Pseudorandom Correlation Generators:

Secure Computation with Silent Preprocessing

Yuval Ishai

Technion

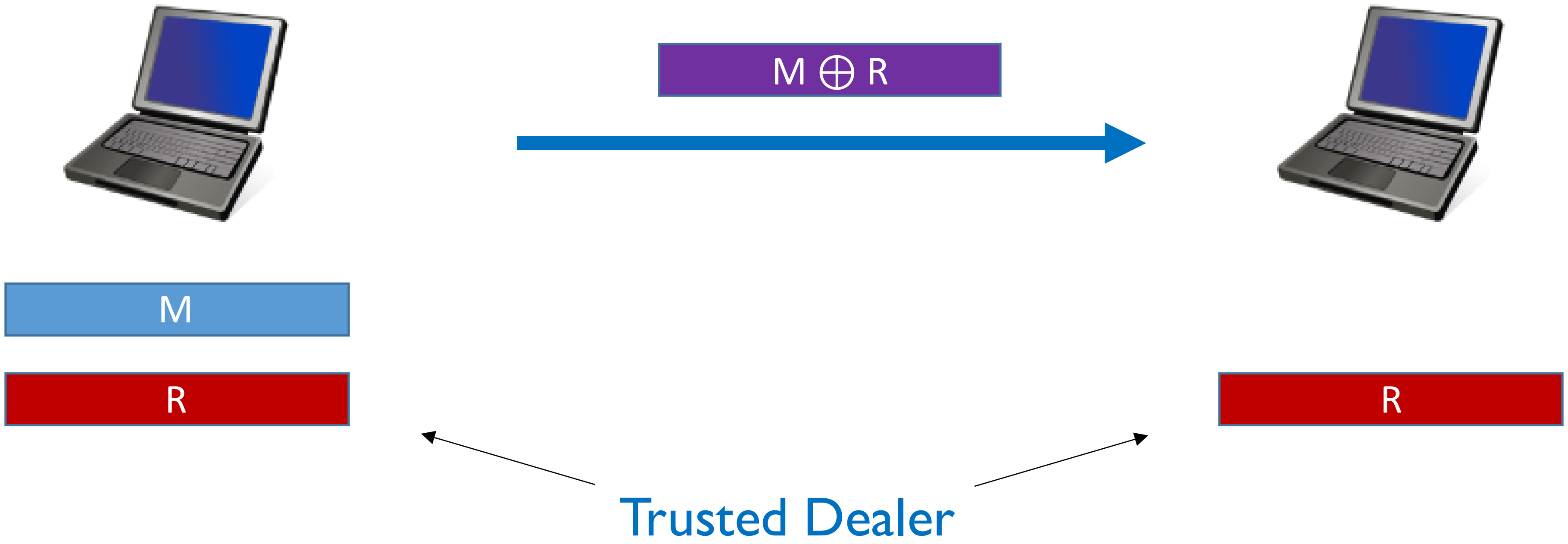
Based on joint works with Elette Boyle, Geoffroy Couteau, Ronald Cramer,
Ivan Damgård, Niv Gilboa, Lisa Kohl, Peter Rindal, and Peter Scholl

This talk

- Motivation: Secure computation with silent preprocessing
- Primitive: Pseudorandom Correlation Generator (PCG)
- Survey of PCG constructions

Secure Communication from common randomness

[Shannon 1944]



$M \oplus R$

M

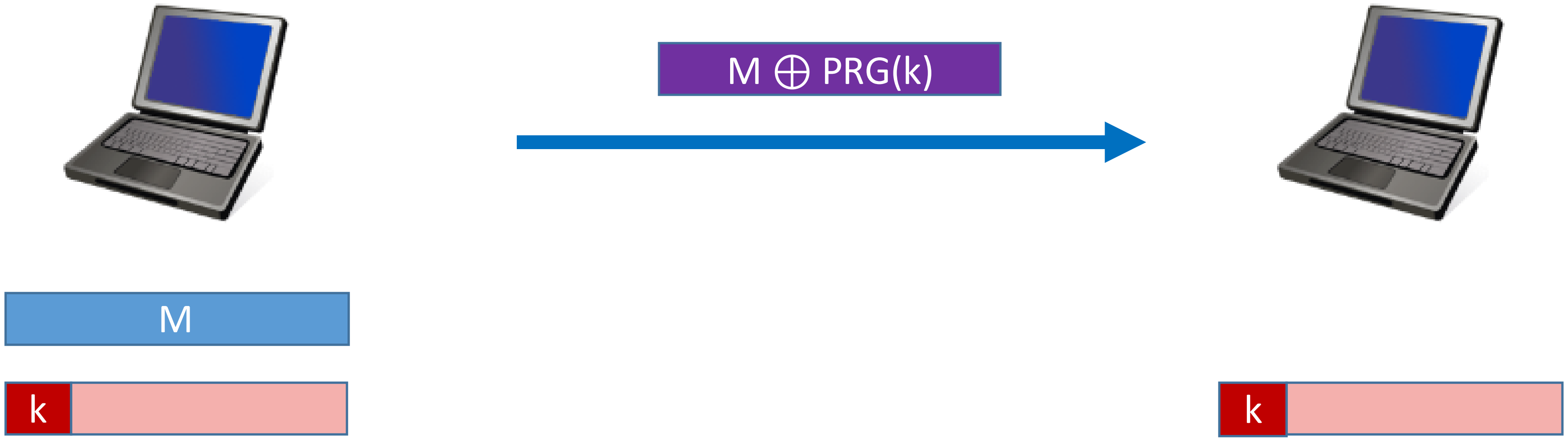
R

R

Trusted Dealer

Secure Communication from pseudorandomness

[Blum-Micali 1982, Yao 1982]



$M \oplus \mathrm{PRG}(k)$

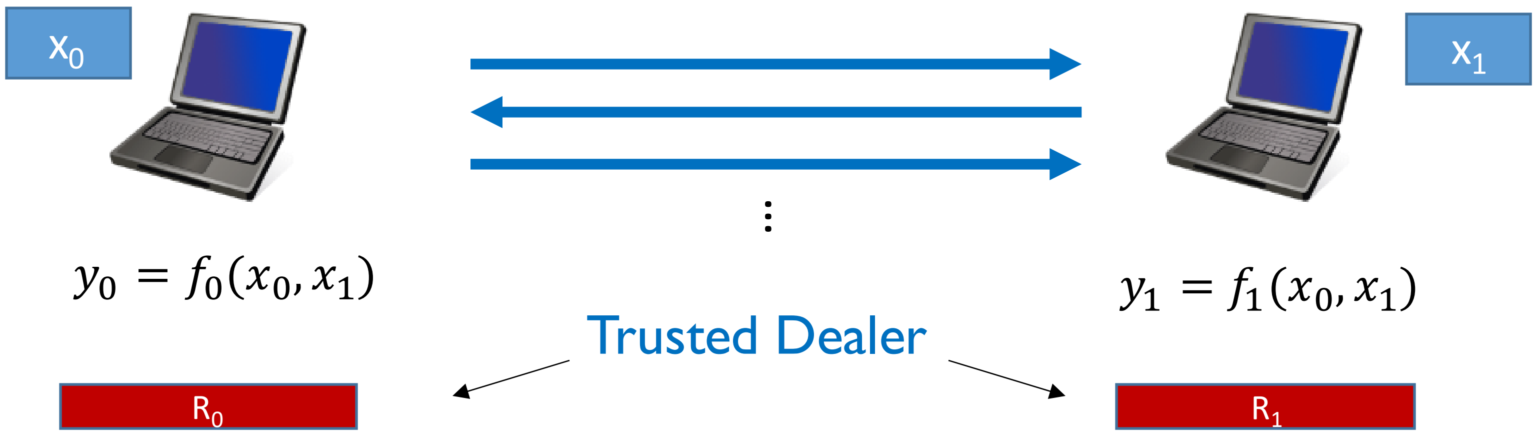
M

k

k

Secure Computation from correlated randomness

[Beaver 1995]



x_0

x_1

$y_0 = f_0(x_0, x_1)$

$y_1 = f_1(x_0, x_1)$

Trusted Dealer

R_0

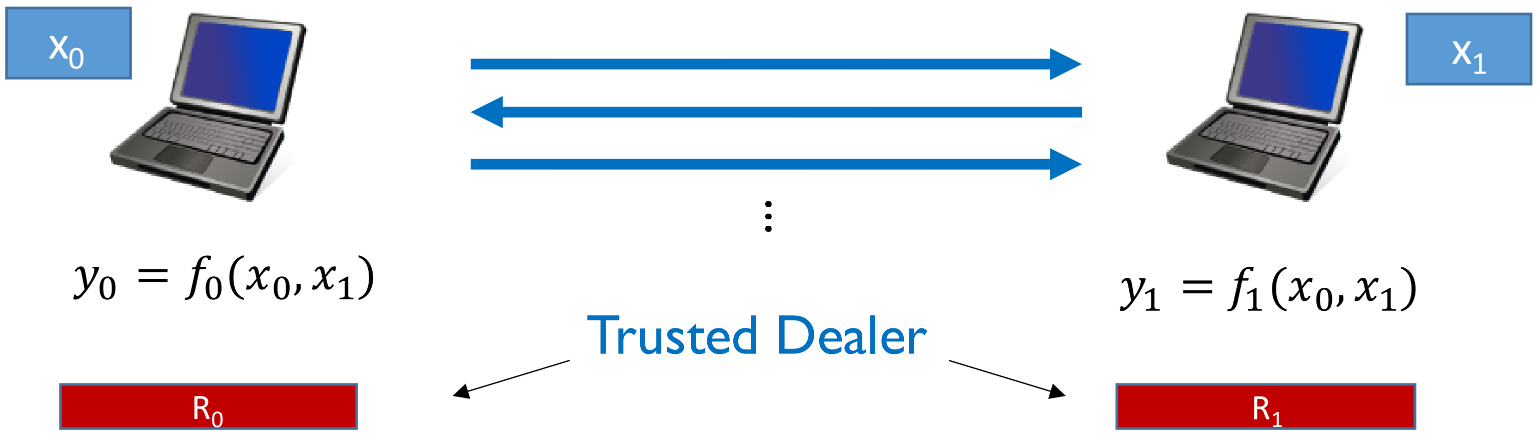
R_1

- Information-theoretic security
- Constant computational overhead

[Bea95, Bea97, IPS08, BDOZ11, BIKW12, NNOB12, DPSZ12, IKMOP13, DZ13, DLT14, BIKK14, LOS14, FKOS15, DZ16, KOS16, DNNR17, C18, BGI19, …]

Secure Computation from correlated randomness

[Beaver 1995]



x_0

x_1

$y_0 = f_0(x_0, x_1)$

$y_1 = f_1(x_0, x_1)$

Trusted Dealer

R_0

R_1

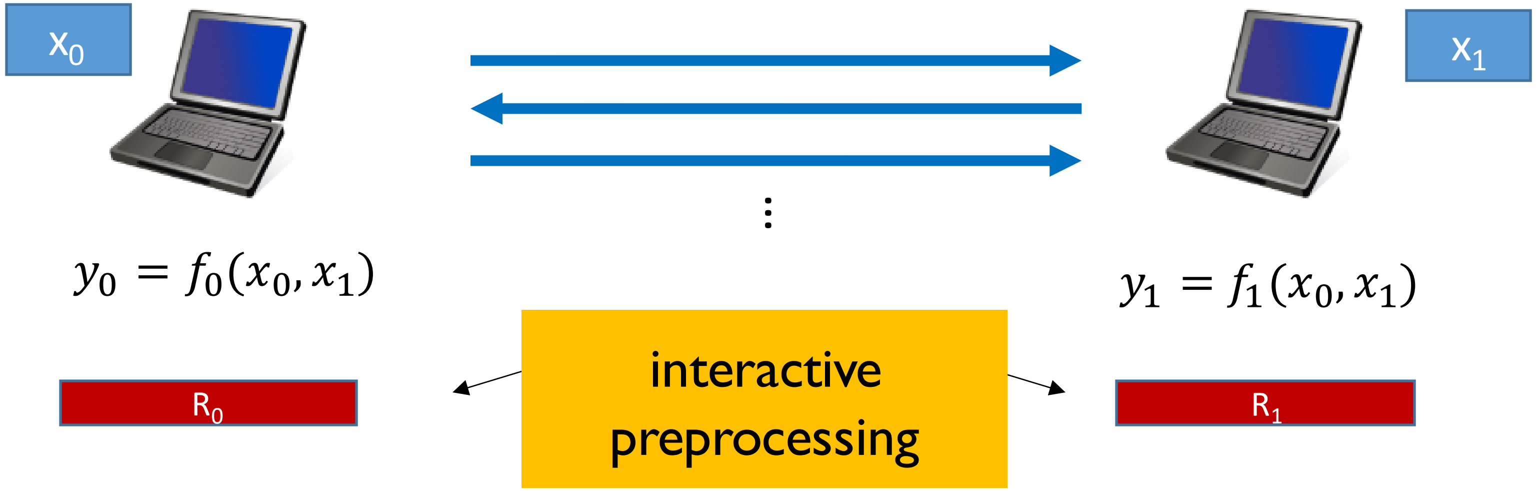
- Information-theoretic security
- Constant computational overhead

Useful correlations:

OT, OLE,VOLE, (authenticated) multiplication triples,
one-time truth-table, **multi-party** linear correlations

Secure Computation from correlated randomness

[Beaver 1995]



x_0

x_1

$y_0 = f_0(x_0, x_1)$

$y_1 = f_1(x_0, x_1)$

R_0

interactive preprocessing

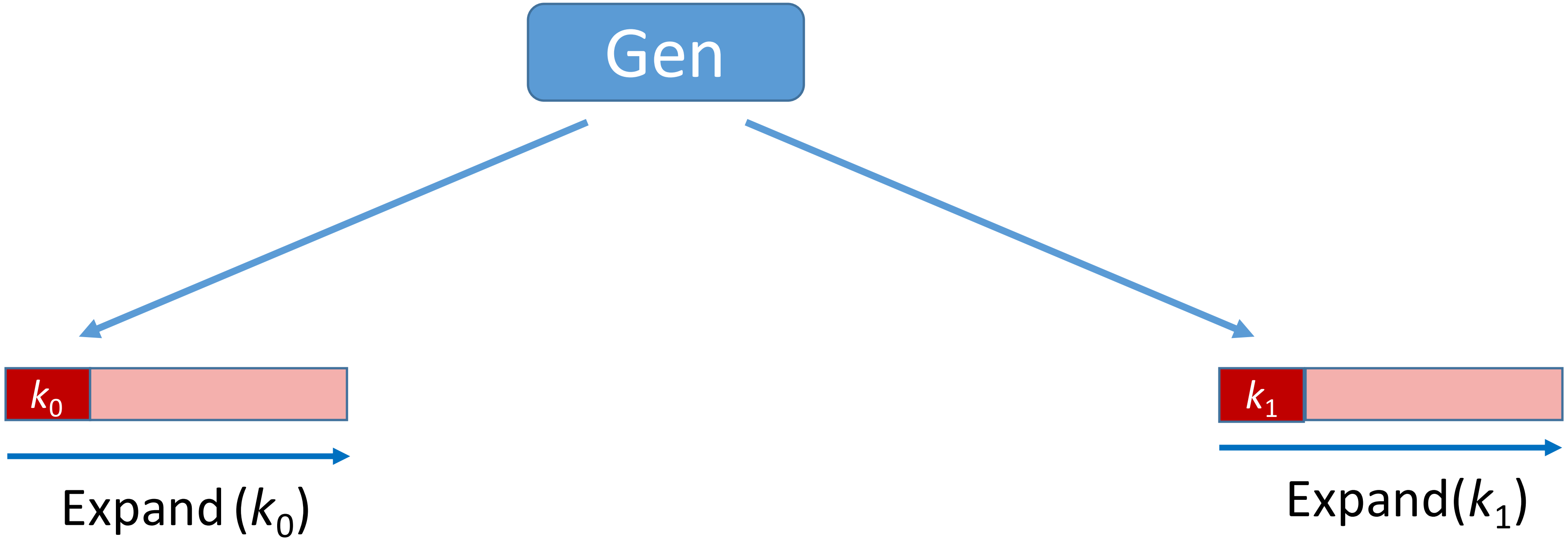
R_1

Dominates the total cost

Secure Computation
from correlated pseudorandomness?

Pseudorandom Correlation Generator (PCG)

[BCGI18, BCGIKS19]

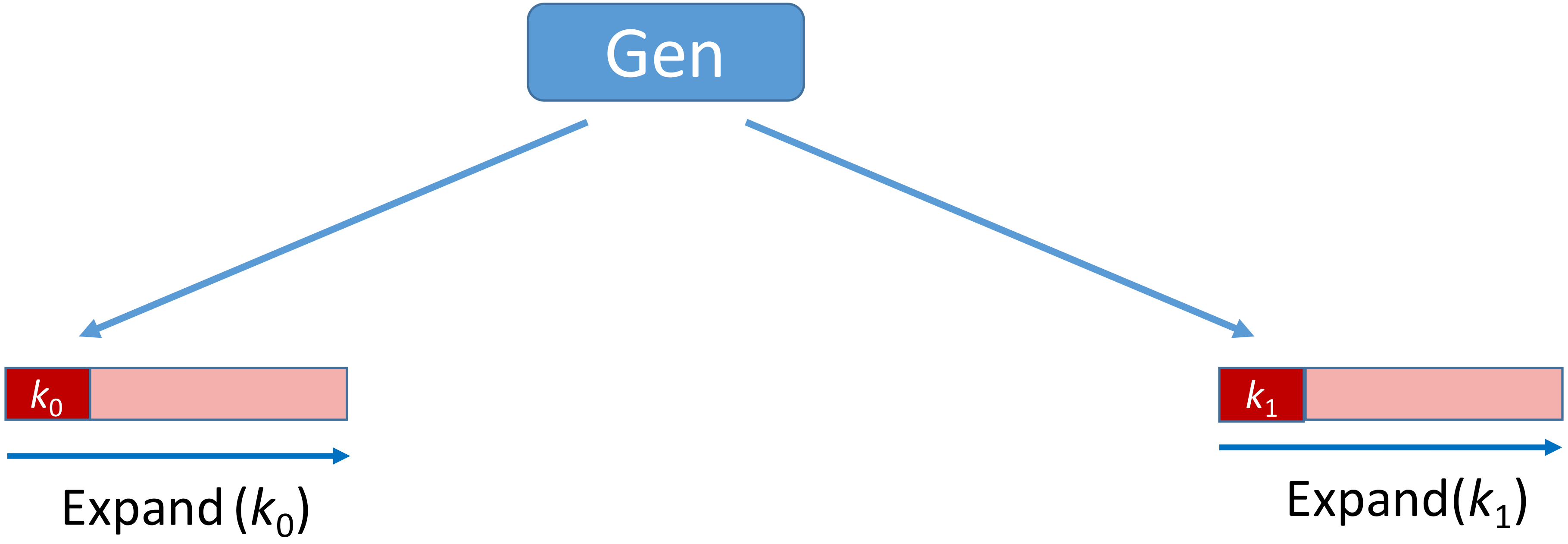


Target correlation: (R_0, R_1)

$$\left(\mathrm{Expand}(k_0), \mathrm{Expand}(k_1)\right) \approx (R_0, R_1)$$

Pseudorandom Correlation Generator (PCG)

[BCGI18, BCGIKS19]



How do we define security against insiders?

PCG: Security Definition

- Take I: Real = $(k_0, \mathrm{Expand}(k_1)) \approx (\mathrm{Sim}(R_0), R_1)$ = Ideal

Securely realizing ideal correlation functionality

Good for all applications

Not realizable even for simple correlations

PCG: Security Definition

- Take I: Real = $(k_0, \mathrm{Expand}(k_1)) \approx (\mathrm{Sim}(R_0), R_1)$ = Ideal
- Take II: Real = $(k_0, \mathrm{Expand}(k_1)) \approx (k_0, [R_1 \mid R_0 = \mathrm{Expand}(k_0)])$

Securely realizing “corruptible” correlation functionality

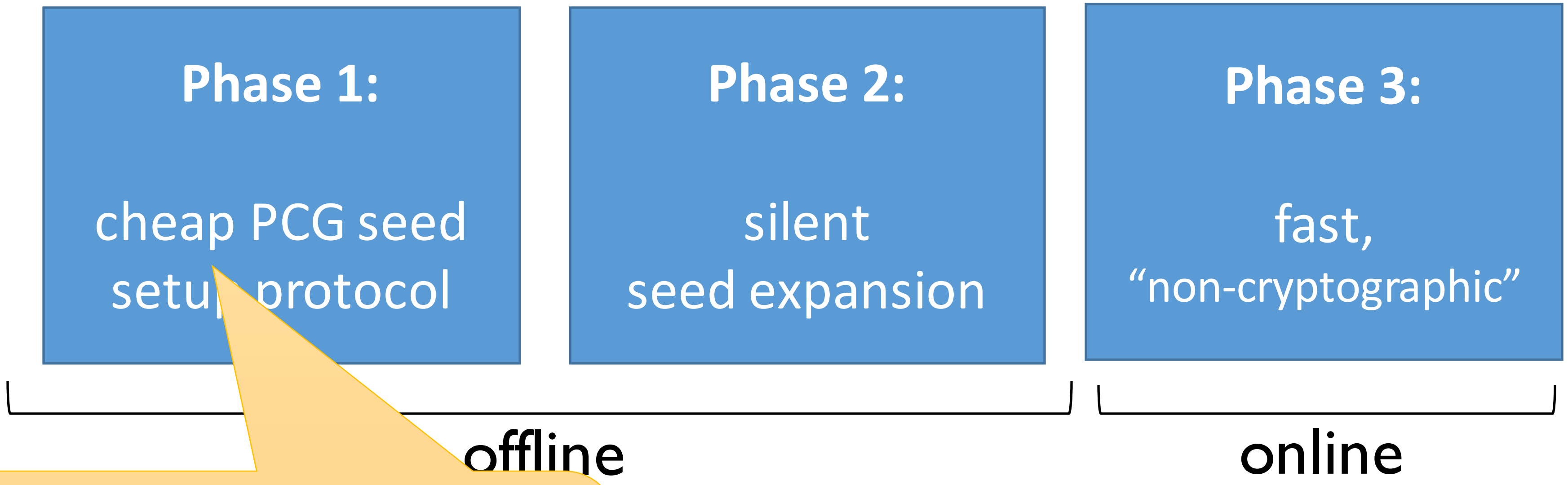
Good for natural applications

Realizable for useful correlations

PCG: Security Definition

- Take I: Real = $(k_0, \mathrm{Expand}(k_1\ )) \approx (\mathrm{Sim}(R_0), R_1)$ = Ideal

MPC with Silent Preprocessing



Phase 1:

cheap PCG seed setup protocol

Phase 2:

silent seed expansion

Phase 3:

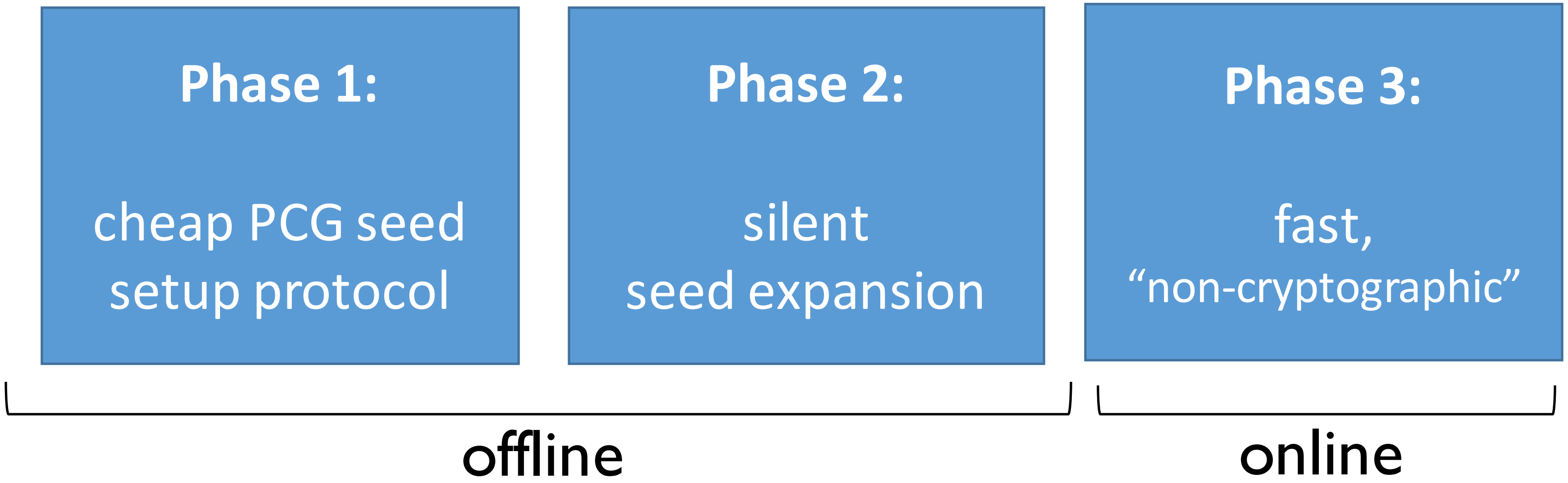
fast, "non-cryptographic"

offline

online

- ✓ Ad-hoc future interactions
- ✓ Hiding communication pattern
- ✓ Hiding future plans

MPC with Silent Preprocessing



Phase 1:

cheap PCG seed
setup protocol

Phase 2:

silent
seed expansion

Phase 3:

fast,
“non-cryptographic”

offline

online

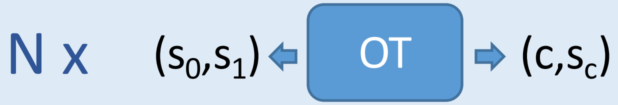
- Improved overall communication
- Near-optimal online computation
- Active security with **vanishing** amortized cost

Useful target correlations: 3+ parties

Linear n-party correlations	$(R_0, \ldots, R_{n-1}) \in_R$ Linear space V N x deg-t Shamir of random secret N x additive shares of 0	VSS, honest-majority MPC Proactive secret sharing Secure aggregation

Useful target correlations: 2+ parties

Oblivious transfer (OT)

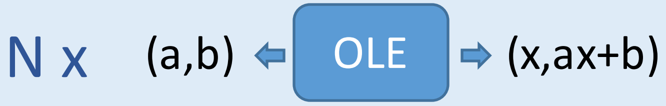

N x (s_0,s_1) ⇦ OT ⇨ (c,s_c)

2PC of Boolean circuits

GMW-style, passive:

2 x bit-OT + 4 comm. bits per AND

Oblivious Linear-function Evaluation (OLE)

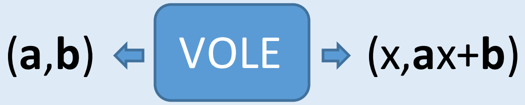

N x (a,b) ⇦ OLE ⇨ $(x,ax+b)$

2PC of Arithmetic circuits

GMW-style, passive:

2 x OLE + 4 ring elements per MULT

Vector OLE (VOLE)


$(\mathbf{a},\mathbf{b})$ ⇦ VOLE ⇨ $(x,\mathbf{a}x+\mathbf{b})$

2PC of scalar-vector product

Zero knowledge

PSI

Useful target correlations: 2+ parties

Authenticated Multiplication Triples	$([a_i],[b_i],[c_i],\ [\alpha a_i],[\alpha b_i],[\alpha c_i])$ $c_i=a_ib_i$	2PC of Arithmetic circuits SPDZ-style, active
Truth-tables	Randomly shifted, Secret-shared TT	2PC of “unstructured” functions
Additive	R0+R1 = R	Generalizes all the above

Current PCG Landscape

“Obfustopia”	iO	General [HW15, HIJKR16]
“Homomorphia”	LWE+	Additive [DHRW16, BCGIKS19]
“Cryptomania”	DDH,LWE	Low-depth [BCGIO17, BCGIKS19]
“Lapland”	LPN	VOLE, OT [BCGI18, BCGIKS19]
	Ring-LPN	OLE, (Auth.) Triples [BCGIKS20a]
	VD-LPN	PCF for VOLE, OT [BCGIKS20b]
“Minicrypt”	PRG	Linear multi-party [GI99, CDI05]
		Truth table [BCGIKS19]

Current PCG Landscape

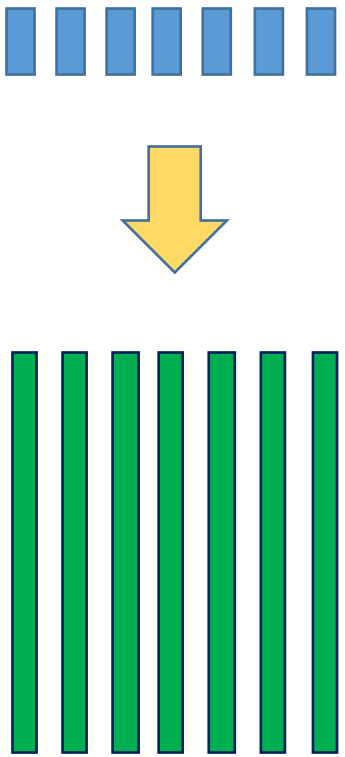
“Obfustopia”	iO	General [HW15, HIJKR16]
“Homomorphia”	LWE+	Additive [DHRW16, BCGIKS19]
“Cryptomania”	DDH,LWE	Low-depth [BCGIO17, BCGIKS19]
“Lapland”	LPN Ring-LPN VD-LPN	Constant-degree additive (poly(N) expansion time)
“Minicrypt”	PRG	Linear multi-party [GI99, CDI05] Truth table [BCGIKS19]

Good concrete efficiency?

“Obfustopia”	iO	General [HW15, HIJKR16]
“Homomorphia”	LWE+	Additive [DHRW16, BCGIKS19]
“Cryptomania”	DDH,LWE	Low-depth [BCGIO17, BCGIKS19]
“Lapland”	LPN	VOLE, OT [BCGI18, BCGIKS19]
	Ring-LPN	OLE, (Auth.) Triples [BCGIKS20a]
	VD-LPN	PCF for VOLE, OT [BCGIKS20b]
“Minicrypt”	PRG	Linear multi-party [GI99, CDI05]
		Truth table [BCGIKS19]

Pseudorandom secret sharing (PRSS)

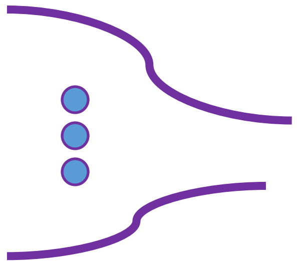
“Minicrypt” PRG Linear multi-party [GI99, CDI05]

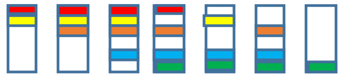


~ 0.3 KB seeds

~ 0.1 second

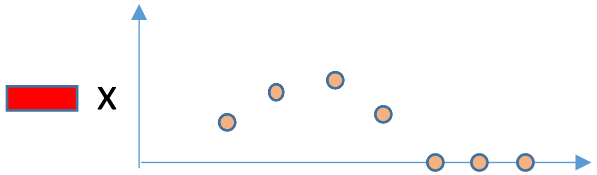
10^6 x deg-3 Shamir





deg-t share vectors

$\binom{n}{t}$ replicated PRG seeds



general linear

seed per min-support codeword

only efficient when $\binom{n}{t}$ is “[illegible]”

Additive shares of 0: $\binom{n}{2}$ seeds

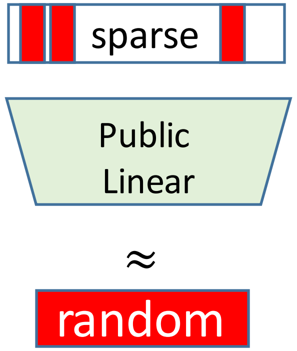
LPN-based PCGs: Tools

"Lapland"	LPN	VOLE, OT [BCGI18, BCGIKS19]
	Ring-LPN	OLE, (Auth.) Triples [BCGIKS20a]

(Dual) LPN

(Quasi)linear time

Peter's talk



Also over large fields / rings

Compressing secret-shared (N,w) sparse vector

Distributed Point Function
Function Secret Sharing
[GI14,BGI15,BGI16]

OLE, Triples
Truth-table, PCF

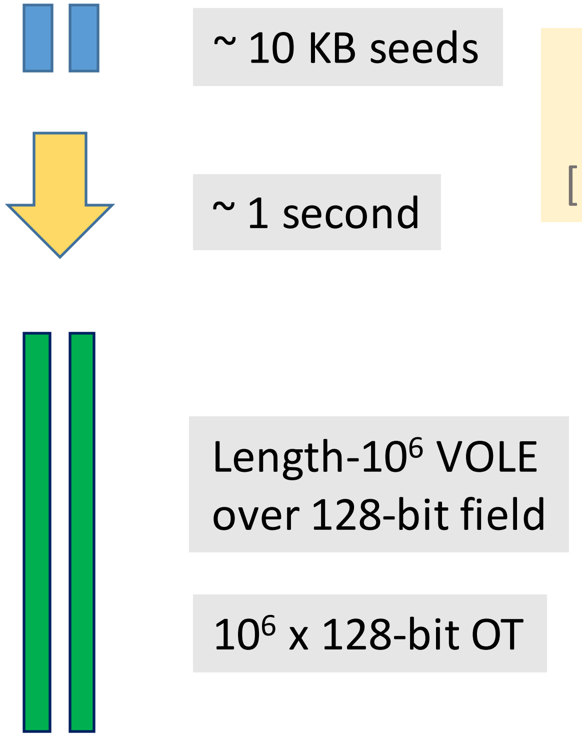
Puncturable PRF
[KPTZ13,BW13,BGI14]

VOLE, OT

GGM-style PRG tree
wlog(N) PRG seeds
O(N) x PRG calls expansion

LPN-based PCGs: VOLE and OT

“Lapland” LPN VOLE, OT [BCGI18, BCGIKS19]



~ 10 KB seeds

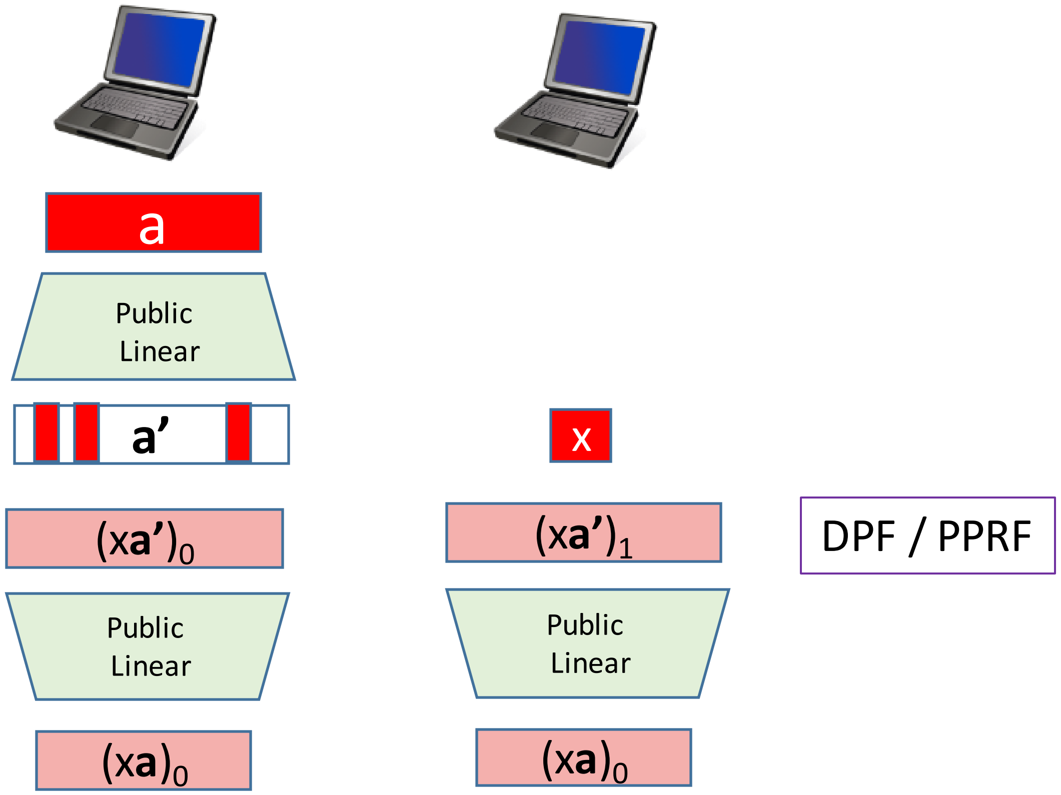
~ 1 second

Length-10^6 VOLE over 128-bit field

10^6 x 128-bit OT

~ 100 KB 2-round seed generation [BCGIKRS19,SGRR19]

Peter’s talk



a

Public Linear

a’

x

$(xa')_0$

$(xa')_1$

DPF / PPRF

Public Linear

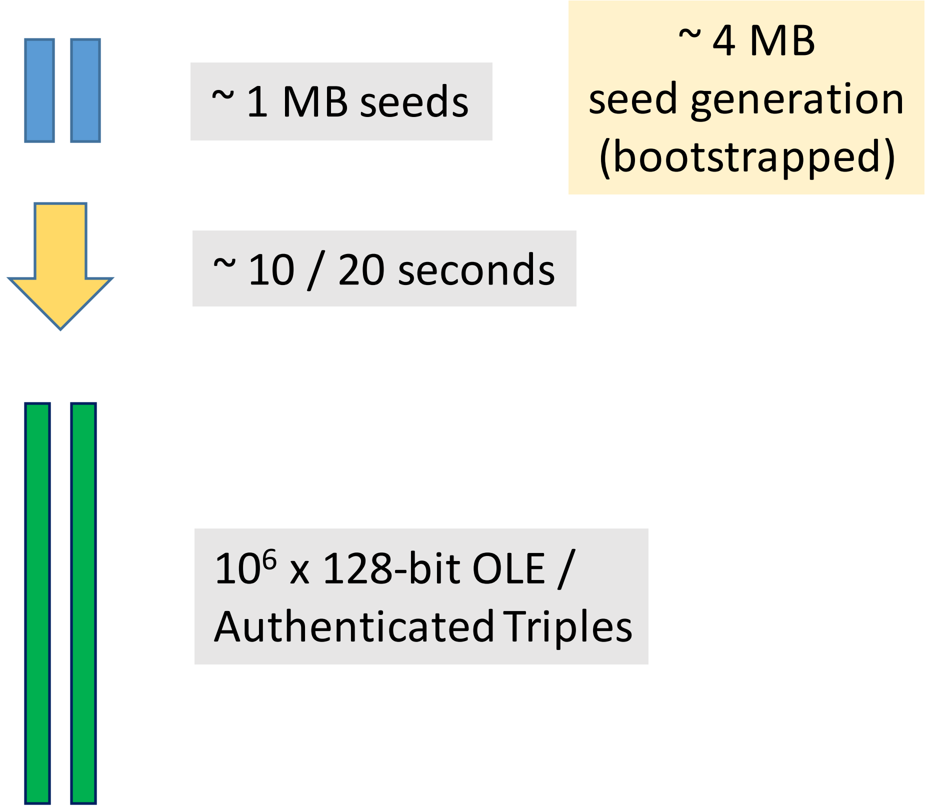
Public Linear

$(xa)_0$

$(xa)_0$

LPN-based PCGs: OLE and Triples

“Lapland” Ring-LPN VOLE, OT [BCGIKS20a]



~ 1 MB seeds

~ 4 MB
seed generation
(bootstrapped)

~ 10 / 20 seconds

10^6 x 128-bit OLE /
Authenticated Triples

Non-silent alternatives:
Overdrive [KPR18]
Leviosa [HIVM19]

x100-x1000 communication
comparable run time

Further Research

Better PCGs

- More correlations?
  - Garbled circuits, N x truth-tables, N x PCG seeds, …
- Multi-party variants
  - Shamir with t=n/2, authenticated triples
- Smaller seeds, faster expansion and seed generation

Better understanding of LPN-style assumptions

- Which codes?
- Which noise patterns?
- LPN vs. LWE

Better PCFs

The End

- Questions?