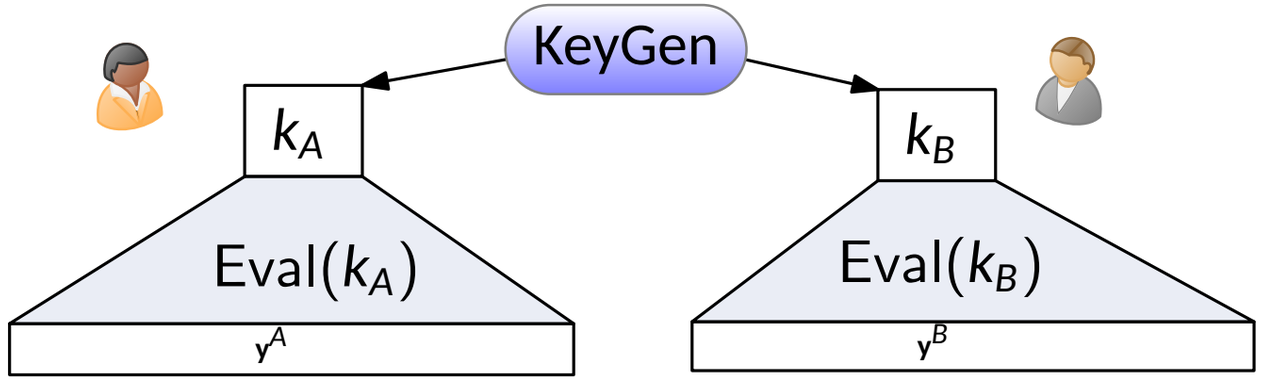
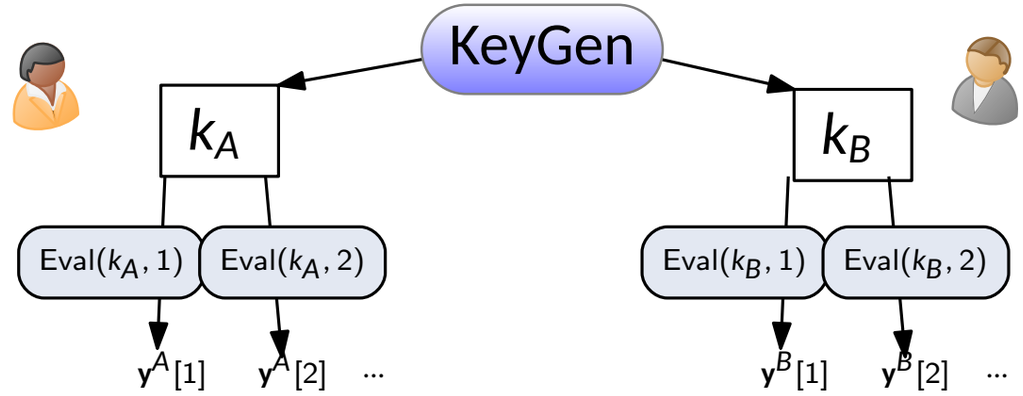
# QuietOT: Lightweight Oblivious Transfer with a Public-Key Setup

Geoffroy Couteau · Lalita Devadas · Srinivas Devadas · Alexander Koch · Sacha Servan-Schreiber

Oct. 24, 2024 · Presented by Hongrui Cui

# What is **P**seudorandom **C**orrelation **G**enerator/**F**unction




## Correlation Examples

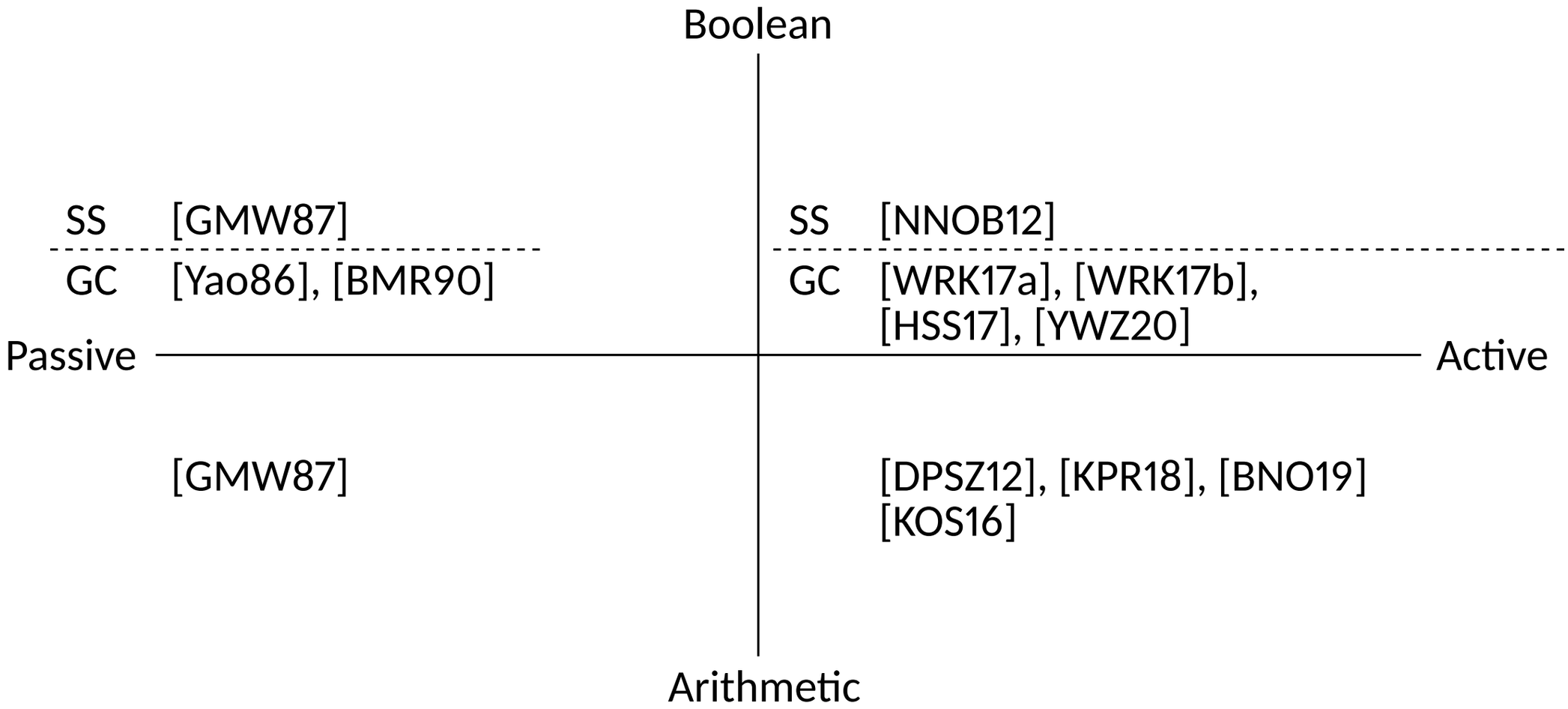
- Shared Randomness: $\mathbf{y}^A = \mathbf{y}^B$
- VOLE: $\mathbf{y}^A = (\mathbf{w}, \Delta), \mathbf{y}^B = (\mathbf{u}, \mathbf{v}),$ s.t. $\mathbf{w} = \mathbf{v} + \mathbf{u} \cdot \Delta$
- Beaver Triple: $\mathbf{y}^A + \mathbf{y}^B = (\mathbf{a}, \mathbf{b}, \mathbf{a} \circ \mathbf{b})$
- Authenticated Beaver Triple: $\mathbf{y}^A + \mathbf{y}^B = (\mathbf{a}, \mathbf{b}, \mathbf{a} \circ \mathbf{b}, \mathbf{a}\Delta, \mathbf{b}\Delta, \mathbf{a} \circ \mathbf{b}\Delta)$

## Motivation of This Line of Work

- **Silent** preprocessing for MPC
- KeyGen can be executed by
  - Trusted Third Party
  - Secure Multiparty Computation
  - Public Key Infrastructure

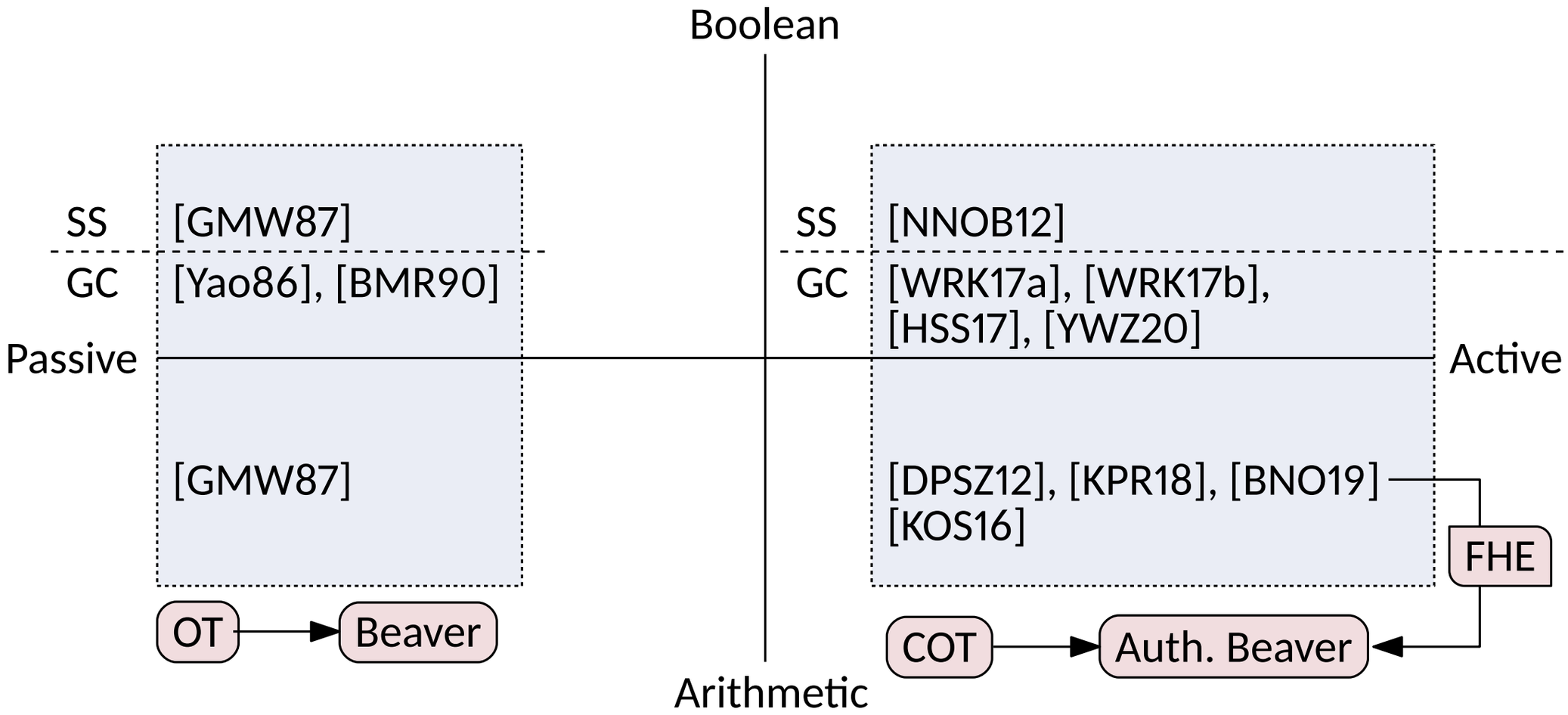
# MPC with All-but-one Corruption Threshold

- Passive Security vs. Active Security
- Boolean Circuit vs. Arithmetic Circuit (blackbox)



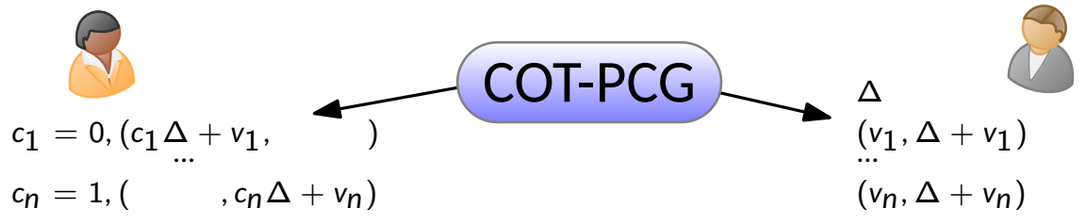
# MPC with All-but-one Corruption Threshold

- Passive Security vs. Active Security
- Boolean Circuit vs. Arithmetic Circuit (blackbox)



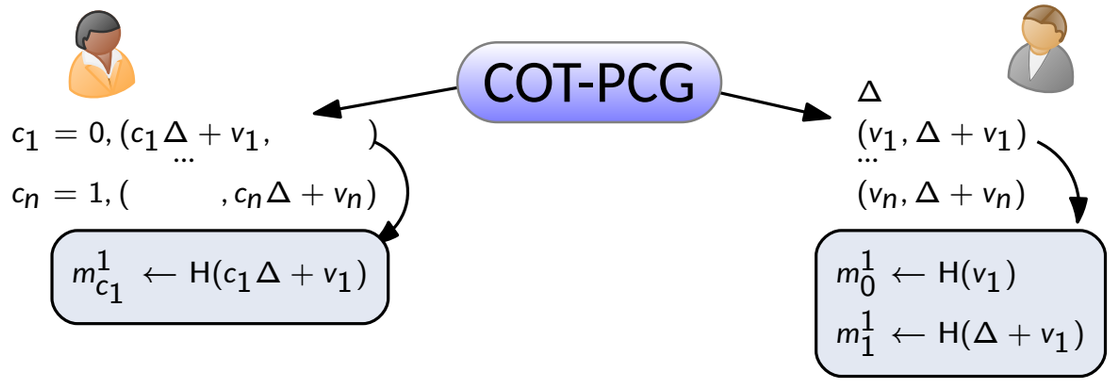
# Useful Correlations for MPC/2PC

- Oblivious Transfer from Correlated OT
- **C**orrelation **R**obust hash to get random OT
- Receiver sends diff. to get OT



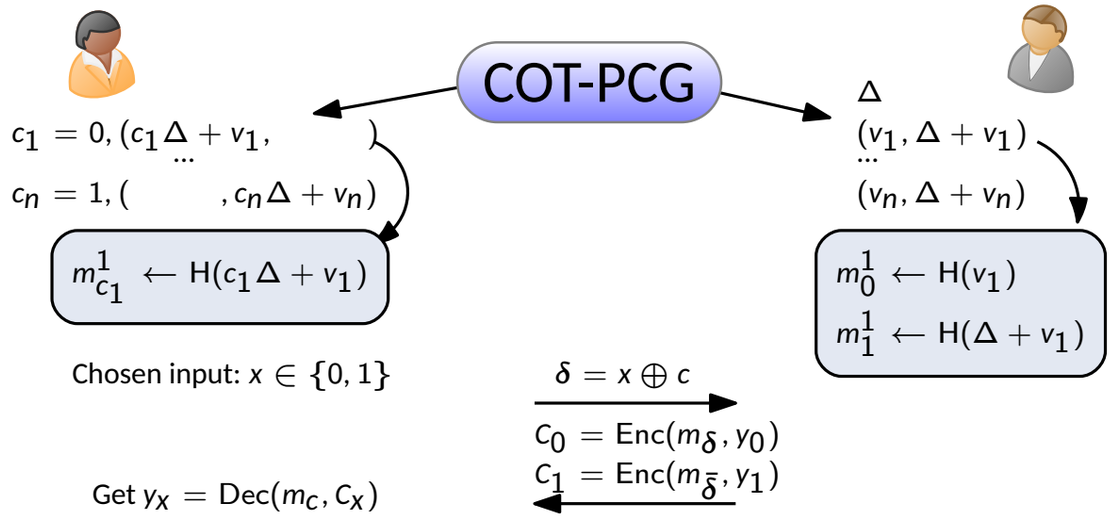
# Useful Correlations for MPC/2PC

- Oblivious Transfer from Correlated OT
- **C**orrelation **R**obust hash to get random OT
- Receiver sends diff. to get OT



# Useful Correlations for MPC/2PC

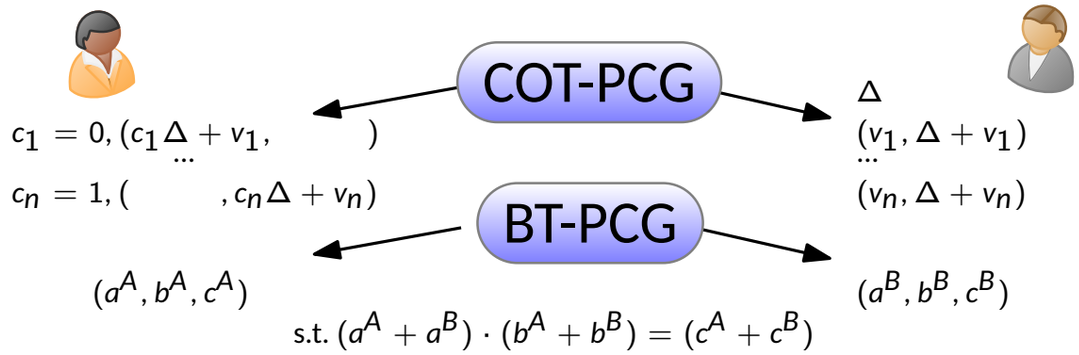
- Oblivious Transfer from Correlated OT
- **C**orrelation **R**obust hash to get random OT
- Receiver sends diff. to get OT



# Useful Correlations for MPC/2PC

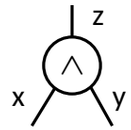
- Oblivious Transfer from Correlated OT
- **C**orrelation **R**obust hash to get random OT
- Receiver sends diff. to get OT

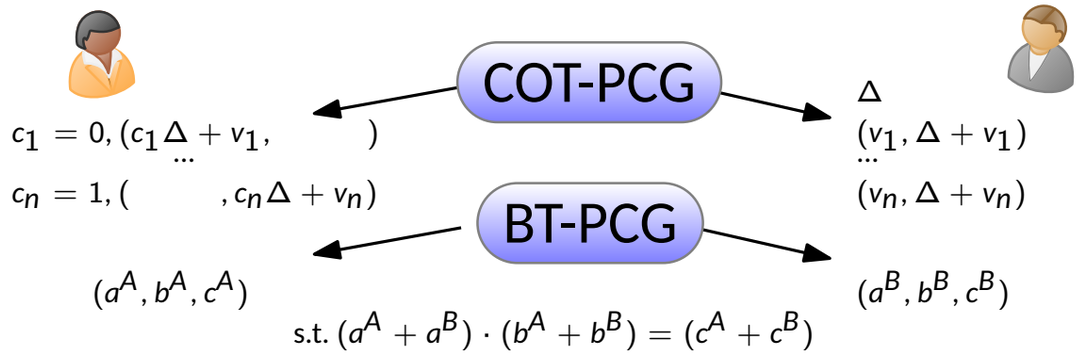
- Beaver Triple



# Useful Correlations for MPC/2PC

- Oblivious Transfer from Correlated OT
- **C**orrelation **R**obust hash to get random OT
- Receiver sends diff. to get OT
- Beaver Triple



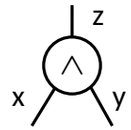




- Alice has $x \in \{0,1\}$ and samples $x^A, x^B$ s.t. $x^A \oplus x^B = x$, sends $x^B$ to Bob
- Bob has $y \in \{0,1\}$ and samples $y^A, y^B$ s.t. $y^A \oplus y^B = y$, sends $y^B$ to Alice
- Parties **open** $e = x \oplus a, f = y \oplus b$

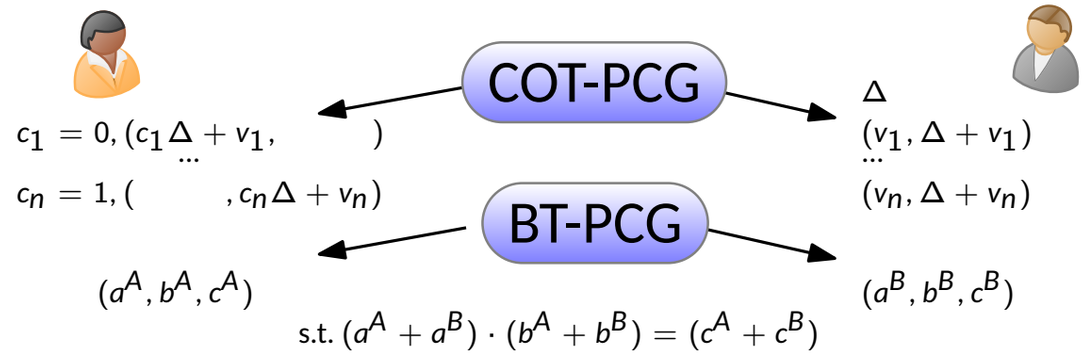
$$x \cdot y = (\underbrace{x \oplus a}_{e} \oplus a) \cdot (\underbrace{y \oplus b}_{f} \oplus b) = ef \oplus eb \oplus af \oplus \underbrace{ab}_{c}$$

# Useful Correlations for MPC/2PC

- Oblivious Transfer from Correlated OT
- **C**orrelation **R**obust hash to get random OT
- Receiver sends diff. to get OT
- Beaver Triple




- Alice has $x \in \{0,1\}$ and samples $x^A, x^B$ s.t. $x^A \oplus x^B = x$, sends $x^B$ to Bob
- Bob has $y \in \{0,1\}$ and samples $y^A, y^B$ s.t. $y^A \oplus y^B = y$, sends $y^B$ to Alice
- Parties **open** $e = x \oplus a, f = y \oplus b$

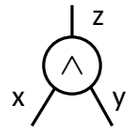



$$x \cdot y = (\underbrace{x \oplus a}_{e} \oplus a) \cdot (\underbrace{y \oplus b}_{f} \oplus b) = ef \oplus eb \oplus af \oplus \underbrace{ab}_{c}$$

$$[\![z]\!] = ef \oplus e[\![b]\!] \oplus f[\![a]\!] \oplus [\![c]\!]$$

# Useful Correlations for MPC/2PC

- Oblivious Transfer from Correlated OT
- **C**orrelation **R**obust hash to get random OT
- Receiver sends diff. to get OT
- Beaver Triple




- Alice has $x \in \{0,1\}$ and samples $x^A, x^B$ s.t. $x^A \oplus x^B = x$, sends $x^B$ to Bob
- Bob has $y \in \{0,1\}$ and samples $y^A, y^B$ s.t. $y^A \oplus y^B = y$, sends $y^B$ to Alice
- Parties **open** $e = x \oplus a, f = y \oplus b$

- Active security with SPDZ-style authentication

COT-PCG

$c_1 = 0, (c_1\Delta + v_1, \ldots)$

$c_n = 1, (\ldots, c_n\Delta + v_n)$

$\Delta$

$(v_1, \Delta + v_1)$

$\ldots$

$(v_n, \Delta + v_n)$

BT-PCG

$(a^A, b^A, c^A)$ $(a^B, b^B, c^B)$

s.t. $(a^A + a^B) \cdot (b^A + b^B) = (c^A + c^B)$

$$x \cdot y = (\underbrace{x \oplus a}_{e} \oplus a) \cdot (\underbrace{y \oplus b}_{f} \oplus b) = ef \oplus eb \oplus af \oplus \underbrace{ab}_{c}$$

$$[\![z]\!] = ef \oplus e[\![b]\!] \oplus f[\![a]\!] \oplus [\![c]\!]$$

ABT-PCG

$(a^A, b^A, c^A)$ $(a^B, b^B, c^B)$

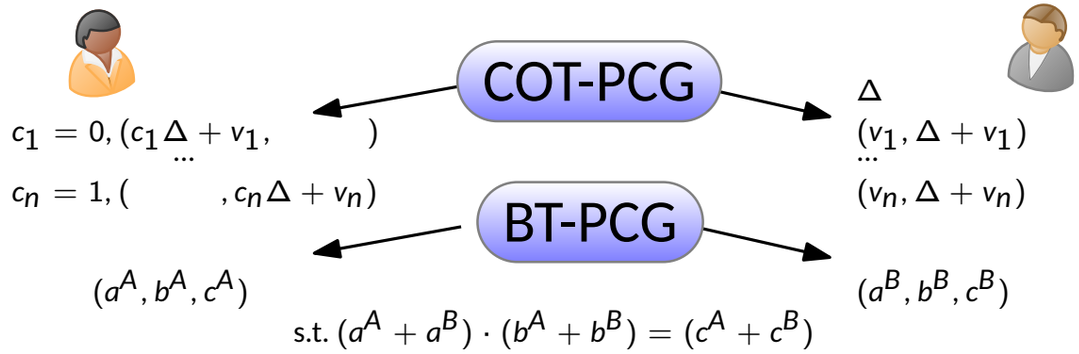
$(\Delta^A, \mathrm{M}[a^A], \mathrm{M}[b^A], \mathrm{M}[c^A])$ $(\Delta^B, \mathrm{M}[a^B], \mathrm{M}[b^B], \mathrm{M}[c^B])$

s.t. $(a^A + a^B) \cdot (b^A + b^B) = (c^A + c^B)$

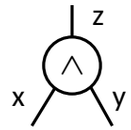
$(u^A + u^B) \cdot (\Delta^A + \Delta^B) = \mathrm{M}[u^A] + \mathrm{M}[u^B]$ over $\mathbb{F}_{2^\lambda}$

# Useful Correlations for MPC/2PC

- Oblivious Transfer from Correlated OT
- **C**orrelation **R**obust hash to get random OT
- Receiver sends diff. to get OT




- Beaver Triple

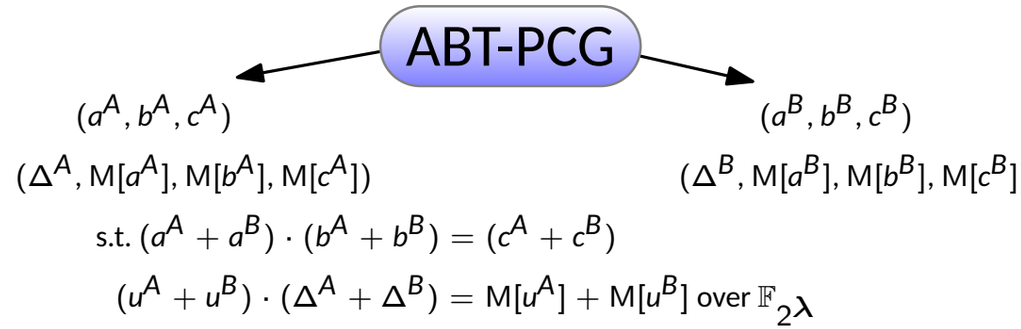



- Alice has $x \in \{0,1\}$ and samples $x^A, x^B$ s.t. $x^A \oplus x^B = x$, sends $x^B$ to Bob
- Bob has $y \in \{0,1\}$ and samples $y^A, y^B$ s.t. $y^A \oplus y^B = y$, sends $y^B$ to Alice
- Parties **open** $e = x \oplus a, f = y \oplus b$

$$x \cdot y = (\underbrace{x \oplus a}_{e} \oplus a) \cdot (\underbrace{y \oplus b}_{f} \oplus b) = ef \oplus eb \oplus af \oplus \underbrace{ab}_{c}$$

$$[\![z]\!] = ef \oplus e[\![b]\!] \oplus f[\![a]\!] \oplus [\![c]\!]$$

- Active security with SPDZ-style authentication




- IT-MAC has additive-homomorphism
- **open** = reveal + check
- reveal$(x)$ = send $x^A, x^B$
- check$(x)$ = check $\mathrm{M}[x^A] + \mathrm{M}[x^B] = (x^A + x^B)(\Delta^A + \Delta^B)$

# Efficient Generation of COT/sVOLE

MPC Jargon: Communication $\approx$ Sound Level

**SoftSpokenOT: Quieter OT Extension from Small-Field Silent VOLE in the Minicrypt Model**

Lawrence Roy(✉)

Oregon State University, Corvallis, USA
ldr709@gmail.com

## OT Extension

- **Prior art**. [IKNP03], [KOS15]. $\lambda$ bit per OT or $\lambda - 1$ bit per COT
- **SOTA**. [Roy22]. $\frac{\lambda}{\tau}$ bit per COT, with $\frac{2^\tau}{\tau}$-time computational overhead

## Silent OT

- **PCG**. [BCGIKS19], [YWLZW20], [RRT23]. $o(1)$ bit per COT/sVOLE
- **PCF**. [BCGIKS20, BCGIKRS22, ] $o(1)$ bit per unlimited COT/sVOLE

# Efficient Generation of COT/sVOLE

**SoftSpokenOT: Quieter OT Extension from Small-Field Silent VOLE in the Minicrypt Model**

Lawrence Roy(✉)

Oregon State University, Corvallis, USA
ldr709@gmail.com

MPC Jargon: Communication $\approx$ Sound Level

OT Extension

- **Prior art**. [IKNP03], [KOS15]. $\lambda$ bit per OT or $\lambda - 1$ bit per COT
- **SOTA**. [Roy22]. $\frac{\lambda}{\tau}$ bit per COT, with $\frac{2^\tau}{\tau}$-time computational overhead

Silent OT

- **PCG**. [BCGIKS19], [YWLZW20], [RRT23]. $o(1)$ bit per COT/sVOLE
- **PCF**. [BCGIKS20, BCGIKRS22, ] $o(1)$ bit per unlimited COT/sVOLE

Public Key PCF [OSY21, KOR23, BCMPR24]

- **Advantage**. Conceptually simpler than LPN-based PCF
- **Public Key Setup**. Requires a PKI-style setup (as opposed to an interactive setup)
- **Disadvantage**. Requires PK operations (e.g., modular exponentiation) per correlation

# Efficient Generation of COT/sVOLE

MPC Jargon: Communication $\approx$ Sound Level

**SoftSpokenOT: Quieter OT Extension from Small-Field Silent VOLE in the Minicrypt Model**

Lawrence Roy(✉)

Oregon State University, Corvallis, USA
ldr709@gmail.com

## OT Extension

- **Prior art**. [IKNP03], [KOS15]. $\lambda$ bit per OT or $\lambda - 1$ bit per COT
- **SOTA**. [Roy22]. $\frac{\lambda}{\tau}$ bit per COT, with $\frac{2^\tau}{\tau}$-time computational overhead

## Silent OT

- **PCG**. [BCGIKS19], [YWLZW20], [RRT23]. $o(1)$ bit per COT/sVOLE
- **PCF**. [BCGIKS20, BCGIKRS22, ] $o(1)$ bit per unlimited COT/sVOLE

A middle ground

**QuietOT: Lightweight Oblivious Transfer with a Public-Key Setup**

Geoffroy Couteau[1,2], Lalita Devadas[3], Srinivas Devadas[3], Alexander Koch[1,2], and Sacha Servan-Schreiber[3]

[1] CNRS
[2] IRIF, Université Paris Cité
[3] MIT

- Relaxing the Output Requirement ✗
- Adding Communication ✗
- Symmetric Operations for Extension ✓
- Public Key Setup ✓

## Public Key PCF [OSY21, KOR23, BCMPR24]

- **Advantage**. Conceptually simpler than LPN-based PCF
- **Public Key Setup**. Requires a PKI-style setup (as opposed to an interactive setup)
- **Disadvantage**. Requires PK operations (e.g., modular exponentiation) per correlation

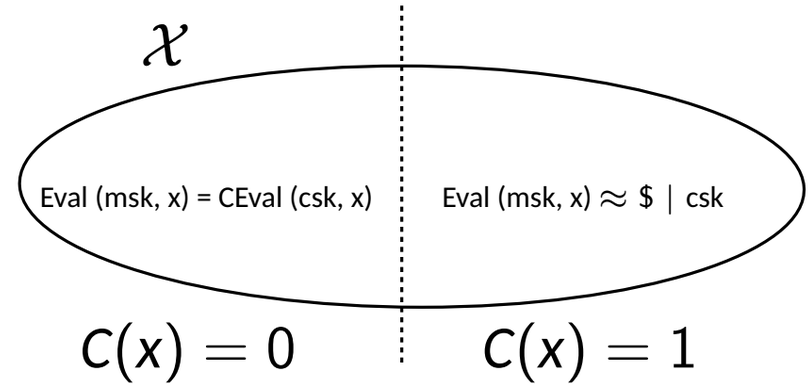
# General Paradigm from [BCMPR23]

- Folklore: *Any CPRF with a weak PRF as a constraint predicate can be used to construct a PCF for OT correlations.* [BGMM20]

**Constrained PRF**: $\mathcal{K} \times \mathcal{X} \to \mathcal{Y}$

- KeyGen $\to$ (pp, msk)
- Eval (msk, x) $\to$ y
- **Constrain** (msk, C) $\to$ csk
- CEval (csk, x) $\to$ y

$$C : \mathcal{X} \to \{0, 1\}$$



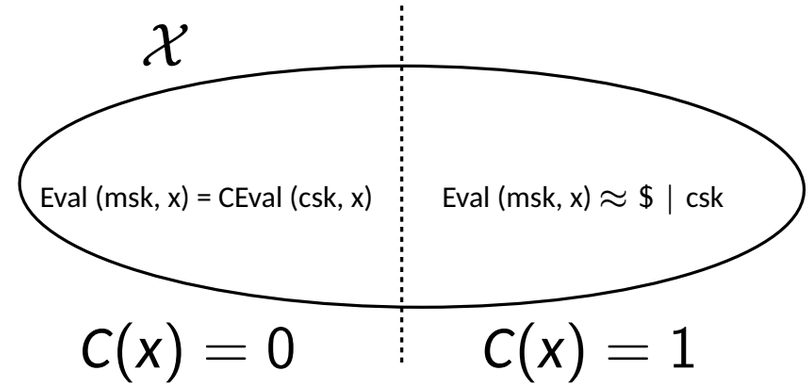
# General Paradigm from [BCMPR23]

- Folklore: *Any CPRF with a weak PRF as a constraint predicate can be used to construct a PCF for OT correlations.* [BGMM20]

**Constrained PRF**: $\mathcal{K} \times \mathcal{X} \to \mathcal{Y}$

- KeyGen $\to$ (pp, msk)
- Eval (msk, x) $\to$ y
- **Constrain** (msk, C) $\to$ csk
- CEval (csk, x) $\to$ y

$C : \mathcal{X} \to \{0, 1\}$




wPRF

- Suppose that $C = f_{\mathbf{z}} : \mathcal{X} \to \{0, 1\}$ is a wPRF
- We can use $f_{\mathbf{z}}, 1 - f_{\mathbf{z}}$ as constraints


Sender

Setup


Receiver

$msk_0$
$msk_1$

$csk_0 = \text{Cstr.}(msk_0, f_{\mathbf{z}})$
$csk_1 = \text{Cstr.}(msk_1, 1 - f_{\mathbf{z}})$
$\mathbf{z}$

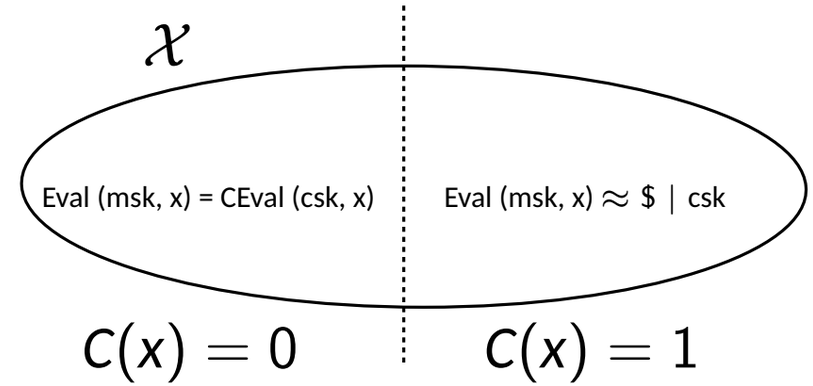
# General Paradigm from [BCMPR23]

- Folklore: *Any CPRF with a weak PRF as a constraint predicate can be used to construct a PCF for OT correlations.* [BGMM20]

**Constrained PRF**: $\mathcal{K} \times \mathcal{X} \to \mathcal{Y}$

- KeyGen $\to$ (pp, msk)
- Eval (msk, x) $\to$ y
- **Constrain** (msk, C) $\to$ csk
- CEval (csk, x) $\to$ y

$$C : \mathcal{X} \to \{0,1\}$$




wPRF

- Suppose that $C = f_{\mathbf{z}} : \mathcal{X} \to \{0,1\}$ is a wPRF
- We can use $f_{\mathbf{z}}, 1 - f_{\mathbf{z}}$ as constraints


Sender

Setup


Receiver

$msk_0$
$msk_1$

$csk_0 = \text{Cstr.}(msk_0, f_{\mathbf{z}})$
$csk_1 = \text{Cstr.}(msk_1, 1 - f_{\mathbf{z}})$
$\mathbf{z}$

$$b = f_{\mathbf{z}}(\mathbf{x})$$

$$y_0 = \text{Eval}(msk_0, \mathbf{x})$$
$$y_1 = \text{Eval}(msk_1, \mathbf{x})$$

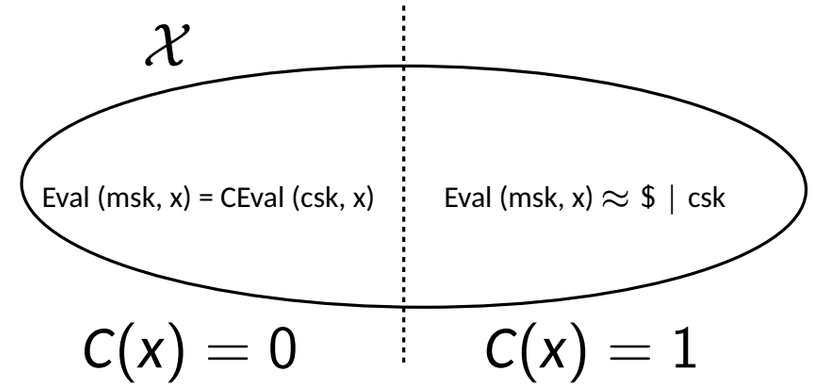
# General Paradigm from [BCMPR23]

- Folklore: *Any CPRF with a weak PRF as a constraint predicate can be used to construct a PCF for OT correlations.* [BGMM20]

**Constrained PRF**: $\mathcal{K} \times \mathcal{X} \to \mathcal{Y}$

- KeyGen $\to$ (pp, msk)
- Eval (msk, x) $\to$ y
- **Constrain** (msk, C) $\to$ csk
- CEval (csk, x) $\to$ y

$C : \mathcal{X} \to \{0,1\}$




wPRF

- Suppose that $C = f_{\mathbf{z}} : \mathcal{X} \to \{0,1\}$ is a wPRF
- We can use $f_{\mathbf{z}}, 1 - f_{\mathbf{z}}$ as constraints


Sender

Setup


Receiver

$msk_0$
$msk_1$

$csk_0 = \text{Cstr.}(msk_0, f_{\mathbf{z}})$
$csk_1 = \text{Cstr.}(msk_1, 1 - f_{\mathbf{z}})$
$\mathbf{z}$

$b = f_{\mathbf{z}}(\mathbf{x})$ Let $b = 1$

$y_0 = \text{Eval}(msk_0, \mathbf{x})$ ⇢ ✗

$y_1 = \text{Eval}(msk_1, \mathbf{x})$ ⇢ $y_b = \text{CEval}(csk_b, \mathbf{x})$

Output $(y_0, y_1)$ Output $(b, y_b)$

# CPRF and IPM-wPRF

- Unfortunately, most CPRF only support inner-product-related constraint functions
- We cannot design a wPRF using $\langle z, x \rangle$ operation alone

Example: NR04 PRF

- Let $(\mathbb{G}, g, q)$ be a DDH group
- $f_{a_1,\ldots,a_n}(x_1, \ldots x_n) = g^{a_1^{x_1} \cdot \ldots \cdot a_n^{x_n}}, \qquad a_1, \ldots, a_n \in \mathbb{Z}_q, x_1, \ldots, x_n \in \{0,1\}$
- Constraint: $z_1, \ldots, z_n \in \{0,1\}$, $C_{\mathbf{z}}(x) = 0 \iff \langle \mathbf{x}, \mathbf{z} \rangle = 0$
- Sample $r \leftarrow \mathbb{Z}_q$, csk: $\alpha_i = r^{z_i} \cdot a_i$, $\mathit{CEval}(x_1, \ldots x_n) = g^{\alpha_1^{x_1} \cdot \ldots \cdot \alpha_n^{x_n}} = g^{r^{\langle \mathbf{x}, \mathbf{z} \rangle} \cdot (a_1^{x_1} \cdot \ldots \cdot a_n^{x_n})}$

# CPRF and IPM-wPRF

- Unfortunately, most CPRF only support inner-product-related constraint functions
- We cannot design a wPRF using $\langle z, x \rangle$ operation alone

## Example: NR04 PRF

- Let $(\mathbb{G}, g, q)$ be a DDH group
- $f_{a_1,\ldots,a_n}(x_1, \ldots x_n) = g^{a_1^{x_1} \cdot \ldots \cdot a_n^{x_n}}, \qquad a_1, \ldots, a_n \in \mathbb{Z}_q, x_1, \ldots, x_n \in \{0,1\}$
- Constraint: $z_1, \ldots, z_n \in \{0,1\}$, $C_{\mathbf{z}}(x) = 0 \iff \langle \mathbf{x}, \mathbf{z} \rangle = 0$
- Sample $r \leftarrow \mathbb{Z}_q$, csk: $\alpha_i = r^{z_i} \cdot a_i$, $\mathit{CEval}(x_1, \ldots x_n) = g^{\alpha_1^{x_1} \cdot \ldots \cdot \alpha_n^{x_n}} = g^{r^{\langle \mathbf{x}, \mathbf{z} \rangle} \cdot (a_1^{x_1} \cdot \ldots \cdot a_n^{x_n})}$

## Inner Product Membership wPRF

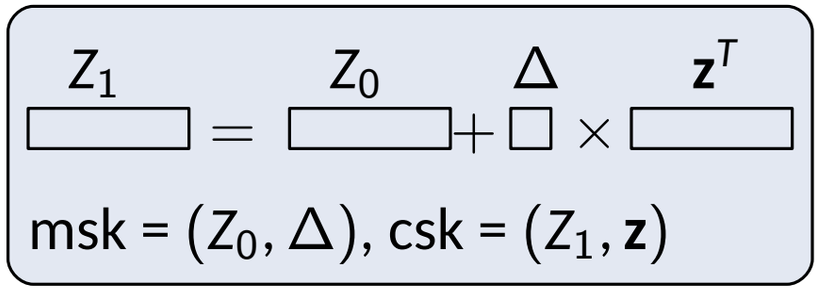
- Defined over a ring $\mathcal{R} = S_0 \cup S_1$ ($S_1 = \mathcal{R} \setminus S_0$)
- $f : \mathcal{R}^n \times \mathcal{R}^n \to \{0,1\}$, for sk = $\mathbf{z} \in \mathcal{R}^n$, define $f_{\mathbf{z}}(\mathbf{x}) = f(\mathbf{z}, \mathbf{x})$
- $f_z = \begin{cases} 0 & \langle \mathbf{x}, \mathbf{z} \rangle \in S_0 \\ 1 & \langle \mathbf{x}, \mathbf{z} \rangle \in S_1 \end{cases}$

## Example

- BIPSW: $\mathcal{R} = \mathbb{Z}_6$, $f_{\mathbf{z}}(\mathbf{x}) = \lfloor \langle \mathbf{x}, \mathbf{z} \rangle \rceil_2$
- GAR: $f_z(\mathcal{I}_1, \mathcal{I}_2) = \mathrm{XOR}(\{z_i : i \in \mathcal{I}_1\}) \oplus \mathrm{MAJ}(\{z_i : i \in \mathcal{I}_2\})$

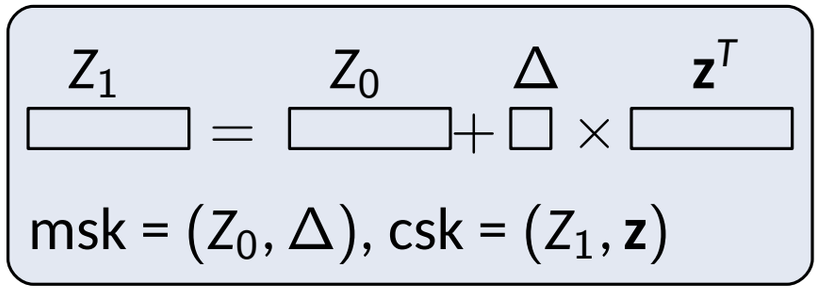
# Using Symmetric-key Primitive For Extension

- The BCMPR construction requires modular exponentiation for every extension
- QuietOT proposed to use RO instead

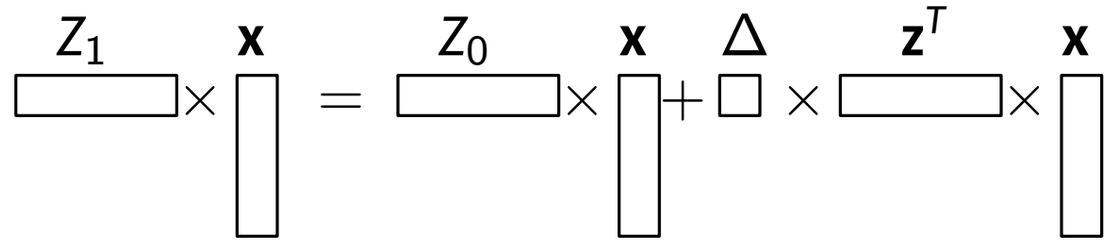


# Using Symmetric-key Primitive For Extension

- The BCMPR construction requires modular exponentiation for every extension
- QuietOT proposed to use RO instead



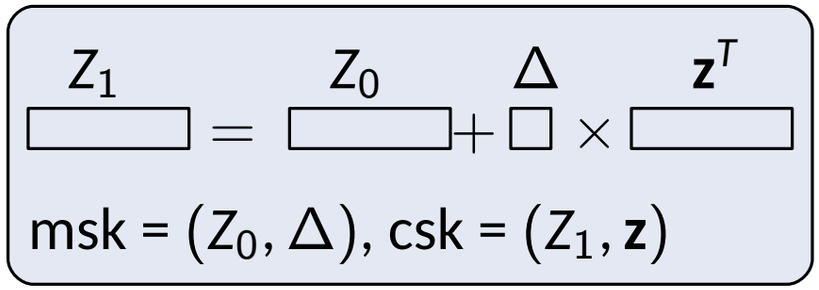

Eval at $x \leftarrow \mathcal{R}^n$



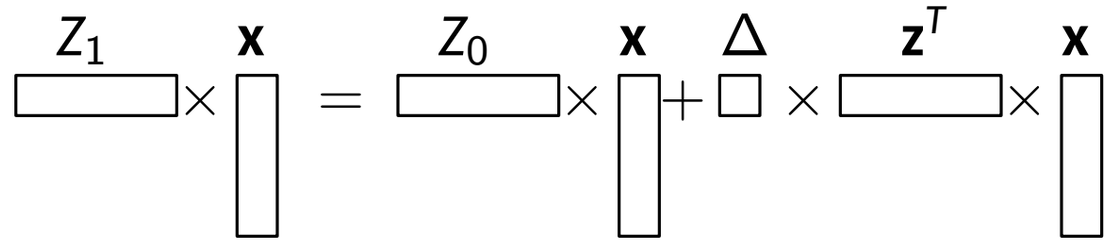

$$
\begin{aligned}
H(Z_1 \cdot \mathbf{x}) &= H(Z_0 \cdot \mathbf{x} + \Delta \cdot (\mathbf{z}^T \cdot \mathbf{x})) \\
&= \begin{cases} H(Z_1 \cdot \mathbf{x}) & \text{if } \mathbf{z}^T \cdot \mathbf{x} = 0 \\ \textit{Uniform} & \text{from RKA of H} \end{cases}
\end{aligned}
$$

# Using Symmetric-key Primitive For Extension

- The BCMPR construction requires modular exponentiation for every extension
- QuietOT proposed to use RO instead

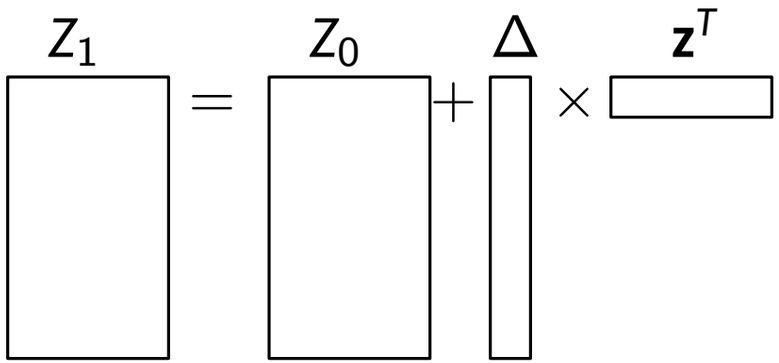



Eval at $x \leftarrow \mathcal{R}^n$



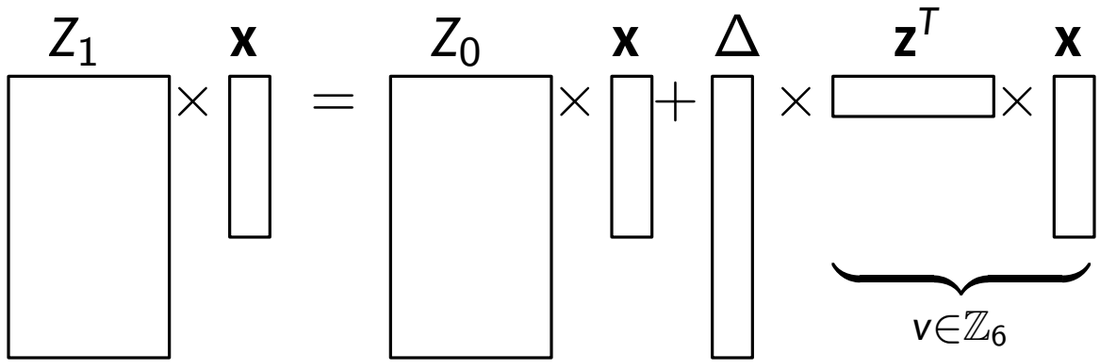

$$
\begin{aligned}
H(Z_1 \cdot \mathbf{x}) &= H(Z_0 \cdot \mathbf{x} + \Delta \cdot (\mathbf{z}^T \cdot \mathbf{x})) \\
&= \begin{cases} H(Z_1 \cdot \mathbf{x}) & \text{if } \mathbf{z}^T \cdot \mathbf{x} = 0 \\ \textit{Uniform} & \text{from RKA of H} \end{cases}
\end{aligned}
$$

- To ensure $\Delta$ has enough entropy, we can increase the dimension of $Z_0, Z_1$



# Shiftable CPRF and ListOT

- Suppose we use BIPSW
- $f_{\mathbf{z}}(\mathbf{x}) = \lfloor \langle \mathbf{x}, \mathbf{z} \rangle \mod 6 \rceil_2$




**ShCPRF for Inner-Product Predicates**

**Public Parameters.** Security parameter $\lambda$, finite ring $\mathcal{R}$ of order $\ell$, integers $m$ such that $m \geq \lambda$, vector length $n \geq 1$, and an RKA-secure PRF family $F\colon \mathcal{R}^m \times \mathcal{R}^n \to \mathcal{Y}$ for affine key-derivation functions.

ShCPRF.KeyGen($1^\lambda$):

1 : $\mathbf{k_0} \xleftarrow{R} \mathcal{R}^m,\ \Delta \xleftarrow{R} \mathcal{R}^m \setminus \{0\}$
2 : $\mathbf{Z_0} \xleftarrow{R} \mathcal{R}^{m \times n}$
3 : msk := $(\mathbf{k_0}, \mathbf{Z_0}, \Delta)$

ShCPRF.Constrain(msk, $\mathbf{z}$):

1 : **parse** msk = $(\mathbf{k_0}, \mathbf{Z_0}, \Delta)$
2 : $\mathbf{Z_1} \leftarrow \mathbf{Z_0} - \Delta \mathbf{z}^\top$
3 : **return** csk := $(\mathbf{k_0}, \mathbf{Z_1})$

ShCPRF.Eval(msk, $\mathbf{x}, \alpha$):

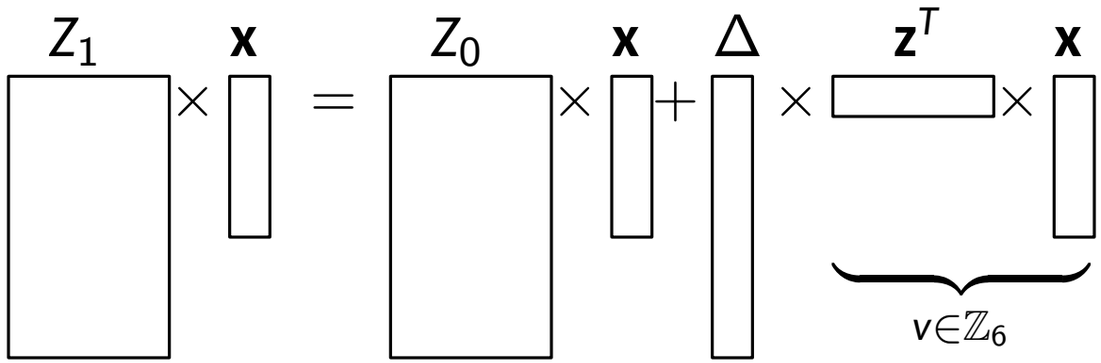
1 : **parse** msk = $(\mathbf{k_0}, \mathbf{Z_0}, \Delta)$
2 : $\mathbf{k} \leftarrow \mathbf{k_0} + \mathbf{Z_0 x} - \Delta \cdot \alpha$
3 : **return** $F_{\mathbf{k}}(\mathbf{x})$

ShCPRF.CEval(csk, $\mathbf{x}$):

1 : **parse** csk := $(\mathbf{k_0}, \mathbf{Z_1})$
2 : $\mathbf{k} \leftarrow \mathbf{k_0} + \mathbf{Z_1 x}$
3 : **return** $F_{\mathbf{k}}(\mathbf{x})$

# Shiftable CPRF and ListOT

- Suppose we use BIPSW
- $f_{\mathbf{z}}(\mathbf{x}) = \lfloor \langle \mathbf{x}, \mathbf{z} \rangle \mod 6 \rceil_2$




**ShCPRF for Inner-Product Predicates**

**Public Parameters.** Security parameter $\lambda$, finite ring $\mathcal{R}$ of order $\ell$, integers $m$ such that $m \geq \lambda$, vector length $n \geq 1$, and an RKA-secure PRF family $F\colon \mathcal{R}^m \times \mathcal{R}^n \to \mathcal{Y}$ for affine key-derivation functions.

ShCPRF.KeyGen($1^\lambda$):

1 : $\mathbf{k_0} \xleftarrow{R} \mathcal{R}^m,\ \Delta \xleftarrow{R} \mathcal{R}^m \setminus \{0\}$
2 : $\mathbf{Z_0} \xleftarrow{R} \mathcal{R}^{m \times n}$
3 : msk := $(\mathbf{k_0}, \mathbf{Z_0}, \Delta)$

ShCPRF.Constrain(msk, $\mathbf{z}$):

1 : **parse** msk = $(\mathbf{k_0}, \mathbf{Z_0}, \Delta)$
2 : $\mathbf{Z_1} \leftarrow \mathbf{Z_0} - \Delta \mathbf{z}^\top$
3 : **return** csk := $(\mathbf{k_0}, \mathbf{Z_1})$

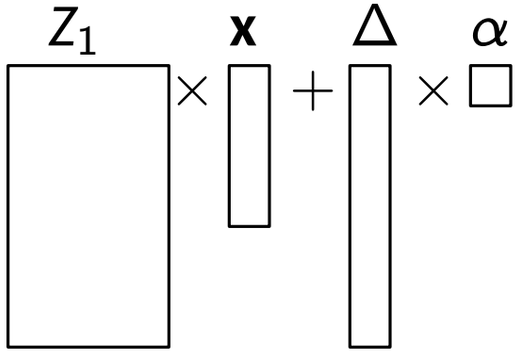
ShCPRF.Eval(msk, $\mathbf{x}, \alpha$):

1 : **parse** msk = $(\mathbf{k_0}, \mathbf{Z_0}, \Delta)$
2 : $\mathbf{k} \leftarrow \mathbf{k_0} + \mathbf{Z_0 x} - \Delta \cdot \alpha$
3 : **return** $F_{\mathbf{k}}(\mathbf{x})$

ShCPRF.CEval(csk, $\mathbf{x}$):

1 : **parse** csk := $(\mathbf{k_0}, \mathbf{Z_1})$
2 : $\mathbf{k} \leftarrow \mathbf{k_0} + \mathbf{Z_1 x}$
3 : **return** $F_{\mathbf{k}}(\mathbf{x})$

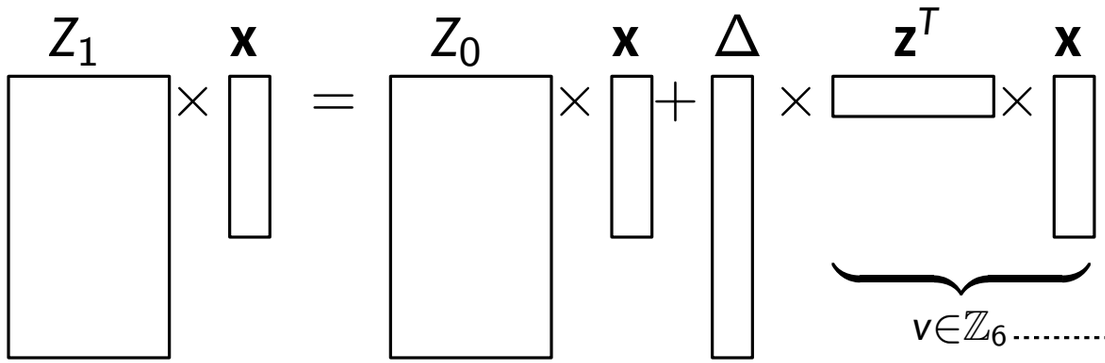
- Sender can **enumerate** over $\alpha \in \mathbb{Z}_6$ and computes




$$H(Z_1 \cdot \mathbf{x} + \Delta \cdot \alpha) = \begin{cases} L_0 = (H(Z_1 \cdot \mathbf{x} + \Delta \cdot \{0,1,2\})) \\ L_1 = (H(Z_1 \cdot \mathbf{x} + \Delta \cdot \{3,4,5\})) \end{cases}$$

# Shiftable CPRF and ListOT

- Suppose we use BIPSW
- $f_{\mathbf{z}}(\mathbf{x}) = \lfloor \langle \mathbf{x}, \mathbf{z} \rangle \mod 6 \rceil_2$




**ShCPRF for Inner-Product Predicates**

**Public Parameters.** Security parameter $\lambda$, finite ring $\mathcal{R}$ of order $\ell$, integers $m$ such that $m \geq \lambda$, vector length $n \geq 1$, and an RKA-secure PRF family $F\colon \mathcal{R}^m \times \mathcal{R}^n \to \mathcal{Y}$ for affine key-derivation functions.

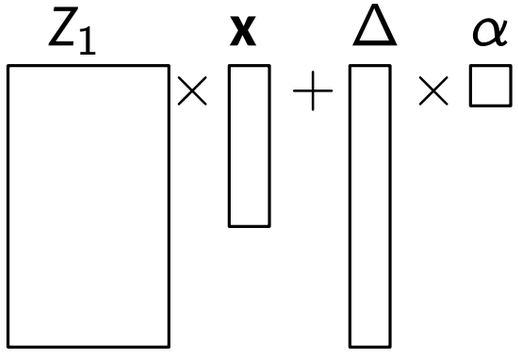
ShCPRF.KeyGen($1^\lambda$):
1 : $\mathbf{k_0} \xleftarrow{R} \mathcal{R}^m,\ \Delta \xleftarrow{R} \mathcal{R}^m \setminus \{0\}$
2 : $\mathbf{Z_0} \xleftarrow{R} \mathcal{R}^{m \times n}$
3 : msk := $(\mathbf{k_0}, \mathbf{Z_0}, \Delta)$

ShCPRF.Constrain(msk, $\mathbf{z}$):
1 : **parse** msk = $(\mathbf{k_0}, \mathbf{Z_0}, \Delta)$
2 : $\mathbf{Z_1} \leftarrow \mathbf{Z_0} - \Delta \mathbf{z}^\top$
3 : **return** csk := $(\mathbf{k_0}, \mathbf{Z_1})$

ShCPRF.Eval(msk, $\mathbf{x}, \alpha$):
1 : **parse** msk = $(\mathbf{k_0}, \mathbf{Z_0}, \Delta)$
2 : $\mathbf{k} \leftarrow \mathbf{k_0} + \mathbf{Z_0 x} - \Delta \cdot \alpha$
3 : **return** $F_{\mathbf{k}}(\mathbf{x})$

ShCPRF.CEval(csk, $\mathbf{x}$):
1 : **parse** csk := $(\mathbf{k_0}, \mathbf{Z_1})$
2 : $\mathbf{k} \leftarrow \mathbf{k_0} + \mathbf{Z_1 x}$
3 : **return** $F_{\mathbf{k}}(\mathbf{x})$

- Sender can **enumerate** over $\alpha \in \mathbb{Z}_6$ and computes

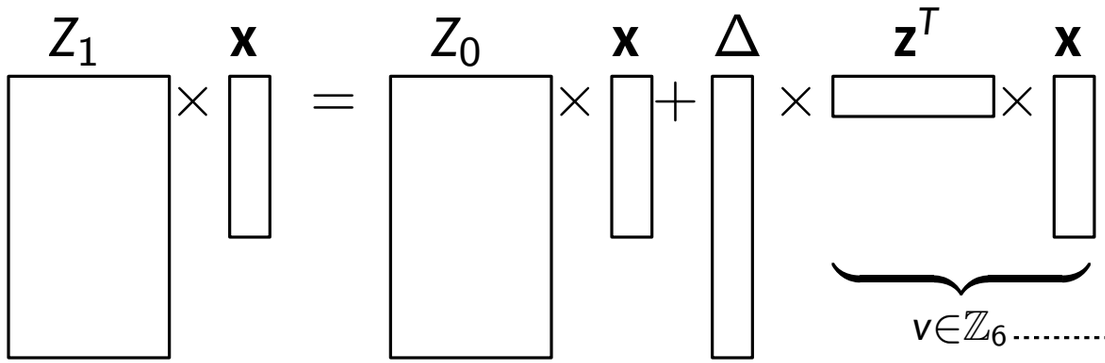



$$H(Z_1 \cdot \mathbf{x} + \Delta \cdot \alpha) = \begin{cases} L_0 = (H(Z_1 \cdot \mathbf{x} + \Delta \cdot \{0,1,2\})) \\ L_1 = (H(Z_1 \cdot \mathbf{x} + \Delta \cdot \{3,4,5\})) \end{cases}$$

"ListOT" $H(Z_0\mathbf{x}) = L_{\lfloor v \rceil_2}[v]$

# Shiftable CPRF and ListOT

- Suppose we use BIPSW
- $f_{\mathbf{z}}(\mathbf{x}) = \lfloor \langle \mathbf{x}, \mathbf{z} \rangle \mod 6 \rceil_2$




**ShCPRF for Inner-Product Predicates**

**Public Parameters.** Security parameter $\lambda$, finite ring $\mathcal{R}$ of order $\ell$, integers $m$ such that $m \geq \lambda$, vector length $n \geq 1$, and an RKA-secure PRF family $F\colon \mathcal{R}^m \times \mathcal{R}^n \to \mathcal{Y}$ for affine key-derivation functions.

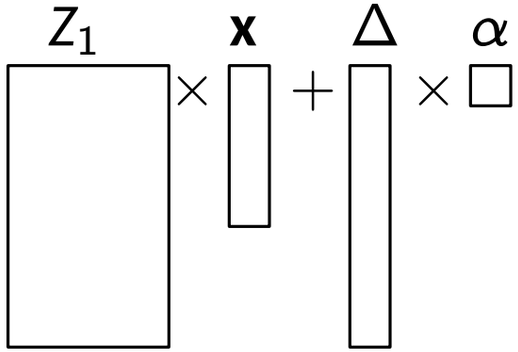
ShCPRF.KeyGen($1^\lambda$):
1 : $\mathbf{k_0} \xleftarrow{R} \mathcal{R}^m,\ \Delta \xleftarrow{R} \mathcal{R}^m \setminus \{0\}$
2 : $\mathbf{Z_0} \xleftarrow{R} \mathcal{R}^{m \times n}$
3 : msk := $(\mathbf{k_0}, \mathbf{Z_0}, \Delta)$

ShCPRF.Constrain(msk, $\mathbf{z}$):
1 : **parse** msk = $(\mathbf{k_0}, \mathbf{Z_0}, \Delta)$
2 : $\mathbf{Z_1} \leftarrow \mathbf{Z_0} - \Delta \mathbf{z}^\top$
3 : **return** csk := $(\mathbf{k_0}, \mathbf{Z_1})$

ShCPRF.Eval(msk, $\mathbf{x}, \alpha$):
1 : **parse** msk = $(\mathbf{k_0}, \mathbf{Z_0}, \Delta)$
2 : $\mathbf{k} \leftarrow \mathbf{k_0} + \mathbf{Z_0}\mathbf{x} - \Delta \cdot \alpha$
3 : **return** $F_{\mathbf{k}}(\mathbf{x})$

ShCPRF.CEval(csk, $\mathbf{x}$):
1 : **parse** csk := $(\mathbf{k_0}, \mathbf{Z_1})$
2 : $\mathbf{k} \leftarrow \mathbf{k_0} + \mathbf{Z_1}\mathbf{x}$
3 : **return** $F_{\mathbf{k}}(\mathbf{x})$

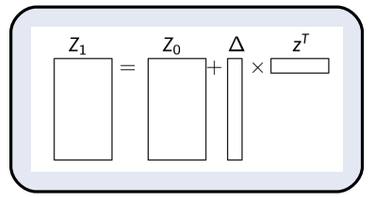
- Sender can **enumerate** over $\alpha \in \mathbb{Z}_6$ and computes




$$H(Z_1 \cdot \mathbf{x} + \Delta \cdot \alpha) = \begin{cases} L_0 = (H(Z_1 \cdot \mathbf{x} + \Delta \cdot \{0,1,2\})) \\ L_1 = (H(Z_1 \cdot \mathbf{x} + \Delta \cdot \{3,4,5\})) \end{cases}$$

"ListOT" $H(Z_0\mathbf{x}) = L_{\lfloor v \rceil_2}[v]$


Receiver





Sender

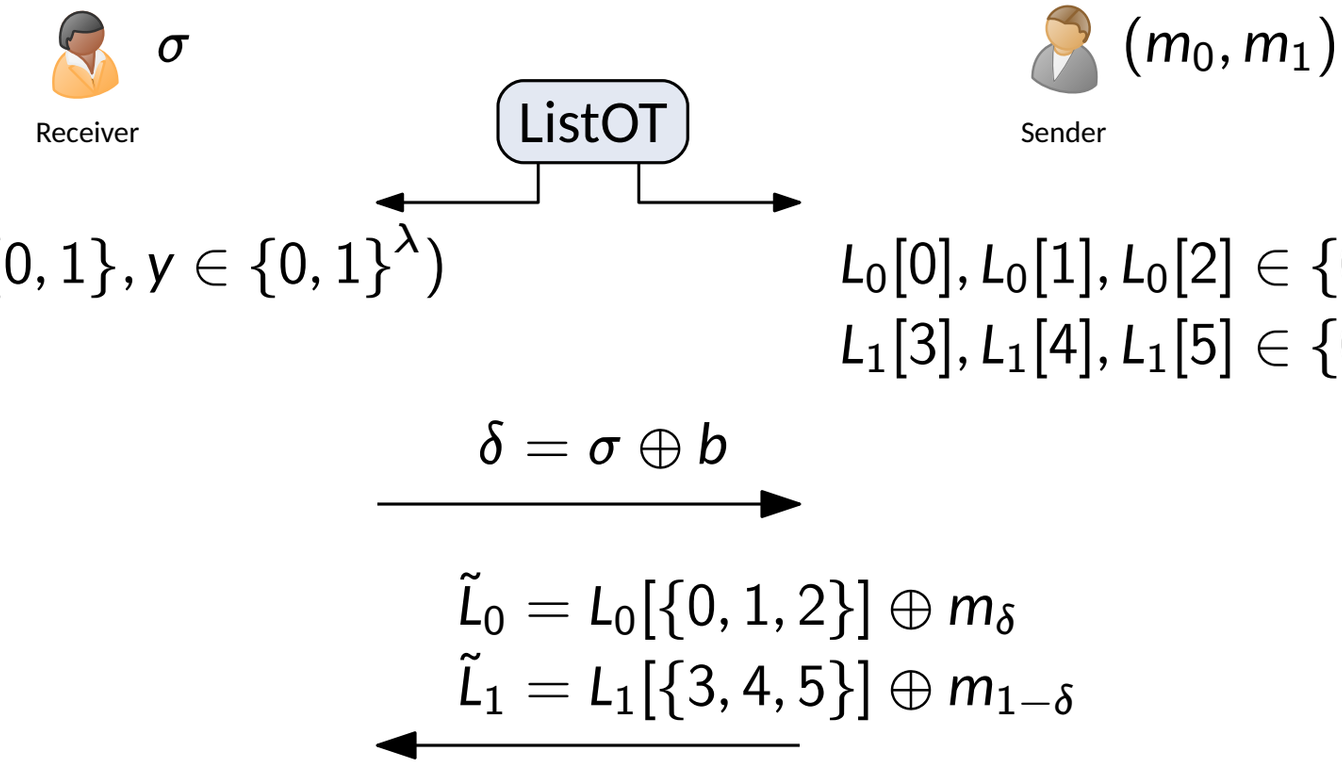
$csk = (Z_0, \mathbf{z}^T)$ $msk = (Z_1, \Delta)$

$\mathbf{x} \leftarrow \mathbb{Z}_6^n$

$(b, v, y)$ $y = L_b[v]$ $(L_0, L_1)$

# How to use ListOT

- Suppose Receiver wants to get $m_\sigma$ from $(m_0, m_1)$




Recover $m_\sigma = L_b[v] \oplus \tilde{L}_b[v] = L_b[v] \oplus L_b[v] \oplus m_{\delta \oplus b} = m_\sigma$

# Public Key Setup

- PKS = 1-round of PK exchanging
- Local rounding to remove error
- Requires **super-polynomial** modulus-to-noise ratio

$\mathsf{Gen}(1^\lambda, S, \mathsf{ltsk} =: \Delta)$:

1 : $s_0^1, \ldots, s_0^m \xleftarrow{R} \chi$
2 : $\mathsf{sk}_S := (\Delta, s_0^1, \ldots, s_0^m)$
3 : $e_0^1, \ldots, e_0^m \xleftarrow{R} \chi$
4 : **foreach** $i \in [m]$ :
5 : $\quad \mathsf{pk}_S^i := \Delta_i \cdot a_0 + s_0^i a_1 + e_0^i$
6 : $\mathbf{k}_0 \xleftarrow{R} \mathcal{R}^m$
7 : **return** $(\mathsf{pk}_S := (\mathbf{k}_0, (\mathsf{pk}_S^i)_{i \le m}), \mathsf{sk}_S)$

$\mathsf{Gen}(1^\lambda, R, C_{\mathbf{z}})$:

1 : **parse** $\mathbf{z} \in \mathcal{R}^n$ from $C_{\mathbf{z}}$
2 : $s_1 \xleftarrow{R} \chi$
3 : $\mathsf{sk}_R := (\mathbf{z}, s_1)$
4 : let $z \in \mathcal{P}$ have coeffs. $\frac{q}{t} \cdot (\mathbf{z} \| 0^{\eta - n})$
5 : $e_1, e_1' \xleftarrow{R} \chi$
6 : $\mathsf{pk}_R := (z + s_1 a_0 + e_1, s_1 a_1 + e_1')$
7 : **return** $(\mathsf{pk}_R, \mathsf{sk}_R)$

$\mathsf{KeyDer}(S, \mathsf{sk}_S, \mathsf{pk}_R)$:

1 : **parse** $\mathsf{sk}_S = (\Delta, s_0^1, \ldots, s_0^m)$
2 : **foreach** $i \in [m]$ :
3 : $\quad z_0^i \leftarrow \left\langle \mathsf{pk}_R, \left(\Delta_i, s_0^i\right) \right\rangle$
4 : $\quad$ **parse** $z_0^i \in \mathcal{P}$ as $\tilde{\mathbf{z}}_0^i \in \mathbb{Z}_q^n$
5 : $\quad$ **round** $\mathbf{z}_0^i = \left\lceil \tilde{\mathbf{z}}_0^i \right\rfloor_t \in \mathbb{Z}_t^n$
6 : $\mathsf{esk} = (\mathbf{k}_0, \mathbf{Z}_0 = (\mathbf{z}_0^i)_{i \le m})$
7 : **return** $K_S := (\Delta, \mathsf{esk})$

$\mathsf{KeyDer}(R, \mathsf{sk}_R, \mathsf{pk}_S)$:

1 : **parse** $\mathsf{sk}_R = (\mathbf{z}, s_1)$, $\mathsf{pk}_S := (\mathbf{k}_0, (\mathsf{pk}_S^i)_{i \le m})$
2 : **foreach** $i \in [m]$ :
3 : $\quad z_1^i \leftarrow \mathsf{pk}_S^i \cdot s_1$
4 : $\quad$ **parse** $z_1^i$ as $\tilde{\mathbf{z}}_1^i \in \mathbb{Z}_q^n$
5 : $\quad$ **round** $\mathbf{z}_1^i = \left\lceil \tilde{\mathbf{z}}_1^i \right\rfloor_t \in \mathbb{Z}_t^n$
6 : $\mathsf{csk} := (\mathbf{k}_0, \mathbf{Z}_1 := (\mathbf{z}_1^i)_{i \le m})$
7 : **return** $K_R := (\mathsf{csk}, \mathbf{z})$

- Correctness

$$\begin{aligned}
&\langle \mathsf{pk}_R, (\Delta_i, s_0^i) \rangle - (\mathsf{pk}_S^i \cdot s_1) \\
&= \Delta_i \cdot z + \Delta_i \cdot a_0 s_1 + \Delta_i \cdot e_1 + s_0^i a_1 s_1 + s_0^i e_1' - \Delta_i \cdot a_0 s_1 - s_0^i a_1 s_1 - e_0^i s_1 \\
&= \Delta_i \cdot z + \underbrace{\Delta_i \cdot e_1 + s_0^i e_1' - e_0^i s_1}_{\text{noise}} \approx \Delta_i \cdot z.
\end{aligned}$$

**Lemma 3 (Rounding of noisy secret shares).** *Let $(t, q)$ be two integers such that $t$ divides $q$. Fix any $z \in \mathbb{Z}_q$ and let $(z_0, z_1)$ be any two random elements of $\mathbb{Z}_q$ subject to $z_0 + z_1 = (q/t) \cdot z + e \bmod q$, where $e$ is such that $q/(t \cdot |e|) \ge C$. Then with probability at least $1 - (|e| + 1) \cdot t/q \ge 1 - C$, it holds that $\lfloor z_0 \rceil_t + \lfloor z_1 \rceil_t = z \bmod t$ and the probability is over the random choice of $(z_0, z_1) \in \mathbb{Z}_q \times \mathbb{Z}_q$.*

- Security: Only semi-honest security is proved

# Experiments

- The authors tested the Extension process (without base OT)

| | $\lvert pk_{sender} \rvert$ | $\lvert pk_{receiver} \rvert$ | OT/s (M1 Pro) | OT/s (c5.metal) | OT/s (t2.small) | Bits/OT |
|---|---|---|---|---|---|---|
| IKNP | | | 2,592,000 | 34,174,000 | 12,264,000 | 128 |
| SoftSpoken ($k = 2$) | | | 2,732,000 | 52,676,000 | 33,121,000 | 64 |
| SoftSpoken ($k = 4$) | | | 1,636,000 | 44,443,000 | 27,504,000 | 32 |
| SoftSpoken ($k = 8$) | | | 249,000 | 9,500,000 | 5,891,000 | 16 |
| SoftSpoken ($k = 16$) | | | 2,000 | 76,000 | 49,000 | 8 |
| RRT | | | 1,230,000 | 6,856,000 | 2,492,000 | 3 |
| OSY | 50kB | 1kB | 0.6 | 0.5 | 0.3 | 3 |
| BCMPR (BIPSW IPM-wPRF) | 63kB | 72kB | 65,000 | 52,000 | 29,000 | 3 |
| BCMPR (GAR IPM-wPRF) | 33kB | 38kB | 45,000 | 35,000 | 20,000 | 3 |
| QuietOT (BIPSW IPM-wPRF) | 5.4MB | 84kB | 1,115,000 | 567,000 | 304,000 | 7 |
| with AVX512 support | | | N/A | 1,265,000 | N/A | 7 |
| QuietOT (GAR IPM-wPRF) | 5.6MB | 88kB | 781,000 | 255,000 | 169,000 | 33 |

**Table 2:** OTs per second on a single core generated by the sender. Note that libOTe is not optimized for M1 since the AVX instructions are not available on M1 processors, hence we report these numbers in gray. The GAR IPM-wPRF cannot benefit from AVX due to limited bit-slicing opportunities. Setup costs are excluded.

# Any Questions?

- Offers a new directions to PCG/PCF research ✓
- Both BCMPR24 and this work are not technically amazing ✗
- CPRF + IPM-wPRF + ELF $\approx$ DV-NIZK [PsV06]
- The Public Key Setup is very similar to HSS, we could try to achieve malicious security
- The Input space in the BIPSW-varient of QuietOT is $\{0,1\}^n$ rather than $\mathbb{Z}_6^n$