Efficient Higher-Order Masking Schemes: Leveraging Amortization and Pre-computation

Weijia Wang

Shandong University, China

December 3, 2023

Table of Contents

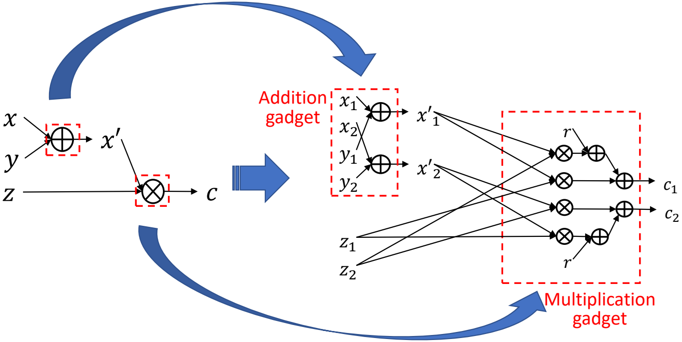
Table of Contents

Masking, two ingredients:

- Randomize the secret
  - Secret variable $x \xrightarrow{\text{rand}}$ shares $\hat{\mathbf{x}}[1], \ldots, \hat{\mathbf{x}}[d+1]$. Any d shares *are independent of* x
    - Boolean masking: $x = \hat{\mathbf{x}}[1] \oplus \ldots \oplus \hat{\mathbf{x}}[d+1]$
- Private computations.
  - Any d intermediates *are independent of* the input secrets: d-privacy, d-probing security

Masking, two ingredients:

- Randomize the secret
  - Secret variable $x \xrightarrow{\text{rand}}$ shares $\hat{\mathbf{x}}[1], \ldots, \hat{\mathbf{x}}[d+1]$. Any d shares *are independent of* x
    - Boolean masking: $x = \hat{\mathbf{x}}[1] \oplus \ldots \oplus \hat{\mathbf{x}}[d+1]$
- Private computations.
  - Any d intermediates *are independent of* the input secrets: d-privacy, d-probing security



Masking Provides Provable Side-channel Security

- Recall the security of RSA:
  - The security of RSA relies on the practical difficulty of factoring the product of two large prime numbers.
  - If there exists a machine can break RSA efficiently, then this machine can factor the product of two large prime numbers efficiently as well.
- Masking:
  - The security of masking relies on some physical assumptions that can be realized by engineering.
    1. noisy leakage;
    2. independent leakage.
  - The security increases exponentially with the number of shares.
  - If there exists a attack can break masking efficiently, then at least one of the assumptions does not hold.

Masking Provides Provable Side-channel Security

- Recall the security of RSA:
  - The security of RSA relies on the practical difficulty of factoring the product of two large prime numbers.
  - If there exists a machine can break RSA efficiently, then this machine can factor the product of two large prime numbers efficiently as well.
- Masking:
  - The security of masking relies on some physical assumptions that can be realized by engineering.
    1. noisy leakage;
    2. independent leakage.
  - The security increases exponentially with the number of shares.
  - If there exists a attack can break masking efficiently, then at least one of the assumptions does not hold.

Masking Provides Provable Side-channel Security

- Recall the security of RSA:
  - The security of RSA relies on the practical difficulty of factoring the product of two large prime numbers.
  - If there exists a machine can break RSA efficiently, then this machine can factor the product of two large prime numbers efficiently as well.
- Masking:
  - The security of masking relies on some physical assumptions that can be realized by engineering.
    1. noisy leakage;
    2. independent leakage.
  - The security increases exponentially with the number of shares.
  - If there exists a attack can break masking efficiently, then at least one of the assumptions does not hold.

Masking Provides Provable Side-channel Security

- Recall the security of RSA:
  - The security of RSA relies on the practical difficulty of factoring the product of two large prime numbers.
  - If there exists a machine can break RSA efficiently, then this machine can factor the product of two large prime numbers efficiently as well.
- Masking:
  - The security of masking relies on some physical assumptions that can be realized by engineering.
    1. noisy leakage;
    2. independent leakage.
  - The security increases exponentially with the number of shares.
  - If there exists a attack can break masking efficiently, then at least one of the assumptions does not hold.

Masking Provides Provable Side-channel Security

- Recall the security of RSA:
  - The security of RSA relies on the practical difficulty of factoring the product of two large prime numbers.
  - If there exists a machine can break RSA efficiently, then this machine can factor the product of two large prime numbers efficiently as well.
- Masking:
  - The security of masking relies on some physical assumptions that can be realized by engineering.
    1. noisy leakage;
    2. independent leakage.
  - The security increases exponentially with the number of shares.
  - If there exists a attack can break masking efficiently, then at least one of the assumptions does not hold.

Masking Provides Provable Side-channel Security

- Recall the security of RSA:
  - The security of RSA relies on the practical difficulty of factoring the product of two large prime numbers.
  - If there exists a machine can break RSA efficiently, then this machine can factor the product of two large prime numbers efficiently as well.
- Masking:
  - The security of masking relies on some physical assumptions that can be realized by engineering.
    1. noisy leakage;
    2. independent leakage.
  - The security increases exponentially with the number of shares.
  - If there exists a attack can break masking efficiently, then at least one of the assumptions does not hold.

Masking Provides Provable Side-channel Security

- Recall the security of RSA:
  - The security of RSA relies on the practical difficulty of factoring the product of two large prime numbers.
  - If there exists a machine can break RSA efficiently, then this machine can factor the product of two large prime numbers efficiently as well.
- Masking:
  - The security of masking relies on some physical assumptions that can be realized by engineering.
    1. noisy leakage;
    2. independent leakage.
  - The security increases exponentially with the number of shares.
  - If there exists a attack can break masking efficiently, then at least one of the assumptions does not hold.

Example: the ISW Multiplicaition with 3 Shares

- Proposed by Yuval **I**shai, Amit **S**ahai and David **W**agner at *CRYPTO '03*.
- Input: $\hat{\mathbf{x}}[1], \hat{\mathbf{x}}[2], \hat{\mathbf{x}}[3]$ and $\hat{\mathbf{y}}[1], \hat{\mathbf{y}}[2], \hat{\mathbf{y}}[3]$, Output: $\hat{\mathbf{z}}[1], \hat{\mathbf{z}}[2], \hat{\mathbf{z}}[3]$
- It requires $\frac{\ell d(d+1)}{2}$ random bits and runs in $\mathcal{O}(\ell d^2)$ to protect a circuit of size $\mathcal{O}(\ell)$.

Example: the ISW Multiplicaition with 3 Shares

- Proposed by Yuval **I**shai, Amit **S**ahai and David **W**agner at *CRYPTO '03*.
- Input: $\hat{\mathbf{x}}[1], \hat{\mathbf{x}}[2], \hat{\mathbf{x}}[3]$ and $\hat{\mathbf{y}}[1], \hat{\mathbf{y}}[2], \hat{\mathbf{y}}[3]$, Output: $\hat{\mathbf{z}}[1], \hat{\mathbf{z}}[2], \hat{\mathbf{z}}[3]$

$\hat{\mathbf{x}}[1]\hat{\mathbf{y}}[1]$	$\hat{\mathbf{x}}[1]\hat{\mathbf{y}}[2]$	$\hat{\mathbf{x}}[1]\hat{\mathbf{y}}[3]$
$\hat{\mathbf{x}}[2]\hat{\mathbf{y}}[1]$	$\hat{\mathbf{x}}[2]\hat{\mathbf{y}}[2]$	$\hat{\mathbf{x}}[2]\hat{\mathbf{y}}[3]$
$\hat{\mathbf{x}}[3]\hat{\mathbf{y}}[1]$	$\hat{\mathbf{x}}[3]\hat{\mathbf{y}}[2]$	$\hat{\mathbf{x}}[3]\hat{\mathbf{y}}[3]$

- It requires $\frac{\ell d(d+1)}{2}$ random bits and runs in $\mathcal{O}(\ell d^2)$ to protect a circuit of size $\mathcal{O}(\ell)$.

Example: the ISW Multiplicaition with 3 Shares

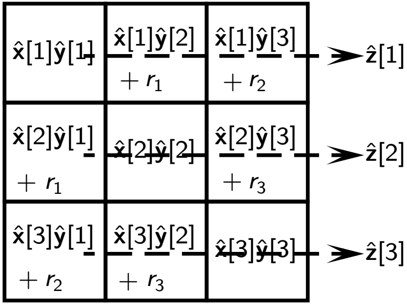
- Proposed by Yuval **I**shai, Amit **S**ahai and David **W**agner at *CRYPTO '03*.
- Input: $\hat{\mathbf{x}}[1], \hat{\mathbf{x}}[2], \hat{\mathbf{x}}[3]$ and $\hat{\mathbf{y}}[1], \hat{\mathbf{y}}[2], \hat{\mathbf{y}}[3]$, Output: $\hat{\mathbf{z}}[1], \hat{\mathbf{z}}[2], \hat{\mathbf{z}}[3]$

$\hat{\mathbf{x}}[1]\hat{\mathbf{y}}[1]$	$\hat{\mathbf{x}}[1]\hat{\mathbf{y}}[2] + r_1$	$\hat{\mathbf{x}}[1]\hat{\mathbf{y}}[3] + r_2$
$\hat{\mathbf{x}}[2]\hat{\mathbf{y}}[1] + r_1$	$\hat{\mathbf{x}}[2]\hat{\mathbf{y}}[2]$	$\hat{\mathbf{x}}[2]\hat{\mathbf{y}}[3] + r_3$
$\hat{\mathbf{x}}[3]\hat{\mathbf{y}}[1] + r_2$	$\hat{\mathbf{x}}[3]\hat{\mathbf{y}}[2] + r_3$	$\hat{\mathbf{x}}[3]\hat{\mathbf{y}}[3]$

- It requires $\frac{\ell d(d+1)}{2}$ random bits and runs in $\mathcal{O}(\ell d^2)$ to protect a circuit of size $\mathcal{O}(\ell)$.

Example: the ISW Multiplicaition with 3 Shares

- Proposed by Yuval **I**shai, Amit **S**ahai and David **W**agner at *CRYPTO '03*.
- Input: $\hat{\mathbf{x}}[1], \hat{\mathbf{x}}[2], \hat{\mathbf{x}}[3]$ and $\hat{\mathbf{y}}[1], \hat{\mathbf{y}}[2], \hat{\mathbf{y}}[3]$, Output: $\hat{\mathbf{z}}[1], \hat{\mathbf{z}}[2], \hat{\mathbf{z}}[3]$



- It requires $\frac{\ell d(d+1)}{2}$ random bits and runs in $\mathcal{O}(\ell d^2)$ to protect a circuit of size $\mathcal{O}(\ell)$.

Example: the ISW Multiplicaition with 3 Shares

- Proposed by Yuval **I**shai, Amit **S**ahai and David **W**agner at *CRYPTO '03*.
- Input: $\hat{\mathbf{x}}[1], \hat{\mathbf{x}}[2], \hat{\mathbf{x}}[3]$ and $\hat{\mathbf{y}}[1], \hat{\mathbf{y}}[2], \hat{\mathbf{y}}[3]$, Output: $\hat{\mathbf{z}}[1], \hat{\mathbf{z}}[2], \hat{\mathbf{z}}[3]$

$\hat{\mathbf{x}}[1]\hat{\mathbf{y}}[1]$	$\hat{\mathbf{x}}[1]\hat{\mathbf{y}}[2]$ $+ r_1$	$\hat{\mathbf{x}}[1]\hat{\mathbf{y}}[3]$ $+ r_2$	$\rightarrow \hat{\mathbf{z}}[1]$
$\hat{\mathbf{x}}[2]\hat{\mathbf{y}}[1]$ $+ r_1$	$\hat{\mathbf{x}}[2]\hat{\mathbf{y}}[2]$	$\hat{\mathbf{x}}[2]\hat{\mathbf{y}}[3]$ $+ r_3$	$\rightarrow \hat{\mathbf{z}}[2]$
$\hat{\mathbf{x}}[3]\hat{\mathbf{y}}[1]$ $+ r_2$	$\hat{\mathbf{x}}[3]\hat{\mathbf{y}}[2]$ $+ r_3$	$\hat{\mathbf{x}}[3]\hat{\mathbf{y}}[3]$	$\rightarrow \hat{\mathbf{z}}[3]$

- It requires $\frac{\ell d(d+1)}{2}$ random bits and runs in $\mathcal{O}(\ell d^2)$ to protect a circuit of size $\mathcal{O}(\ell)$.

Goal: Reducing the Overheads

- Two approaches
  - Cost amortization
    - Weijia Wang et al.: Side-Channel Masking with Common Shares. TCHES 2022.
  - Precomputation
    - Weijia Wang et al.: Efficient Private Circuits with Precomputation. TCHES 2023.
- Application to the masked AES and SKINNY

Table of Contents

Table of Contents

Cost amortization

Goal: reducing the required random bits.

- Common shares: some shares of different variables are the same.
- Randomness can be reused among different operations.

Asymptotic complexity for a circuit of size $\mathcal{O}(\ell)$:

- The randomness complexity decrease: $\mathcal{O}(\ell d^2) \rightarrow \tilde{\mathcal{O}}(d^2)$
- The computational complexity does not change: $\tilde{\mathcal{O}}(\ell d^2)$

Two Types of Sharings

- Boolean sharing:
  - Secret variable $x \xrightarrow{\text{rand}}$ shares $\hat{\mathbf{x}}[1], \ldots, \hat{\mathbf{x}}[d+1]$ such that $x = \hat{\mathbf{x}}[1] \oplus \ldots \oplus \hat{\mathbf{x}}[d+1]$
  - Common shares are insecure.
    - Sharing of x: $\hat{\mathbf{x}}[1], \hat{\mathbf{s}}[1], \ldots, \hat{\mathbf{s}}[d]$
    - Sharing of y: $\hat{\mathbf{y}}[1], \hat{\mathbf{s}}[1], \ldots, \hat{\mathbf{s}}[d]$
    - $\hat{\mathbf{x}}[1] \oplus \hat{\mathbf{y}}[1] = x \oplus y$
- Inner product sharing:
  - Secret variable $x \xrightarrow{\text{rand}}$ shares $\hat{\mathbf{x}}[1], \ldots, \hat{\mathbf{x}}[d+1]$ such that $x = \hat{\mathbf{x}}[1] \oplus a_1\hat{\mathbf{x}}[2] \oplus \ldots, a_d\hat{\mathbf{x}}[d+1]$
  - Common shares can be secure!
    - Sharing of x: $\hat{\mathbf{x}}[1], \hat{\mathbf{s}}[1], \ldots, \hat{\mathbf{s}}[d]$ such that $x = \hat{\mathbf{x}}[1] \oplus a_1\hat{\mathbf{s}}[1] \oplus \ldots \oplus a_d\hat{\mathbf{s}}[d]$
    - Sharing of y: $\hat{\mathbf{y}}[1], \hat{\mathbf{s}}[1], \ldots, \hat{\mathbf{s}}[d]$ such that $y = \hat{\mathbf{x}}[1] \oplus b_1\hat{\mathbf{s}}[1] \oplus \ldots \oplus b_d\hat{\mathbf{s}}[d]$
    - Still d-probing secure if $(1, a_1, \ldots, a_d)$ and $(1, b_1, \ldots, b_d)$ are linearly independent.

Two Types of Sharings

- Boolean sharing:
  - Secret variable $x \xrightarrow{\text{rand}}$ shares $\hat{\mathbf{x}}[1], \ldots, \hat{\mathbf{x}}[d+1]$ such that $x = \hat{\mathbf{x}}[1] \oplus \ldots \oplus \hat{\mathbf{x}}[d+1]$
  - Common shares are insecure.
    - Sharing of x: $\hat{\mathbf{x}}[1], \hat{\mathbf{s}}[1], \ldots, \hat{\mathbf{s}}[d]$
    - Sharing of y: $\hat{\mathbf{y}}[1], \hat{\mathbf{s}}[1], \ldots, \hat{\mathbf{s}}[d]$
    - $\hat{\mathbf{x}}[1] \oplus \hat{\mathbf{y}}[1] = x \oplus y$
- Inner product sharing:
  - Secret variable $x \xrightarrow{\text{rand}}$ shares $\hat{\mathbf{x}}[1], \ldots, \hat{\mathbf{x}}[d+1]$ such that $x = \hat{\mathbf{x}}[1] \oplus a_1\hat{\mathbf{x}}[2] \oplus \ldots, a_d\hat{\mathbf{x}}[d+1]$
  - Common shares can be secure!
    - Sharing of x: $\hat{\mathbf{x}}[1], \hat{\mathbf{s}}[1], \ldots, \hat{\mathbf{s}}[d]$ such that $x = \hat{\mathbf{x}}[1] \oplus a_1\hat{\mathbf{s}}[1] \oplus \ldots \oplus a_d\hat{\mathbf{s}}[d]$
    - Sharing of y: $\hat{\mathbf{y}}[1], \hat{\mathbf{s}}[1], \ldots, \hat{\mathbf{s}}[d]$ such that $y = \hat{\mathbf{x}}[1] \oplus b_1\hat{\mathbf{s}}[1] \oplus \ldots \oplus b_d\hat{\mathbf{s}}[d]$
    - Still d-probing secure if $(1, a_1, \ldots, a_d)$ and $(1, b_1, \ldots, b_d)$ are linearly independent.

Masked Multiplications with Common Shares

- Input of Refresh: Boolean sharings.
- Output of Refresh: inner product sharings, allowing:
  - common shares;
  - randomness reuse.
- Output of Multiplicaiton:
  - Boolean shares.

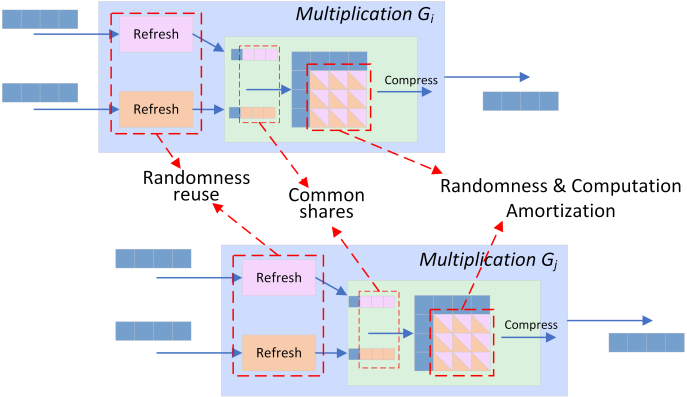
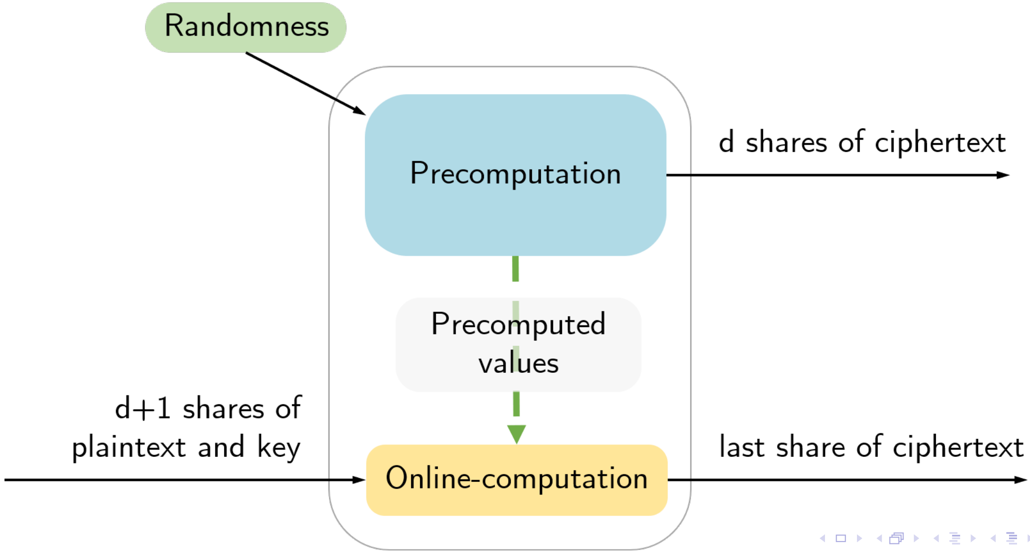
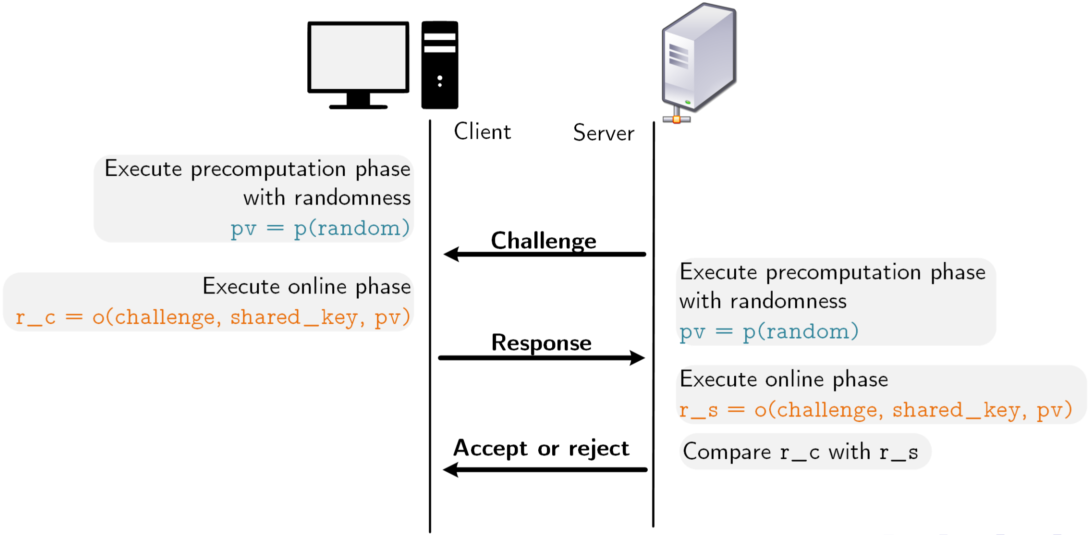


Table of Contents

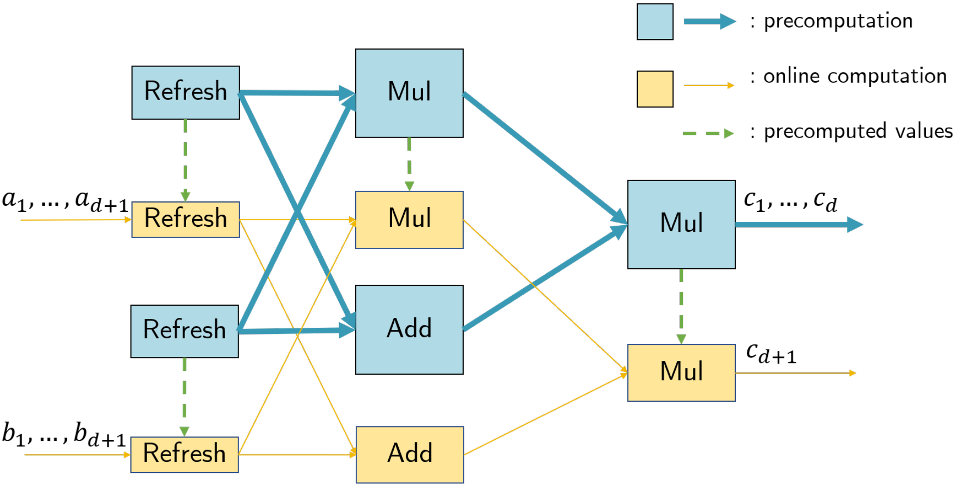
Precomputation-based Design Paradigm



Challenge-Response Protocol

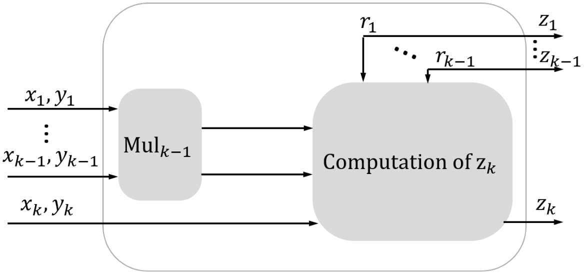


An Example of the Paradigm



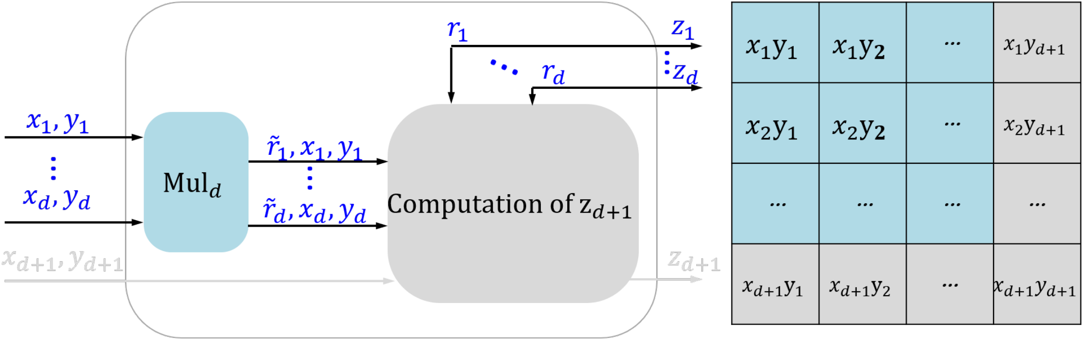
An example for $c = ab(a \oplus b)$ using multiplication, addition and refresh gadgets

New Masking Multiplication: Mul_k $(k \leq d + 1)$



- Input: $x_{1...k}$, $y_{1...k}$. Output: $z_{1...k}$
- The Mul_k is a recursive structure composed of 2 parts: Mul_{k-1} and computation of z_k
  - Mul_{k-1} computes temporary values $u_{1:k-1}$.
  - Random variables $r_{1...k-1}$ are used as output shares $z_{1...k-1}$
- Carefully arrange operation orders for the security.
  - Each output probe gives knowledge of at most one input share in the same index as the output probe
  - Each internal probe gives knowledge of at most one input share

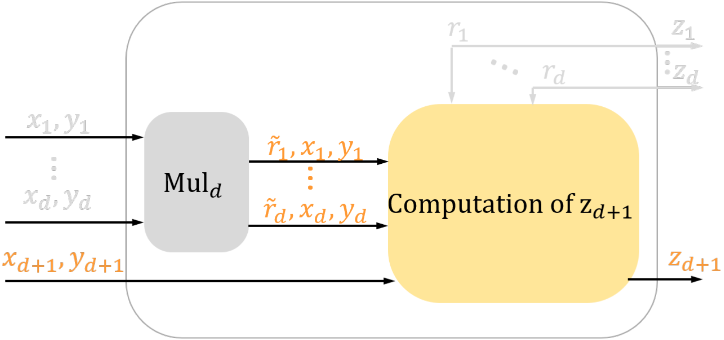
Mul_{d+1} with precomputation



Precomputation of Mul_{d+1}

Run in $O(d^2)$, produce $O(d)$ values and require $O(d^2)$ random values

Mul_{d+1} with precomputation



x_1y_1	x_1y_2	...	x_1y_{d+1}
x_2y_1	x_2y_2	...	x_2y_{d+1}
...	...	...	...
$x_{d+1}y_1$	$x_{d+1}y_2$	...	$x_{d+1}y_{d+1}$

Online computation of Mul_{d+1}

Run in $O(d)$ without any random value

Table of Contents

Implementation Results

		d	Kcycles for precomp.	Random bits	RAM for precomp.	Kcycles for online.
AES	[GR 17]	2	-	3.75 KB	3.75 KB	83.9
	[VV 21]	2	72590	**0.011 KB**	40.1 KB	423
	Our work A	2	705	96 Bytes	5.63 KB	60
	Our work B	2	**67.98**	2.22 KB	**2.91 KB**	**50.03**
	[GR 17]	8	-	45 KB	45 KB	404.5
	[VV 21]	8	3265303	**0.56 KB**	40.8 KB	2873
	Our work A	8	**3 662**	1.5 KB	11 KB	**137**
	Our work B	8	**446.34**	23.88 KB	11.66 KB	**92.27**
SKINNY -128	Our work B	2	159.28	1.91 KB	3.03 KB	75.48
	Our work B	8	749.2	22.62 KB	12.12 KB	117.72

- [GR 17]: State-of-the-art result with bitslicing without cost amortization or precomputation
- [VV 21]: State-of-the-art result with precomputation using look-up tables
- Our work A: Cost amortization & precomputation, but no bitslicing
- Our work B: Bitslicing & precomputation, but no cost amortization

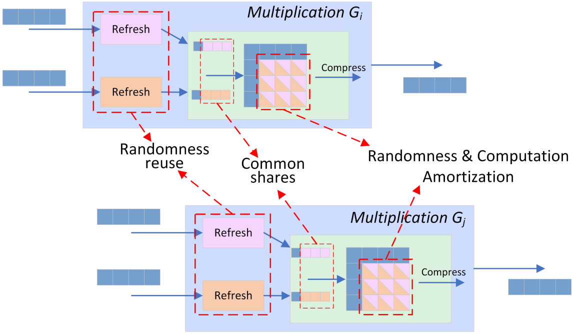
Table of Contents

Conclusion

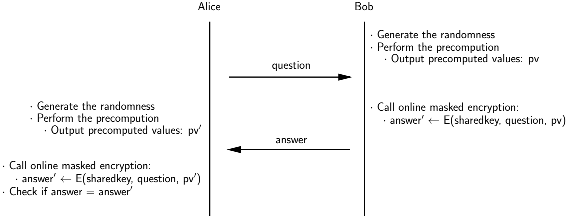
- Reducing the overhead of masking:
  - Cost amortized multiplication gadget with common shares
    - The randomness decreases: $\tilde{\mathcal{O}}(\ell d^2) \rightarrow \tilde{\mathcal{O}}(d^2)$
  - Precomputation-based design paradigm for masking
    - Pre-computation phase: $\tilde{\mathcal{O}}(\ell d^2)$ (computational), $\tilde{\mathcal{O}}(d^2)$ (randomness).
    - Online phase: $\mathcal{O}(\ell d)$ (computational), without any randomness.
- Applications
  - Saving a large amount of random bits
  - A speed-up for the online phase.

Conclusion

- Reducing the overhead of masking:
  - Cost amortized multiplication gadget with common shares
    - The randomness decreases: $\tilde{\mathcal{O}}(\ell d^2) \rightarrow \tilde{\mathcal{O}}(d^2)$
  - Precomputation-based design paradigm for masking
    - Pre-computation phase: $\tilde{\mathcal{O}}(\ell d^2)$ (computational), $\tilde{\mathcal{O}}(d^2)$ (randomness).
    - Online phase: $\mathcal{O}(\ell d)$ (computational), without any randomness.
- Applications
  - Saving a large amount of random bits
  - A speed-up for the online phase.



Cost amortization



Precomputation-based design paradigm

Thank You!