

New Pseudorandom Correlation Functions and Exponent VRFs from DCR

To appear at Asiacrypt 2026



Archisman Dutta



AARHUS UNIVERSITY



Lisa Kohl



Lawrence Roy



Peter Scholl



AARHUS UNIVERSITY

NordriCrypt Fall 2026
30 October 2026

Secure multiparty computation



Alice

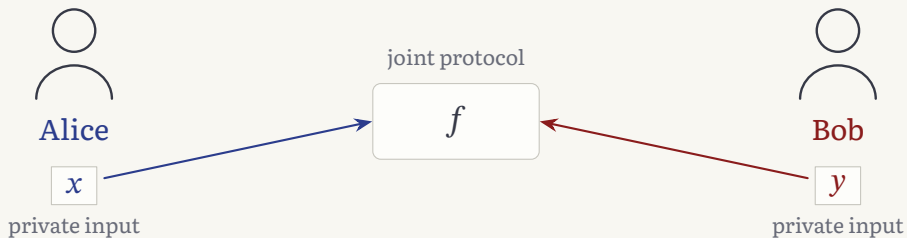
x

private input

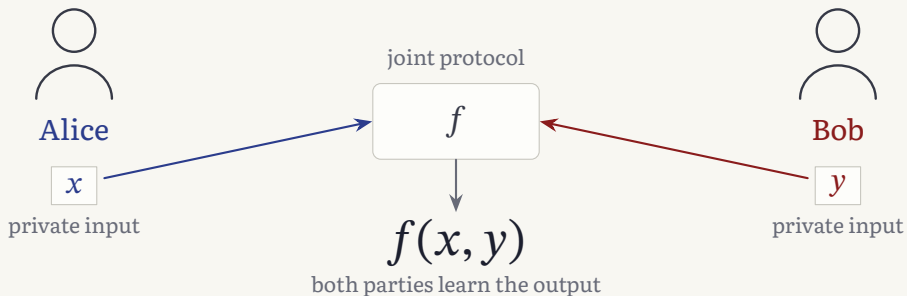
Secure multiparty computation



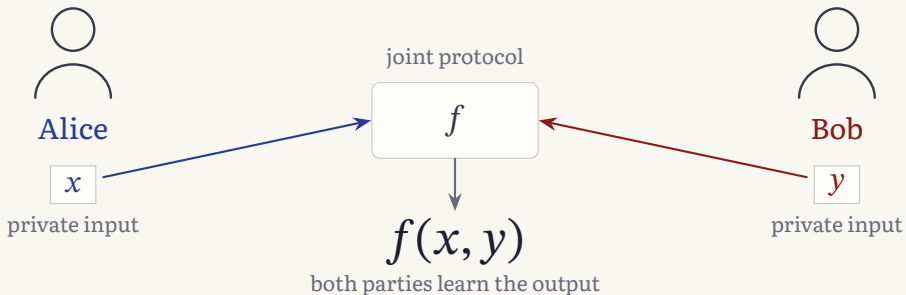
Secure multiparty computation



Secure multiparty computation



Secure multiparty computation



Each party learns $f(x, y)$ and nothing more about the other input.

Preprocessing and online phases



Alice

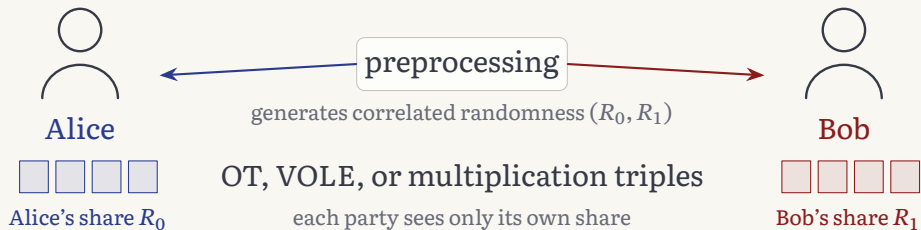


Bob

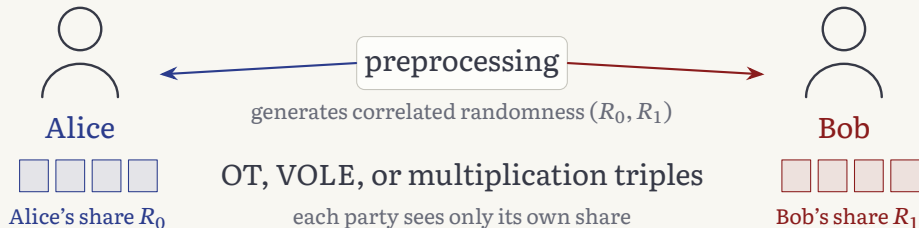
Preprocessing and online phases



Preprocessing and online phases



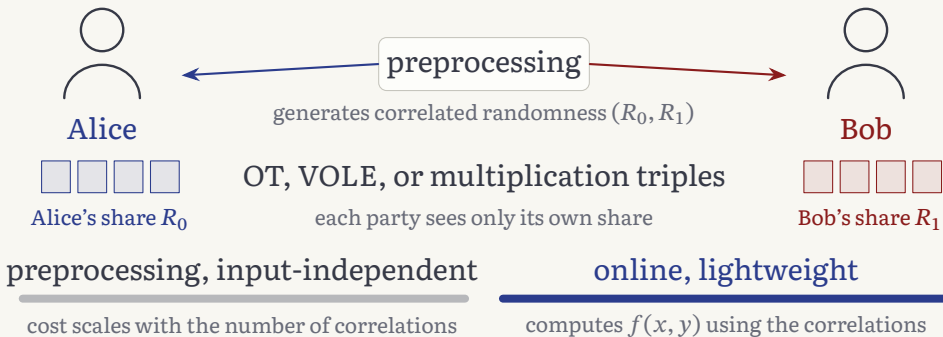
Preprocessing and online phases



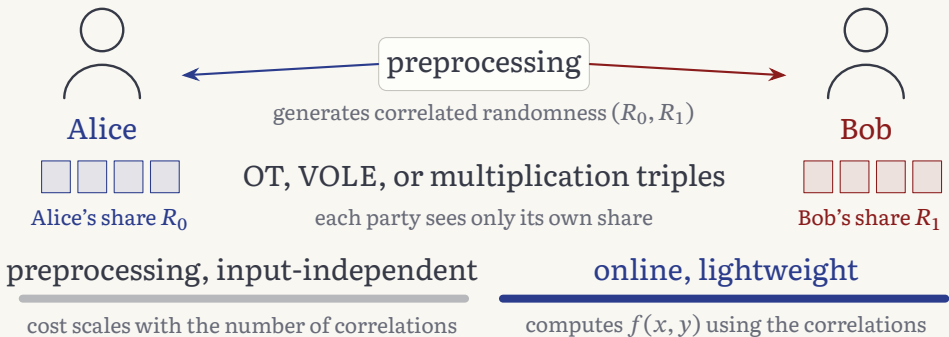
preprocessing, input-independent

cost scales with the number of correlations

Preprocessing and online phases



Preprocessing and online phases



Pseudorandom correlation functions compress this into short seeds.

Pseudorandom correlation functions

[BCG+20a]

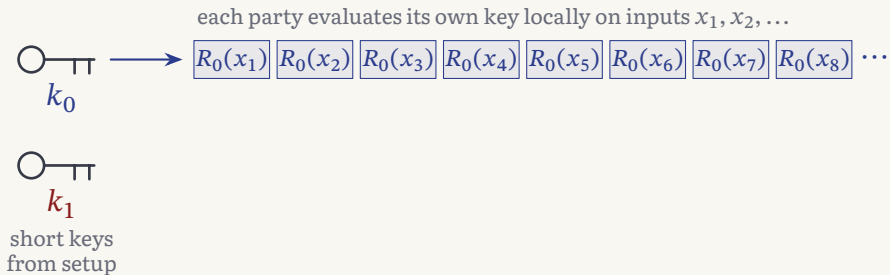

 k_0


 k_1

short keys
from setup

Pseudorandom correlation functions

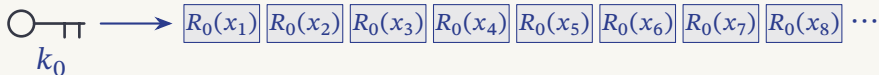
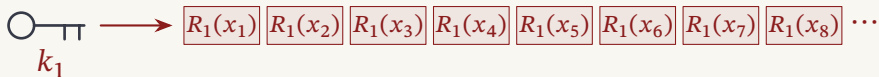
[BCG+20a]



Pseudorandom correlation functions

[BCG+20a]

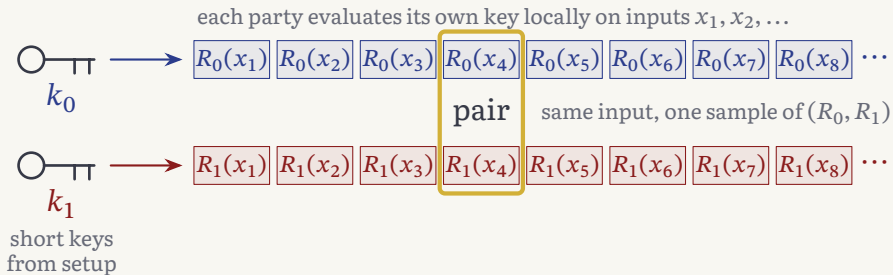
each party evaluates its own key locally on inputs $x_1, x_2, \dots$



short keys
from setup

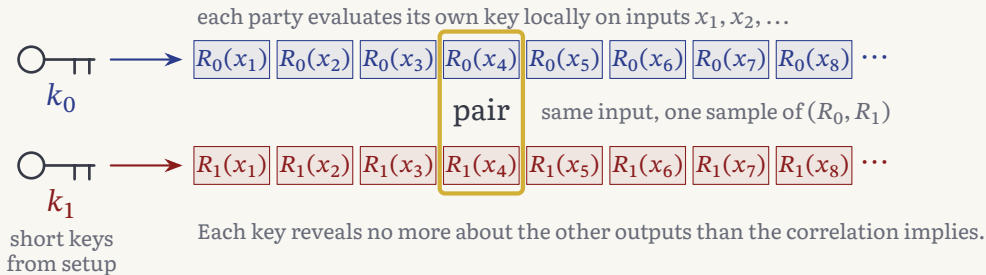
Pseudorandom correlation functions

[BCG+20a]



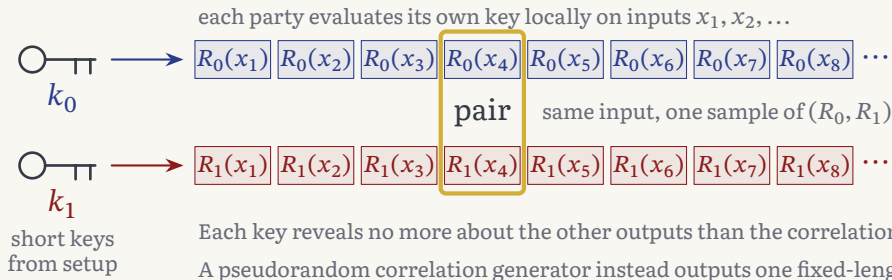
Pseudorandom correlation functions

[BCG+20a]



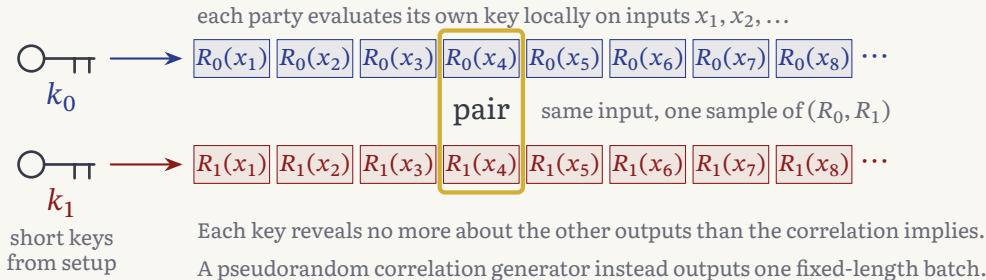
Pseudorandom correlation functions

[BCG+20a]



Pseudorandom correlation functions

[BCG+20a]



Our weak PCF needs random inputs, so we hash with a random oracle.

Correlated oblivious transfer

Messages lie in $\mathbb{Z}_N$ for an RSA modulus N , and the secret offset Δ is the same in every sample.

Correlated oblivious transfer

Messages lie in $\mathbb{Z}_N$ for an RSA modulus N , and the secret offset Δ is the same in every sample.

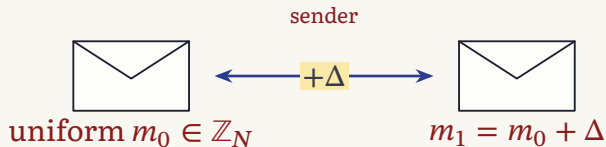
sender



uniform $m_0 \in \mathbb{Z}_N$

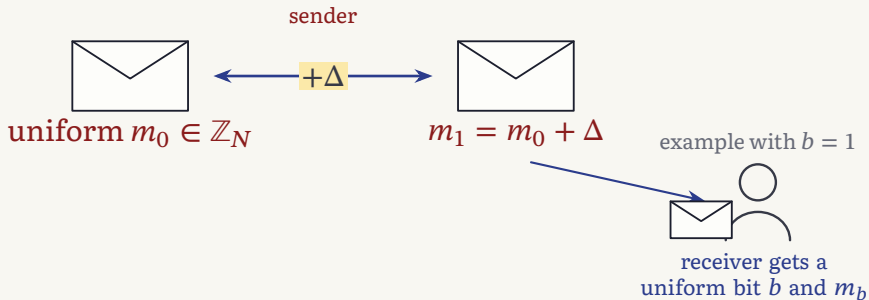
Correlated oblivious transfer

Messages lie in $\mathbb{Z}_N$ for an RSA modulus N , and the secret offset Δ is the same in every sample.



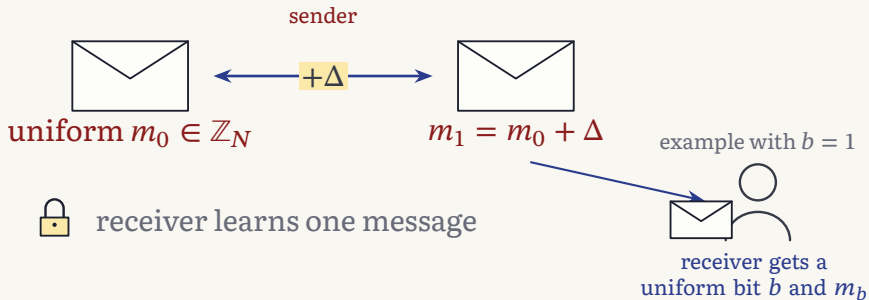
Correlated oblivious transfer

Messages lie in $\mathbb{Z}_N$ for an RSA modulus N , and the secret offset Δ is the same in every sample.



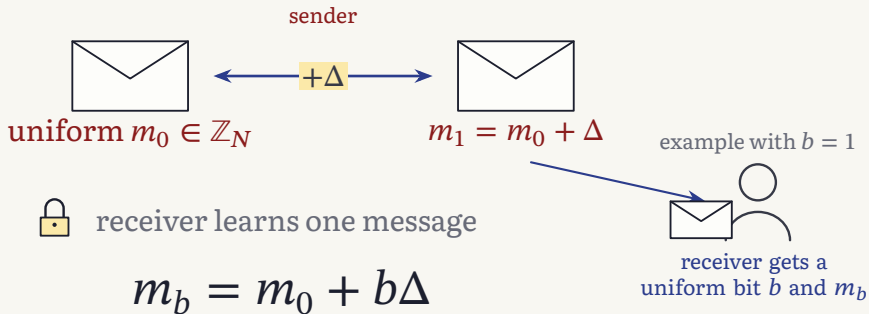
Correlated oblivious transfer

Messages lie in $\mathbb{Z}_N$ for an RSA modulus N , and the secret offset Δ is the same in every sample.



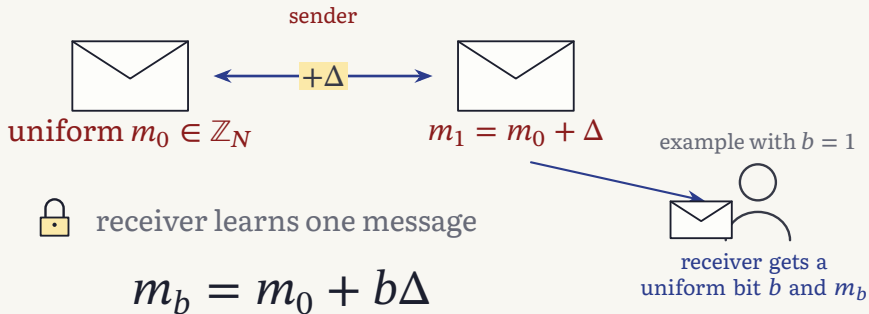
Correlated oblivious transfer

Messages lie in $\mathbb{Z}_N$ for an RSA modulus N , and the secret offset Δ is the same in every sample.



Correlated oblivious transfer

Messages lie in $\mathbb{Z}_N$ for an RSA modulus N , and the secret offset Δ is the same in every sample.



The sender does not learn b , and the receiver does not learn Δ .

Prior work

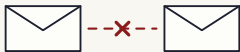
constrained Naor-Reingold [BCM+24]



fast random OT

Prior work

constrained Naor–Reingold [BCM+24]

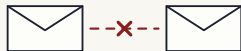


fast random OT

no fixed offset $m_1 - m_0$

Prior work

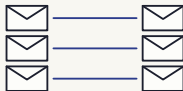
constrained Naor–Reingold [BCM+24]



fast random OT

no fixed offset $m_1 - m_0$

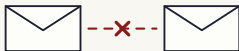
sparse LPN [BCM+25]



fast COT and VOLE

Prior work

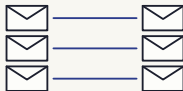
constrained Naor–Reingold [BCM+24]



fast random OT

no fixed offset $m_1 - m_0$

sparse LPN [BCM+25]

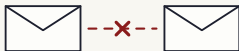


fast COT and VOLE

expensive, non-black-box
distributed key generation

Prior work

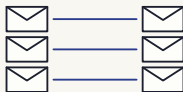
constrained Naor–Reingold [BCM+24]



fast random OT

no fixed offset $m_1 - m_0$

sparse LPN [BCM+25]



fast COT and VOLE

expensive, non-black-box
distributed key generation

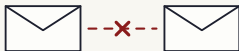
this work



COT over $\mathbb{Z}_N$

Prior work

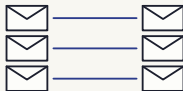
constrained Naor–Reingold [BCM+24]



fast random OT

no fixed offset $m_1 - m_0$

sparse LPN [BCM+25]



fast COT and VOLE

expensive, non-black-box
distributed key generation

this work



COT over $\mathbb{Z}_N$

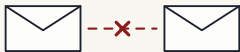


VOLE, any modulus

much slower receiver evaluation

Prior work

constrained Naor-Reingold [BCM+24]



fast random OT

no fixed offset $m_1 - m_0$

sparse LPN [BCM+25]



fast COT and VOLE

expensive, non-black-box
distributed key generation

this work



COT over $\mathbb{Z}_N$



VOLE, any modulus

much slower receiver evaluation



Our setup is two rounds of OT, then evaluation is local.

PCF for OT from constrained PRFs

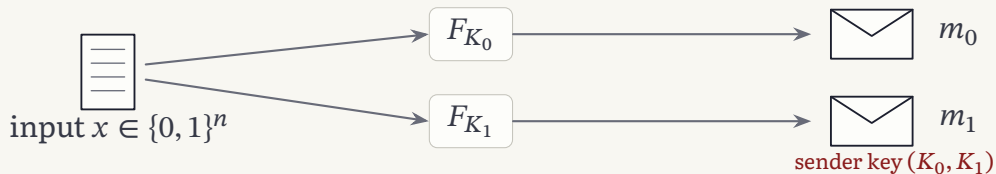
[BCM+24]



input $x \in \{0, 1\}^n$

PCF for OT from constrained PRFs

[BCM+24]

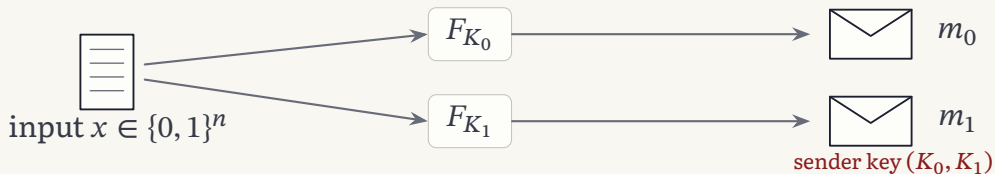


outer PRF $F_K(x) = g^{a_0 \prod_{i:x_i=1} a_i}$ with $K = (a_0, \dots, a_n)$ and g of prime order

PCF for OT from constrained PRFs

[BCM+24]

The receiver's bit is $b = f_z(x)$, for an inner PRF f_z with secret key z . Here $b = 0$.

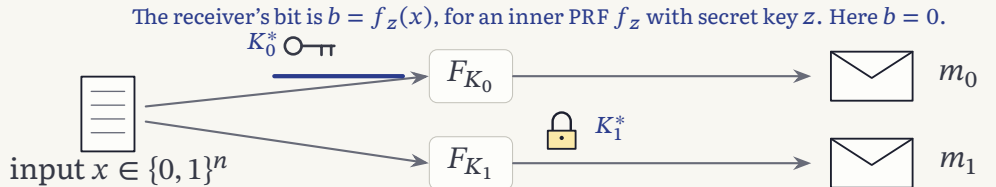


outer PRF $F_K(x) = g^{a_0 \prod_{i: x_i=1} a_i}$ with $K = (a_0, \dots, a_n)$ and g of prime order

inner PRF $f_z(x) = \chi_S(\langle z, x \rangle)$ with χ_S the indicator function of a public set $S \subseteq \{0, \dots, n\}$

PCF for OT from constrained PRFs

[BCM+24]



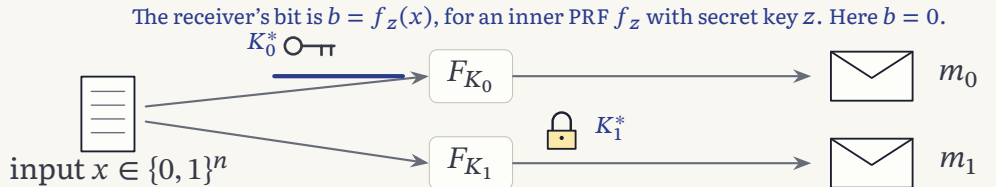
The receiver's constrained key K_b^* evaluates F_{K_b} only where $f_z(x) = b$. sender key (K_0, K_1)

outer PRF $F_K(x) = g^{a_0 \prod_{i: x_i=1} a_i}$ with $K = (a_0, \dots, a_n)$ and g of prime order

inner PRF $f_z(x) = \chi_S(\langle z, x \rangle)$ with χ_S the indicator function of a public set $S \subseteq \{0, \dots, n\}$

PCF for OT from constrained PRFs

[BCM+24]



The receiver's constrained key K_b^* evaluates F_{K_b} only where $f_z(x) = b$.

outer PRF $F_K(x) = g^{a_0 \prod_{i: x_i=1} a_i}$ with $K = (a_0, \dots, a_n)$ and g of prime order

inner PRF $f_z(x) = \chi_S(\langle z, x \rangle)$ with χ_S the indicator function of a public set $S \subseteq \{0, \dots, n\}$



Independent keys give random OT. COT needs correlated keys K_0, K_1 .

Inner-product-membership weak PRF

For a public $S \subseteq \{0, \dots, n\}$ and a secret $z \in \{0, 1\}^n$, let $f_z(x) = \chi_S(\langle z, x \rangle)$.
An inner-product-membership weak PRF is such an f_z that looks random on random inputs x .

Inner-product-membership weak PRF

For a public $S \subseteq \{0, \dots, n\}$ and a secret $z \in \{0, 1\}^n$, let $f_z(x) = \chi_S(\langle z, x \rangle)$.

An inner-product-membership weak PRF is such an f_z that looks random on random inputs x .

public input x 1 1 0 1 1 0

receiver's secret z 1 0 1 1 0 0

Inner-product-membership weak PRF

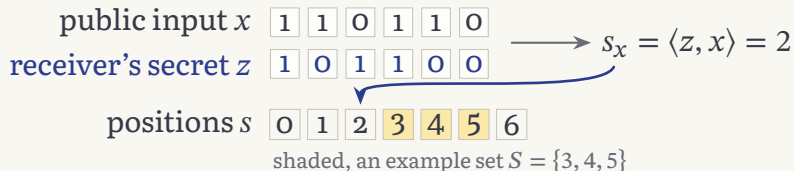
For a public $S \subseteq \{0, \dots, n\}$ and a secret $z \in \{0, 1\}^n$, let $f_z(x) = \chi_S(\langle z, x \rangle)$.

An inner-product-membership weak PRF is such an f_z that looks random on random inputs x .

public input x	<table border="1"><tr><td>1</td><td>1</td><td>0</td><td>1</td><td>1</td><td>0</td></tr></table>	1	1	0	1	1	0	$\longrightarrow s_x = \langle z, x \rangle = 2$
1	1	0	1	1	0			
receiver's secret z	<table border="1"><tr><td>1</td><td>0</td><td>1</td><td>1</td><td>0</td><td>0</td></tr></table>	1	0	1	1	0	0	
1	0	1	1	0	0			

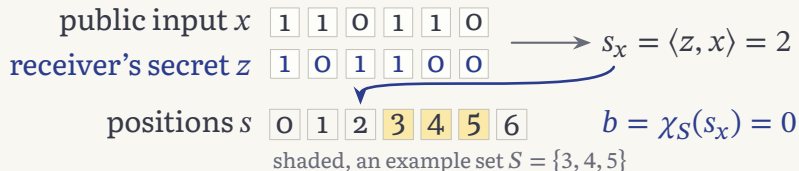
Inner-product-membership weak PRF

For a public $S \subseteq \{0, \dots, n\}$ and a secret $z \in \{0, 1\}^n$, let $f_z(x) = \chi_S(\langle z, x \rangle)$.
An inner-product-membership weak PRF is such an f_z that looks random on random inputs x .



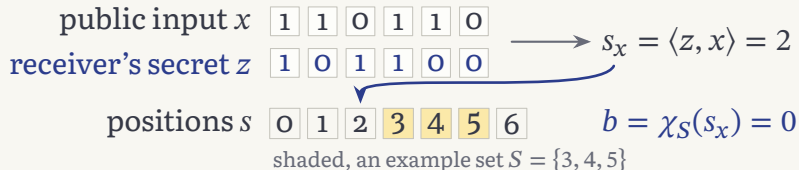
Inner-product-membership weak PRF

For a public $S \subseteq \{0, \dots, n\}$ and a secret $z \in \{0, 1\}^n$, let $f_z(x) = \chi_S(\langle z, x \rangle)$.
An inner-product-membership weak PRF is such an f_z that looks random on random inputs x .



Inner-product-membership weak PRF

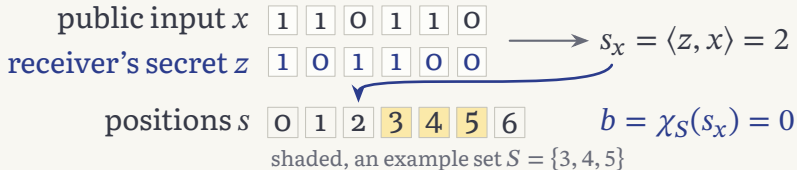
For a public $S \subseteq \{0, \dots, n\}$ and a secret $z \in \{0, 1\}^n$, let $f_z(x) = \chi_S(\langle z, x \rangle)$.
An inner-product-membership weak PRF is such an f_z that looks random on random inputs x .



Here $b = 0$, so the receiver outputs $(0, m_0)$. If $s_x \in S$, it outputs $(1, m_1)$.

Inner-product-membership weak PRF

For a public $S \subseteq \{0, \dots, n\}$ and a secret $z \in \{0, 1\}^n$, let $f_z(x) = \chi_S(\langle z, x \rangle)$.
An inner-product-membership weak PRF is such an f_z that looks random on random inputs x .



Here $b = 0$, so the receiver outputs $(0, m_0)$. If $s_x \in S$, it outputs $(1, m_1)$.



The sender's key has no z , so to the sender b looks random.

Constraining Naor–Reingold

[BCM+24]

Key $K = (a_0, \dots, a_n)$, input x , receiver secret z , and $s_x = \langle z, x \rangle$.

Constraining Naor–Reingold

[BCM+24]

Key $K = (a_0, \dots, a_n)$, input x , receiver secret z , and $s_x = \langle z, x \rangle$.

Sample r and set $\alpha_i = a_i r^{-z_i}$, $h_s = g^{a_0 r^s}$, $e_x = \prod_{i: x_i=1} \alpha_i$.

Constraining Naor–Reingold

[BCM+24]

Key $K = (a_0, \dots, a_n)$, input x , receiver secret z , and $s_x = \langle z, x \rangle$.

Sample r and set $\alpha_i = a_i r^{-z_i}$, $h_s = g^{a_0 r^s}$, $e_x = \prod_{i: x_i=1} \alpha_i$.

$$F_K(x) = g^{a_0 \prod_{i: x_i=1} \alpha_i r^{z_i}} = g^{a_0 r^{s_x} \prod_{i: x_i=1} \alpha_i} = h_{s_x}^{e_x}$$

Constraining Naor–Reingold

[BCM+24]

Key $K = (a_0, \dots, a_n)$, input x , receiver secret z , and $s_x = \langle z, x \rangle$.

Sample r and set $\alpha_i = a_i r^{-z_i}$, $h_s = g^{a_0 r^s}$, $e_x = \prod_{i: x_i=1} \alpha_i$.

$$F_K(x) = g^{a_0 \prod_{i: x_i=1} \alpha_i r^{z_i}} = g^{a_0 r^{s_x} \prod_{i: x_i=1} \alpha_i} = h_{s_x}^{e_x}$$

The receiver gets all α_i , but only the table entries h_s with $\chi_S(s) = 0$.

Constraining Naor–Reingold

[BCM+24]

Key $K = (a_0, \dots, a_n)$, input x , receiver secret z , and $s_x = \langle z, x \rangle$.

Sample r and set $\alpha_i = a_i r^{-z_i}$, $h_s = g^{a_0 r^s}$, $e_x = \prod_{i: x_i=1} \alpha_i$.

$$F_K(x) = g^{a_0 \prod_{i: x_i=1} \alpha_i r^{z_i}} = g^{a_0 r^{s_x} \prod_{i: x_i=1} \alpha_i} = h_{s_x}^{e_x}$$

The receiver gets all α_i , but only the table entries h_s with $\chi_S(s) = 0$.

It can compute $F_{K_0}(x)$ only when $f_Z(x) = 0$.

The constrained key for K_1 uses the entries with $\chi_S(s) = 1$.

Constraining Naor–Reingold

[BCM+24]

Key $K = (a_0, \dots, a_n)$, input x , receiver secret z , and $s_x = \langle z, x \rangle$.

Sample r and set $\alpha_i = a_i r^{-z_i}$, $h_s = g^{a_0 r^s}$, $e_x = \prod_{i: x_i=1} \alpha_i$.

$$F_K(x) = g^{a_0 \prod_{i: x_i=1} \alpha_i r^{z_i}} = g^{a_0 r^{s_x} \prod_{i: x_i=1} \alpha_i} = h_{s_x}^{e_x}$$

The receiver gets all α_i , but only the table entries h_s with $\chi_S(s) = 0$.

It can compute $F_{K_0}(x)$ only when $f_Z(x) = 0$.

The constrained key for K_1 uses the entries with $\chi_S(s) = 1$.



The receiver needs only one table entry and one exponent.

Distributed discrete logarithm

[OSY21, RS21]

For $N = pq$, every $U \in \mathbb{Z}_{N^2}^*$ is $U_0(1 + N)^\Delta$ with an N -th residue $U_0 = u^N$ and an offset $\Delta \in \mathbb{Z}_N$.

Distributed discrete logarithm

[OSY21, RS21]

For $N = pq$, every $U \in \mathbb{Z}_{N^2}^*$ is $U_0(1 + N)^\Delta$ with an N -th residue $U_0 = u^N$ and an offset $\Delta \in \mathbb{Z}_N$.



Distributed discrete logarithm

[OSY21, RS21]

For $N = pq$, every $U \in \mathbb{Z}_{N^2}^*$ is $U_0(1 + N)^\Delta$ with an N -th residue $U_0 = u^N$ and an offset $\Delta \in \mathbb{Z}_N$.



U

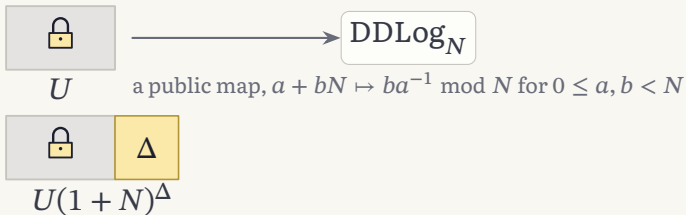


$U(1 + N)^\Delta$

Distributed discrete logarithm

[OSY21, RS21]

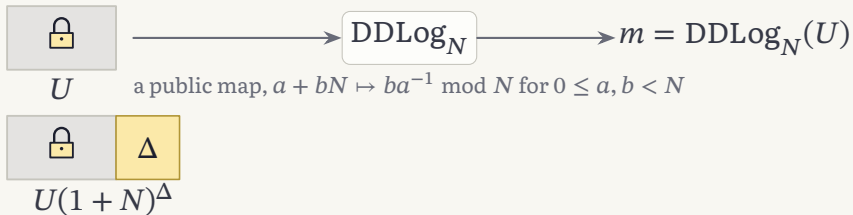
For $N = pq$, every $U \in \mathbb{Z}_{N^2}^*$ is $U_0(1 + N)^\Delta$ with an N -th residue $U_0 = u^N$ and an offset $\Delta \in \mathbb{Z}_N$.



Distributed discrete logarithm

[OSY21, RS21]

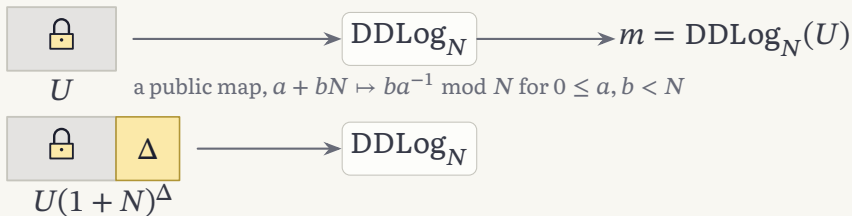
For $N = pq$, every $U \in \mathbb{Z}_{N^2}^*$ is $U_0(1 + N)^\Delta$ with an N -th residue $U_0 = u^N$ and an offset $\Delta \in \mathbb{Z}_N$.



Distributed discrete logarithm

[OSY21, RS21]

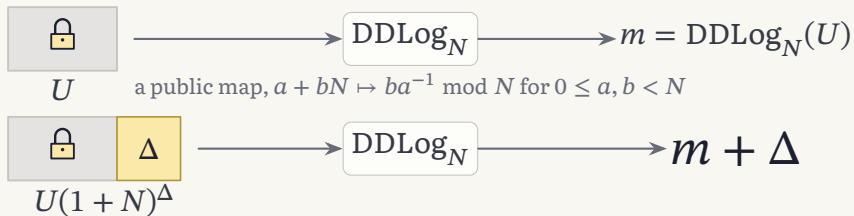
For $N = pq$, every $U \in \mathbb{Z}_{N^2}^*$ is $U_0(1 + N)^\Delta$ with an N -th residue $U_0 = u^N$ and an offset $\Delta \in \mathbb{Z}_N$.



Distributed discrete logarithm

[OSY21, RS21]

For $N = pq$, every $U \in \mathbb{Z}_{N^2}^*$ is $U_0(1 + N)^\Delta$ with an N -th residue $U_0 = u^N$ and an offset $\Delta \in \mathbb{Z}_N$.



Distributed discrete logarithm

[OSY21, RS21]

For $N = pq$, every $U \in \mathbb{Z}_{N^2}^*$ is $U_0(1 + N)^\Delta$ with an N -th residue $U_0 = u^N$ and an offset $\Delta \in \mathbb{Z}_N$.



$$\text{DDLog}_N(U(1 + N)^\Delta) = \text{DDLog}_N(U) + \Delta \pmod{N}$$

Multiplicative shares U and $U(1 + N)^\Delta$ of $(1 + N)^\Delta$ become additive shares m and $m + \Delta$ of Δ .

Distributed discrete logarithm

[OSY21, RS21]

For $N = pq$, every $U \in \mathbb{Z}_{N^2}^*$ is $U_0(1 + N)^\Delta$ with an N -th residue $U_0 = u^N$ and an offset $\Delta \in \mathbb{Z}_N$.



$$\text{DDLog}_N(U(1 + N)^\Delta) = \text{DDLog}_N(U) + \Delta \pmod{N}$$

Multiplicative shares U and $U(1 + N)^\Delta$ of $(1 + N)^\Delta$ become additive shares m and $m + \Delta$ of Δ .



DDLog converts multiplicative shares into additive shares modulo N .

One key with a secret shift

Replace K_0, K_1 by one key $K = (a_0, \dots, a_n)$ and a secret shift Δ .

One key with a secret shift

Replace K_0, K_1 by one key $K = (a_0, \dots, a_n)$ and a secret shift Δ .

$F_{K_0}(x) = F_K(x) = g^{A_x}$ with $A_x = a_0 \prod_{i: x_i=1} a_i$ and an N -th residue g

$$F_{K_1}(x) = F_{K,\Delta}(x) = F_K(x) (1 + N)^\Delta$$

One key with a secret shift

Replace K_0, K_1 by one key $K = (a_0, \dots, a_n)$ and a secret shift Δ .

$F_{K_0}(x) = F_K(x) = g^{A_x}$ with $A_x = a_0 \prod_{i: x_i=1} a_i$ and an N -th residue g

$F_{K_1}(x) = F_{K,\Delta}(x) = F_K(x) (1 + N)^\Delta$

$m_0 = \text{DDLog}_N(F_K(x)), \quad m_1 = \text{DDLog}_N(F_{K,\Delta}(x)) = m_0 + \Delta \quad \checkmark$

One key with a secret shift

Replace K_0, K_1 by one key $K = (a_0, \dots, a_n)$ and a secret shift Δ .

$F_{K_0}(x) = F_K(x) = g^{A_x}$ with $A_x = a_0 \prod_{i: x_i=1} a_i$ and an N -th residue g

$$F_{K_1}(x) = F_{K,\Delta}(x) = F_K(x) (1 + N)^\Delta$$

$$m_0 = \text{DDLog}_N(F_K(x)), \quad m_1 = \text{DDLog}_N(F_{K,\Delta}(x)) = m_0 + \Delta \quad \checkmark$$

The receiver gets the α_i and one table for all s , shifted only where $s \in S$.

$$h_s = g^{a_0 r^s} (1 + N)^{\chi_s(s) \Delta}$$

One key with a secret shift

Replace K_0, K_1 by one key $K = (a_0, \dots, a_n)$ and a secret shift Δ .

$F_{K_0}(x) = F_K(x) = g^{A_x}$ with $A_x = a_0 \prod_{i: x_i=1} a_i$ and an N -th residue g

$F_{K_1}(x) = F_{K,\Delta}(x) = F_K(x)(1+N)^\Delta$

$m_0 = \text{DDLog}_N(F_K(x)), \quad m_1 = \text{DDLog}_N(F_{K,\Delta}(x)) = m_0 + \Delta \quad \checkmark$

The receiver gets the α_i and one table for all s , shifted only where $s \in S$.

$h_s = g^{a_0 r^s} (1+N)^{\chi_s(s)\Delta}$

It raises h_{s_x} to e_x , hoping to obtain $g^{A_x}(1+N)^{b\Delta} = F_{K_b}(x)$ with $b = f_z(x)$.

One key with a secret shift

Replace K_0, K_1 by one key $K = (a_0, \dots, a_n)$ and a secret shift Δ .

$F_{K_0}(x) = F_K(x) = g^{A_x}$ with $A_x = a_0 \prod_{i: x_i=1} a_i$ and an N -th residue g

$$F_{K_1}(x) = F_{K,\Delta}(x) = F_K(x)(1+N)^\Delta$$

$$m_0 = \text{DDLog}_N(F_K(x)), \quad m_1 = \text{DDLog}_N(F_{K,\Delta}(x)) = m_0 + \Delta \quad \checkmark$$

The receiver gets the α_i and one table for all s , shifted only where $s \in S$.

$$h_s = g^{a_0 r^s} (1+N)^{\chi_s(s)\Delta}$$

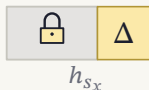
It raises h_{s_x} to e_x , hoping to obtain $g^{A_x}(1+N)^{b\Delta} = F_{K_b}(x)$ with $b = f_z(x)$.



The sender side works, but e_x also acts on the shift.

Exponents congruent to 1 modulo N

The receiver raises its table entry h_{s_x} to e_x . Here $s_x \in S$, so h_{s_x} carries the shift.



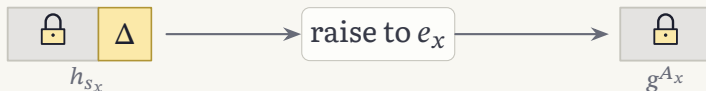
Exponents congruent to 1 modulo N

The receiver raises its table entry h_{s_x} to e_x . Here $s_x \in S$, so h_{s_x} carries the shift.



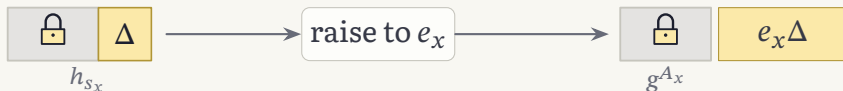
Exponents congruent to 1 modulo N

The receiver raises its table entry h_{s_x} to e_x . Here $s_x \in S$, so h_{s_x} carries the shift.



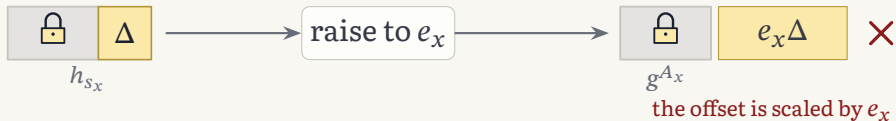
Exponents congruent to 1 modulo N

The receiver raises its table entry h_{s_x} to e_x . Here $s_x \in S$, so h_{s_x} carries the shift.



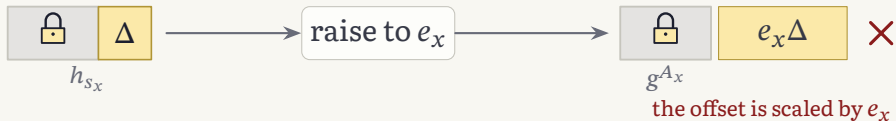
Exponents congruent to 1 modulo N

The receiver raises its table entry h_{s_x} to e_x . Here $s_x \in S$, so h_{s_x} carries the shift.



Exponents congruent to 1 modulo N

The receiver raises its table entry h_{s_x} to e_x . Here $s_x \in S$, so h_{s_x} carries the shift.

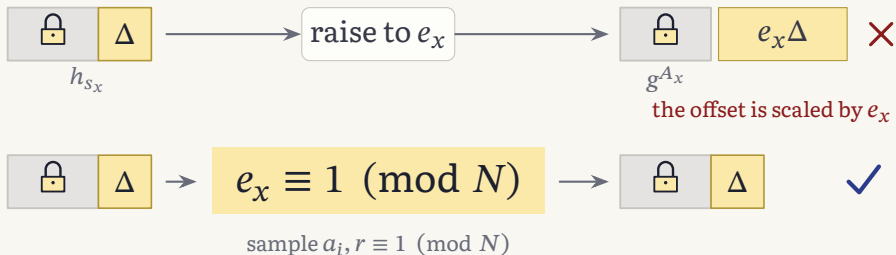


$$e_x \equiv 1 \pmod{N}$$

sample $a_i, r \equiv 1 \pmod{N}$

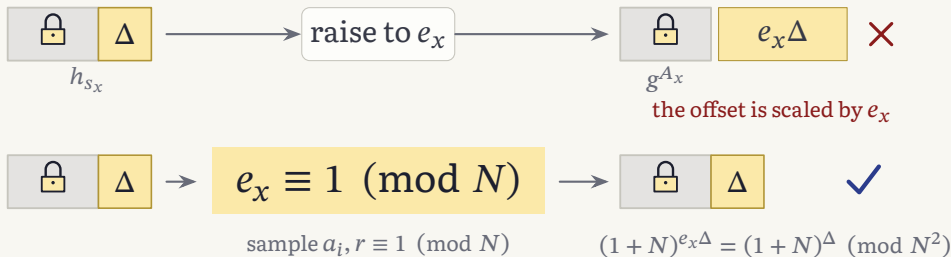
Exponents congruent to 1 modulo N

The receiver raises its table entry h_{s_x} to e_x . Here $s_x \in S$, so h_{s_x} carries the shift.



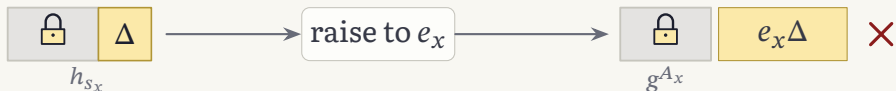
Exponents congruent to 1 modulo N

The receiver raises its table entry h_{s_x} to e_x . Here $s_x \in S$, so h_{s_x} carries the shift.

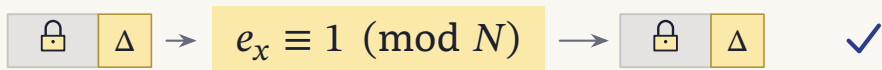


Exponents congruent to 1 modulo N

The receiver raises its table entry h_{s_x} to e_x . Here $s_x \in S$, so h_{s_x} carries the shift.



the offset is scaled by e_x



sample $a_i, r \equiv 1 \pmod{N}$

$$(1 + N)^{e_x \Delta} = (1 + N)^\Delta \pmod{N^2}$$



Now $h_{s_x}^{e_x} = g^{A_x} (1 + N)^{\chi_S(s_x) \Delta}$, so the receiver gets m_b .

Sparse-Power DCR

The receiver's table must hide Δ .

Sparse-Power DCR

The receiver's table must hide Δ .

If all entries are shifted, decisional composite residuosity (DCR) suffices.

By DCR, random N -th residues look like random units, and $r^S \equiv 1 \pmod{N}$ keeps Δ in every entry.

Sparse-Power DCR

The receiver's table must hide Δ .

If all entries are shifted, decisional composite residuosity (DCR) suffices.

By DCR, random N -th residues look like random units, and $r^S \equiv 1 \pmod{N}$ keeps Δ in every entry.

Our table shifts only the entries $s \in S$.

$$(g^{r^s})_s \approx_c (g^{r^s} (1 + N)^{\chi_{S(s)} \Delta})_s$$

Sparse-Power DCR, with $\approx_c$ computational indistinguishability and g a random N -th residue

Sparse-Power DCR

The receiver's table must hide Δ .

If all entries are shifted, decisional composite residuosity (DCR) suffices.

By DCR, random N -th residues look like random units, and $r^S \equiv 1 \pmod{N}$ keeps Δ in every entry.

Our table shifts only the entries $s \in S$.

$$(g^{r^s})_s \approx_c (g^{r^s} (1 + N) \chi_{s(s)\Delta})_s$$

Sparse-Power DCR, with $\approx_c$ computational indistinguishability and g a random N -th residue

We prove that DCR implies Sparse-Power DCR.

Sparse-Power DCR

The receiver's table must hide Δ .

If all entries are shifted, decisional composite residuosity (DCR) suffices.

By DCR, random N -th residues look like random units, and $r^S \equiv 1 \pmod{N}$ keeps Δ in every entry.

Our table shifts only the entries $s \in S$.

$$(g^{r^s})_s \approx_c (g^{r^s} (1 + N) \chi_{S^c}(s) \Delta)_s$$

Sparse-Power DCR, with $\approx_c$ computational indistinguishability and g a random N -th residue

We prove that DCR implies Sparse-Power DCR.

We also assume that f_Z is an inner-product-membership weak PRF.

Sparse-Power DCR

The receiver's table must hide Δ .

If all entries are shifted, decisional composite residuosity (DCR) suffices.

By DCR, random N -th residues look like random units, and $r^S \equiv 1 \pmod{N}$ keeps Δ in every entry.

Our table shifts only the entries $s \in S$.

$$(g^{r^s})_s \approx_c (g^{r^s} (1 + N) \chi_{s(s)\Delta})_s$$

Sparse-Power DCR, with $\approx_c$ computational indistinguishability and g a random N -th residue

We prove that DCR implies Sparse-Power DCR.

We also assume that f_Z is an inner-product-membership weak PRF.



Security rests on DCR and an inner-product-membership weak PRF.

Correctness and security

Both parties add a weak-PRF mask $F_k(x) \in \mathbb{Z}_N$, with k in k_0 and k_1 .

Correctness and security

Both parties add a weak-PRF mask $F_k(x) \in \mathbb{Z}_N$, with k in k_0 and k_1 .

Sender $m_0 = \text{DDLog}_N(g^{A_x}) + F_k(x), \quad m_1 = m_0 + \Delta$

Correctness and security

Both parties add a weak-PRF mask $F_k(x) \in \mathbb{Z}_N$, with k in k_0 and k_1 .

Sender $m_0 = \text{DDLog}_N(g^{Ax}) + F_k(x), \quad m_1 = m_0 + \Delta$

Receiver $\text{DDLog}_N(h_{s_x}^{e_x}) + F_k(x) = m_0 + b\Delta = m_b$

No assumption is needed here, only $e_x \equiv 1 \pmod{N}$ and the shift identity of DDLog.

Correctness and security

Both parties add a weak-PRF mask $F_k(x) \in \mathbb{Z}_N$, with k in k_0 and k_1 .

Sender $m_0 = \text{DDLog}_N(g^{Ax}) + F_k(x), \quad m_1 = m_0 + \Delta$

Receiver $\text{DDLog}_N(h_{s_x}^{e_x}) + F_k(x) = m_0 + b\Delta = m_b$

No assumption is needed here, only $e_x \equiv 1 \pmod{N}$ and the shift identity of DDLog.

Outputs b and m_0 look uniform

weak PRFs f_z and F_k

Correctness and security

Both parties add a weak-PRF mask $F_k(x) \in \mathbb{Z}_N$, with k in k_0 and k_1 .

Sender $m_0 = \text{DDLog}_N(g^{Ax}) + F_k(x), \quad m_1 = m_0 + \Delta$

Receiver $\text{DDLog}_N(h_{s_x}^{e_x}) + F_k(x) = m_0 + b\Delta = m_b$

No assumption is needed here, only $e_x \equiv 1 \pmod{N}$ and the shift identity of DDLog.

Outputs b and m_0 look uniform

weak PRFs f_z and F_k

Sender cannot learn b

weak PRF f_z , since k_1 has no z

Correctness and security

Both parties add a weak-PRF mask $F_k(x) \in \mathbb{Z}_N$, with k in k_0 and k_1 .

Sender $m_0 = \text{DDLog}_N(g^{Ax}) + F_k(x), \quad m_1 = m_0 + \Delta$

Receiver $\text{DDLog}_N(h_{s_x}^{e_x}) + F_k(x) = m_0 + b\Delta = m_b$

No assumption is needed here, only $e_x \equiv 1 \pmod{N}$ and the shift identity of DDLog.

Outputs b and m_0 look uniform

weak PRFs f_z and F_k

Sender cannot learn b

weak PRF f_z , since k_1 has no z

Receiver cannot learn Δ or m_{1-b}

Sparse-Power DCR, from DCR

Correctness and security

Both parties add a weak-PRF mask $F_k(x) \in \mathbb{Z}_N$, with k in k_0 and k_1 .

Sender $m_0 = \text{DDLog}_N(g^{Ax}) + F_k(x), \quad m_1 = m_0 + \Delta$

Receiver $\text{DDLog}_N(h_{s_x}^{e_x}) + F_k(x) = m_0 + b\Delta = m_b$

No assumption is needed here, only $e_x \equiv 1 \pmod{N}$ and the shift identity of DDLog.

Outputs b and m_0 look uniform

weak PRFs f_z and F_k

Sender cannot learn b

weak PRF f_z , since k_1 has no z

Receiver cannot learn Δ or m_{1-b}

Sparse-Power DCR, from DCR



Correctness is algebra, and security reduces to the assumptions.

From COT to VOLE

VECTOR OBLIVIOUS LINEAR EVALUATION

COT sample j gives the receiver (b_j, μ_j) and the sender $(m_{0,j}, m_{1,j})$, with $\mu_j = m_{0,j} + b_j \Delta \bmod N$.

$$\boxed{1} \quad \times 1$$

receiver's bits
 $b_j = f_z(x_j)$

From COT to VOLE

VECTOR OBLIVIOUS LINEAR EVALUATION

COT sample j gives the receiver (b_j, μ_j) and the sender $(m_{0,j}, m_{1,j})$, with $\mu_j = m_{0,j} + b_j \Delta \bmod N$.

1 $\times 1$

0 $\times 2$

receiver's bits

$$b_j = f_z(x_j)$$

From COT to VOLE

VECTOR OBLIVIOUS LINEAR EVALUATION

COT sample j gives the receiver (b_j, μ_j) and the sender $(m_{0,j}, m_{1,j})$, with $\mu_j = m_{0,j} + b_j \Delta \bmod N$.

1 $\times 1$

0 $\times 2$

1 $\times 4$

receiver's bits

$$b_j = f_z(x_j)$$

From COT to VOLE

VECTOR OBLIVIOUS LINEAR EVALUATION

COT sample j gives the receiver (b_j, μ_j) and the sender $(m_{0,j}, m_{1,j})$, with $\mu_j = m_{0,j} + b_j \Delta \bmod N$.

1 $\times 1$

0 $\times 2$

1 $\times 4$

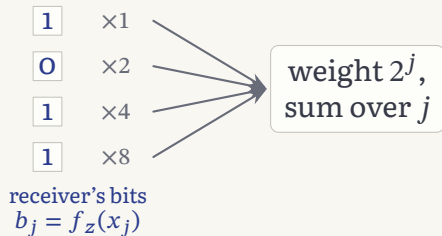
1 $\times 8$

receiver's bits
 $b_j = f_z(x_j)$

From COT to VOLE

VECTOR OBLIVIOUS LINEAR EVALUATION

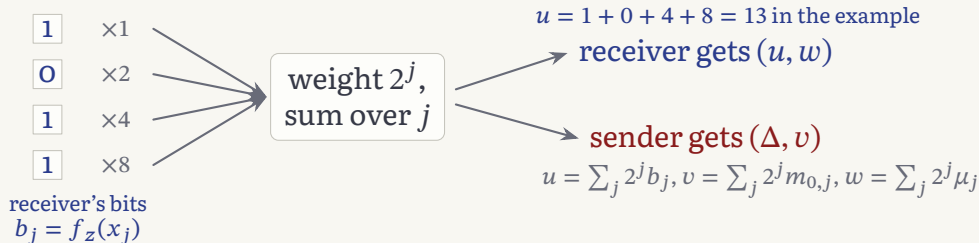
COT sample j gives the receiver (b_j, μ_j) and the sender $(m_{0,j}, m_{1,j})$, with $\mu_j = m_{0,j} + b_j \Delta \bmod N$.



From COT to VOLE

VECTOR OBLIVIOUS LINEAR EVALUATION

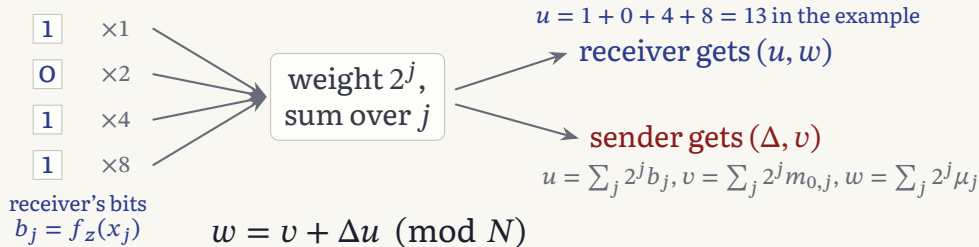
COT sample j gives the receiver (b_j, μ_j) and the sender $(m_{0,j}, m_{1,j})$, with $\mu_j = m_{0,j} + b_j \Delta \bmod N$.



From COT to VOLE

VECTOR OBLIVIOUS LINEAR EVALUATION

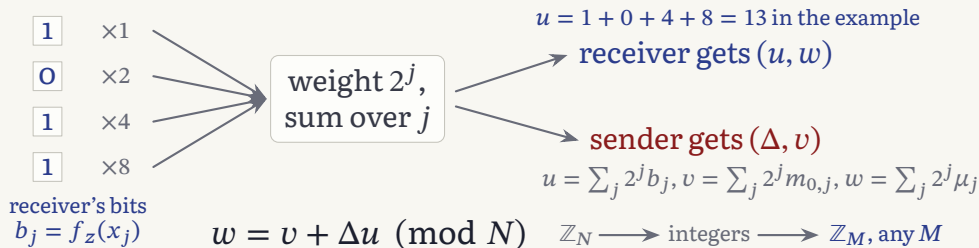
COT sample j gives the receiver (b_j, μ_j) and the sender $(m_{0,j}, m_{1,j})$, with $\mu_j = m_{0,j} + b_j \Delta \bmod N$.



From COT to VOLE

VECTOR OBLIVIOUS LINEAR EVALUATION

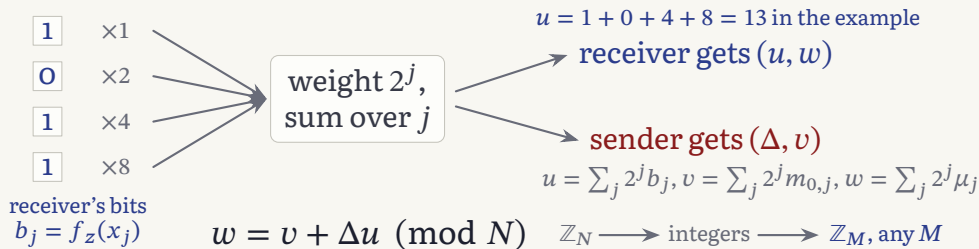
COT sample j gives the receiver (b_j, μ_j) and the sender $(m_{0,j}, m_{1,j})$, with $\mu_j = m_{0,j} + b_j \Delta \bmod N$.



From COT to VOLE

VECTOR OBLIVIOUS LINEAR EVALUATION

COT sample j gives the receiver (b_j, μ_j) and the sender $(m_{0,j}, m_{1,j})$, with $\mu_j = m_{0,j} + b_j \Delta \bmod N$.



Without wraparound, the relation holds over $\mathbb{Z}$ and so modulo any M .

Exponent-verifiable random functions

[BHLS25]

The group $\mathbb{G} = \langle g \rangle$ has prime order q , and $r \in \mathbb{Z}_q$.

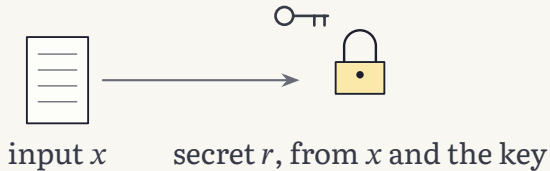


input x

Exponent-verifiable random functions

[BHLS25]

The group $\mathbb{G} = \langle g \rangle$ has prime order q , and $r \in \mathbb{Z}_q$.



Exponent-verifiable random functions

[BHLS25]

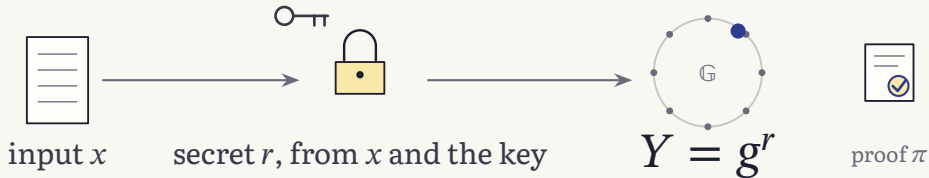
The group $\mathbb{G} = \langle g \rangle$ has prime order q , and $r \in \mathbb{Z}_q$.



Exponent-verifiable random functions

[BHLS25]

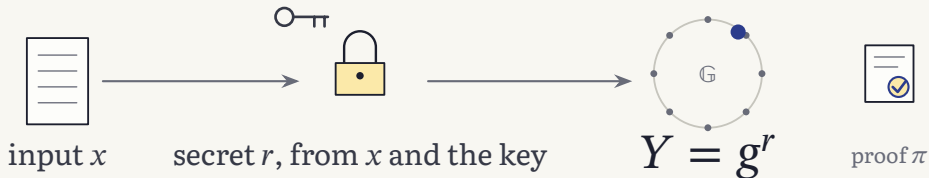
The group $\mathbb{G} = \langle g \rangle$ has prime order q , and $r \in \mathbb{Z}_q$.



Exponent-verifiable random functions

[BHLS25]

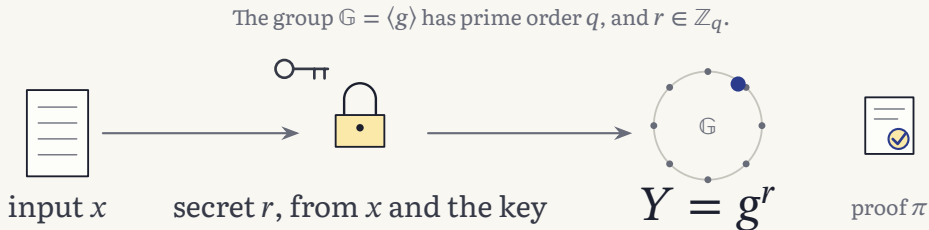
The group $\mathbb{G} = \langle g \rangle$ has prime order q , and $r \in \mathbb{Z}_q$.



A verifier checks Y without learning r .

Exponent-verifiable random functions

[BHLS25]



A verifier checks Y without learning r .



Threshold signing needs verifiable, deterministic per-message nonces.

Designated-verifier eVRF

[KOR23]



prover = PCF receiver

Group $\mathbb{G} = \langle g \rangle$ of prime order q , and a uniform secret $\Delta \in \mathbb{Z}_q$.
The PCF is evaluated at $H_{\text{VOLF}}(x)$ for a random oracle H_{VOLF} .

Designated-verifier eVRF

[KOR23]

Group $\mathbb{G} = \langle g \rangle$ of prime order q , and a uniform secret $\Delta \in \mathbb{Z}_q$.
The PCF is evaluated at $H_{\text{VOLE}}(x)$ for a random oracle H_{VOLE} .



prover = PCF receiver

u

w

Designated-verifier eVRF

[KOR23]

Group $\mathbb{G} = \langle g \rangle$ of prime order q , and a uniform secret $\Delta \in \mathbb{Z}_q$.
The PCF is evaluated at $H_{\text{VOLE}}(x)$ for a random oracle H_{VOLE} .



prover = PCF receiver

u

w

VOLE over $\mathbb{Z}_q$ with $w = v + \Delta u$



verifier = PCF sender

Δ

v

Designated-verifier eVRF

[KOR23]

Group $\mathbb{G} = \langle g \rangle$ of prime order q , and a uniform secret $\Delta \in \mathbb{Z}_q$.
The PCF is evaluated at $H_{\text{VOLE}}(x)$ for a random oracle H_{VOLE} .


prover = PCF receiver

u

w

$$r = u$$

VOLE over $\mathbb{Z}_q$ with $w = v + \Delta u$

Y

$$Y = g^u$$



$$\pi = g^w$$


verifier = PCF sender

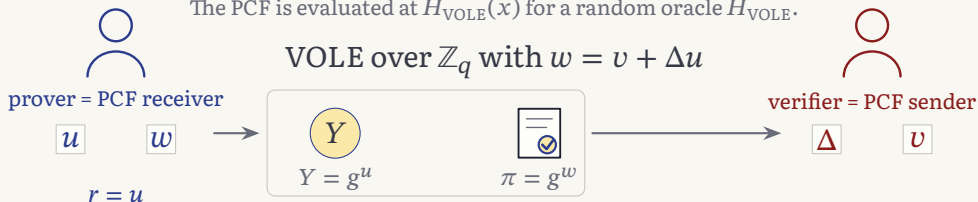
Δ

v

Designated-verifier eVRF

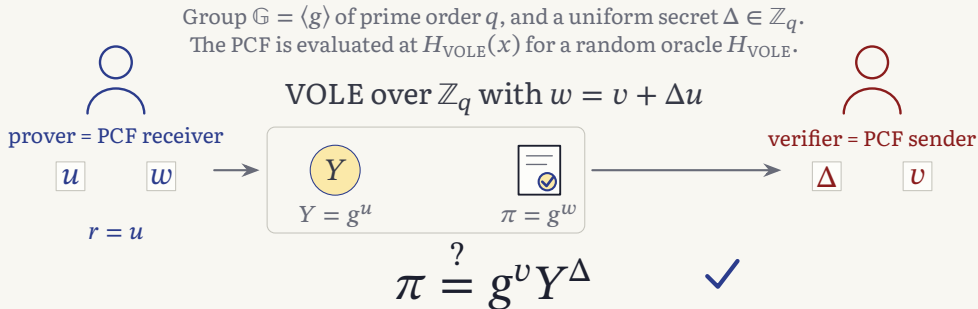
[KOR23]

Group $\mathbb{G} = \langle g \rangle$ of prime order q , and a uniform secret $\Delta \in \mathbb{Z}_q$.
The PCF is evaluated at $H_{\text{VOLE}}(x)$ for a random oracle H_{VOLE} .



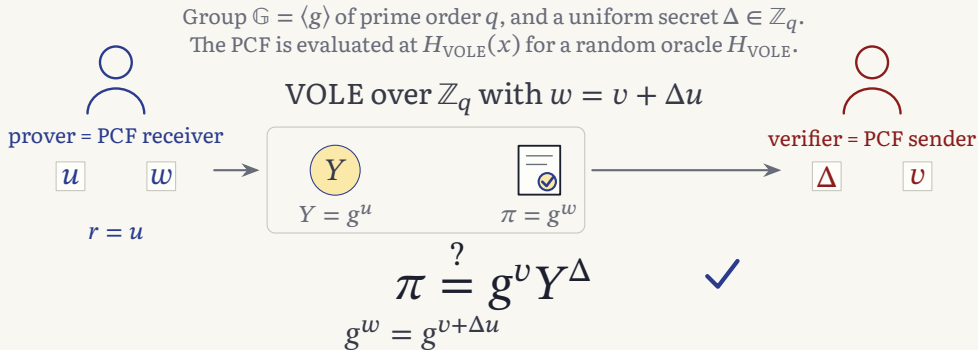
Designated-verifier eVRF

[KOR23]



Designated-verifier eVRF

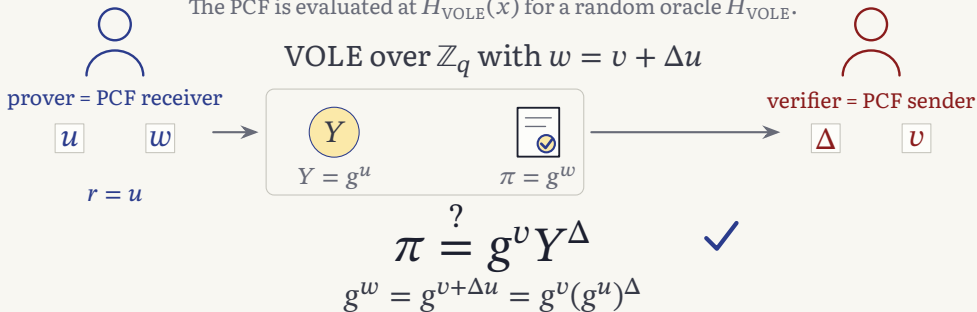
[KOR23]



Designated-verifier eVRF

[KOR23]

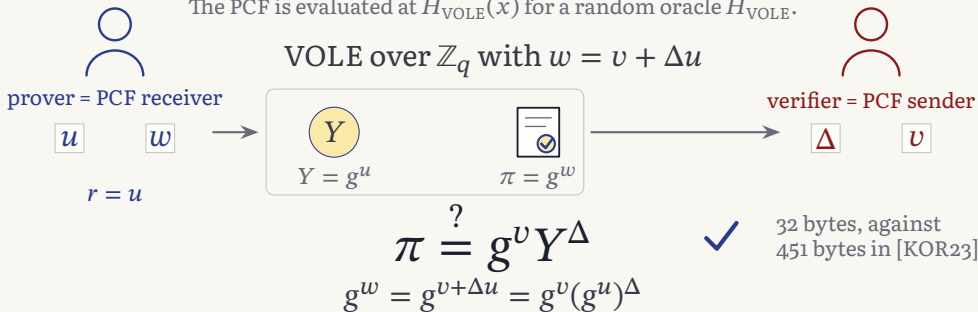
Group $\mathbb{G} = \langle g \rangle$ of prime order q , and a uniform secret $\Delta \in \mathbb{Z}_q$.
The PCF is evaluated at $H_{\text{VOLE}}(x)$ for a random oracle H_{VOLE} .



Designated-verifier eVRF

[KOR23]

Group $\mathbb{G} = \langle g \rangle$ of prime order q , and a uniform secret $\Delta \in \mathbb{Z}_q$.
The PCF is evaluated at $H_{\text{VOLF}}(x)$ for a random oracle H_{VOLF} .



The prover never sees Δ , so it cannot forge a proof for a wrong Y .

Summary



COT over $\mathbb{Z}_N$

Summary

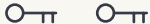


Summary

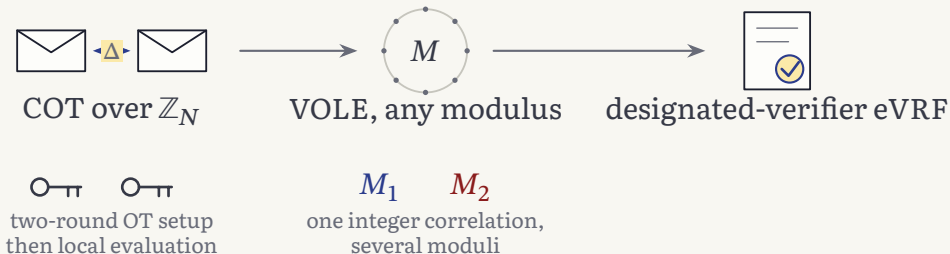


Summary

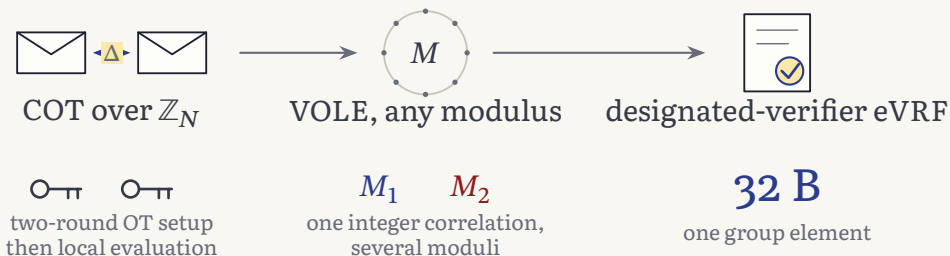



two-round OT setup
then local evaluation

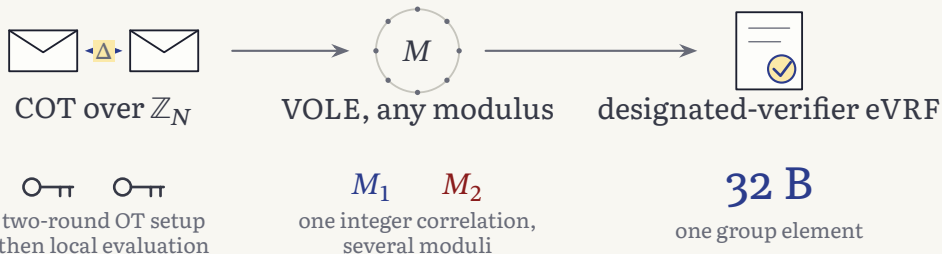
Summary



Summary

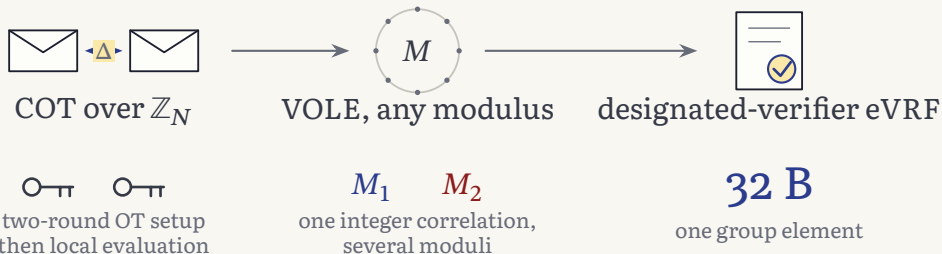


Summary



Assumptions are DCR, an inner-product-membership weak PRF, and a random oracle.

Summary



Assumptions are DCR, an inner-product-membership weak PRF, and a random oracle.



Two short keys give correlated randomness on demand.

Thank you

Questions?

eprint.iacr.org/2026/2030



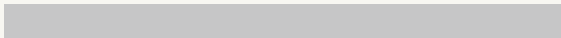
Proof size and cost

Designated-verifier eVRF

32-byte encodings, proof size only



[KOR23]



451 B



this work

32 B

one group element

Amortised cost per COT

sender 74 μ s

receiver (prover) 31 ms

2048-bit N , 32 threads. A 256-COT VOLE block takes 18.9 ms for the sender and 7.9 s for the receiver.

Publicly verifiable eVRF

[BCK25] 928 B $\longrightarrow$ 128 B

Uses offline/online homomorphic secret sharing, not the Paillier PCF. Sizes exclude the output.

Proofs are shorter, but the prover runs the slow PCF receiver.

COT correctness

Let $s_x = \langle z, x \rangle$ and $b_x = \chi_S(s_x)$, where χ_S is the indicator function of the public set S .

receiver's public table

$$h_s = g^{a_0 r^s} (1 + N)^{\chi_S(s)\Delta}$$

$$\alpha_i = a_i r^{-z_i}, \quad e_x = \prod_{i: x_i=1} \alpha_i \equiv 1 \pmod{N}$$

sender's PRF exponent

$$A_x = a_0 \prod_{i: x_i=1} a_i$$

$$m_0 = \text{DDLog}_N(g^{A_x}) + F_k(x)$$

$$h_{s_x}^{e_x} = g^{A_x} (1 + N)^{b_x \Delta}$$

$$m_{b_x} = \text{DDLog}_N(h_{s_x}^{e_x}) + F_k(x) = m_0 + b_x \Delta$$

The base g is an N -th residue, and both keys contain the weak-PRF key k .

Setup exponents are modulo $N\varphi(N)$, with r invertible and

$$\varphi(N) = (p-1)(q-1).$$

PCF comparison

Scheme	Correlation	Assumption	Setup	Keys (k_1/k_0)	S / R (ms)
[BCM+24]	random OT	power-DDH	2 rds., OT	16 B / 30 KiB	0.1 / 0.1
[OSY21]	$\mathbb{F}_2^\lambda$ -COT	DCR, QR	1 rd., PK	≈ 50 KiB each	105 / 105
[BCM+25]	COT, VOLE	sparse LPN	DKG	≈ 3.4 MiB pair	$\ll 1$ / $\ll 1$
This work	$\mathbb{Z}_N$ -COT	DCR	2 rds., OT	65.8 / 129.3 KiB	1.53 / 523

Here λ is the security parameter and QR is quadratic residuosity. PK is a non-interactive public-key setup, and DKG is distributed key generation with non-black-box use of a PRG. Times are single-core per correlation.

Converting random OT into COT needs one correction value per output, so evaluation is no longer silent.

Setup and evaluation costs

Two-round pairwise OT setup

Security is static and semi-honest, and the sender knows the factorisation of N .

	Sender	Receiver
Key size	65.8 KiB	129.3 KiB
Single COT evaluation	1.53 ms	522.80 ms
Batched COT, amortised	73.87 μ s	31.02 ms

Measurements from the paper, with 2048-bit N , input length $n = 128$, batch 256, a Xeon Silver 4314, and 32 batch threads.

The sender reduces exponents using $\varphi(N)$, which the receiver does not know.
Each VOLE uses several COT evaluations.

Integer lifting and uniformity

For ℓ ideal independent COT samples, $\mu_j = m_{0,j} + b_j \Delta \bmod N$, and each $m_{0,j}$ is uniform conditioned on b_j and Δ . If $0 \leq \Delta < B < N$, the probability of any wraparound is at most $\ell B/N$. Without wraparound, $w = v + \Delta u$ holds over the integers and modulo every M .

$$u = \sum_{j=0}^{\ell-1} 2^j b_j, \quad \text{SD}(u \bmod M, U_{\mathbb{Z}_M}) \leq \frac{M}{2^\ell}.$$

Here SD is statistical distance and $U_{\mathbb{Z}_M}$ is uniform on $\mathbb{Z}_M$. For a statistical parameter κ and $\ell \geq \lceil \log_2 M \rceil + \kappa$, this term is at most $2^{-\kappa}$.

Uniformity of the full VOLE pair also needs $v \bmod M$ close to uniform, which the paper's parameter conditions ensure.

Weak PCF security

The inner predicate is $f_z(x) = \chi_S(\langle z, x \rangle)$ for $z, x \in \{0, 1\}^n$, where χ_S is the indicator function of a public set $S \subseteq \{0, \dots, n\}$. We assume that it is a weak PRF on uniformly random inputs.

The construction assumes this weak PRF, a weak PRF F_k into $\mathbb{Z}_N$, and Sparse-Power DCR. We derive Sparse-Power DCR from DCR for the prescribed exponent distribution.

Arbitrary inputs are hashed to random inputs by a domain-separated random oracle. With Q hashed inputs and ℓ COT inputs per VOLE, the collision probability is at most

$$\binom{Q\ell}{2} 2^{-n}.$$

Publicly verifiable eVRF

For a PRF F_K with secret key K , homomorphic secret sharing (HSS) gives shares W_0, W_1 . The challenge δ lies in $\mathbb{Z}_q^*$. This construction is independent of the Paillier PCF.

$$W_1 - W_0 = \delta F_K(x), \quad Y = g^{F_K(x)}, \quad T = g^{-W_0}.$$

The prover commits to its offline share and T before deriving the challenge δ by Fiat–Shamir. Homomorphic commitments enforce consistency of the HSS shares.

$$Y^\delta \stackrel{?}{=} g^{W_1} T.$$

It uses offline/online HSS, a PRF, homomorphic commitments, and a random oracle. In the paper's instantiation, the proof has about 128 bytes, excluding the 32-byte output.

References

- BCG+20a Correlated pseudorandom functions from variable-density LPN. FOCS 2020.
- BIP+18 Exploring crypto dark matter: New simple PRF candidates and their applications. TCC 2018.
- BCM+24 Fast public-key silent OT and more from constrained Naor–Reingold. EUROCRYPT 2024.
- BCM+25 Fast pseudorandom correlation functions from sparse LPN. ASIACRYPT 2025.
- OSY21 The rise of Paillier: Homomorphic secret sharing and public-key silent OT. EUROCRYPT 2021.
- RS21 Large message homomorphic secret sharing from DCR and applications. CRYPTO 2021.
- KOR23 Two-round stateless deterministic two-party Schnorr signatures from pseudorandom correlation functions. CRYPTO 2023.
- BHLS25 Exponent-VRFs and their applications. EUROCRYPT 2025.
- BCK25 Golden: Lightweight non-interactive distributed key generation. ePrint 2025/1924.