Revisiting Time-Memory Trade-offs in Secure Cryptographic Implementations on Resource Constrained Devices

Dr. Srinivas Vivek

IIIT Bangalore

srinivas.vivek@iiitb.ac.in

5th March, 2019
ICIRE 2019

Motivation

Information Security & Cryptography

Information Security: study of techniques to prevent unauthorised access or use of information.

Basic goals:

- Confidentiality
- Authentication
- Data integrity
- Non-repudiation

Cryptography: provides mathematical foundations and techniques to realise the above goals.

Cryptography

Cryptography has been used since the Roman times, and also in the World Wars.

Figure: Enigma Cipher Machine

Cryptanalysis & Core Problems

Cryptanalysis: study of techniques to defeat the goals of cryptographic primitives and protocols.

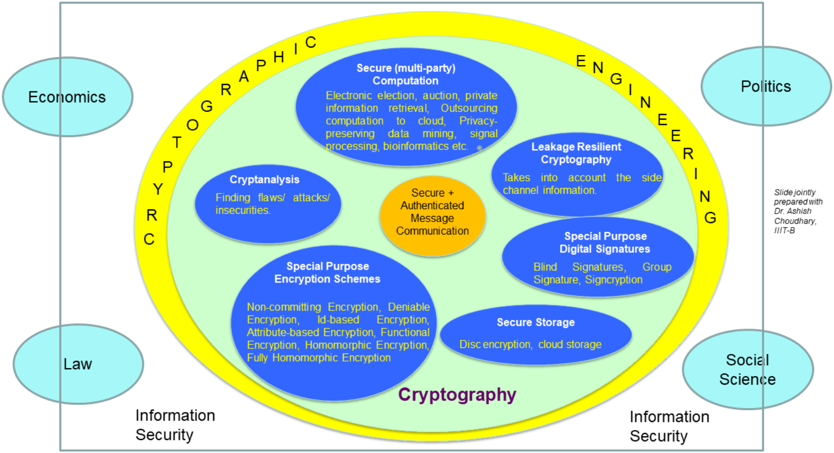
Core problems in traditional cryptography (upto 1980s)

- Key establishment
- Secure communication
  - Confidentiality
  - Integrity

Cryptanalysis & Core Problems

Figure: Evesdropping Adversary

Domain of Cryptology in Information Security



Slide jointly prepared with Dr. Ashish Choudhary, IIIT-B

Side-Channel Attacks

Traditionally, cryptosystems were viewed as black-boxes.

Change of view in the crypto research community since mid-90s due to Kocher et al.

Figure: Adversary inspecting an execution

Side-Channel Attacks

Traditionally, cryptosystems were viewed as black-boxes.

Change of view in the crypto research community since mid-90s due to Kocher et al.

Pre-history of implementation-based attacks

- WWI: Eavesdropping field telephones.
- WWII: Bell Labs electromagnetic side-channel attack.
- MI5/GCHQ acoustic side-channel attacks.
- TEMPEST: US government classified program.

Side-Channel Attacks

Examples:

- Timing attacks
- Power analysis attacks
  - Simple power analysis
  - Differential power analysis
  - Template attacks
- Electro-magnetic attacks
- Cache attacks
- Others: acoustics, thermal, photonic emmision attacks

Different **operations + data** $\Longrightarrow$ Different **physical “leakage”**.

Side-Channel Attacks

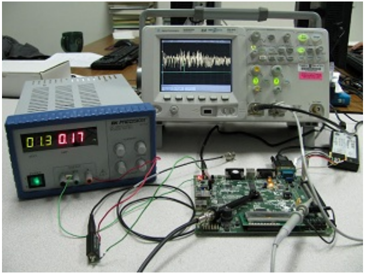


Figure: PAA experiment setup

Side-Channel Attacks

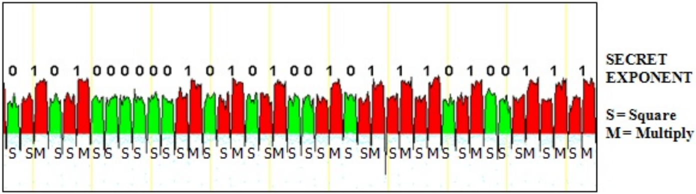


Figure: SPA attack on an RSA implementation

Side-Channel Attacks

Practical threat for embedded device implementations.

- Microcontrollers and smart cards vulnerable to power analysis attacks.
- Other IoT devices are vulnerable too.

Even advanced architectures are prone to cache, timing, and power attacks.

Possible to mount side-channel attacks *remotely* by injecting malware.

"*Attacks only get better*" – K. G. Patterson.

SCA & countermeasures - active research area since two decades.

Countermeasures against SCA

Goal: minimise the effect of side-channel leakage.

In this talk, we focus only on countermeasures against power analysis attacks.

Countermeasures against PAA can be broadly categorised as:

- Make the leakage of the device independent of intermediate variables.
  - E.g: *Hiding countermeasure*
- Make intermediate variables independent of secret variables.
  - E.g.: *Masking countermeasure*

Countermeasures against SCA

Goal: minimise the effect of side-channel leakage.

In this talk, we focus only on countermeasures against power analysis attacks.

Countermeasures against PAA can be broadly categorised as:

- Make the leakage of the device independent of intermediate variables.
  - E.g: *Hiding countermeasure*
- Make intermediate variables independent of secret variables.
  - E.g.: *Masking countermeasure*

Masking Countermeasure

Masking: a popular countermeasure against *DPA* attacks.

Well-suited to protect block cipher s/w and h/w implementations.

Method: each sensitive variable $x \in \mathbb{F}_{2^n}$ is secret shared.

- $x = x_0 \oplus x_1 \oplus \ldots \oplus x_v$
- *Security*: (any subset of) intermediate variables are independent of x.

Security offered has been relatively well analysed

- *probing* [*ISW03*] & *noisy* leakage model [*CJJR99, RP13, DDF14*].
- Loosely speaking, SCA complexity is exponential w.r.t. v.

[*ISW03*] Y. Ishai, A. Sahai, D. Wagner. *Private circuits: Securing hardware against probing attacks.* CRYPTO'03.
[*CJRR99*] S. Chari, C.S. Jutla, J.R. Rao, P. Rohatgi. *Towards sound approaches to counteract PAA*. CRYPTO'99.
[*RP10*] M. Rivain, E. Prouff. *Provably secure higher-order masking of AES*. CHES'10.
[*DDF14*] A. Duc, S. Dziembowski, S. Faust. *Unifying Leakage Models: From Probing Attacks to Noisy Leakage.* EUROCRYPT'14.

Masking Countermeasure

Masking: a popular countermeasure against *DPA* attacks.

Well-suited to protect block cipher s/w and h/w implementations.

Method: each sensitive variable $x \in \mathbb{F}_{2^n}$ is secret shared.

- $x = x_0 \oplus x_1 \oplus \ldots \oplus x_v$
- *Security*: (any subset of) intermediate variables are independent of x.

Security offered has been relatively well analysed

- *probing* [*ISW03*] & *noisy* leakage model [*CJJR99, RP13, DDF14*].
- Loosely speaking, SCA complexity is exponential w.r.t. v.

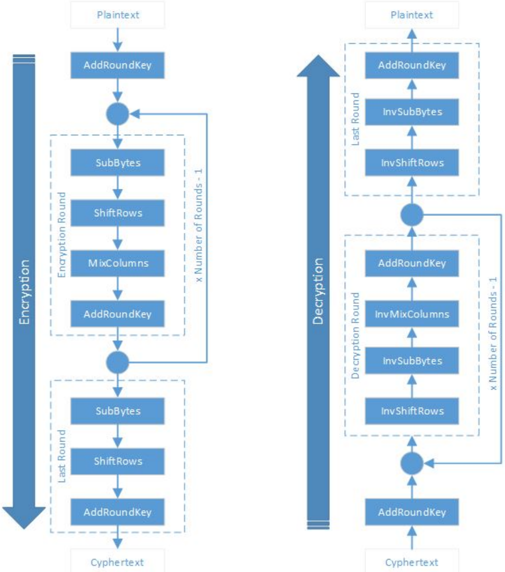
[*ISW03*] Y. Ishai, A. Sahai, D. Wagner. *Private circuits: Securing hardware against probing attacks*. CRYPTO'03.
[*CJRR99*] S. Chari, C.S. Jutla, J.R. Rao, P. Rohatgi. *Towards sound approaches to counteract PAA*. CRYPTO'99.
[*RP10*] M. Rivain, E. Prouff. *Provably secure higher-order masking of AES*. CHES'10.
[*DDF14*] A. Duc, S. Dziembowski, S. Faust. *Unifying Leakage Models: From Probing Attacks to Noisy Leakage*. EUROCRYPT'14.

Masking of Block Ciphers

Block cipher: a symmetric-key cryptographic primitive used in many cryptographic constructions

- *E.g.*: DES, AES, PRESENT, etc.

AES



Plaintext
AddRoundKey
SubBytes
ShiftRows
MixColumns
AddRoundKey
Encryption Round
x Number of Rounds - 1
SubBytes
ShiftRows
AddRoundKey
Last Round
Cyphertext
Encryption
Plaintext
AddRoundKey
InvSubBytes
InvShiftRows
Last Round
AddRoundKey
InvMixColumns
InvSubBytes
InvShiftRows
Decryption Round
x Number of Rounds - 1
AddRoundKey
Cyphertext
Decryption

Masking of Block Ciphers

Linear/Affine functions are straightforward to compute in presence of shares.

- $f(x) = f(x_0) \oplus f(x_1) \oplus \ldots \oplus f(x_v)$
- Time and randomness complexity are both linear in the number of shares.

Main challenge is to securely compute *non-linear* functions.

- Various masking schemes differ mainly in how these functions are evaluated.
- For block ciphers, this reduces to securing their S-boxes.

Masking of Block Ciphers

Linear/Affine functions are straightforward to compute in presence of shares.

- $f(x) = f(x_0) \oplus f(x_1) \oplus \ldots \oplus f(x_v)$
- Time and randomness complexity are both linear in the number of shares.

Main challenge is to securely compute *non-linear* functions.

- Various masking schemes differ mainly in how these functions are evaluated.
- For block ciphers, this reduces to securing their S-boxes.

Table-based S-box Masking

Originally proposed in [CJJR99].

Input:

- (n, m)-S-box
- Two input shares x_1, x_2, s.t.

$$x = x_1 \oplus x_2$$

Output:

- Two output shares y_1, y_2, s.t.

$$S(x) = y_1 \oplus y_2$$

Table-based S-box Masking

Originally proposed in [CJJR99].

Input:

- (n, m)-S-box
- Two input shares x_1, x_2, s.t.

$$x = x_1 \oplus x_2$$

Output:

- Two output shares y_1, y_2, s.t.

$$S(x) = y_1 \oplus y_2$$

Table-based 1-O S-box Masking (cont'd)

Method:

- Create a temporary table T in RAM s.t.

$$T(a) = S(x_1 \oplus a) \oplus y_1 \qquad \forall\, a \in \{0,1\}^n$$

- Compute: $y_2 = T(x_2)$
- *Output shares*: y_1, y_2

Correctness: $S(x) = y_1 \oplus y_2$

1-O Security: first-order secure in the probing model.

- Every intermediate variable (incl. i/o) independent of x.

Table-based 1-O S-box Masking (cont'd)

Method:

- Create a temporary table T in RAM s.t.

$$T(a) = S(x_1 \oplus a) \oplus y_1 \qquad \forall\, a \in \{0,1\}^n$$

- Compute: $y_2 = T(x_2)$
- *Output shares*: y_1, y_2

Correctness: $S(x) = y_1 \oplus y_2$

1-O Security: first-order secure in the probing model.

- Every intermediate variable (incl. i/o) independent of x.

Table-based 1-O S-box Masking (cont'd)

For an (n, m)-S-box:

- Pre-processing (offline) run time: $O((n + m) \cdot 2^n)$
- *Look-up (online) time*: $O(n + m)$
- **RAM memory**: $O(m \cdot 2^n)$ bits
- Randomess: None

AES: time overhead factor: **2 to 4**, RAM memory = **256 bytes**.

RAM Memory can be *expensive* for highly resource-constrained environments.

Alternate approaches exist ([*PR07*]): $O(1)$ RAM but time overhead factor ≥ 30 .

[*PR07*] E. Prouff, M. Rivain. *A generic method for secure Sbox implementation*. WISA'07.

Table-based 1-O S-box Masking (cont'd)

For an (n, m)-S-box:

- Pre-processing (offline) run time: $O((n+m) \cdot 2^n)$
- *Look-up (online) time*: $O(n+m)$
- **RAM memory**: $O(m \cdot 2^n)$ bits
- Randomess: None

AES: time overhead factor: **2 to 4**, RAM memory = **256 bytes.**

RAM Memory can be *expensive* for highly resource-constrained environments.

Alternate approaches exist ([*PR07*]): $O(1)$ RAM but time overhead factor ≥ 30 .

[*PR07*] E. Prouff, M. Rivain. *A generic method for secure Sbox implementation*. WISA'07.

Look-up Table Compression

A look-up table compression scheme was proposed in [*RRST02*].

RAM Memory reduced by a factor ℓ.

- **Compression level**: ℓ $(1 \leq \ell \leq m)$
- Size of Table $T \approx \frac{(m \cdot 2^n)}{\ell}$ bits

Time-memory trade offs by varying ℓ

- bigger $\ell \Rightarrow$ lesser RAM
- bigger $\ell \Rightarrow$ greater online time

[*RRST02*] J.R. Rao, P. Rohatgi, H. Scherzer, S. Tinguely. *Partitioning attacks: Or how to rapidly clone some GSM cards*. IEEE S&P'02.

Improved First-Order Look-up Table Compression

An *improved* look-up table compression scheme was by Vadnala [*Vad17*].

- Variant of [*RRST02*].

RAM Memory reduced by a factor $\approx \mathbf{2}^{\ell}$ (**instead** of ℓ).

- **Compression level**: ℓ ($1 \leq \ell \leq \mathbf{n}$)
- Size of Table $T \approx m \cdot 2^{n-\ell} + (n - \ell) \cdot 2^{\ell}$ bits

Time-memory trade offs by varying ℓ

[*Vad17*] P.K. Vadnala. *Time-memory trade-offs for side-channel resistant implementations of block ciphers*. CT-RSA'17.

[Vad17] Table Compression Scheme (cont'd)

Idea: "pack" 2^ℓ table entries of the original T.

$$a = \underbrace{a^{(1)}}_{n-\ell} \,||\, \underbrace{a^{(2)}}_{\ell}$$

Step 1: create Table $T_1 : \{0,1\}^{n-\ell} \to \{0,1\}^m$ s.t.

$$T_1(a^{(1)}) = \left(\bigoplus_{i \in \{0,1\}^\ell} S((a^{(1)} \oplus r_i) \,||\, i) \right) \oplus y_1, \quad \forall\, a^{(1)} \in \{0,1\}^{n-\ell}$$

- $r_i \in \{0,1\}^{n-\ell}$ uniform random and independent
- y_1: an output share

Randomess complexity: 2^ℓ ($n-\ell$-bit) words.

- *Recall*: original table-based method needs no additional randomness.

[Vad17] Table Compression Scheme (cont'd)

Idea: "pack" 2^ℓ table entries of the original T.

$$a = \underbrace{a^{(1)}}_{n-\ell} \,||\, \underbrace{a^{(2)}}_{\ell}$$

Step 1: create Table $T_1 : \{0,1\}^{n-\ell} \rightarrow \{0,1\}^m$ s.t.

$$T_1(a^{(1)}) = \left(\bigoplus_{i \in \{0,1\}^\ell} S((a^{(1)} \oplus r_i) \,||\, i) \right) \oplus y_1, \quad \forall\, a^{(1)} \in \{0,1\}^{n-\ell}$$

- $r_i \in \{0,1\}^{n-\ell}$ uniform random and independent
- y_1: an output share

Randomess complexity: 2^ℓ ($n - \ell$-bit) words.

- *Recall*: original table-based method needs no additional randomness.

[Vad17] Table Compression Scheme (cont'd)

Idea: "pack" 2^ℓ table entries of the original T.

$$a = \underbrace{a^{(1)}}_{n-\ell} \,||\, \underbrace{a^{(2)}}_{\ell}$$

Step 1: create Table $T_1 : \{0,1\}^{n-\ell} \to \{0,1\}^m$ s.t.

$$T_1(a^{(1)}) = \left(\bigoplus_{i \in \{0,1\}^\ell} S((a^{(1)} \oplus r_i) \,||\, i) \right) \oplus y_1, \quad \forall\, a^{(1)} \in \{0,1\}^{n-\ell}$$

- $r_i \in \{0,1\}^{n-\ell}$ uniform random and independent
- y_1: an output share

Randomess complexity: 2^ℓ ($n-\ell$-bit) words.

- *Recall*: original table-based method needs no additional randomness.

[Vad17] Table Compression Scheme (cont'd)

Let the (secret) input be x

$$x = \underbrace{x^{(1)}}_{n-\ell} \,||\, \underbrace{x^{(2)}}_{\ell}$$

Step 2: "create" Table $U : \{0,1\}^{\ell} \rightarrow \{0,1\}^{m}$ in RAM s.t.

$$U(i) = S(x^{(1)} \,||\, i) \oplus y_1, \quad \forall\, i \in \{0,1\}^{\ell}$$

by "securely" accessing tables T_1 and S.

NOTE: U **cannot** be directly computed from S and x.

Step 3: "securely" compute the second output share

$$y_2 = U(x^{(2)}) = S(x) \oplus y_1$$

NOTE: Actually need Table T_2 = Table U shifted by shares of $x^{(2)}$.

[Vad17] Table Compression Scheme (cont'd)

Let the (secret) input be x

$$x = \underbrace{x^{(1)}}_{n-\ell} \mid\mid \underbrace{x^{(2)}}_{\ell}$$

Step 2: "create" Table $U : \{0,1\}^{\ell} \rightarrow \{0,1\}^{m}$ in RAM s.t.

$$U(i) = S(x^{(1)} \mid\mid i) \oplus y_1, \quad \forall\, i \in \{0,1\}^{\ell}$$

by "securely" accessing tables T_1 and S.

NOTE: U **cannot** be directly computed from S and x.

Step 3: "securely" compute the second output share

$$y_2 = U(x^{(2)}) = S(x) \oplus y_1$$

NOTE: Actually need Table T_2 = Table U shifted by shares of $x^{(2)}$.

Our Contribution - Part 1

Improving Randomness Complexity

We improved the **randomness complexity** of the first-order secure table compression masking scheme from [*Vad17*].

- Still retaining first-order security (in the probing model).

New randomness complexity is $\approx \ell$ ($n - \ell$-bit) words, *instead* of $\approx \mathbf{2}^{\ell}$ ($n - \ell$-bit) words.

We prove that the achieved complexity is **optimal**.

RAM memory remains unchanged.

Running time "essentially" remains unchanged on big-architectures.

- May possibly improve for highly-resource constrained environments.

Improving Randomness Complexity: Our Method

Recall: first step in [*Vad17*] needs to generate $r_i \xleftarrow{\$} \{0,1\}^{n-\ell}$

$$T_1(a^{(1)}) = \left(\bigoplus_{i \in \{0,1\}^\ell} S((a^{(1)} \oplus r_i) \parallel i) \right) \oplus y_1, \quad \forall\, a^{(1)} \in \{0,1\}^{n-\ell}$$

Our Idea

- Sufficient for r_i to be **pair-wise independent** (and unif. random).
- Sample $\ell + 1$ no. of $\gamma_j \xleftarrow{\$} \{0,1\}^{n-\ell}$.
- Compute r_i as **subset xor sum** of γ_j.
- Rest of the method essentially remains the same

One extra γ is needed as otherwise $r_0 = 0$.

Security proof: enumerate all intermediate variables and show independence.

Improving Randomness Complexity: Our Method

Recall: first step in [*Vad17*] needs to generate $r_i \xleftarrow{\$} \{0,1\}^{n-\ell}$

$$T_1(a^{(1)}) = \left(\bigoplus_{i \in \{0,1\}^\ell} S((a^{(1)} \oplus r_i) \parallel i) \right) \oplus y_1, \quad \forall\, a^{(1)} \in \{0,1\}^{n-\ell}$$

Our Idea

- Sufficient for r_i to be **pair-wise independent** (and unif. random).
- Sample $\ell + 1$ no. of $\gamma_j \xleftarrow{\$} \{0,1\}^{n-\ell}$.
- Compute r_i as **subset xor sum** of γ_j.
- Rest of the method essentially remains the same

One extra γ is needed as otherwise $r_0 = 0$.

Security proof: enumerate all intermediate variables and show independence.

Improving Randomness Complexity: Our Method

Recall: first step in [*Vad17*] needs to generate $r_i \xleftarrow{\$} \{0,1\}^{n-\ell}$

$$T_1(a^{(1)}) = \left(\bigoplus_{i \in \{0,1\}^\ell} S((a^{(1)} \oplus r_i) \,||\, i) \right) \oplus y_1, \quad \forall\, a^{(1)} \in \{0,1\}^{n-\ell}$$

Our Idea

- Sufficient for r_i to be **pair-wise independent** (and unif. random).
- Sample $\ell + 1$ no. of $\gamma_j \xleftarrow{\$} \{0,1\}^{n-\ell}$.
- Compute r_i as **subset xor sum** of γ_j.
- Rest of the method essentially remains the same

One extra γ is needed as otherwise $r_0 = 0$.

Security proof: enumerate all intermediate variables and show independence.

Randomness Complexity: Proof of Optimality

Algebraic lower bound: At least ℓ values of γ_j are needed.

- **Computation model**: only $\mathbb{F}_2$-linear operations are performed.

(Nearly) all the known table-based masking schemes use **only xor** for arithmetic operations.

Proof idea: assume only **t** $< \ell$ many γ_j were used.

$$r_i = c_i \bigoplus_{1 \le j \le t} b_j \cdot \gamma_j, \qquad \text{where} \quad b_j \in \mathbb{F}_2,\ c_i \in \{0,1\}^{n-l}$$

There exist r_p, r_q ($p \neq q$) s.t.

$$r_p \oplus r_q = c_p \oplus c_q$$

Then an intermediate variable depends on bits of x

$$ind_2 = x^{(1)} \oplus c_p \oplus c_q$$

Randomness Complexity: Proof of Optimality

Algebraic lower bound: At least ℓ values of γ_j are needed.

- **Computation model**: only $\mathbb{F}_2$-linear operations are performed.

(Nearly) all the known table-based masking schemes use **only xor** for arithmetic operations.

Proof idea: assume only **t** $< \ell$ many γ_j were used.

$$r_i = c_i \bigoplus_{1 \leq j \leq t} b_j \cdot \gamma_j, \qquad \text{where} \quad b_j \in \mathbb{F}_2,\ c_i \in \{0,1\}^{n-l}$$

There exist r_p, r_q ($p \neq q$) s.t.

$$r_p \oplus r_q = c_p \oplus c_q$$

Then an intermediate variable depends on bits of x

$$ind_2 = x^{(1)} \oplus c_p \oplus c_q$$

Our Contribution - Part 2

Attack on the 2-O Table Compression Scheme [Vad17]

[*Vad17*] also proposed a **second-order** table compression scheme

- Generalisation of the first-order method.
- **Claimed** to be **second-order secure** in the probing leakage model.

We **contradict** the second-order security

- **Attack**: there exist **several pairs** of intermediate variables that **jointly depend** on the secret input.

Attack on the 2-O Table Compression Scheme [Vad17]

[*Vad17*] also proposed a **second-order** table compression scheme

- Generalisation of the first-order method.
- **Claimed** to be **second-order secure** in the probing leakage model.

We **contradict** the second-order security

- **Attack**: there exist **several pairs** of intermediate variables that **jointly depend** on the secret input.

2-O Table Compression Scheme [Vad17]

Three steps *similar* to the first-order scheme.

Step 1: Create Table $T_1 : \{0,1\}^{n-\ell} \rightarrow \{0,1\}^m$

2-O Table Compression Scheme [Vad17]

Three steps *similar* to the first-order scheme.

Step 1: Create Table $T_1 : \{0,1\}^{n-\ell} \rightarrow \{0,1\}^m$

$$T_1(b^{(1)}) := \left(\left(\bigoplus_{i\in\{0,1\}^\ell} S((x_3{}^{(1)} \oplus a^{(1)} \oplus r_i) \,||\, i)\right) \oplus y_1\right) \oplus y_2$$

2-O Table Compression Scheme [Vad17]

Three steps *similar* to the first-order scheme.

Step 1: Create Table $T_1 : \{0,1\}^{n-\ell} \to \{0,1\}^m$

Step 2: Create Table $T_2 : \{0,1\}^{\ell} \to \{0,1\}^m$

2-O Table Compression Scheme [Vad17]

Three steps *similar* to the first-order scheme.

Step 1: Create Table $T_1 : \{0,1\}^{n-\ell} \rightarrow \{0,1\}^m$

Step 2: Create Table $T_2 : \{0,1\}^{\ell} \rightarrow \{0,1\}^m$

$$T_2(b^{(2)}) := T_1(v^{(1)} \oplus r_{(x_3{}^{(2)} \oplus a^{(2)})}) \oplus \bigoplus_{j \in \{0,1\}^{\ell}, j \neq a^{(2)}} S_{(x_3{}^{(2)} \oplus j)}(x^{(1)} \oplus r_{(x_3{}^{(2)} \oplus a^{(2)})} \oplus r_{(x_3{}^{(2)} \oplus j)})$$

2-O Table Compression Scheme [Vad17]

Three steps *similar* to the first-order scheme.

Step 1: Create Table $T_1 : \{0,1\}^{n-\ell} \to \{0,1\}^m$

Step 2: Create Table $T_2 : \{0,1\}^{\ell} \to \{0,1\}^m$

Step 3: Access Table T_2 to compute the third output share.

$$y_3 = T_2(v^{(2)})$$

Our Attack on 2-O Scheme of [Vad17]

We show that any pair of entries in Table T_2 jointly leak up to $n - \ell$ bits of x.

Lemma

Let $\beta_1, \beta_2 \in \{0,1\}^l$. *Then*

$$T_2(\beta_1) \oplus T_2(\beta_2) = S(\mathbf{x}^{(1)} \mid\mid (\beta_1 \oplus x^{(2)} \oplus v^{(2)})) \oplus S(\mathbf{x}^{(1)} \mid\mid (\beta_2 \oplus x^{(2)} \oplus v^{(2)}))$$

Our Attack on 2-O Scheme of [Vad17]

We show that any pair of entries in Table T_2 jointly leak up to $n - \ell$ bits of x.

Implies when $\ell = 1$ all but one bit of x may leak

Attack does not apply for

- $\ell = 0$
- $\ell = n$
- if output of S only depends on least significant ℓ bits of input

Conclusion

We improved the randomness complexity of 1-O table compression scheme in [Vad17].

- Time & memory complexity essentially remains unchanged.

The new randomness complexity is optimal in an algebraic sense.

Attack on the 2-O table compression scheme in [Vad17].

Open problem: to construct second + higher-order table compression schemes.

Conclusion

We improved the randomness complexity of 1-O table compression scheme in [*Vad17*].

- Time & memory complexity essentially remains unchanged.

The new randomness complexity is optimal in an algebraic sense.

Attack on the 2-O table compression scheme in [*Vad17*].

Open problem: to construct second + higher-order table compression schemes.

Thank You!

&

Questions?

Reference

Srinivas Vivek, **Revisiting a Masked Lookup-Table Compression Scheme**, INDOCRYPT 2017.

Sources of images:

- Internet of Things (on Slide 3): www.bestvpn.com
- Enigma cipher (on S. 4): www.extravaganzi.com
- Evesdropping (on S. 5): www.harekrsna.de, www.dollsofindia.com, www.reddit.com, www.globe-views.com
- SCA attack depiction (on S. 7): www.tau.ac.il
- SCA experiment setup (on S. 9): www.cesca.centers.vt.edu
- SPA attack on RSA (on S. 9): www.eetimes.com
- AES encryption/decryption (on S. 13): www.arduinolab.net