Side-Channel Masking with Common Shares

Weijia Wang Chun Guo Yu Yu Fanjie Ji Yang Su

Shandong University, China
Shanghai Jiao Tong University, China

September. 20th, 2022

Table of Contents

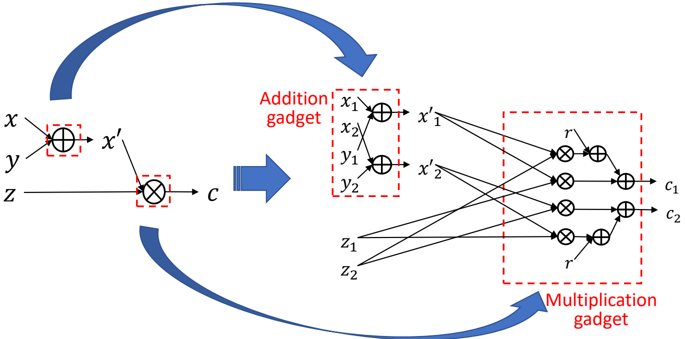
Table of Contents

Masking

- Randomize the secret (Enc)
  - Secret variable $x \xrightarrow{\text{rand}}$ shares $\hat{\mathbf{x}}[1], \ldots, \hat{\mathbf{x}}[d]$
    - Boolean masking: $x = \hat{\mathbf{x}}[0] \oplus \ldots \oplus \hat{\mathbf{x}}[d]$
  - Any d shares *are independent of* x
- Private computations (various gadgets, especially multiplication gadgets).
  - Any d intermediates *are independent of* the input secrets: d-privacy, d-probing security

Masking

- Randomize the secret (Enc)
  - Secret variable $x \xrightarrow{\text{rand}}$ shares $\hat{\mathbf{x}}[1], \ldots, \hat{\mathbf{x}}[d]$
    - Boolean masking: $x = \hat{\mathbf{x}}[0] \oplus \ldots \oplus \hat{\mathbf{x}}[d]$
  - Any d shares *are independent of* x
- Private computations (various gadgets, especially multiplication gadgets).
  - Any d intermediates *are independent of* the input secrets: d-privacy, d-probing security



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 $\frac{d(d+1)}{2}$ random variables and runs in complexity $\mathcal{O}(d^2)$.

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: $\hat{\mathbf{z}}[1], \hat{\mathbf{z}}[2], \hat{\mathbf{z}}[3]$

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

- It requires $\frac{d(d+1)}{2}$ random variables and runs in complexity $\mathcal{O}(d^2)$.

Example: the ISW Multiplicaition with 3 Shares

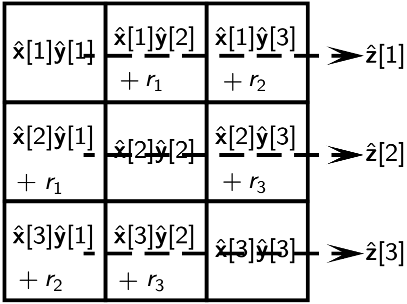
- Proposed by Yuval **I**shai, Amit **S**ahai and David **W**agner at *CRYPTO '03*.
- Input: $\hat{\mathbf{x}}[1], \hat{\mathbf{x}}[2], \hat{\mathbf{x}}[3]$ and $\hat{\mathbf{y}}[1], \hat{\mathbf{y}}[2], \hat{\mathbf{y}}[3]$, Output: $\hat{\mathbf{z}}[1], \hat{\mathbf{z}}[2], \hat{\mathbf{z}}[3]$

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

- It requires $\frac{d(d+1)}{2}$ random variables and runs in complexity $\mathcal{O}(d^2)$.

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 $\frac{d(d+1)}{2}$ random variables and runs in complexity $\mathcal{O}(d^2)$.

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: $\hat{\mathbf{z}}[1], \hat{\mathbf{z}}[2], \hat{\mathbf{z}}[3]$

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

- It requires $\frac{d(d+1)}{2}$ random variables and runs in complexity $\mathcal{O}(d^2)$.

A Summary of Contributions

Goal: reducing the overheads

- Theoretical contributions
  - Masked multiplication with common shares
  - Precomputation-based design paradigm.
  - New security notion: from parallel to general compositions.
- Application to the masked AES

Table of Contents

Table of Contents

Cost amortization

- A part of shares of different variables is the same.
- Randomness and intermediate variables can be reused among different operations.

Two Types of Sharings

- Boolean sharing:
  - Secret variable $x \xrightarrow{\text{rand}}$ shares $\hat{\mathbf{x}}[0], \ldots, \hat{\mathbf{x}}[d]$ such that $x = \hat{\mathbf{x}}[0] \oplus \ldots \oplus \hat{\mathbf{x}}[d]$
  - Common shares are insecure.
    - Sharing of a: $\hat{a}, \hat{\mathbf{s}}[1], \ldots, \hat{\mathbf{s}}[d]$
    - Sharing of b: $\hat{b}, \hat{\mathbf{s}}[1], \ldots, \hat{\mathbf{s}}[d]$
    - $\hat{a} \oplus \hat{b} = a \oplus b$
- Inner product sharing:
  - Secret variable $x \xrightarrow{\text{rand}}$ shares $\hat{\mathbf{x}}[0], \ldots, \hat{\mathbf{x}}[d]$ such that $x = \hat{\mathbf{x}}[0] \oplus \alpha[1]\hat{\mathbf{x}}[1] \oplus \ldots, \alpha[d]\hat{\mathbf{x}}[d]$
  - Common shares can be secure!
    - Sharing of a: $\hat{a}, \hat{\mathbf{s}}[1], \ldots, \hat{\mathbf{s}}[d]$ such that $a = \hat{a} \oplus \alpha_a[1]\hat{\mathbf{s}}[1] \oplus \ldots \oplus \alpha_a[d]\hat{\mathbf{s}}[d]$
    - Sharing of b: $\hat{b}, \hat{\mathbf{s}}[1], \ldots, \hat{\mathbf{s}}[d]$ such that $b = \hat{b} \oplus \alpha_b[1]\hat{\mathbf{s}}[1] \oplus \ldots \oplus \alpha_b[d]\hat{\mathbf{s}}[d]$
    - Still d-probing secure if $(1, \alpha_a)$ and $(1, \alpha_b)$ are linearly independent.

Two Types of Sharings

- Boolean sharing:
  - Secret variable $x \xrightarrow{\text{rand}}$ shares $\hat{\mathbf{x}}[0], \ldots, \hat{\mathbf{x}}[d]$ such that $x = \hat{\mathbf{x}}[0] \oplus \ldots \oplus \hat{\mathbf{x}}[d]$
  - Common shares are insecure.
    - Sharing of a: $\hat{a}, \hat{\mathbf{s}}[1], \ldots, \hat{\mathbf{s}}[d]$
    - Sharing of b: $\hat{b}, \hat{\mathbf{s}}[1], \ldots, \hat{\mathbf{s}}[d]$
    - $\hat{a} \oplus \hat{b} = a \oplus b$
- Inner product sharing:
  - Secret variable $x \xrightarrow{\text{rand}}$ shares $\hat{\mathbf{x}}[0], \ldots, \hat{\mathbf{x}}[d]$ such that $x = \hat{\mathbf{x}}[0] \oplus \boldsymbol{\alpha}[1]\hat{\mathbf{x}}[1] \oplus \ldots, \boldsymbol{\alpha}[d]\hat{\mathbf{x}}[d]$
  - Common shares can be secure!
    - Sharing of a: $\hat{a}, \hat{\mathbf{s}}[1], \ldots, \hat{\mathbf{s}}[d]$ such that $a = \hat{a} \oplus \boldsymbol{\alpha}_a[1]\hat{\mathbf{s}}[1] \oplus \ldots \oplus \boldsymbol{\alpha}_a[d]\hat{\mathbf{s}}[d]$
    - Sharing of b: $\hat{b}, \hat{\mathbf{s}}[1], \ldots, \hat{\mathbf{s}}[d]$ such that $b = \hat{b} \oplus \boldsymbol{\alpha}_b[1]\hat{\mathbf{s}}[1] \oplus \ldots \oplus \boldsymbol{\alpha}_b[d]\hat{\mathbf{s}}[d]$
    - Still d-probing secure if $(1, \boldsymbol{\alpha}_a)$ and $(1, \boldsymbol{\alpha}_b)$ are linearly independent.

Masked Multiplications with Common Shares

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

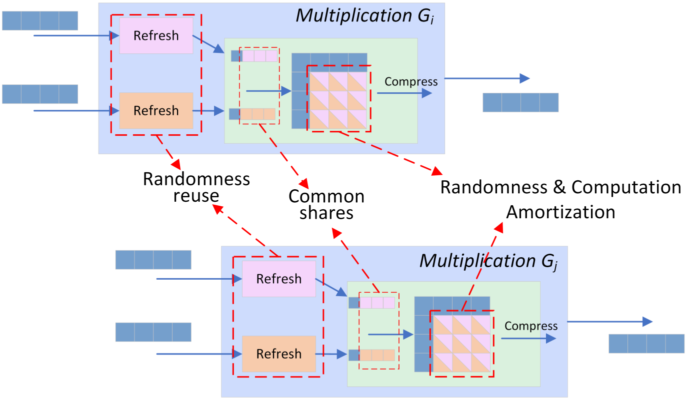
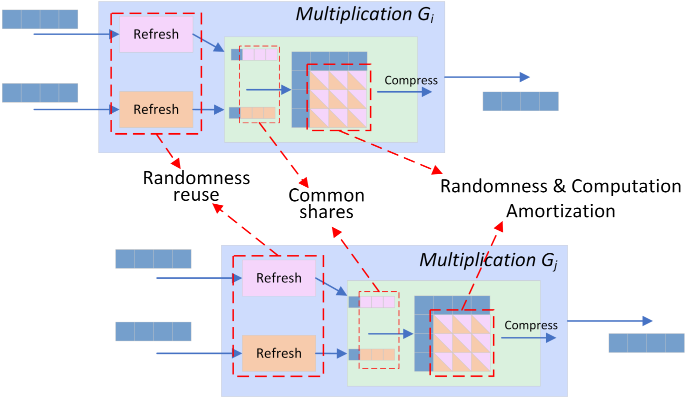


Table of Contents

Precomputation-based Design Paradigm



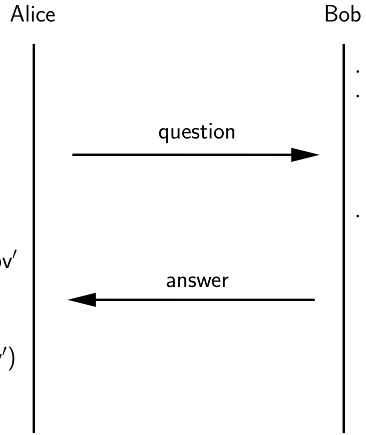
A promising feature: most of the intermediate variables can be precomputed

Precomputation-based Design Paradigm

A masked implementation of a crytographic function f can be divided into:

- Precomputation phase:
  - Precompute a set of vairbles $\mathcal{V}$.
  - Runs in $O(\ell d^2)$ and requires $O(\ell d^2)$ random bits
  - Not one-time effort: it runs for each calls of f.
    - But (important!): the precomputation does not requires the input of f.
- Online-computation phase:
  - Very efficient: runs in $O(\ell d)$ without any random bits.

Precomputation-based Design Paradigm



Bob:

- Generate the randomness
- Perform the precomption
  - Output precomputed values: pv

Alice:

- Generate the randomness
- Perform the precomption
  - Output precomputed values: pv′

Bob:

- Call online masked encryption:
  - answer′ ← E(sharedkey, question, pv)

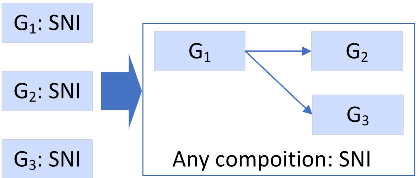
Alice:

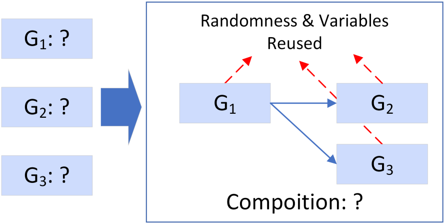
- Call online masked encryption:
  - answer′ ← E(sharedkey, question, pv′)
- Check if answer = answer′

Table of Contents

Problem Statement

- Composable security notions: one can concentrate on analyzing every single gadget, and leave the rest to the probe propagation.
  - Non-Inference/Strong Non-Inference (NI/SNI)
  - Probe-Isolating Non-Inference (PINI)
- The randomness used in different gadget should be independent.
  - It is not complied to our case (and other masking schemes with randomness reuse).

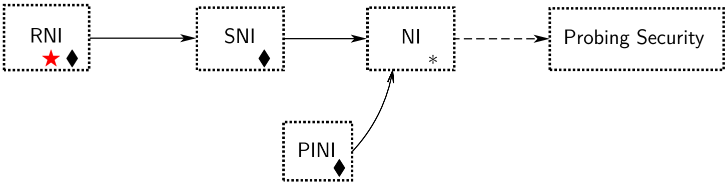




New Security Notion

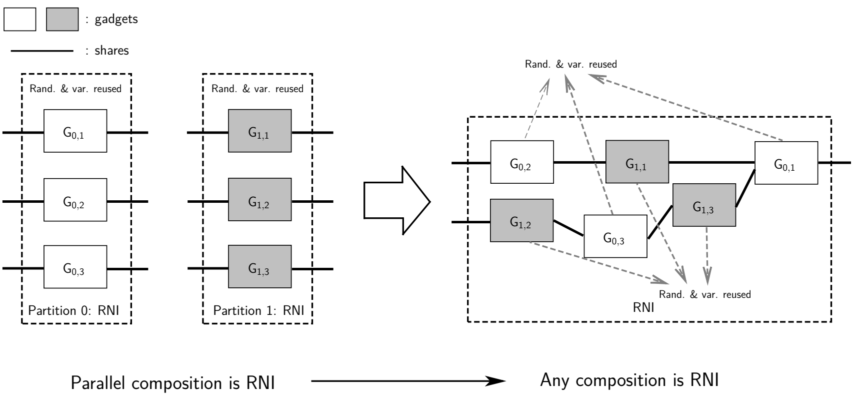
- Randomness Reusable Non-Inference: RNI

⟶: implication
- - -⟶: implication if any d input shares are independent of the secret
★: Supporting compositions even if randomness/variables are reused
♦: Supporting trivial composition if random bits are independent
$*$: Supporting compositions (with SNI refreshing) if random bits are independent



Relations of different security notions

Composability of RNI



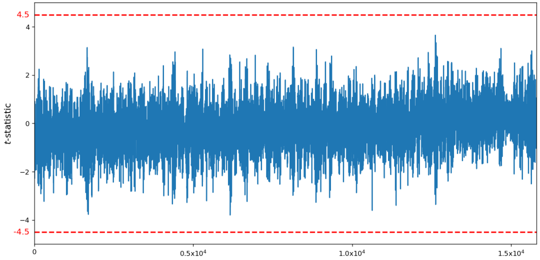
An arbitrary composition (on the right) that can be described a bipartite graph is RNI, as long as the parallel compositions (on the left) of gadgets in each partition is RNI.

Table of Contents

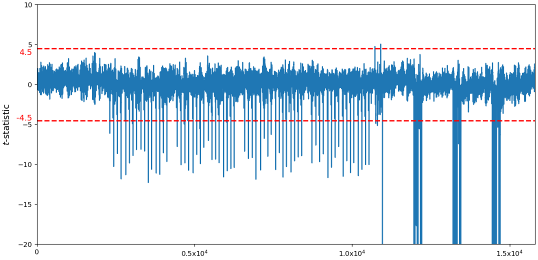
Summary of Masked AES Implementations

	d	KCycles for Precomp.	RAM size	KCycles/penalty factor for online
Unprotected	-	-	-	9.33 / 1
BS method	2	2 880 (random gen.)	1.92 KB	62 / 6.65
LUT method	2	15 360 (random gen.)	10.24 KB	435 / 46.62
Our work	2	144(random gen.) + 705	5.63 KB	**60** / 6.43
BS method	8	34 560 (random gen.)	23.04 KB	330 / 35.36
LUT method	8	245 760 (random gen.)	164 KB	unreported
Our work	8	2 304(random gen.) + 3 66	11 KB	**137** / 14.68

T-test results, security order $d = 1$



(a) New masking



(b) Boolean masking

- In the implementation, we do not attempt to eliminate all the transitional leakage that may damage the independent leakage assumption.

Discussion on the results

- This (good) result for the case of $d = 1$ is a bit surprising, since:
  - there is transitional leakage, but it is still secure
- The new scheme is more robust to some lapses (e.g., transitional leakage) in implementation.
  - We contribute this advantage to the relatively more complex algebraic structure than the Boolean masking.

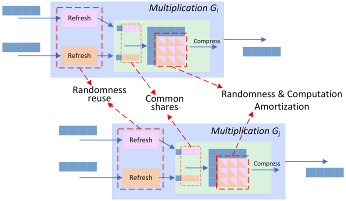
Table of Contents

Conclusion

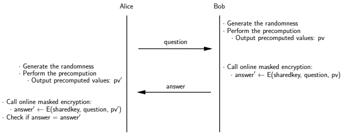
- Theoretical contributions:
  - Cost amortized multiplication gadget with common shares
    - The computational decreases: $\tilde{\mathcal{O}}(\ell d^2)$
    - The randomness decreases: $\tilde{\mathcal{O}}(d^2)$
  - Precomputation-based design paradigm for masking
    - Pre-computation phase: $\tilde{\mathcal{O}}(\ell d^2)$ (computational), $\tilde{\mathcal{O}}(d^2)$ (randomness).
    - Online phase: $\tilde{\mathcal{O}}(\ell d)$ (computational), without any randomness.
  - New security notion for proofs: from parallel to general compositions
    - Intrinsically supports randomness/variables reusing.
- Application to AES.
  - A speed-up for the online phase.
  - More robust to some lapses (e.g., transitional leakage) in implementation.

Conclusion

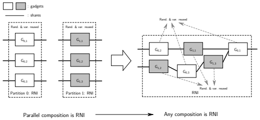
- Theoretical contributions:
  - Cost amortized multiplication gadget with common shares
    - The computational decreases: $\tilde{\mathcal{O}}(\ell d^2)$
    - The randomness decreases: $\tilde{\mathcal{O}}(d^2)$
  - Precomputation-based design paradigm for masking
    - Pre-computation phase: $\tilde{\mathcal{O}}(\ell d^2)$ (computational), $\tilde{\mathcal{O}}(d^2)$ (randomness).
    - Online phase: $\tilde{\mathcal{O}}(\ell d)$ (computational), without any randomness.
  - New security notion for proofs: from parallel to general compositions
    - Intrinsically supports randomness/variables reusing.
- Application to AES.
  - A speed-up for the online phase.
  - More robust to some lapses (e.g., transitional leakage) in implementation.



New construction



New paradigm



New proof method

Thank You!