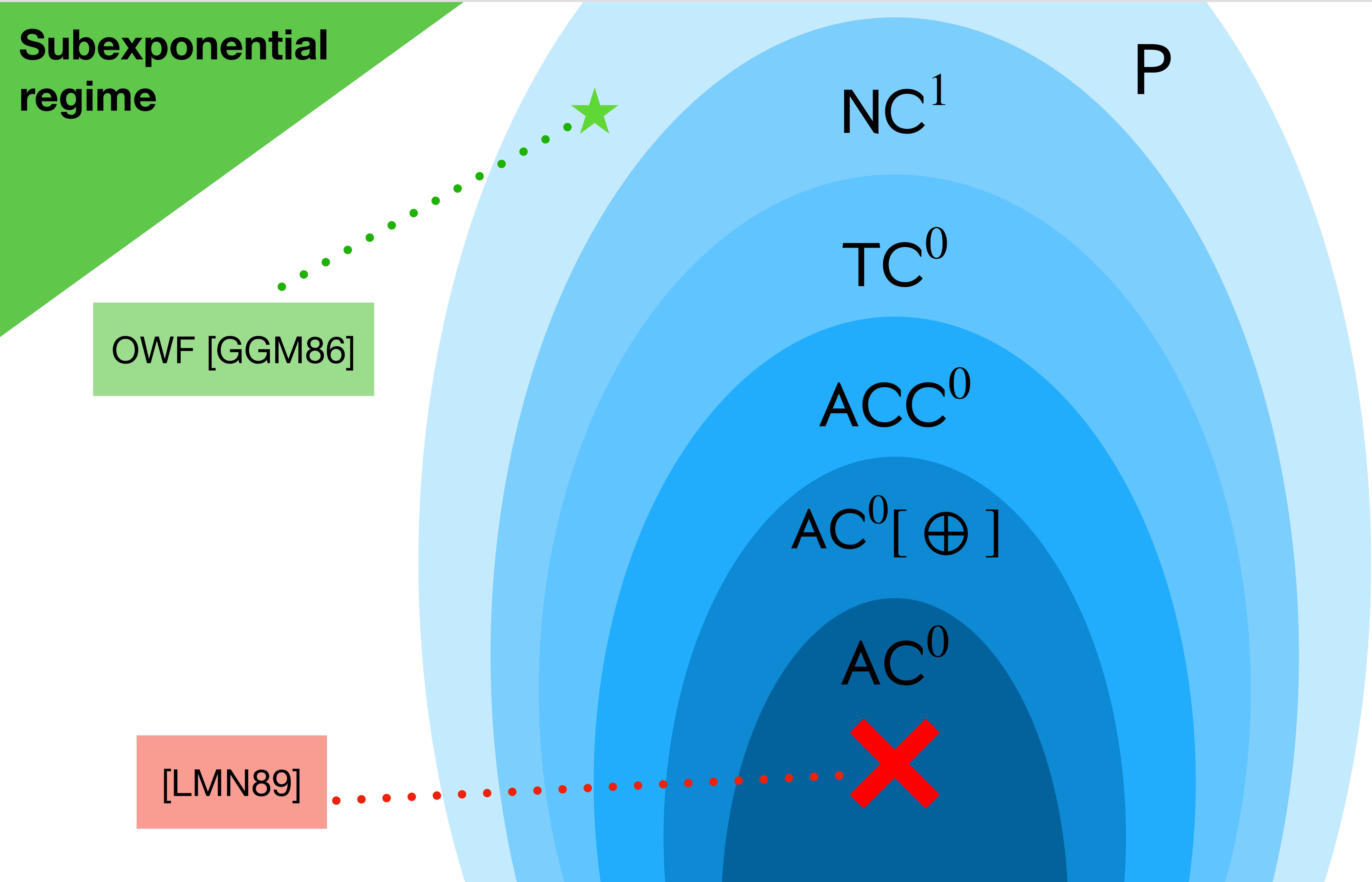
Low-Complexity Weak Pseudorandom Functions



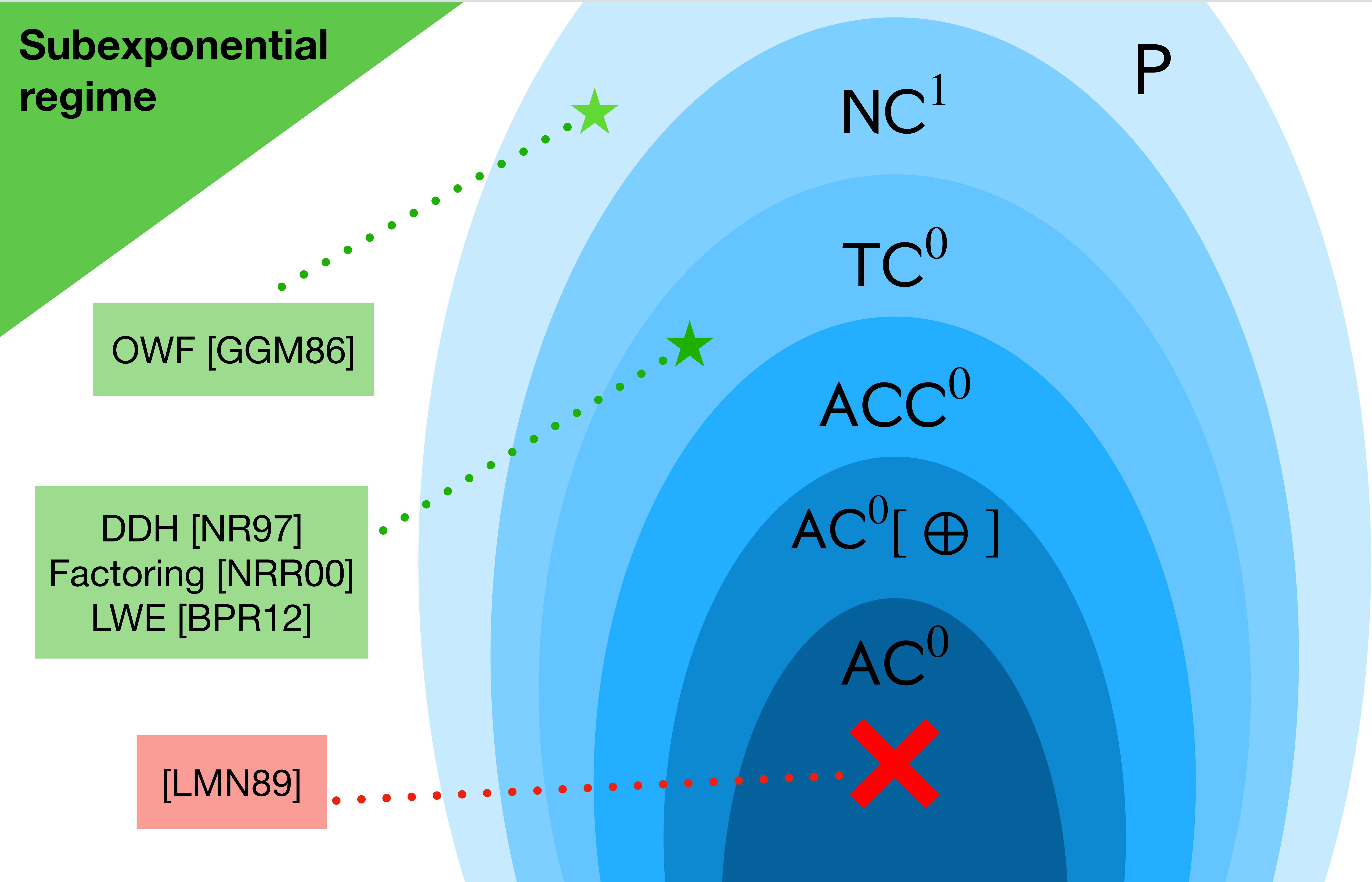
Low-Complexity Weak Pseudorandom Functions



Low-Complexity Weak Pseudorandom Functions



Low-Complexity Weak Pseudorandom Functions



OWF [GGM86]

DDH [NR97]
Factoring [NRR00]
LWE [BPR12]

[LMN89]

Low-Complexity Weak Pseudorandom Functions

Subexponential regime
P
NC^1
TC^0
ACC^0
$AC^0[\oplus]$
AC^0
OWF [GGM86]
DDH [NR97]
Factoring [NRR00]
LWE [BPR12]
[LMN89]
Heuristic [BIPSW18]

Low-Complexity Weak Pseudorandom Functions

Subexponential regime

P

NC^1

TC^0

ACC^0

$AC^0[\oplus]$

AC^0

? ? ? ? ? ?

OWF [GGM86]

DDH [NR97]
Factoring [NRR00]
LWE [BPR12]

Heuristic [BIPSW18]

[LMN89]

Low-Complexity Weak Pseudorandom Functions

P
NC^1
TC^0
ACC^0
$AC^0[\oplus]$
AC^0
?
?
?
?
?
?

Subexponential regime

OWF [GGM86]

DDH [NR97]
Factoring [NRR00]
LWE [BPR12]

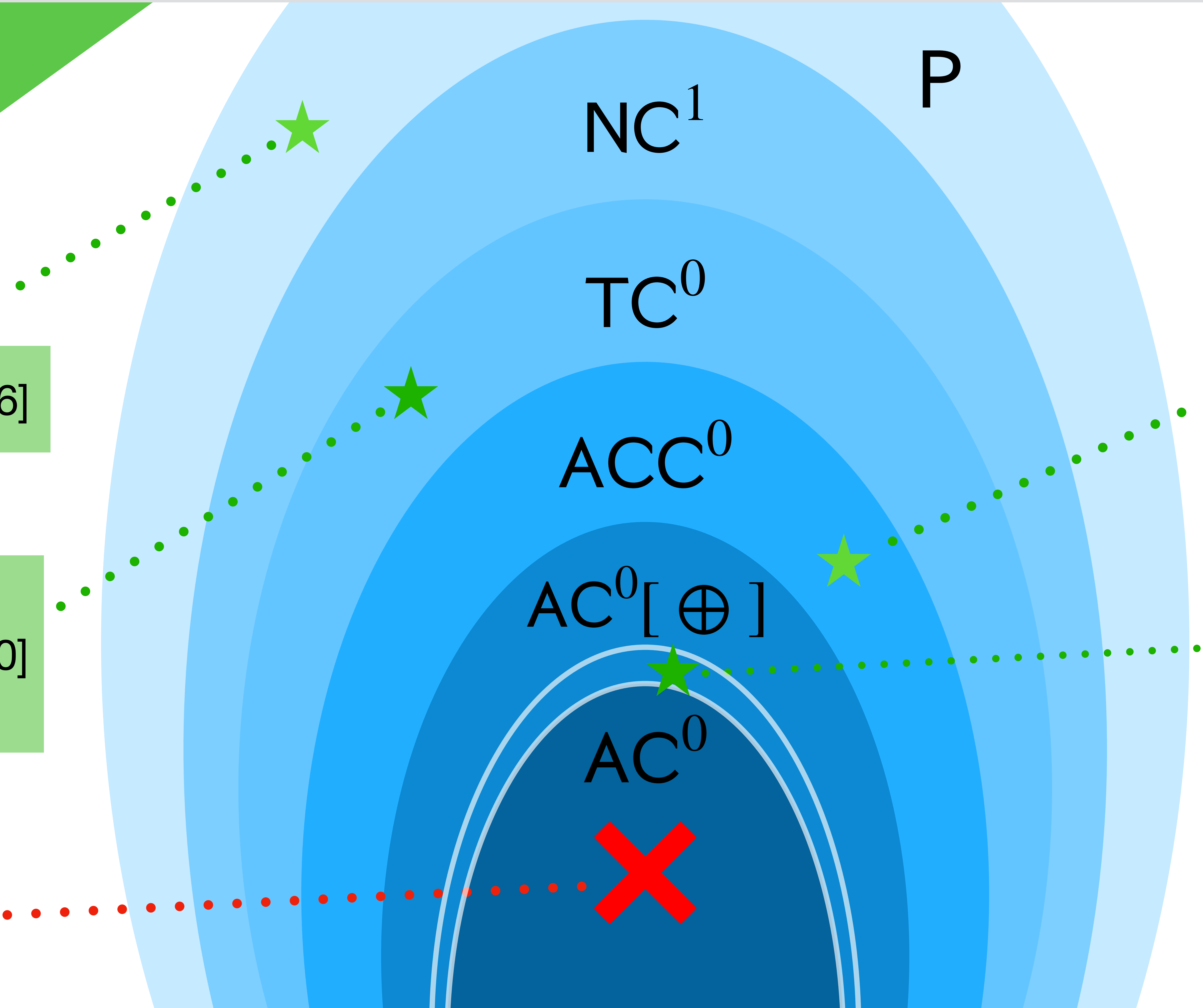
[LMN89]

Heuristic [BIPSW18]

Heuristic [ABGKR14]
… Broken in [BR17]

Low-Complexity Weak Pseudorandom Functions

Subexponential regime



OWF [GGM86]

DDH [NR97]
Factoring [NRR00]
LWE [BPR12]

[LMN89]

Heuristic [BIPSW18]

This work: candidate WPRF for the class of XNF formulas (Xor Normal Form)

Depth 2, one layer of ANDs, followed by a single XOR

The Class of XNF Formulas

XNF formulas are (polynomial-size) depth-2 boolean circuits over literals (inputs and their negation) with one layer of (arbitrary fan-in) AND gates, followed by a single (arbitrary fan-in) XOR gate.

Example: $(\neg X_1 \wedge X_2 \wedge \neg X_3) \oplus (\neg X_4 \wedge X_5 \wedge \neg X_6) \oplus \cdots$

We get the following **conjecture:** XNF formulas are (subexponentially) hard to learn under the uniform distribution.

Concrete structure:

The Class of XNF Formulas

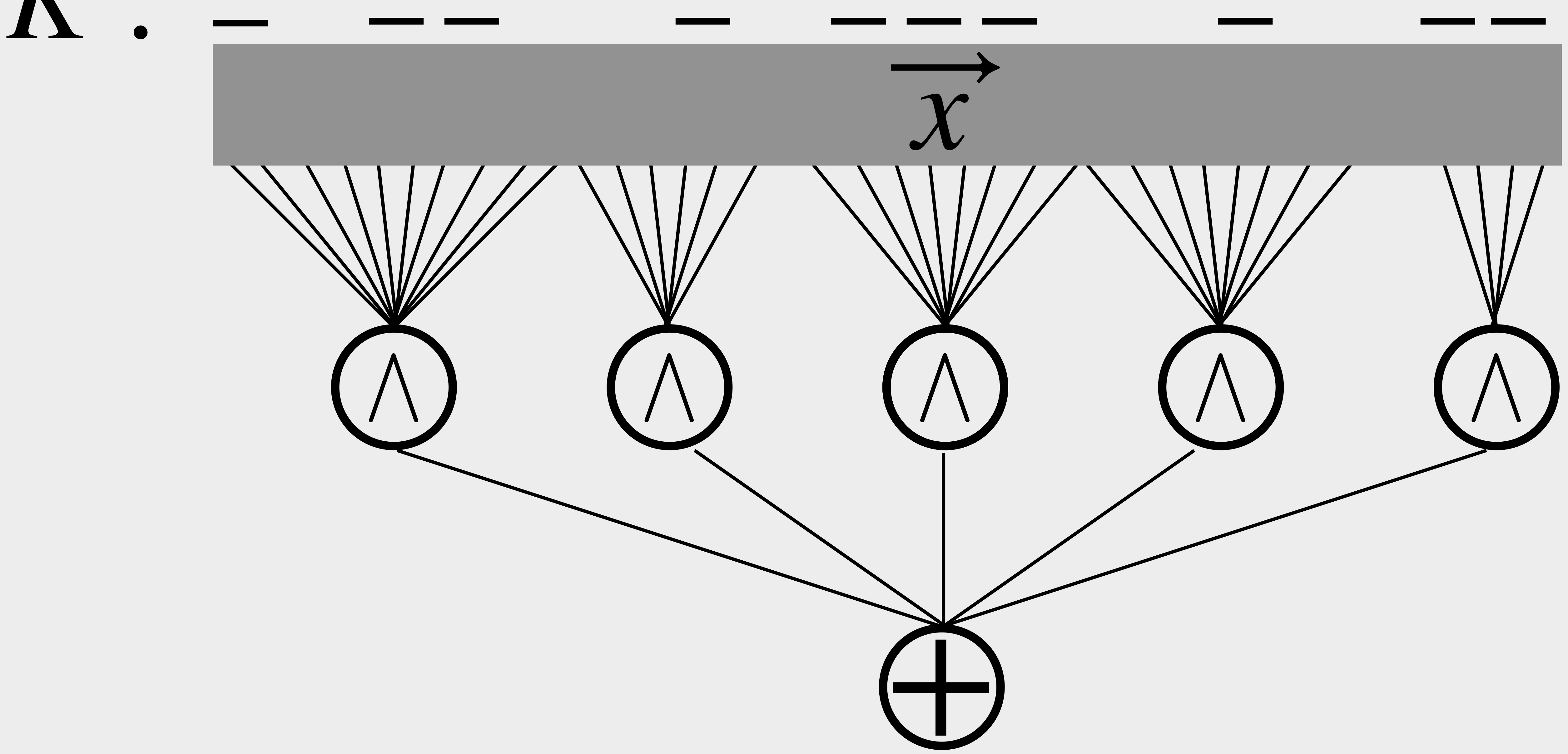
XNF formulas are (polynomial-size) depth-2 boolean circuits over literals (inputs and their negation) with one layer of (arbitrary fan-in) AND gates, followed by a single (arbitrary fan-in) XOR gate.

Example: $(\neg X_1 \wedge X_2 \wedge \neg X_3) \oplus (\neg X_4 \wedge X_5 \wedge \neg X_6) \oplus \cdots$

We get the following **conjecture:** XNF formulas are (subexponentially) hard to learn under the uniform distribution.

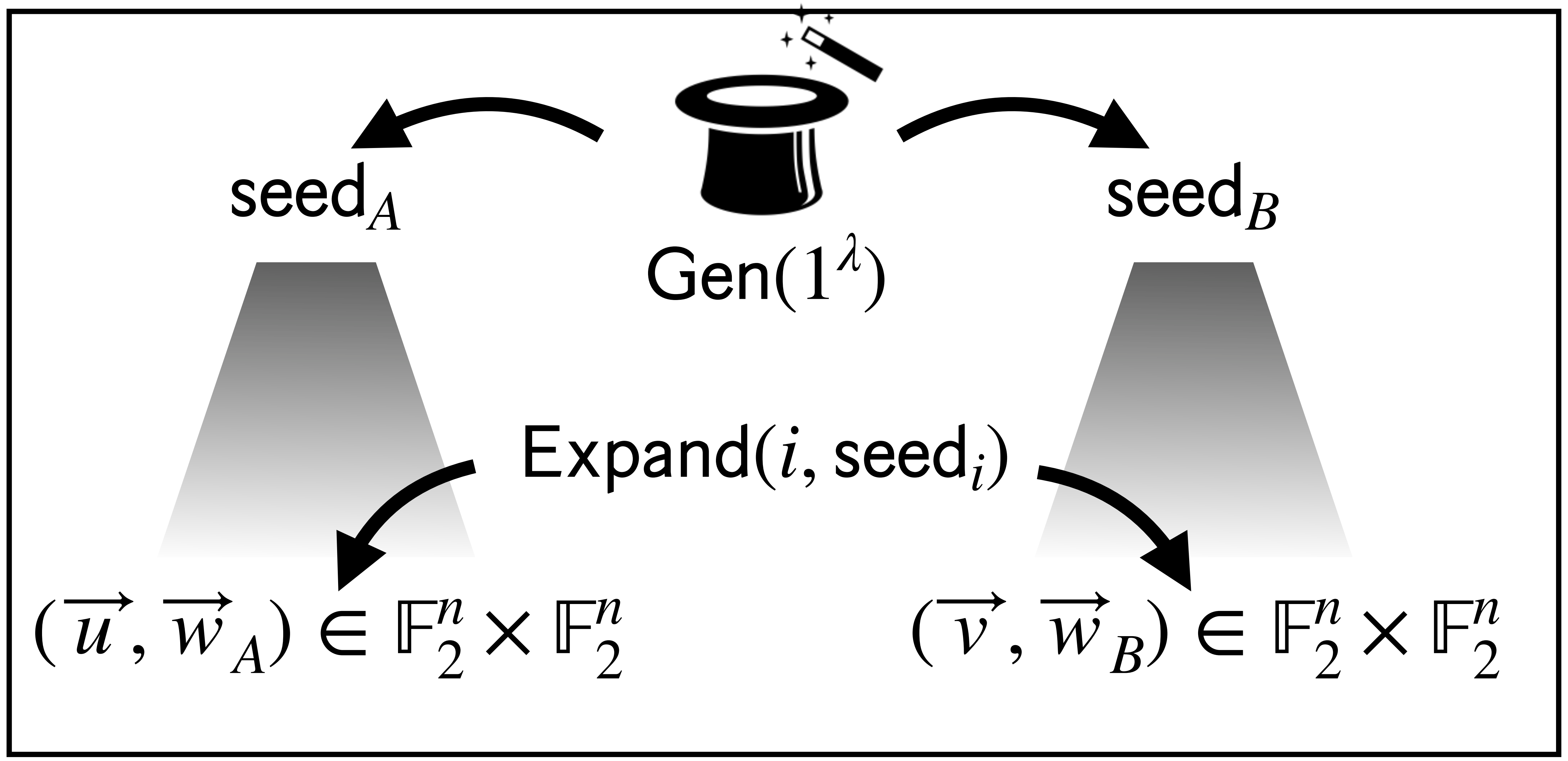
Concrete structure:

The key tells which input bits to negate



Pseudorandom Correlation Generators - Walkthrough

A quick reminder of what we want: Gen generates *short correlated seeds* which can be *locally expanded* into pseudorandom instances of a target correlation.



seed_A
seed_B
$\mathsf{Gen}(1^\lambda)$
$\mathsf{Expand}(i, \mathsf{seed}_i)$
$(\overrightarrow{u}, \overrightarrow{w}_A) \in \mathbb{F}_2^n \times \mathbb{F}_2^n$
$(\overrightarrow{v}, \overrightarrow{w}_B) \in \mathbb{F}_2^n \times \mathbb{F}_2^n$

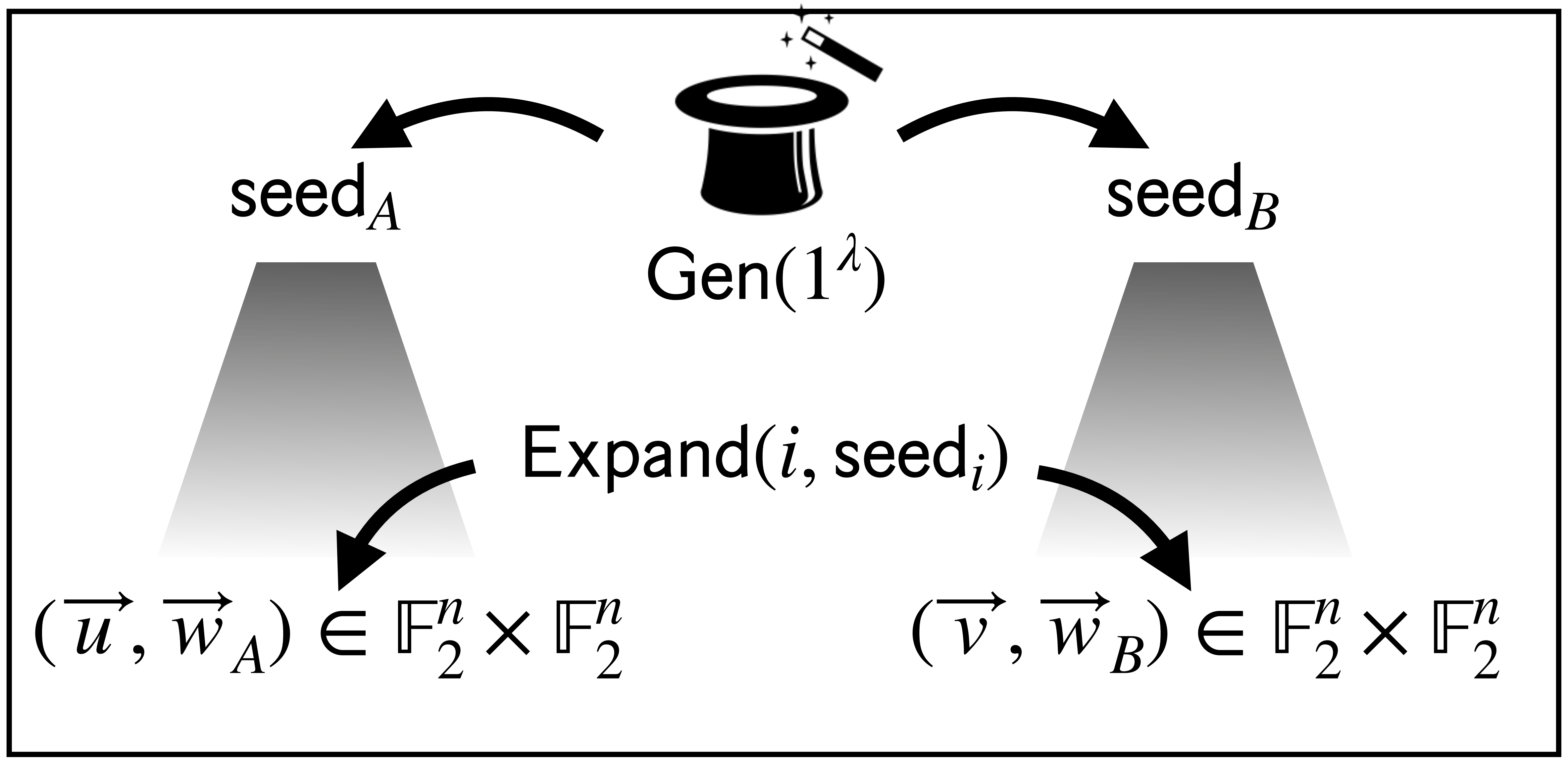
Oblivious transfer correlation:

$$\overrightarrow{w}_A + \overrightarrow{w}_B = \overrightarrow{u} \star \overrightarrow{v}$$

A construction from LPN

Pseudorandom Correlation Generators - Walkthrough

A quick reminder of what we want: Gen generates *short correlated seeds* which can be *locally expanded* into pseudorandom instances of a target correlation.



seed_A
seed_B
$\text{Gen}(1^\lambda)$
$\text{Expand}(i, \text{seed}_i)$
$(\vec{u}, \vec{w}_A) \in \mathbb{F}_2^n \times \mathbb{F}_2^n$
$(\vec{v}, \vec{w}_B) \in \mathbb{F}_2^n \times \mathbb{F}_2^n$

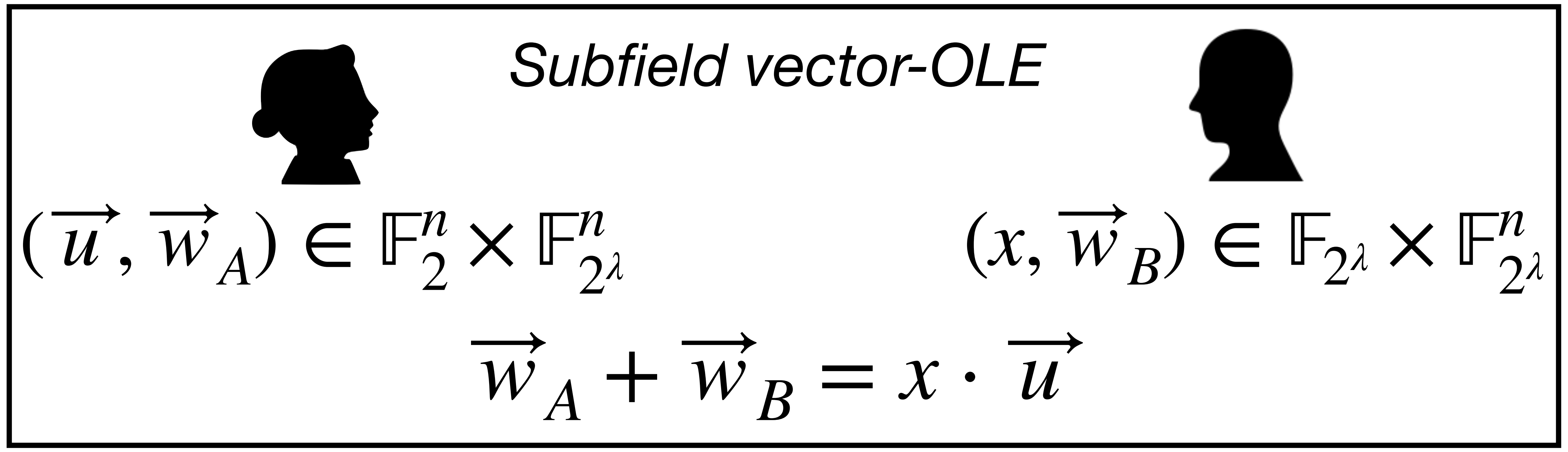
Oblivious transfer correlation:

$$\vec{w}_A + \vec{w}_B = \vec{u} \star \vec{v}$$

A construction from LPN

1. Reduction to subfield-VOLE

[IKNP03]: *subfied vector-OLE* correlation + *correlation-robust hash functions* gives (pseudorandom) OT correlations.



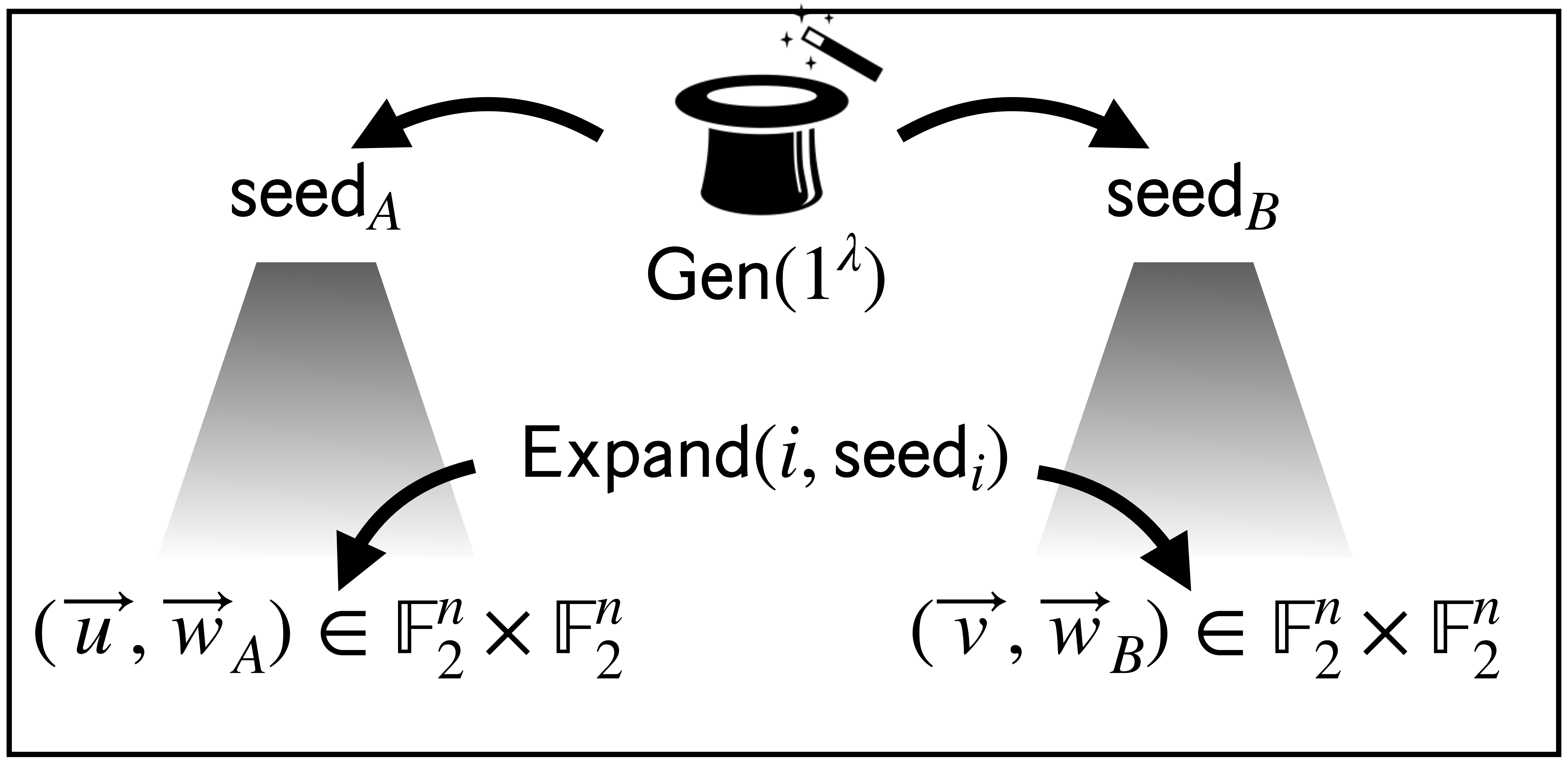
Subfield vector-OLE

$$(\vec{u}, \vec{w}_A) \in \mathbb{F}_2^n \times \mathbb{F}_{2^\lambda}^n \qquad (x, \vec{w}_B) \in \mathbb{F}_{2^\lambda} \times \mathbb{F}_{2^\lambda}^n$$

$$\vec{w}_A + \vec{w}_B = x \cdot \vec{u}$$

Pseudorandom Correlation Generators - Walkthrough

A quick reminder of what we want: Gen generates *short correlated seeds* which can be *locally expanded* into pseudorandom instances of a target correlation.



seed_A

$\mathsf{Gen}(1^\lambda)$

seed_B

$\mathsf{Expand}(i, \mathsf{seed}_i)$

$(\overrightarrow{u}, \overrightarrow{w}_A) \in \mathbb{F}_2^n \times \mathbb{F}_2^n$

$(\overrightarrow{v}, \overrightarrow{w}_B) \in \mathbb{F}_2^n \times \mathbb{F}_2^n$

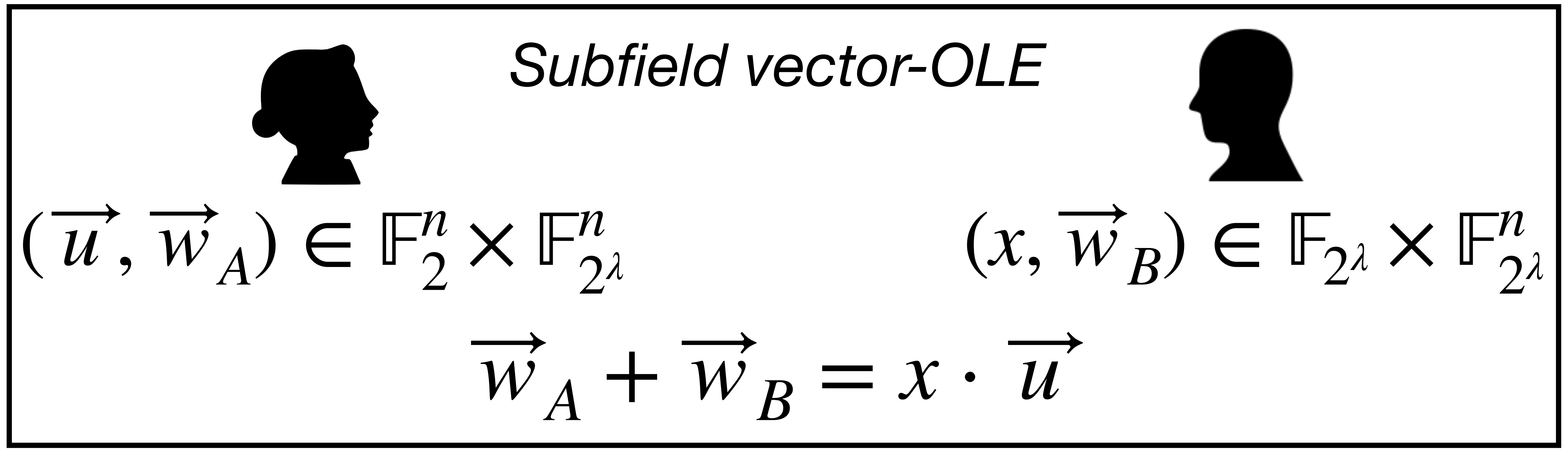
Oblivious transfer correlation:

$$\overrightarrow{w}_A + \overrightarrow{w}_B = \overrightarrow{u} \star \overrightarrow{v}$$

A construction from LPN

1. Reduction to subfield-VOLE

[IKNP03]: *subfied vector-OLE* correlation + *correlation-robust hash functions* gives (pseudorandom) OT correlations.



Subfield vector-OLE

$(\overrightarrow{u}, \overrightarrow{w}_A) \in \mathbb{F}_2^n \times \mathbb{F}_{2^\lambda}^n$

$(x, \overrightarrow{w}_B) \in \mathbb{F}_{2^\lambda} \times \mathbb{F}_{2^\lambda}^n$

$$\overrightarrow{w}_A + \overrightarrow{w}_B = x \cdot \overrightarrow{u}$$

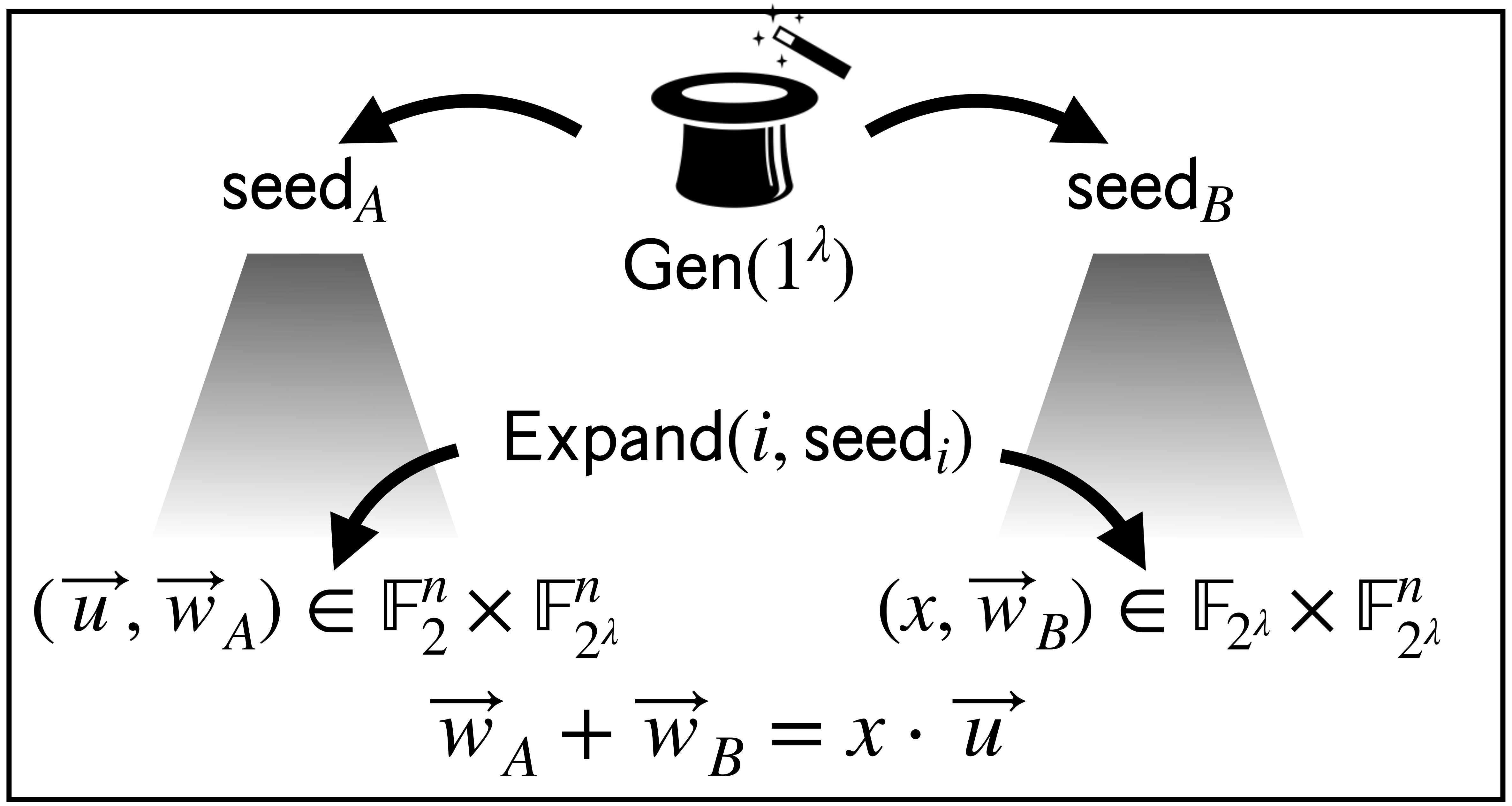
Intuition. the i-th (string-) OT is:

- $(s_0, s_1) = (H(-w_{B,i}), H(x - w_{B,i}))$
- $(b, s_b) = (u_i, H(w_{A,i}))$

where H is a correlation-robust hash function.

Pseudorandom Correlation Generators - Walkthrough

A quick reminder of what we want: Gen generates *short correlated seeds* which can be *locally expanded* into pseudorandom instances of a target correlation.



seed_A
seed_B
$\mathsf{Gen}(1^\lambda)$
$\mathsf{Expand}(i, \mathsf{seed}_i)$

$(\vec{u}, \vec{w}_A) \in \mathbb{F}_2^n \times \mathbb{F}_{2^\lambda}^n$ $\quad (x, \vec{w}_B) \in \mathbb{F}_{2^\lambda} \times \mathbb{F}_{2^\lambda}^n$

$$\vec{w}_A + \vec{w}_B = x \cdot \vec{u}$$

New target

A construction from LPN

1. Reduction to subfield-VOLE

2. Constructing a PCG for subfield-VOLE

Three steps:

1. Construction for a random *unit vector* $\vec{u}$
from puncturable pseudorandom functions

2. Construction for a random t-*sparse vector* $\vec{u}$
via t parallel repetitions of (1)

3. Construction for a pseudorandom vector $\vec{u}$
using dual-LPN

Pseudorandom Correlation Generators - Walkthrough

1 Construction for a random *unit vector* $\overrightarrow{u}$

from puncturable pseudorandom functions

$(\alpha : \; u_\alpha = 1)$

$\mathsf{seed}_A = (x,$

$\mathsf{seed}_B = (\overrightarrow{u},$

A construction from LPN

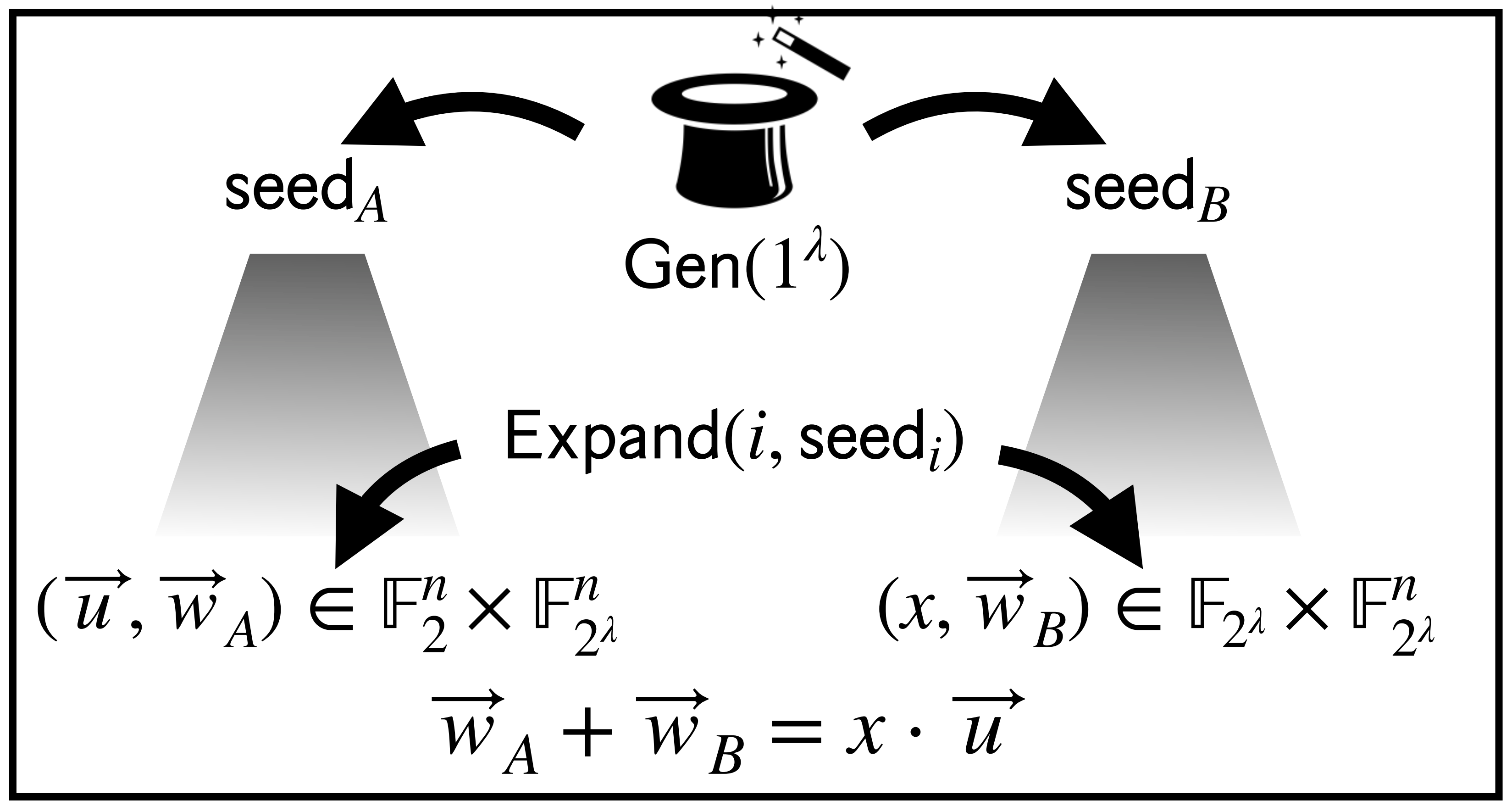
1. Reduction to subfield-VOLE

2. Constructing a PCG for subfield-VOLE

Three steps:

1 Construction for a random *unit vector* $\overrightarrow{u}$

from puncturable pseudorandom functions



seed_A

$\mathsf{Gen}(1^\lambda)$

seed_B

$\mathsf{Expand}(i, \mathsf{seed}_i)$

$(\overrightarrow{u}, \overrightarrow{w}_A) \in \mathbb{F}_2^n \times \mathbb{F}_{2^\lambda}^n$

$(x, \overrightarrow{w}_B) \in \mathbb{F}_{2^\lambda} \times \mathbb{F}_{2^\lambda}^n$

$\overrightarrow{w}_A + \overrightarrow{w}_B = x \cdot \overrightarrow{u}$

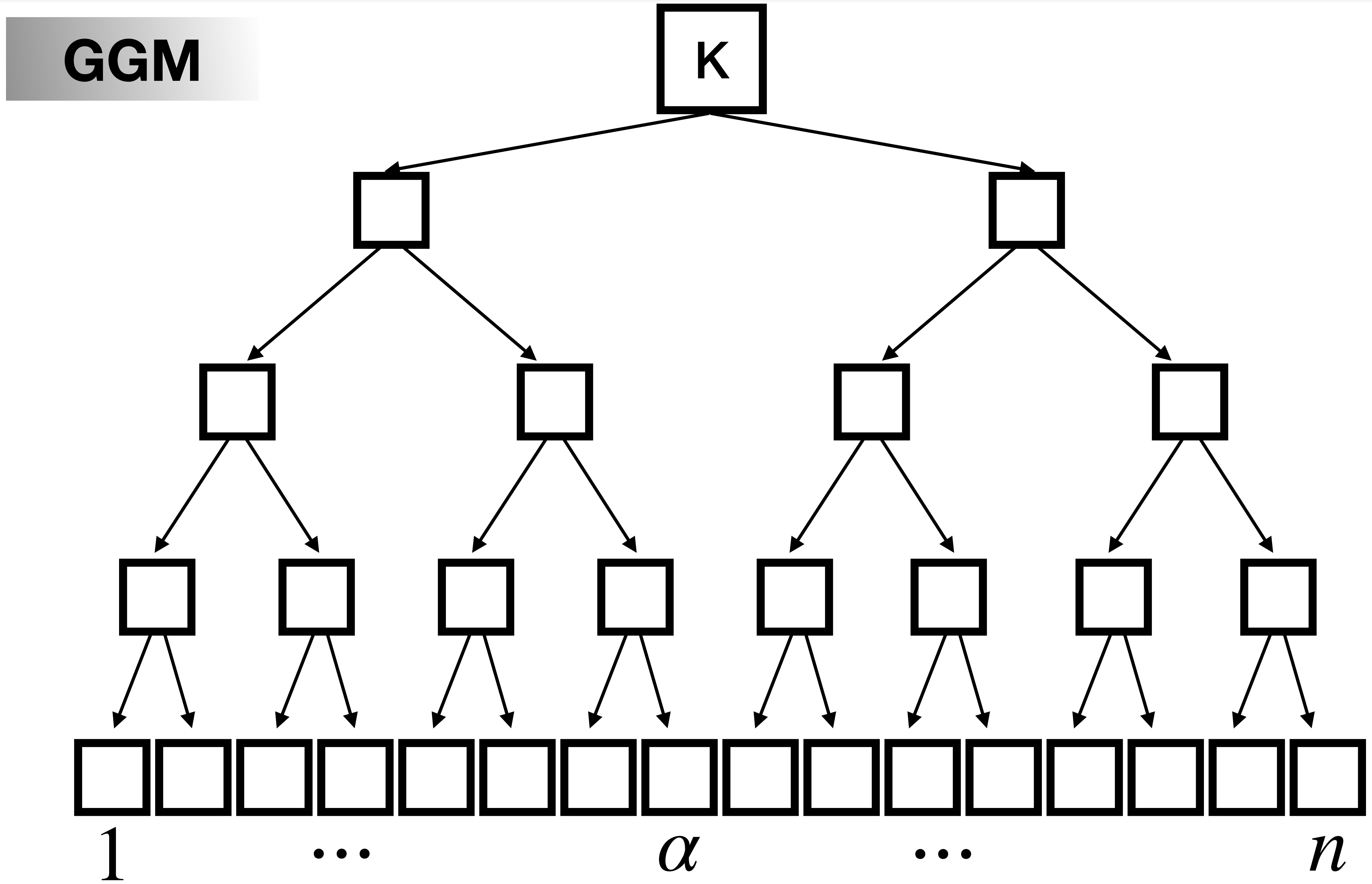
Pseudorandom Correlation Generators - Walkthrough

1 Construction for a random *unit vector* $\overrightarrow{u}$
from puncturable pseudorandom functions

$(\alpha : \ u_\alpha = 1)$

$\text{seed}_A = (x,$

$\text{seed}_B = (\overrightarrow{u},$



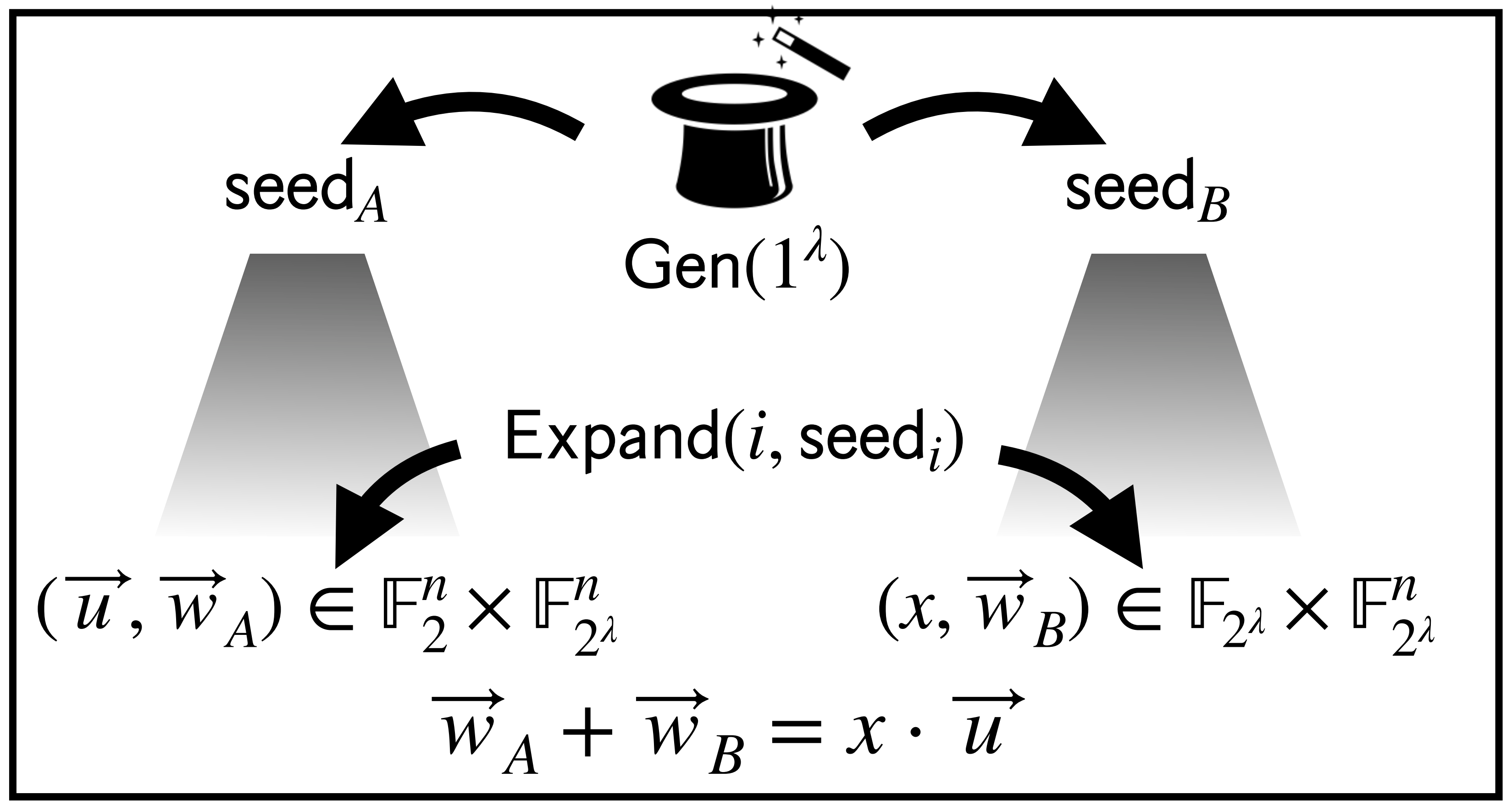
A construction from LPN

1. Reduction to subfield-VOLE

2. Constructing a PCG for subfield-VOLE

Three steps:

1 Construction for a random *unit vector* $\overrightarrow{u}$
from puncturable pseudorandom functions

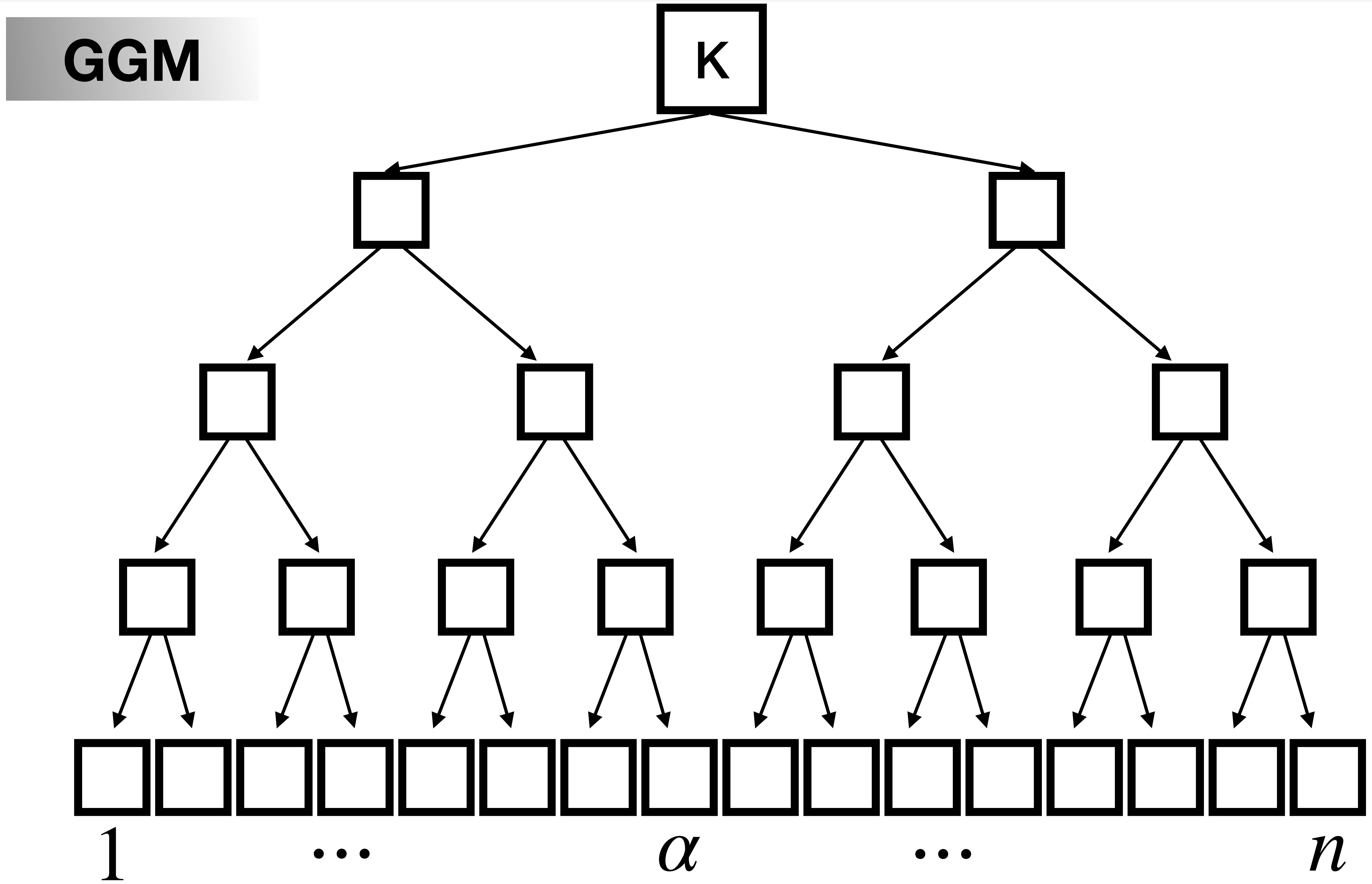

seed_A
$\text{Gen}(1^\lambda)$
seed_B
$\text{Expand}(i, \text{seed}_i)$
$(\overrightarrow{u}, \overrightarrow{w}_A) \in \mathbb{F}_2^n \times \mathbb{F}_{2^\lambda}^n$
$(x, \overrightarrow{w}_B) \in \mathbb{F}_{2^\lambda} \times \mathbb{F}_{2^\lambda}^n$
$\overrightarrow{w}_A + \overrightarrow{w}_B = x \cdot \overrightarrow{u}$

Pseudorandom Correlation Generators - Walkthrough

1 Construction for a random *unit vector* $\overrightarrow{u}$
from puncturable pseudorandom functions

$(\alpha : \; u_\alpha = 1)$

$\mathsf{seed}_A = (x, K)$ $\mathsf{seed}_B = (\overrightarrow{u},$



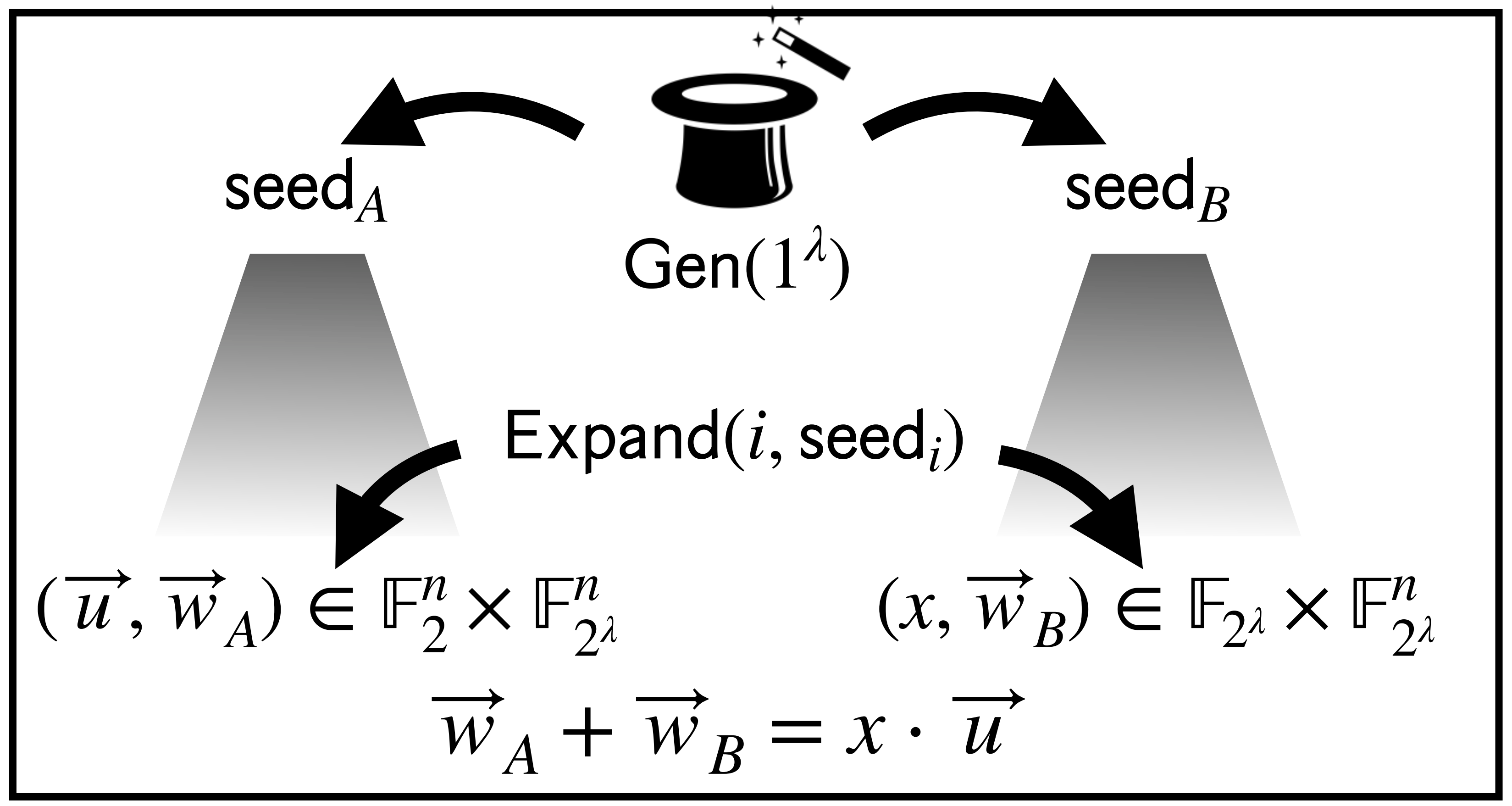
A construction from LPN

1. Reduction to subfield-VOLE

2. Constructing a PCG for subfield-VOLE

Three steps:

1 Construction for a random *unit vector* $\overrightarrow{u}$
from puncturable pseudorandom functions


seed_A
$\mathsf{Gen}(1^\lambda)$
seed_B
$\mathsf{Expand}(i, \mathsf{seed}_i)$

$(\overrightarrow{u}, \overrightarrow{w}_A) \in \mathbb{F}_2^n \times \mathbb{F}_{2^\lambda}^n$ $(x, \overrightarrow{w}_B) \in \mathbb{F}_{2^\lambda} \times \mathbb{F}_{2^\lambda}^n$

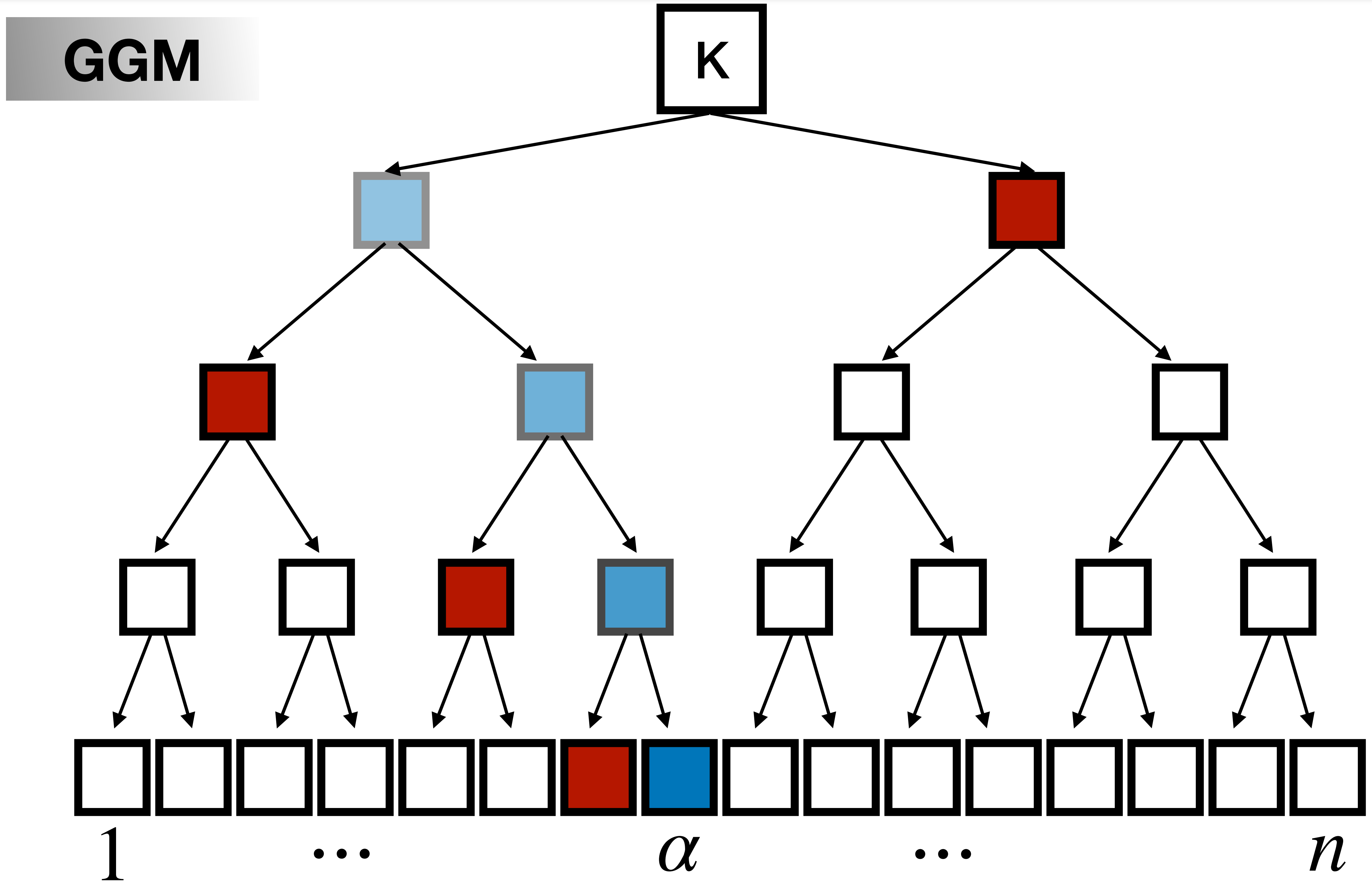
$\overrightarrow{w}_A + \overrightarrow{w}_B = x \cdot \overrightarrow{u}$

Pseudorandom Correlation Generators - Walkthrough

1 Construction for a random *unit vector* $\overrightarrow{u}$
from puncturable pseudorandom functions

$(\alpha : \; u_\alpha = 1)$

$\text{seed}_A = (x, K)$ $\text{seed}_B = (\overrightarrow{u},$



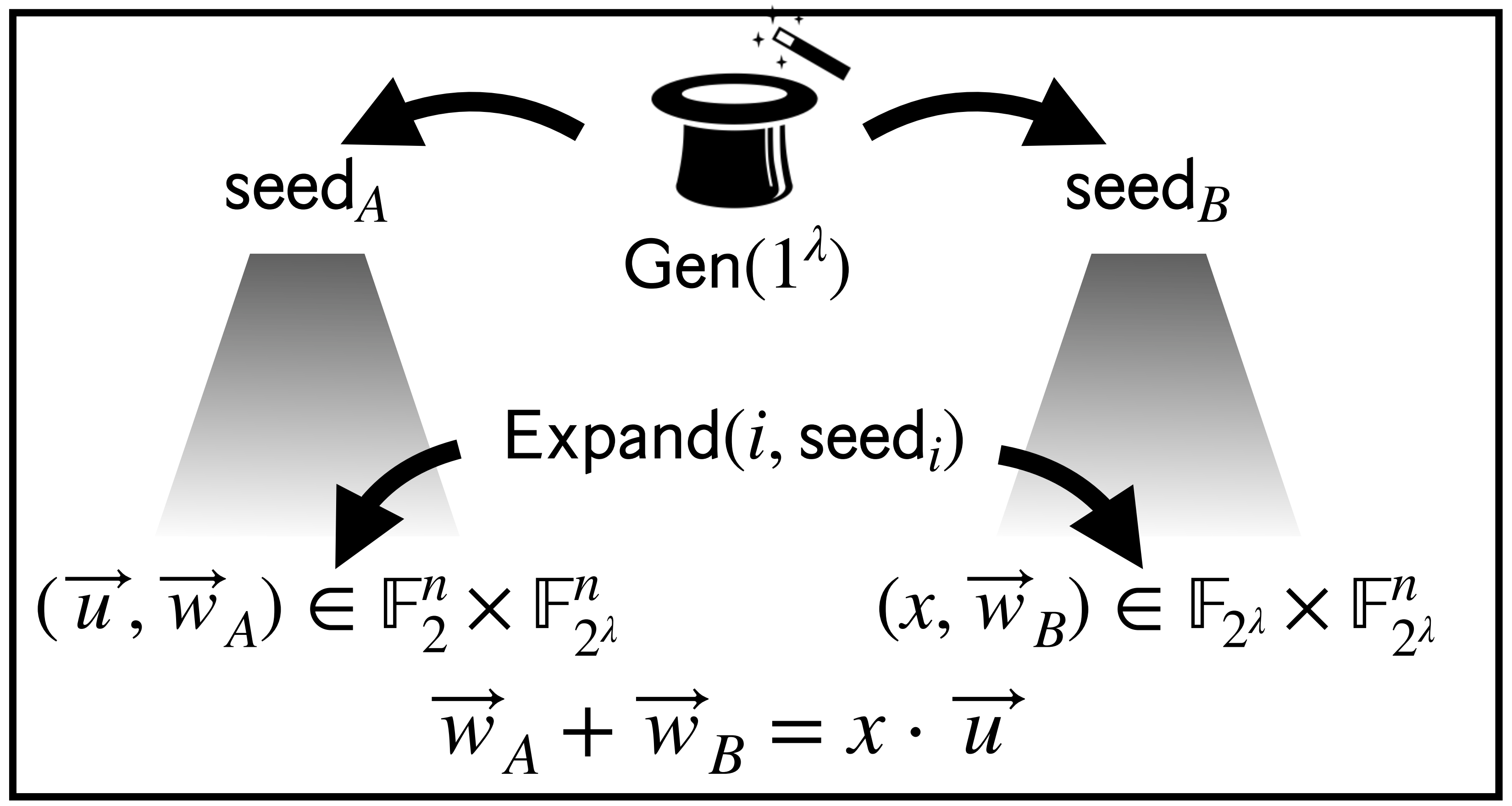
A construction from LPN

1. Reduction to subfield-VOLE

2. Constructing a PCG for subfield-VOLE

Three steps:

1 Construction for a random *unit vector* $\overrightarrow{u}$
from puncturable pseudorandom functions

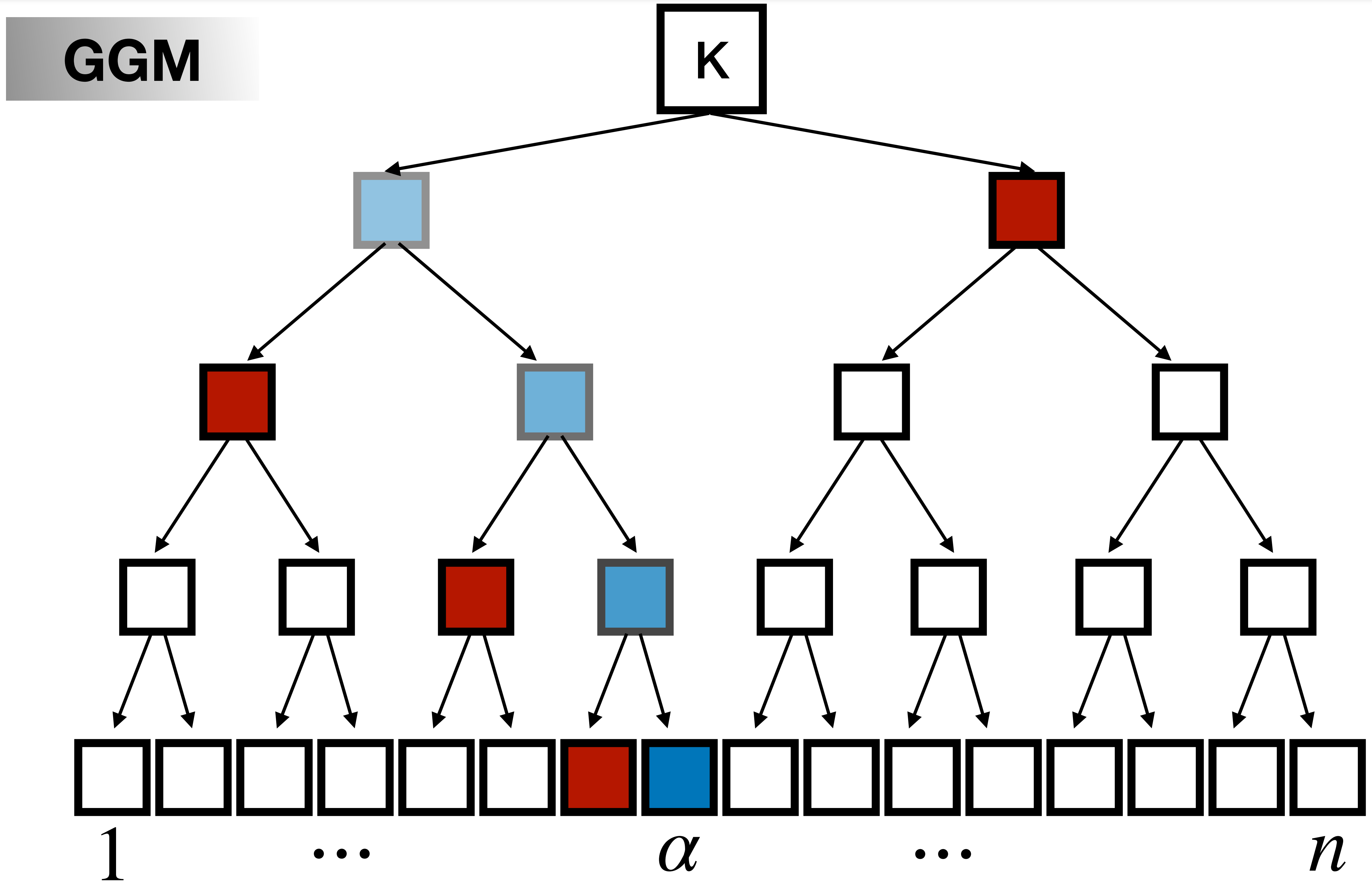

seed_A
$\text{Gen}(1^\lambda)$
seed_B
$\text{Expand}(i, \text{seed}_i)$
$(\overrightarrow{u}, \overrightarrow{w}_A) \in \mathbb{F}_2^n \times \mathbb{F}_{2^\lambda}^n$
$(x, \overrightarrow{w}_B) \in \mathbb{F}_{2^\lambda} \times \mathbb{F}_{2^\lambda}^n$
$\overrightarrow{w}_A + \overrightarrow{w}_B = x \cdot \overrightarrow{u}$

Pseudorandom Correlation Generators - Walkthrough

1 Construction for a random *unit vector* $\vec{u}$
from puncturable pseudorandom functions

$(\alpha : \ u_\alpha = 1)$

$\mathsf{seed}_A = (x, K)$ $\mathsf{seed}_B = (\vec{u},$ 🟥🟥🟥🟥, 🟦 $\oplus\, x)$



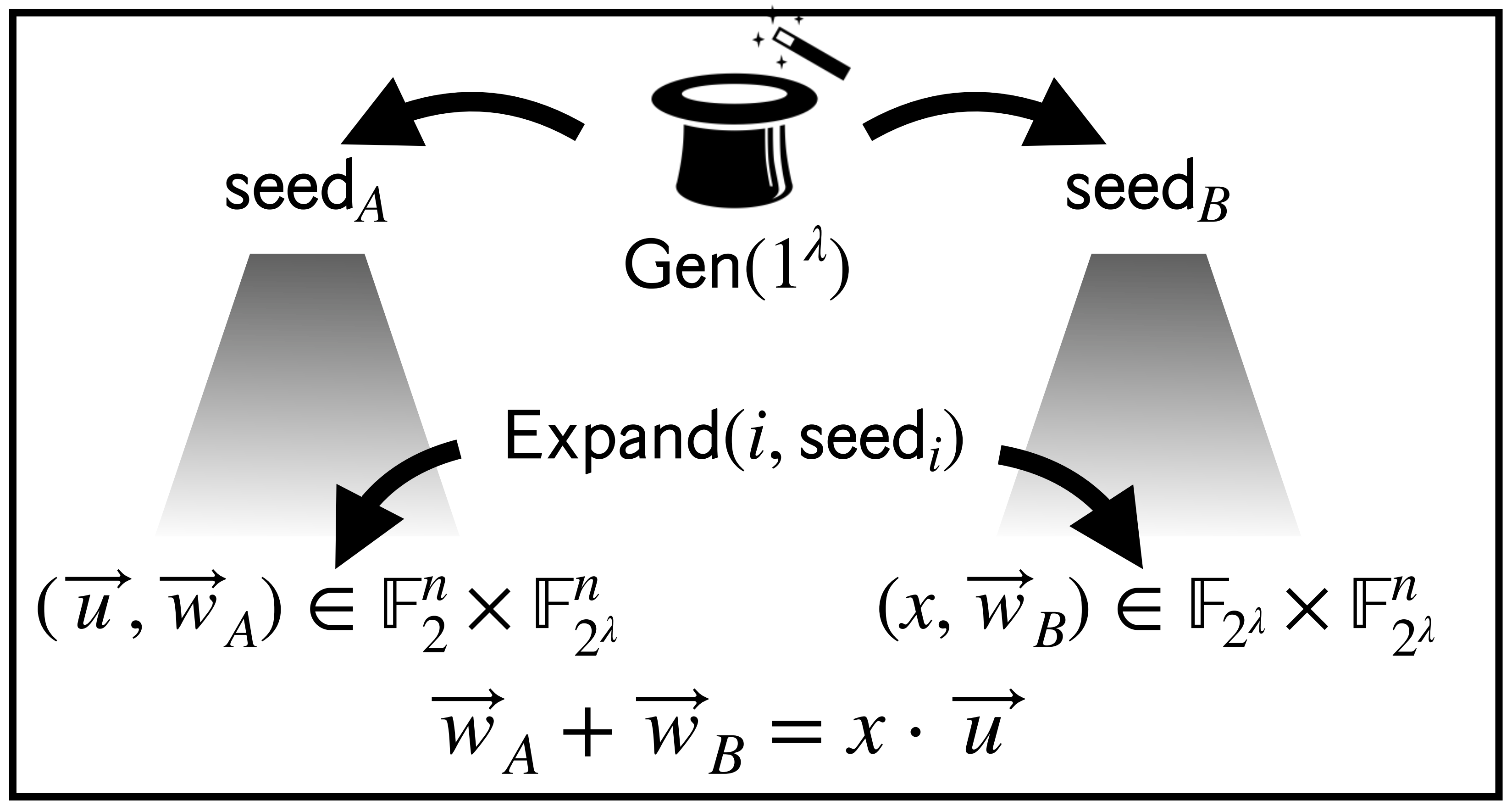
A construction from LPN

1. Reduction to subfield-VOLE

2. Constructing a PCG for subfield-VOLE

Three steps:

1 Construction for a random *unit vector* $\vec{u}$
from puncturable pseudorandom functions



seed_A
$\mathsf{Gen}(1^\lambda)$
seed_B
$\mathsf{Expand}(i, \mathsf{seed}_i)$
$(\vec{u}, \vec{w}_A) \in \mathbb{F}_2^n \times \mathbb{F}_{2^\lambda}^n$
$(x, \vec{w}_B) \in \mathbb{F}_{2^\lambda} \times \mathbb{F}_{2^\lambda}^n$
$\vec{w}_A + \vec{w}_B = x \cdot \vec{u}$

Pseudorandom Correlation Generators - Walkthrough

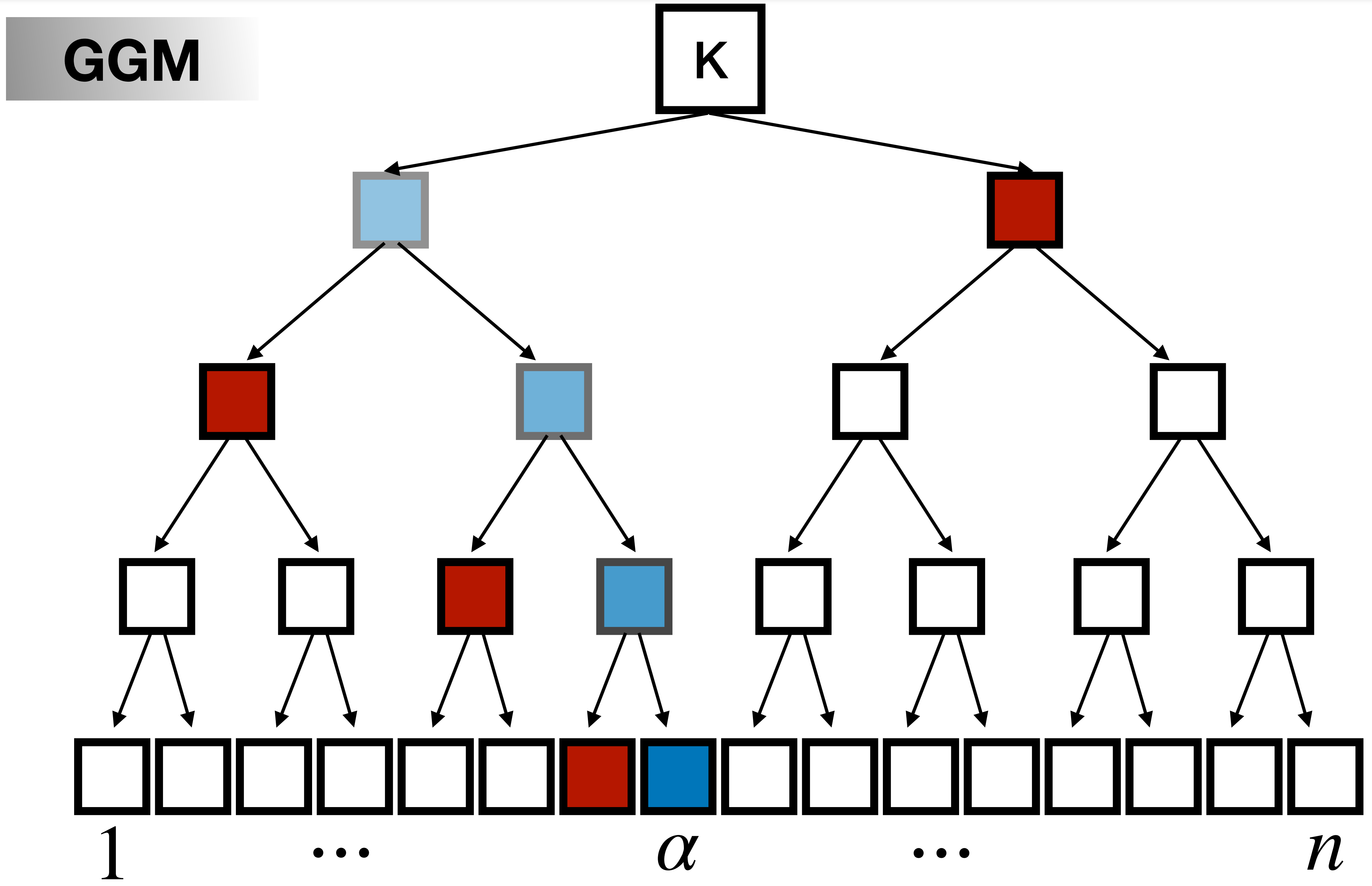
1 Construction for a random *unit vector* $\vec{u}$

from puncturable pseudorandom functions

$(\alpha : \; u_\alpha = 1)$

seed$_A = (x, K)$ seed$_B = (\vec{u},$ 🟥🟥🟥🟥, 🟦 $\oplus \, x)$

$\vec{w}_A \leftarrow \text{FullEval}(K)$ $\vec{w}_B \leftarrow \text{Insert}(x \oplus F_K(\alpha), \text{FullEval}(K_{\{\alpha\}}))$



A construction from LPN

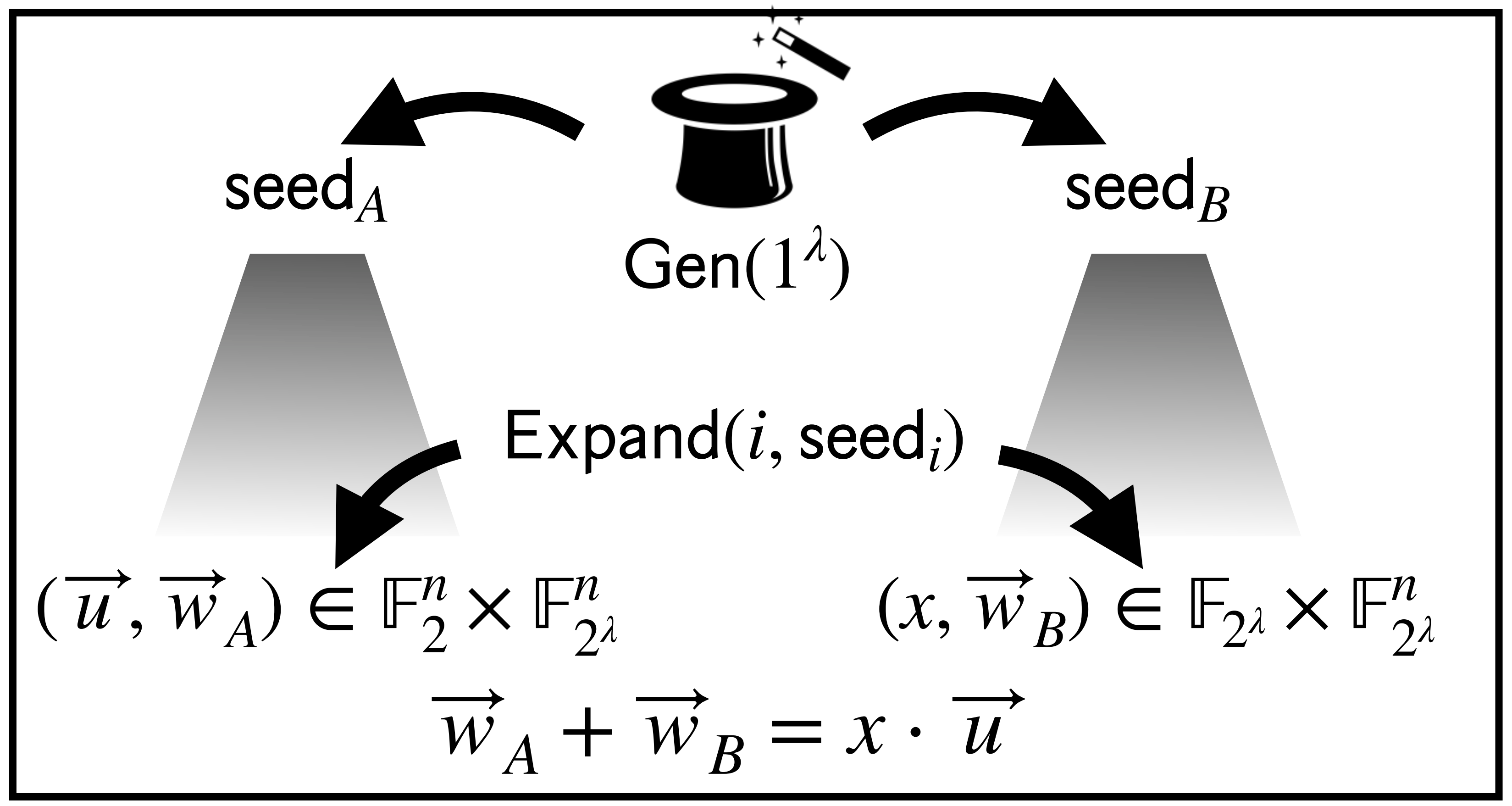
1. Reduction to subfield-VOLE

2. Constructing a PCG for subfield-VOLE

Three steps:

1 Construction for a random *unit vector* $\vec{u}$

from puncturable pseudorandom functions


seed$_A$
Gen(1^λ)
seed$_B$
Expand(i, seed_i)
$(\vec{u}, \vec{w}_A) \in \mathbb{F}_2^n \times \mathbb{F}_{2^\lambda}^n$
$(x, \vec{w}_B) \in \mathbb{F}_{2^\lambda} \times \mathbb{F}_{2^\lambda}^n$
$\vec{w}_A + \vec{w}_B = x \cdot \vec{u}$

Pseudorandom Correlation Generators - Walkthrough

2 Construction for a random *t-sparse vector* $\vec{u}$

via t parallel repetitions of (1)

$(\alpha : \; u_\alpha = 1)$

$$\begin{array}{r|l}
\text{seed}_A = (x, K^1) & \text{seed}_B = (\vec{u}_1, K^1_{\{\alpha_1\}}, F_{K^1}(\alpha_1) \oplus x) \\
K^2 & (\vec{u}_2, K^2_{\{\alpha_1\}}, F_{K^2}(\alpha_2) \oplus x) \\
\vdots & \vdots \\
K^t & (\vec{u}_t, K^t_{\{\alpha_t\}}, F_{K^t}(\alpha_t) \oplus x)
\end{array}$$

- Write $\vec{u}$ as a sum of t unit vectors $\vec{u}_1 \cdots \vec{u}_t$
- Apply the previous construction t times (with the same x)
- After expansion, the parties locally sum their shares:

$$\left(\bigoplus_{i=1}^{t} \vec{w}_A^i\right) \oplus \left(\bigoplus_{i=1}^{t} \vec{w}_B^i\right) = x \cdot \bigoplus_{i=1}^{t} \vec{u}_i = x \cdot \vec{u}$$

A construction from LPN

1. Reduction to subfield-VOLE

2. Constructing a PCG for subfield-VOLE

Three steps:

1 Construction for a random *unit vector* $\vec{u}$

from puncturable pseudorandom functions

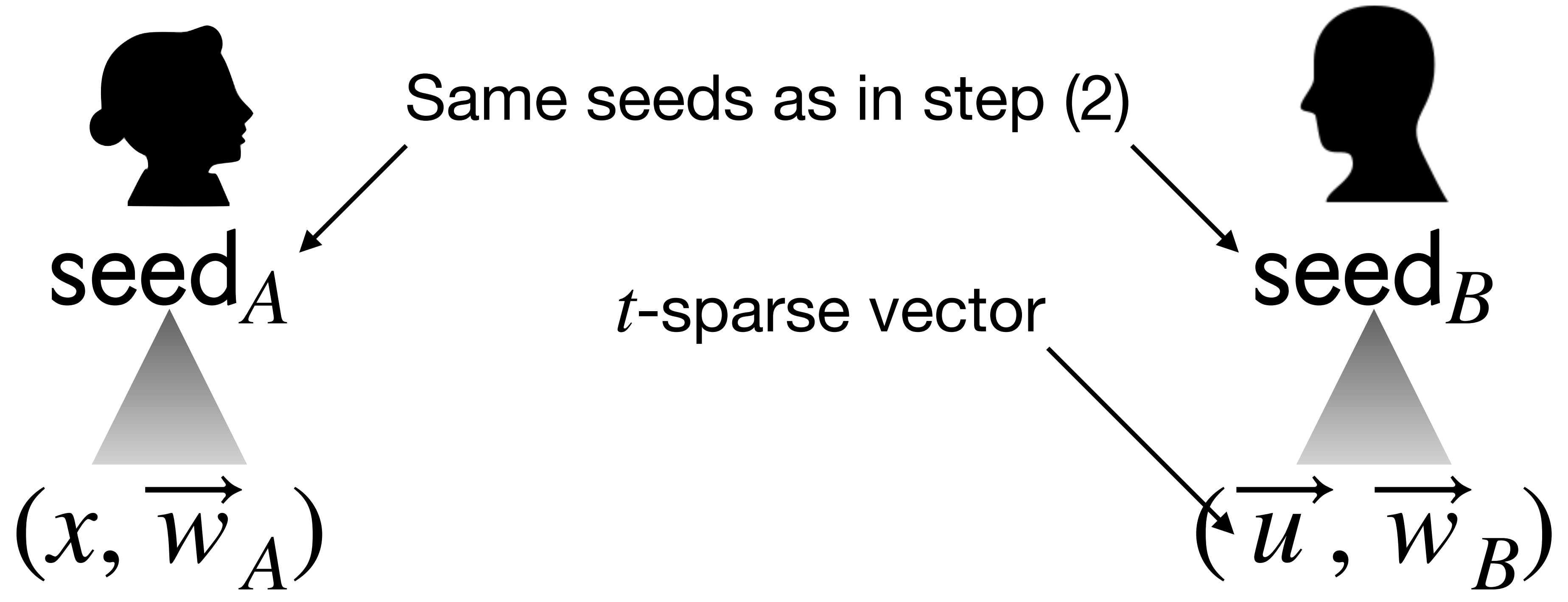
2 Construction for a random *t-sparse vector* $\vec{u}$

via t parallel repetitions of (1)

Pseudorandom Correlation Generators - Walkthrough

3 Construction for a pseudorandom vector $\overrightarrow{u}$

using dual-LPN


Same seeds as in step (2)
seed_A
t-sparse vector
seed_B
$(x, \overrightarrow{w}_A)$
$(\overrightarrow{u}, \overrightarrow{w}_B)$

The LPN assumption - primal

A construction from LPN

1. Reduction to subfield-VOLE

2. Constructing a PCG for subfield-VOLE

Three steps:

1 Construction for a random *unit vector* $\overrightarrow{u}$

from puncturable pseudorandom functions

2 Construction for a random t*-sparse vector* $\overrightarrow{u}$

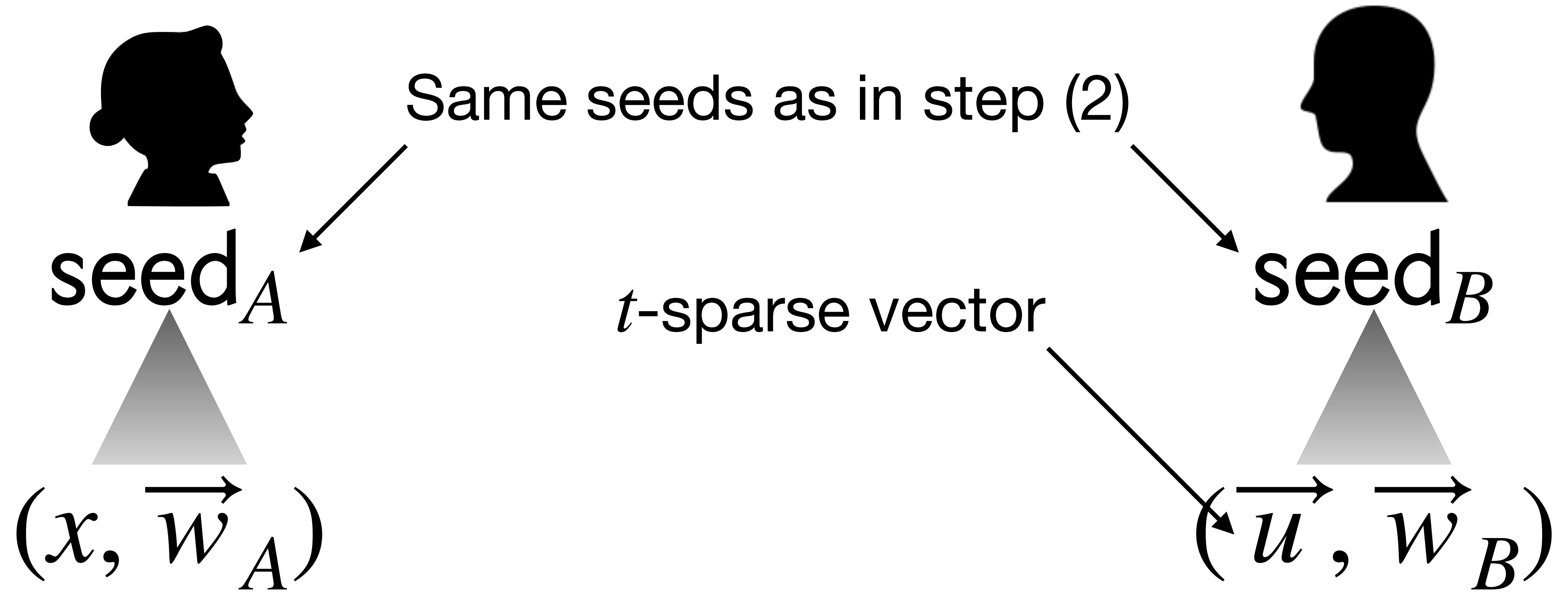
via t parallel repetitions of (1)

3 Construction for a pseudorandom vector $\overrightarrow{u}$

using dual-LPN

Pseudorandom Correlation Generators - Walkthrough

3 Construction for a pseudorandom vector $\vec{u}$
using dual-LPN



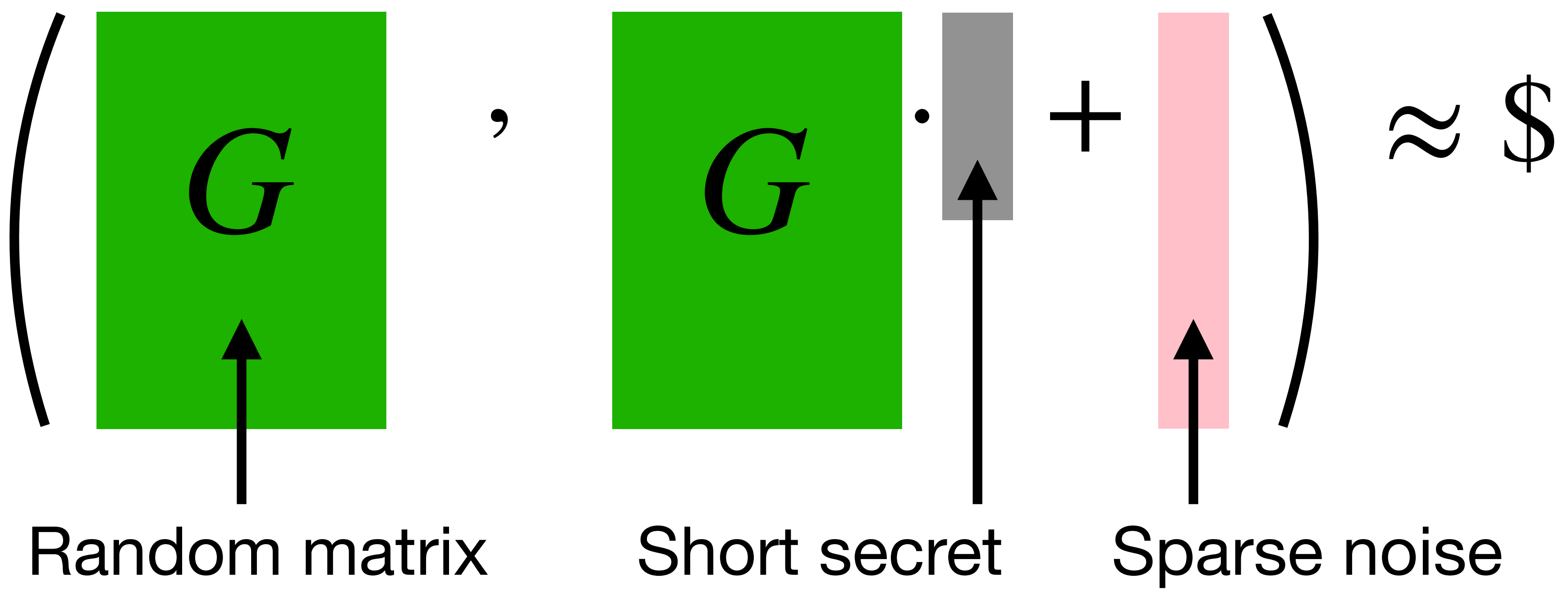
Same seeds as in step (2)

seed_A seed_B

t-sparse vector

$(x, \vec{w}_A)$ $(\vec{u}, \vec{w}_B)$

The LPN assumption - primal



$$\left(G \;,\; G \cdot \vec{s} + \vec{e} \right) \approx \$$$

Random matrix

Short secret

Sparse noise

A construction from LPN

1. Reduction to subfield-VOLE

2. Constructing a PCG for subfield-VOLE

Three steps:

1 Construction for a random *unit vector* $\vec{u}$
from puncturable pseudorandom functions

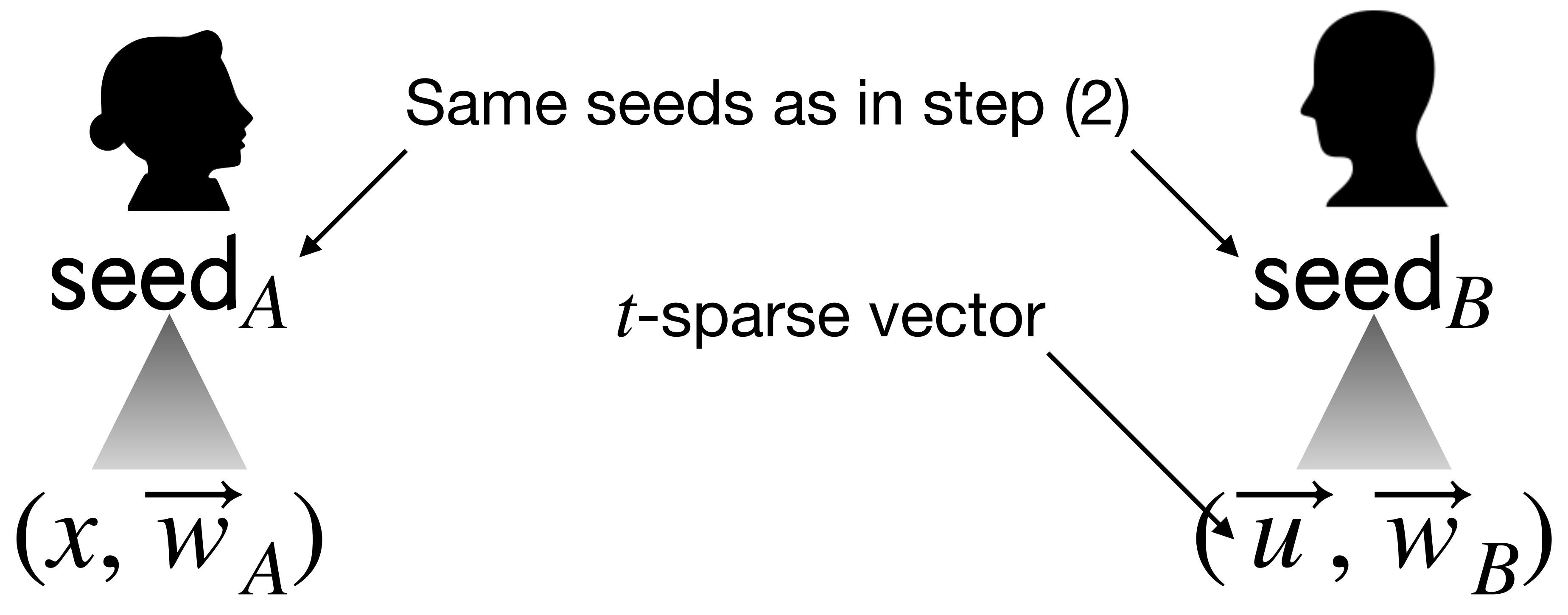
2 Construction for a random t*-sparse vector* $\vec{u}$
via t parallel repetitions of (1)

3 Construction for a pseudorandom vector $\vec{u}$
using dual-LPN

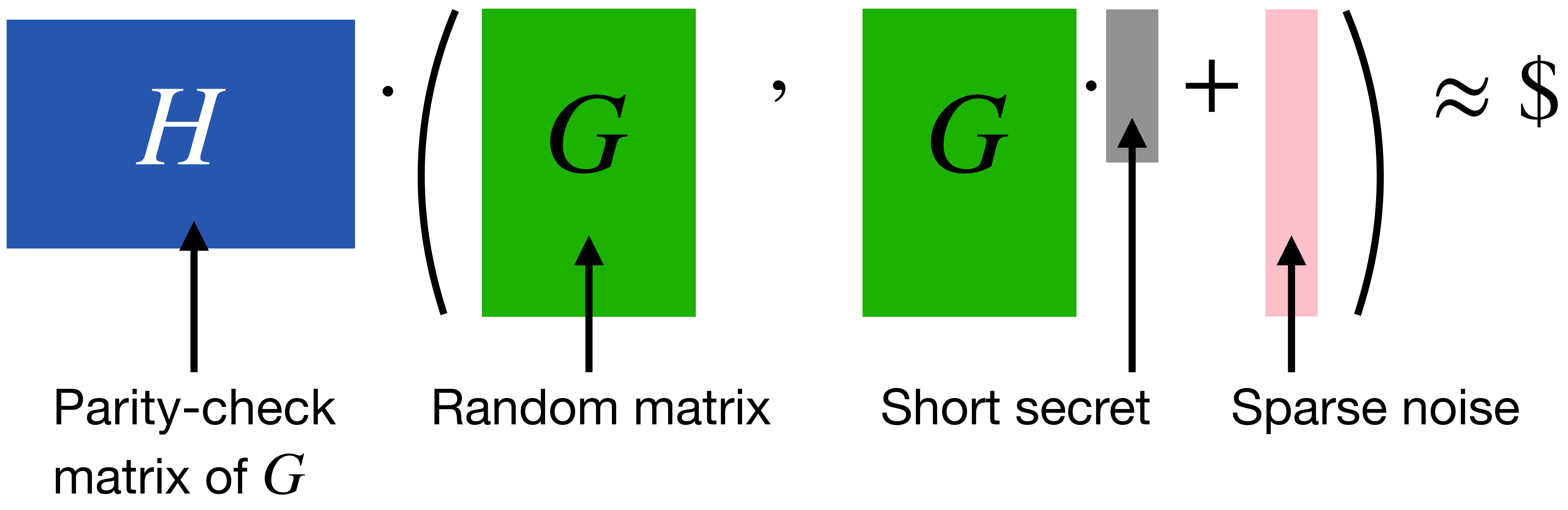
Pseudorandom Correlation Generators - Walkthrough

3 Construction for a pseudorandom vector $\vec{u}$

using dual-LPN



The LPN assumption - primal



$H \cdot \left(G \;,\; G \cdot \;+\; \right) \approx \$$
Parity-check matrix of G
Random matrix
Short secret
Sparse noise

A construction from LPN

1. Reduction to subfield-VOLE

2. Constructing a PCG for subfield-VOLE

Three steps:

1 Construction for a random *unit vector* $\vec{u}$

from puncturable pseudorandom functions

2 Construction for a random *t-sparse vector* $\vec{u}$

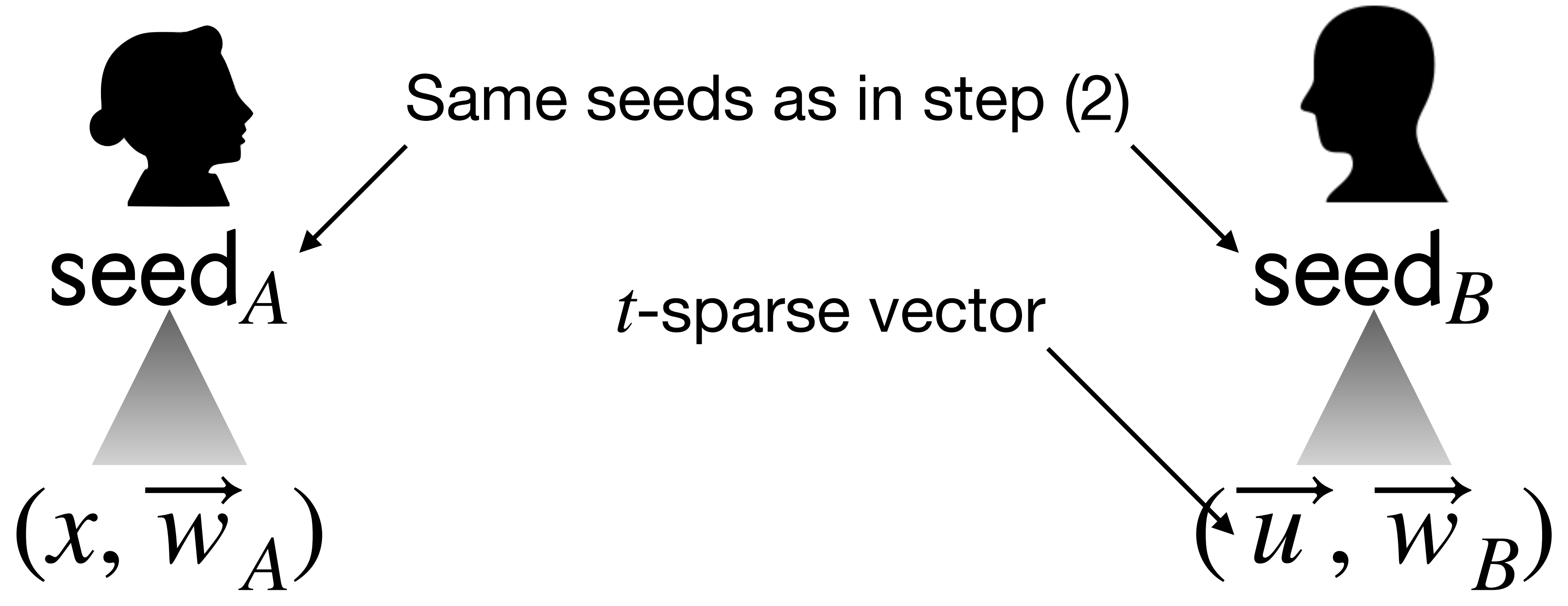
via t parallel repetitions of (1)

3 Construction for a pseudorandom vector $\vec{u}$

using dual-LPN

Pseudorandom Correlation Generators - Walkthrough

3 Construction for a pseudorandom vector $\vec{u}$
using dual-LPN



Same seeds as in step (2)

seed_A

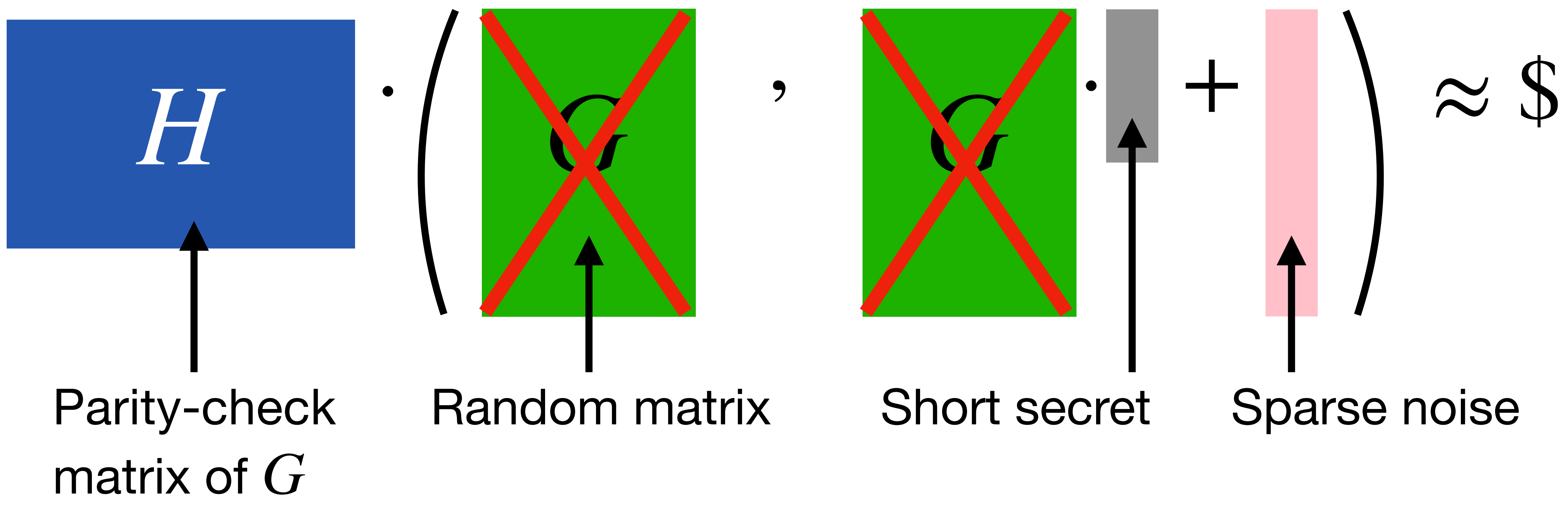
seed_B

t-sparse vector

$(x, \vec{w}_A)$

$(\vec{u}, \vec{w}_B)$

The LPN assumption - primal



$$H \cdot \left(\cancel{G}, \cancel{G} \cdot \vec{s} + \vec{e} \right) \approx \$$$

A construction from LPN

1. Reduction to subfield-VOLE

2. Constructing a PCG for subfield-VOLE

Three steps:

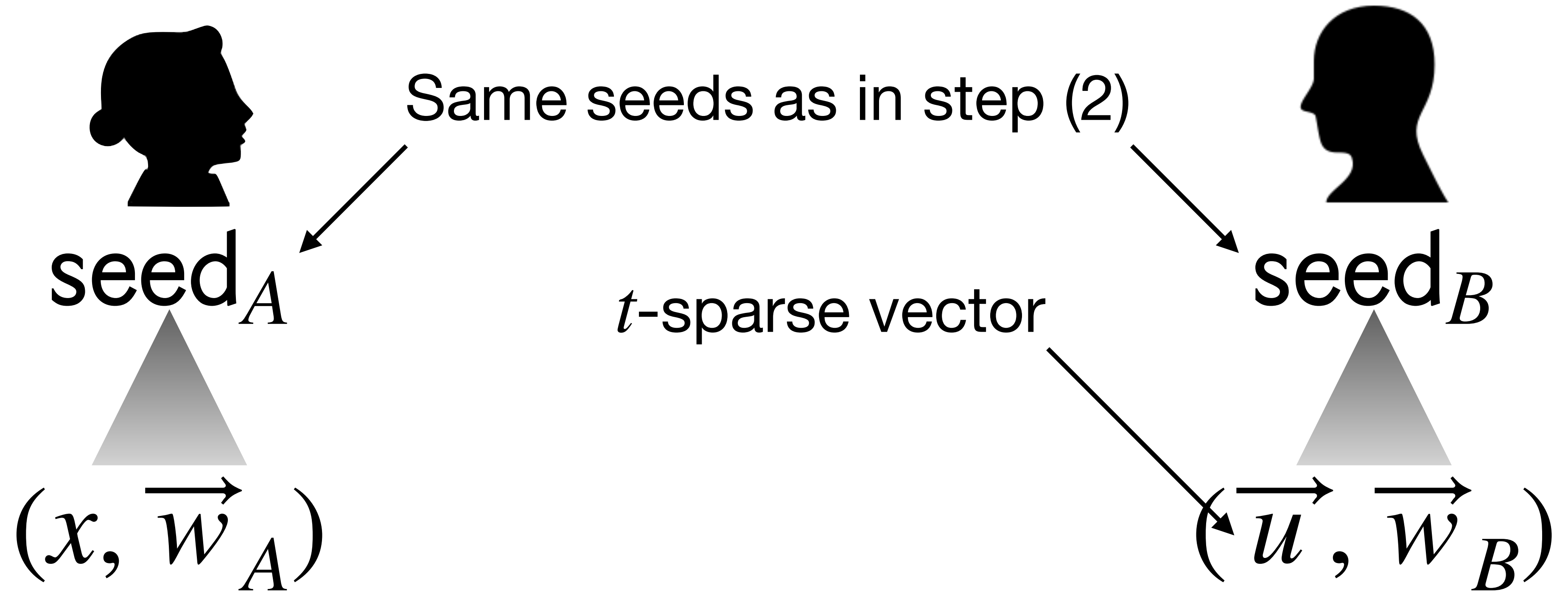
1 Construction for a random *unit vector* $\vec{u}$
from puncturable pseudorandom functions

2 Construction for a random *t-sparse vector* $\vec{u}$
via t parallel repetitions of (1)

3 Construction for a pseudorandom vector $\vec{u}$
using dual-LPN

Pseudorandom Correlation Generators - Walkthrough

3 Construction for a pseudorandom vector $\vec{u}$
using dual-LPN



Same seeds as in step (2)

seed_A

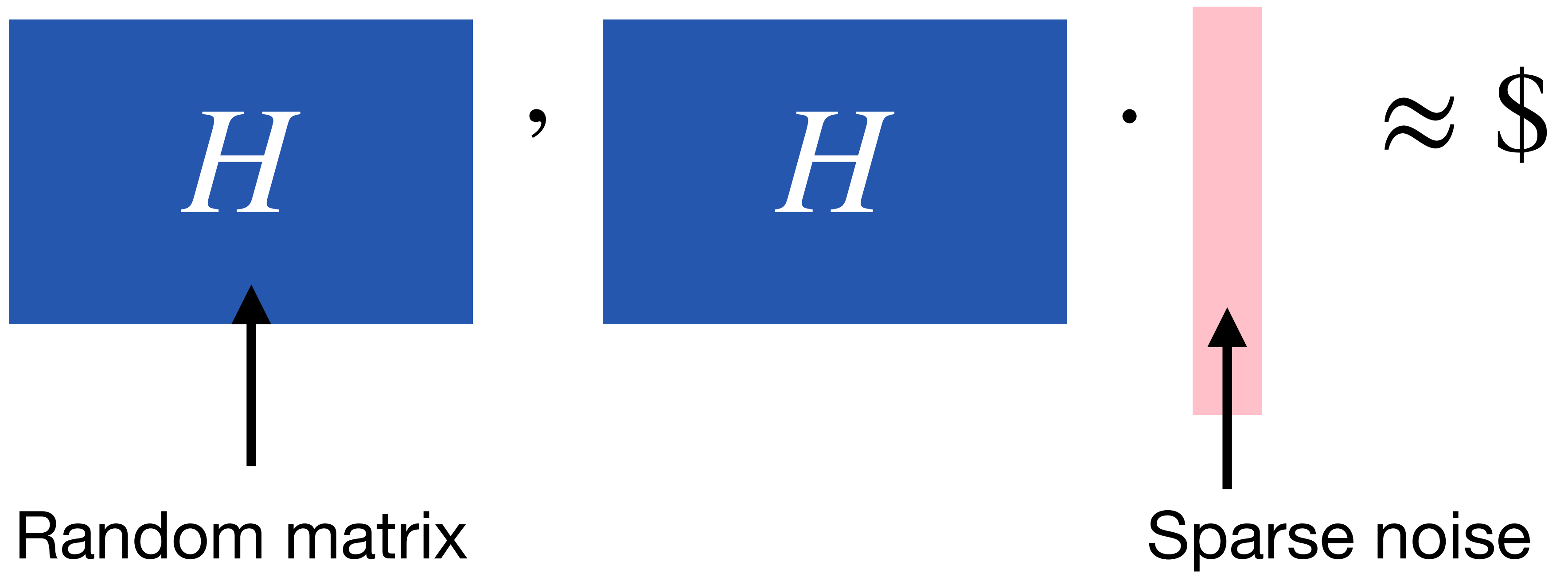
seed_B

t-sparse vector

$(x, \vec{w}_A)$

$(\vec{u}, \vec{w}_B)$

The LPN assumption - dual



$H \;,\; H \;\cdot \quad \approx \$$

Random matrix

Sparse noise

A construction from LPN

1. Reduction to subfield-VOLE

2. Constructing a PCG for subfield-VOLE

Three steps:

1 Construction for a random *unit vector* $\vec{u}$
from puncturable pseudorandom functions

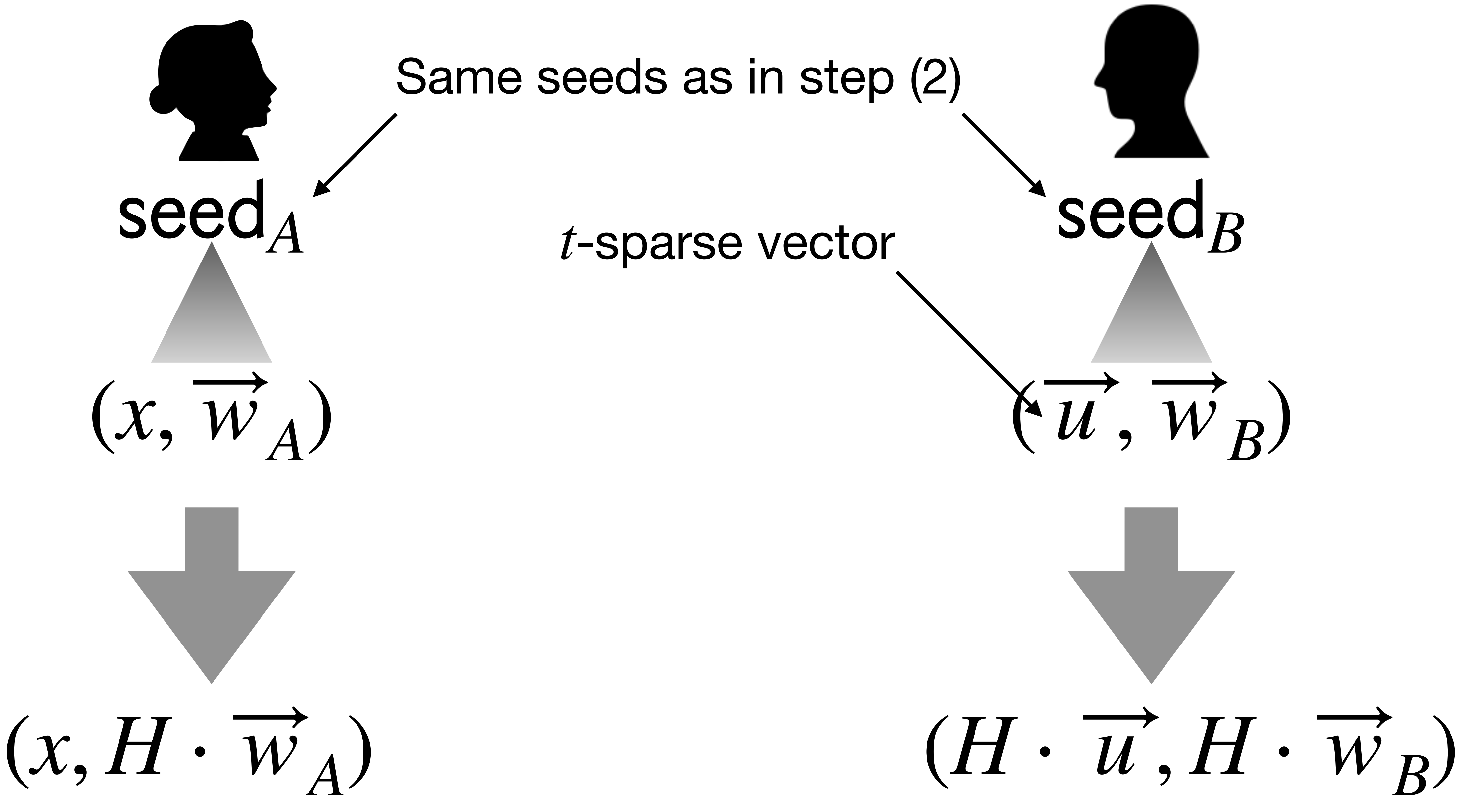
2 Construction for a random *t-sparse vector* $\vec{u}$
via t parallel repetitions of (1)

3 Construction for a pseudorandom vector $\vec{u}$
using dual-LPN

Pseudorandom Correlation Generators - Walkthrough

3 Construction for a pseudorandom vector $\vec{u}$

using dual-LPN



$$\boxed{H \cdot \vec{w}_A + H \cdot \vec{w}_B = H \cdot (x \cdot \vec{u}) = x \cdot (H \cdot \vec{u})}$$

Pseudorandom under the LPN assumption

A construction from LPN

1. Reduction to subfield-VOLE

2. Constructing a PCG for subfield-VOLE

Three steps:

1 Construction for a random *unit vector* $\vec{u}$
from puncturable pseudorandom functions

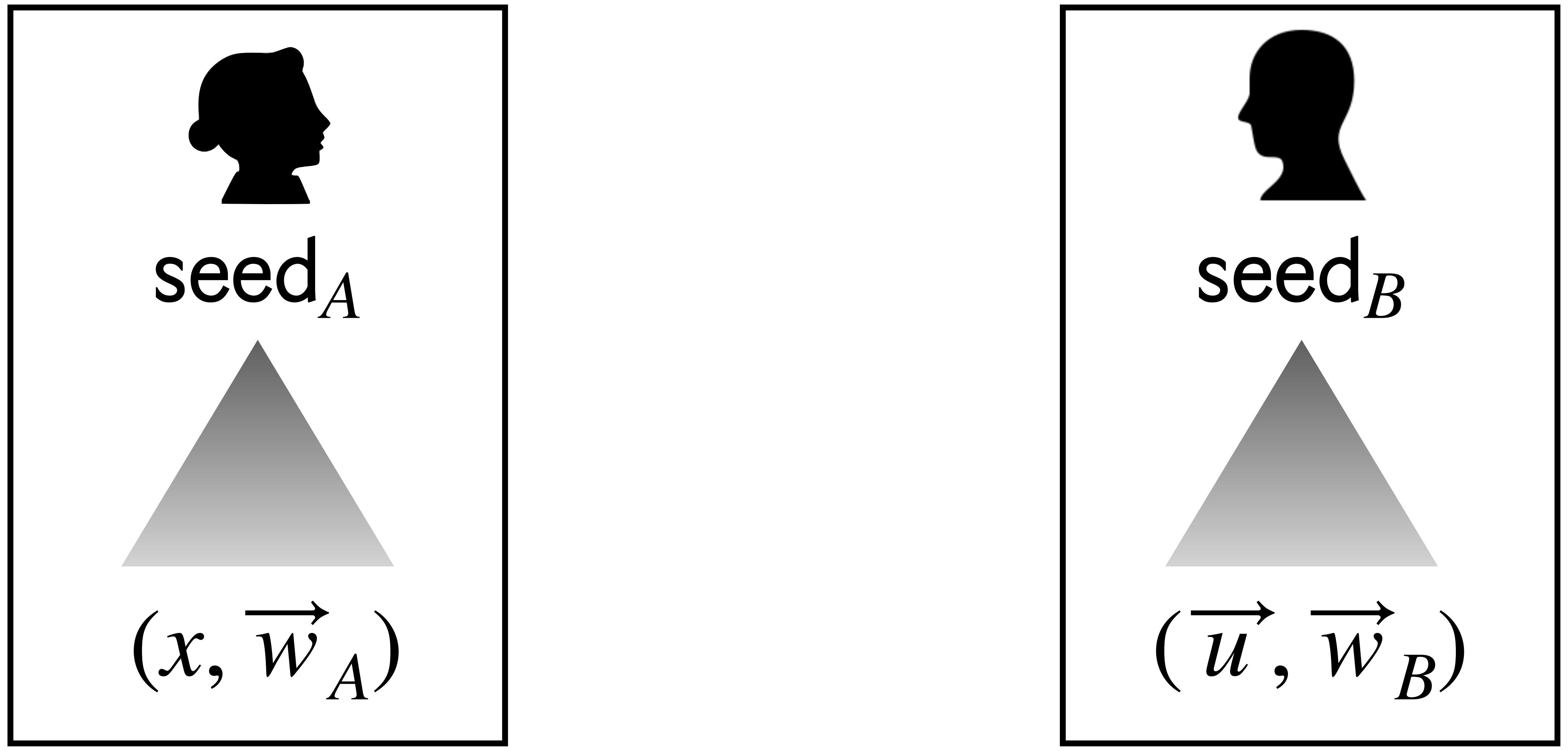
2 Construction for a random *t-sparse vector* $\vec{u}$
via t parallel repetitions of (1)

3 Construction for a pseudorandom vector $\vec{u}$
using dual-LPN

Pseudorandom Correlation *Functions*

Wrapping-up

$$\vec{w}_A + \vec{w}_B = x \cdot \vec{u}$$



seed$_A$

$(x, \vec{w}_A)$

seed$_B$

$(\vec{u}, \vec{w}_B)$

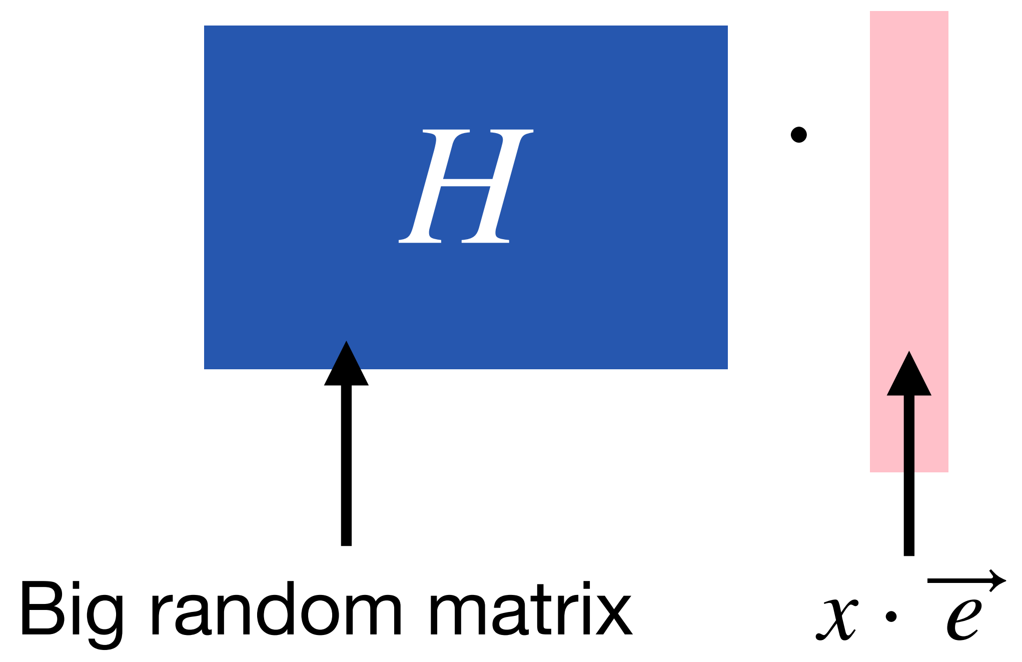
$|\text{seed}_A| \approx \lambda \cdot t$

$|\text{seed}_B| \approx \lambda \cdot t \cdot \log n$

- λ is a security parameter, t is an LPN noise parameter, n is the vector length.
- Converted to n pseudorandom OTs via a correlation-robust hash function.

Can we turn this into a PC*F*?

The expansion of the PCG boils down to the computation of



Where $\vec{e}$ is a very sparse vector, and (the shares of) the entries of $x \cdot \vec{e}$ can be computed individually in log-time.

Intuitively, to get a PCF, we want to

- make H exponentially big, and
- compute each $H_i \cdot \langle x \cdot \vec{e} \rangle$ in time polylog(dim(H))

Idea:

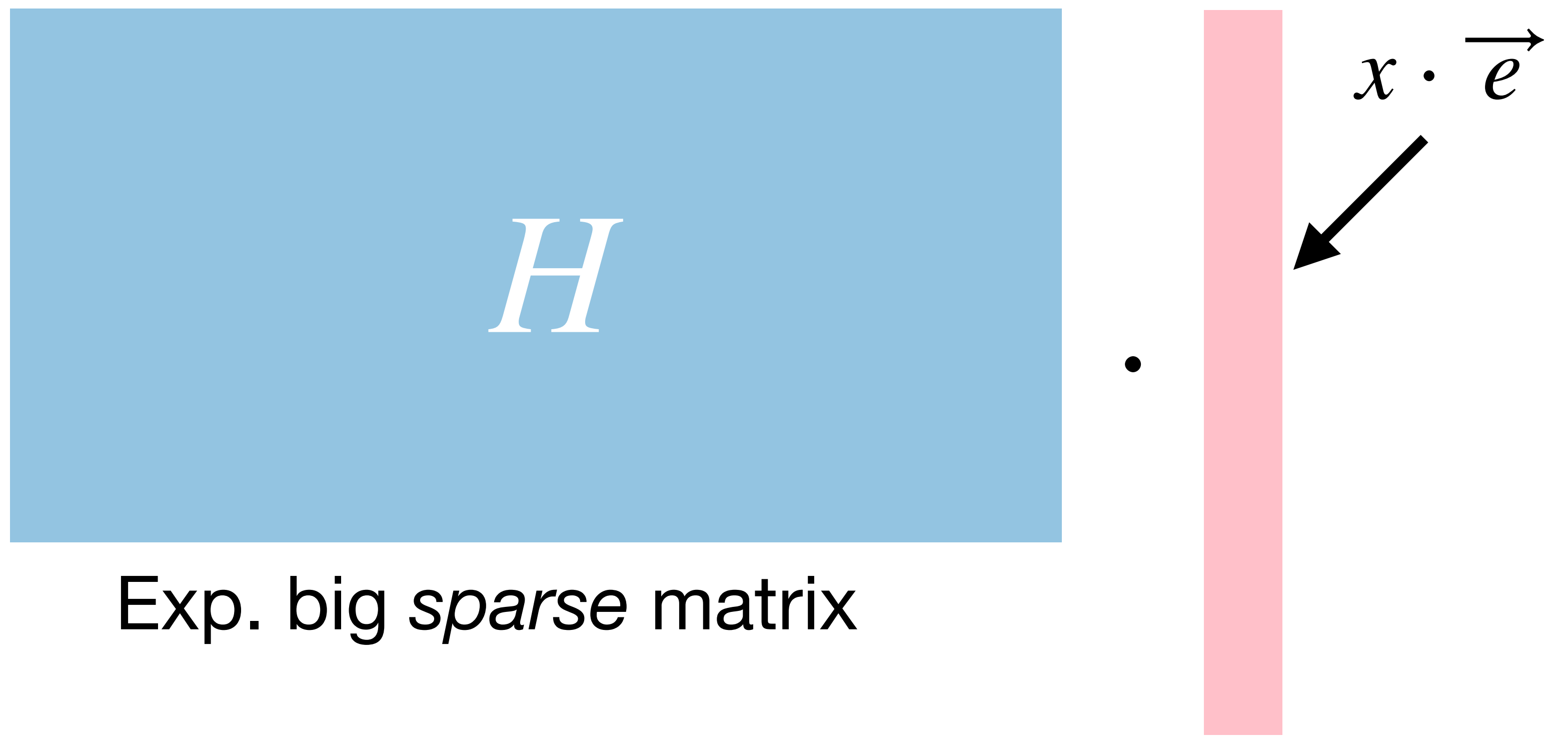
Make H exponentially sparse?

Pseudorandom Correlation *Functions*

If H is exponentially large and exponentially sparse…
Then $H \cdot \overrightarrow{e}$ is sparse, hence not pseudorandom

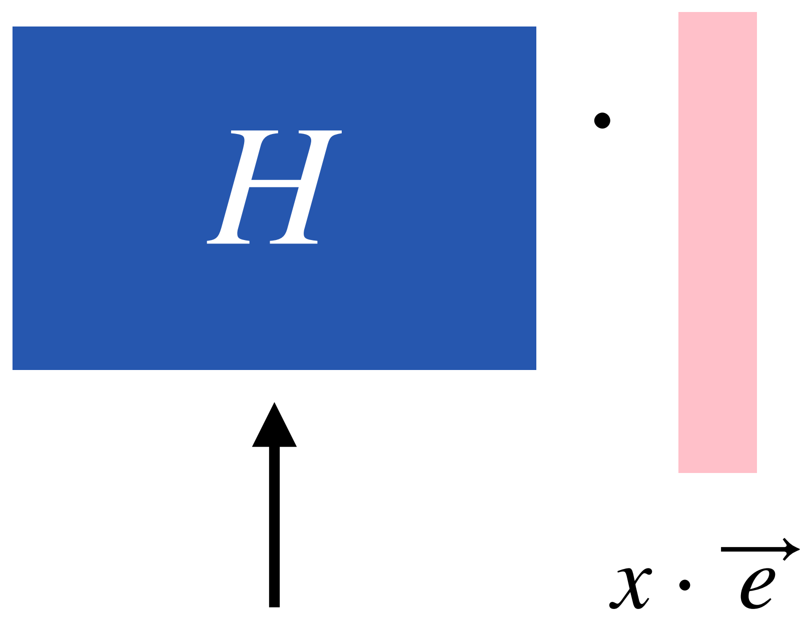
If H is dense enough such that $H \cdot \overrightarrow{e}$ is not sparse…
Then H is necessarily ‘small’

☹



Exp. big *sparse* matrix

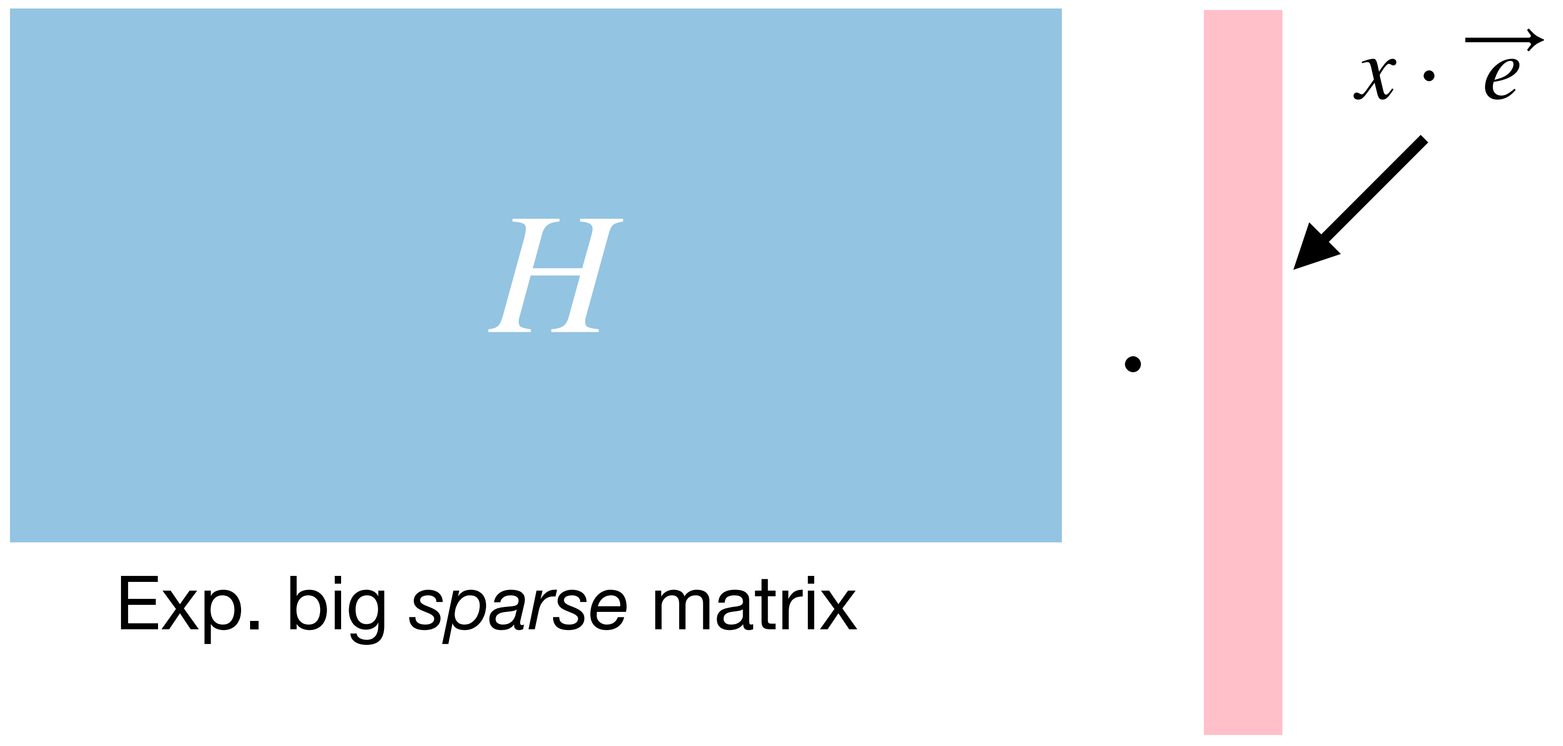
☹



Denser, smaller matrix

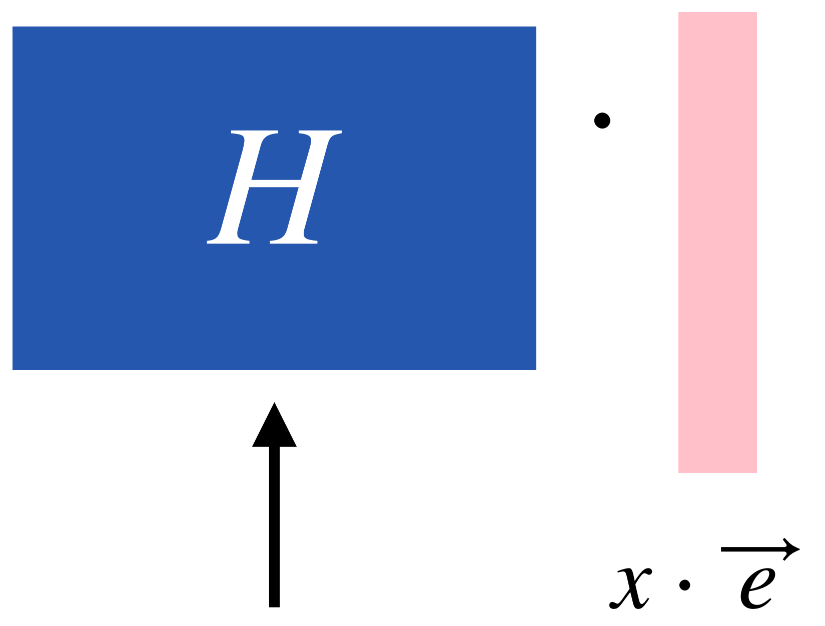
Pseudorandom Correlation *Functions*

If H is exponentially large and exponentially sparse…
Then $H \cdot \vec{e}$ is sparse, hence not pseudorandom



Exp. big *sparse* matrix

If H is dense enough such that $H \cdot \vec{e}$ is not sparse…
Then H is necessarily 'small'



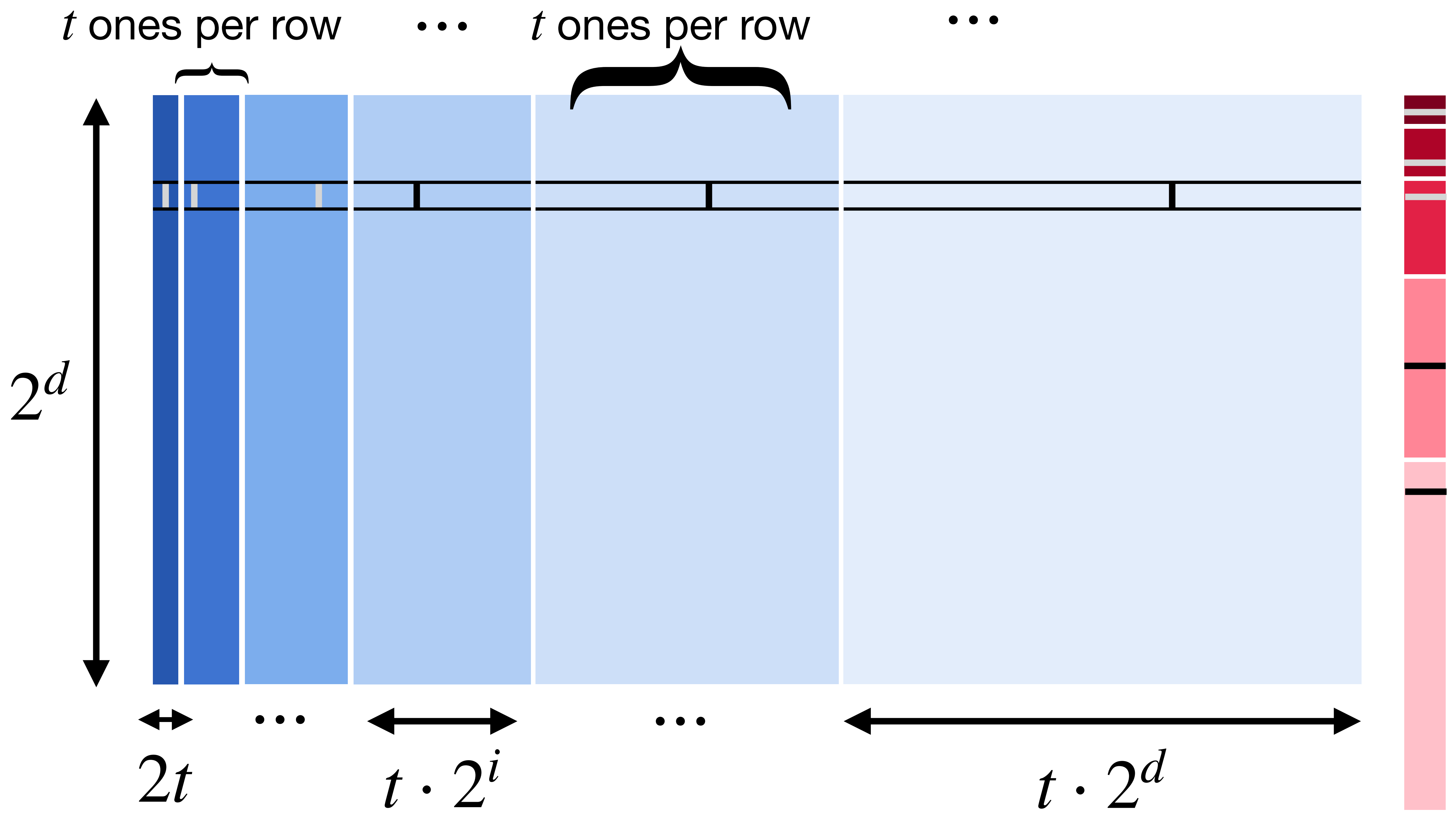
Denser, smaller matrix

Having our cake and eating it too?

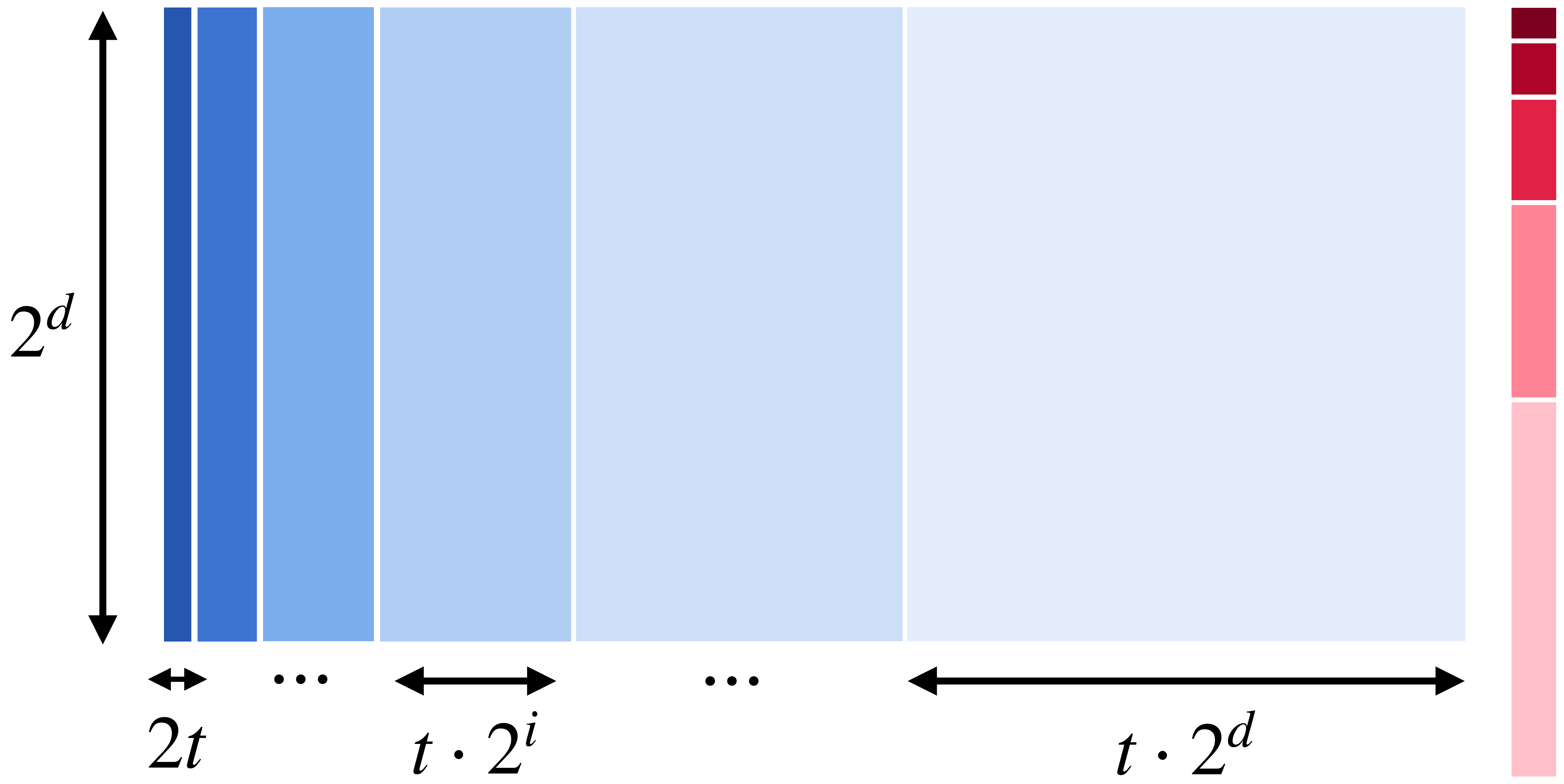
What if we make H and $\vec{e}$ exponentially large, with *variable density?*

Description:

- A row of H has d blocks, each block has t sub-blocks of size 2^i with a single random 1.
- $\vec{e}$ is distributed as a row of H.
- We allow up to 2^d rows; think: $d \approx t \approx \lambda$



Low-Complexity WPRF from VDLPN

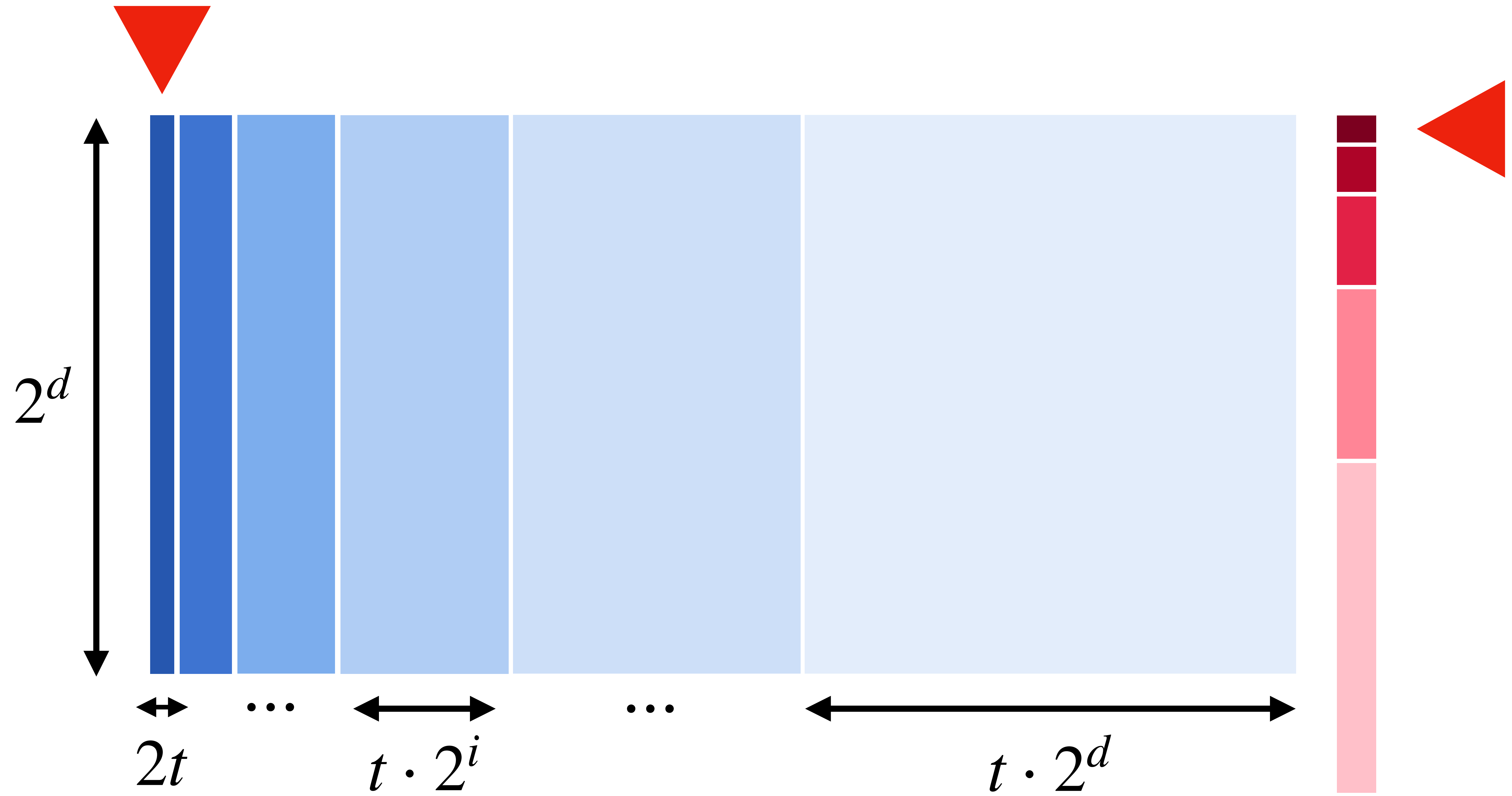


$\approx \$$

Equation:

$$\bigoplus_{i=1}^{d}\bigoplus_{j=1}^{t}\left\langle \vec{h}_{i,j}, \vec{e}_{i,j}\right\rangle = \bigoplus_{i=1}^{d}\bigoplus_{j=1}^{t}\mathrm{EQ}(x_{i,j}, \bar{K}_{i,j}) = \bigoplus_{i=1}^{d}\bigoplus_{j=1}^{t}\bigwedge_{\ell=1}^{i}\left(x_{i,j} \oplus K_{i,j}\right)$$

Low-Complexity WPRF from VDLPN

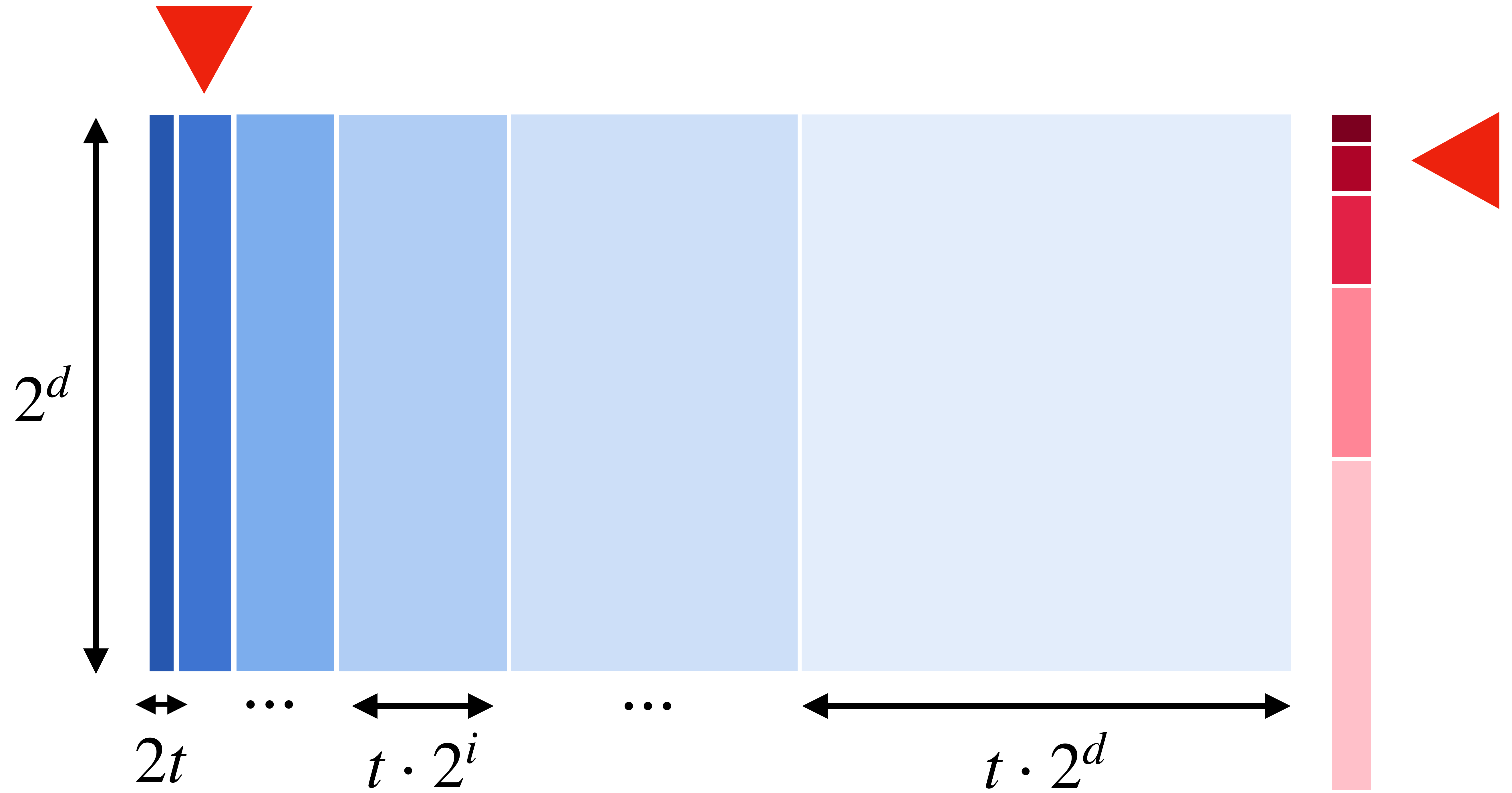


$\approx \$$

Equation:

$$\bigoplus_{i=1}^{d} \bigoplus_{j=1}^{t} \left\langle \vec{h}_{i,j}, \vec{e}_{i,j} \right\rangle = \bigoplus_{i=1}^{d} \bigoplus_{j=1}^{t} \mathrm{EQ}(x_{i,j}, \bar{K}_{i,j}) = \bigoplus_{i=1}^{d} \bigoplus_{j=1}^{t} \bigwedge_{\ell=1}^{i} \left(x_{i,j} \oplus K_{i,j} \right)$$

Low-Complexity WPRF from VDLPN

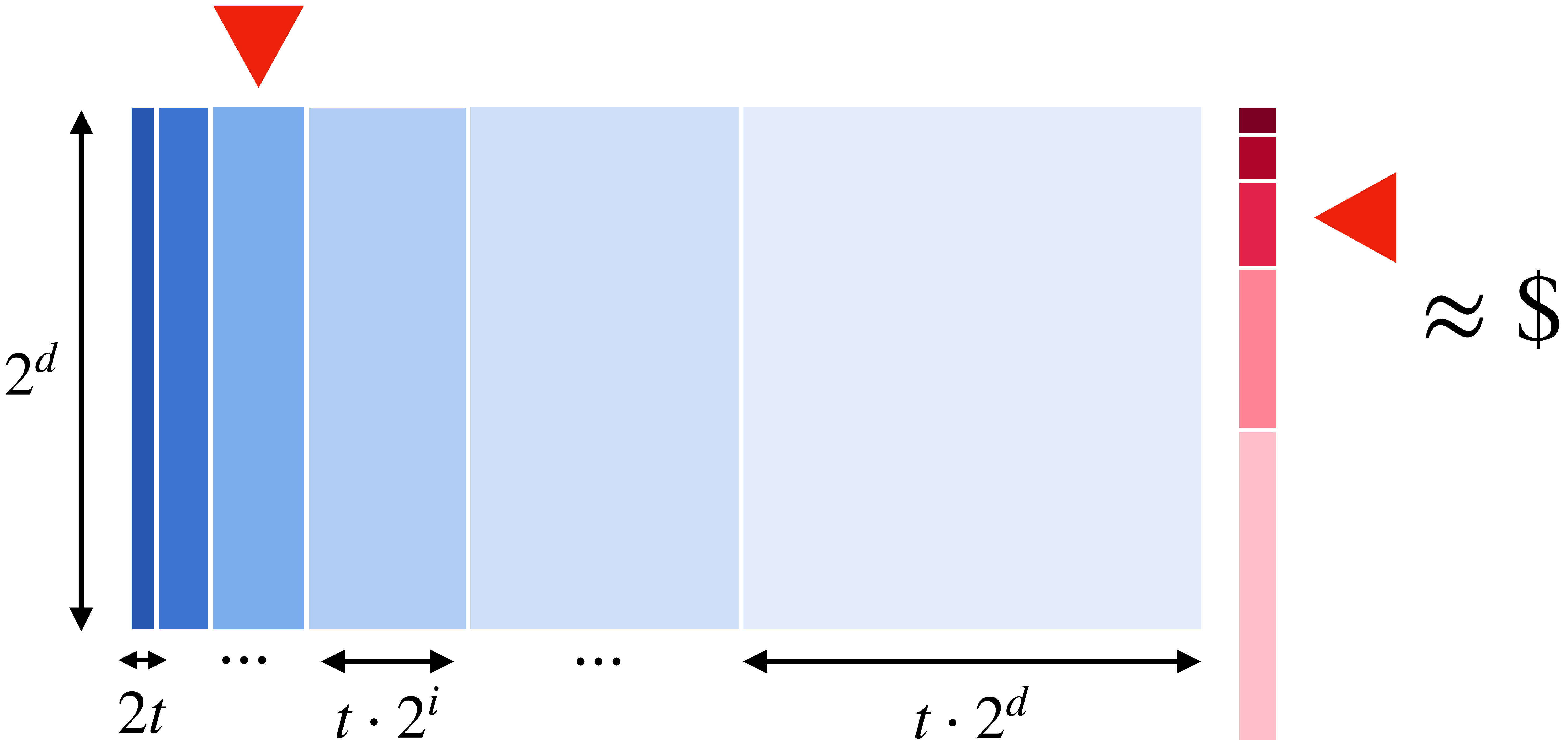


$\approx \$$

Equation:

$$\bigoplus_{i=1}^{d} \bigoplus_{j=1}^{t} \left\langle \overrightarrow{h}_{i,j}, \overrightarrow{e}_{i,j} \right\rangle = \bigoplus_{i=1}^{d} \bigoplus_{j=1}^{t} \mathsf{EQ}(x_{i,j}, \bar{K}_{i,j}) = \bigoplus_{i=1}^{d} \bigoplus_{j=1}^{t} \bigwedge_{\ell=1}^{i} \left(x_{i,j} \oplus K_{i,j} \right)$$

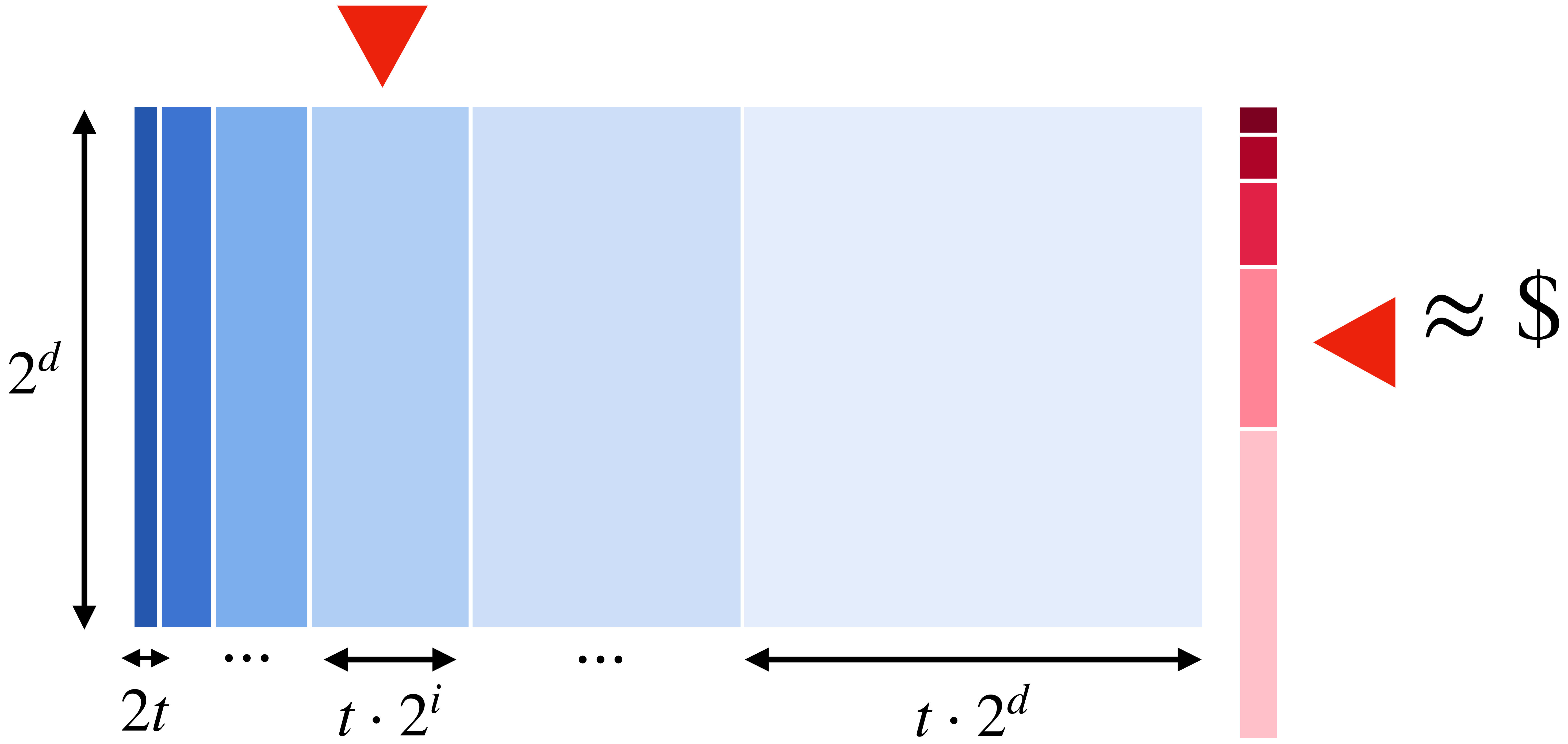
Low-Complexity WPRF from VDLPN



Equation:

$$\bigoplus_{i=1}^{d}\bigoplus_{j=1}^{t}\left\langle \vec{h}_{i,j}, \vec{e}_{i,j}\right\rangle = \bigoplus_{i=1}^{d}\bigoplus_{j=1}^{t}\mathrm{EQ}(x_{i,j}, \bar{K}_{i,j}) = \bigoplus_{i=1}^{d}\bigoplus_{j=1}^{t}\bigwedge_{\ell=1}^{i}\left(x_{i,j}\oplus K_{i,j}\right)$$

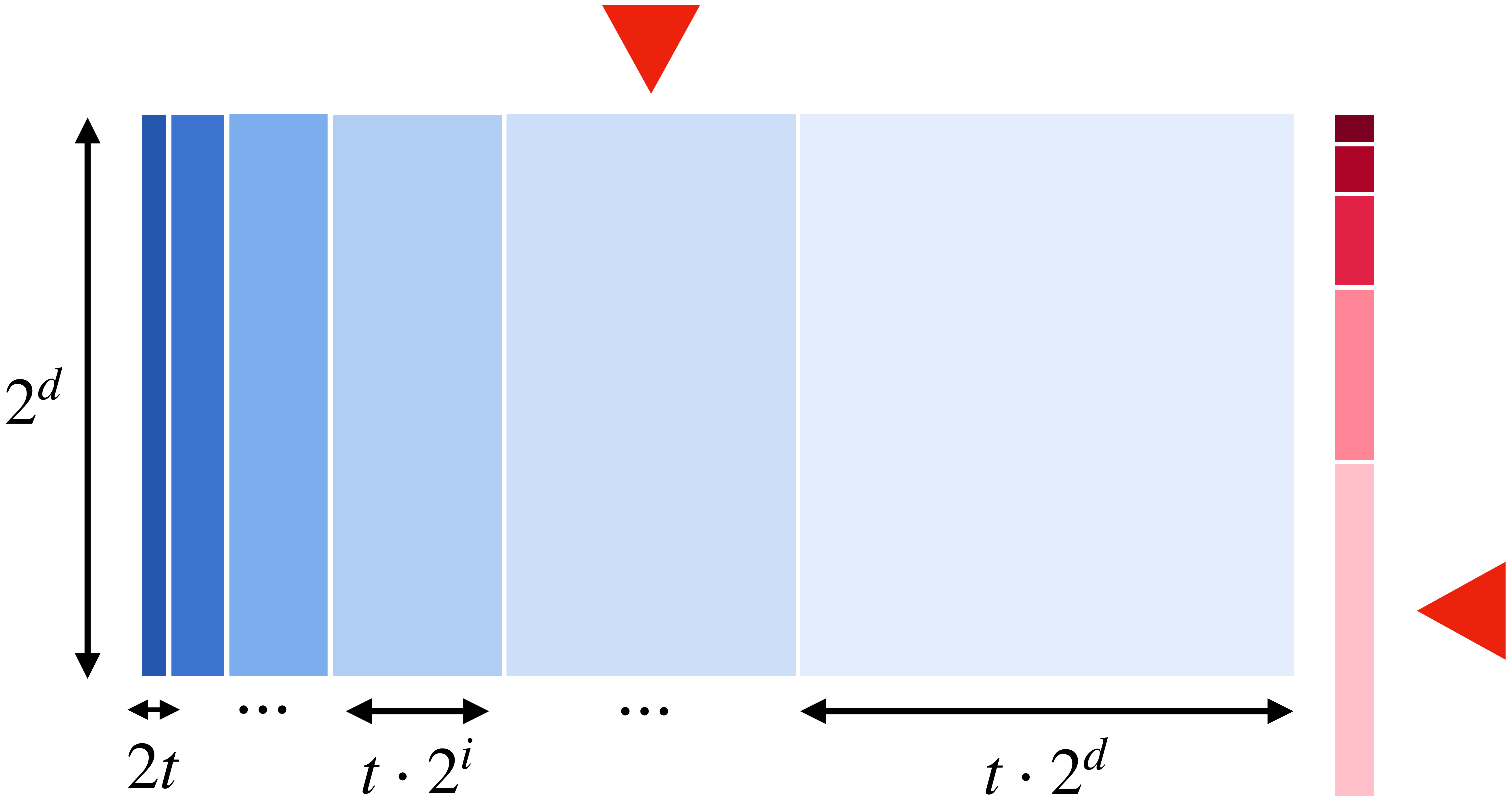
Low-Complexity WPRF from VDLPN



Equation:

$$\bigoplus_{i=1}^{d}\bigoplus_{j=1}^{t}\left\langle \vec{h}_{i,j}, \vec{e}_{i,j}\right\rangle = \bigoplus_{i=1}^{d}\bigoplus_{j=1}^{t}\mathrm{EQ}(x_{i,j}, \bar{K}_{i,j}) = \bigoplus_{i=1}^{d}\bigoplus_{j=1}^{t}\bigwedge_{\ell=1}^{i}\left(x_{i,j} \oplus K_{i,j}\right)$$

Low-Complexity WPRF from VDLPN

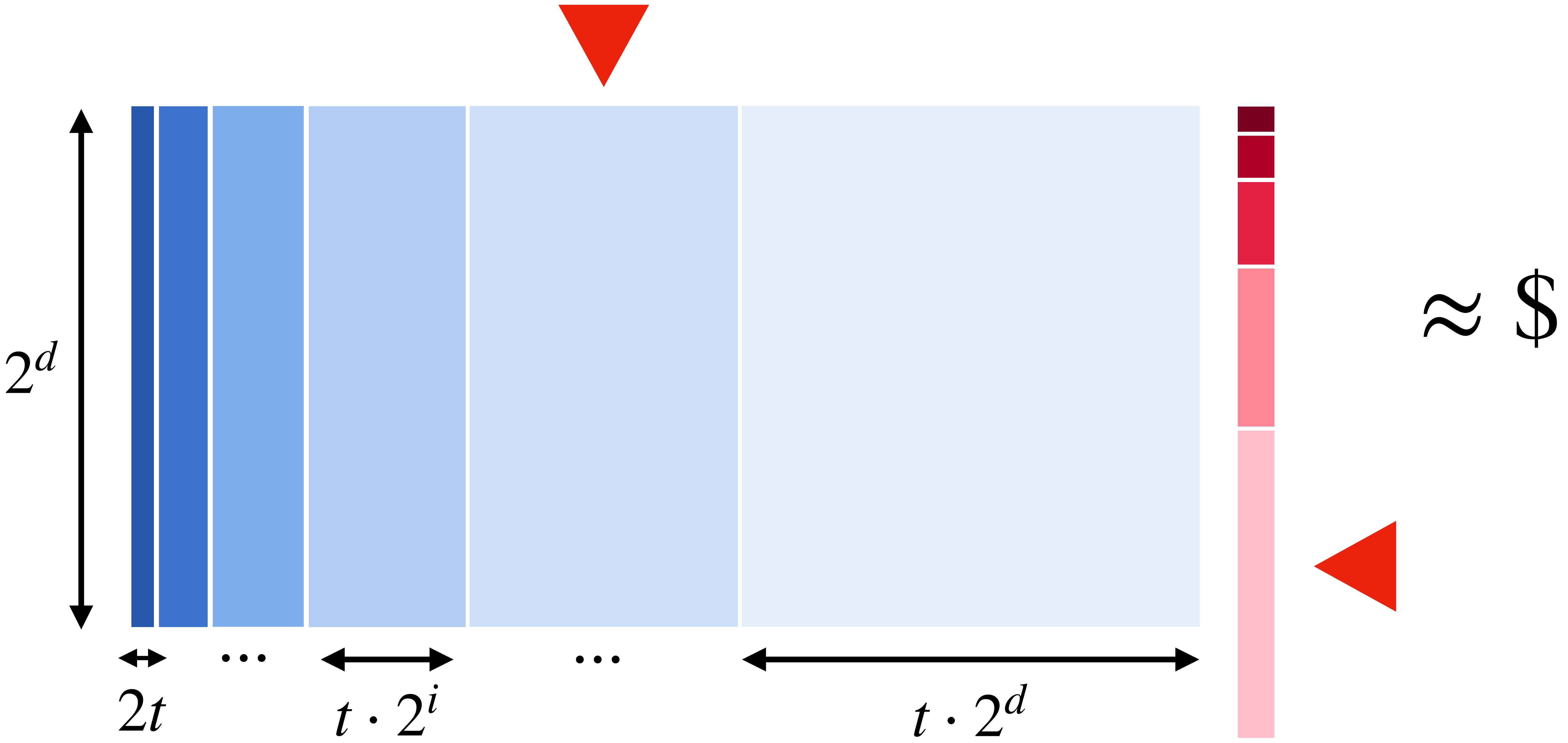


$\approx \$$

Equation:

$$\bigoplus_{i=1}^{d}\bigoplus_{j=1}^{t}\left\langle \vec{h}_{i,j}, \vec{e}_{i,j}\right\rangle = \bigoplus_{i=1}^{d}\bigoplus_{j=1}^{t} \mathrm{EQ}(x_{i,j}, \bar{K}_{i,j}) = \bigoplus_{i=1}^{d}\bigoplus_{j=1}^{t}\bigwedge_{\ell=1}^{i}\left(x_{i,j} \oplus K_{i,j}\right)$$

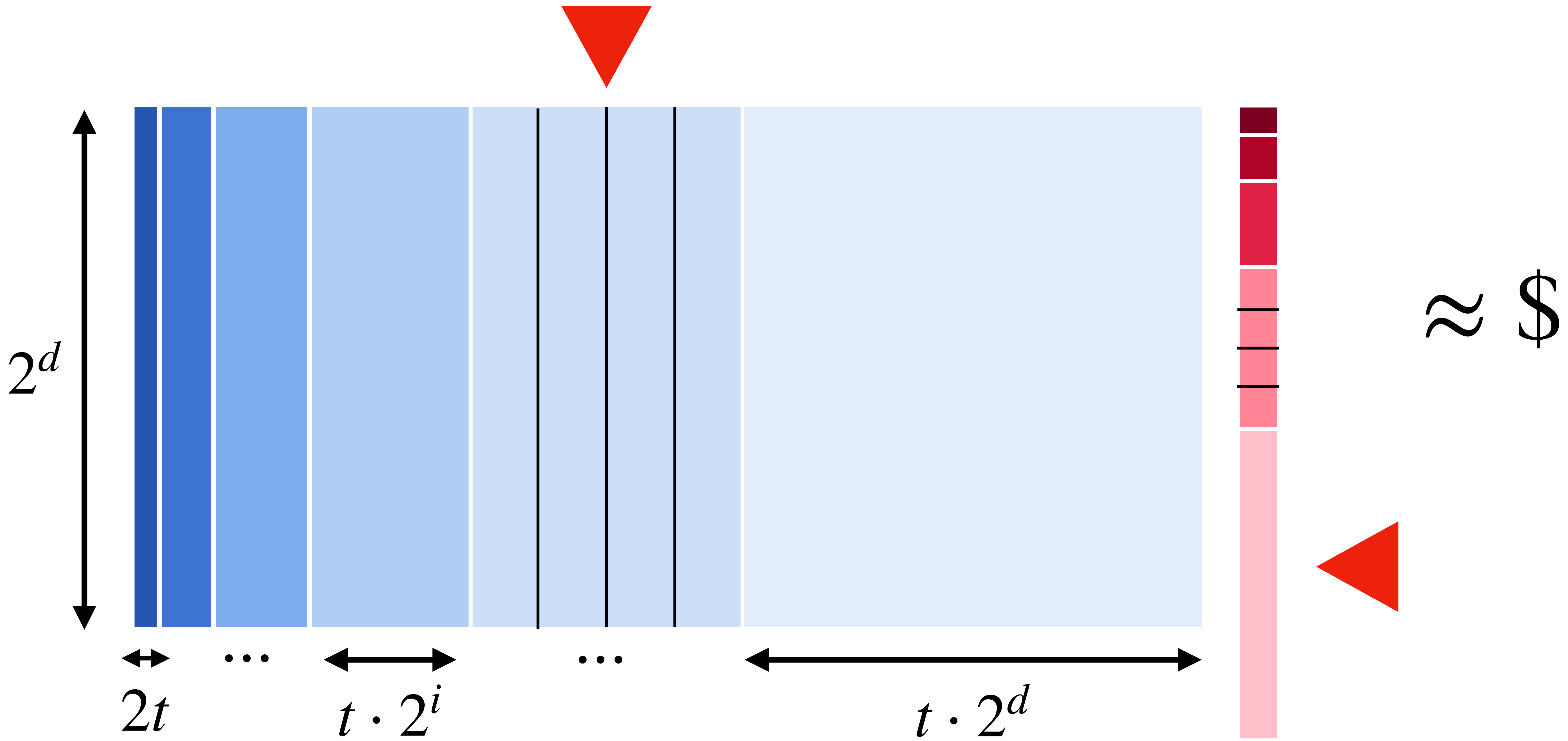
Low-Complexity WPRF from VDLPN



Equation:

$$\bigoplus_{i=1}^{d} \bigoplus_{j=1}^{t} \left\langle \vec{h}_{i,j}, \vec{e}_{i,j} \right\rangle = \bigoplus_{i=1}^{d} \bigoplus_{j=1}^{t} \mathrm{EQ}(x_{i,j}, \bar{K}_{i,j}) = \bigoplus_{i=1}^{d} \bigoplus_{j=1}^{t} \bigwedge_{\ell=1}^{i} \left(x_{i,j} \oplus K_{i,j} \right)$$

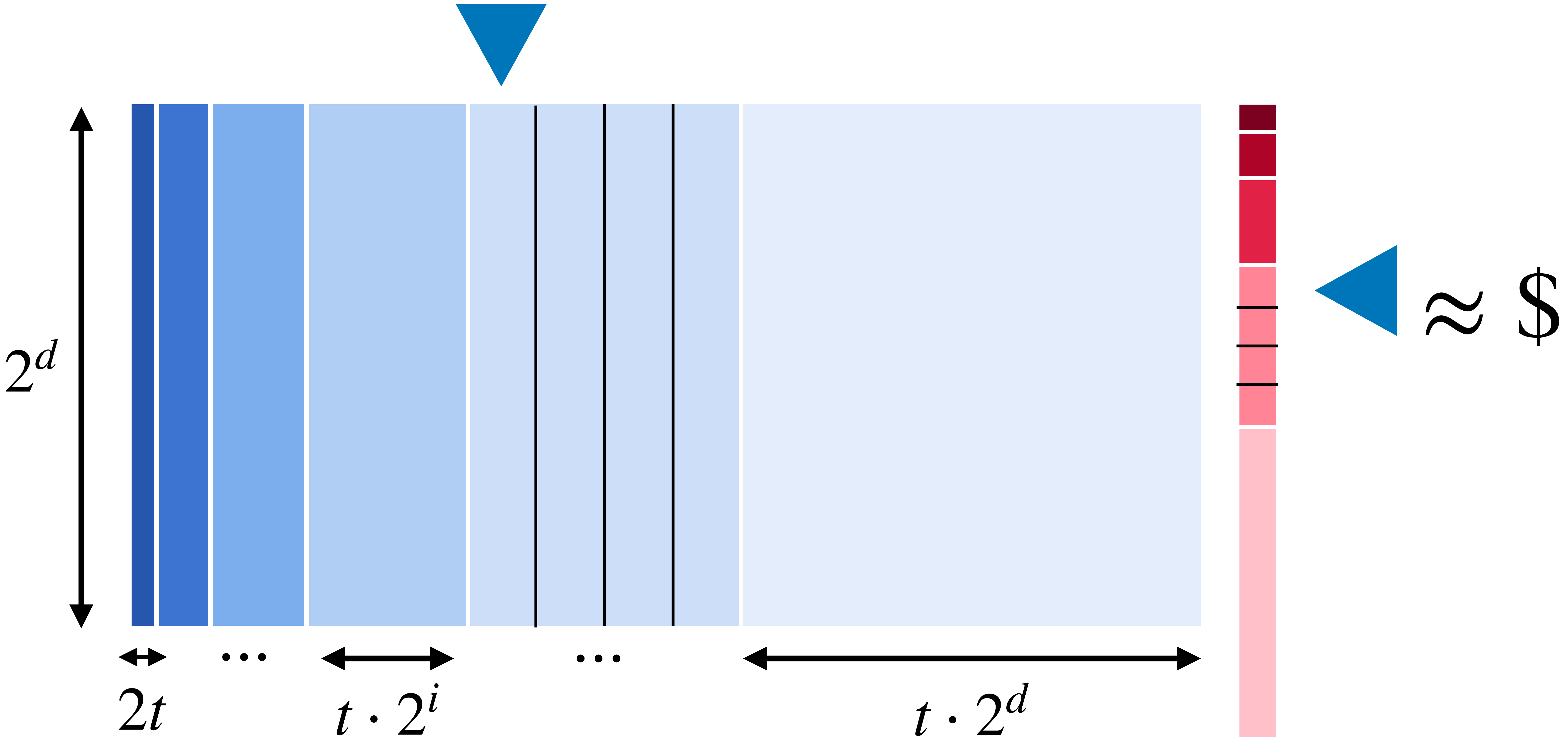
Low-Complexity WPRF from VDLPN



Equation:

$$\bigoplus_{i=1}^{d} \bigoplus_{j=1}^{t} \left\langle \vec{h}_{i,j}, \vec{e}_{i,j} \right\rangle = \bigoplus_{i=1}^{d} \bigoplus_{j=1}^{t} \mathrm{EQ}(x_{i,j}, \bar{K}_{i,j}) = \bigoplus_{i=1}^{d} \bigoplus_{j=1}^{t} \bigwedge_{\ell=1}^{i} \left(x_{i,j} \oplus K_{i,j} \right)$$

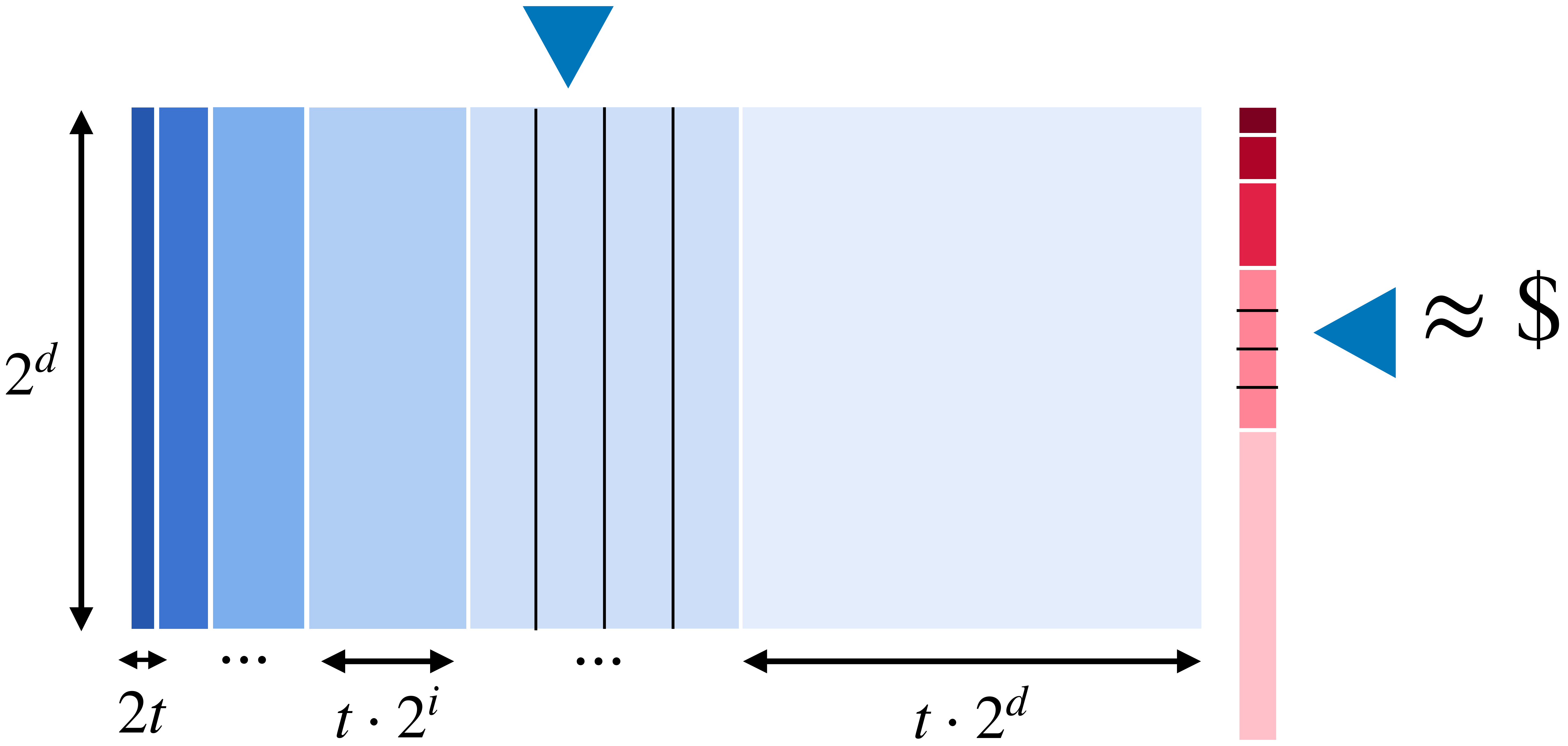
Low-Complexity WPRF from VDLPN



Equation:

$$\bigoplus_{i=1}^{d} \bigoplus_{j=1}^{t} \left\langle \vec{h}_{i,j}, \vec{e}_{i,j} \right\rangle = \bigoplus_{i=1}^{d} \bigoplus_{j=1}^{t} \mathrm{EQ}(x_{i,j}, \bar{K}_{i,j}) = \bigoplus_{i=1}^{d} \bigoplus_{j=1}^{t} \bigwedge_{\ell=1}^{i} \left(x_{i,j} \oplus K_{i,j} \right)$$

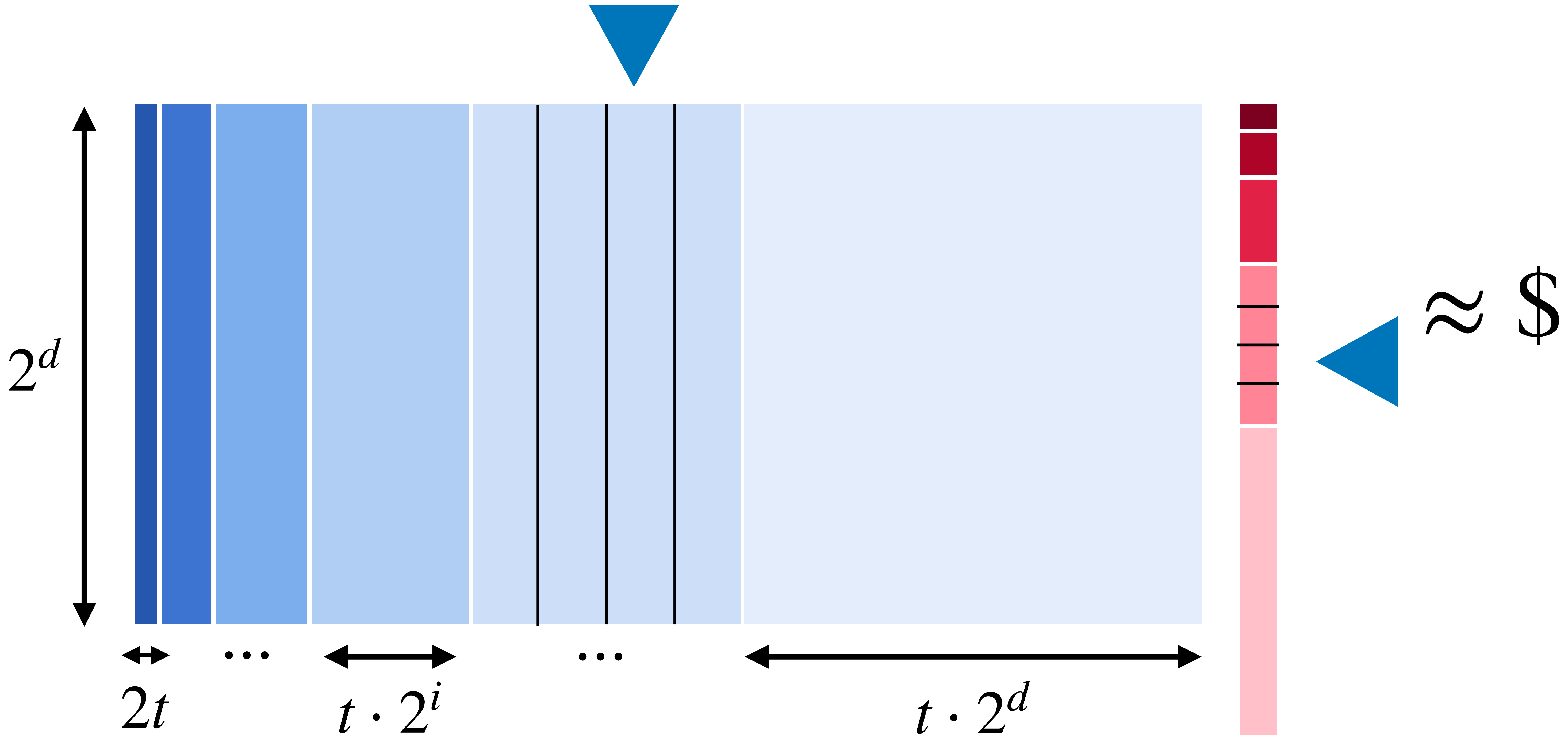
Low-Complexity WPRF from VDLPN



Equation:

$$\bigoplus_{i=1}^{d} \bigoplus_{j=1}^{t} \left\langle \vec{h}_{i,j}, \vec{e}_{i,j} \right\rangle = \bigoplus_{i=1}^{d} \bigoplus_{j=1}^{t} \mathrm{EQ}(x_{i,j}, \bar{K}_{i,j}) = \bigoplus_{i=1}^{d} \bigoplus_{j=1}^{t} \bigwedge_{\ell=1}^{i} \left(x_{i,j} \oplus K_{i,j} \right)$$

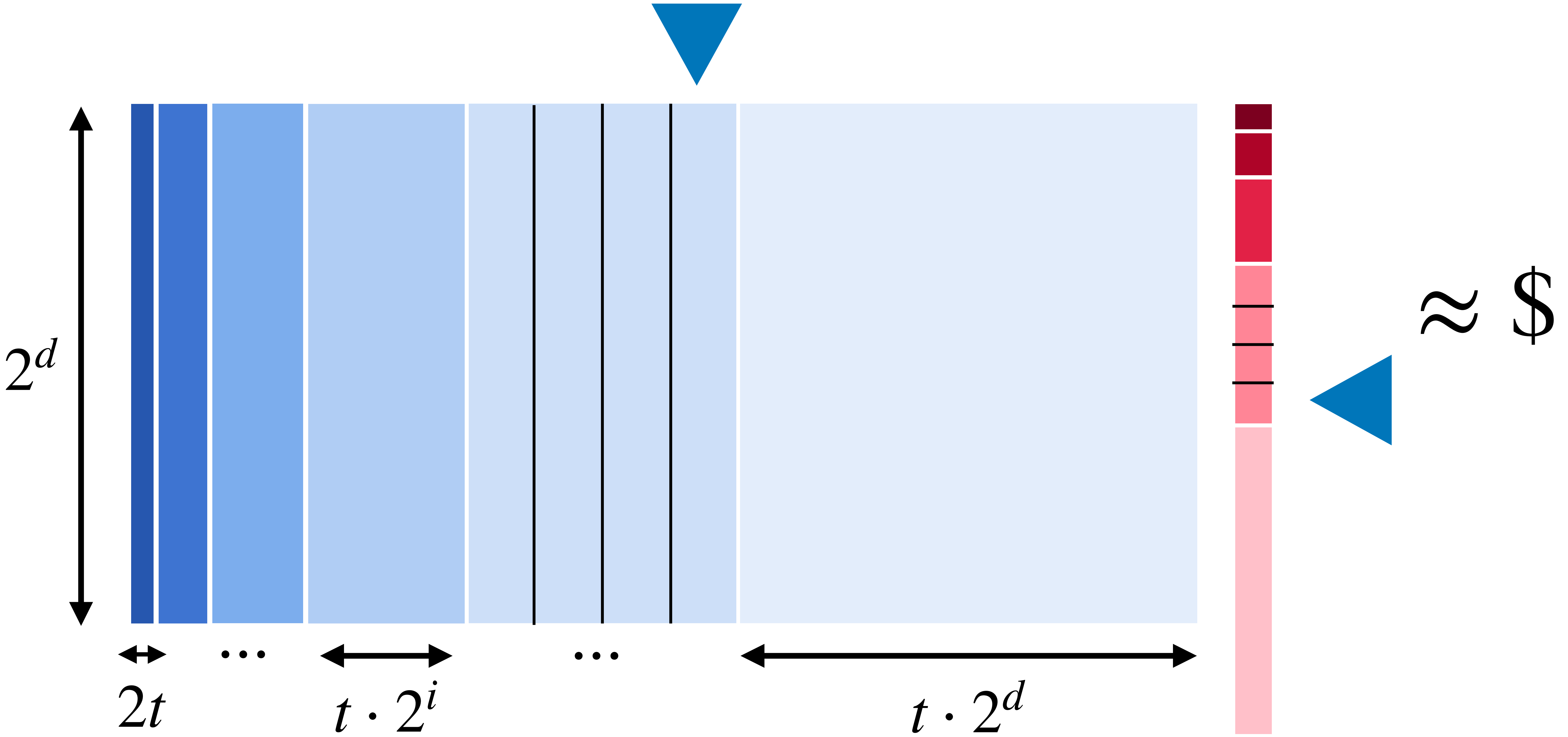
Low-Complexity WPRF from VDLPN



Equation:

$$\bigoplus_{i=1}^{d} \bigoplus_{j=1}^{t} \left\langle \overrightarrow{h}_{i,j}, \overrightarrow{e}_{i,j} \right\rangle = \bigoplus_{i=1}^{d} \bigoplus_{j=1}^{t} \mathrm{EQ}(x_{i,j}, \bar{K}_{i,j}) = \bigoplus_{i=1}^{d} \bigoplus_{j=1}^{t} \bigwedge_{\ell=1}^{i} \left(x_{i,j} \oplus K_{i,j} \right)$$

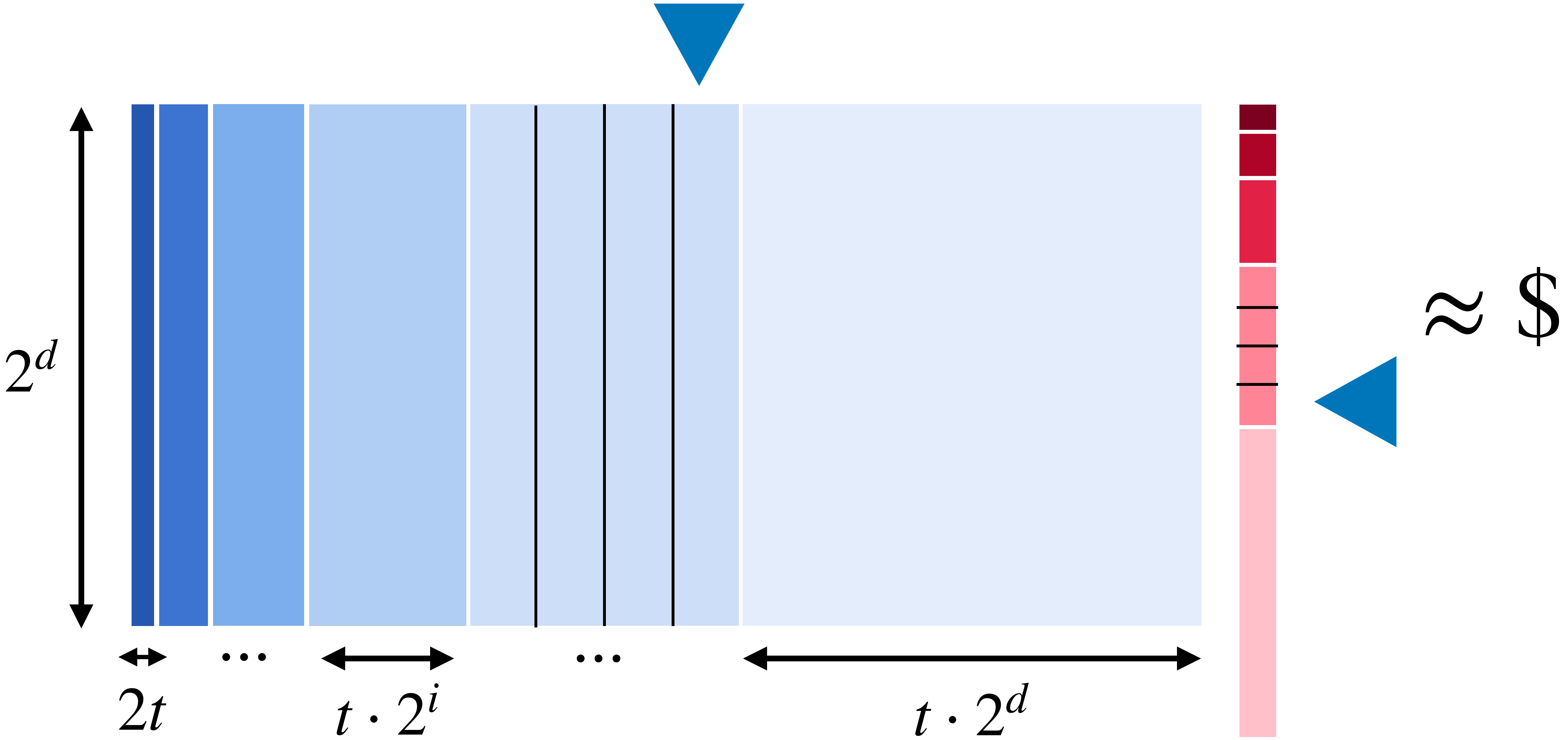
Low-Complexity WPRF from VDLPN



Equation:

$$\bigoplus_{i=1}^{d} \bigoplus_{j=1}^{t} \left\langle \vec{h}_{i,j}, \vec{e}_{i,j} \right\rangle = \bigoplus_{i=1}^{d} \bigoplus_{j=1}^{t} \mathrm{EQ}(x_{i,j}, \bar{K}_{i,j}) = \bigoplus_{i=1}^{d} \bigoplus_{j=1}^{t} \bigwedge_{\ell=1}^{i} \left(x_{i,j} \oplus K_{i,j} \right)$$

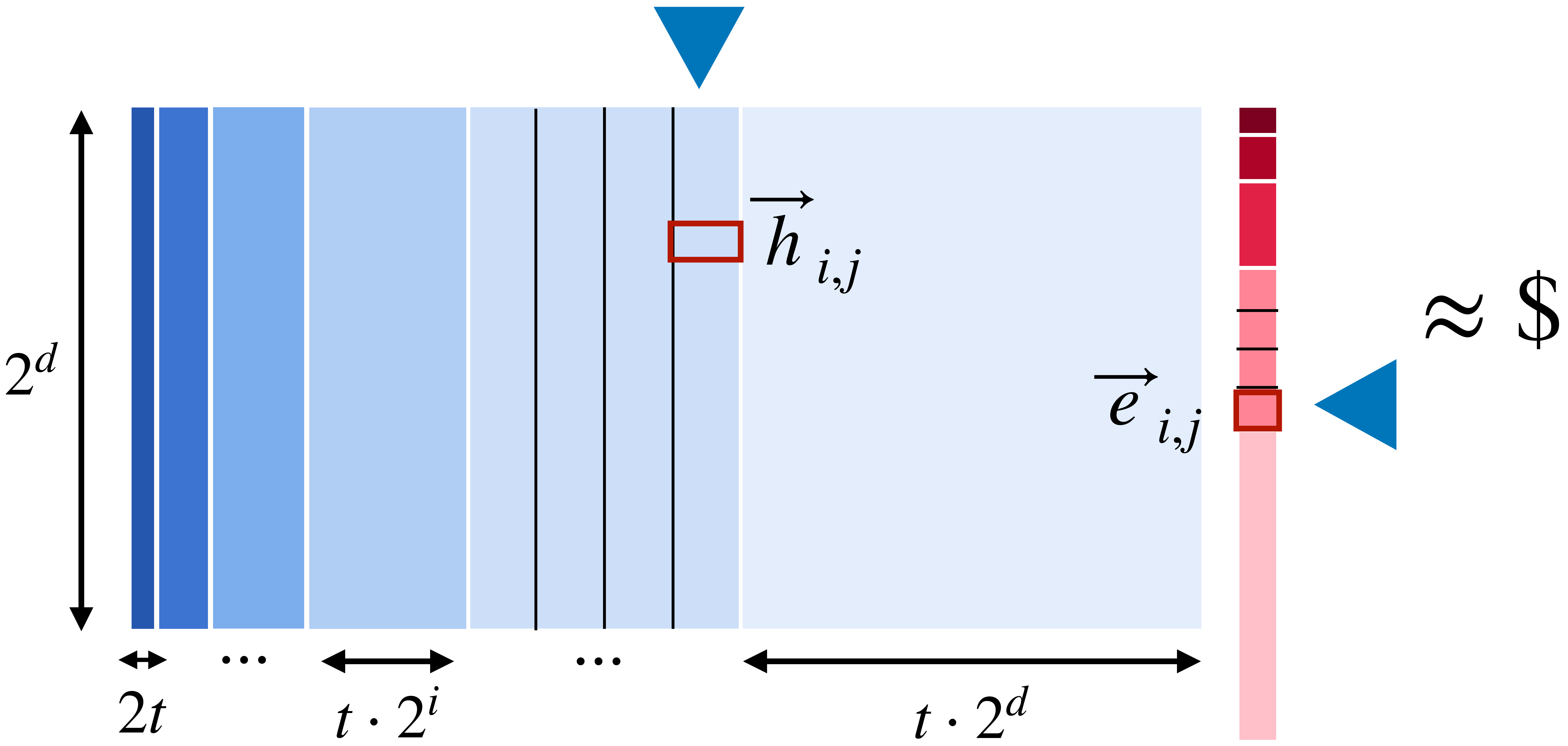
Low-Complexity WPRF from VDLPN



Equation:

$$\bigoplus_{i=1}^{d} \bigoplus_{j=1}^{t} \left\langle \vec{h}_{i,j}, \vec{e}_{i,j} \right\rangle = \bigoplus_{i=1}^{d} \bigoplus_{j=1}^{t} \mathrm{EQ}(x_{i,j}, \bar{K}_{i,j}) = \bigoplus_{i=1}^{d} \bigoplus_{j=1}^{t} \bigwedge_{\ell=1}^{i} \left(x_{i,j} \oplus K_{i,j} \right)$$

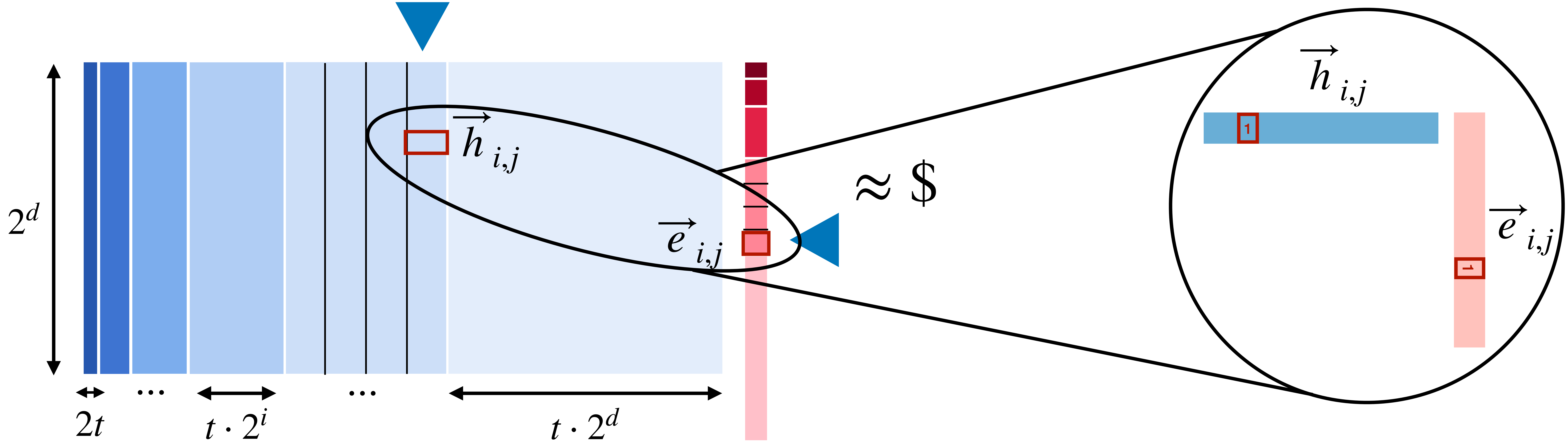
Low-Complexity WPRF from VDLPN



Equation:

$$\bigoplus_{i=1}^{d}\bigoplus_{j=1}^{t}\left\langle \vec{h}_{i,j}, \vec{e}_{i,j}\right\rangle = \bigoplus_{i=1}^{d}\bigoplus_{j=1}^{t}\mathsf{EQ}(x_{i,j}, \bar{K}_{i,j}) = \bigoplus_{i=1}^{d}\bigoplus_{j=1}^{t}\bigwedge_{\ell=1}^{i}\left(x_{i,j} \oplus K_{i,j}\right)$$

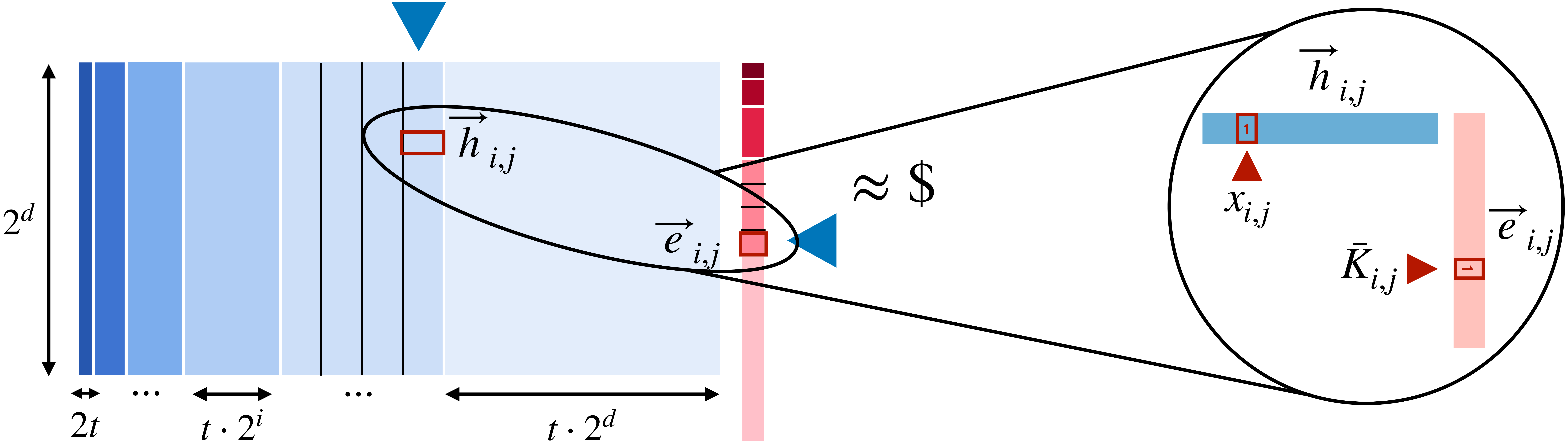
Low-Complexity WPRF from VDLPN



Equation:

$$\bigoplus_{i=1}^{d}\bigoplus_{j=1}^{t}\left\langle \vec{h}_{i,j}, \vec{e}_{i,j}\right\rangle = \bigoplus_{i=1}^{d}\bigoplus_{j=1}^{t}\mathrm{EQ}(x_{i,j}, \bar{K}_{i,j}) = \bigoplus_{i=1}^{d}\bigoplus_{j=1}^{t}\bigwedge_{\ell=1}^{i}\left(x_{i,j} \oplus K_{i,j}\right)$$

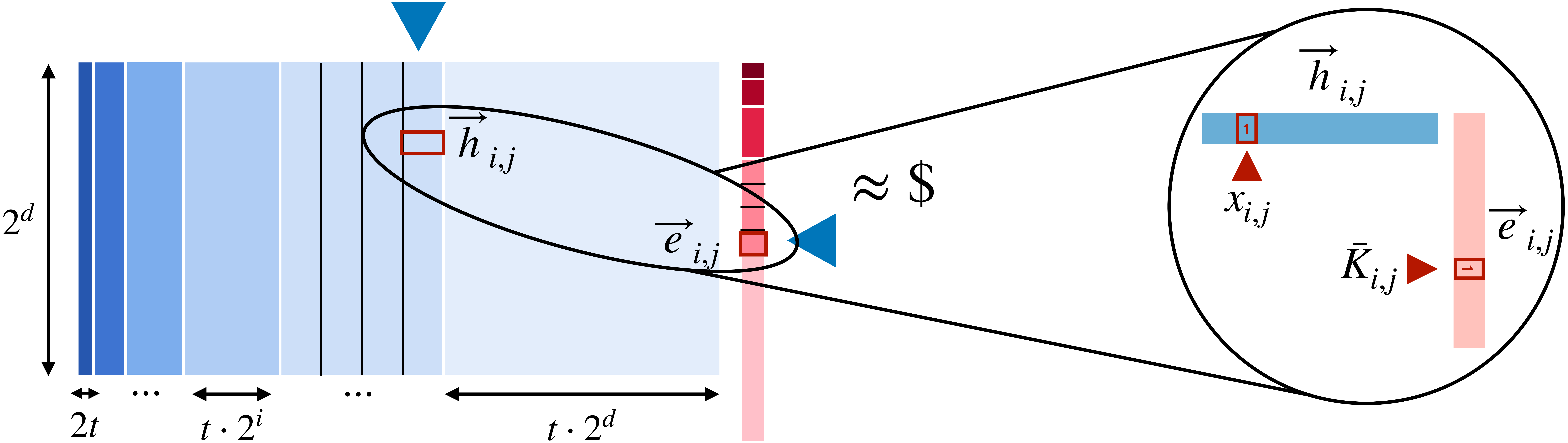
Low-Complexity WPRF from VDLPN



Equation:

$$\bigoplus_{i=1}^{d}\bigoplus_{j=1}^{t}\left\langle \vec{h}_{i,j}, \vec{e}_{i,j}\right\rangle = \bigoplus_{i=1}^{d}\bigoplus_{j=1}^{t}\mathrm{EQ}(x_{i,j}, \bar{K}_{i,j}) = \bigoplus_{i=1}^{d}\bigoplus_{j=1}^{t}\bigwedge_{\ell=1}^{i}\left(x_{i,j} \oplus K_{i,j}\right)$$

Low-Complexity WPRF from VDLPN

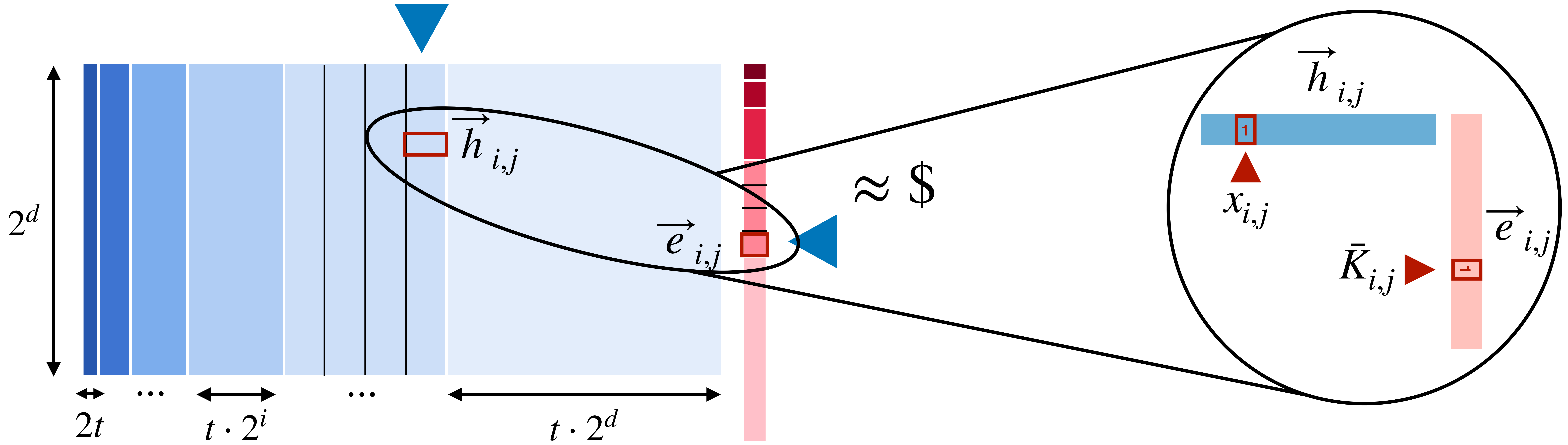


2^d
$2t$
$\cdots$
$t \cdot 2^i$
$\cdots$
$t \cdot 2^d$
$\overrightarrow{h}_{i,j}$
$\overrightarrow{e}_{i,j}$
$\approx \$$
$\overrightarrow{h}_{i,j}$
1
$x_{i,j}$
$\overrightarrow{e}_{i,j}$
$\bar{K}_{i,j}$
1

Equation:

$$\bigoplus_{i=1}^{d}\bigoplus_{j=1}^{t}\left\langle \overrightarrow{h}_{i,j}, \overrightarrow{e}_{i,j}\right\rangle = \bigoplus_{i=1}^{d}\bigoplus_{j=1}^{t}\mathrm{EQ}(x_{i,j}, \bar{K}_{i,j}) = \bigoplus_{i=1}^{d}\bigoplus_{j=1}^{t}\bigwedge_{\ell=1}^{i}\left(x_{i,j} \oplus K_{i,j}\right)$$

Low-Complexity WPRF from VDLPN



Equation:

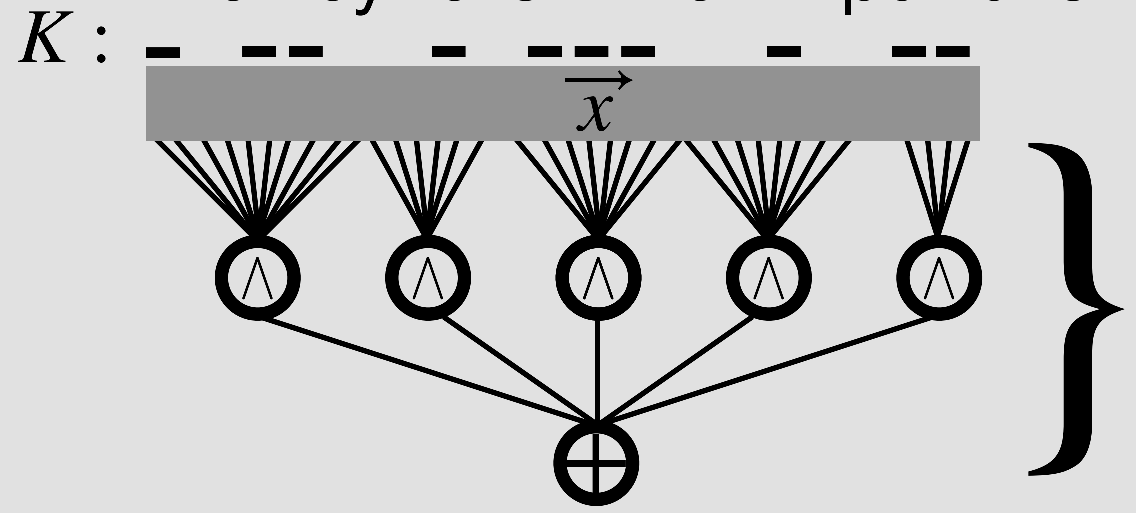
$$\bigoplus_{i=1}^{d}\bigoplus_{j=1}^{t}\left\langle \vec{h}_{i,j}, \vec{e}_{i,j} \right\rangle = \bigoplus_{i=1}^{d}\bigoplus_{j=1}^{t}\mathrm{EQ}(x_{i,j}, \bar{K}_{i,j}) = \bigoplus_{i=1}^{d}\bigoplus_{j=1}^{t}\bigwedge_{\ell=1}^{i}\left(x_{i,j} \oplus K_{i,j}\right)$$

Candidate low-complexity WPRF:

- $n = |x| = |K| = t \cdot d \cdot (d-1)/2$
- Security up to $2^d = 2^{n^{1/3}}$ samples against $2^{n^{1/3}}$-time adversaries?

$$F_K(x) = \bigoplus_{i=1}^{d}\bigoplus_{j=1}^{t}\bigwedge_{\ell=1}^{i}\left(x_{i,j,\ell} \oplus K_{i,j,\ell}\right)$$

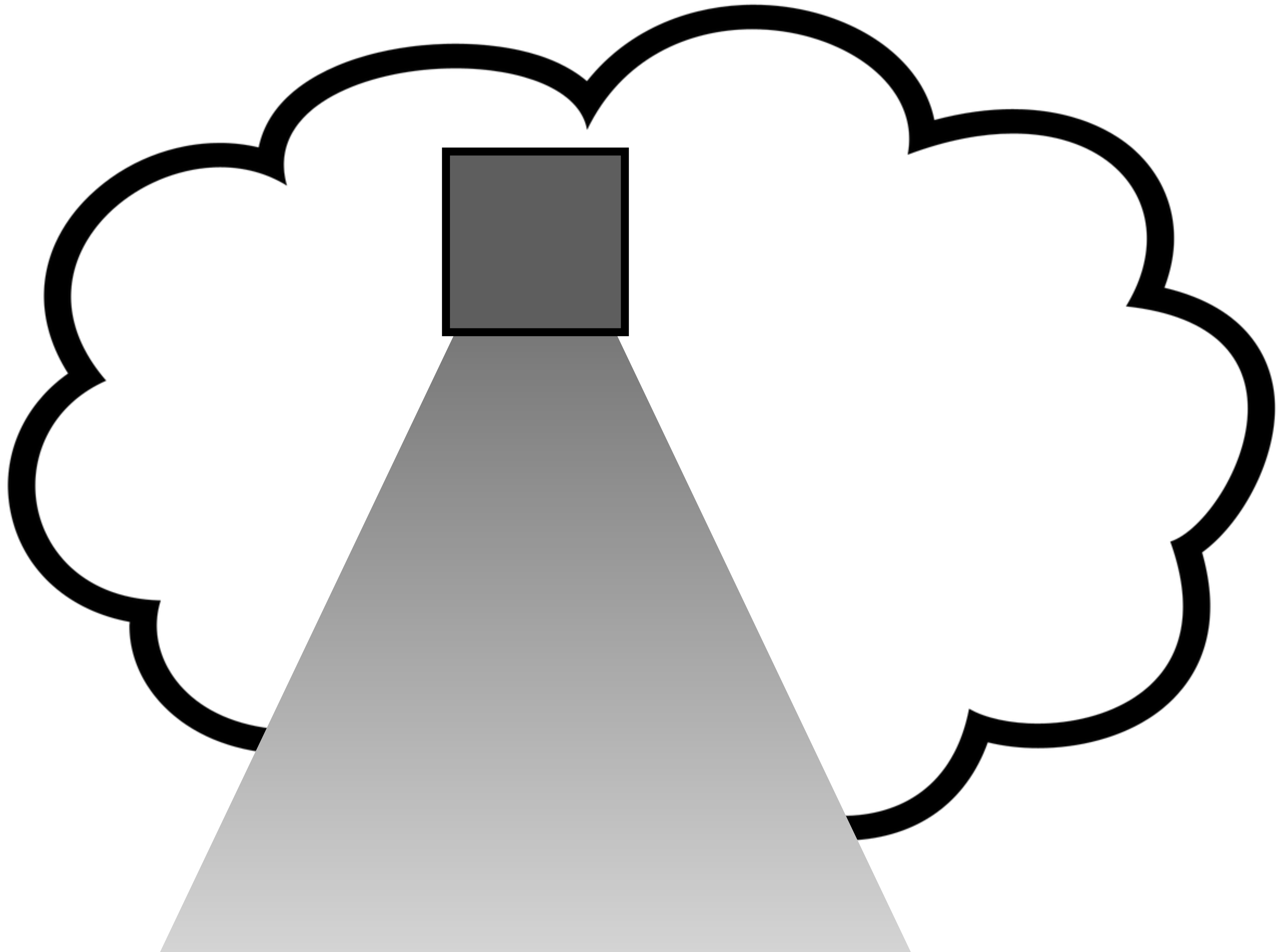
The key tells which input bits to negate:



The circuit is fixed

Security of Variable-Density LPN - Linear Tests

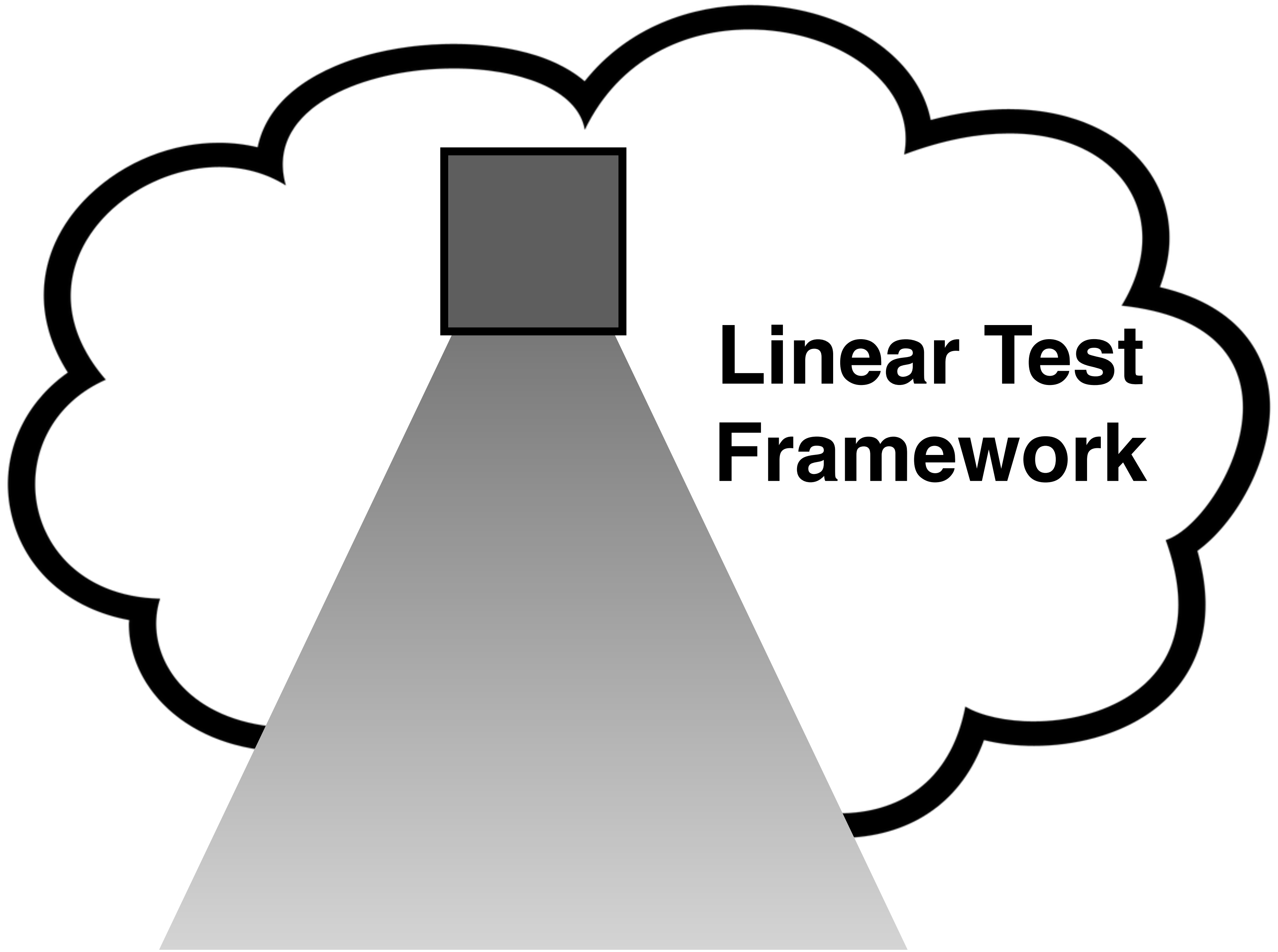
A tremendous number of attacks on LPN have been published…



- **Gaussian Elimination attacks**
  - Standard gaussian elimination
  - Blum-Kalai-Wasserman [J.ACM:BKW03]
  - Sample-efficient BKW [A-R:Lyu05]
  - Pooled Gauss [CRYPTO:EKM17]
  - Well-pooled Gauss [CRYPTO:EKM17]
  - Leviel-Fouque [SCN:LF06]
  - Covering codes [JC:GJL19]
  - Covering codes+ [BTV15]
  - Covering codes++ [BV:AC16]
  - Covering codes+++ [EC:ZJW16]
- **Statistical Decoding Attacks**
  - Jabri's attack [ICCC:Jab01]
  - Overbeck's variant [ACISP:Ove06]
  - FKI's variant [Trans.IT:FKI06]
  - Debris-Tillich variant [ISIT:DT17]
- **Information Set Decoding Attacks**
  - Prange's algorithm [Prange62]
  - Stern's variant [ICIT:Stern88]
  - Finiasz and Sendrier's variant [AC:FS09]
  - BJMM variant [EC:BJMM12]
  - May-Ozerov variant [EC:MO15]
  - Both-May variant [PQC:BM18]
  - MMT variant [AC:MMT11]
  - Well-pooled MMT [CRYPTO:EKM17]
  - BLP variant [CRYPTO:BLP11]
- **Other Attacks**
  - Generalized birthday [CRYPTO:Wag02]
  - Improved GBA [Kirchner11]
  - Linearization [EC:BM97]
  - Linearization 2 [INDO:Saa07]
  - Low-weight parity-check [Zichron17]
  - Low-deg approx [ITCS:ABGKR17]

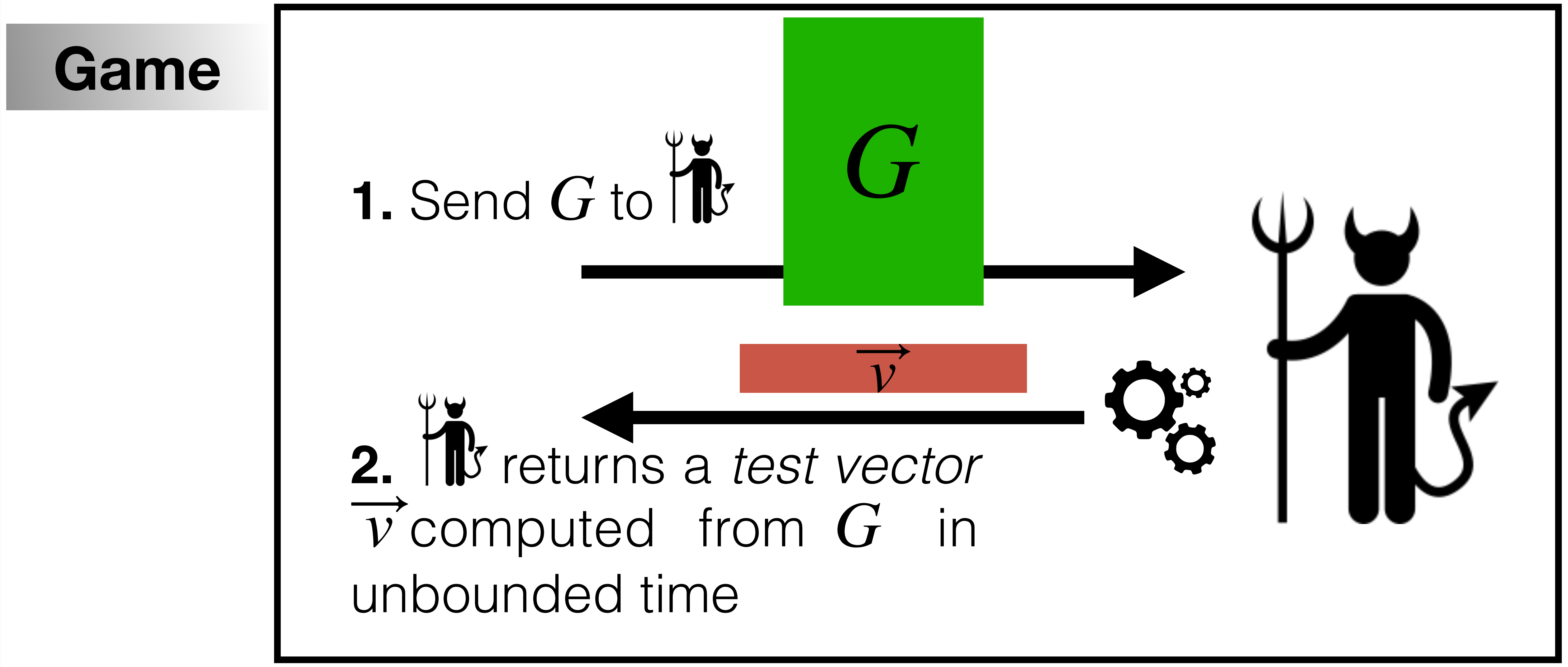
Security of Variable-Density LPN - Linear Tests

A tremendous number of attacks on LPN have been published…



- **Gaussian Elimination attacks**
  - Standard gaussian elimination
  - Blum-Kalai-Wasserman [J.ACM:BKW03]
  - Sample-efficient BKW [A-R:Lyu05]
  - Pooled Gauss [CRYPTO:EKM17]
  - Well-pooled Gauss [CRYPTO:EKM17]
  - Leviel-Fouque [SCN:LF06]
  - Covering codes [JC:GJL19]
  - Covering codes+ [BTV15]
  - Covering codes++ [BV:AC16]
  - Covering codes+++ [EC:ZJW16]
- **Statistical Decoding Attacks**
  - Jabri's attack [ICCC:Jab01]
  - Overbeck's variant [ACISP:Ove06]
  - FKI's variant [Trans.IT:FKI06]
  - Debris-Tillich variant [ISIT:DT17]
- **Information Set Decoding Attacks**
  - Prange's algorithm [Prange62]
  - Stern's variant [ICIT:Stern88]
  - Finiasz and Sendrier's variant [AC:FS09]
  - BJMM variant [EC:BJMM12]
  - May-Ozerov variant [EC:MO15]
  - Both-May variant [PQC:BM18]
  - MMT variant [AC:MMT11]
  - Well-pooled MMT [CRYPTO:EKM17]
  - BLP variant [CRYPTO:BLP11]
- **Other Attacks**
  - Generalized birthday [CRYPTO:Wag02]
  - Improved GBA [Kirchner11]
  - Linearization [EC:BM97]
  - Linearization 2 [INDO:Saa07]
  - Low-weight parity-check [Zichron17]
  - Low-deg approx [ITCS:ABGKR17]

Crucial observation: *all* these attacks fit in the same framework, the *linear test framework.*

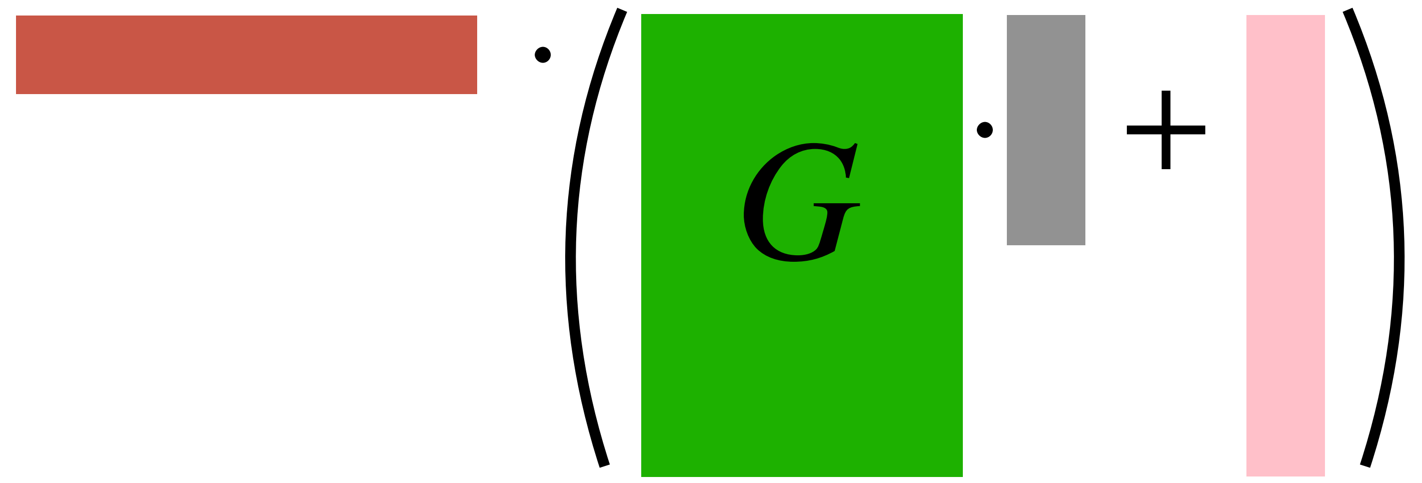


1. Send G to 😈

2. 😈 returns a *test vector* $\vec{v}$ computed from G in unbounded time

Check

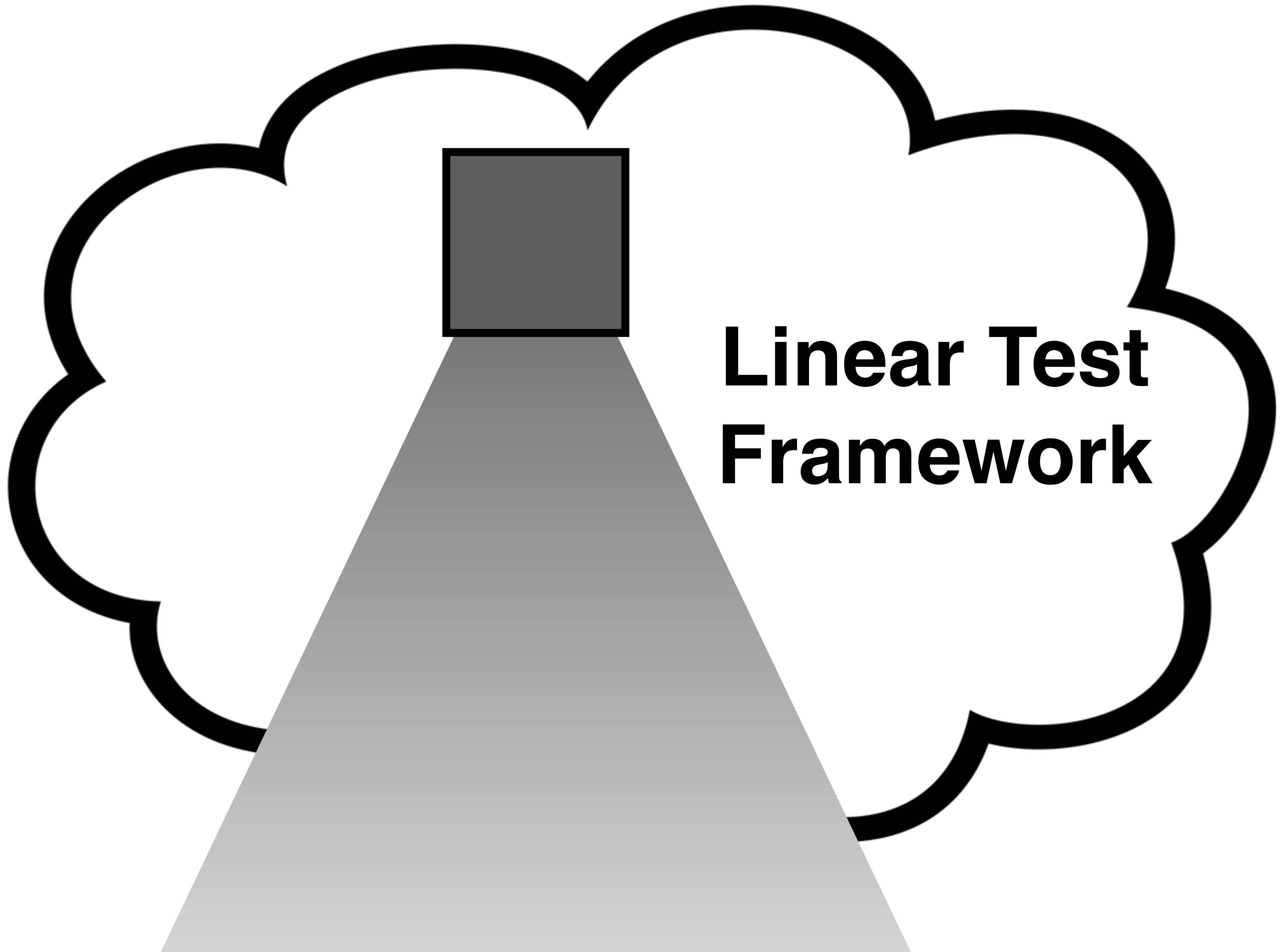
The adversary wins in the distribution induced by



(over a random choice of secret and sparse noise) is non-negligibly *biased*.

Security of Variable-Density LPN - Linear Tests

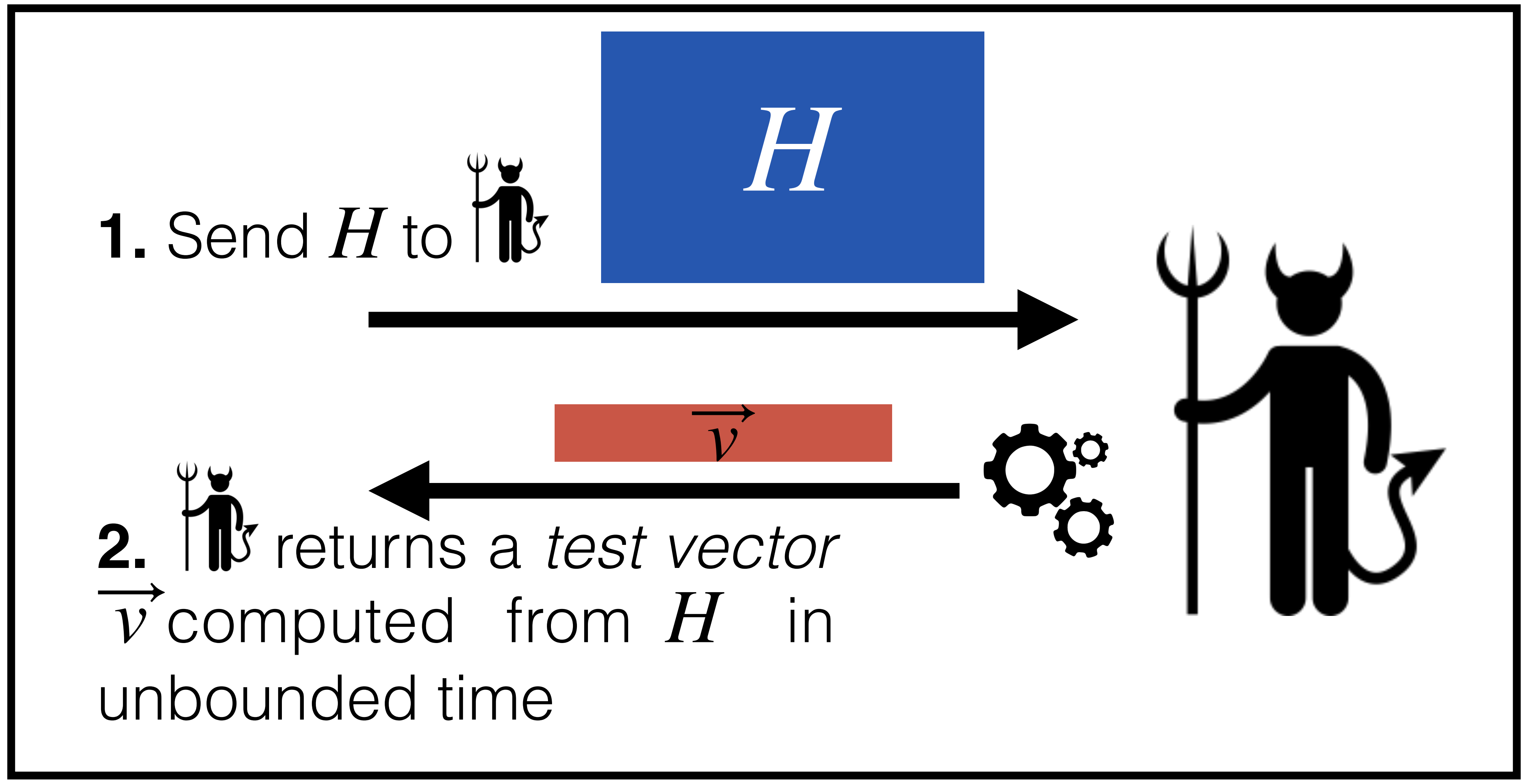
A tremendous number of attacks on LPN have been published…



- **Gaussian Elimination attacks**
  - Standard gaussian elimination
  - Blum-Kalai-Wasserman [J.ACM:BKW03]
  - Sample-efficient BKW [A-R:Lyu05]
  - Pooled Gauss [CRYPTO:EKM17]
  - Well-pooled Gauss [CRYPTO:EKM17]
  - Leviel-Fouque [SCN:LF06]
  - Covering codes [JC:GJL19]
  - Covering codes+ [BTV15]
  - Covering codes++ [BV:AC16]
  - Covering codes+++ [EC:ZJW16]
- **Statistical Decoding Attacks**
  - Jabri's attack [ICCC:Jab01]
  - Overbeck's variant [ACISP:Ove06]
  - FKI's variant [Trans.IT:FKI06]
  - Debris-Tillich variant [ISIT:DT17]
- **Information Set Decoding Attacks**
  - Prange's algorithm [Prange62]
  - Stern's variant [ICIT:Stern88]
  - Finiasz and Sendrier's variant [AC:FS09]
  - BJMM variant [EC:BJMM12]
  - May-Ozerov variant [EC:MO15]
  - Both-May variant [PQC:BM18]
  - MMT variant [AC:MMT11]
  - Well-pooled MMT [CRYPTO:EKM17]
  - BLP variant [CRYPTO:BLP11]
- **Other Attacks**
  - Generalized birthday [CRYPTO:Wag02]
  - Improved GBA [Kirchner11]
  - Linearization [EC:BM97]
  - Linearization 2 [INDO:Saa07]
  - Low-weight parity-check [Zichron17]
  - Low-deg approx [ITCS:ABGKR17]

Crucial observation: *all* these attacks fit in the same framework, the *linear test framework.*

Game

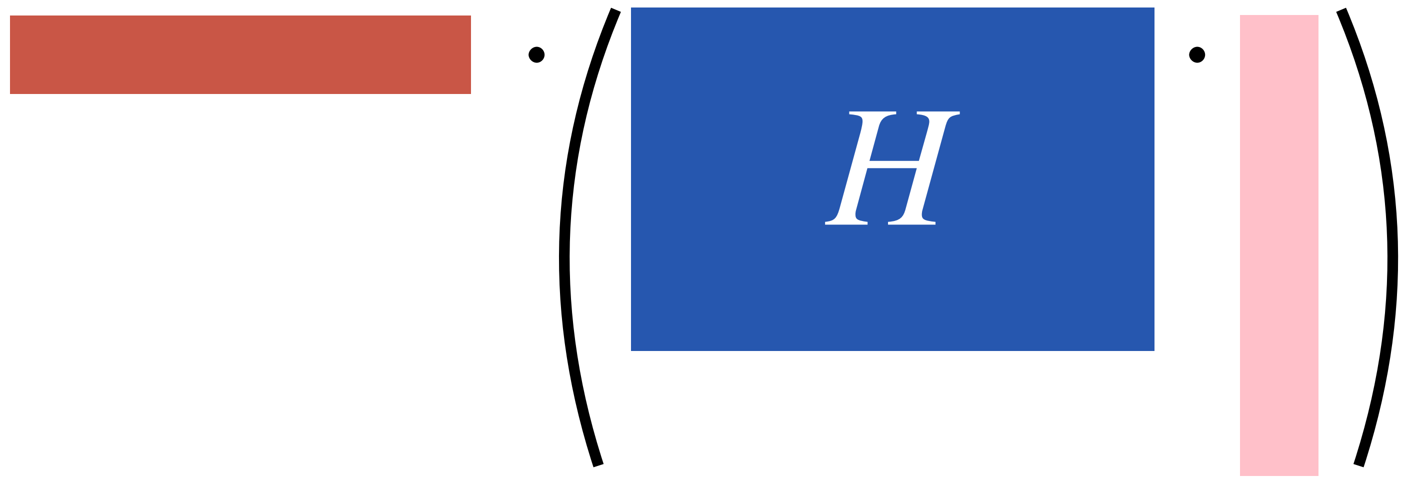


1. Send H to

2. returns a *test vector* $\vec{v}$ computed from H in unbounded time

Check

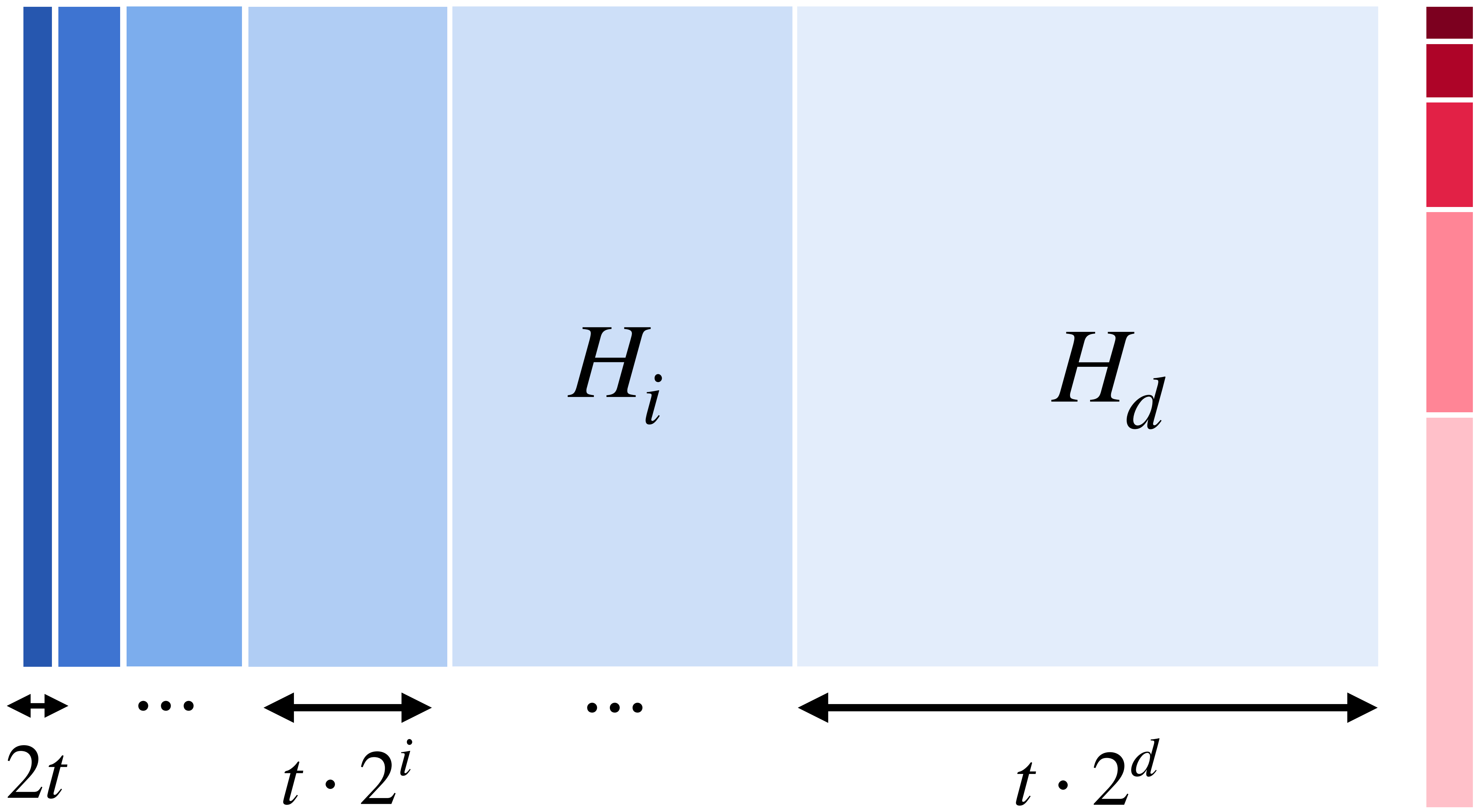
The adversary wins in the distribution induced by



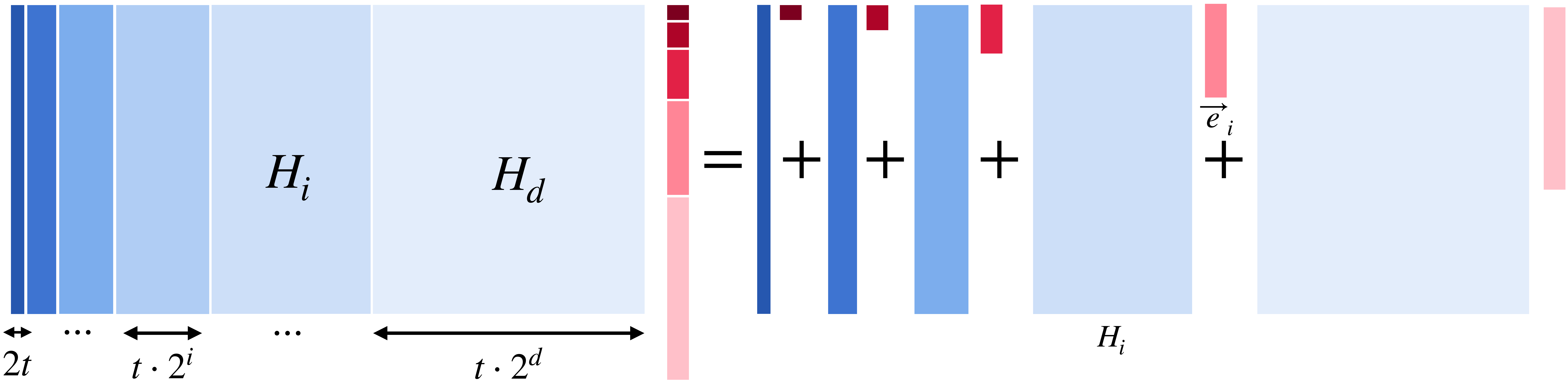
(over a random choice of secret and sparse noise) is non-negligibly *biased*.

Security of Variable-Density LPN - Linear Tests

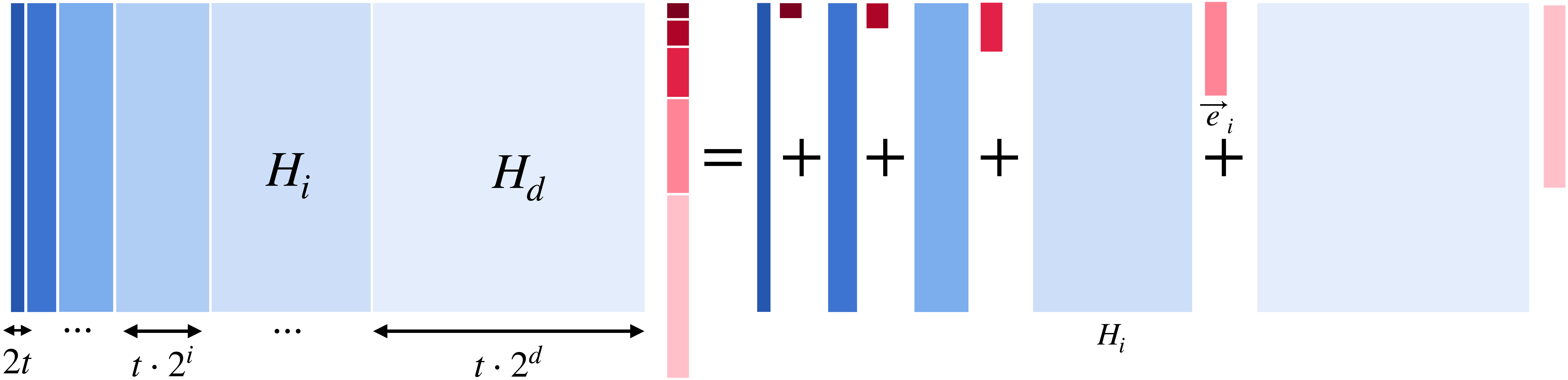
Security of Variable-Density LPN - Linear Tests



Security of Variable-Density LPN - Linear Tests

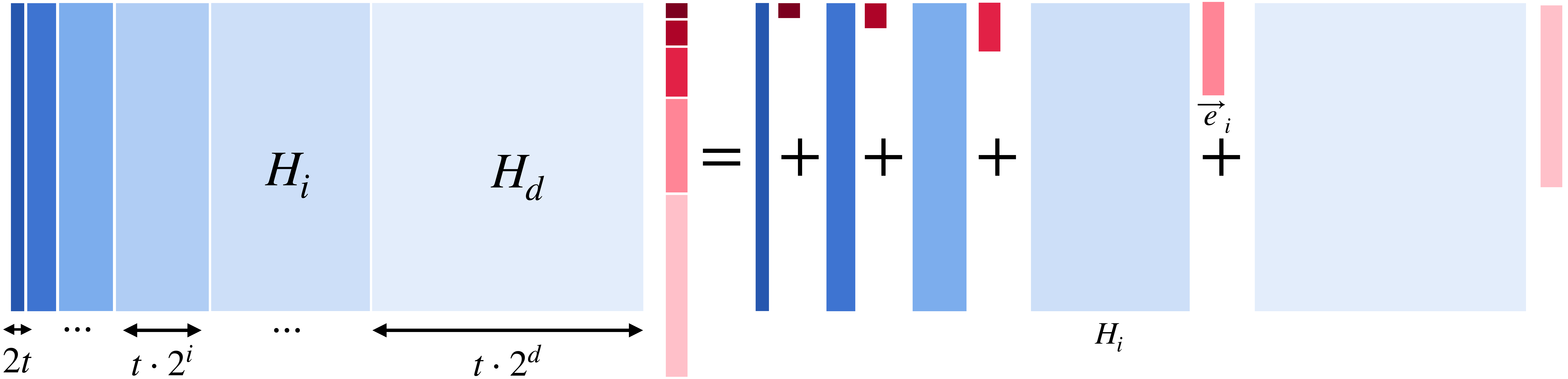


Security of Variable-Density LPN - Linear Tests

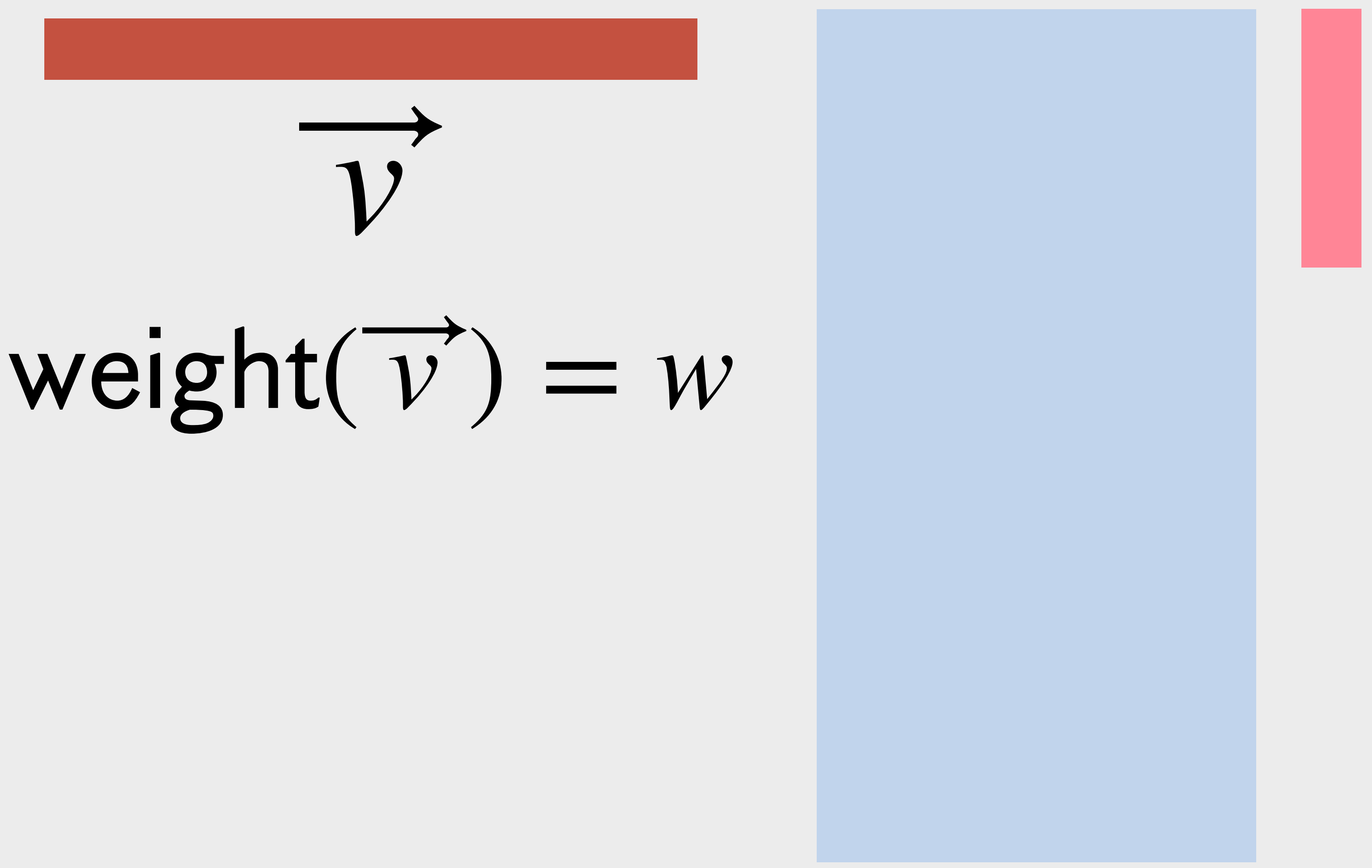


Claim: w.h.p. over the choice of H_i, the distribution $\vec{v} \cdot (H_i \cdot \vec{e}_i)$ has bias $2^{-O(t)}$ for *all* $\vec{v}$ of Hamming weight in $[2^{i-1}, 2^i]$.

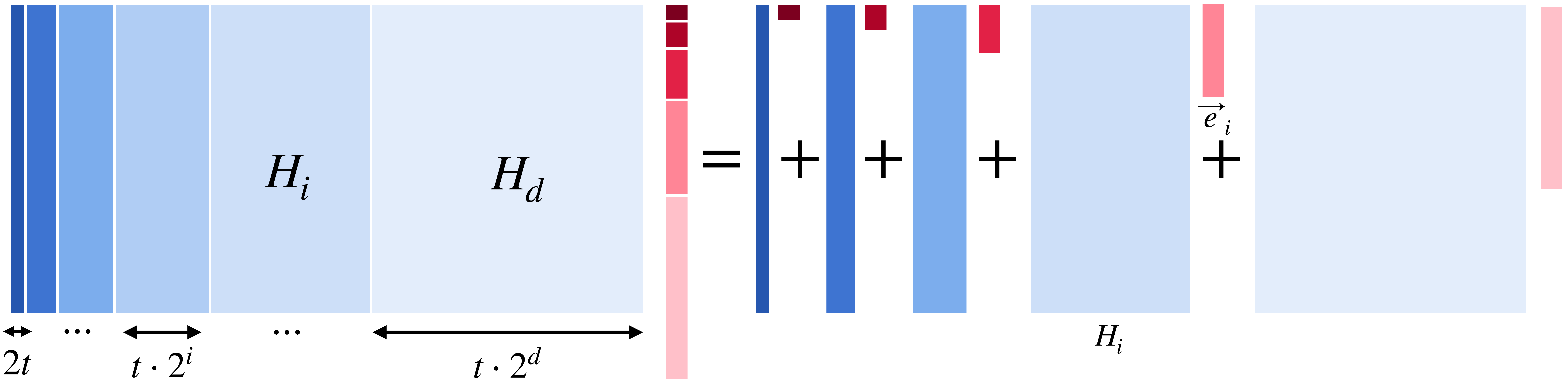
Security of Variable-Density LPN - Linear Tests



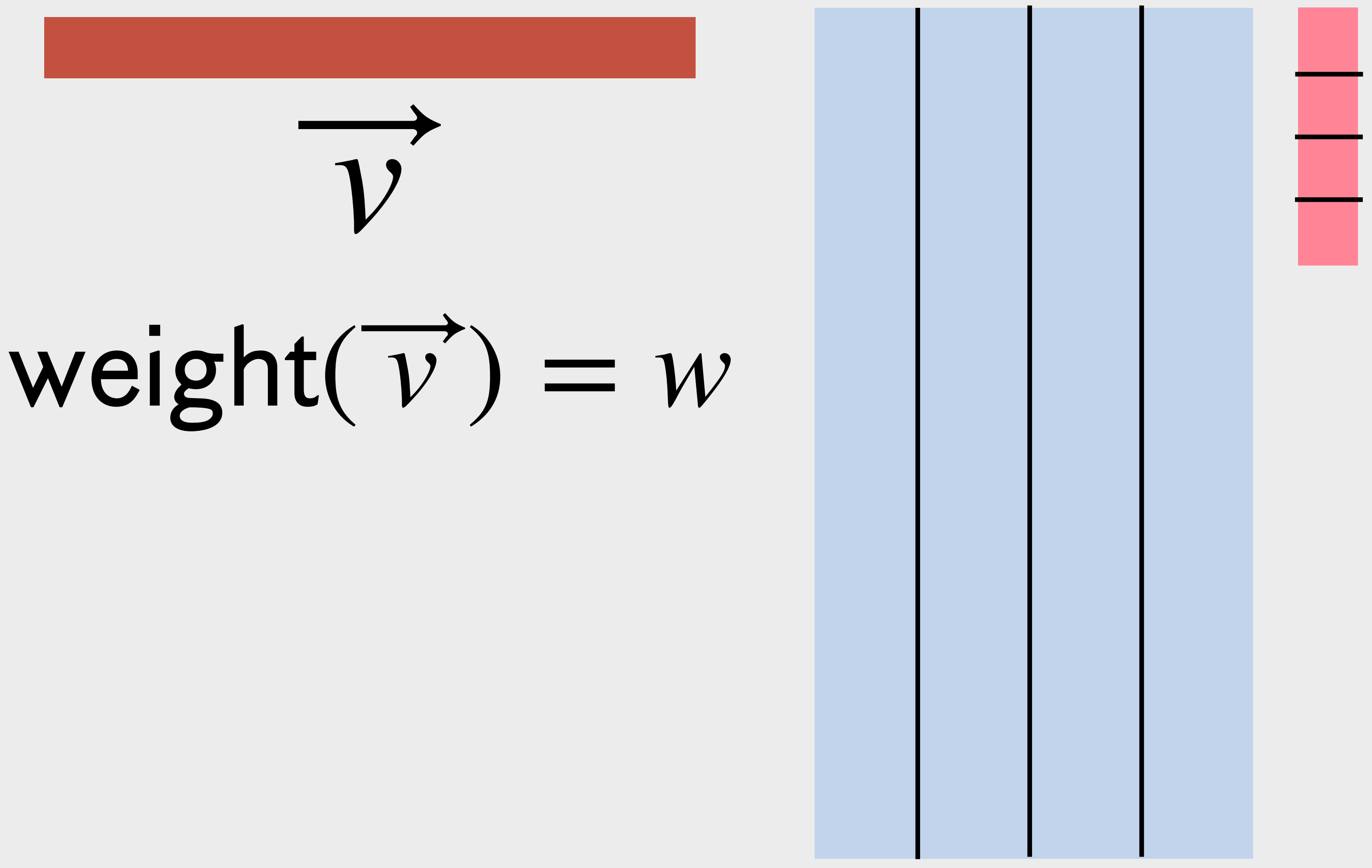
Claim: w.h.p. over the choice of H_i, the distribution $\vec{v} \cdot (H_i \cdot \vec{e}_i)$ has bias $2^{-O(t)}$ for *all* $\vec{v}$ of Hamming weight in $[2^{i-1}, 2^i]$.



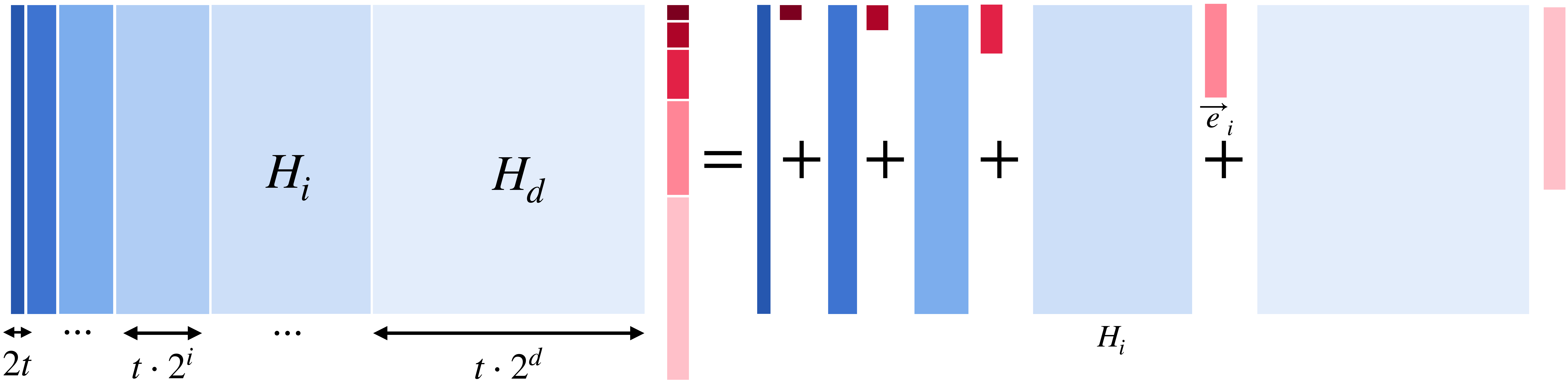
Security of Variable-Density LPN - Linear Tests



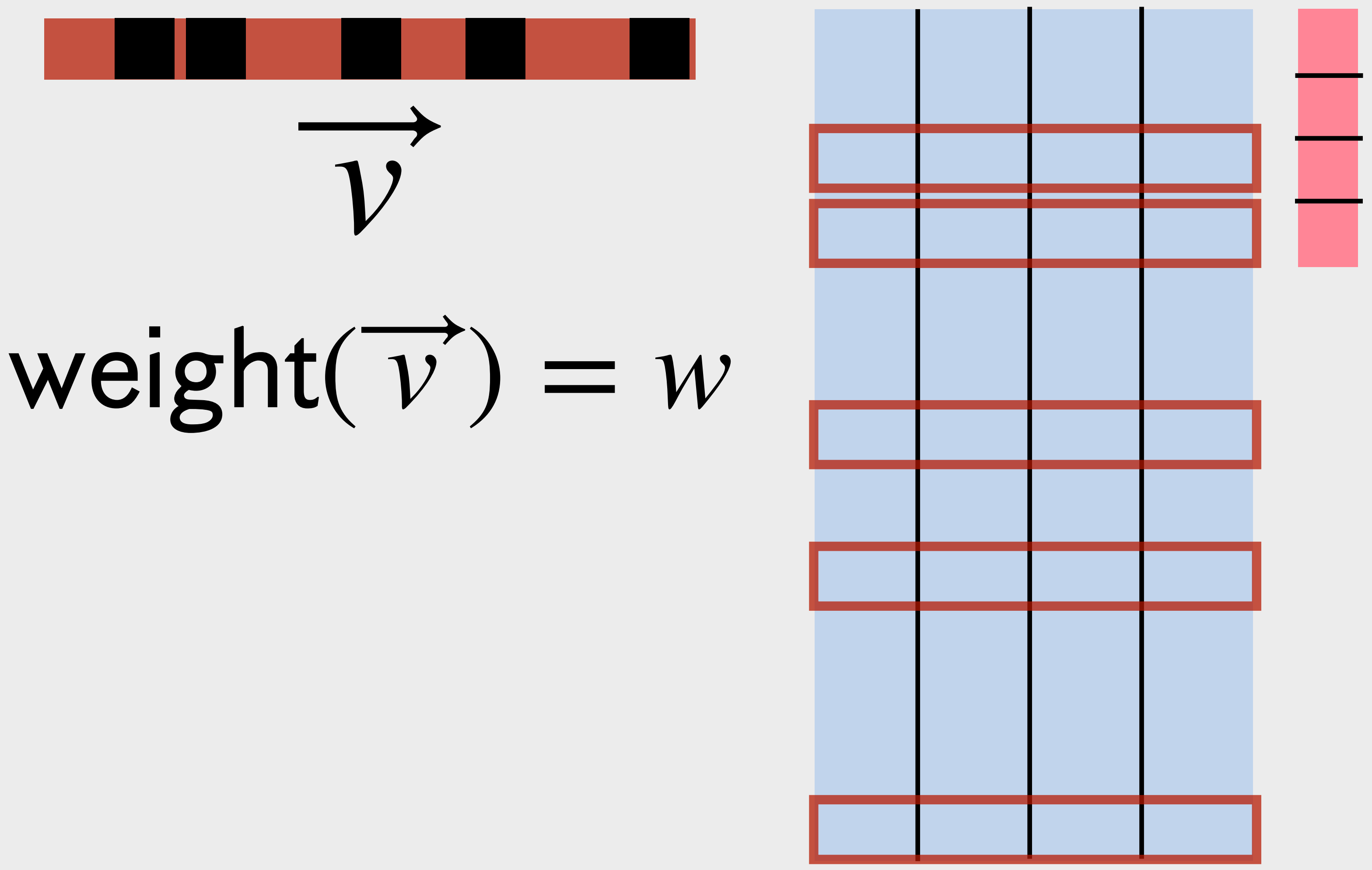
Claim: w.h.p. over the choice of H_i , the distribution $\vec{v} \cdot (H_i \cdot \vec{e}_i)$ has bias $2^{-O(t)}$ for *all* $\vec{v}$ of Hamming weight in $[2^{i-1}, 2^i]$.



Security of Variable-Density LPN - Linear Tests

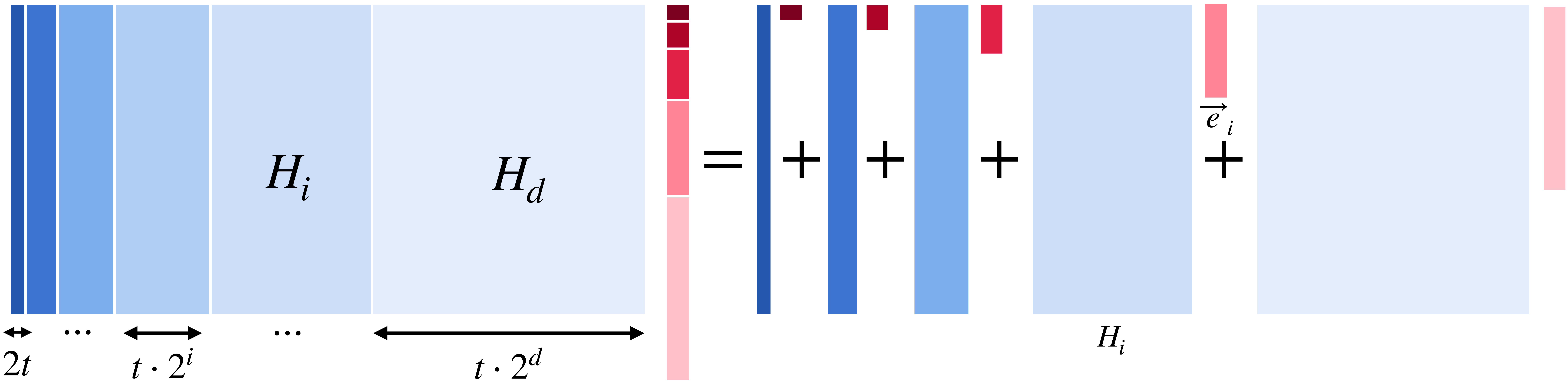


Claim: w.h.p. over the choice of H_i , the distribution $\vec{v} \cdot (H_i \cdot \vec{e}_i)$ has bias $2^{-O(t)}$ for *all* $\vec{v}$ of Hamming weight in $[2^{i-1}, 2^i]$.

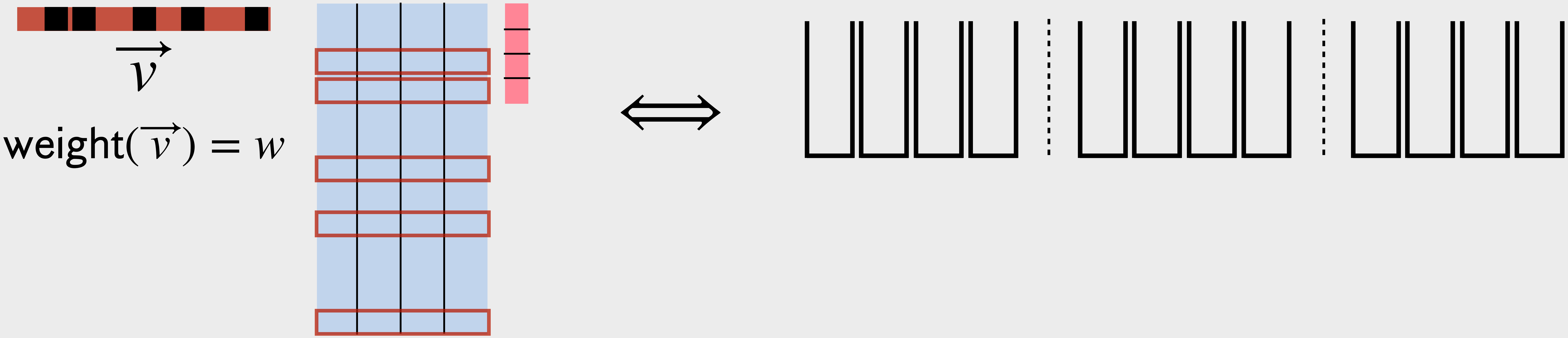


$\text{weight}(\vec{v}) = w$

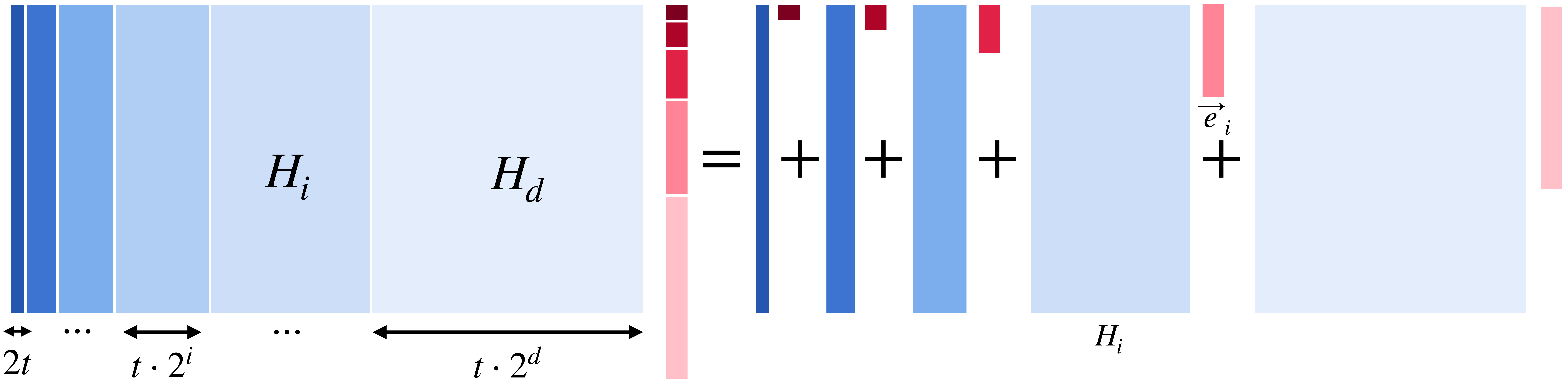
Security of Variable-Density LPN - Linear Tests



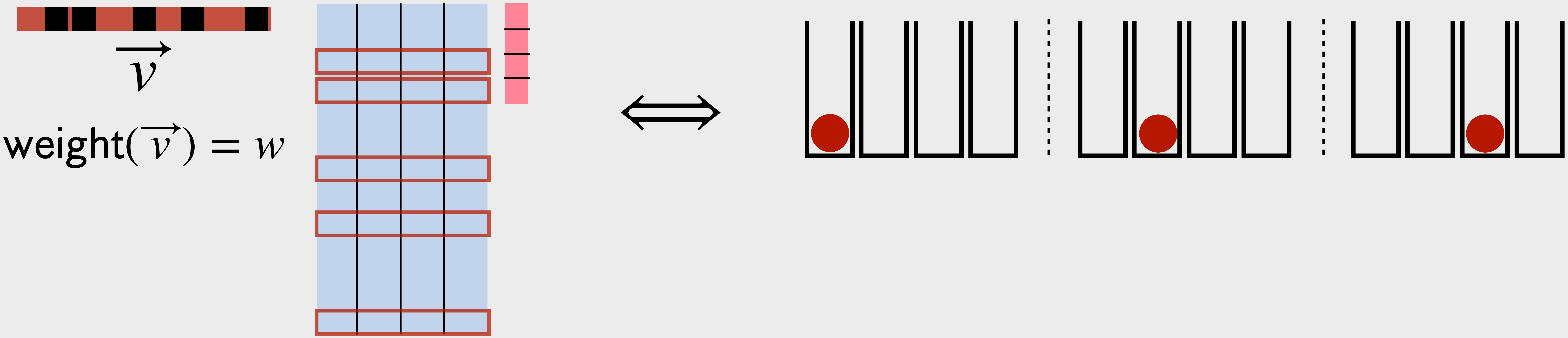
Claim: w.h.p. over the choice of H_i , the distribution $\vec{v} \cdot (H_i \cdot \vec{e}_i)$ has bias $2^{-O(t)}$ for *all* $\vec{v}$ of Hamming weight in $[2^{i-1}, 2^i]$.



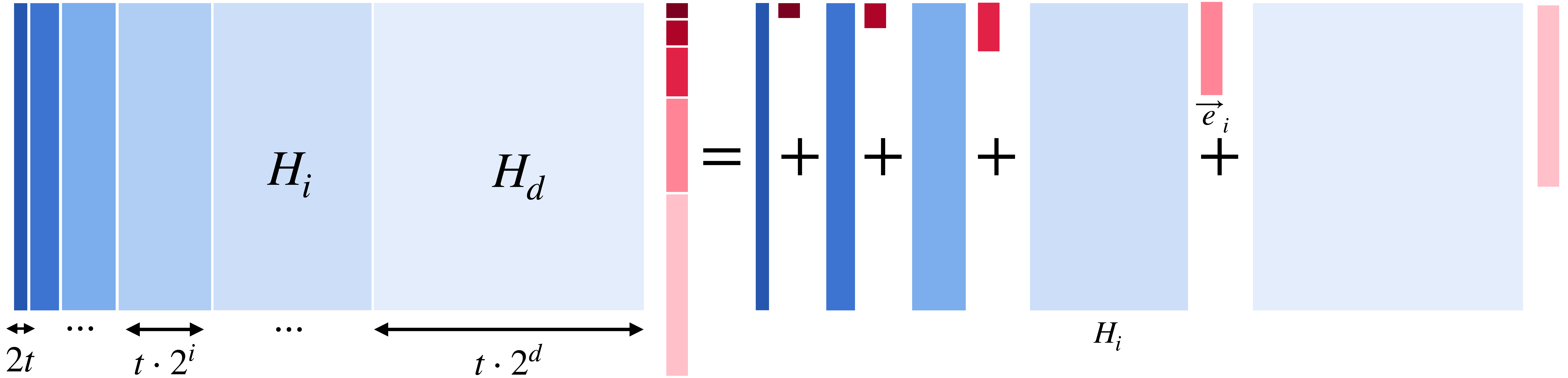
Security of Variable-Density LPN - Linear Tests



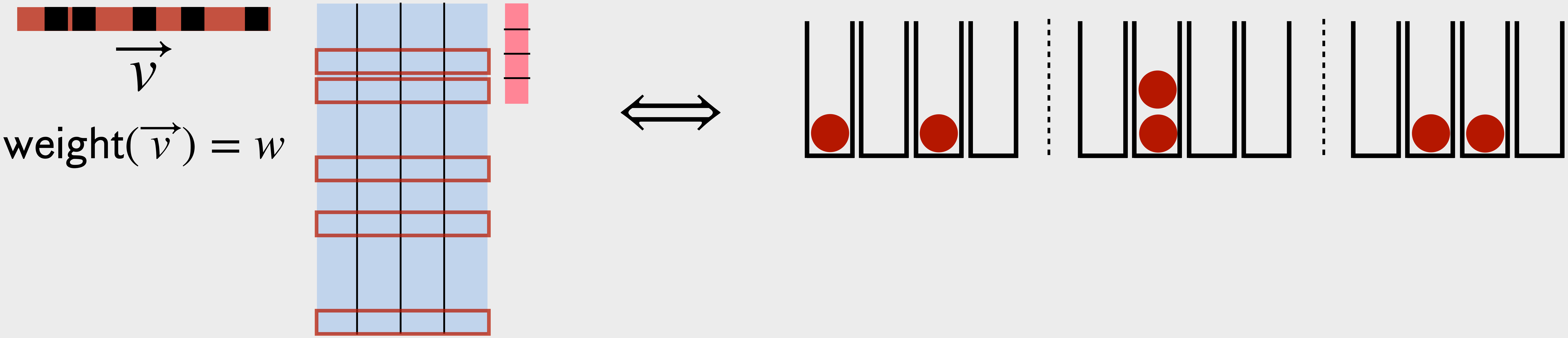
Claim: w.h.p. over the choice of H_i, the distribution $\vec{v} \cdot (H_i \cdot \vec{e}_i)$ has bias $2^{-O(t)}$ for *all* $\vec{v}$ of Hamming weight in $[2^{i-1}, 2^i]$.



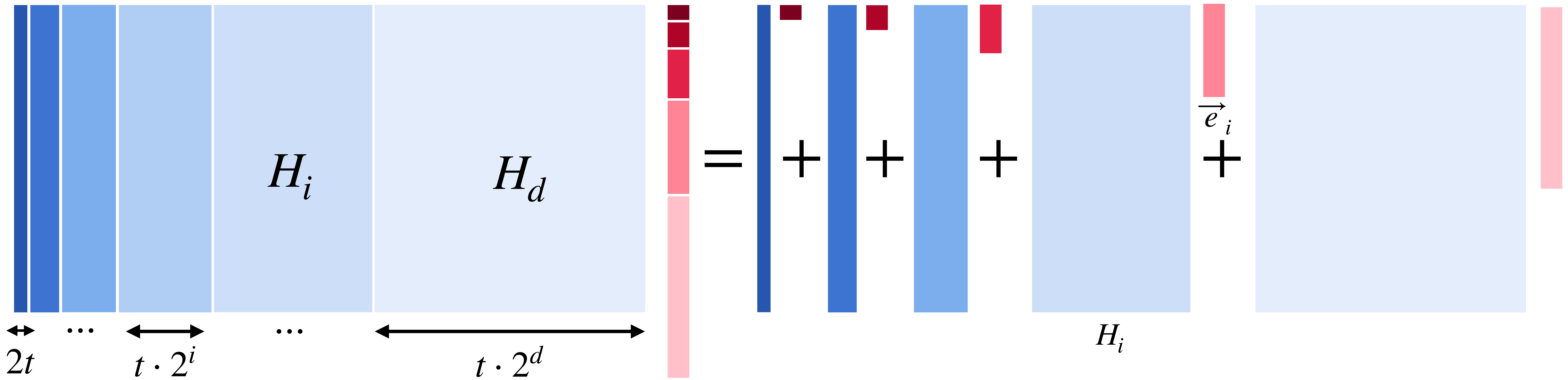
Security of Variable-Density LPN - Linear Tests



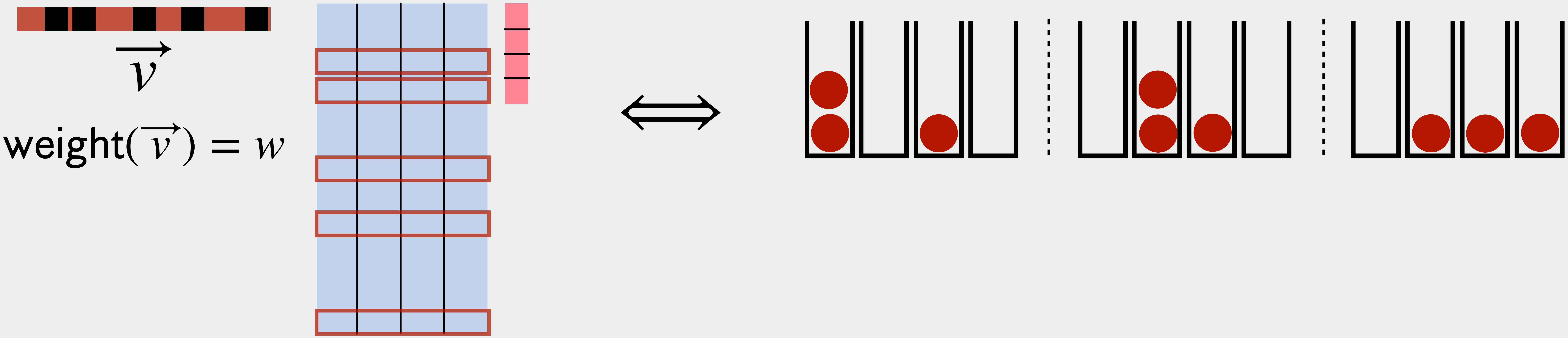
Claim: w.h.p. over the choice of H_i, the distribution $\vec{v} \cdot (H_i \cdot \vec{e}_i)$ has bias $2^{-O(t)}$ for *all* $\vec{v}$ of Hamming weight in $[2^{i-1}, 2^i]$.



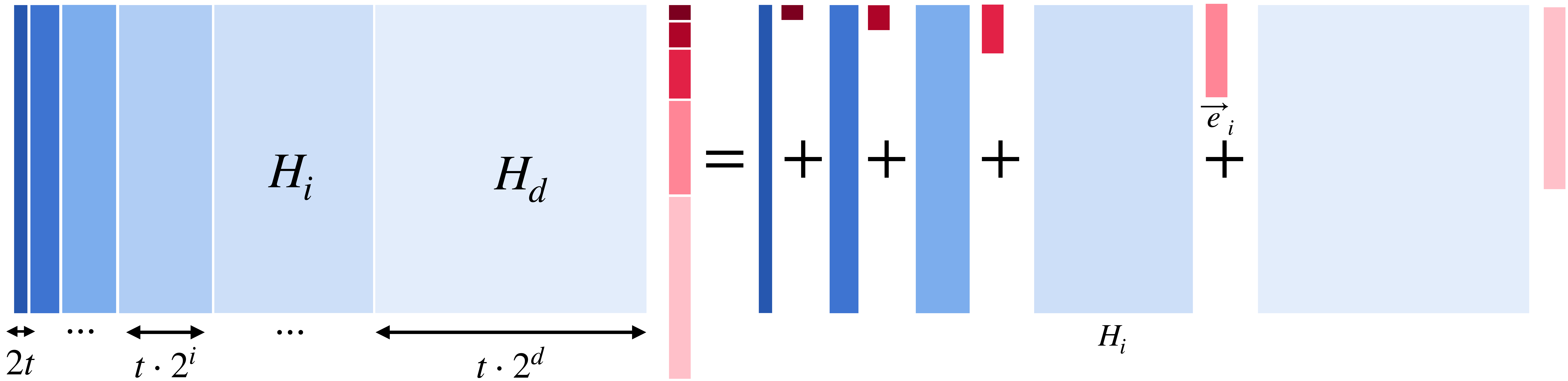
Security of Variable-Density LPN - Linear Tests



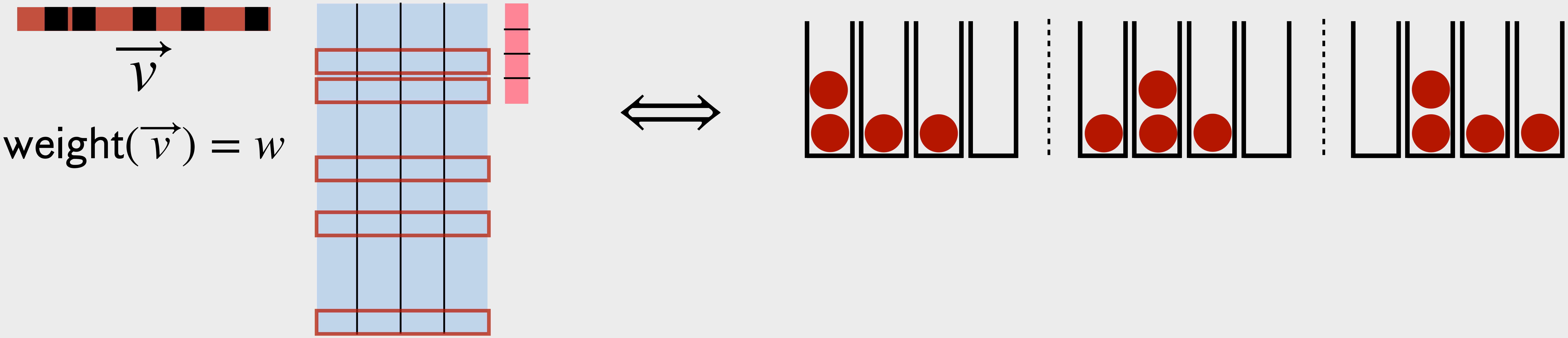
Claim: w.h.p. over the choice of H_i , the distribution $\vec{v} \cdot (H_i \cdot \vec{e}_i)$ has bias $2^{-O(t)}$ for *all* $\vec{v}$ of Hamming weight in $[2^{i-1}, 2^i]$.



Security of Variable-Density LPN - Linear Tests

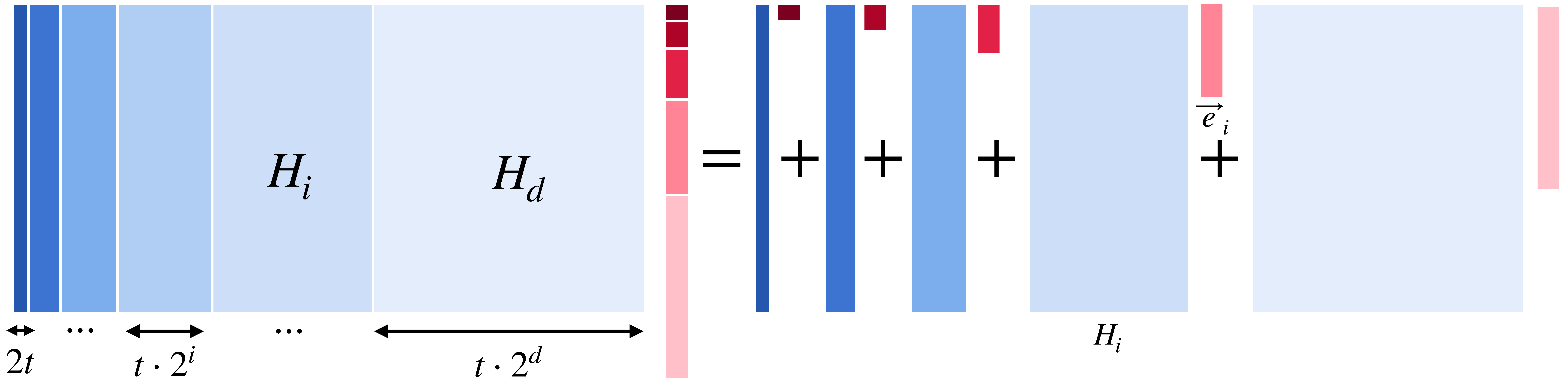


Claim: w.h.p. over the choice of H_i, the distribution $\vec{v} \cdot (H_i \cdot \vec{e}_i)$ has bias $2^{-O(t)}$ for *all* $\vec{v}$ of Hamming weight in $[2^{i-1}, 2^i]$.

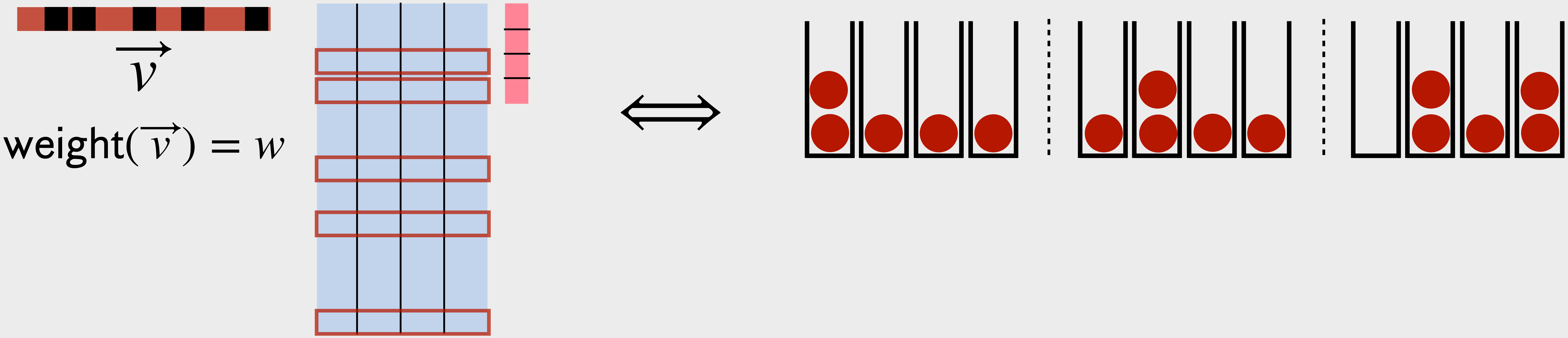


$\text{weight}(\vec{v}) = w$

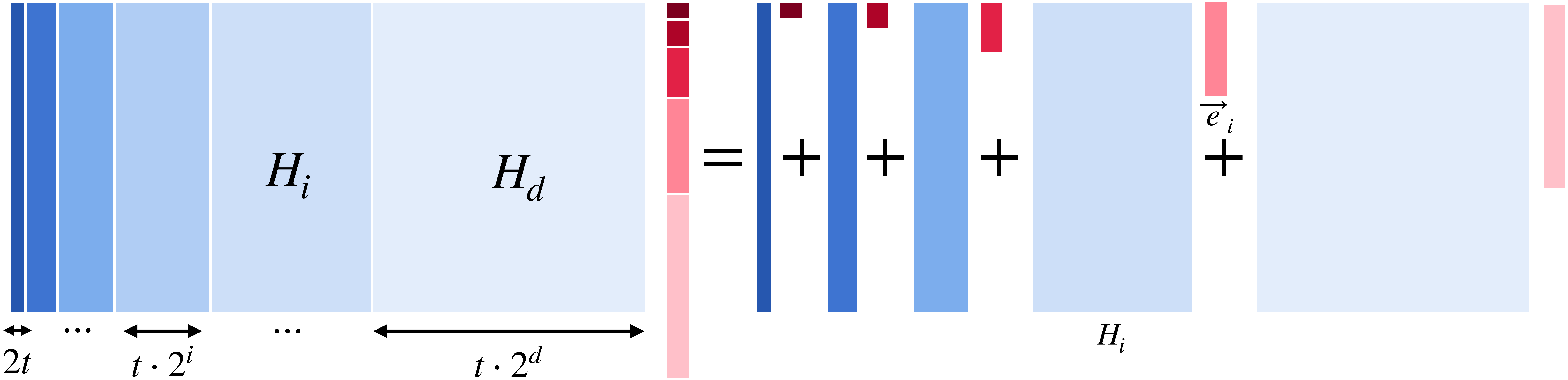
Security of Variable-Density LPN - Linear Tests



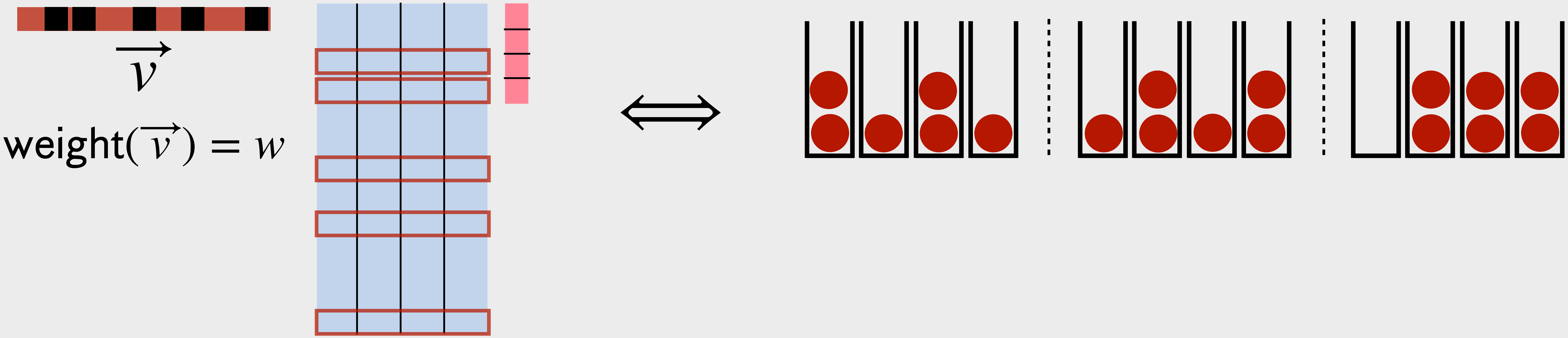
Claim: w.h.p. over the choice of H_i , the distribution $\vec{v} \cdot (H_i \cdot \vec{e}_i)$ has bias $2^{-O(t)}$ for *all* $\vec{v}$ of Hamming weight in $[2^{i-1}, 2^i]$.



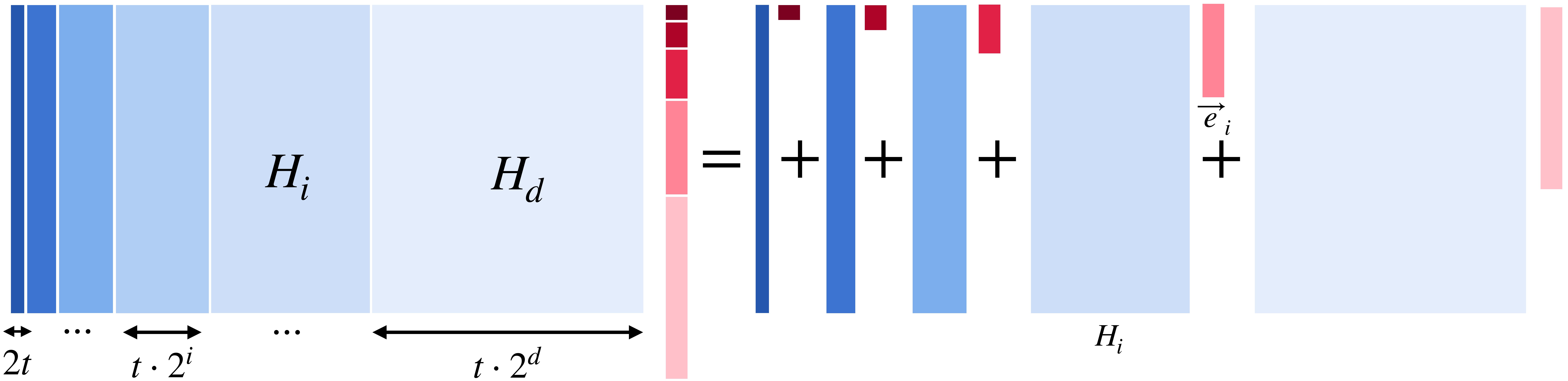
Security of Variable-Density LPN - Linear Tests



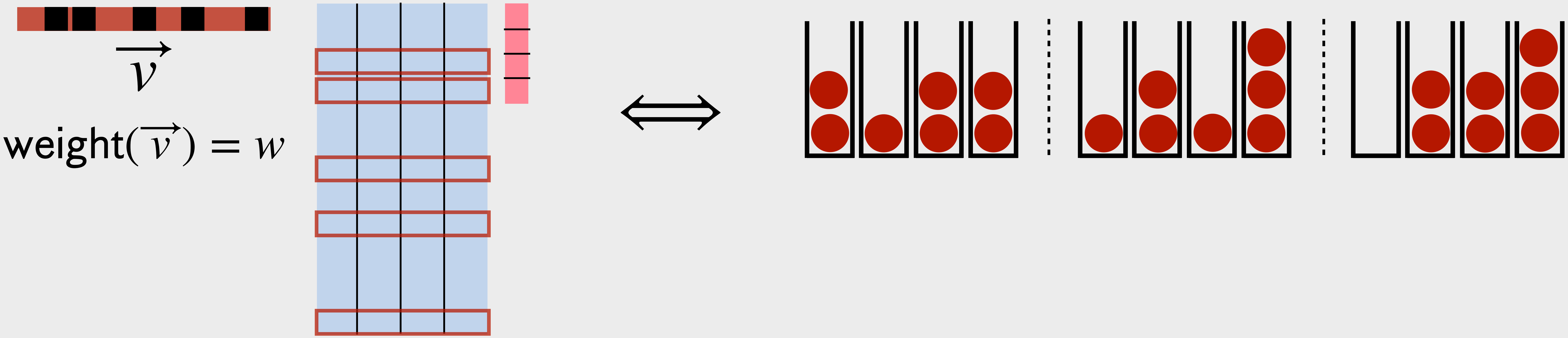
Claim: w.h.p. over the choice of H_i, the distribution $\vec{v} \cdot (H_i \cdot \vec{e}_i)$ has bias $2^{-O(t)}$ for *all* $\vec{v}$ of Hamming weight in $[2^{i-1}, 2^i]$.



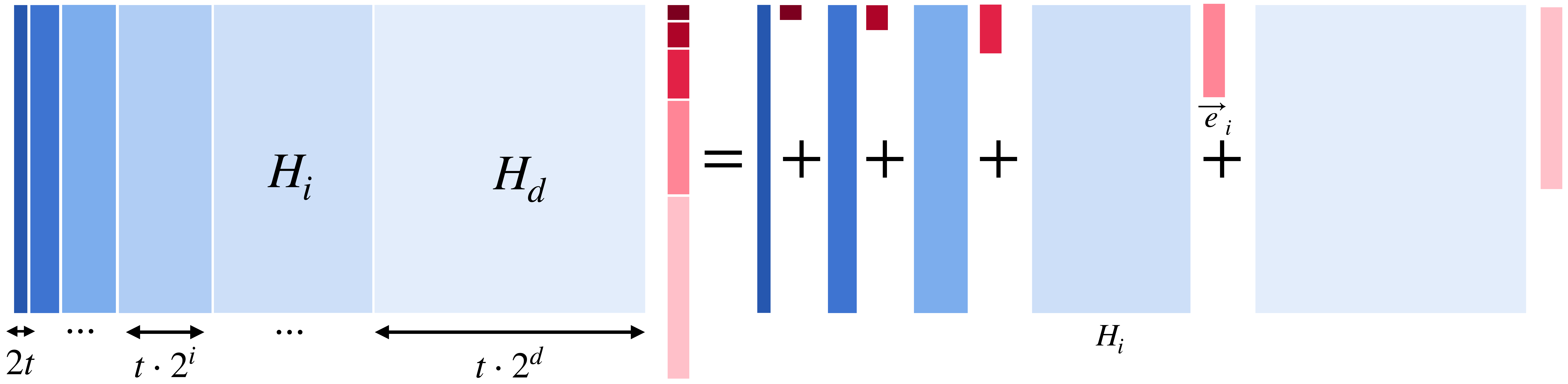
Security of Variable-Density LPN - Linear Tests



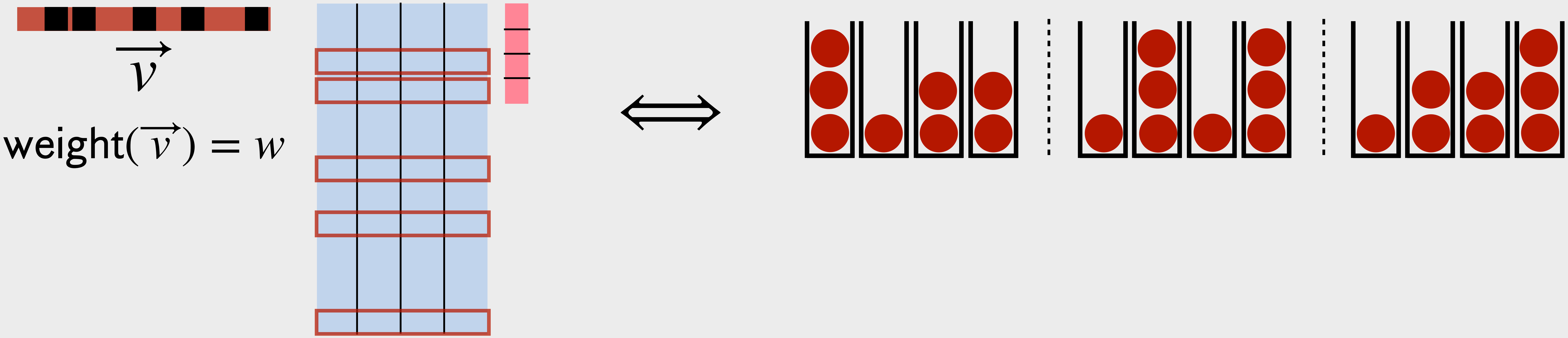
Claim: w.h.p. over the choice of H_i , the distribution $\vec{v} \cdot (H_i \cdot \vec{e}_i)$ has bias $2^{-O(t)}$ for *all* $\vec{v}$ of Hamming weight in $[2^{i-1},2^i]$.



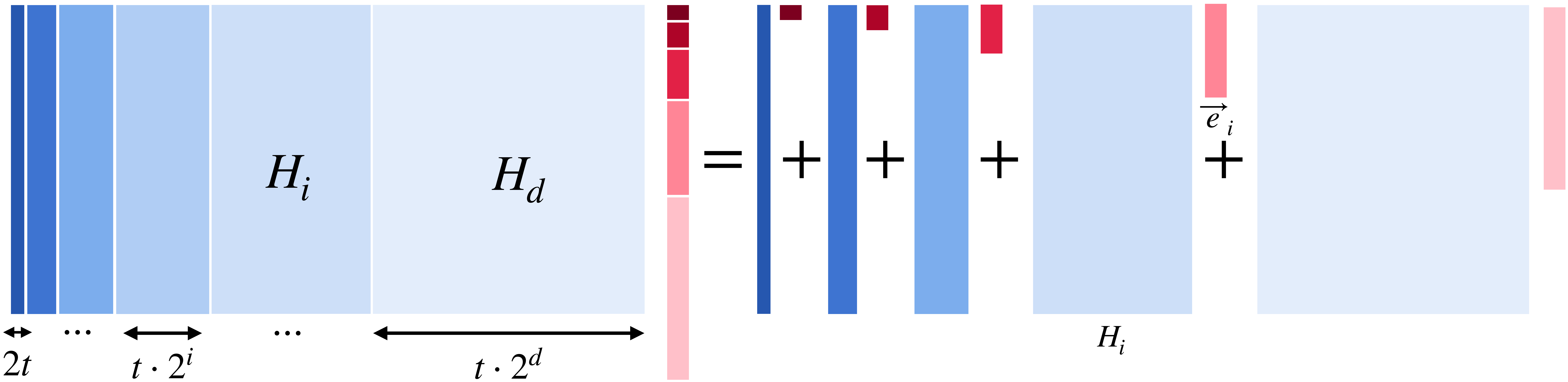
Security of Variable-Density LPN - Linear Tests



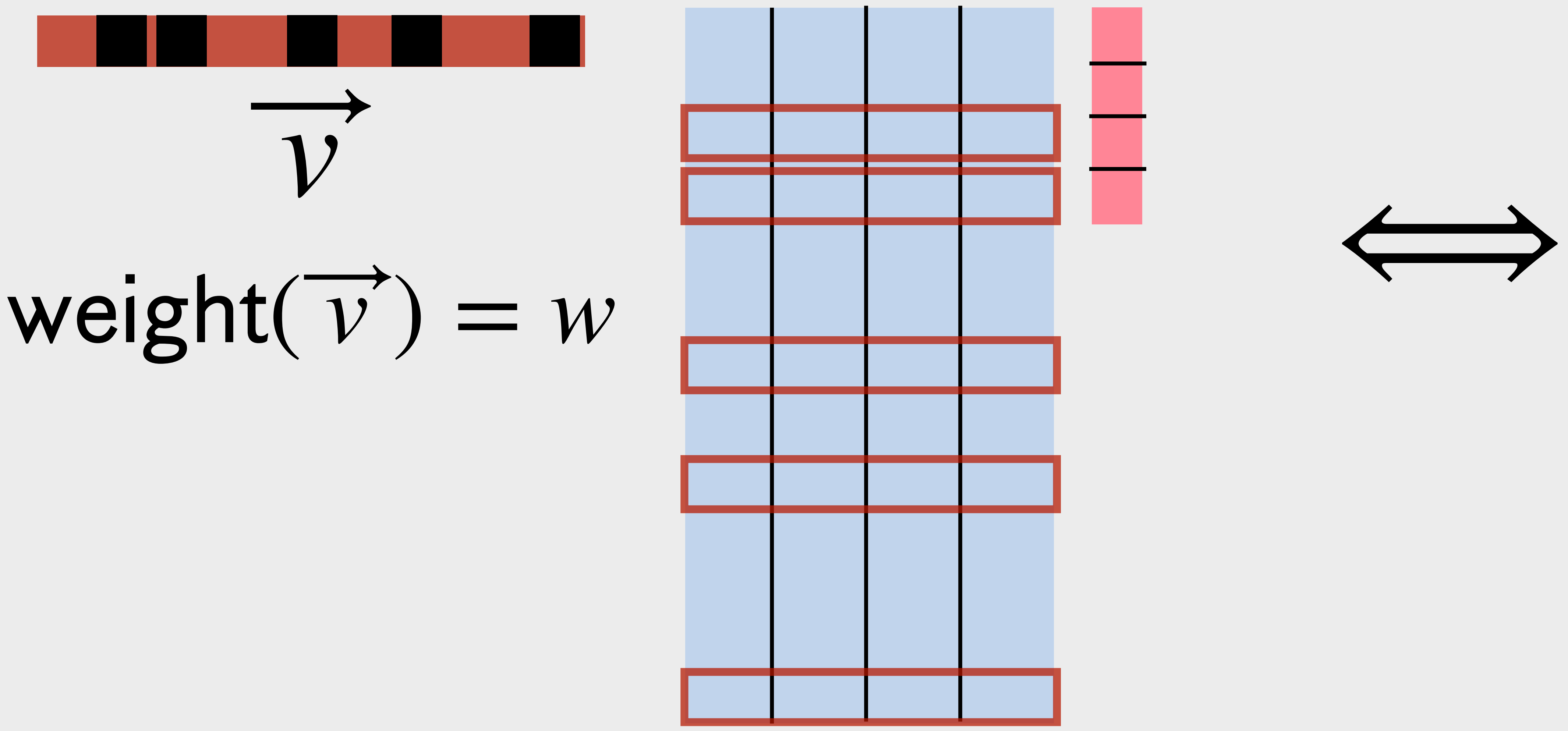
Claim: w.h.p. over the choice of H_i, the distribution $\vec{v} \cdot (H_i \cdot \vec{e}_i)$ has bias $2^{-O(t)}$ for *all* $\vec{v}$ of Hamming weight in $[2^{i-1}, 2^i]$.

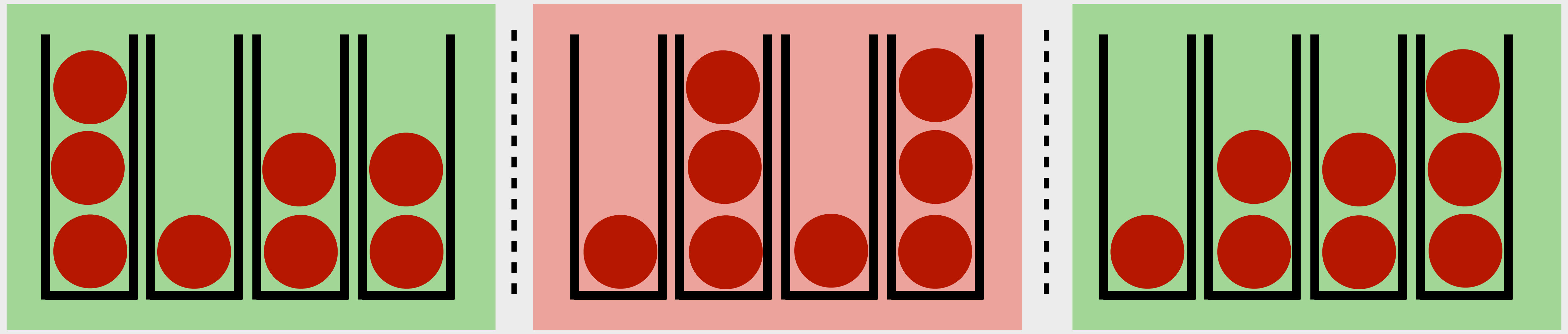


Security of Variable-Density LPN - Linear Tests



Claim: w.h.p. over the choice of H_i, the distribution $\vec{v} \cdot (H_i \cdot \vec{e}_i)$ has bias $2^{-O(t)}$ for *all* $\vec{v}$ of Hamming weight in $[2^{i-1}, 2^i]$.





After w rounds, a set of bins is *good* if the fraction of bins with an odd number of balls is in $[1/3, 2/3]$.

If $> w/2$ sets of bins are *good*, the bias of the distribution is $2^{-\Omega(t)}$.

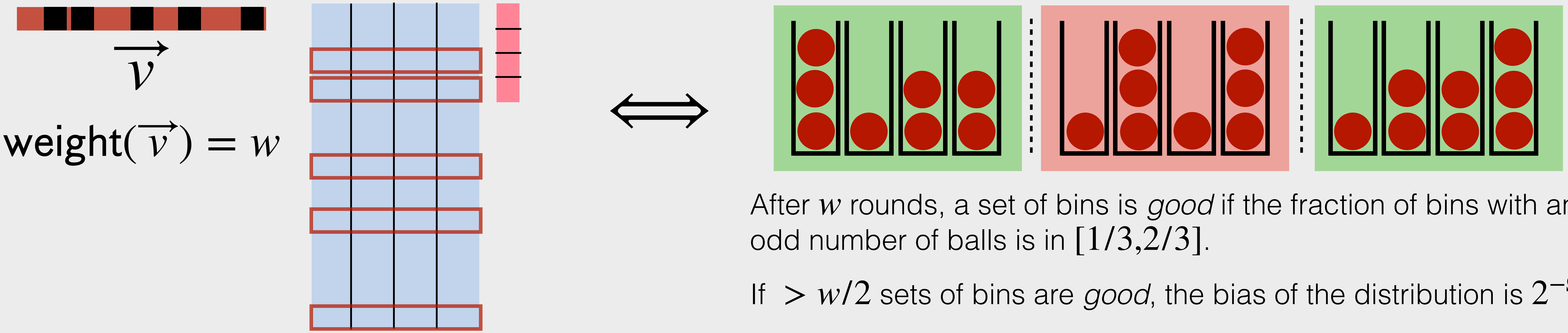
Security of Variable-Density LPN - Linear Tests

Using an Azuma-style concentration bound (McDiarmid's bounded difference inequality):

$$\Pr\left[\#\left\{\text{bad sets} > \frac{w}{2}\right\}\right] \leq \exp\left(-\Omega\left(t\cdot 2^i\right)\right)$$

- **Union bound**: $\sum_{\ell=2^{i-1}}^{2^i}\binom{2^d}{\ell}\cdot\exp(-\Omega(t\cdot 2^i))$ is $2^{-\Omega(t)}$ if $t > 100\cdot d$.
- Bias of $\vec{v}\cdot(H\cdot\vec{e}) = \vec{v}\cdot\left(\bigoplus_i H_i\cdot\vec{e}_i\right) \leq$ bias of $\vec{v}\cdot(H_i\cdot\vec{e}_i)$ $\forall i$

Claim: w.h.p. over the choice of H_i, the distribution $\vec{v}\cdot(H_i\cdot\vec{e}_i)$ has bias $2^{-O(t)}$ for *all* $\vec{v}$ of Hamming weight in $[2^{i-1},2^i]$.



After w rounds, a set of bins is *good* if the fraction of bins with an odd number of balls is in $[1/3,2/3]$.

If $> w/2$ sets of bins are *good*, the bias of the distribution is $2^{-\Omega(t)}$.

Security of Variable-Density LPN - Linear Tests

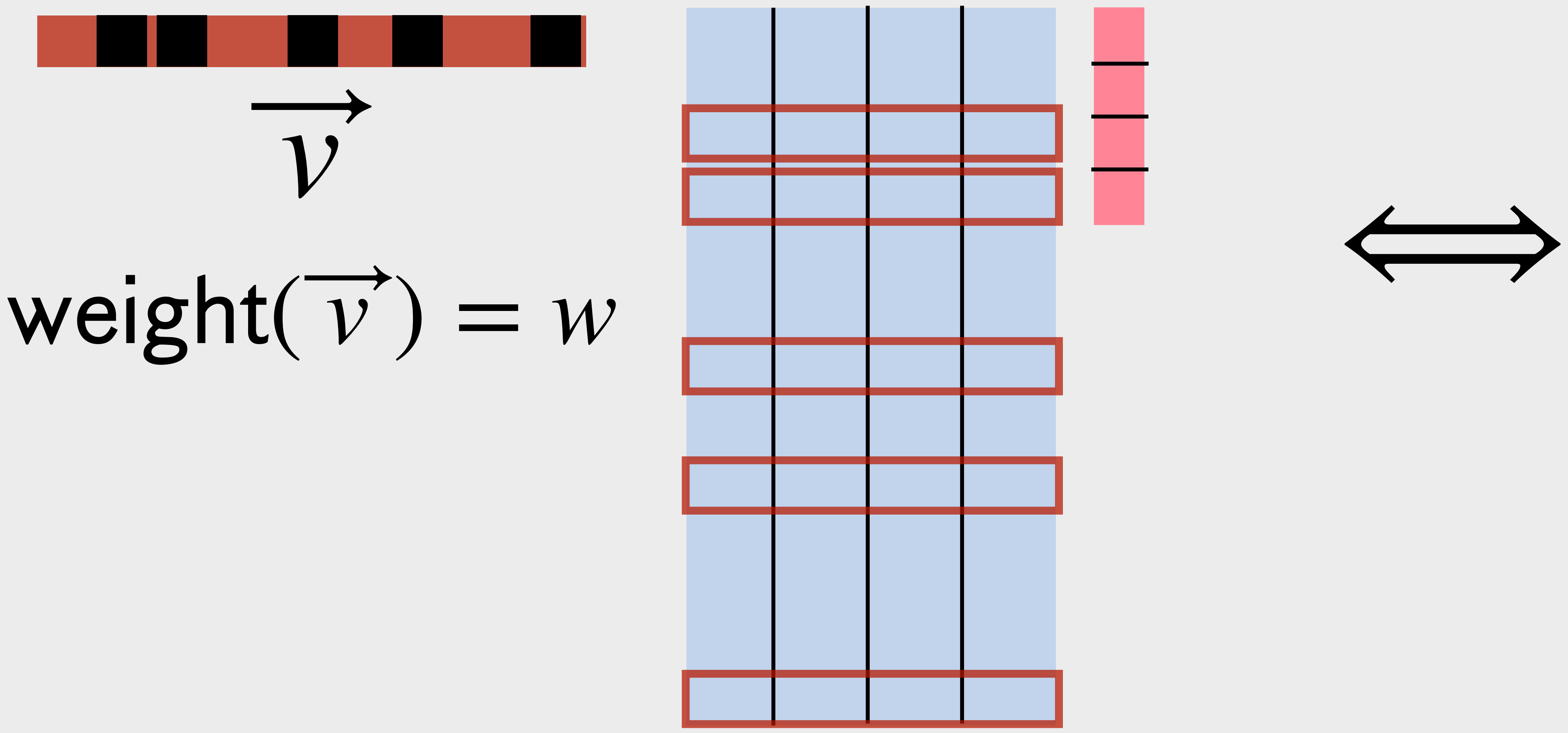
Using an Azuma-style concentration bound (McDiarmid's bounded difference inequality):

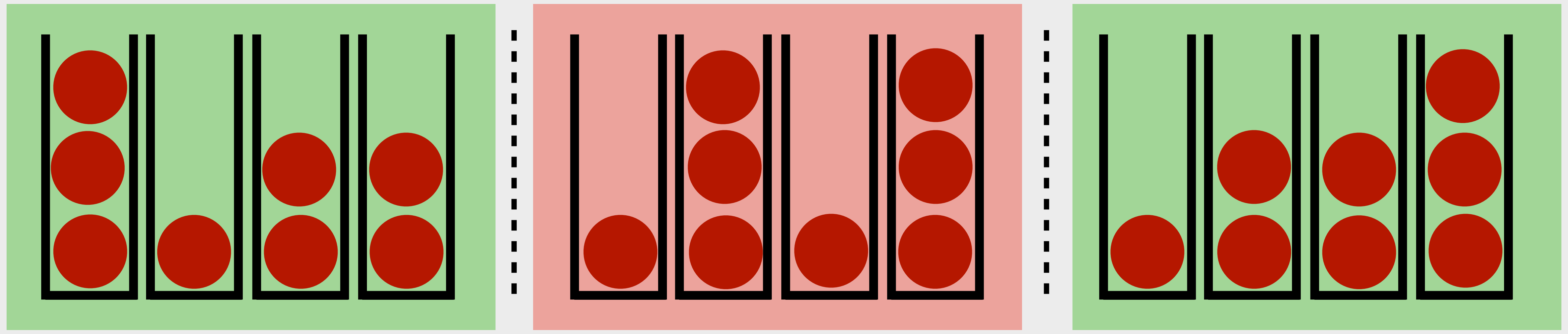
$$\Pr\left[\#\left\{\text{bad sets} > \frac{w}{2}\right\}\right] \leq \exp\left(-\Omega\left(t \cdot 2^i\right)\right)$$

- **Union bound**: $\sum_{\ell=2^{i-1}}^{2^i} \binom{2^d}{\ell} \cdot \exp(-\Omega(t \cdot 2^i))$ is $2^{-\Omega(t)}$ if $t > 100 \cdot d$.
- Bias of $\overrightarrow{v} \cdot (H \cdot \overrightarrow{e}) = \overrightarrow{v} \cdot \left(\bigoplus_i H_i \cdot \overrightarrow{e}_i\right) \leq$ bias of $\overrightarrow{v} \cdot (H_i \cdot \overrightarrow{e}_i)$ $\forall i$

Conclusion: with probability at least $1 - 2^{-\Omega(t)}$ over the choice of H, the bias of the distribution with respect to *any* test vector $\overrightarrow{v}$ is at most $2^{-\Omega(t)}$.

Claim: w.h.p. over the choice of H_i, the distribution $\overrightarrow{v} \cdot (H_i \cdot \overrightarrow{e}_i)$ has bias $2^{-O(t)}$ for *all* $\overrightarrow{v}$ of Hamming weight in $[2^{i-1}, 2^i]$.





After w rounds, a set of bins is *good* if the fraction of bins with an odd number of balls is in $[1/3, 2/3]$.

If $> w/2$ sets of bins are *good*, the bias of the distribution is $2^{-\Omega(t)}$.

Security of Variable-Density LPN - Algebraic Attacks

Recall the alternative formulation of the candidate: $F_K(x) = \bigoplus_{i=1}^{d} \bigoplus_{j=1}^{t} \bigwedge_{\ell=1}^{i} \left(x_{i,j,\ell} \oplus K_{i,j,\ell} \right) = F(x \oplus K)$, where $F(x) = \bigoplus_{i=1}^{d} \bigoplus_{j=1}^{t} \bigwedge_{\ell=1}^{i} x_{i,j,\ell}$.

Security of Variable-Density LPN - Algebraic Attacks

Recall the alternative formulation of the candidate: $F_K(x) = \bigoplus_{i=1}^{d} \bigoplus_{j=1}^{t} \bigwedge_{\ell=1}^{i} \left(x_{i,j,\ell} \oplus K_{i,j,\ell}\right) = F(x \oplus K)$, where $F(x) = \bigoplus_{i=1}^{d} \bigoplus_{j=1}^{t} \bigwedge_{\ell=1}^{i} x_{i,j,\ell}$.

Resistance against algebraic attacks:

Algebraic attacks: find low-degree polynomials (p, q) such that for any x, $F_K(x) \cdot q(x) = p(x)$. Then the WPRF candidate can be broken using $\sim |x|^m$ samples, where $m \geq \deg(p), \deg(q)$.

Note: the only previous candidate WPRF in $\mathsf{AC}^0[\oplus]$ of [ABGKR15] was broken (in quasi-polynomial time) in [BR17], using an algebraic attack.

Claim: for any K, the *rational degree* $m = \min_{r \neq 0}\{\deg(r) \mid F_K \cdot r = 0 \ \vee \ (F_k \oplus 1) \cdot r = 0\}$ of F_K is at least d.

Follows from known results [MJSC16]: F is a direct sum of triangular functions, where the d-th triangular function is given by $T_d(x) = x_1 \oplus x_2x_3 \oplus \cdots \oplus \bigwedge_{k=d'-d}^{d'} x_k$, where $d' = d(d-1)/2$. The d-th triangular function has rational degree d, and a direct sum of functions has rational degree lower bounded by the largest rational degree.

Security of Variable-Density LPN - and Many More!

Recall the alternative formulation of the candidate: $F_K(x) = \bigoplus_{i=1}^{d} \bigoplus_{j=1}^{t} \bigwedge_{\ell=1}^{i} \left(x_{i,j,\ell} \oplus K_{i,j,\ell} \right) = F(x \oplus K)$, where $F(x) = \bigoplus_{i=1}^{d} \bigoplus_{j=1}^{t} \bigwedge_{\ell=1}^{i} x_{i,j,\ell}$.

Security of Variable-Density LPN - and Many More!

Recall the alternative formulation of the candidate: $F_K(x) = \bigoplus_{i=1}^{d} \bigoplus_{j=1}^{t} \bigwedge_{\ell=1}^{i} \left(x_{i,j,\ell} \oplus K_{i,j,\ell} \right) = F(x \oplus K)$, where $F(x) = \bigoplus_{i=1}^{d} \bigoplus_{j=1}^{t} \bigwedge_{\ell=1}^{i} x_{i,j,\ell}$.

See the paper:

- AC^0 attackers
- Low-degree polynomial tests (generalization of the linear test framework)
- Statistical query algorithms (generalization of the Linial, Mansour, and Nisan algorithm)
- Linear cryptanalysis (as formalized by Miles and Viola [MV11])

All cannot break the new candidate.

Security of Variable-Density LPN - and Many More!

Recall the alternative formulation of the candidate: $F_K(x) = \bigoplus_{i=1}^{d} \bigoplus_{j=1}^{t} \bigwedge_{\ell=1}^{i} \left(x_{i,j,\ell} \oplus K_{i,j,\ell}\right) = F(x \oplus K)$, where $F(x) = \bigoplus_{i=1}^{d} \bigoplus_{j=1}^{t} \bigwedge_{\ell=1}^{i} x_{i,j,\ell}$.

See the paper:

- AC^0 attackers
- Low-degree polynomial tests (generalization of the linear test framework)
- Statistical query algorithms (generalization of the Linial, Mansour, and Nisan algorithm)
- Linear cryptanalysis (as formalized by Miles and Viola [MV11])

All cannot break the new candidate.

Can you break it?

Challenge: breaking the candidate using less than $2^{O(n^{1/3})}$ time and samples, with $d = t = O(n^{1/3})$. *Note:* the variant where $x_{i,j,\ell}$ is replaced by $x_{j,\ell}$ also resists the same attacks! (And the conjectured security bound becomes $2^{O(\sqrt{n})}$ (which is tight!)).

I can also provide *concrete* challenge parameters! E.g. $t = 150$, $d = 40$, 2^d samples.

Sample Applications

Of PCFs

Of the new WPRF

🟩 : applications to secure computation and zero-knowledge

🟦 : applications to learning theory

🟥 : other applications

Sample Applications

Of PCFs

- Secure computation with one-time, indefinitely reusable, short setup, for correlations such as OT, vector OLE over larger fields, (authenticated) Beaver triples, etc.
- Black-box 2-round secure 2-party computation, with fully-reusable preprocessing
- Preprocessing NIZKs with fully reusable preprocessing
- Homomorphic secret sharing for constant-degree polynomials
- Programmable PCFs (gives applications to N-party secure computation for $N > 2$)

🟩 : applications to secure computation and zero-knowledge

🟦 : applications to learning theory

🟥 : other applications

Of the new WPRF

Sample Applications

Of PCFs

- Secure computation with one-time, indefinitely reusable, short setup, for correlations such as OT, vector OLE over larger fields, (authenticated) Beaver triples, etc.
- Black-box 2-round secure 2-party computation, with fully-reusable preprocessing
- Preprocessing NIZKs with fully reusable preprocessing
- Homomorphic secret sharing for constant-degree polynomials
- Programmable PCFs (gives applications to N-party secure computation for $N > 2$)

Of the new WPRF

Assuming VD-LPN,

- XNF are subexponentially hard to learn under the uniform distribution
- Sparse polynomials are subexponentially hard to learn under *some* (artificial) distribution

  $\Longrightarrow$ upcoming improvements!

🟩 : applications to secure computation and zero-knowledge

🟦 : applications to learning theory

🟥 : other applications

Sample Applications

Of PCFs

- Secure computation with one-time, indefinitely reusable, short setup, for correlations such as OT, vector OLE over larger fields, (authenticated) Beaver triples, etc.
- Black-box 2-round secure 2-party computation, with fully-reusable preprocessing
- Preprocessing NIZKs with fully reusable preprocessing
- Homomorphic secret sharing for constant-degree polynomials
- Programmable PCFs (gives applications to N-party secure computation for $N > 2$)

🟩 : applications to secure computation and zero-knowledge

🟦 : applications to learning theory

🟥 : other applications

Of the new WPRF

Assuming VD-LPN,

- XNF are subexponentially hard to learn under the uniform distribution
- Sparse polynomials are subexponentially hard to learn under *some* (artificial) distribution

  $\Longrightarrow$ upcoming improvements!

But also…

- Correlation-robust hash functions
- XOR-RKA secure PRGs and WPRFs (first candidate without multilinear maps)

Simply because getting access to random samples of the form $(x, F_K(x))$ and $(x, F_{K\oplus\Delta}(x))$ for an offset Δ does not help: $F_{K\oplus\Delta}(x) = F_K(x \oplus \Delta)$, and $x \oplus \Delta$ is randomly distributed.

Thank You for Your Attention!

Questions?