First-Order Side-Channel Attacks on the Permutation Tables Countermeasure

Emmanuel Prouff and Robert McEvoy

Oberthur Technologies, France & University College Cork, Ireland

Side Channel Attacks and Countermeasures

Attacks

- Algorithm Processing leaks information about the manipulated data
- Results in information leakage about secret keys
- Side Channel Analyses (SCA) exploit this leakage:
  (HO-)CPA [BrierClavierOlivier04],
  MIA [GierlichsBatinaTuylsPreneel08],
  Template Attacks [ChariRaoRohatgi02].

Side Channel Attacks and Countermeasures

Attacks

- Algorithm Processing leaks information about the manipulated data
- Results in information leakage about secret keys
- Side Channel Analyses (SCA) exploit this leakage:
  (HO-)CPA [BrierClavierOlivier04],
  MIA [GierlichsBatinaTuylsPreneel08],
  Template Attacks [ChariRaoRohatgi02].

Software Countermeasures

- Shuffling [HerbstOswaldMangard06]: each signal containing information about a sensitive variable is spread over random signals leaking at different times.
- Masking [ChariJultaRaoRohatgi99,GoubinPatarin99]: every sensitive data is modified by a random transformation.

Masking Methods

Masking Methods

- Translation [CJRR99,GP99]: a random value M

$$\tilde{\mathbf{U}} = \mathbf{U} + M \ ,$$

  + efficient to mask linear operation on **U**.
  - data/masked-data relation is very simple.

Masking Methods

- Translation [CJRR99,GP99]: a random value M

$$\tilde{\mathbf{U}} = \mathbf{U} + M \ ,$$

  + efficient to mask linear operation on **U**.
  - data/masked-data relation is very simple.

- Linear Isomorphism [GP99]: a random linear function L

$$\tilde{\mathbf{U}} = L(\mathbf{U}) \ ,$$

  + efficient to mask linear operation on **U**.
  + data/masked-data relation is more complex.
  - flawed (zero is never masked) [FumaroliMayerDubois07].

Masking Methods

- Translation [CJRR99,GP99]: a random value M

$$\tilde{\mathbf{U}} = \mathbf{U} + M \ ,$$

  + efficient to mask linear operation on **U**.
  - data/masked-data relation is very simple.
- Linear Isomorphism [GP99]: a random linear function L

$$\tilde{\mathbf{U}} = L(\mathbf{U}) \ ,$$

  + efficient to mask linear operation on **U**.
  + data/masked-data relation is more complex.
  - flawed (zero is never masked) [FumaroliMayerDubois07].
- Permutation [Coron08]: randomly generate a permutation P

$$\tilde{\mathbf{U}} = P(\mathbf{U}) \ ,$$

  + data/masked-data relation is more complex.
  - less efficient than the two others and flawed [This paper]

The Permutation-Table Countermeasure - 1

The Permutation-Table Countermeasure - 1

Protect a Block Cipher: every intermediate data **U** is presented under the form $P(\mathbf{U})$.

The Permutation-Table Countermeasure - 1

Let **U** and **V** be two 8-bit long (sensitive) data.

The Permutation-Table Countermeasure - 1

Let **U** and **V** be two 8-bit long (sensitive) data.

Question: how to compute the sum $\mathbf{U} \oplus \mathbf{V}$ with the Permutation Masking without manipulating data that depend on **U** and/or **V**?

The Permutation-Table Countermeasure - 1

Let **U** and **V** be two 8-bit long (sensitive) data.

Question: how to compute the sum $\mathbf{U} \oplus \mathbf{V}$ with the Permutation Masking without manipulating data that depend on **U** and/or **V**?

Question [2nd formulation]: how to compute $P(\mathbf{U} \oplus \mathbf{V})$ from $P(\mathbf{U})$ and $P(\mathbf{V})$ without manipulating data that depend on **U** and/or **V**?

The Permutation-Table Countermeasure - 1

Let **U** and **V** be two 8-bit long (sensitive) data.

Question: how to compute the sum $\mathbf{U} \oplus \mathbf{V}$ with the Permutation Masking without manipulating data that depend on **U** and/or **V**?

Question [2nd formulation]: how to compute $P(\mathbf{U} \oplus \mathbf{V})$ from $P(\mathbf{U})$ and $P(\mathbf{V})$ without manipulating data that depend on **U** and/or **V**?

Obvious Answer: at each execution, generate the look-up table of the function $\mathtt{XT}_8(x, y) = P(P^{-1}(x) \oplus P^{-1}(y))$.

Then, to compute $P(\mathbf{U} \oplus \mathbf{V})$ from $P(\mathbf{U})$ and $P(\mathbf{V})$ process

$$\mathtt{XT}_8(P(\mathbf{U}), P(\mathbf{V})) \qquad [= P(\mathbf{U} \oplus \mathbf{V})]$$

The Permutation-Table Countermeasure - 1

Let **U** and **V** be two 8-bit long (sensitive) data.

Question: how to compute the sum $\mathbf{U} \oplus \mathbf{V}$ with the Permutation Masking without manipulating data that depend on **U** and/or **V**?

Question [2nd formulation]: how to compute $P(\mathbf{U} \oplus \mathbf{V})$ from $P(\mathbf{U})$ and $P(\mathbf{V})$ without manipulating data that depend on **U** and/or **V**?

Obvious Answer: at each execution, generate the look-up table of the function $\mathtt{XT}_8(x, y) = P(P^{-1}(x) \oplus P^{-1}(y))$.

Then, to compute $P(\mathbf{U} \oplus \mathbf{V})$ from $P(\mathbf{U})$ and $P(\mathbf{V})$ process

$$\mathtt{XT}_8(P(\mathbf{U}), P(\mathbf{V})) \qquad [= P(\mathbf{U} \oplus \mathbf{V})]$$

- allocation of table of size 2^{16} in RAM

The Permutation-Table Countermeasure - 1

Let $\mathbf{U}$ and $\mathbf{V}$ be two 8-bit long (sensitive) data.

Question: how to compute the sum $\mathbf{U} \oplus \mathbf{V}$ with the Permutation Masking without manipulating data that depend on $\mathbf{U}$ and/or $\mathbf{V}$?

Question [2nd formulation]: how to compute $P(\mathbf{U} \oplus \mathbf{V})$ from $P(\mathbf{U})$ and $P(\mathbf{V})$ without manipulating data that depend on $\mathbf{U}$ and/or $\mathbf{V}$?

More Tricky Answer: generate two random permutations P_1 and P_2 operating on 4-bit data and define P such that $P = P_2 || P_1$

Then ...

The Permutation-Table Countermeasure - 2

- Design two 8-bit to 4-bit look-up tables s.t.:

$$\begin{aligned} \mathtt{XT}_4^1(x||y) &= P_1(P_1^{-1}(x) \oplus P_1^{-1}(y)) \ , \\ \mathtt{XT}_4^2(x||y) &= P_2(P_2^{-1}(x) \oplus P_2^{-1}(y)) \ . \end{aligned}$$

The Permutation-Table Countermeasure - 2

- Design two 8-bit to 4-bit look-up tables s.t.:

$$\begin{aligned}\mathtt{XT}_4^1(x||y) &= P_1(P_1^{-1}(x) \oplus P_1^{-1}(y)) \ , \\ \mathtt{XT}_4^2(x||y) &= P_2(P_2^{-1}(x) \oplus P_2^{-1}(y)) \ .\end{aligned}$$

- See **U** and **V** as two 4-bit data: $\mathbf{U} = U'||U$ and $\mathbf{V} = V'||V$.

The Permutation-Table Countermeasure - 2

- Design two 8-bit to 4-bit look-up tables s.t.:

$$\begin{aligned} \mathtt{XT}_4^1(x||y) &= P_1(P_1^{-1}(x) \oplus P_1^{-1}(y)) \ , \\ \mathtt{XT}_4^2(x||y) &= P_2(P_2^{-1}(x) \oplus P_2^{-1}(y)) \ . \end{aligned}$$

- See $\mathbf{U}$ and $\mathbf{V}$ as two 4-bit data: $\mathbf{U} = U'||U$ and $\mathbf{V} = V'||V$.
- Use the tables to compute $P(\mathbf{U} \oplus \mathbf{V})$ from $P(\mathbf{U})$ and $P(\mathbf{V})$:

The Permutation-Table Countermeasure - 2

- Design two 8-bit to 4-bit look-up tables s.t.:

$$\begin{aligned} \mathtt{XT}_4^1(x||y) &= P_1(P_1^{-1}(x) \oplus P_1^{-1}(y)) \ , \\ \mathtt{XT}_4^2(x||y) &= P_2(P_2^{-1}(x) \oplus P_2^{-1}(y)) \ . \end{aligned}$$

- See **U** and **V** as two 4-bit data: $\mathbf{U} = U'||U$ and $\mathbf{V} = V'||V$.
- Use the tables to compute $P(\mathbf{U} \oplus \mathbf{V})$ from $P(\mathbf{U})$ and $P(\mathbf{V})$:
  1. Compute

$$\mathtt{XT}_4^1(P_1(U)||P_1(V)).$$

It equals $P_1(U \oplus V)$.

The Permutation-Table Countermeasure - 2

- Design two 8-bit to 4-bit look-up tables s.t.:

$$\begin{aligned} \mathtt{XT}_4^1(x||y) &= P_1(P_1^{-1}(x) \oplus P_1^{-1}(y)) \ , \\ \mathtt{XT}_4^2(x||y) &= P_2(P_2^{-1}(x) \oplus P_2^{-1}(y)) \ . \end{aligned}$$

- See $\mathbf{U}$ and $\mathbf{V}$ as two 4-bit data: $\mathbf{U} = U'||U$ and $\mathbf{V} = V'||V$.
- Use the tables to compute $P(\mathbf{U} \oplus \mathbf{V})$ from $P(\mathbf{U})$ and $P(\mathbf{V})$:
  1. Compute
  $$\mathtt{XT}_4^1(P_1(U)||P_1(V)).$$
  It equals $P_1(U \oplus V)$.
  2. Compute
  $$\mathtt{XT}_4^2(P_2(U')||P_2(V')).$$
  It equals $P_2(U' \oplus V')$.

The Permutation-Table Countermeasure - 2

- Design two 8-bit to 4-bit look-up tables s.t.:

$$\begin{aligned} \mathtt{XT}_4^1(x||y) &= P_1(P_1^{-1}(x) \oplus P_1^{-1}(y)) \ , \\ \mathtt{XT}_4^2(x||y) &= P_2(P_2^{-1}(x) \oplus P_2^{-1}(y)) \ . \end{aligned}$$

- See $\mathbf{U}$ and $\mathbf{V}$ as two 4-bit data: $\mathbf{U} = U'||U$ and $\mathbf{V} = V'||V$.
- Use the tables to compute $P(\mathbf{U} \oplus \mathbf{V})$ from $P(\mathbf{U})$ and $P(\mathbf{V})$:
  1. Compute
     $$\mathtt{XT}_4^1(P_1(U)||P_1(V)).$$
     It equals $P_1(U \oplus V)$.
  2. Compute
     $$\mathtt{XT}_4^2(P_2(U')||P_2(V')).$$
     It equals $P_2(U' \oplus V')$.
  3. Concatenate
     $$P_2(U' \oplus V')||P_1(U \oplus V).$$
     We get $P(\mathbf{U} \oplus \mathbf{V})$.

The Flaw

Focus on the computation $P_1(U \oplus V) = \mathtt{XT}_4^1(P_1(U)||P_1(V))$.

The Flaw

Focus on the computation $P_1(U \oplus V) = \mathtt{XT}_4^1(P_1(U)||P_1(V))$.

Pseudo code:

1. Store $P_1(U)||P_1(V)$ into register R.
2. Load $\mathtt{XT}_4^1[\mathrm{R}]$ into output register R'.
3. Output R'

The Flaw

Focus on the computation $P_1(U \oplus V) = \mathtt{XT}_4^1(P_1(U)||P_1(V))$.

Pseudo code:

1. Store $P_1(U)||P_1(V)$ into register R.
2. Load $\mathtt{XT}_4^1[\mathrm{R}]$ into output register R'.
3. Output R'

Observation: data $U||V$ is manipulated under the form $P_1(U)||P_1(V)$ and not $P(U||V)$

The Flaw

Focus on the computation $P_1(U \oplus V) = \mathtt{XT}_4^1(P_1(U)||P_1(V))$.

Pseudo code:

1. Store $P_1(U)||P_1(V)$ into register R.
2. Load $\mathtt{XT}_4^1[\mathrm{R}]$ into output register R'.
3. Output R'

Observation: data $U||V$ is manipulated under the form $P_1(U)||P_1(V)$ and not $P(U||V)$

Flaw: since the same random permutation P_1 is applied to U and V, variable $P_1(U)||P_1(V)$ statistically depends on $U||V$.

The Flaw

Focus on the computation $P_1(U \oplus V) = \mathtt{XT}_4^1(P_1(U)||P_1(V))$.

Pseudo code:

1. Store $P_1(U)||P_1(V)$ into register R.
2. Load $\mathtt{XT}_4^1[\mathrm{R}]$ into output register R'.
3. Output R'

Observation: data $U||V$ is manipulated under the form $P_1(U)||P_1(V)$ and not $P(U||V)$

Flaw: since the same random permutation P_1 is applied to U and V, variable $P_1(U)||P_1(V)$ statistically depends on $U||V$.

For instance: if U equals V then $P_1(U)$ equals $P_1(V)$.

CPA Attack

CPA Attack

- We assume that $U||V$ is a guessable key-dependent random variable.

CPA Attack

- We assume that $U||V$ is a guessable key-dependent random variable.
- Due to the Flaw, this implies that $P_1(U)||P_1(V)$ is a key-dependent random variable.

CPA Attack

- We assume that $U||V$ is a guessable key-dependent random variable.
- Due to the Flaw, this implies that $P_1(U)||P_1(V)$ is a key-dependent random variable.

The CPA

CPA Attack

- We assume that $U||V$ is a guessable key-dependent random variable.
- Due to the Flaw, this implies that $P_1(U)||P_1(V)$ is a key-dependent random variable.

The CPA

- Let L be the leakage on $P_1(U)||P_1(V)$:

$$L = \phi(P_1(U)||P_1(V)) + \text{Noise} \ .$$

CPA Attack

- We assume that $U||V$ is a guessable key-dependent random variable.
- Due to the Flaw, this implies that $P_1(U)||P_1(V)$ is a key-dependent random variable.

The CPA

- Let L be the leakage on $P_1(U)||P_1(V)$:

$$L = \phi(P_1(U)||P_1(V)) + \text{Noise} \quad .$$

- Let $\hat{\phi}$ be a consumption model.

CPA Attack

- We assume that $U||V$ is a guessable key-dependent random variable.
- Due to the Flaw, this implies that $P_1(U)||P_1(V)$ is a key-dependent random variable.

The CPA

- Let L be the leakage on $P_1(U)||P_1(V)$:

$$L = \phi(P_1(U)||P_1(V)) + \text{Noise} \quad .$$

- Let $\hat{\phi}$ be a consumption model.
- For every key hypothesis compute a prediction $\hat{U}||\hat{V}$ on $U||V$ and estimate:

$$\rho(L, \hat{\phi}(\hat{U}||\hat{V})) \quad .$$

Pre-processing(s)

Pre-processing(s)

Result: depending on the nature of ϕ the attack sometimes fails!

Pre-processing(s)

Result: depending on the nature of ϕ the attack sometimes fails!
In fact, it fails iff there exist two functions ϕ_1 and ϕ_2 such that

$$\phi(P_1(U)||P_1(V)) = \phi_1(P_1(U)) + \phi_2(P_1(V)) \ .$$

Pre-processing(s)

Result: depending on the nature of ϕ the attack sometimes fails!
In fact, it fails iff there exist two functions ϕ_1 and ϕ_2 such that

$$\phi(P_1(U)||P_1(V)) = \phi_1(P_1(U)) + \phi_2(P_1(V)) \ .$$

Example: for $\phi = \mathrm{HW}$ then we have

$$\mathrm{HW}(P_1(U)||P_1(V)) = \mathrm{HW}(P_1(U)) + \mathrm{HW}(P_1(V)) \ .$$

Pre-processing(s)

Result: depending on the nature of ϕ the attack sometimes fails!

In fact, it fails iff there exist two functions ϕ_1 and ϕ_2 such that

$$\phi(P_1(U)||P_1(V)) = \phi_1(P_1(U)) + \phi_2(P_1(V)) \ .$$

Example: for $\phi = \mathrm{HW}$ then we have

$$\mathrm{HW}(P_1(U)||P_1(V)) = \mathrm{HW}(P_1(U)) + \mathrm{HW}(P_1(V)) \ .$$

Why? Because in this case the mean of $\phi(P_1(U)||P_1(V))$ does not depend on $U||V$.

Pre-processing(s)

Result: depending on the nature of ϕ the attack sometimes fails!
In fact, it fails iff there exist two functions ϕ_1 and ϕ_2 such that

$$\phi(P_1(U)||P_1(V)) = \phi_1(P_1(U)) + \phi_2(P_1(V)) \ .$$

Example: for $\phi = \mathrm{HW}$ then we have

$$\mathrm{HW}(P_1(U)||P_1(V)) = \mathrm{HW}(P_1(U)) + \mathrm{HW}(P_1(V)) \ .$$

Why? Because in this case the mean of $\phi(P_1(U)||P_1(V))$ does not depend on $U||V$.
What to do? Focus on higher order statistical central moments, *e.g.* the central moments of order 2:

$$\rho((L-\mathrm{E}\,[L])^2, f(\hat{U}||\hat{V})) \ ,$$

where f is a well-chosen function.

Define a Sound Prediction Function

Define a Sound Prediction Function

Example of choice for f: choose $f = \phi$ or $f = \phi^2$.

Define a Sound Prediction Function

Example of choice for f: choose $f = \phi$ or $f = \phi^2$.

Result: does not work!

Define a Sound Prediction Function

Example of choice for f: choose $f = \phi$ or $f = \phi^2$.

Result: does not work!

Surprising! attack works if P_1 is a simple translation instead of a permutation [WaddleWagner04].

Define a Sound Prediction Function

Example of choice for f: choose $f = \phi$ or $f = \phi^2$.

Result: does not work!

Surprising! attack works if P_1 is a simple translation instead of a permutation [WaddleWagner04].

Explanation: relation between $U||V$ and $P_1(U)||P_1(V)$ is much more complex than for classical masking by translation.

Define a Sound Prediction Function

Example of choice for f: choose $f = \phi$ or $f = \phi^2$.
Result: does not work!
Surprising! attack works if P_1 is a simple translation instead of a permutation [WaddleWagner04].
Explanation: relation between $U||V$ and $P_1(U)||P_1(V)$ is much more complex than for classical masking by translation.
Proposal: use the function $f(\hat{U}||\hat{V})) = \delta_{\hat{U}}(\hat{V})$ defined by $\delta_{\hat{U}}(\hat{V})$ equals 1 if $\hat{U} = \hat{V}$ and equals 0 otherwise.

Define a Sound Prediction Function

Example of choice for f: choose $f = \phi$ or $f = \phi^2$.

Result: does not work!

Surprising! attack works if P_1 is a simple translation instead of a permutation [WaddleWagner04].

Explanation: relation between $U||V$ and $P_1(U)||P_1(V)$ is much more complex than for classical masking by translation.

Proposal: use the function $f(\hat{U}||\hat{V})) = \delta_{\hat{U}}(\hat{V})$ defined by $\delta_{\hat{U}}(\hat{V})$ equals 1 if $\hat{U} = \hat{V}$ and equals 0 otherwise.

Proved to be an optimal choice in the Gaussian Model with $\phi = \mathrm{HW}$. It corresponds to an estimation of the function

$$\hat{u}, \hat{v} \mapsto \mathrm{E}\left[(L - \mathrm{E}[L])^2 | \hat{U} = \hat{u}, \hat{V} = \hat{v}\right] \quad ,$$

(proved to be the optimal choice in [ProuffRivain09]).

Define a Sound Prediction Function

Example of choice for f: choose $f = \phi$ or $f = \phi^2$.
Result: does not work!
Surprising! attack works if P_1 is a simple translation instead of a permutation [WaddleWagner04].
Explanation: relation between $U||V$ and $P_1(U)||P_1(V)$ is much more complex than for classical masking by translation.
Proposal: use the function $f(\hat{U}||\hat{V})) = \delta_{\hat{U}}(\hat{V})$ defined by $\delta_{\hat{U}}(\hat{V})$ equals 1 if $\hat{U} = \hat{V}$ and equals 0 otherwise.
Proved to be an optimal choice in the Gaussian Model with $\phi = \mathrm{HW}$. It corresponds to an estimation of the function

$$\hat{u}, \hat{v} \mapsto \mathrm{E}\left[(L - \mathrm{E}[L])^2 | \hat{U} = \hat{u}, \hat{V} = \hat{v}\right] \quad ,$$

(proved to be the optimal choice in [ProuffRivain09]).
Alternative (more complex) functions are proposed in more general models.

Application of the attack

Application of the attack

- AES implementation protected with Permutation Countermeasure.

Application of the attack

- AES implementation protected with Permutation Countermeasure.
- Two scenarios

Application of the attack

- AES implementation protected with Permutation Countermeasure.
- Two scenarios

1. During the first AddRoundKey operation:

$$U = X_l \text{ and } V = K_l \ ,$$

where X_l is a plaintext nibble and K_l is key nibble.
Goal: retrieve K_l.

Application of the attack

- AES implementation protected with Permutation Countermeasure.
- Two scenarios

1. During the first AddRoundKey operation:

$$U = X_l \text{ and } V = K_l \ ,$$

where X_l is a plaintext nibble and K_l is key nibble.
Goal: retrieve K_l.

2. During the first MixColumn operation:

$$U = S_l[X \oplus K] \text{ and } V = S_l[X' \oplus K'] \ ,$$

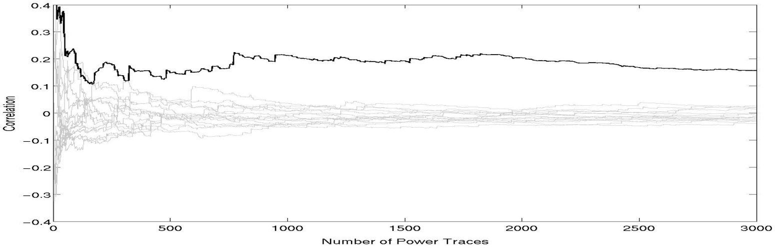
where (X, X') is a pair of plaintext bytes, (K, K') is a pair of key bytes and S_l corresponds to the 4 lowest bits of the AES Sbox.
Goal: retrieve (K, K').

First Scenario

Simulations

Noise std	0	0.5	5	7	10
Nb of measurements	100	1,000	60,000	230,000	900,000

Experiments

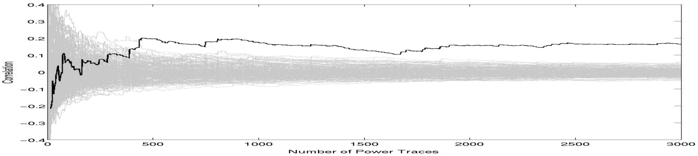


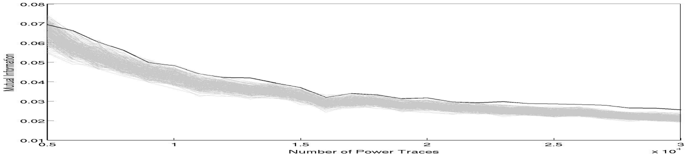
Second Scenario

Simulations

Noise standard deviation	0	0.5	1	2	5	7	10
Nb of measurements [MIA with Kernel]	2,500	20,000	60,000	290,000	$> 10^6$	$> 10^6$	$> 10^6$
Nb of measurements [Parametric MIA]	na	3,000	4,000	25,000	250,000	500,000	800,000
Nb of measurements [CPA with f_{opt}]	1,000	1,000	1,500	6,500	120,000	550,000	$> 10^6$

Experiments





Conclusion

- A first-order flaw exists in the permutation tables countermeasure proposed in [C08].
- To exploit this leakage, 2 attacks have been developed in different scenarii: CPA and MIA.
- Attacks have been verified in both simulation and practice.
- A patch for the permutation tables countermeasure is proposed in the extended version of this paper.
- Even if the permutation tables countermeasure is flawed, exploiting this flaw requires more traces than an attack on a flawed masking scheme: when patched this masking must therefore be a good alternative against HO-SCA.

Thank you!
Questions and/or Comments?