Delegating Bilinear Pairings: Systematization, Amortized Efficiency, and Future Directions

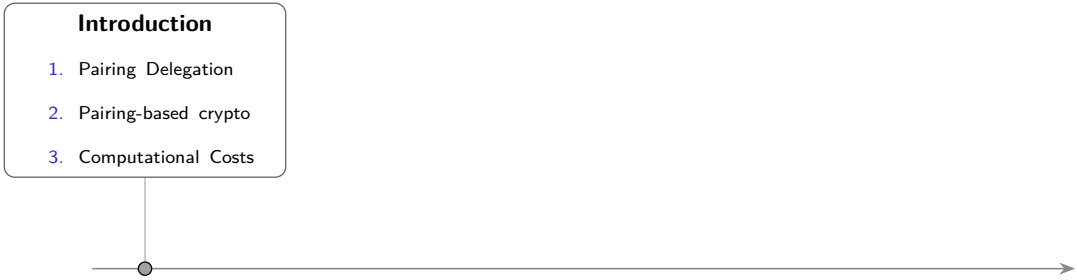
Adrián P. Keilty

Thesis for the Degree of Licentiate of Philosophy

CHALMERS

Outline

Outline


Introduction
1. Pairing Delegation
2. Pairing-based crypto
3. Computational Costs

Outline

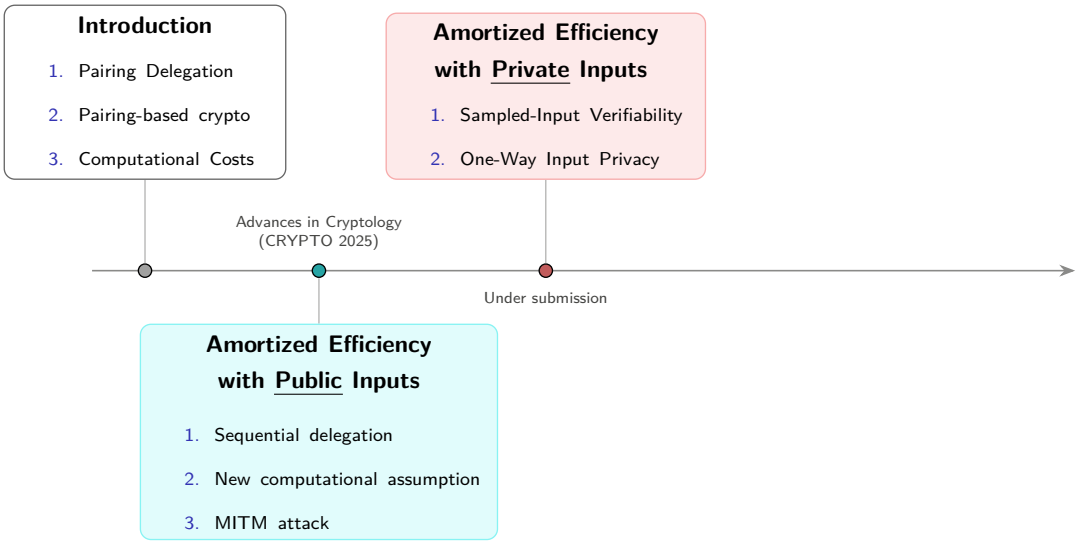
Introduction

1. Pairing Delegation
2. Pairing-based crypto
3. Computational Costs

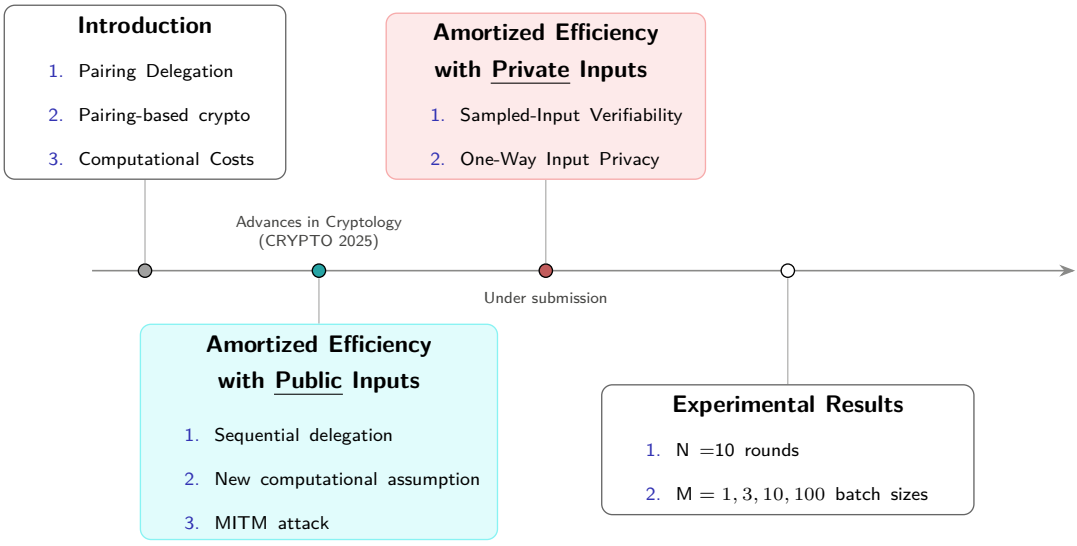
Amortized Efficiency with Public Inputs

1. Sequential delegation
2. New computational assumption
3. MITM attack

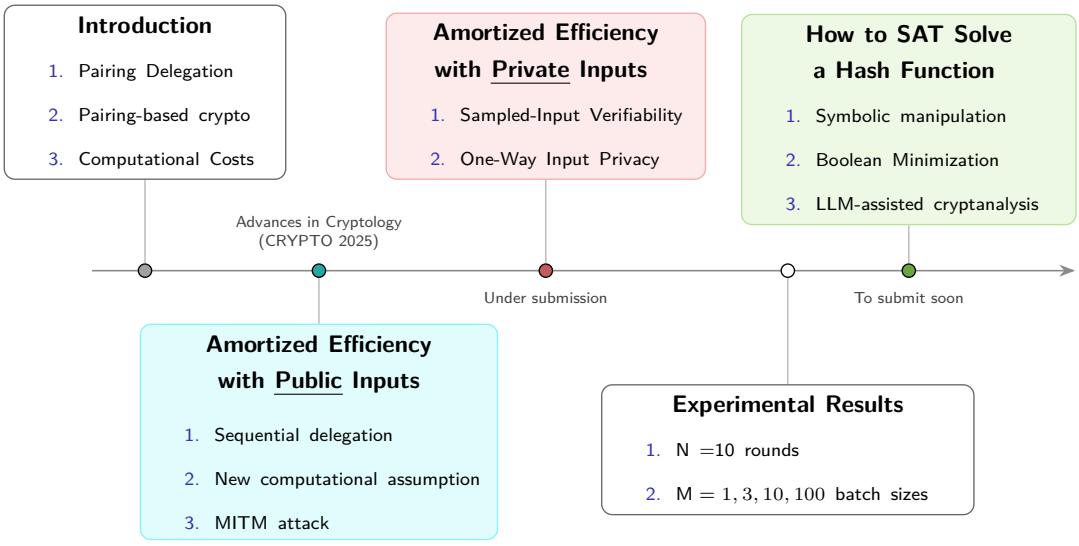
Outline


Introduction
1. Pairing Delegation
2. Pairing-based crypto
3. Computational Costs
Amortized Efficiency with Private Inputs
1. Sampled-Input Verifiability
2. One-Way Input Privacy
Advances in Cryptology (CRYPTO 2025)
Under submission
Amortized Efficiency with Public Inputs
1. Sequential delegation
2. New computational assumption
3. MITM attack

Outline


Introduction
1. Pairing Delegation
2. Pairing-based crypto
3. Computational Costs
Amortized Efficiency with Private Inputs
1. Sampled-Input Verifiability
2. One-Way Input Privacy
Advances in Cryptology (CRYPTO 2025)
Under submission
Amortized Efficiency with Public Inputs
1. Sequential delegation
2. New computational assumption
3. MITM attack
Experimental Results
1. N =10 rounds
2. M = 1, 3, 10, 100 batch sizes

Outline


Introduction
1. Pairing Delegation
2. Pairing-based crypto
3. Computational Costs
Amortized Efficiency with Private Inputs
1. Sampled-Input Verifiability
2. One-Way Input Privacy
How to SAT Solve a Hash Function
1. Symbolic manipulation
2. Boolean Minimization
3. LLM-assisted cryptanalysis
Advances in Cryptology (CRYPTO 2025)
Under submission
To submit soon
Amortized Efficiency with Public Inputs
1. Sequential delegation
2. New computational assumption
3. MITM attack
Experimental Results
1. N =10 rounds
2. M = 1, 3, 10, 100 batch sizes

What is pairing delegation?

What is pairing delegation?

Cryptography

What is pairing delegation?

Cryptography

Elliptic Curve Cryptography

What is pairing delegation?

Cryptography

Elliptic Curve Cryptography

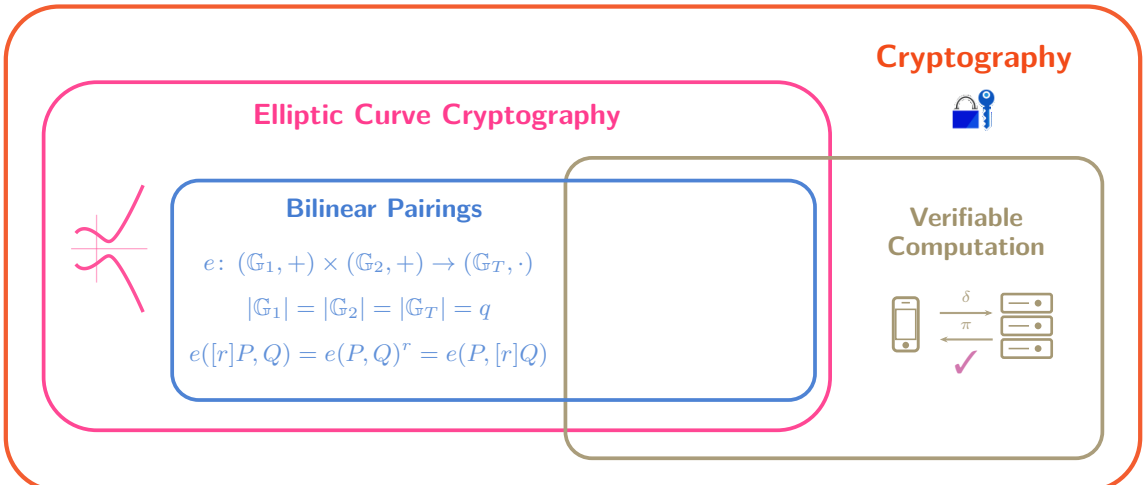
Bilinear Pairings

$$e\colon (\mathbb{G}_1, +) \times (\mathbb{G}_2, +) \to (\mathbb{G}_T, \cdot)$$

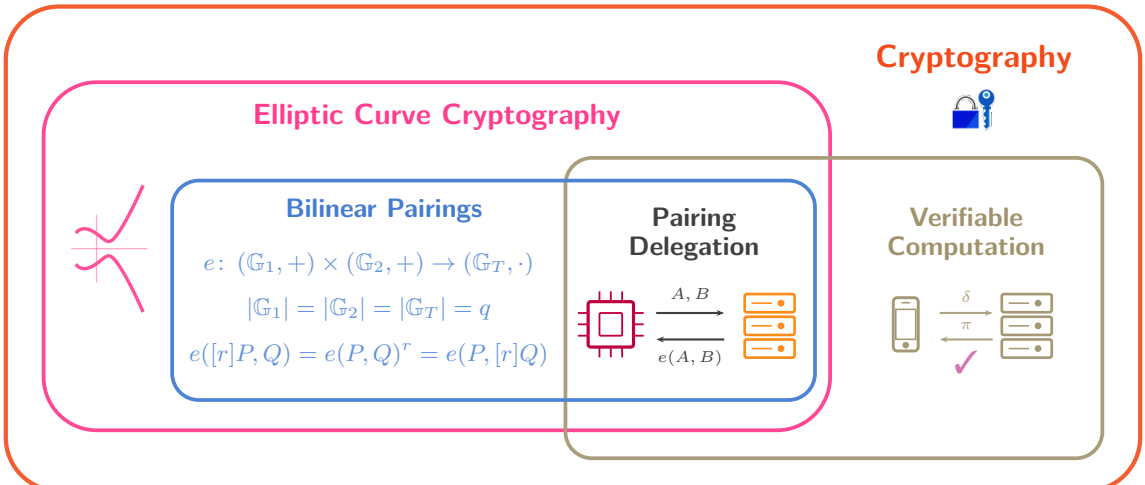
$$|\mathbb{G}_1| = |\mathbb{G}_2| = |\mathbb{G}_T| = q$$

$$e([r]P, Q) = e(P, Q)^r = e(P, [r]Q)$$

What is pairing delegation?



What is pairing delegation?



Why use pairing delegation?

Why use pairing delegation?

1. Pairing-based Cryptography

Why use pairing delegation?

1. Pairing-based Cryptography

- Three-party DH key agreement (Joux)
- Short Signatures (Boneh-Lynn-Shacham)
- Identity-Based Encryption (Boneh-Franklin)
- Constant-size commitments (Kate-Zaverucha-Goldberg)
- Constant-size zkSNARK proofs (Groth)

Why use pairing delegation?

1. Pairing-based Cryptography

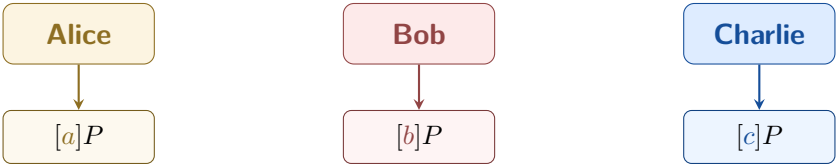
- Three-party DH key agreement (Joux)
- Short Signatures (Boneh-Lynn-Shacham)
- Identity-Based Encryption (Boneh-Franklin)
- Constant-size commitments (Kate-Zaverucha-Goldberg)
- Constant-size zkSNARK proofs (Groth)

- EUROCRYPT 2026:
  - "Silent Threshold Cryptography from **Pairings**: Expressive Policies in the Plain Model", (Waters-Wu).
  - "Threshold Batched Identity-Based Encryption from **Pairings** in the Plain Model", (Gong et al).

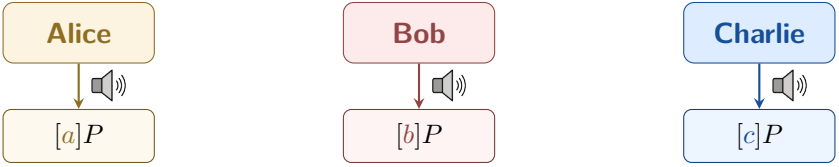
An example: Joux's tripartite Diffie-Hellman

Alice **Bob** **Charlie**

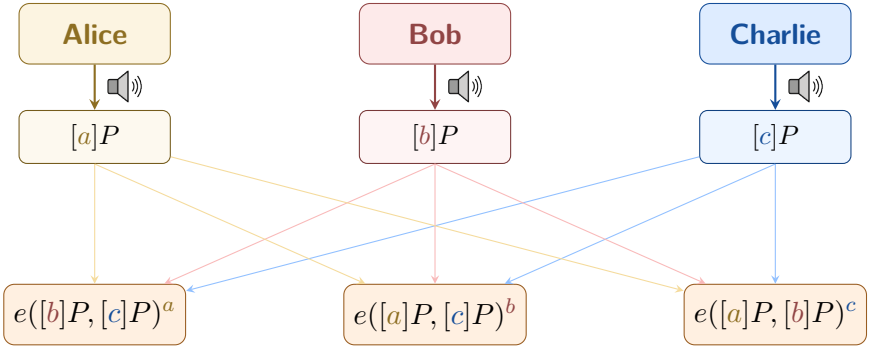
An example: Joux's tripartite Diffie-Hellman



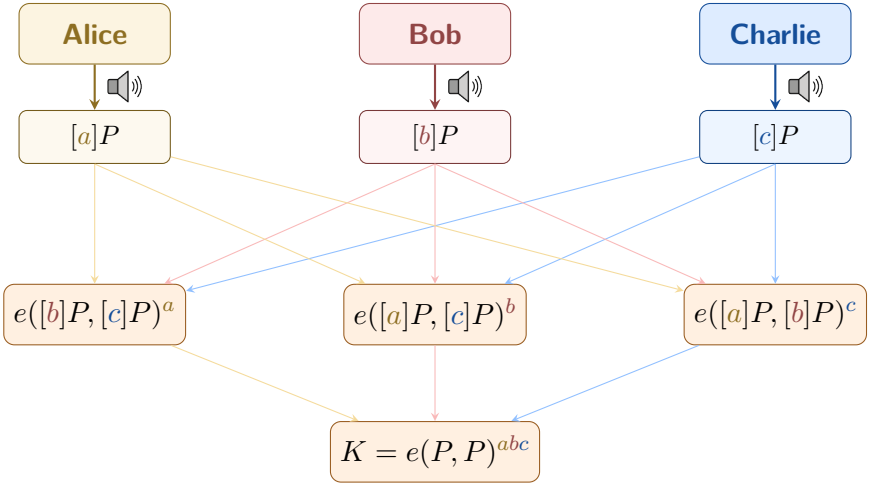
An example: Joux's tripartite Diffie-Hellman



An example: Joux's tripartite Diffie-Hellman



An example: Joux's tripartite Diffie-Hellman

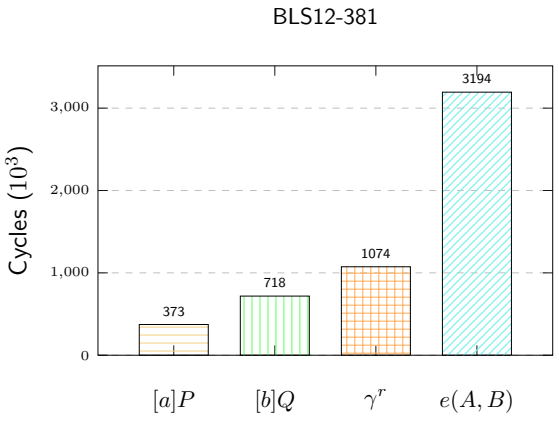


Why use pairing delegation?

2. Pairings are expensive

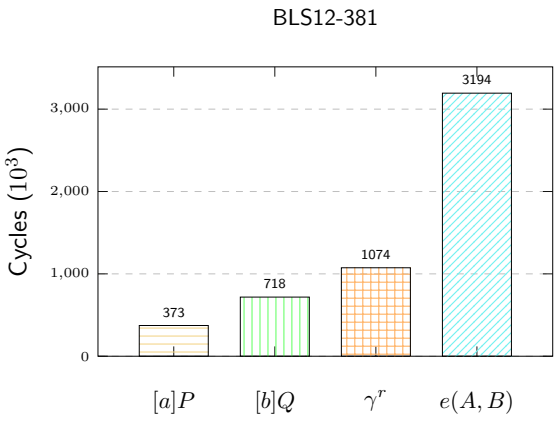
Why use pairing delegation?

2. Pairings are expensive



Why use pairing delegation?

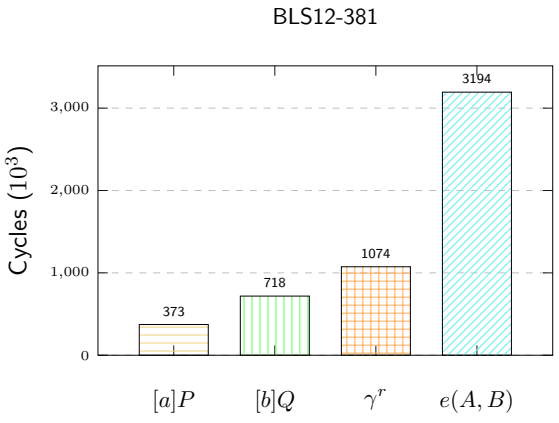
2. Pairings are expensive



- A single pairing is $\times 6$ to $\times 9$ more expensive than group scalar multiplications.
- Requires a specialized library.

Why use pairing delegation?

2. Pairings are expensive



- A single pairing is $\times 6$ to $\times 9$ more expensive than group scalar multiplications.
- Requires a specialized library.

Prohibitive for hardware wallets and smartcards

Delegation Properties

Delegation Properties

Correctness

If the server behaves honestly, the result obtained is the intended pairing(s).

Delegation Properties

Correctness

If the server behaves honestly, the result obtained is the intended pairing(s).

Verifiability

The client detects with high probability if the server returns an incorrect result.

Delegation Properties

Correctness

If the server behaves honestly, the result obtained is the intended pairing(s).

Verifiability

The client detects with high probability if the server returns an incorrect result.

Efficiency

Delegating L pairings is cheaper than computing them locally.

Delegation Properties

Correctness

If the server behaves honestly, the result obtained is the intended pairing(s).

Verifiability

The client detects with high probability if the server returns an incorrect result.

Efficiency

Delegating L pairings is cheaper than computing them locally.

Privacy (optional)

One or both of the client's inputs must be kept secret from the server.

That's AmorE: Amortized Efficiency for Pairing Delegation

Adrián P. Keilty[1], Diego F. Aranha[2], Elena Pagnin[1], and Francisco Rodríguez-Henríquez[3]

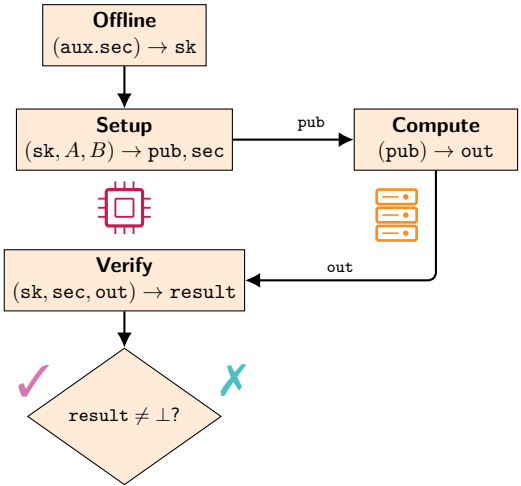
[1] Chalmers University of Technology, Sweden
[2] Aarhus University, Denmark
[3] Technology Innovation Institute, UAE

(Appeared in: Advances in Cryptology - CRYPTO 2025)

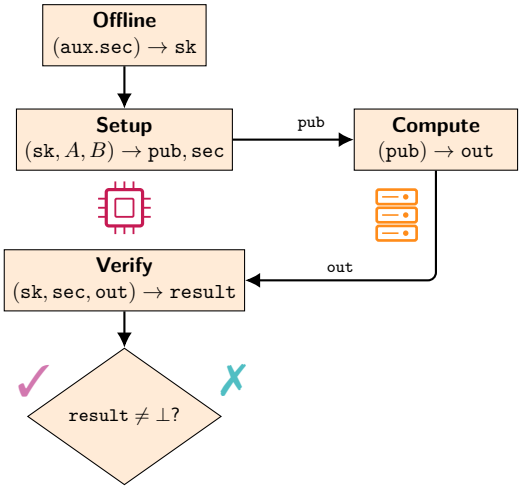
Current approach:

One-shot delegation

One-Shot Pairing Delegation

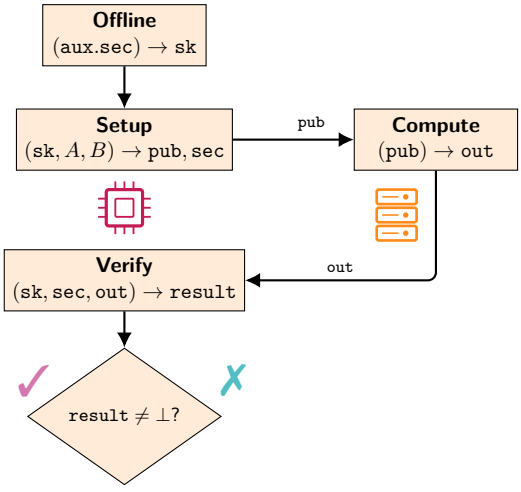


One-Shot Pairing Delegation



Re-start required

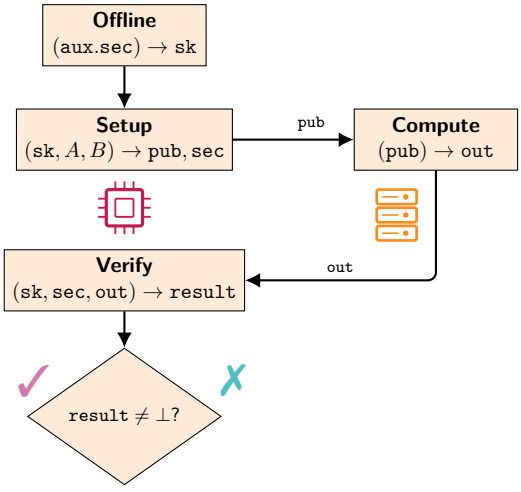
One-Shot Pairing Delegation



Re-start required

Info theoretic security

One-Shot Pairing Delegation



Re-start required

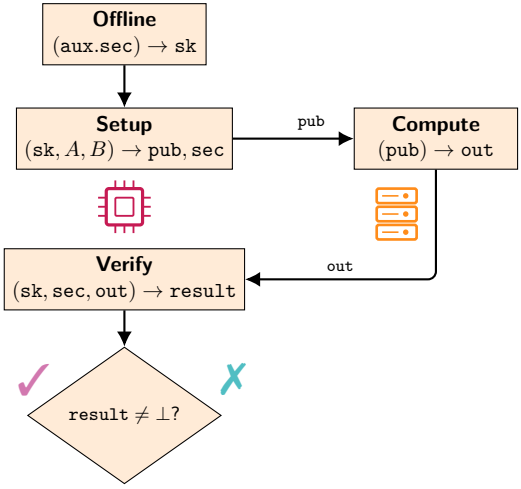
Info theoretic security

High client workload

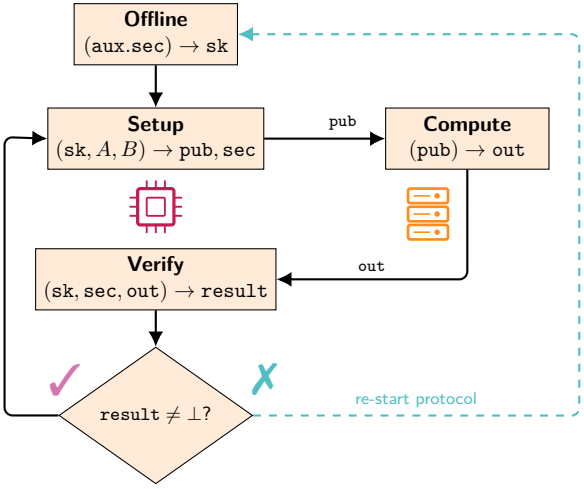
A new approach:

Sequential delegation

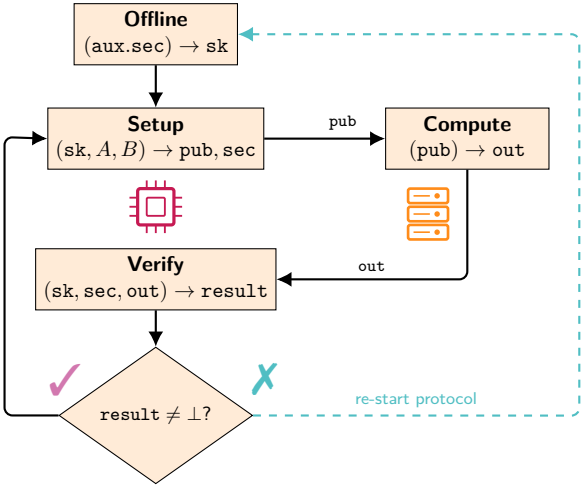
Sequential Pairing Delegation



Sequential Pairing Delegation



Sequential Pairing Delegation

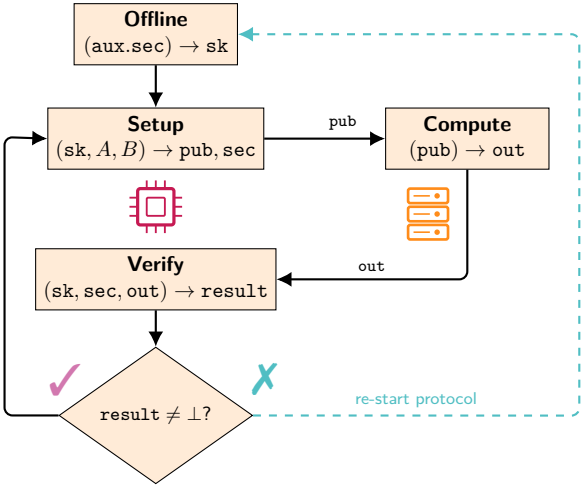


One-time setup

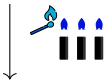
↓

N delegations

Sequential Pairing Delegation

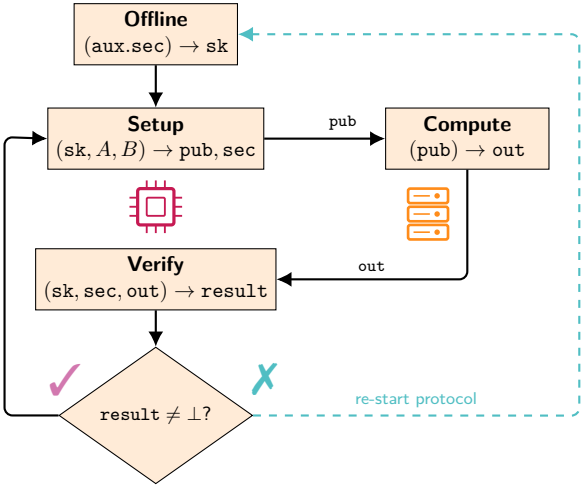


One-time setup

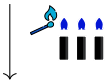


N delegations

Sequential Pairing Delegation



One-time setup



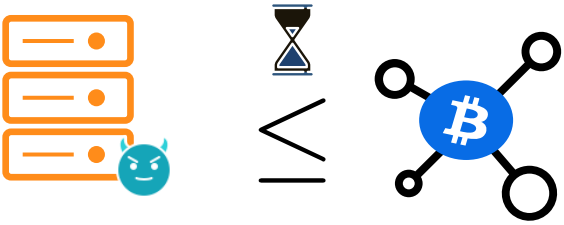
N delegations

Do we achieve efficiency over several rounds?

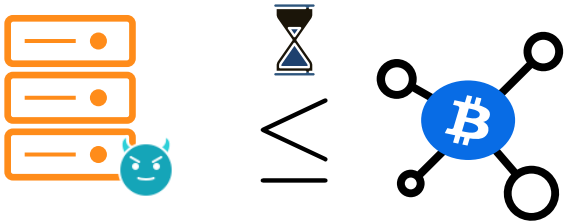
New Computational Assumption

New Computational Assumption

New Computational Assumption

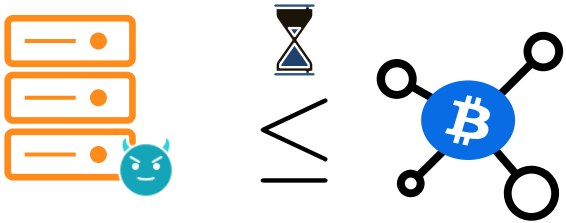


New Computational Assumption



1. Updatable bound: all-time highest Hashrate (hashes per second)
2. cost (SHA256 (header-block)) $\leq$ cost ($[a]P$: $P \in \mathbb{G}_1$)

New Computational Assumption

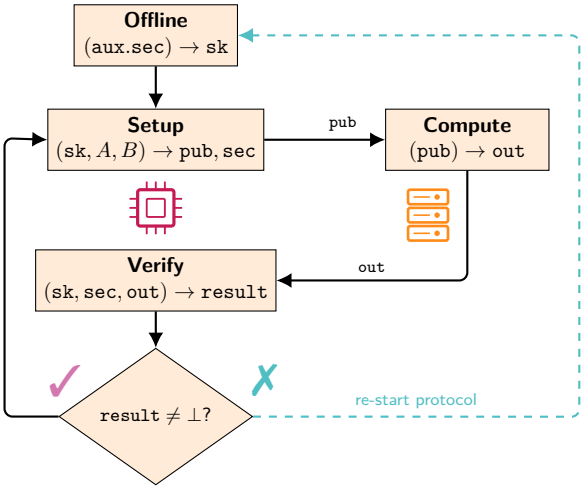


1. Updatable bound: all-time highest Hashrate (hashes per second)
2. cost (SHA256 (header-block)) $\leq$ cost ($[a]P$: $P \in \mathbb{G}_1$)

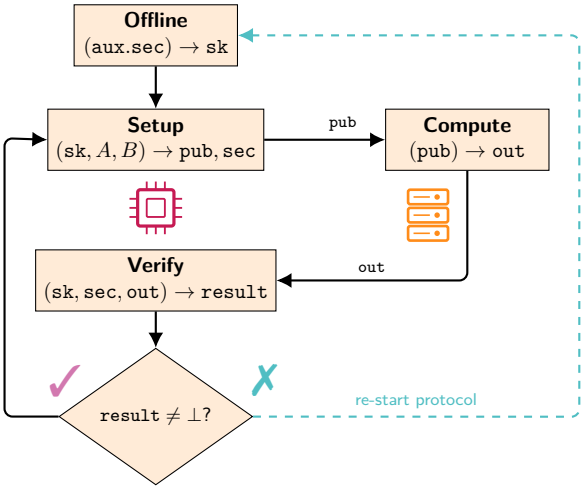
Our assumption: $\mathcal{A}$ computes up to 2^κ short scalar multiplications per second where $\kappa = \log_2(\text{Bitcoin hashrate})$.

Towards Amortized Efficiency

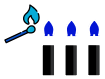
Towards Amortized Efficiency



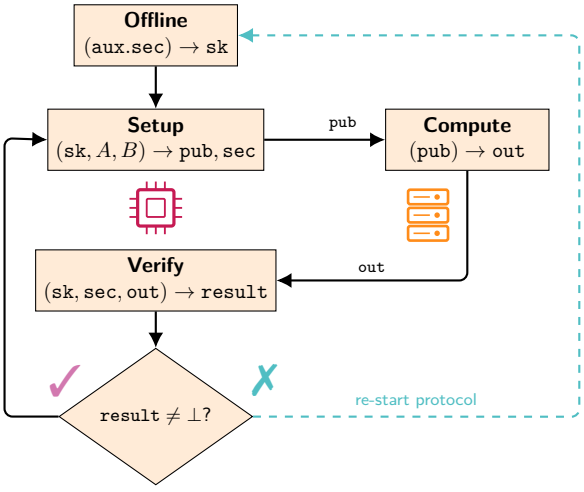
Towards Amortized Efficiency



Sequential delegation

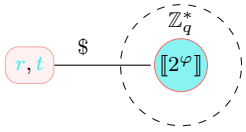


Towards Amortized Efficiency

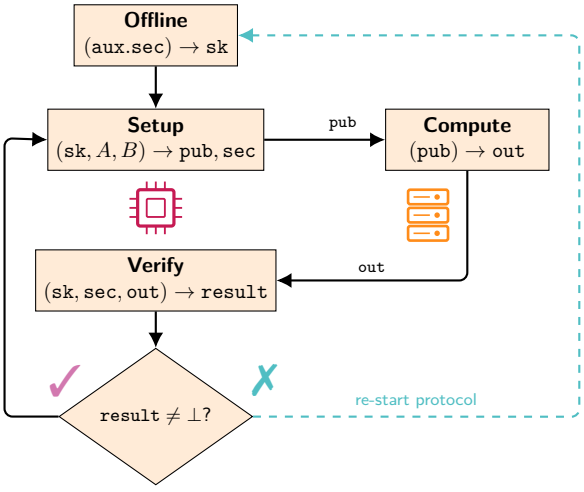


Sequential delegation

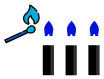
Short scalar sampling



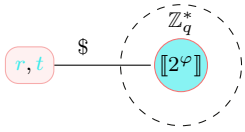
Towards Amortized Efficiency



Sequential delegation



Short scalar sampling



Low(er) client workload

From Honest Ouput to Forgery

server honest output: $\texttt{out} = (\rho, \gamma)$

client verification: $\left(\xi \stackrel{?}{=} \rho^{r} \cdot \gamma\right) \wedge \left(\rho \stackrel{?}{\in} \mathbb{G}_T\right)$

From Honest Ouput to Forgery

$\mathcal{A}$ forgery attempt

$\rho^* \neq \rho = e(A, B)$

server honest output: $\texttt{out} = (\rho, \gamma)$

client verification: $\left(\xi \stackrel{?}{=} \rho^r \cdot \gamma\right) \wedge \left(\rho \stackrel{?}{\in} \mathbb{G}_T\right)$

From Honest Ouput to Forgery

$\mathcal{A}$ forgery attempt

$\rho^* \neq \rho = e(A, B)$

ρ^* can be expressed as a deviation from ρ:

$\exists\, \eta \neq 1 \colon \rho^* = \rho \cdot \eta$

server honest output: $\texttt{out} = (\rho, \gamma)$

client verification: $\left(\xi \stackrel{?}{=} \rho^{r} \cdot \gamma\right) \wedge \left(\rho \stackrel{?}{\in} \mathbb{G}_T\right)$

From Honest Ouput to Forgery

$\mathcal{A}$ forgery attempt

$\rho^* \neq \rho = e(A, B)$

ρ^* can be expressed as a deviation from ρ:

$\exists\, \eta \neq 1 \colon \rho^* = \rho \cdot \eta$

server honest output: $\texttt{out} = (\rho, \gamma)$

client verification: $\left(\xi \stackrel{?}{=} \rho^r \cdot \gamma\right) \wedge \left(\rho \stackrel{?}{\in} \mathbb{G}_T\right)$

To balance $\xi = (\rho \cdot \eta)^r \cdot \gamma^*$,

$\mathcal{A}$ **must set** $\gamma^* = \gamma \cdot \eta^{-r}$

From Honest Ouput to Forgery

$\mathcal{A}$ forgery attempt

$\rho^* \neq \rho = e(A, B)$

server honest output: $\texttt{out} = (\rho, \gamma)$

client verification: $\left(\xi \stackrel{?}{=} \rho^r \cdot \gamma\right) \wedge \left(\rho \stackrel{?}{\in} \mathbb{G}_T\right)$

ρ^* can be expressed as a deviation from ρ:

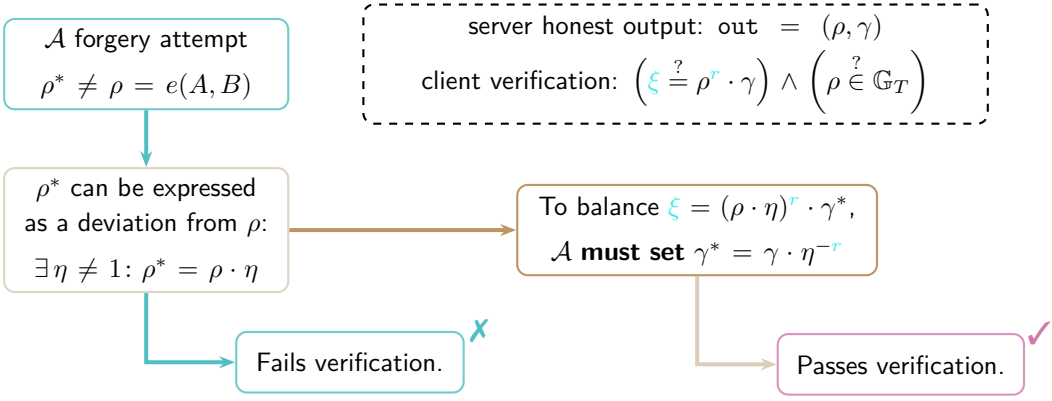
$\exists\, \eta \neq 1 \colon \rho^* = \rho \cdot \eta$

To balance $\xi = (\rho \cdot \eta)^r \cdot \gamma^*$,

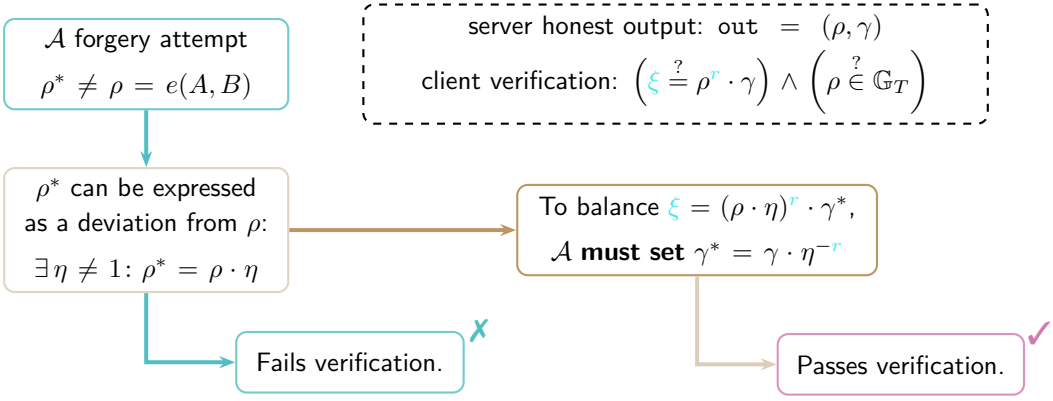
$\mathcal{A}$ **must set** $\gamma^* = \gamma \cdot \eta^{-r}$

Passes verification. ✓

From Honest Ouput to Forgery



From Honest Ouput to Forgery



Lemma 1 (Identifying forgeries)

To forge a pairing, an algebraic adversary must extract r from the public tuple pub.

Setup(sk, $\vec{A}, \vec{B}$) → (pub, sec)

$\mathsf{M} \leftarrow \mathtt{len}(\vec{A})$
if $\mathsf{M} = 1$:
  $r \xleftarrow{\$} [\![\min\{2^{\psi}, q-1\}]\!]$
  $C \leftarrow \left[-s \cdot u^{-1}\right](U + A)$
  $D \leftarrow V - [r]B$
  $(X, Y) \leftarrow (\perp, \perp)$
else :
  $W \xleftarrow{\$} \mathbb{G}_1 \setminus \{\mathcal{O}\}$
  $D \leftarrow \sum_{j=1}^{\mathsf{M}} B_j$
  $(X, Y) \leftarrow ([u](D - V), W - U)$
  for $j \in [\![\mathsf{M}]\!]$:
    $r_j \xleftarrow{\$} [\![\min\{2^{\psi}, q-1\}]\!]$
    $C_j \leftarrow \left[r_j\right] A_j + W$
if $\mathcal{O} \in \{\vec{C}, D, X, Y\}$: **return** $\perp$
$\mathsf{pub} \leftarrow \left(\vec{A}, \vec{B}, \vec{C}, D, X, Y\right)$
$\mathsf{sec} \leftarrow \vec{r}$
return (pub, sec)

OneTimeSetup(aux.sec) → sk

parse aux.sec = (τ, φ)
$s \xleftarrow{\$} \mathbb{Z}_q^*$; $\xi \leftarrow \gamma_T^s$
$\mathsf{t}_{\mathsf{start}} \leftarrow \mathtt{time.now}()$
return $(s, \xi, \tau, \varphi, \mathsf{t}_{\mathsf{start}})$

Compute(pub) → out

parse $\mathsf{pub} = \left(\vec{A}, \vec{B}, \vec{C}, D, X, Y\right)$
$\mathsf{M} \leftarrow \mathtt{len}(\vec{A})$
for $j \in [\![\mathsf{M}]\!]$: $\rho_j = e(A_j, B_j)$
if $\mathsf{M} = 1$: $\gamma \leftarrow e(A, D) \cdot e(C, Q)$
else :
  $\gamma \leftarrow \left(\prod_{j=1}^{\mathsf{M}} e(C_j, -B_j)\right) \cdot e(Y, D) \cdot e(P, X)$
return $(\gamma, \vec{\rho})$

Verify(sk, sec, out) → result

parse $\mathsf{sk} = (\cdot, \xi, \tau, \cdot, \mathsf{t}_{\mathsf{start}})$
if $\mathtt{time.now}() - \mathsf{t}_{\mathsf{start}} > \tau$: **return** $\perp$
parse $\mathsf{sec} = \vec{r}$; **parse** $\mathtt{out} = (\gamma, \vec{\rho})$
$\mathsf{M} \leftarrow \mathtt{len}(\vec{r})$
if $\bigvee_{j \in [\![\mathsf{M}]\!]} \left(\rho_j \notin \mathbb{G}_T\right)$: **return** $\perp$
if $\xi = \left(\prod_{j=1}^{\mathsf{M}} \rho_j^{r_j}\right) \cdot \gamma$: **return** $\vec{\rho}$
return $\perp$

Setup$(\mathsf{sk}, \vec{A}, \vec{B}) \to (\mathsf{pub}, \mathsf{sec})$

1: **parse** $\mathsf{sk} = (s, \cdot, \cdot, \varphi, \cdot)$
2: **if** $\mathrm{len}(\vec{A}) \neq \mathrm{len}(\vec{B})$:
3: **return** $\perp$
4: **if** $\mathcal{O} \in \vec{A} \cup \vec{B}$: **return** $\perp$
5: $u \xleftarrow{\$} \mathbb{Z}_q^*$
6: $(U, V) \leftarrow \left([u]P,\ [s \cdot u^{-1}]Q\right)$
7: $\mathsf{M} \leftarrow \mathrm{len}(\vec{A})$
8: **if** $\mathsf{M} = 1$:
9: $r \xleftarrow{\$} [\![\min\{2^\varphi, q-1\}]\!]$
10: $C \leftarrow \left[-s \cdot u^{-1}\right](U + A)$
11: $D \leftarrow V - [r]B$
12: $(X, Y) \leftarrow (\perp, \perp)$
13: **else :**
14: $W \xleftarrow{\$} \mathbb{G}_1 \setminus \{\mathcal{O}\}$
15: $D \leftarrow \sum_{j=1}^{\mathsf{M}} B_j$
16: $(X, Y) \leftarrow ([u](D - V), W - U)$
17: **for** $j \in [\![\mathsf{M}]\!]$:
18: $r_j \xleftarrow{\$} [\![\min\{2^\varphi, q-1\}]\!]$
19: $C_j \leftarrow \left[r_j\right]A_j + W$
20: **if** $\mathcal{O} \in \{\vec{C}, D, X, Y\}$: **return** $\perp$
21: $\mathsf{pub} \leftarrow \left(\vec{A}, \vec{B}, \vec{C}, D, X, Y\right)$
22: $\mathsf{sec} \leftarrow \vec{r}$
23: **return** $(\mathsf{pub}, \mathsf{sec})$

OneTimeSetup$(\mathsf{aux.sec}) \to \mathsf{sk}$

1: **parse** $\mathsf{aux.sec} = (\tau, \varphi)$
2: $s \xleftarrow{\$} \mathbb{Z}_q^*$; $\xi \leftarrow \gamma_T^s$
3: $\mathsf{t_{start}} \leftarrow \mathtt{time.now}()$
4: **return** $(s, \xi, \tau, \varphi, \mathsf{t_{start}})$

Compute$(\mathsf{pub}) \to \mathsf{out}$

1: **parse** $\mathsf{pub} = \left(\vec{A}, \vec{B}, \vec{C}, D, X, Y\right)$
2: $\mathsf{M} \leftarrow \mathrm{len}(\vec{A})$
3: **for** $j \in [\![\mathsf{M}]\!]$: $\rho_j = e(A_j, B_j)$
4: **if** $\mathsf{M} = 1$: $\gamma \leftarrow e(A, D) \cdot e(C, Q)$
5: **else :**
6: $\gamma \leftarrow \left(\prod_{j=1}^{\mathsf{M}} e(C_j, -B_j)\right) \cdot e(Y, D) \cdot e(P, X)$
7: **return** $(\gamma, \vec{\rho})$

Verify$(\mathsf{sk}, \mathsf{sec}, \mathsf{out}) \to \mathtt{result}$

1: **parse** $\mathsf{sk} = (\cdot, \xi, \tau, \cdot, \mathsf{t_{start}})$
2: **if** $\mathtt{time.now}() - \mathsf{t_{start}} > \tau$: **return** $\perp$
3: **parse** $\mathsf{sec} = \vec{r}$; **parse** $\mathsf{out} = (\gamma, \vec{\rho})$
4: $\mathsf{M} \leftarrow \mathrm{len}(\vec{r})$
5: **if** $\bigvee_{j \in [\![\mathsf{M}]\!]} \left(\rho_j \notin \mathbb{G}_T\right)$: **return** $\perp$
6: **if** $\xi = \left(\prod_{j=1}^{\mathsf{M}} \rho_j^{r_j}\right) \cdot \gamma$: **return** $\vec{\rho}$
7: **return** $\perp$

AmorE

Setup$(\mathsf{sk}, \vec{A}, \vec{B}) \to (\mathsf{pub}, \mathsf{sec})$

1: **parse** $\mathsf{sk} = (s, \cdot, \cdot, \varphi, \cdot)$
2: **if** $\mathtt{len}(\vec{A}) \neq \mathtt{len}(\vec{B})$:
3: **return** $\perp$
4: **if** $\mathcal{O} \in \vec{A} \cup \vec{B}$: **return** $\perp$
5: $u \xleftarrow{\$} \mathbb{Z}_q^*$
6: $(U, V) \leftarrow \left([u]P,\ [s \cdot u^{-1}]Q\right)$
7: $\mathsf{M} \leftarrow \mathtt{len}(\vec{A})$
8: **if** $\mathsf{M} = 1$:
9: $r \xleftarrow{\$} [\![\min\{2^{\varphi}, q-1\}]\!]$
10: $C \leftarrow \left[-s \cdot u^{-1}\right](U + A)$
11: $D \leftarrow V - [r]B$
12: $(X, Y) \leftarrow (\perp, \perp)$
13: **else :**
14: $W \xleftarrow{\$} \mathbb{G}_1 \setminus \{\mathcal{O}\}$
15: $D \leftarrow \sum_{j=1}^{\mathsf{M}} B_j$
16: $(X, Y) \leftarrow ([u](D - V), W - U)$
17: **for** $j \in [\![\mathsf{M}]\!]$:
18: $r_j \xleftarrow{\$} [\![\min\{2^{\varphi}, q-1\}]\!]$
19: $C_j \leftarrow \left[r_j\right]A_j + W$
20: **if** $\mathcal{O} \in \{\vec{C}, D, X, Y\}$: **return** $\perp$
21: $\mathsf{pub} \leftarrow \left(\vec{A}, \vec{B}, \vec{C}, D, X, Y\right)$
22: $\mathsf{sec} \leftarrow \vec{r}$
23: **return** $(\mathsf{pub}, \mathsf{sec})$

OneTimeSetup$(\mathsf{aux.sec}) \to \mathsf{sk}$

1: **parse** $\mathsf{aux.sec} = (\tau, \varphi)$
2: $s \xleftarrow{\$} \mathbb{Z}_q^*$; $\xi \leftarrow \gamma_T^s$
3: $\mathsf{t_{start}} \leftarrow \mathtt{time.now}()$
4: **return** $(s, \xi, \tau, \varphi, \mathsf{t_{start}})$

Compute$(\mathsf{pub}) \to \mathsf{out}$

1: **parse** $\mathsf{pub} = \left(\vec{A}, \vec{B}, \vec{C}, D, X, Y\right)$
2: $\mathsf{M} \leftarrow \mathtt{len}(\vec{A})$
3: **for** $j \in [\![\mathsf{M}]\!]$: $\rho_j = e(A_j, B_j)$
4: **if** $\mathsf{M} = 1$: $\gamma \leftarrow e(A, D) \cdot e(C, Q)$
5: **else :**
6: $\gamma \leftarrow \left(\prod_{j=1}^{\mathsf{M}} e(C_j, -B_j)\right) \cdot e(Y, D) \cdot e(P, X)$
7: **return** $(\gamma, \vec{\rho})$

Verify$(\mathsf{sk}, \mathsf{sec}, \mathsf{out}) \to \mathtt{result}$

1: **parse** $\mathsf{sk} = (\cdot, \xi, \tau, \cdot, \mathsf{t_{start}})$
2: **if** $\mathtt{time.now}() - \mathsf{t_{start}} > \tau$: **return** $\perp$
3: **parse** $\mathsf{sec} = \vec{r}$; **parse** $\mathsf{out} = (\gamma, \vec{\rho})$
4: $\mathsf{M} \leftarrow \mathtt{len}(\vec{r})$
5: **if** $\bigvee_{j \in [\![\mathsf{M}]\!]} \left(\rho_j \notin \mathbb{G}_T\right)$: **return** $\perp$
6: **if** $\xi = \left(\prod_{j=1}^{\mathsf{M}} \rho_j^{r_j}\right) \cdot \gamma$: **return** $\vec{\rho}$
7: **return** $\perp$

AmorE

Theorem 1:

$(n$ public values$)$
$\wedge$
$\left(n+1 \text{ secrets in } \mathbb{Z}_q^*\right)$
$\Downarrow$
Unconditional security.

Setup$(\mathsf{sk}, \vec{A}, \vec{B}) \to (\mathsf{pub}, \mathsf{sec})$

1: **parse** $\mathsf{sk} = (s, \cdot, \cdot, \varphi, \cdot)$
2: **if** $\mathtt{len}(\vec{A}) \neq \mathtt{len}(\vec{B})$:
3: **return** $\bot$
4: **if** $\mathcal{O} \in \vec{A} \cup \vec{B}$: **return** $\bot$
5: $u \xleftarrow{\$} \mathbb{Z}_q^*$
6: $(U, V) \leftarrow \left([u]P,\ [s \cdot u^{-1}]Q\right)$
7: $\mathsf{M} \leftarrow \mathtt{len}(\vec{A})$
8: **if** $\mathsf{M} = 1$:
9: $r \xleftarrow{\$} [\![\min\{2^\varphi, q-1\}]\!]$
10: $C \leftarrow \left[-s \cdot u^{-1}\right](U + A)$
11: $D \leftarrow V - [r]B$
12: $(X, Y) \leftarrow (\bot, \bot)$
13: **else :**
14: $W \xleftarrow{\$} \mathbb{G}_1 \setminus \{\mathcal{O}\}$
15: $D \leftarrow \sum_{j=1}^{\mathsf{M}} B_j$
16: $(X, Y) \leftarrow ([u](D - V), W - U)$
17: **for** $j \in [\![\mathsf{M}]\!]$:
18: $r_j \xleftarrow{\$} [\![\min\{2^\varphi, q-1\}]\!]$
19: $C_j \leftarrow \left[r_j\right] A_j + W$
20: **if** $\mathcal{O} \in \{\vec{C}, D, X, Y\}$: **return** $\bot$
21: $\mathsf{pub} \leftarrow \left(\vec{A}, \vec{B}, \vec{C}, D, X, Y\right)$
22: $\mathsf{sec} \leftarrow \vec{r}$
23: **return** $(\mathsf{pub}, \mathsf{sec})$

OneTimeSetup$(\mathsf{aux.sec}) \to \mathsf{sk}$

1: **parse** $\mathsf{aux.sec} = (\tau, \varphi)$
2: $s \xleftarrow{\$} \mathbb{Z}_q^*$; $\xi \leftarrow \gamma_T^s$
3: $\mathsf{t_{start}} \leftarrow \mathtt{time.now}()$
4: **return** $(s, \xi, \tau, \varphi, \mathsf{t_{start}})$

Compute$(\mathsf{pub}) \to \mathsf{out}$

1: **parse** $\mathsf{pub} = \left(\vec{A}, \vec{B}, \vec{C}, D, X, Y\right)$
2: $\mathsf{M} \leftarrow \mathtt{len}(\vec{A})$
3: **for** $j \in [\![\mathsf{M}]\!]$: $\rho_j = e(A_j, B_j)$
4: **if** $\mathsf{M} = 1$: $\gamma \leftarrow e(A, D) \cdot e(C, Q)$
5: **else :**
6: $\gamma \leftarrow \left(\prod_{j=1}^{\mathsf{M}} e(C_j, -B_j)\right) \cdot e(Y, D) \cdot e(P, X)$
7: **return** $(\gamma, \vec{\rho})$

Verify$(\mathsf{sk}, \mathsf{sec}, \mathsf{out}) \to \mathtt{result}$

1: **parse** $\mathsf{sk} = (\cdot, \xi, \tau, \cdot, \mathsf{t_{start}})$
2: **if** $\mathtt{time.now}() - \mathsf{t_{start}} > \tau$: **return** $\bot$
3: **parse** $\mathsf{sec} = \vec{r}$; **parse** $\mathsf{out} = (\gamma, \vec{\rho})$
4: $\mathsf{M} \leftarrow \mathtt{len}(\vec{r})$
5: **if** $\bigvee_{j \in [\![\mathsf{M}]\!]} \left(\rho_j \notin \mathbb{G}_T\right)$: **return** $\bot$
6: **if** $\xi = \left(\prod_{j=1}^{\mathsf{M}} \rho_j^{r_j}\right) \cdot \gamma$: **return** $\vec{\rho}$
7: **return** $\bot$

AmorE

Theorem 1:

$(n$ public values$)$
$\wedge$
$\left(n+1 \text{ secrets in } \mathbb{Z}_q^*\right)$
$\Downarrow$
Unconditional security.

Setup(sk, $\vec{A}, \vec{B}$) → (pub, sec)

1: **parse** sk = $(s, \cdot, \cdot, \varphi, \cdot)$
2: **if** $\mathrm{len}(\vec{A}) \neq \mathrm{len}(\vec{B})$:
3: **return** $\perp$
4: **if** $\mathcal{O} \in \vec{A} \cup \vec{B}$: **return** $\perp$
5: $u \xleftarrow{\$} \mathbb{Z}_q^*$
6: $(U, V) \leftarrow \left([u]P,\ [s \cdot u^{-1}]Q\right)$
7: $\mathsf{M} \leftarrow \mathrm{len}(\vec{A})$
8: **if** $\mathsf{M} = 1$:
9: $r \xleftarrow{\$} [\![\min\{2^\varphi, q-1\}]\!]$
10: $C \leftarrow \left[-s \cdot u^{-1}\right](U + A)$
11: $D \leftarrow V - [r]B$
12: $(X, Y) \leftarrow (\perp, \perp)$
13: **else :**
14: $W \xleftarrow{\$} \mathbb{G}_1 \setminus \{\mathcal{O}\}$
15: $D \leftarrow \sum_{j=1}^{\mathsf{M}} B_j$
16: $(X, Y) \leftarrow ([u](D - V), W - U)$
17: **for** $j \in [\![\mathsf{M}]\!]$:
18: $r_j \xleftarrow{\$} [\![\min\{2^\varphi, q-1\}]\!]$
19: $C_j \leftarrow \left[r_j\right]A_j + W$
20: **if** $\mathcal{O} \in \{\vec{C}, D, X, Y\}$: **return** $\perp$
21: $\mathsf{pub} \leftarrow \left(\vec{A}, \vec{B}, \vec{C}, D, X, Y\right)$
22: $\mathsf{sec} \leftarrow \vec{r}$
23: **return** (pub, sec)

OneTimeSetup(aux.sec) → sk

1: **parse** aux.sec = (τ, φ)
2: $s \xleftarrow{\$} \mathbb{Z}_q^*$; $\xi \leftarrow \gamma_T^s$
3: $\mathsf{t}_{\mathrm{start}} \leftarrow \mathtt{time.now}()$
4: **return** $(s, \xi, \tau, \varphi, \mathsf{t}_{\mathrm{start}})$

Compute(pub) → out

1: **parse** $\mathsf{pub} = \left(\vec{A}, \vec{B}, \vec{C}, D, X, Y\right)$
2: $\mathsf{M} \leftarrow \mathrm{len}(\vec{A})$
3: **for** $j \in [\![\mathsf{M}]\!]$: $\rho_j = e(A_j, B_j)$
4: **if** $\mathsf{M} = 1$: $\gamma \leftarrow e(A, D) \cdot e(C, Q)$
5: **else :**
6: $\gamma \leftarrow \left(\prod_{j=1}^{\mathsf{M}} e(C_j, -B_j)\right) \cdot e(Y, D) \cdot e(P, X)$
7: **return** $(\gamma, \vec{\rho})$

Verify(sk, sec, out) → result

1: **parse** $\mathsf{sk} = (\cdot, \xi, \tau, \cdot, \mathsf{t}_{\mathrm{start}})$
2: **if** $\mathtt{time.now}() - \mathsf{t}_{\mathrm{start}} > \tau$: **return** $\perp$
3: **parse** $\mathsf{sec} = \vec{r}$; **parse** $\mathsf{out} = (\gamma, \vec{\rho})$
4: $\mathsf{M} \leftarrow \mathrm{len}(\vec{r})$
5: **if** $\bigvee_{j \in [\![\mathsf{M}]\!]} \left(\rho_j \notin \mathbb{G}_T\right)$: **return** $\perp$
6: **if** $\xi = \left(\prod_{j=1}^{\mathsf{M}} \rho_j^{r_j}\right) \cdot \gamma$: **return** $\vec{\rho}$
7: **return** $\perp$

AmorE

Theorem 1:

$(n$ public values$)$
$\wedge$
$\left(n+1 \text{ secrets in } \mathbb{Z}_q^*\right)$
$\Downarrow$
Unconditional security.

Theorem 2:

$(n$ public values$)$
$\wedge$
$\left(n+1 \text{ secrets in } [\![2^\varphi]\!] \subset \mathbb{Z}_q^*\right)$
$\Downarrow$
Everlasting security.

Setup(sk, $\vec{A}, \vec{B}$) → (pub, sec)

1: **parse** sk = $(s, \cdot, \cdot, \varphi, \cdot)$
2: **if** $\mathrm{len}(\vec{A}) \neq \mathrm{len}(\vec{B})$:
3: **return** $\perp$
4: **if** $\mathcal{O} \in \vec{A} \cup \vec{B}$: **return** $\perp$
5: $u \xleftarrow{\$} \mathbb{Z}_q^*$
6: $(U, V) \leftarrow \left([u]P,\ [s \cdot u^{-1}]Q\right)$
7: $\mathsf{M} \leftarrow \mathrm{len}(\vec{A})$
8: **if** $\mathsf{M} = 1$:
9: $r \xleftarrow{\$} [\![\min\{2^\varphi, q-1\}]\!]$
10: $C \leftarrow \left[-s \cdot u^{-1}\right](U + A)$
11: $D \leftarrow V - [r]B$
12: $(X, Y) \leftarrow (\perp, \perp)$
13: **else :**
14: $W \xleftarrow{\$} \mathbb{G}_1 \setminus \{\mathcal{O}\}$
15: $D \leftarrow \sum_{j=1}^{\mathsf{M}} B_j$
16: $(X, Y) \leftarrow ([u](D - V), W - U)$
17: **for** $j \in [\![\mathsf{M}]\!]$:
18: $r_j \xleftarrow{\$} [\![\min\{2^\varphi, q-1\}]\!]$
19: $C_j \leftarrow \left[r_j\right]A_j + W$
20: **if** $\mathcal{O} \in \{\vec{C}, D, X, Y\}$: **return** $\perp$
21: $\mathsf{pub} \leftarrow \left(\vec{A}, \vec{B}, \vec{C}, D, X, Y\right)$
22: $\mathsf{sec} \leftarrow \vec{r}$
23: **return** (pub, sec)

OneTimeSetup(aux.sec) → sk

1: **parse** aux.sec = (τ, φ)
2: $s \xleftarrow{\$} \mathbb{Z}_q^*$; $\xi \leftarrow \gamma_T^s$
3: $\mathrm{t_{start}} \leftarrow \mathrm{time.now}()$
4: **return** $(s, \xi, \tau, \varphi, \mathrm{t_{start}})$

Compute(pub) → out

1: **parse** $\mathsf{pub} = \left(\vec{A}, \vec{B}, \vec{C}, D, X, Y\right)$
2: $\mathsf{M} \leftarrow \mathrm{len}(\vec{A})$
3: **for** $j \in [\![\mathsf{M}]\!]$: $\rho_j = e(A_j, B_j)$
4: **if** $\mathsf{M} = 1$: $\gamma \leftarrow e(A, D) \cdot e(C, Q)$
5: **else :**
6: $\gamma \leftarrow \left(\prod_{j=1}^{\mathsf{M}} e(C_j, -B_j)\right) \cdot e(Y, D) \cdot e(P, X)$
7: **return** $(\gamma, \vec{\rho})$

Verify(sk, sec, out) → result

1: **parse** $\mathsf{sk} = (\cdot, \xi, \tau, \cdot, \mathrm{t_{start}})$
2: **if** $\mathrm{time.now}() - \mathrm{t_{start}} > \tau$: **return** $\perp$
3: **parse** $\mathsf{sec} = \vec{r}$; **parse** $\mathsf{out} = (\gamma, \vec{\rho})$
4: $\mathsf{M} \leftarrow \mathrm{len}(\vec{r})$
5: **if** $\bigvee_{j \in [\![\mathsf{M}]\!]} \left(\rho_j \notin \mathbb{G}_T\right)$: **return** $\perp$
6: **if** $\xi = \left(\prod_{j=1}^{\mathsf{M}} \rho_j^{r_j}\right) \cdot \gamma$: **return** $\vec{\rho}$
7: **return** $\perp$

AmorE

Theorem 1:

$(n$ public values$)$
$\wedge$
$\left(n+1 \text{ secrets in } \mathbb{Z}_q^*\right)$
$\Downarrow$
Unconditional security.

Theorem 2:

$(n$ public values$)$
$\wedge$
$\left(n+1 \text{ secrets in } [\![2^\varphi]\!] \subset \mathbb{Z}_q^*\right)$
$\Downarrow$
Everlasting security.

Meet-in-the-middle attack: A toy example

$$\texttt{pub} = \begin{cases} C = [r]\xi + X \\ D = [t]\xi + Y \end{cases}$$

Meet-in-the-middle attack: A toy example

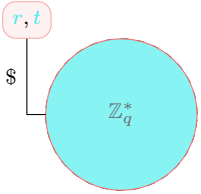
$$\texttt{pub} = \begin{cases} C = [r]\xi + X \\ D = [t]\xi + Y \end{cases}$$

$$\begin{aligned} \xi &= [r^{-1}](C - X) \\ &= [t^{-1}](D - Y) \end{aligned}$$

Meet-in-the-middle attack: A toy example

$$\mathtt{pub} = \begin{cases} C = [r]\xi + X \\ D = [t]\xi + Y \end{cases}$$

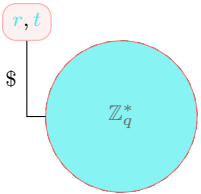
$$\begin{aligned} \xi &= [r^{-1}](C - X) \\ &= [t^{-1}](D - Y) \end{aligned}$$

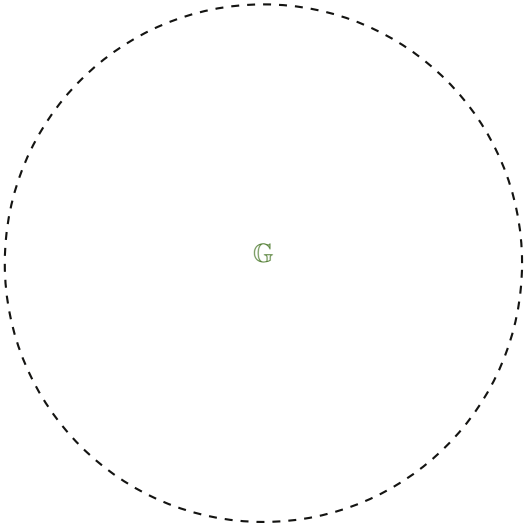


Meet-in-the-middle attack: A toy example

$$\mathtt{pub} = \begin{cases} C = [r]\xi + X \\ D = [t]\xi + Y \end{cases}$$

$$\begin{aligned} \xi &= [r^{-1}](C - X) \\ &= [t^{-1}](D - Y) \end{aligned}$$

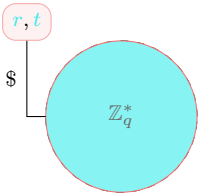


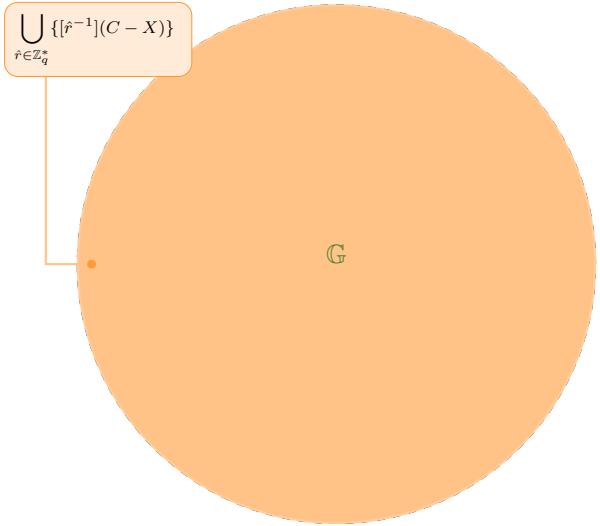


Meet-in-the-middle attack: A toy example

$$\mathtt{pub} = \begin{cases} C = [r]\xi + X \\ D = [t]\xi + Y \end{cases}$$

$$\begin{aligned} \xi &= [r^{-1}](C - X) \\ &= [t^{-1}](D - Y) \end{aligned}$$

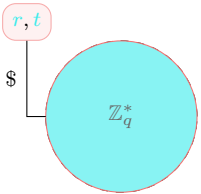


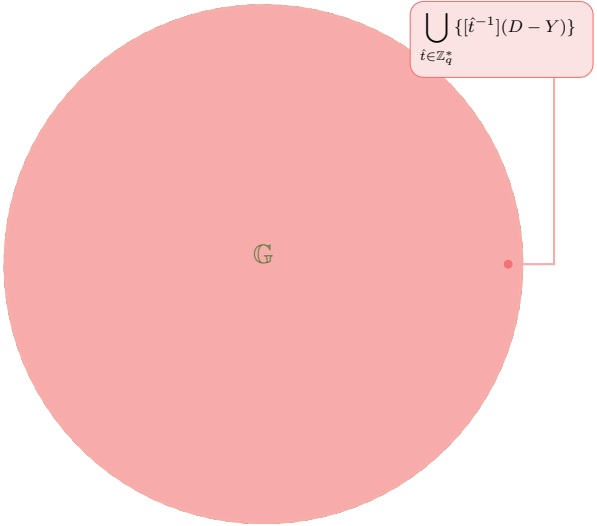


Meet-in-the-middle attack: A toy example

$$\mathbf{pub} = \begin{cases} C = [r]\xi + X \\ D = [t]\xi + Y \end{cases}$$

$$\begin{aligned} \xi &= [r^{-1}](C - X) \\ &= [t^{-1}](D - Y) \end{aligned}$$

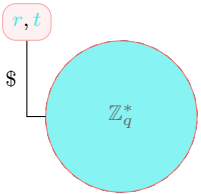


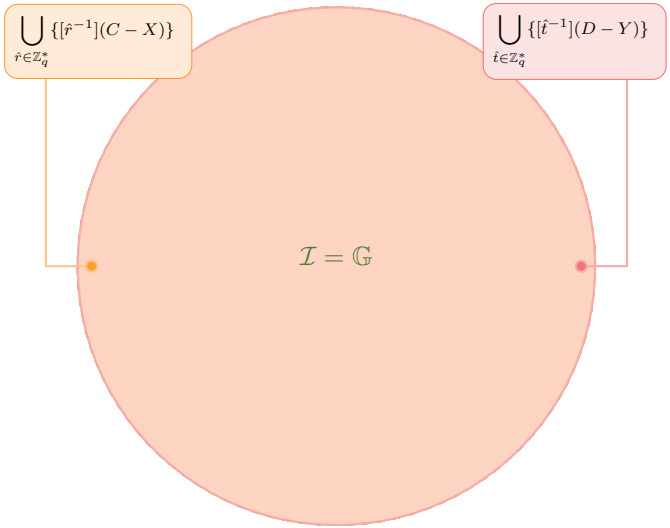


Meet-in-the-middle attack: A toy example

$$\mathbf{pub} = \begin{cases} C = [r]\xi + X \\ D = [t]\xi + Y \end{cases}$$

$$\begin{aligned} \xi &= [r^{-1}](C - X) \\ &= [t^{-1}](D - Y) \end{aligned}$$

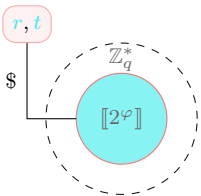


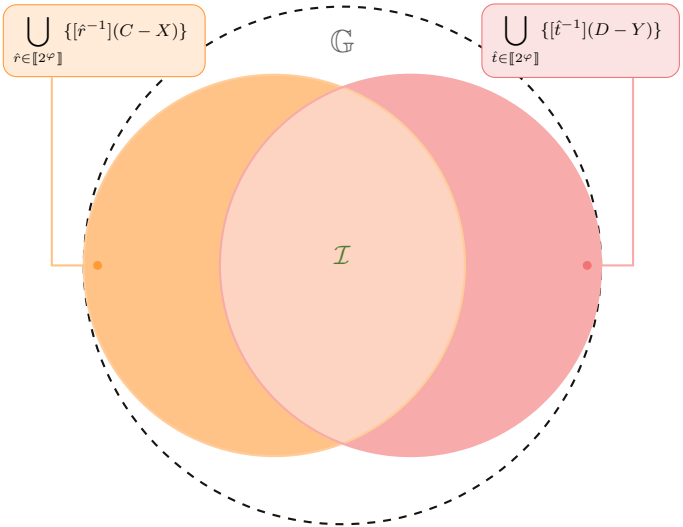


Meet-in-the-middle attack: A toy example

$$\mathbf{pub} = \begin{cases} C = [r]\xi + X \\ D = [t]\xi + Y \end{cases}$$

$$\begin{aligned} \xi &= [r^{-1}](C - X) \\ &= [t^{-1}](D - Y) \end{aligned}$$

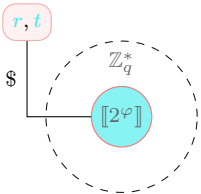


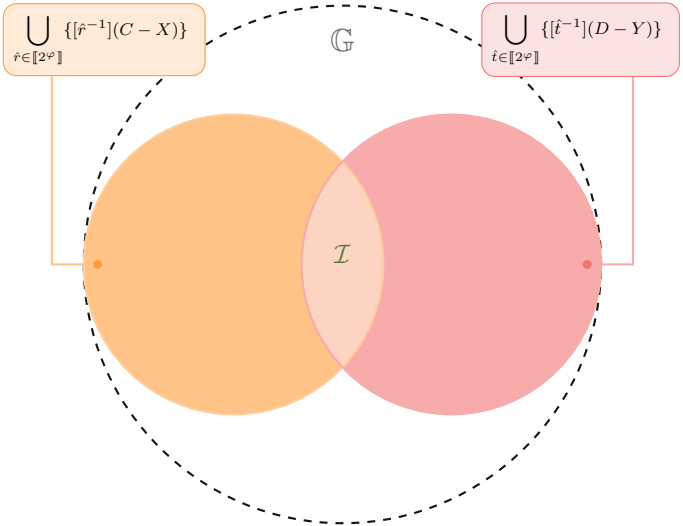


Meet-in-the-middle attack: A toy example

$$\text{pub} = \begin{cases} C = [r]\xi + X \\ D = [t]\xi + Y \end{cases}$$

$$\begin{aligned} \xi &= [r^{-1}](C - X) \\ &= [t^{-1}](D - Y) \end{aligned}$$

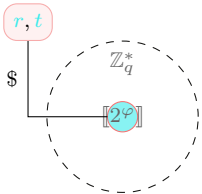


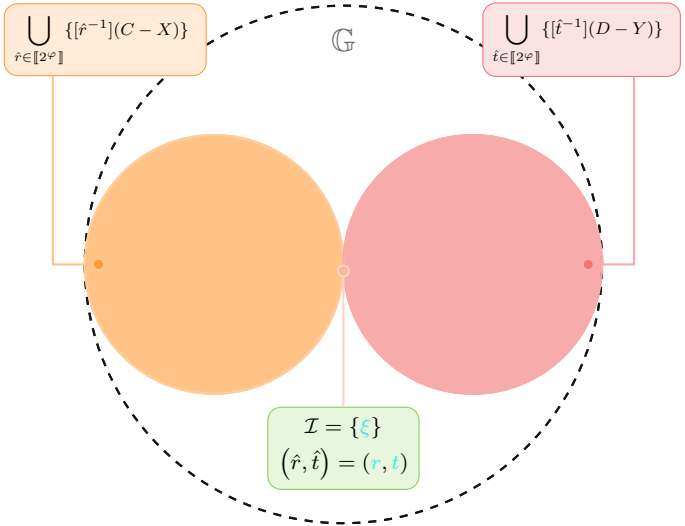


Meet-in-the-middle attack: A toy example

$$\mathbf{pub} = \begin{cases} C = [r]\xi + X \\ D = [t]\xi + Y \end{cases}$$

$$\begin{aligned} \xi &= [r^{-1}](C - X) \\ &= [t^{-1}](D - Y) \end{aligned}$$

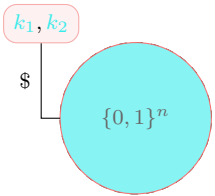


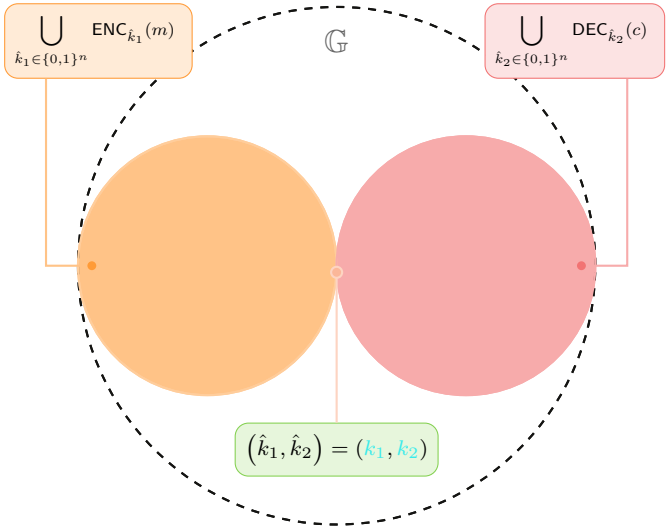


Analogy: Double DES

$$\begin{cases} m = \mathsf{DEC}_{k_1}\left(\mathsf{DEC}_{k_2}\left(c\right)\right) \\ c = \mathsf{ENC}_{k_2}\left(\mathsf{ENC}_{k_1}\left(m\right)\right) \end{cases}$$

$$\mathsf{DEC}_{k_2}\left(c\right) = \mathsf{ENC}_{k_1}\left(m\right)$$





Adversary winning probability

Time-bound: $\mathcal{A}$ computes $\leq 2^{\kappa}$ short exponentations in τ seconds.

Adversary winning probability

Time-bound: $\mathcal{A}$ computes $\leq 2^{\kappa}$ short exponentations in τ seconds.
Best strategy: Choose $S_1, S_2 \subset [\![2^{\varphi}]\!] : |S_1| + |S_2| \leq 2^{\kappa}$ and intersect sets.

Adversary winning probability

Time-bound: $\mathcal{A}$ computes $\leq 2^{\kappa}$ short exponentations in τ seconds.
Best strategy: Choose $S_1, S_2 \subset [\![2^{\varphi}]\!] : |S_1| + |S_2| \leq 2^{\kappa}$ and intersect sets.

$$\mathcal{P}\left[\begin{array}{l|c} \mathtt{out}^* \leftarrow \mathcal{A}(\mathtt{pub}) & \begin{array}{c} \mathtt{out}^* \neq \mathtt{out} \\ \mathsf{Verify}(\mathtt{sk}, \mathtt{sec}, \mathtt{out}) \neq \bot \end{array} \end{array} \right]$$

Adversary winning probability

Time-bound: $\mathcal{A}$ computes $\leq 2^{\kappa}$ short exponentations in τ seconds.
Best strategy: Choose $S_1, S_2 \subset [\![2^{\varphi}]\!] : |S_1| + |S_2| \leq 2^{\kappa}$ and intersect sets.

$$\mathcal{P}\left[\begin{array}{l} \mathtt{out}^* \leftarrow \mathcal{A}(\mathtt{pub}) \end{array} \;\middle|\; \begin{array}{c} \mathtt{out}^* \neq \mathtt{out} \\ \mathsf{Verify}(\mathtt{sk}, \mathtt{sec}, \mathtt{out}) \neq \perp \end{array} \right]$$

$$\leq \mathcal{P}\left[\xi \in \begin{array}{c} \text{set } \{[\hat{r}^{-1}](C - X) : \ \hat{r} \in S_1\} \\ \cap \\ \text{set } \{[\hat{t}^{-1}](D - Y) : \ \hat{t} \in S_2\} \end{array} \right]$$

Adversary winning probability

Time-bound: $\mathcal{A}$ computes $\leq 2^{\kappa}$ short exponentations in τ seconds.
Best strategy: Choose $S_1, S_2 \subset [\![2^{\varphi}]\!] : \ |S_1| + |S_2| \leq 2^{\kappa}$ and intersect sets.

$$\mathcal{P}\left[\begin{array}{l} \mathtt{out}^* \leftarrow \mathcal{A}(\mathtt{pub}) \end{array} \middle| \begin{array}{c} \mathtt{out}^* \neq \mathtt{out} \\ \mathsf{Verify}(\mathtt{sk}, \mathtt{sec}, \mathtt{out}) \neq \perp \end{array} \right]$$

$$\leq \mathcal{P}\left[\xi \in \begin{array}{c} \text{set } \{[\hat{r}^{-1}](C - X) : \ \hat{r} \in S_1\} \\ \cap \\ \text{set } \{[\hat{t}^{-1}](D - Y) : \ \hat{t} \in S_2\} \end{array} \right]$$

$$\leq \mathcal{P}\left[\begin{array}{c} \text{event } \{r \in S_1\} \\ \wedge \\ \text{event } \{t \in S_2\} \end{array} \right] \leq 2^{-\sigma} \quad \text{if } \varphi = \left\lceil \frac{\sigma - 1}{2} + \kappa \right\rceil$$

Amortized efficiency for pairing delegation with public inputs

Amortized efficiency for pairing delegation with public inputs ✓

Amortized efficiency for pairing delegation with public inputs ✓

Can we apply similar techniques in the input-privacy case? 

This AmorE is Private: Amortized Efficiency for Private Pairing Delegation

Adrián P. Keilty[1], Diego F. Aranha[2], and Elena Pagnin[1]

[1] Chalmers University of Technology, Sweden
[2] Aarhus University, Denmark

(Under submission)

Private-Input Pairing Delegation: Applications

Convention: S = Secret, P = Public, V = Variable, C = Constant.

Type	Single	Batch
PVPV	[39, 47, 7, 40, 24, 62]	[25, 16, 23, 55, 13, 49, 68, 3, 56, 32]
PVSV **or** SVPV	[27, 84, 52, 41]	[26]
SVSV	Basis for building delegation protocol variants in [38, 66].	(no documented applications)
SVSC	[17, 35, 41]	Potential but undocumented application: verifying multiple signatures from the same signer.
SVPC	[73, 59]	[15]
PVSC	[22, 52, 81, 40, 44, 31, 92, 34, 35]	[15, 54, 1]
PVPC	[22, 92, 19, 33, 47, 59, 67, 88, 3]	[16, 23, 29, 39]

Applications: Verifiable pairing [39]; VRF [47, 33]; Ciphertext validity checking [7, 40, 52]; Doubly homomorphic encryption [24]; Tripartite key exchange [62]; BLS short signatures [25]; Multisignature/blind signature [16]; Aggregate/ring/verifiably encrypted signature [23]; Groth16 [55]; SNARKs [13]; SnarkPack [49]; Polynomial commitments [68, 3]; Homomorphic proof commitments [56]; Secret leader election [32]; Searchable encryption [27, 84]; ID-based key agreement [41, 35]; Trace-and-revoke broadcast [26]; Used as a basis for constructing different delegation protocol types [38, 66]; Inversion IBE [17]; ID-based identification [73]; ID-based signature [59]; Ciphertext-policy ABE [15]; IBE [22, 52, 81]; Certificateless encryption [40, 44]; Forward-secure encryption [31, 92]; Key agreement [34]; ABE [54]; ID-based broadcast encryption [1]; Public key encryption with keyword search [19]; Identity-based onion routing [67]; Trusted computing [88]; aPlonk inner product argument [3]; Group signature [29].

Current Input-privacy Experiment: **IND-privacy**

IND-privacy: $\mathbf{Exp}^{\mathsf{seq,IND-priv}}_{\mathbf{\Pi},\mathcal{A}}(\lambda, \mathtt{aux.sec}, L)$

1 : $\mathtt{pp} \leftarrow \mathsf{GlobalSetup}(\lambda)$; $\mathtt{sk} \leftarrow \mathsf{OneTimeSetup}(\mathtt{aux.sec})$

2 : $i \leftarrow 0$; $\mathsf{ctr} \leftarrow 0$; $\mathtt{st} \leftarrow (\lambda, \mathtt{aux.sec}, \mathtt{pp})$

3 : **while** true **:**

4 : $\left((\vec{A}_0, \vec{B}_0), (\vec{A}_1, \vec{B}_1)\right) \leftarrow \mathcal{A}(\mathtt{st})$

5 : $\mathsf{ctr} \leftarrow \mathsf{ctr} + \mathtt{len}\left(\vec{A}_i\right)$; **if** $\mathsf{ctr} \geq L$ **:** **break**

6 : $i \leftarrow i + 1$; $b_i \xleftarrow{\$} \{0,1\}$; $\left(\vec{A}, \vec{B}\right) \leftarrow \left(\vec{A}_{b_i}, \vec{B}_{b_i}\right)$

7 : $(\mathtt{pub}, \mathtt{sec}) \leftarrow \mathsf{Setup}\left(\mathtt{sk}, \vec{A}_i, \vec{B}_i\right)$; $\mathtt{st} \leftarrow \mathcal{A}(\mathtt{pub}, \mathtt{st})$

8 : $(i^*, b^*) \leftarrow \mathcal{A}(\mathtt{st})$; **if** $b^* = b_{i^*}$ **:** **return** 1

9 : **return** 0

IND-privacy: $\mathbf{Exp}^{\mathsf{seq,IND-priv}}_{\mathbf{\Pi},\mathcal{A}}(\lambda, \mathtt{aux.sec}, L)$

1 : $\mathtt{pp} \leftarrow \mathsf{GlobalSetup}(\lambda)$; $\mathtt{sk} \leftarrow \mathsf{OneTimeSetup}(\mathtt{aux.sec})$

2 : $i \leftarrow 0$; $\mathsf{ctr} \leftarrow 0$; $\mathtt{st} \leftarrow (\lambda, \mathtt{aux.sec}, \mathtt{pp})$

3 : **while** true **:**

4 : $((\vec{A}_0, \vec{B}_0), (\vec{A}_1, \vec{B}_1)) \leftarrow \mathcal{A}(\mathtt{st})$

5 : $\mathsf{ctr} \leftarrow \mathsf{ctr} + \mathtt{len}\left(\vec{A}_i\right)$; **if** $\mathsf{ctr} \geq L$ **:** **break**

6 : $i \leftarrow i + 1$; $b_i \xleftarrow{\$} \{0,1\}$; $\left(\vec{A}, \vec{B}\right) \leftarrow \left(\vec{A}_{b_i}, \vec{B}_{b_i}\right)$

7 : $(\mathtt{pub}, \mathtt{sec}) \leftarrow \mathsf{Setup}\left(\mathtt{sk}, \vec{A}_i, \vec{B}_i\right)$; $\mathtt{st} \leftarrow \mathcal{A}(\mathtt{pub}, \mathtt{st})$

8 : $(i^*, b^*) \leftarrow \mathcal{A}(\mathtt{st})$; **if** $b^* = b_{i^*}$ **:** **return** 1

9 : **return** 0

1. $\mathcal{A}$ adaptively selects two pairs of inputs.

IND-privacy: $\mathbf{Exp}^{\mathsf{seq,IND-priv}}_{\mathbf{\Pi},\mathcal{A}}(\lambda, \mathtt{aux.sec}, L)$

1 : $\mathtt{pp} \leftarrow \mathsf{GlobalSetup}(\lambda)$; $\mathtt{sk} \leftarrow \mathsf{OneTimeSetup}(\mathtt{aux.sec})$

2 : $i \leftarrow 0$; $\mathtt{ctr} \leftarrow 0$; $\mathtt{st} \leftarrow (\lambda, \mathtt{aux.sec}, \mathtt{pp})$

3 : **while** true **:**

4 : $((\vec{A}_0, \vec{B}_0), (\vec{A}_1, \vec{B}_1)) \leftarrow \mathcal{A}(\mathtt{st})$

5 : $\mathtt{ctr} \leftarrow \mathtt{ctr} + \mathtt{len}\left(\vec{A}_i\right)$; **if** $\mathtt{ctr} \geq L$ **:** **break**

6 : $i \leftarrow i + 1$; $b_i \xleftarrow{\$} \{0,1\}$; $\left(\vec{A}, \vec{B}\right) \leftarrow \left(\vec{A}_{b_i}, \vec{B}_{b_i}\right)$

7 : $(\mathtt{pub}, \mathtt{sec}) \leftarrow \mathsf{Setup}\left(\mathtt{sk}, \vec{A}_i, \vec{B}_i\right)$; $\mathtt{st} \leftarrow \mathcal{A}(\mathtt{pub}, \mathtt{st})$

8 : $(i^*, b^*) \leftarrow \mathcal{A}(\mathtt{st})$; **if** $b^* = b_{i^*}$ **:** **return** 1

9 : **return** 0

1. $\mathcal{A}$ adaptively selects two pairs of inputs.

2. $\mathcal{C}$ randomly delegates one of them.

IND-privacy: $\mathbf{Exp}^{\mathsf{seq,IND-priv}}_{\mathbf{\Pi},\mathcal{A}}(\lambda, \mathtt{aux.sec}, L)$

1 : $\mathtt{pp} \leftarrow \mathsf{GlobalSetup}(\lambda)$; $\mathtt{sk} \leftarrow \mathsf{OneTimeSetup}(\mathtt{aux.sec})$

2 : $i \leftarrow 0$; $\mathsf{ctr} \leftarrow 0$; $\mathtt{st} \leftarrow (\lambda, \mathtt{aux.sec}, \mathtt{pp})$

3 : **while** true **:**

4 : $((\vec{A}_0, \vec{B}_0), (\vec{A}_1, \vec{B}_1)) \leftarrow \mathcal{A}(\mathtt{st})$

5 : $\mathsf{ctr} \leftarrow \mathsf{ctr} + \mathtt{len}\left(\vec{A}_i\right)$; **if** $\mathsf{ctr} \geq L$ **:** **break**

6 : $i \leftarrow i + 1$; $b_i \xleftarrow{\$} \{0,1\}$; $\left(\vec{A}, \vec{B}\right) \leftarrow \left(\vec{A}_{b_i}, \vec{B}_{b_i}\right)$

7 : $(\mathtt{pub}, \mathtt{sec}) \leftarrow \mathsf{Setup}\left(\mathtt{sk}, \vec{A}_i, \vec{B}_i\right)$; $\mathtt{st} \leftarrow \mathcal{A}(\mathtt{pub}, \mathtt{st})$

8 : $(i^*, b^*) \leftarrow \mathcal{A}(\mathtt{st})$; **if** $b^* = b_{i^*}$ **:** **return** 1

9 : **return** 0

1. $\mathcal{A}$ adaptively selects two pairs of inputs.

2. $\mathcal{C}$ randomly delegates one of them.

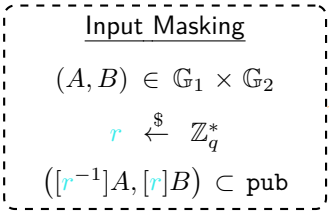
3. From pub, $\mathcal{A}$ guesses which one was delegated.

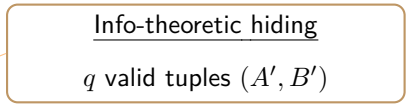
Input Masking

$$(A, B) \in \mathbb{G}_1 \times \mathbb{G}_2$$

$$r \xleftarrow{\$} \mathbb{Z}_q^*$$

$$\left([r^{-1}]A, [r]B\right) \subset \texttt{pub}$$


Input Masking
$(A, B) \in \mathbb{G}_1 \times \mathbb{G}_2$
$r \xleftarrow{\$} \mathbb{Z}_q^*$
$\left([r^{-1}]A, [r]B\right) \subset \texttt{pub}$


Info-theoretic hiding
q valid tuples (A', B')

Input Masking

$(A, B) \in \mathbb{G}_1 \times \mathbb{G}_2$

$r \xleftarrow{\$} \mathbb{Z}_q^*$

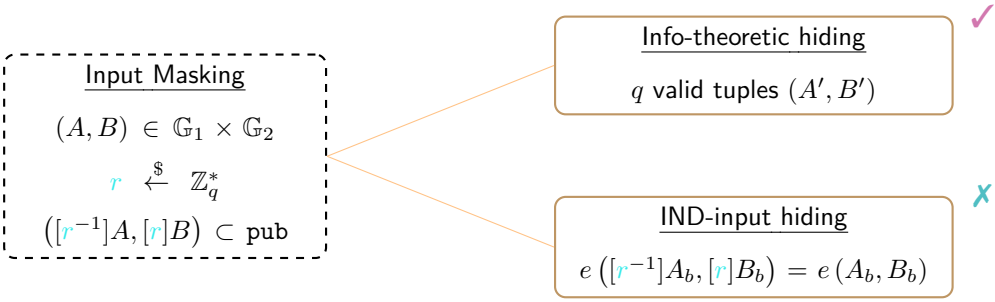
$\left([r^{-1}]A, [r]B\right) \subset \texttt{pub}$

Info-theoretic hiding ✓

q valid tuples (A', B')

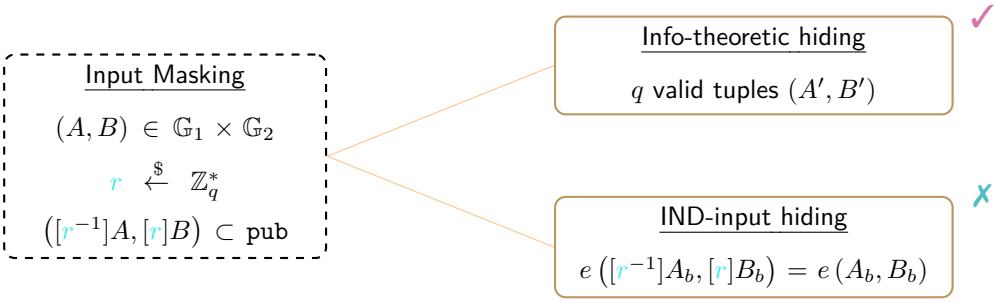
IND-input hiding ✗

$e\left([r^{-1}]A_b, [r]B_b\right) = e\left(A_b, B_b\right)$



Lemma 2

A protocol $\mathbf{\Pi}$ that is IND-input private must also be output-private.



Lemma 2

A protocol $\mathbf{\Pi}$ that is IND-input private must also be output-private.

Can we come up with an alternate experiment?

Proposed Variant:

One-Way Input Privacy

OW-privacy: $\mathbf{Exp}^{\mathsf{seq,OW-priv}}_{\mathbf{\Pi},\mathcal{A}}(\lambda, \mathsf{aux.sec}, L, \mathsf{type})$

$\mathsf{pp} \leftarrow \mathsf{GlobalSetup}(\lambda)$; $\mathtt{sk} \leftarrow \mathsf{OneTimeSetup}(\mathsf{aux.sec})$

$i \leftarrow 0$; $\mathsf{ctr} \leftarrow 0$; $\mathtt{st} \leftarrow (\lambda, \mathsf{aux.sec}, \mathsf{pp})$

while true **:**

  switch (type)

    case SVSV **:** // A SECRET, B SECRET

      $\mathsf{Dist}^{\lambda}_{\infty}(\mathbb{G}_1 \times \mathbb{G}_2)\big|_{\mathsf{marg}} \ni \vec{\mathcal{D}} \leftarrow \mathcal{A}(\mathtt{st})$; $\vec{A} \xleftarrow{\vec{\mathcal{D}}_{\mathbb{G}_1}} \mathbb{G}_1$; $\vec{B} \xleftarrow{\vec{\mathcal{D}}_{\mathbb{G}_2}} \mathbb{G}_2$

    case SVPV **:** // A SECRET, B PUBLIC

      $\mathsf{Dist}^{\lambda}_{\infty}(\mathbb{G}_1) \ni \vec{\mathcal{D}} \leftarrow \mathcal{A}(\mathtt{st})$; $\vec{A} \xleftarrow{\vec{\mathcal{D}}} \mathbb{G}_1$; $\vec{B} \leftarrow \mathcal{A}(\mathtt{st})$

    case PVSV **:** // A PUBLIC, B SECRET

      $\mathsf{Dist}^{\lambda}_{\infty}(\mathbb{G}_2) \ni \vec{\mathcal{D}} \leftarrow \mathcal{A}(\mathtt{st})$; $\vec{B} \xleftarrow{\vec{\mathcal{D}}} \mathbb{G}_2$; $\vec{A} \leftarrow \mathcal{A}(\mathtt{st})$

  $\mathsf{ctr} \leftarrow \mathsf{ctr} + \mathtt{len}\left(\vec{A}_i\right)$; **if** $\mathsf{ctr} \geq L$ **:** **break**

  $(\mathsf{pub}, \mathsf{sec}) \leftarrow \mathsf{Setup}\left(\mathtt{sk}, \vec{A}_i, \vec{B}_i\right)$; $\mathtt{st} \leftarrow \mathcal{A}(\mathsf{pub}, \mathtt{st})$

if $\mathsf{type} \in \{\mathsf{SVSV}, \mathsf{SVPV}\}$ **:**

  $A^* \leftarrow \mathcal{A}(\mathtt{st})$; **if** $A^* \in \vec{A}$ **:** **return** 1

if $\mathsf{type} \in \{\mathsf{SVSV}, \mathsf{PVSV}\}$ **:**

  $B^* \leftarrow \mathcal{A}(\mathtt{st})$; **if** $B^* \in \vec{B}$ **:** **return** 1

return 0

OW-privacy: $\mathbf{Exp}^{\mathsf{seq,OW-priv}}_{\mathbf{\Pi},\mathcal{A}}(\lambda, \mathsf{aux.sec}, L, \mathsf{type})$

1 : $\mathsf{pp} \leftarrow \mathsf{GlobalSetup}(\lambda)$; $\mathtt{sk} \leftarrow \mathsf{OneTimeSetup}(\mathsf{aux.sec})$

2 : $i \leftarrow 0$; $\mathsf{ctr} \leftarrow 0$; $\mathtt{st} \leftarrow (\lambda, \mathsf{aux.sec}, \mathsf{pp})$

3 : **while** true **:**

4 : **switch** (type)

5 : **case** SVSV **:** // A SECRET, B SECRET

6 : $\mathsf{Dist}^{\lambda}_{\infty}(\mathbb{G}_1 \times \mathbb{G}_2)\big|_{\mathsf{marg}} \supset \vec{\mathcal{D}} \leftarrow \mathcal{A}(\mathtt{st})$; $\vec{A} \xleftarrow{\vec{\mathcal{D}}_{\mathbb{G}_1}} \mathbb{G}_1$; $\vec{B} \xleftarrow{\vec{\mathcal{D}}_{\mathbb{G}_2}} \mathbb{G}_2$

7 : **case** SVPV **:** // A SECRET, B PUBLIC

8 : $\mathsf{Dist}^{\lambda}_{\infty}(\mathbb{G}_1) \supset \vec{\mathcal{D}} \leftarrow \mathcal{A}(\mathtt{st})$; $\vec{A} \xleftarrow{\vec{\mathcal{D}}} \mathbb{G}_1$; $\vec{B} \leftarrow \mathcal{A}(\mathtt{st})$

9 : **case** PVSV **:** // A PUBLIC, B SECRET

10 : $\mathsf{Dist}^{\lambda}_{\infty}(\mathbb{G}_2) \supset \vec{\mathcal{D}} \leftarrow \mathcal{A}(\mathtt{st})$; $\vec{B} \xleftarrow{\vec{\mathcal{D}}} \mathbb{G}_2$; $\vec{A} \leftarrow \mathcal{A}(\mathtt{st})$

11 : $\mathsf{ctr} \leftarrow \mathsf{ctr} + \mathtt{len}\left(\vec{A}_i\right)$; **if** $\mathsf{ctr} \geq L$ **:** **break**

12 : $(\mathsf{pub}, \mathsf{sec}) \leftarrow \mathsf{Setup}\left(\mathtt{sk}, \vec{A}_i, \vec{B}_i\right)$; $\mathtt{st} \leftarrow \mathcal{A}(\mathsf{pub}, \mathtt{st})$

13 : **if** $\mathsf{type} \in \{\mathsf{SVSV}, \mathsf{SVPV}\}$ **:**

14 : $A^* \leftarrow \mathcal{A}(\mathtt{st})$; **if** $A^* \in \vec{A}$ **:** **return** 1

15 : **if** $\mathsf{type} \in \{\mathsf{SVSV}, \mathsf{PVSV}\}$ **:**

16 : $B^* \leftarrow \mathcal{A}(\mathtt{st})$; **if** $B^* \in \vec{B}$ **:** **return** 1

17 : **return** 0

For each secret input:

1. $\mathcal{A}$ chooses a λ-min entropy distribution $\mathcal{D}$

OW-privacy: $\mathbf{Exp}^{\mathsf{seq,OW-priv}}_{\mathbf{\Pi},\mathcal{A}}(\lambda, \mathsf{aux.sec}, L, \mathsf{type})$

1 : $\mathsf{pp} \leftarrow \mathsf{GlobalSetup}(\lambda)$; $\mathtt{sk} \leftarrow \mathsf{OneTimeSetup}(\mathsf{aux.sec})$

2 : $i \leftarrow 0$; $\mathsf{ctr} \leftarrow 0$; $\mathtt{st} \leftarrow (\lambda, \mathsf{aux.sec}, \mathsf{pp})$

3 : **while** true **:**

4 : **switch** (type)

5 : **case** SVSV **:** // A SECRET, B SECRET

6 : $\mathsf{Dist}^{\lambda}_{\infty}(\mathbb{G}_1 \times \mathbb{G}_2)\big|_{\mathsf{marg}} \supset \vec{\mathcal{D}} \leftarrow \mathcal{A}(\mathtt{st})$; $\vec{A} \xleftarrow{\vec{\mathcal{D}}_{\mathbb{G}_1}} \mathbb{G}_1$; $\vec{B} \xleftarrow{\vec{\mathcal{D}}_{\mathbb{G}_2}} \mathbb{G}_2$

7 : **case** SVPV **:** // A SECRET, B PUBLIC

8 : $\mathsf{Dist}^{\lambda}_{\infty}(\mathbb{G}_1) \supset \vec{\mathcal{D}} \leftarrow \mathcal{A}(\mathtt{st})$; $\vec{A} \xleftarrow{\vec{\mathcal{D}}} \mathbb{G}_1$; $\vec{B} \leftarrow \mathcal{A}(\mathtt{st})$

9 : **case** PVSV **:** // A PUBLIC, B SECRET

10 : $\mathsf{Dist}^{\lambda}_{\infty}(\mathbb{G}_2) \supset \vec{\mathcal{D}} \leftarrow \mathcal{A}(\mathtt{st})$; $\vec{B} \xleftarrow{\vec{\mathcal{D}}} \mathbb{G}_2$; $\vec{A} \leftarrow \mathcal{A}(\mathtt{st})$

11 : $\mathsf{ctr} \leftarrow \mathsf{ctr} + \mathtt{len}\left(\vec{A}_i\right)$; **if** $\mathsf{ctr} \geq L$ **:** **break**

12 : $(\mathsf{pub}, \mathsf{sec}) \leftarrow \mathsf{Setup}\left(\mathtt{sk}, \vec{A}_i, \vec{B}_i\right)$; $\mathtt{st} \leftarrow \mathcal{A}(\mathsf{pub}, \mathtt{st})$

13 : **if** $\mathsf{type} \in \{\mathsf{SVSV}, \mathsf{SVPV}\}$ **:**

14 : $A^* \leftarrow \mathcal{A}(\mathtt{st})$; **if** $A^* \in \vec{A}$ **:** **return** 1

15 : **if** $\mathsf{type} \in \{\mathsf{SVSV}, \mathsf{PVSV}\}$ **:**

16 : $B^* \leftarrow \mathcal{A}(\mathtt{st})$; **if** $B^* \in \vec{B}$ **:** **return** 1

17 : **return** 0

For each secret input:

1. $\mathcal{A}$ chooses a λ-min entropy distribution $\mathcal{D}$

2. $\mathcal{C}$ samples input from $\mathcal{D}$ and computes pub

OW-privacy: $\mathbf{Exp}^{\mathsf{seq,OW-priv}}_{\mathbf{\Pi},\mathcal{A}}(\lambda, \mathsf{aux.sec}, L, \mathsf{type})$

1 : $\mathsf{pp} \leftarrow \mathsf{GlobalSetup}(\lambda)$; $\mathtt{sk} \leftarrow \mathsf{OneTimeSetup}(\mathsf{aux.sec})$

2 : $i \leftarrow 0$; $\mathsf{ctr} \leftarrow 0$; $\mathtt{st} \leftarrow (\lambda, \mathsf{aux.sec}, \mathsf{pp})$

3 : **while** true **:**

4 : **switch** (type)

5 : **case** SVSV **:** // A SECRET, B SECRET

6 : $\mathsf{Dist}^{\lambda}_{\infty}(\mathbb{G}_1 \times \mathbb{G}_2)\big|_{\mathsf{marg}} \supset \vec{\mathcal{D}} \leftarrow \mathcal{A}(\mathtt{st})$; $\vec{A} \xleftarrow{\vec{\mathcal{D}}_{\mathbb{G}_1}} \mathbb{G}_1$; $\vec{B} \xleftarrow{\vec{\mathcal{D}}_{\mathbb{G}_2}} \mathbb{G}_2$

7 : **case** SVPV **:** // A SECRET, B PUBLIC

8 : $\mathsf{Dist}^{\lambda}_{\infty}(\mathbb{G}_1) \supset \vec{\mathcal{D}} \leftarrow \mathcal{A}(\mathtt{st})$; $\vec{A} \xleftarrow{\vec{\mathcal{D}}} \mathbb{G}_1$; $\vec{B} \leftarrow \mathcal{A}(\mathtt{st})$

9 : **case** PVSV **:** // A PUBLIC, B SECRET

10 : $\mathsf{Dist}^{\lambda}_{\infty}(\mathbb{G}_2) \supset \vec{\mathcal{D}} \leftarrow \mathcal{A}(\mathtt{st})$; $\vec{B} \xleftarrow{\vec{\mathcal{D}}} \mathbb{G}_2$; $\vec{A} \leftarrow \mathcal{A}(\mathtt{st})$

11 : $\mathsf{ctr} \leftarrow \mathsf{ctr} + \mathtt{len}\left(\vec{A}_i\right)$; **if** $\mathsf{ctr} \geq L$ **:** **break**

12 : $(\mathsf{pub}, \mathsf{sec}) \leftarrow \mathsf{Setup}\left(\mathtt{sk}, \vec{A}_i, \vec{B}_i\right)$; $\mathtt{st} \leftarrow \mathcal{A}(\mathsf{pub}, \mathtt{st})$

13 : **if** $\mathsf{type} \in \{\mathsf{SVSV}, \mathsf{SVPV}\}$ **:**

14 : $A^* \leftarrow \mathcal{A}(\mathtt{st})$; **if** $A^* \in \vec{A}$**:** **return** 1

15 : **if** $\mathsf{type} \in \{\mathsf{SVSV}, \mathsf{PVSV}\}$ **:**

16 : $B^* \leftarrow \mathcal{A}(\mathtt{st})$; **if** $B^* \in \vec{B}$**:** **return** 1

17 : **return** 0

For each secret input:

1. $\mathcal{A}$ chooses a λ-min entropy distribution $\mathcal{D}$
2. $\mathcal{C}$ samples input from $\mathcal{D}$ and computes pub
3. From pub, $\mathcal{A}$ extracts one of the inputs.

Current Verifiability Experiment:

Chosen-Input Verifiability

CI-Verifiability: $\mathbf{Exp}^{\mathsf{seq,ver}}_{\mathbf{\Pi},\mathcal{A}}(\lambda, \mathsf{aux.sec}, L)$

1 : $\mathsf{pp} \leftarrow \mathsf{GlobalSetup}(\lambda)$; $\mathtt{sk} \leftarrow \mathsf{OneTimeSetup}\,(\mathtt{aux.sec})$

2 : $\mathtt{st} \leftarrow (\lambda, \mathtt{aux.sec}, \mathsf{pp})$; $\mathsf{ctr} \leftarrow 0$

3 : **while** true **:** // DELEGATION ROUNDS

4 : $\left(\vec{A}, \vec{B}\right) \leftarrow \mathcal{A}(\mathtt{st})$

5 : $\mathsf{ctr} \leftarrow \mathsf{ctr} + \mathtt{len}(\vec{A})$; **if** $\mathsf{ctr} \geq L$ **:** **break**

6 : $(\mathtt{pub}, \mathtt{sec}) \leftarrow \mathsf{Setup}(\mathtt{sk}, \vec{A}, \vec{B})$;

7 : $(\mathtt{out}^*, \mathtt{st}) \leftarrow \mathcal{A}\,(\mathtt{pub}, \mathtt{st})$

8 : $\mathtt{result}^* \leftarrow \mathsf{Verify}(\mathtt{sk}, \mathtt{sec}, \mathtt{out}^*)$

9 : **if** $\mathtt{result}^* \notin \{\perp, e(\vec{A}, \vec{B})\}$ **:** **return** 1

10 : **if** $\mathtt{result}^* = \perp$ **:** **return** 0

11 : **return** 0

CI-Verifiability: $\mathbf{Exp}^{\mathsf{seq,ver}}_{\mathbf{\Pi},\mathcal{A}}(\lambda, \mathsf{aux.sec}, L)$

1 : $\mathsf{pp} \leftarrow \mathsf{GlobalSetup}(\lambda)$; $\mathtt{sk} \leftarrow \mathsf{OneTimeSetup}\,(\mathtt{aux.sec})$

2 : $\mathtt{st} \leftarrow (\lambda, \mathtt{aux.sec}, \mathsf{pp})$; $\mathsf{ctr} \leftarrow 0$

3 : **while** true **:** // DELEGATION ROUNDS

4 : $\left(\vec{A}, \vec{B}\right) \leftarrow \mathcal{A}(\mathtt{st})$

5 : $\mathsf{ctr} \leftarrow \mathsf{ctr} + \mathsf{len}(\vec{A})$; **if** $\mathsf{ctr} \geq L$ **:** **break**

6 : $(\mathtt{pub}, \mathtt{sec}) \leftarrow \mathsf{Setup}(\mathtt{sk}, \vec{A}, \vec{B})$;

7 : $(\mathtt{out}^*, \mathtt{st}) \leftarrow \mathcal{A}\,(\mathtt{pub}, \mathtt{st})$

8 : $\mathtt{result}^* \leftarrow \mathsf{Verify}(\mathtt{sk}, \mathtt{sec}, \mathtt{out}^*)$

9 : **if** $\mathtt{result}^* \notin \{\perp, e(\vec{A}, \vec{B})\}$ **:** **return** 1

10 : **if** $\mathtt{result}^* = \perp$ **:** **return** 0

11 : **return** 0

1. $\mathcal{A}$ chooses the pairing inputs for all privacy types.

CI-Verifiability: $\mathbf{Exp}^{\mathsf{seq,ver}}_{\mathbf{\Pi},\mathcal{A}}(\lambda, \mathsf{aux.sec}, L)$

1 : $\mathsf{pp} \leftarrow \mathsf{GlobalSetup}(\lambda)$; $\mathsf{sk} \leftarrow \mathsf{OneTimeSetup}(\mathsf{aux.sec})$

2 : $\mathsf{st} \leftarrow (\lambda, \mathsf{aux.sec}, \mathsf{pp})$; $\mathsf{ctr} \leftarrow 0$

3 : **while** true **:** // DELEGATION ROUNDS

4 : $\left(\vec{A}, \vec{B}\right) \leftarrow \mathcal{A}(\mathsf{st})$

5 : $\mathsf{ctr} \leftarrow \mathsf{ctr} + \mathsf{len}(\vec{A})$; **if** $\mathsf{ctr} \geq L$ **:** **break**

6 : $(\mathsf{pub}, \mathsf{sec}) \leftarrow \mathsf{Setup}(\mathsf{sk}, \vec{A}, \vec{B})$;

7 : $(\mathtt{out}^*, \mathsf{st}) \leftarrow \mathcal{A}(\mathsf{pub}, \mathsf{st})$

8 : $\mathtt{result}^* \leftarrow \mathsf{Verify}(\mathsf{sk}, \mathsf{sec}, \mathtt{out}^*)$

9 : **if** $\mathtt{result}^* \notin \{\perp, e(\vec{A}, \vec{B})\}$ **:** **return** 1

10 : **if** $\mathtt{result}^* = \perp$ **:** **return** 0

11 : **return** 0

1. $\mathcal{A}$ chooses the pairing inputs for all privacy types.

2. Given pub, $\mathcal{A}$ attempts a forgery.

Proposed Variant:

Sampled-Input Verifiability

SI-Verifiability: $\mathbf{Exp}^{\mathsf{seq,ver}}_{\mathbf{\Pi},\mathcal{A}}(\lambda, \mathsf{aux.sec}, L, \mathsf{type})$

1 : $\mathsf{pp} \leftarrow \mathsf{GlobalSetup}(\lambda)$; $\mathsf{sk} \leftarrow \mathsf{OneTimeSetup}(\mathsf{aux.sec})$

2 : $\mathsf{st} \leftarrow (\lambda, \mathsf{aux.sec}, \mathsf{pp})$; $\mathsf{ctr} \leftarrow 0$

3 : **while** true **:** // DELEGATION ROUNDS

4 : **switch** (type)

5 : **case** SVSV **:** // A SECRET, B SECRET

6 : $\mathsf{Dist}^{\lambda}_{\infty}(\mathbb{G}_1 \times \mathbb{G}_2)\big|_{\mathsf{marg}} \ni \vec{\mathcal{D}} \leftarrow \mathcal{A}(\mathsf{st})$; $\vec{A} \overset{\vec{\mathcal{D}}_{\mathbb{G}_1}}{\leftarrow} \mathbb{G}_1$; $\vec{B} \overset{\vec{\mathcal{D}}_{\mathbb{G}_2}}{\leftarrow} \mathbb{G}_2$

7 : **case** SVPV **:** $\mathsf{Dist}^{\lambda}_{\infty}(\mathbb{G}_1) \ni \vec{\mathcal{D}} \leftarrow \mathcal{A}(\mathsf{st})$; $\vec{A} \overset{\vec{\mathcal{D}}}{\leftarrow} \mathbb{G}_1$; $\vec{B} \leftarrow \mathcal{A}(\mathsf{st})$

8 : **case** PVSV **:** $\mathsf{Dist}^{\lambda}_{\infty}(\mathbb{G}_2) \ni \vec{\mathcal{D}} \leftarrow \mathcal{A}(\mathsf{st})$; $\vec{B} \overset{\vec{\mathcal{D}}}{\leftarrow} \mathbb{G}_2$; $\vec{A} \leftarrow \mathcal{A}(\mathsf{st})$

9 : **case** PVPV **:** $\left(\vec{A}, \vec{B}\right) \leftarrow \mathcal{A}(\mathsf{st})$

10 : $\mathsf{ctr} \leftarrow \mathsf{ctr} + \mathtt{len}(\vec{A})$; **if** $\mathsf{ctr} \geq L$ **:** **break**

11 : $(\mathtt{pub}, \mathtt{sec}) \leftarrow \mathsf{Setup}(\mathtt{sk}, \vec{A}, \vec{B})$; $(\mathtt{out}^*, \mathtt{st}) \leftarrow \mathcal{A}(\mathtt{pub}, \mathtt{st})$

12 : $\mathtt{result}^* \leftarrow \mathsf{Verify}(\mathtt{sk}, \mathtt{sec}, \mathtt{out}^*)$

13 : **if** $\mathtt{result}^* \notin \{\bot, e(\vec{A}, \vec{B})\}$ **:** **return** 1

14 : **if** $\mathtt{result}^* = \bot$ **:** **return** 0

15 : **return** 0

SI-Verifiability: $\mathbf{Exp}^{\mathsf{seq,ver}}_{\mathbf{\Pi},\mathcal{A}}(\lambda, \mathsf{aux.sec}, L, \mathsf{type})$

1 : $\mathsf{pp} \leftarrow \mathsf{GlobalSetup}(\lambda)$; $\mathsf{sk} \leftarrow \mathsf{OneTimeSetup}(\mathsf{aux.sec})$

2 : $\mathsf{st} \leftarrow (\lambda, \mathsf{aux.sec}, \mathsf{pp})$; $\mathsf{ctr} \leftarrow 0$

3 : **while** true **:** // DELEGATION ROUNDS

4 : **switch** (type)

5 : **case** SVSV **:** // A SECRET, B SECRET

6 : $\mathsf{Dist}^{\lambda}_{\infty}(\mathbb{G}_1 \times \mathbb{G}_2)\big|_{\mathsf{marg}} \supset \vec{\mathcal{D}} \leftarrow \mathcal{A}(\mathtt{st})$; $\vec{A} \overset{\vec{\mathcal{D}}_{\mathbb{G}_1}}{\leftarrow} \mathbb{G}_1$; $\vec{B} \overset{\vec{\mathcal{D}}_{\mathbb{G}_2}}{\leftarrow} \mathbb{G}_2$

7 : **case** SVPV **:** $\mathsf{Dist}^{\lambda}_{\infty}(\mathbb{G}_1) \supset \vec{\mathcal{D}} \leftarrow \mathcal{A}(\mathtt{st})$; $\vec{A} \overset{\vec{\mathcal{D}}}{\leftarrow} \mathbb{G}_1$; $\vec{B} \leftarrow \mathcal{A}(\mathtt{st})$

8 : **case** PVSV **:** $\mathsf{Dist}^{\lambda}_{\infty}(\mathbb{G}_2) \supset \vec{\mathcal{D}} \leftarrow \mathcal{A}(\mathtt{st})$; $\vec{B} \overset{\vec{\mathcal{D}}}{\leftarrow} \mathbb{G}_2$; $\vec{A} \leftarrow \mathcal{A}(\mathtt{st})$

9 : **case** PVPV **:** $\left(\vec{A}, \vec{B}\right) \leftarrow \mathcal{A}(\mathtt{st})$

10 : $\mathsf{ctr} \leftarrow \mathsf{ctr} + \mathtt{len}(\vec{A})$; **if** $\mathsf{ctr} \geq L$ **:** **break**

11 : $(\mathtt{pub}, \mathtt{sec}) \leftarrow \mathsf{Setup}(\mathtt{sk}, \vec{A}, \vec{B})$; $(\mathtt{out}^*, \mathtt{st}) \leftarrow \mathcal{A}(\mathtt{pub}, \mathtt{st})$

12 : $\mathtt{result}^* \leftarrow \mathsf{Verify}(\mathtt{sk}, \mathtt{sec}, \mathtt{out}^*)$

13 : **if** $\mathtt{result}^* \notin \{\bot, e(\vec{A}, \vec{B})\}$ **:** **return** 1

14 : **if** $\mathtt{result}^* = \bot$ **:** **return** 0

15 : **return** 0

For each secret input:

1. $\mathcal{A}$ chooses a λ-min entropy distribution $\mathcal{D}$

SI-Verifiability: $\mathbf{Exp}^{\mathsf{seq,ver}}_{\mathbf{\Pi},\mathcal{A}}(\lambda, \mathsf{aux.sec}, L, \mathsf{type})$

1 : $\mathsf{pp} \leftarrow \mathsf{GlobalSetup}(\lambda)$; $\mathsf{sk} \leftarrow \mathsf{OneTimeSetup}(\mathsf{aux.sec})$

2 : $\mathsf{st} \leftarrow (\lambda, \mathsf{aux.sec}, \mathsf{pp})$; $\mathsf{ctr} \leftarrow 0$

3 : **while** true **:** // DELEGATION ROUNDS

4 : **switch** (type)

5 : **case** SVSV **:** // A SECRET, B SECRET

6 : $\mathsf{Dist}^{\lambda}_{\infty}(\mathbb{G}_1 \times \mathbb{G}_2)\big|_{\mathsf{marg}} \supset \vec{\mathcal{D}} \leftarrow \mathcal{A}(\mathtt{st})$; $\vec{A} \overset{\vec{\mathcal{D}}_{\mathbb{G}_1}}{\leftarrow} \mathbb{G}_1$; $\vec{B} \overset{\vec{\mathcal{D}}_{\mathbb{G}_2}}{\leftarrow} \mathbb{G}_2$

7 : **case** SVPV **:** $\mathsf{Dist}^{\lambda}_{\infty}(\mathbb{G}_1) \supset \vec{\mathcal{D}} \leftarrow \mathcal{A}(\mathtt{st})$; $\vec{A} \overset{\vec{\mathcal{D}}}{\leftarrow} \mathbb{G}_1$; $\vec{B} \leftarrow \mathcal{A}(\mathtt{st})$

8 : **case** PVSV **:** $\mathsf{Dist}^{\lambda}_{\infty}(\mathbb{G}_2) \supset \vec{\mathcal{D}} \leftarrow \mathcal{A}(\mathtt{st})$; $\vec{B} \overset{\vec{\mathcal{D}}}{\leftarrow} \mathbb{G}_2$; $\vec{A} \leftarrow \mathcal{A}(\mathtt{st})$

9 : **case** PVPV **:** $\left(\vec{A}, \vec{B}\right) \leftarrow \mathcal{A}(\mathtt{st})$

10 : $\mathsf{ctr} \leftarrow \mathsf{ctr} + \mathtt{len}(\vec{A})$; **if** $\mathsf{ctr} \geq L$ **:** **break**

11 : $(\mathtt{pub}, \mathtt{sec}) \leftarrow \mathsf{Setup}(\mathtt{sk}, \vec{A}, \vec{B})$; $(\mathtt{out}^*, \mathtt{st}) \leftarrow \mathcal{A}(\mathtt{pub}, \mathtt{st})$

12 : $\mathtt{result}^* \leftarrow \mathsf{Verify}(\mathtt{sk}, \mathtt{sec}, \mathtt{out}^*)$

13 : **if** $\mathtt{result}^* \notin \{\bot, e(\vec{A}, \vec{B})\}$ **:** **return** 1

14 : **if** $\mathtt{result}^* = \bot$ **:** **return** 0

15 : **return** 0

For each secret input:

1. $\mathcal{A}$ chooses a λ-min entropy distribution $\mathcal{D}$
2. $\mathcal{C}$ samples input from $\mathcal{D}$ and computes pub

SI-Verifiability: $\mathbf{Exp}^{\mathsf{seq,ver}}_{\mathbf{\Pi},\mathcal{A}}(\lambda, \mathrm{aux.sec}, L, \mathrm{type})$

1 : $\mathtt{pp} \leftarrow \mathsf{GlobalSetup}(\lambda)$; $\mathtt{sk} \leftarrow \mathsf{OneTimeSetup}(\mathrm{aux.sec})$

2 : $\mathtt{st} \leftarrow (\lambda, \mathrm{aux.sec}, \mathtt{pp})$; $\mathtt{ctr} \leftarrow 0$

3 : **while** true **:** // DELEGATION ROUNDS

4 : **switch** (type)

5 : **case** SVSV **:** // A SECRET, B SECRET

6 : $\mathsf{Dist}^{\lambda}_{\infty}(\mathbb{G}_1 \times \mathbb{G}_2)\big|_{\mathsf{marg}} \ni \vec{\mathcal{D}} \leftarrow \mathcal{A}(\mathtt{st})$; $\vec{A} \overset{\vec{\mathcal{D}}_{\mathbb{G}_1}}{\leftarrow} \mathbb{G}_1$; $\vec{B} \overset{\vec{\mathcal{D}}_{\mathbb{G}_2}}{\leftarrow} \mathbb{G}_2$

7 : **case** SVPV **:** $\mathsf{Dist}^{\lambda}_{\infty}(\mathbb{G}_1) \ni \vec{\mathcal{D}} \leftarrow \mathcal{A}(\mathtt{st})$; $\vec{A} \overset{\vec{\mathcal{D}}}{\leftarrow} \mathbb{G}_1$; $\vec{B} \leftarrow \mathcal{A}(\mathtt{st})$

8 : **case** PVSV **:** $\mathsf{Dist}^{\lambda}_{\infty}(\mathbb{G}_2) \ni \vec{\mathcal{D}} \leftarrow \mathcal{A}(\mathtt{st})$; $\vec{B} \overset{\vec{\mathcal{D}}}{\leftarrow} \mathbb{G}_2$; $\vec{A} \leftarrow \mathcal{A}(\mathtt{st})$

9 : **case** PVPV **:** $\left(\vec{A}, \vec{B}\right) \leftarrow \mathcal{A}(\mathtt{st})$

10 : $\mathtt{ctr} \leftarrow \mathtt{ctr} + \mathtt{len}(\vec{A})$; **if** $\mathtt{ctr} \geq L$ **:** **break**

11 : $(\mathtt{pub}, \mathtt{sec}) \leftarrow \mathsf{Setup}(\mathtt{sk}, \vec{A}, \vec{B})$; $(\mathtt{out}^*, \mathtt{st}) \leftarrow \mathcal{A}(\mathtt{pub}, \mathtt{st})$

12 : $\mathtt{result}^* \leftarrow \mathsf{Verify}(\mathtt{sk}, \mathtt{sec}, \mathtt{out}^*)$

13 : **if** $\mathtt{result}^* \notin \{\bot, e(\vec{A}, \vec{B})\}$ **:** **return** 1

14 : **if** $\mathtt{result}^* = \bot$ **:** **return** 0

15 : **return** 0

For each secret input:

1. $\mathcal{A}$ chooses a λ-min entropy distribution $\mathcal{D}$
2. $\mathcal{C}$ samples input from $\mathcal{D}$ and computes pub
3. From pub, $\mathcal{A}$ attempts to forge a pairing.

Ingredients for Efficient Private Delegation

Ingredients for Efficient Private Delegation

λ

Computational hardness

Use computational hardness assumption: $\mathcal{A}$ bounded in λ

Ingredients for Efficient Private Delegation

Computational hardness

Use computational hardness assumption: $\mathcal{A}$ bounded in λ

Secret scalar sampling

Sample secret scalars from the set $\{1, ..., 2^{\lambda}\} \subset Z_q^*$ (recall that $\log_2(q) = 2 \cdot \lambda$)

Ingredients for Efficient Private Delegation

λ

Computational hardness

Use computational hardness assumption: $\mathcal{A}$ bounded in λ

$

Secret scalar sampling

Sample secret scalars from the set $\{1, ..., 2^{\lambda}\} \subset Z_q^*$ (recall that $\log_2(q) = 2 \cdot \lambda$)

1

Single degree of freedom

Design the construction so that only extra secret can parametrize the rest.

Ingredients for Efficient Private Delegation

λ

Computational hardness

Use computational hardness assumption: $\mathcal{A}$ bounded in λ

$\$$

Secret scalar sampling

Sample secret scalars from the set $\{1, ..., 2^{\lambda}\} \subset Z_q^*$ (recall that $\log_2(q) = 2 \cdot \lambda$)

1

Single degree of freedom

Design the construction so that only extra secret can parametrize the rest.

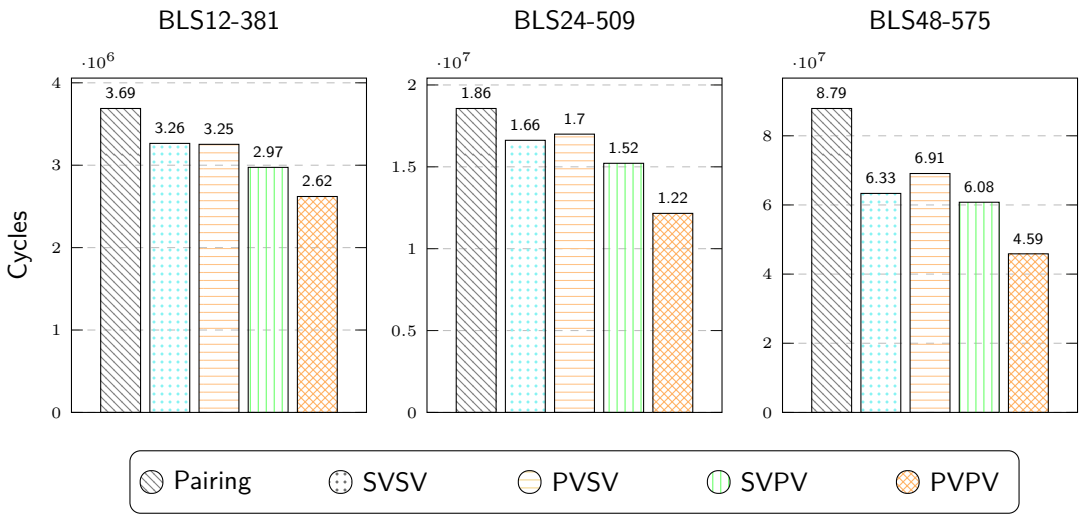
π

AmorE proof technique

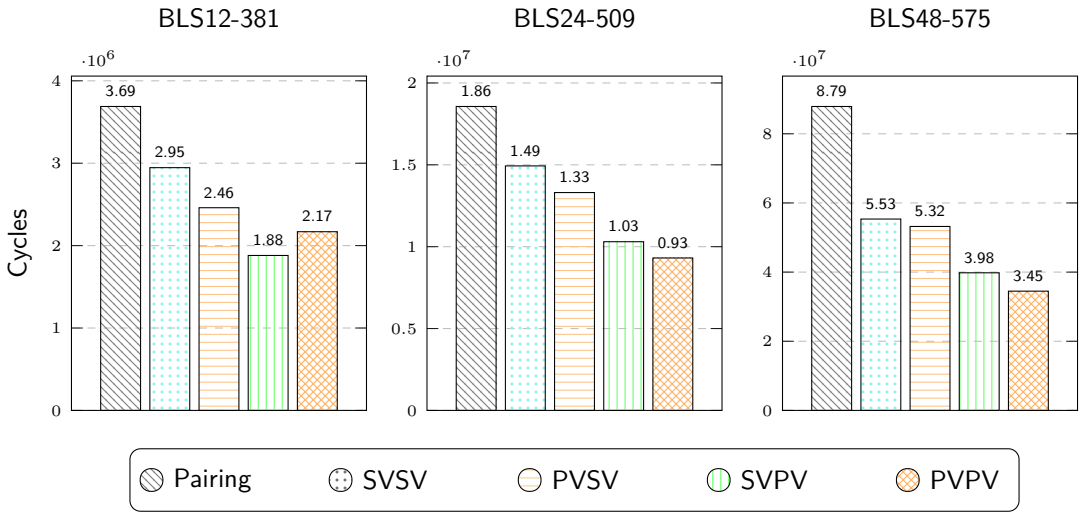
If $\mathcal{A}$ can only succeed in the MITM attack with negl(λ) probability.

Experimental Results

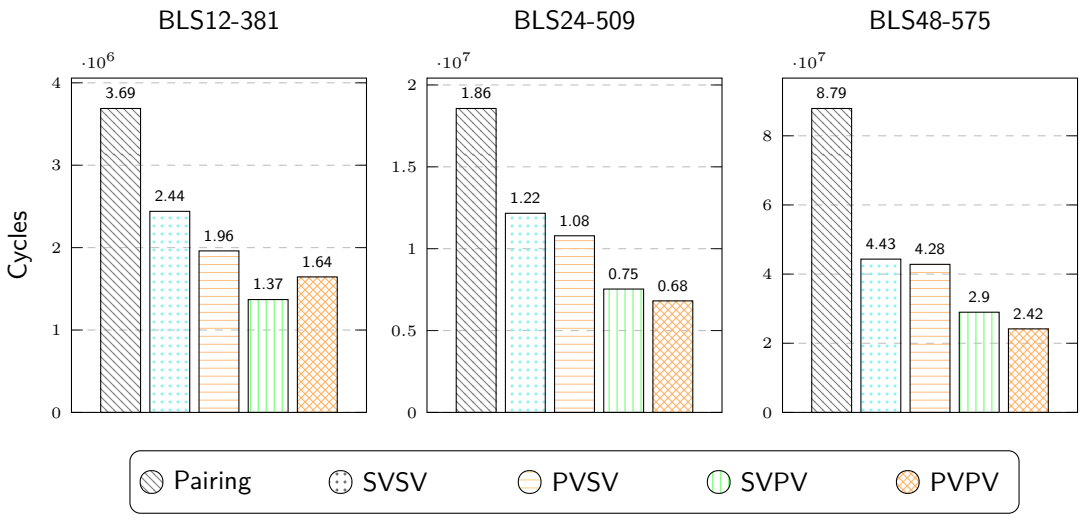
Experimental Results: Single Delegation $(\mathsf{N}, \mathsf{M}) = (25, 1)$



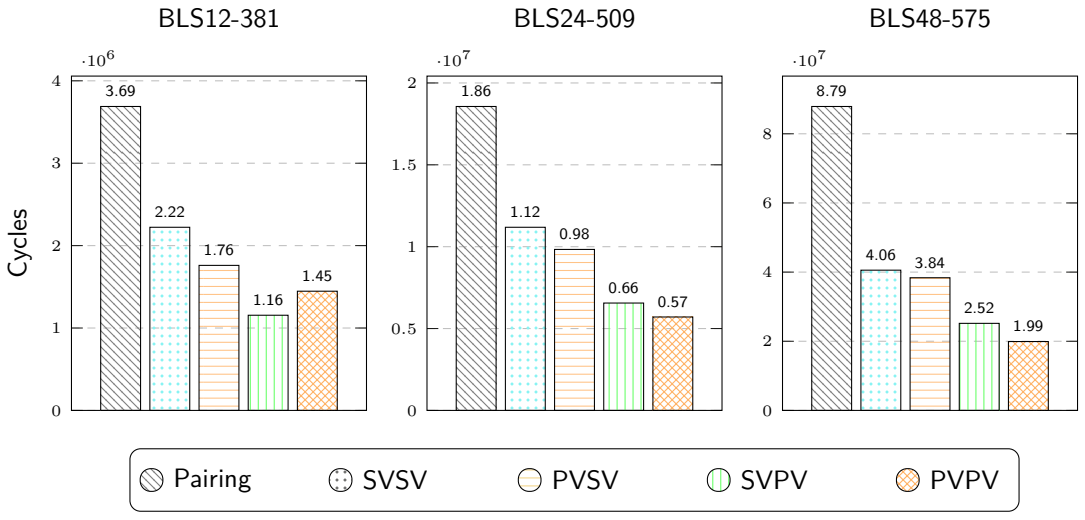
Experimental Results: Batch Delegation $(\mathsf{N}, \mathsf{M}) = (25, 3)$



Experimental Results: Batch Delegation $(\mathsf{N}, \mathsf{M}) = (25, 10)$



Experimental Results: Batch Delegation $(\mathsf{N}, \mathsf{M}) = (25, 100)$



How to SAT solve a Hash Function

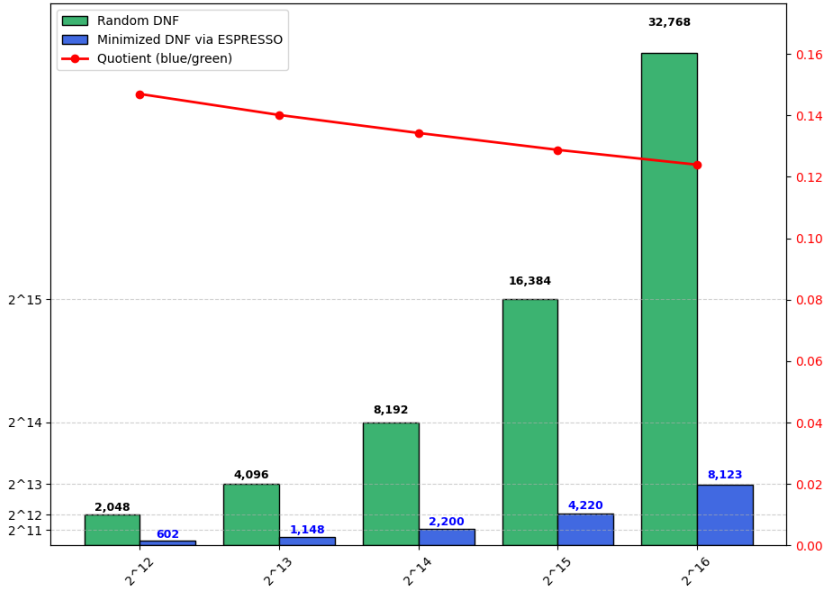
Adrián P. Keilty[1]

[1] Chalmers University of Technology, Sweden

(WIP)

Algebraic cryptanalysis on hash functions

- Algebraic cryptanalysis seeks to represent the hash function as a system of boolean equations.
- But boolean expressions evolving from a symbolic plaintext blow up very quickly... Or do they?
- The average minimal representation size of a DNF grows sub-exponentially with the table size ("The Shortest Disjunctive Normal Form of a Random Boolean Function", Nicholas Pippenger)


Boolean Minimization: Random expression (green), Minimized (blue), and Quotient (red line)
Random DNF
Minimized DNF via ESPRESSO
Quotient (blue/green)
2,048
602
4,096
1,148
8,192
2,200
16,384
4,220
32,768
8,123
2^15
2^14
2^13
2^12
2^11
0.16
0.14
0.12
0.10
0.08
0.06
0.04
0.02
0.00
2^12
2^13
2^14
2^15
2^16

Algebraic cryptanalysis on hash functions

- But... Full boolean minimization is NP-hard. Can we do close to minimal efficiently?
- *ESPRESSO* developed at Berkeley is the state-of-the-art heuristic boolean minimization tool. It provides a near-minimal representation of a boolean expression expressed in Disjuntive Normal Form (DNF) in close to linear time.
- Good, but SHA3-256 uses XORs and other transformations that require complementation. Obtaining the complement of a DNF in DNF is NP-hard too...
- Hmmm, why not process in parallel a negated version of SHA3-256 so that both processes can fetch each other's complements?
- Good. Any other things to worry about? Well, suppose we have successfully gone through all of SHA3-256's rounds and we can solve against a given hash. SAT solvers are optimized for CNF, not DNF. And DNF to CNF conversion is... NP-hard!

Algebraic cryptanalysis on hash functions

- Still not a problem. We have every complement in DNF form as well remember? So we can convert to CNF in linear time by simply applying De Morgan's complementation law once to the complement of each respective DNF.
- Great, we can now
  - Process an input through the hash function solely by performing ANDs and ORs of DNFs.
  - Minimizing expressions via ESPRESSO.
  - SAT-solve against against any hash (preimage attack), or
  - Process two distinct inputs and then SAT-solve against eachother (collision attack).
- ...ok yes, SAT solving is NP-complete. But a tremendous amount of work has been done in that field and in practice even huge instances can be solved efficiently. Why not put SHA3-256 to the test?

Relevant insight: Do the minimization process once and then SAT solve against unlimited hashes.

Algebraic cryptanalysis on hash functions

Results so far:

- A codebase built from *ESPRESSO* that allows to set specific bits of a plaintext as symbolic variables, keeps track of the symbolic expressions through SHA3-256's 24 rounds, minimizes them with *ESPRESSO* at each step, and finally saves 256 DNFs.
- The symbolic process currently uses 8 threads: 4 that handle SHA and the other 4 that handle the negated version $\neg SHA$.
- A 'solving' flag (`-s`): given a number of inputs and a target hash builds the CNF from the stored DNFs and feeds it to a SAT solver.
- Z3 obtains a partial preimage faster than brute force.
- For now a maximum of 16 bits have been set as symbolic (on an intel i5, 16 GB RAM).

Algebraic cryptanalysis on hash functions

TODOs:

- Ideally, parallelization should work as follows:
  - 2 main processes: *SHA* and $\neg SHA$. Each composed of
  - 320 threads for θ
  - 1600 threads for ρ, π and χ
  - 64 threads for ι

  Max number of simultaneous active threads: 3200. Each of the 1600 waiting for the other 1600 for free DNF complementation.
- Additional parallelization:
  - "ESPRESSO-GPU: Blazingly Fast Two-Level Logic Minimization" (ieeexplore.ieee.org/document/9473961)
  - The AND operation for DNFs.
- Allow for arbitrary expression embedding into the input instead of just variable hardcoding. The hash-specific cryptanalysis starts here: setting clever expression embeddings may lead to smaller expressions at later rounds.

Questions:

- Currently, the AND/OR operations and the ESPRESSO algorithm are performed separately. Can these be merged meaningfully so that all is done at once?
- Empirically, n variables lead to a n-bit partial preimage. How many symbolic bits and function compression rounds can we feasibly reach for SHA3-256?
- How will the complexity of SAT solving for SHA look like?

That's it. Thank you!