MPC with Silent Preprocessing via Pseudorandom Correlation Generators

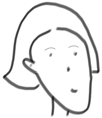
Lisa Kohl

Technion
Israel Institute of Technology

Based on joint works with Elette Boyle, Geoffroy Couteau, Niv Gilboa, Yuval Ishai, Peter Rindal, and Peter Scholl

Secure multi-party computation (MPC)

[Yao86; GMW87; BGW88; CCD88]



public function f

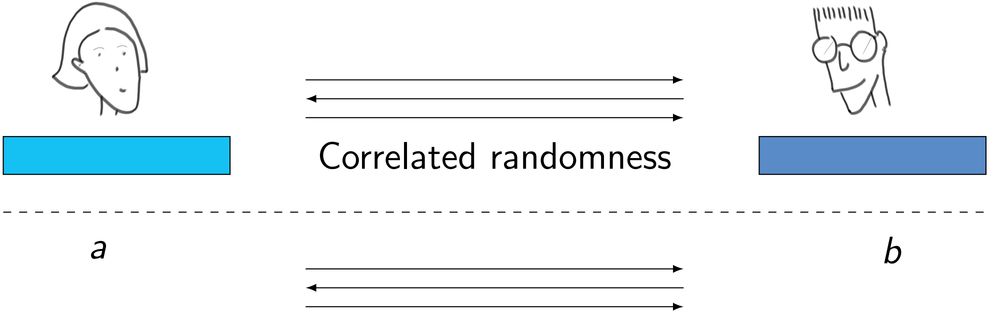
a

b

Goal: Parties learn $f(a,b)$ and *nothing more*

Secure MPC with preprocessing

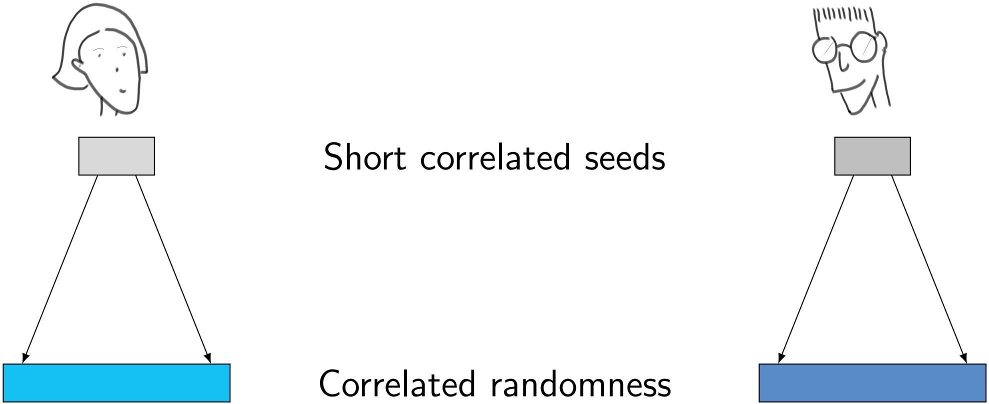
[Beaver91]



+ Fast online phase, security against dishonest majority
– Preprocessing expensive (communication & storage)

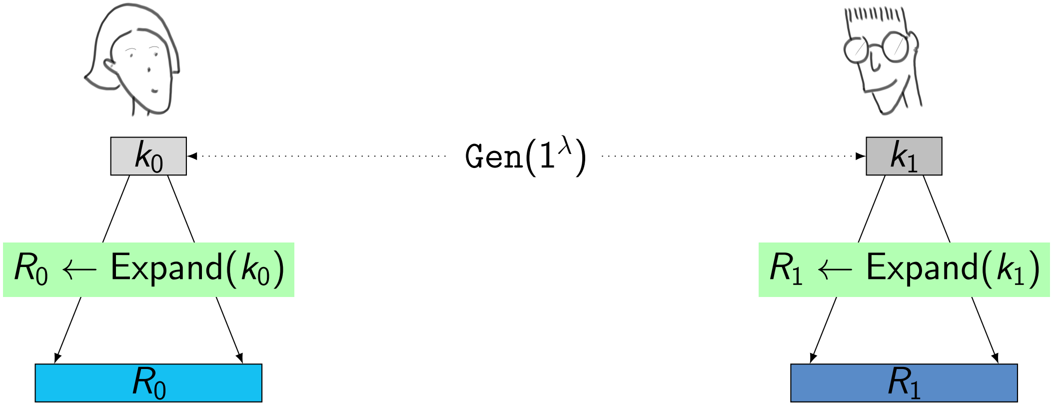
Pseudorandom correlation generator (PCG)

[BCGI18; BCGI**K**S19]



Pseudorandom correlation generator (PCG)

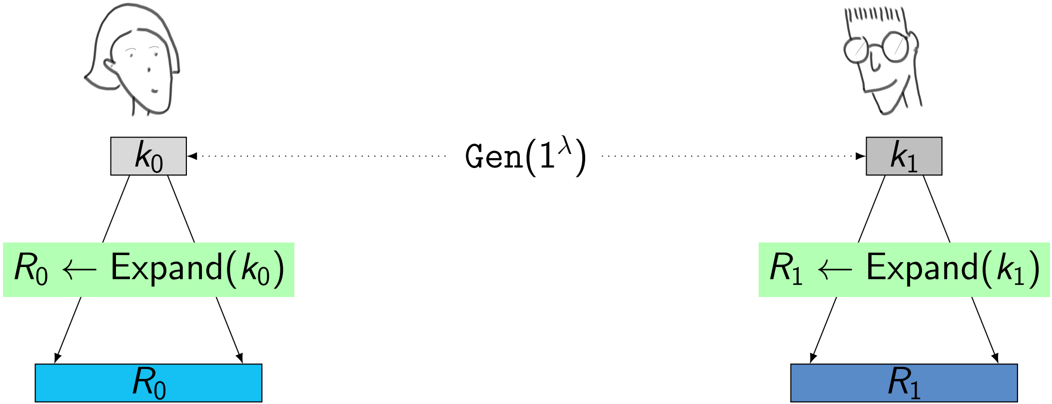
[BCGI18; BCGI**K**S19]



Correctness: $R_0 \sim R_1$

Pseudorandom correlation generator (PCG)

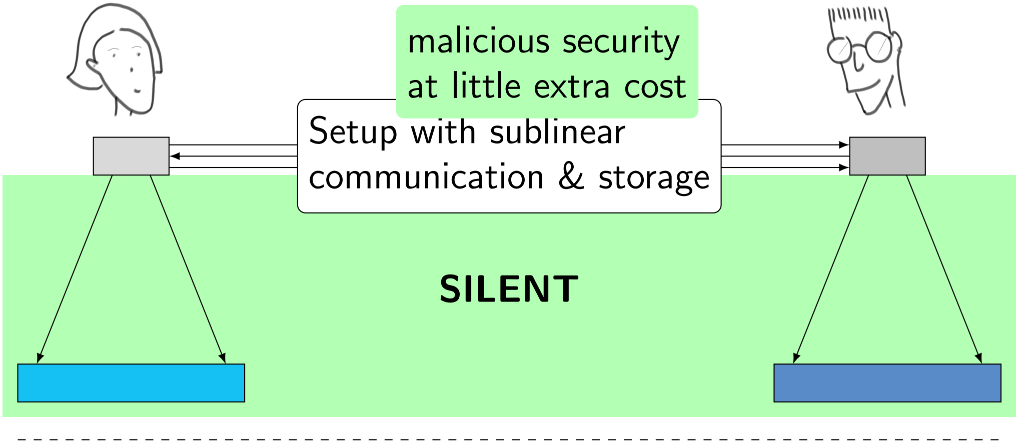
[BCGI18; BCGIKS19]



Security: $(k_0, R_1) \approx_c (k_0, [R_1 \mid R_0 \sim R_1])$

Secure MPC with *silent* preprocessing

[BCGIKS19]



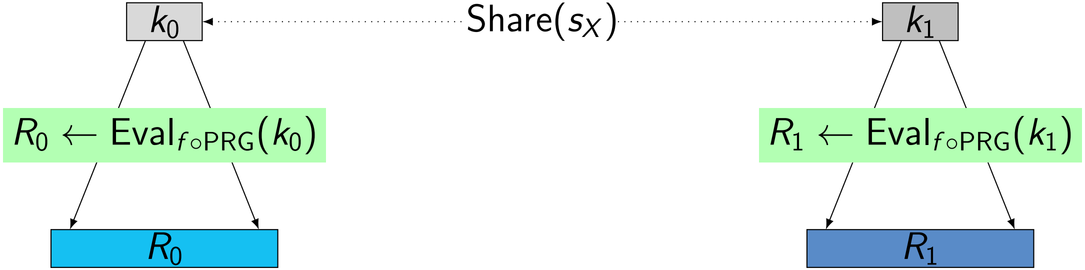
Generic construction of PCGs

[BCGI**K**S19]

General additive correlations:

R_0 R_1 $R_0 + R_1 = f(X)$

Feasibility: PRG + Homomorphic secret sharing



Landscape of PCGs

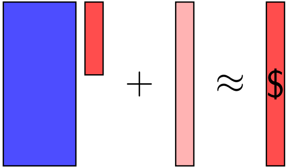
“Gentryland”	LWE+:	General additive [BCGI**K**S19]
“Cryptomania”	DDH + PRG*:	Log-space [BCGIO17]
	LWE + PRG*:	Bounded depth* [BCGI**K**S19]
“Lapland”	LPN:	Vector OLE [BCGI18]
		OT, Constant-degree [BCGI**K**S19]
	Ring-LPN:	OLE [BCGI**K**S20]
“Minicrypt”	OWF:	Linear [GI99; CDI05]
		Truth tables [BCGI**K**S19]

*low-degree

*concretely efficient

Learning with errors vs. learning parity with noise

LWE:

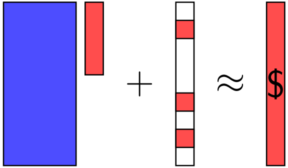


$p > 2$

s over $\mathbb{Z}_p$

$\|e\|_\infty$ small

LPN:

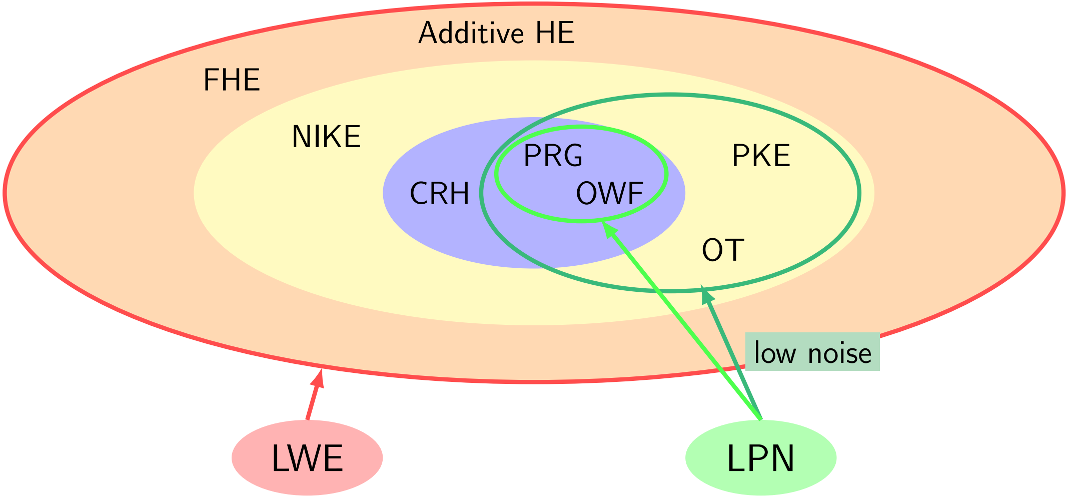


$p = 2$ (*here:* $p \geq 2$)

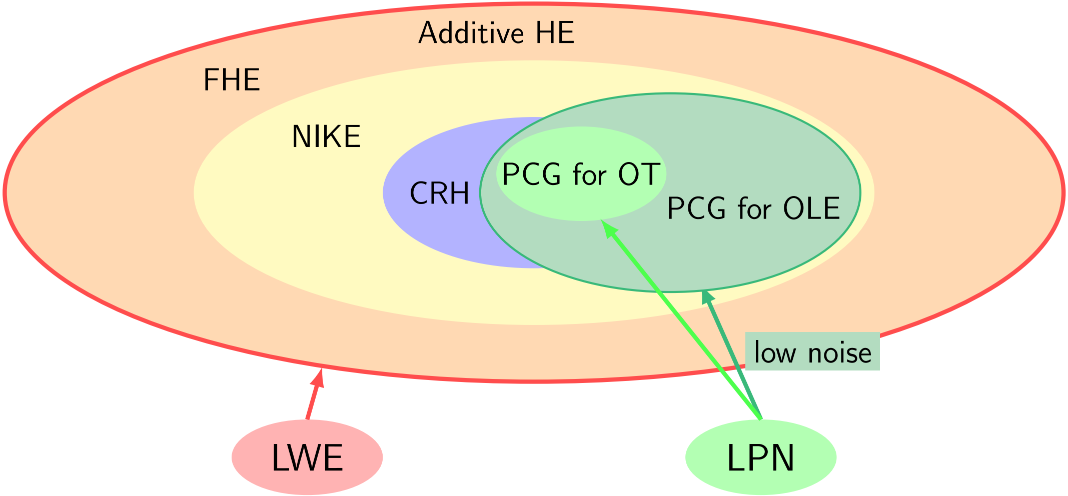
s over $\mathbb{Z}_p$

$\mathrm{HW}(e)$ small

Cryptography from LWE vs. LPN

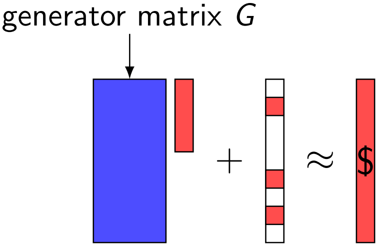


Cryptography from LWE vs. LPN

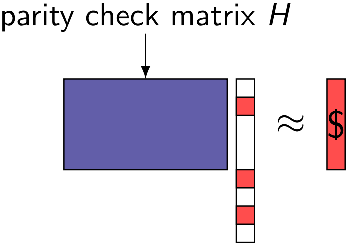


A simple PRG from LPN

LPN:



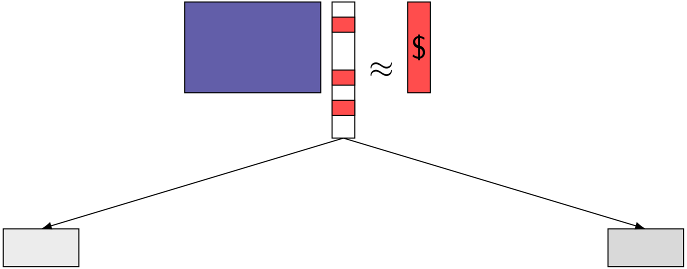
Dual-LPN:



limited to quadratic stretch

arbitrary polynomial stretch

Why LPN is a perfect match for PCGs



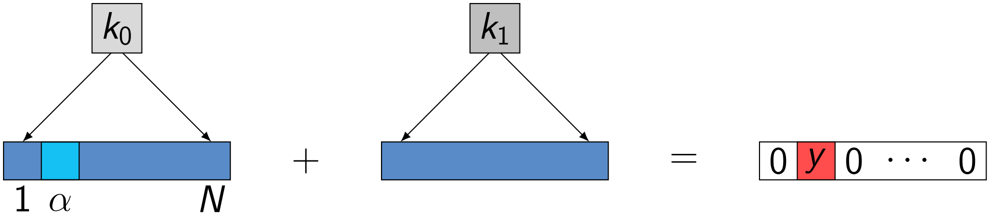
- Sparse vector can be distributed via compressed secret shares
- LPN assumption is linear $\rightsquigarrow$ homomorphic properties

How to distribute a sparse vector efficiently

[GI14]

Point Function: $\mathsf{F}^{\alpha}\colon \{1,\ldots,N\} \to \mathbb{F}_{2^\lambda}, \mathsf{F}^{\alpha}(x) = \begin{cases} y & \text{, if } x = \alpha \\ 0 & \text{, else} \end{cases}$

Distributed Point Function:

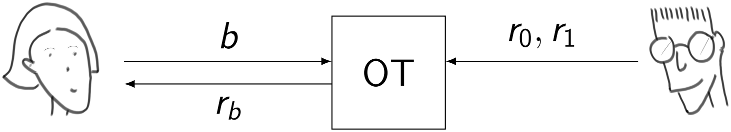


- Efficient constructions from OWFs [GI14; BGI16]
- Efficient distributed setup [Ds17]

Part I: PCG for oblivious transfer from LPN

Oblivious transfer (OT)

[Rab81; EGL85]

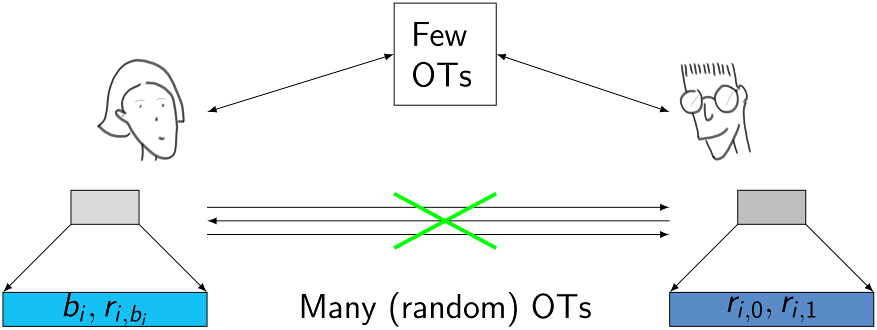


Security: Alice learns only r_b, Bob doesn't learn b

GMW Protocol: Secure MPC with 2 OTs per AND-Gate

Problem: OT is expensive ("public-key primitive")

OT extension



OT extension: Few base OTs + "cheap crypto" [Bea96; IKNP03]
Silent OT extension: Local expansion [BCGI**K**S19; BCGI**K**RS19]

Comparison of OT extension protocols

128-bit 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
[BCGI**K**S19]	$\log N$	$0 - 3$	✓	✗	LPN, $\text{crh}^{\star\star}$
[BCGI**K**RS19]	$2^{\star}$	0.1	✓	✓	LPN, $\text{crh}^{\star\star}$

*Fiat-Shamir for active security, **correlated-input secure hash function

[GMMM18]: RO $\not\Rightarrow$ 2-round OT extension

Comparison of OT extension protocols

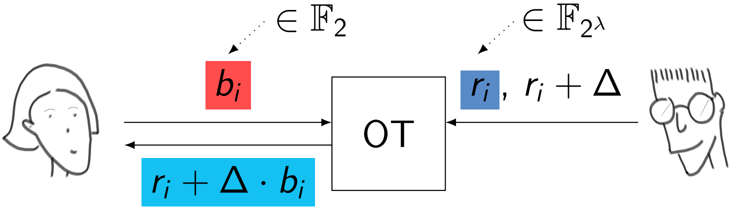
128-bit security

Reference	Rounds	Comm. per random OT	Silent	Active	Based on
[Bea96]	2	poly	✗	✗	OWF
[IKNP03; ALSZ13; KOS15]	3*	128	✗	✓	crh
[KK13] (short strings)	3	≈ 78	✗	✗	crh
[BCGI**K**S19]	$\log N$	$0-3$	✓	✗	LPN, crh**
[BCGI**K**RS19]	2*	0.1	✓	✓	LPN, crh**

*Fiat-Shamir for active security, **correlated-input secure hash function

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

Correlated OT



Correlated OT + correlation robust hash function ⇒ OT [IKNP03]

As vectors: $\triangleq$ Subfield vector oblivious linear evaluation

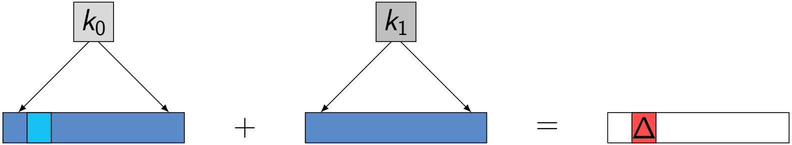
$$\Delta \cdot \mathbf{b} = \mathbf{r} + \Delta \cdot \mathbf{b} + \mathbf{r}$$

Overview: PCG for correlated OT

[BCGI**K**S19]

Idea:

1. Via distributed point functions:



2. Via addition:

3. Via LPN:

1a. Towards 2-round setup

[SGRR19; BCGI**K**RS19]

Problem: DPF require $\log N$ rounds for distributed setup!

Observation:

- Receiver knows **b**

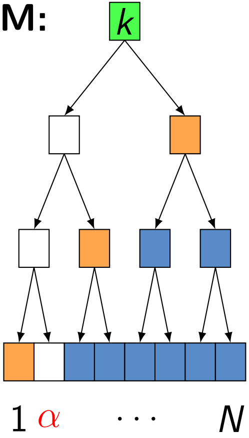
$\rightsquigarrow$ Receiver knows the point α, where PF $\neq 0$

$\rightsquigarrow$ Puncturable pseudorandom functions sufficient!

1b. Puncturable pseudorandom function

[BGI13; BW13; KPTZ13]

Via GGM:



$|k| = \lambda$, $|k^\star| = \lambda \log N$

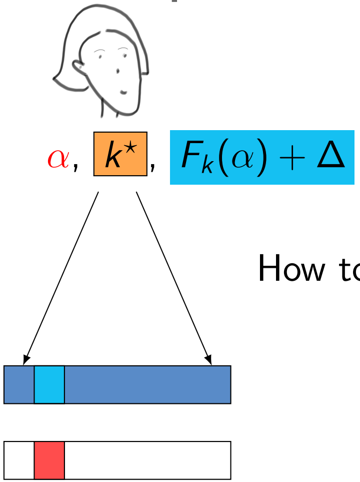
Puncturable PRF (PPRF):

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

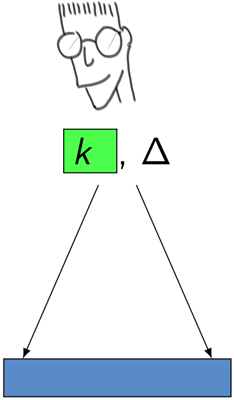
- k $\rightsquigarrow$ $F_k(x)$ for all x
- $k^\star$ $\rightsquigarrow$ $F_k(x)$ for all $x \neq \alpha$

1c. PCG for unit vector via PPRF

[SGRR19; BCGI**K**RS19]

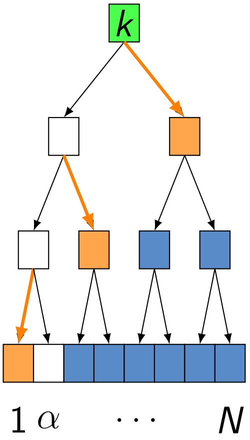


How to set up $k^\star, k$?



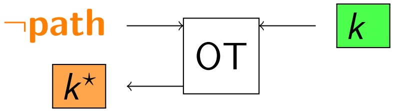
1d. 2-Round setup for unit vector

[SGRR19; BCGI**K**RS19]



Strategy: (based on [Ds17])

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

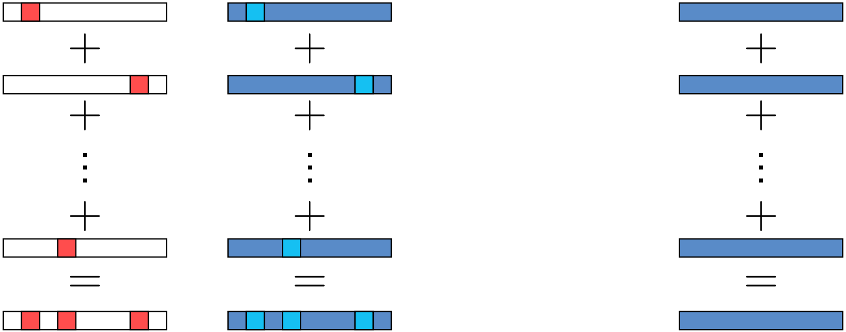


- **Note:** OTs can be executed *in parallel!*

2. From unit to sparse vectors

[BCGI18; BCGI**K**S19]

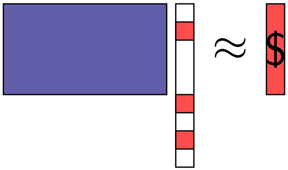
Repeat t times:



Alternative: Concatenation + LPN with *regular* noise

3. From sparse to pseudorandom vectors

[BCGI18; BCGIKS19]



Main challenge: Parity check matrix is big!

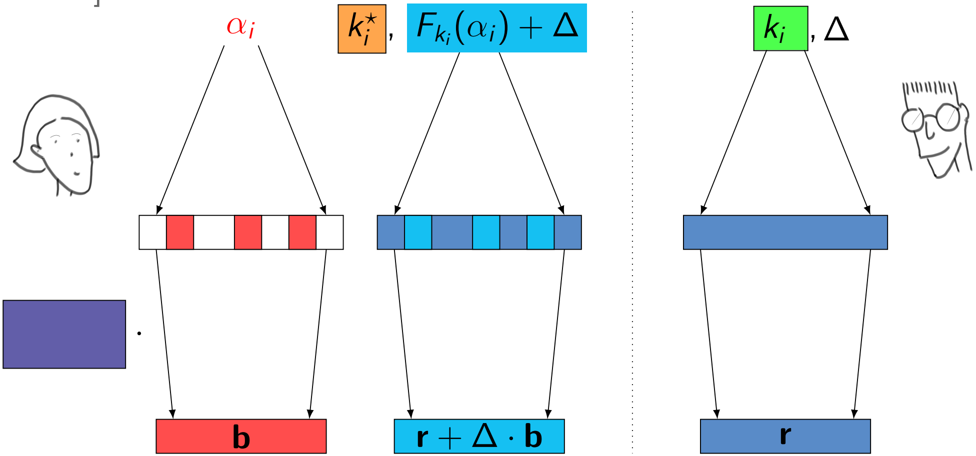
- use quasi-cyclic codes $\rightsquigarrow$ multiplication in $\widetilde{\mathcal{O}}(N)$

Security

- Similar to PQ cryptosystems BIKE, HQC [AAB+19; ABB+19]

PCG for correlated OT from LPN - Recap

[BCGIKS19]



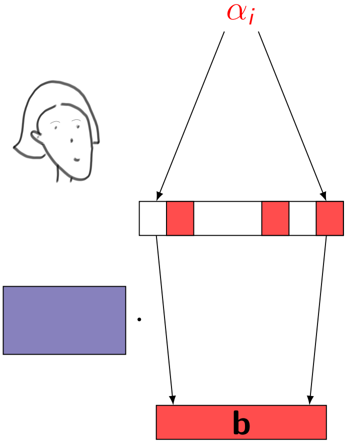
From correlated OT to chosen OT

1. **Break correlations:**
   - *Locally* apply crh [IKNP03]

⇝ MPC with 2-round silent preprocessing

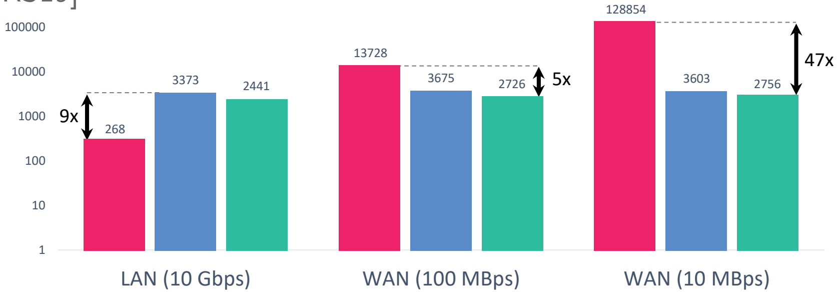
2. **Derandomization:**
   - Depends only on **b**
   - Can be sent along with first message

⇝ 2-round OT extension



Runtimes (ms) for 10 million random OTs

[BCGIKRS19]

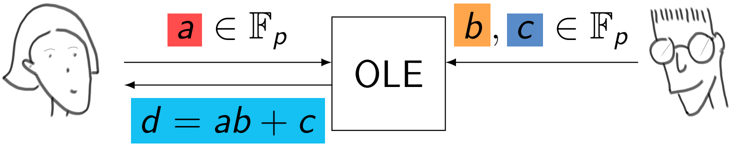


[IKNP03] vs 2-round silent vs 3-round hybrid

- Total communication: 160 MB vs 145 kB vs 127 kB

Part II: PCGs for OLE from LPN and ring-LPN

Oblivious linear evaluation (OLE)



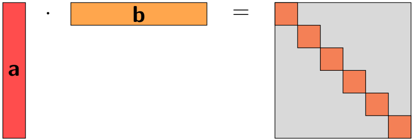
- Generalization of OT to $\mathbb{F}_p$
- 2 OLEs can be locally transformed into a multiplication triple

$$\mathbf{a} * \mathbf{b} = \mathbf{d} - \mathbf{c}$$

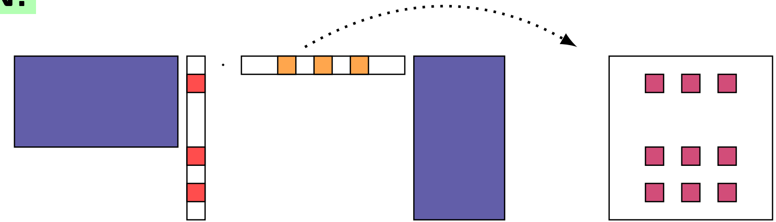
Towards PCG for OLE from LPN

[BCGIKS19,BCGIKS20]

Idea: Rewrite $\mathbf{a} * \mathbf{b}$ and use linearity of LPN



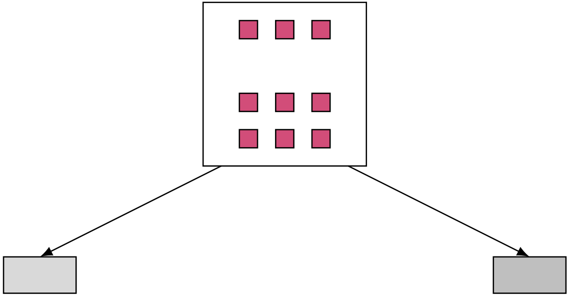
Via LPN:



PCG for OLE via LPN

[BCGI**K**S19,BCGI**K**S20]

Via DPF:



Problem: Dimension ($\rightsquigarrow$ computational cost) *quadratic* in N

A different perspective

[BCGIKS20]

Observations:

- Generalizes to more dimensions
- Better efficiency via choosing H such that $H * H$ *compressible*

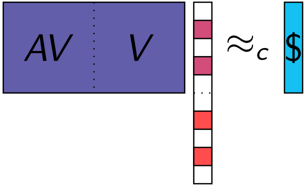
More efficient PCG for OLE from ring-LPN

[BCGIKS20]

public: $a(X)$; sparse: $s(X)$, $e(X)$

$$a(X) \cdot s(X) + e(X) \approx_c u(X) \text{ over } \mathbb{Z}_p[X]/\varphi(X)$$

If $\varphi(X)$ (of degree N) fully splits over $\mathbb{Z}_p[X]$:



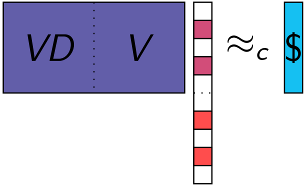
More efficient PCG for OLE from ring-LPN

[BCGI**K**S20]

public — sparse

$$a(X) \cdot s(X) + e(X) \approx_c u(X) \text{ over } \mathbb{Z}_p[X]/\varphi(X)$$

If $\varphi(X)$ (of degree N) fully splits over $\mathbb{Z}_p[X]$:



$\rightsquigarrow$ N OLEs over $\mathbb{Z}_p$ in $\widetilde{\mathcal{O}}(N)$ computation time

Efficiency of our PCG construction for OLE

[BCGI**K**S20]

To generate 1 Mio OLEs over $\mathbb{Z}_q$ (q composite of 62-bit primes):

Reference	Amount	Seed size	Communication	OLEs/second
[KPR18]	32 MB	32 MB	$>$ 1 GB	30 K
[BCGI**K**S19]	17 GB	3 GB	6 GB	6 K*
[BCGI**K**S20]	32 MB	1.25 MB	7 MB	100 K*

*expansion only, estimated costs

- Setup with malicious security
- Generalizes to authenticated multiplication triples at $\approx \times 2$ cost!

Conclusion

PCGs for OT from LPN [BCGI**K**S19; BCGI**K**RS19]

- Random OT: *practical*, almost *zero communication*
- 2-Round OT extension (malicious security, implementation)

PCGs for OLE [BCGI**K**S20]

- More efficient instantiation based on *fully splittable ring-LPN*

Open problems/ Ongoing work:

- Optimize OT: Better codes
- Efficient PCGs for more correlations
- Better understanding of LPN-flavored assumptions

Thank you!