Side-channel attacks on PKC and countermeasures

Lejla Batina

Institute for Computing and Information Sciences – Digital Security
Radboud University Nijmegen

10th June 2016

Radboud University Nijmegen

Outline

What SPA adversary can

- Learn some info on instructions/data being processed

What SPA adversary can

- Learn some info on instructions/data being processed
- Sometimes even recover the key from one (or a few traces)

What SPA adversary can

- Learn some info on instructions/data being processed
- Sometimes even recover the key from one (or a few traces)
- Exploit new attack techniques

What SPA adversary can

- Learn some info on instructions/data being processed
- Sometimes even recover the key from one (or a few traces)
- Exploit new attack techniques
  - Online Template Attacks

What SPA adversary can

- Learn some info on instructions/data being processed
- Sometimes even recover the key from one (or a few traces)
- Exploit new attack techniques
  - Online Template Attacks
  - Combine side-channel leakages with some other (theoretical) attacks
- Challenge: recent horizontal attacks belong to SPA techniques

What SPA adversary can

- Learn some info on instructions/data being processed
- Sometimes even recover the key from one (or a few traces)
- Exploit new attack techniques
  - Online Template Attacks
  - Combine side-channel leakages with some other (theoretical) attacks
- Challenge: recent horizontal attacks belong to SPA techniques
- Can defeat some countermeasures such as

What SPA adversary can

- Learn some info on instructions/data being processed
- Sometimes even recover the key from one (or a few traces)
- Exploit new attack techniques
  - Online Template Attacks
  - Combine side-channel leakages with some other (theoretical) attacks
- Challenge: recent horizontal attacks belong to SPA techniques
- Can defeat some countermeasures such as
  $\rightarrow$ scalar randomization

Basic algorithm for modular exponentiation

Square-and-multiply algorithm

Input: x, $d = (d_{t-1}, d_{t-2}, ..., k_0)_2$
Output: $y = x^d \mod N$

1: $R_0 \leftarrow 1$
2: **for** $i = t - 1$ downto 0 **do**
3: $\quad R_0 \leftarrow R_0^2 \mod N$
4: $\quad$ If $i = 1, R_0 \leftarrow R_0 \cdot x \mod N$
5: **end for**
6: **return** R_0

SPA-resistant modular exponentiation

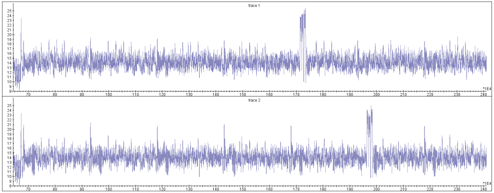
Square-and-multiply always

Input: x, $d = (d_{t-1}, d_{t-2}, ..., k_0)_2$
Output: $y = x^d \mod N$

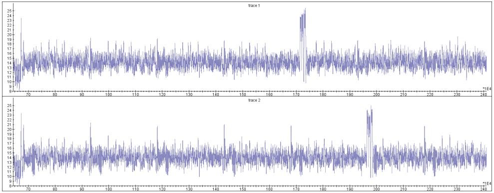
1: $R_0 \leftarrow 1, R_1 \leftarrow 1, R_2 \leftarrow x$
2: **for** $i = t - 1$ downto 0 **do**
3: $\quad R_0 \leftarrow R_0^2 \mod N$
4: $\quad b \leftarrow 1 - d_i; R_b \leftarrow R_b \cdot R_2 \mod N$
5: **end for**
6: **return** R_0

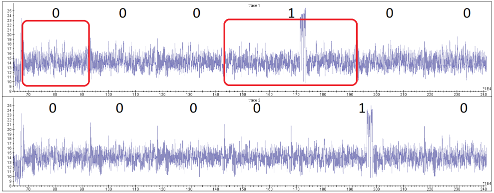
When $d_i = 0$ there is a dummy multiplication!

SPA on ECC double-and-add



SPA on ECC double-and-add





Slide credit: L. Chmielewski.

DPA-resistant modular exponentiation

Randomizing message

Input: $m, d, N,$
Output: $c = m^d \mod N$

1: $r = Random()$
2: $m_s \leftarrow rm$
3: $v \leftarrow {m_s}^d \mod N$
4: $u \leftarrow r^d \mod N$
5: $c \leftarrow \frac{v}{u} \mod N$
6: **return** c

DPA-resistant modular exponentiation

Randomizing exponent

Input: $m, d, N, \phi(N)$,
Output: $c = m^d \mod N$

1: $r = Random()$
2: $d' \leftarrow d + r\phi(N)$
3: $c \leftarrow m^{d'} \mod N$
4: **return** c

ECDLP and scalar multiplication

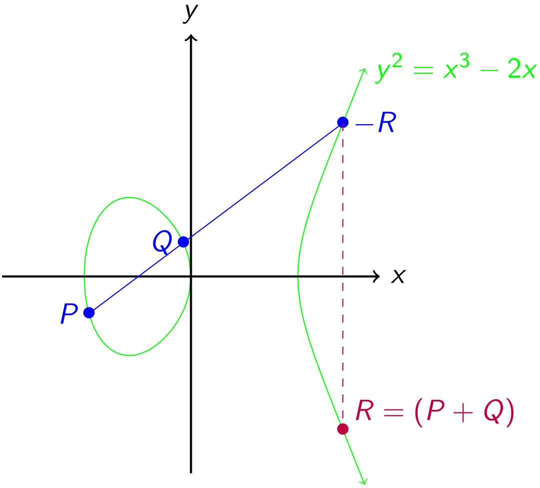
ECDLP

Let E be an elliptic curve over a finite field $\mathbb{F}_q$, $G =< P >$ a cyclic subgroup of $E(\mathbb{F}_q)$ and $Q \in G$. ECDLP is the problem of finding $k \in \mathbb{Z}$ such that $Q = kP$

The scalar multiplication kP is the crucial computation in ECC.

$$kP = \underbrace{P + P + ... + P}_{k\text{-times}}$$

Addition rule for Weierstrass equation: $E : y^2 = x^3 - 2x$



Attacks on ECC

- Simple SPA attacks can be counteracted by a balanced scalar multiplication algorithm e.g. double-and-add always.

Attacks on ECC

- Simple SPA attacks can be counteracted by a balanced scalar multiplication algorithm e.g. double-and-add always.
- The choice of attack varies for different protocols e.g. the protocol determines scenario.

Attacks on ECC

- Simple SPA attacks can be counteracted by a balanced scalar multiplication algorithm e.g. double-and-add always.
- The choice of attack varies for different protocols e.g. the protocol determines scenario.
  $\rightarrow$ Example: Attacks on ECDSA are attacks on modular multiplication or on scalar multiplication

ECDSA: Signature generation

Key generation:

- Alice chooses $E(\mathbb{F}_q)$ and a point $G \in E(\mathbb{F}_q)$ ($ord(G) = r$ is a large prime)
- Alice also chooses a secret, random integer a and computes $Q = aG$
- Alice's public info is $(\mathbb{F}_q, E(\mathbb{F}_q), r, G, Q)$ and she keeps a private.

ECDSA: Signature generation

Key generation:

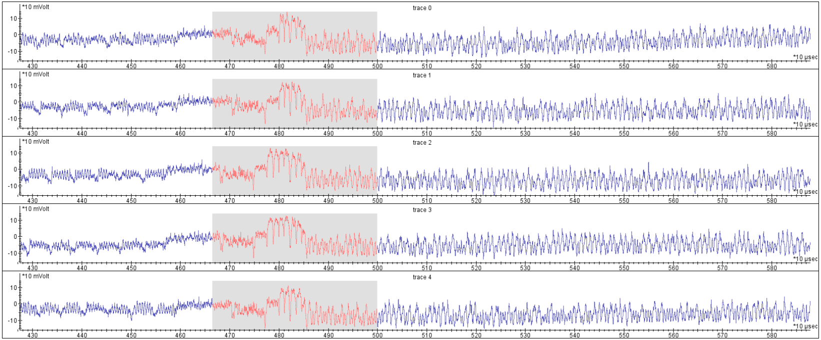
- Alice chooses $E(\mathbb{F}_q)$ and a point $G \in E(\mathbb{F}_q)$ ($ord(G) = r$ is a large prime)
- Alice also chooses a secret, random integer a and computes $Q = aG$
- Alice's public info is $(\mathbb{F}_q, E(\mathbb{F}_q), r, G, Q)$ and she keeps a private.

Signature generation: To sign a doc. m, Alice does the following:

- Choose a random integer k, $1 \leq k < r$ and compute $R = kG = (x, y)$
- Compute $s \equiv k^{-1}(m + ax) \mod r$
- The signature is (m, R, s).

SPA on ECC scalar multiplication

5 traces of the first round of Lim-Lee algorithm. Pattern: 11001



Slide credit: L. Chmielewski.

Attacking Schnorr identification protocol

Table 1. Schnorr Identification Protocol

Prover		Verifier
$r \in_R \mathbb{Z}_n$		
$X \leftarrow rP$	$\xrightarrow{X}$	
	$\xleftarrow{e}$	$e \in_R Z_{2^t}$
$y = ae + r$	$\xrightarrow{y}$	
		if $yP + eZ = X$ then accept

What are the attack points in this protocol?

Attacking Schnorr identification protocol

Table 1. Schnorr Identification Protocol

Prover		Verifier
$r \in_R \mathbb{Z}_n$		
$X \leftarrow rP$	$\xrightarrow{X}$	
	$\xleftarrow{e}$	$e \in_R Z_{2^t}$
$y = ae + r$	$\xrightarrow{y}$	
		if $yP + eZ = X$ then accept

What are the attack points in this protocol?

- SPA on rP might reveal r. But, is knowing r useful?

Attacking Schnorr identification protocol

Table 1. Schnorr Identification Protocol

Prover		Verifier
$r \in_R \mathbb{Z}_n$		
$X \leftarrow rP$	$\xrightarrow{X}$	
	$\xleftarrow{e}$	$e \in_R Z_{2^t}$
$y = ae + r$	$\xrightarrow{y}$	
		if $yP + eZ = X$ then accept

What are the attack points in this protocol?

- SPA on rP might reveal r. But, is knowing r useful?
  Yes, if r is known, compute $a = (y - r)e^{-1}$

Attacking Schnorr identification protocol

Table 1. Schnorr Identification Protocol

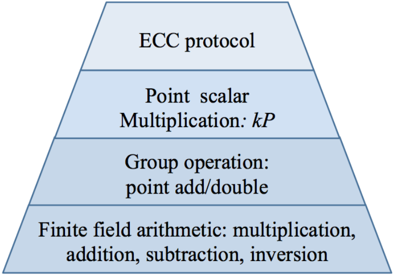
Prover		Verifier
$r \in_R \mathbb{Z}_n$		
$X \leftarrow rP$	$\xrightarrow{X}$	
	$\xleftarrow{e}$	$e \in_R Z_{2^t}$
$y = ae + r$	$\xrightarrow{y}$	
		if $yP + eZ = X$ then accept

What are the attack points in this protocol?

- SPA on rP might reveal r. But, is knowing r useful? Yes, if r is known, compute $a = (y - r)e^{-1}$
- Another option: DPA on $a \cdot e$

What do we want from countermeasures?

- Countermeasures can be applied on all levels of the hierarchy
- One should make sure that leaked information is useless



ECC countermeasures

- Protocol level: leakage-aware protocol design i.e. limit the number of times the key is used

ECC countermeasures

- Protocol level: leakage-aware protocol design i.e. limit the number of times the key is used
- Scalar-mult level:
  - Random scalar-splitting, randomizing scalar and points

ECC countermeasures

- Protocol level: leakage-aware protocol design i.e. limit the number of times the key is used
- Scalar-mult level:
  - Random scalar-splitting, randomizing scalar and points
  - Special scalar-indistinguishable group operations: double-and-add alway, add-always, Montgomery ladder

ECC countermeasures

- Protocol level: leakage-aware protocol design i.e. limit the number of times the key is used
- Scalar-mult level:
  - Random scalar-splitting, randomizing scalar and points
  - Special scalar-indistinguishable group operations: double-and-add alway, add-always, Montgomery ladder
- Randomize intermediate results: projective coordinates, curve isomorphisms

ECC countermeasures

- Protocol level: leakage-aware protocol design i.e. limit the number of times the key is used
- Scalar-mult level:
    - Random scalar-splitting, randomizing scalar and points
    - Special scalar-indistinguishable group operations: double-and-add alway, add-always, Montgomery ladder
- Randomize intermediate results: projective coordinates, curve isomorphisms
- ECC-non-specific: secure hardware, randomization in the field

ECC countermeasures

- Protocol level: leakage-aware protocol design i.e. limit the number of times the key is used
- Scalar-mult level:
  - Random scalar-splitting, randomizing scalar and points
  - Special scalar-indistinguishable group operations: double-and-add alway, add-always, Montgomery ladder
- Randomize intermediate results: projective coordinates, curve isomorphisms
- ECC-non-specific: secure hardware, randomization in the field

Template Attacks

- Combination of statistical modeling and power-analysis attacks
- Similar ideas are used in detection and estimation theory
- Template attacks consist of two stages:
    - Template-Building Phase (profiling the unprotected device to create the templates)
    - Template-Matching Phase (use the templates for secret data recovery)

Reduced Templates - Side Channels

- Side-channel analysis focuses on modelling traces

Reduced Templates - Side Channels

- Side-channel analysis focuses on modelling traces
- We model measurements through a random variable V

Reduced Templates - Side Channels

- Side-channel analysis focuses on modelling traces
- We model measurements through a random variable V
- A *realization* of V is a measurement v, e.g. $v = 5.1$ Volts

Reduced Templates - Side Channels

- Side-channel analysis focuses on modelling traces
- We model measurements through a random variable V
- A *realization* of V is a measurement v, e.g. $v = 5.1$ Volts
- A trace consists of many measured samples

Reduced Templates - Side Channels

- Side-channel analysis focuses on modelling traces
- We model measurements through a random variable V
- A *realization* of V is a measurement v, e.g. $v = 5.1$ Volts
- A trace consists of many measured samples
- We represent it as a vector of random variables, also called a random vector $\mathbf{L} = [L^1, L^2, \ldots, L^{no\_samples}]$

Reduced Templates - Side Channels

- Side-channel analysis focuses on modelling traces
- We model measurements through a random variable V
- A *realization* of V is a measurement v, e.g. $v = 5.1$ Volts
- A trace consists of many measured samples
- We represent it as a vector of random variables, also called a random vector $\mathbf{L} = [L^1, L^2, \ldots, L^{no\_samples}]$
- A *realization* of the random vector $\mathbf{L}$ is a trace $\mathbf{l}$ e.g. $\mathbf{l} = [0.41, 0.10, 0.12, 0.17, 0.36]$

Reduced Templates - Template Building

Template Building:

1. Force the cryptographic device to encrypt n times with key $K = key_0$
2. Measure n traces $\mathbf{l}_i$ with $K = key_0$
3. The template for $\mathbf{T_{key_0}} = (\mathbf{L}|K = key_0)$ is the mean vector ${}^{key_0}\overline{\mathbf{l}} = 1/n * \sum_{i=1}^{n} {}^{key_0}\mathbf{l}_i$
4. Similarly, force the cryptographic device to encrypt n times with $K = key_1$
5. Measure n traces with $Key = key_1$
6. The template for $\mathbf{T_{key_1}} = (\mathbf{L}|K = key_1)$ is the mean vector ${}^{key_1}\overline{\mathbf{l}} = 1/n * \sum_{i=1}^{n} {}^{key_1}\mathbf{l}_i$

Reduced Templates - Template Matching

Template Matching:

1. Observe a trace $\mathbf{t}$
2. Match the trace $\mathbf{t}$ to templates T_{key_0} and T_{key_1}
3. Use the matching score to decide the key used by the trace $\mathbf{t}$
4. $score_0 = (\mathbf{t} -^{key_0} \bar{\mathbf{I}}) * (\mathbf{t} -^{key_0} \bar{\mathbf{I}})^T$
   $score_1 = (\mathbf{t} -^{key_1} \bar{\mathbf{I}}) * (\mathbf{t} -^{key_1} \bar{\mathbf{I}})^T$
5. Decide key_0 or key_1

Template Attacks for PKC

- Messerges, Dabbish, Sloan [1999]
  - MESD attack requires the attacker to run about 200 trial exponentiations for each bit of the secret exponent
- Medwed and Oswald [2008]
  - Template-based SPA attack on ECDSA (attacking scalar multiplication)
  - 33 templates used (device leaking Hamming weight)
  - Template-traces for 50 intermediate values per key-bit required for successful template matching

Other SPA-like attacks

- Collision attacks

Other SPA-like attacks

- Collision attacks
  $\rightarrow$ when processing the same data, the same computations will result in the similar patterns (in SCA measurements)

Other SPA-like attacks

- Collision attacks
  $\rightarrow$ when processing the same data, the same computations will result in the similar patterns (in SCA measurements)
- Horizontal attacks

Other SPA-like attacks

- Collision attacks
  → when processing the same data, the same computations will result in the similar patterns (in SCA measurements)
- Horizontal attacks
  → considering similar computations/data in horizontal direction (require a single power trace)

Other SPA-like attacks

- Collision attacks
  → when processing the same data, the same computations will result in the similar patterns (in SCA measurements)
- Horizontal attacks
  → considering similar computations/data in horizontal direction (require a single power trace)
- Online Template Attacks

Other SPA-like attacks

- Collision attacks
  → when processing the same data, the same computations will result in the similar patterns (in SCA measurements)
- Horizontal attacks
  → considering similar computations/data in horizontal direction (require a single power trace)
- Online Template Attacks
  → use ideas from horizontal and template attacks

Main ideas behind Online Template Attacks

- OTA: One full target trace and one template trace per key-bit are enough to recover the secret scalar.
- Focus on key dependent assignments within scalar multiplication.
- A variant of multiple-shot SPA, combining techniques from horizontal-collision and template attacks.

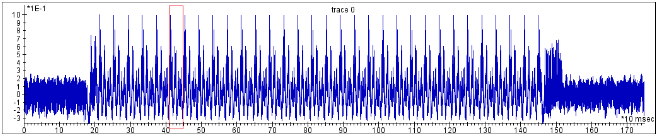


Figure: Target trace: 32 rounds of scalar multiplication for Edwards curve

Attack assumptions

1. The attacker obtains only *1 target trace*. He may obtain several *template traces per key-bit*.
   (For PA: 1 template trace, for EM: 10 template traces)
2. Template traces are generated *after* obtaining the target trace, i.e. "online" or "on-the-fly".
3. Template traces are obtained on the target device or a similar device with *limited control* over it.
4. The attacker can change input points in the similar device.
5. No branches in algorithm, but at least *one key-dependent assignment*.

Attack methodology: 1.Profiling of the device

- Acquire a full target trace during execution of scalar multiplication.
- Locate the doubling and addition performed at each round.
- Find multiples mP of the input point P.

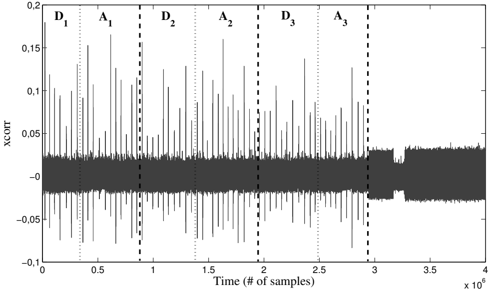


Figure: Distinguishing Doublings and Addings

Attack methodology: 2.Template Matching

- Obtain template traces with mP, m is chosen according to the algorithm used in the target device.
- Correlate the output of $(i+1)$-iteration of target trace with input of i-iteration of template trace for each scalar bit (for unblinded scalar).

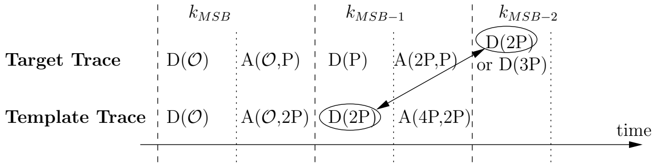


Figure: Correlation of $(i+1)$-iteration of target with i-iteration of template

Advantages of the technique

- No cumbersome pre-processing template building

Advantages of the technique

- No cumbersome pre-processing template building
- No previous knowledge of the leakage model

Advantages of the technique

- No cumbersome pre-processing template building
- No previous knowledge of the leakage model
- Works against scalar randomization and changing point representation

Advantages of the technique

- No cumbersome pre-processing template building
- No previous knowledge of the leakage model
- Works against scalar randomization and changing point representation
- Works against SPA and some DPA protected implementations

Advantages of the technique

- No cumbersome pre-processing template building
- No previous knowledge of the leakage model
- Works against scalar randomization and changing point representation
- Works against SPA and some DPA protected implementations
- Applicable to Montgomery ladder and constant-time implementations

Advantages of the technique

- No cumbersome pre-processing template building
- No previous knowledge of the leakage model
- Works against scalar randomization and changing point representation
- Works against SPA and some DPA protected implementations
- Applicable to Montgomery ladder and constant-time implementations
- Experimentally confirmed on the twisted Edwards curve used in Ed25519 signature scheme, Brainpool BP256r1 curve and NIST SecP256r1 curve

OTA on double-and-add-always

Optimized double-add-always on twisted Edwards curve

Input: P,
$k = (k_{x-1}, k_{x-2}, ..., k_0)_2$
Output: $Q = kP$

$R_0 \leftarrow P$
for $i = x - 2$ downto 0 **do**
  $R_0 \leftarrow 2R_0$
  $R_1 \leftarrow R_0 + P$
  $R_0 \leftarrow R_{k_i}$
end for
return R_0

$k = 100$
$R_0 = P$
$R_0 = 2P, R_1 = 3P$, return $2P$
$R_0 = 4P, R_1 = 5P$, return $4P$
$k = 110$
$R_0 = P$
$R_0 = 2P, R_1 = 3P$, return $3P$
$R_0 = 6P, R_1 = 7P$, return $6P$

OTA on Montgomery Ladder

Montgomery ladder
on twisted Edwards curve

Input: P,
$k = (k_{x-1}, k_{x-2}, ..., k_0)_2$
Output: $Q = k \cdot P$

$R_0 \leftarrow P$
$R_1 \leftarrow 2 \cdot P$
for $i = x - 2$ downto 0 **do**
  $b = 1 - k_i$
  $R_b = R_0 + R_1$
  $R_{k_i} = 2 \cdot R_{k_i}$
end for
return R_0

$k = 100$	
$R_0 = P, R_1 = 2P$	
$b = 1$	$R_1 = 3P, R_0 = 2P$, return $2P$
$b = 1$	$R_1 = 5P, R_0 = 4P$, return $4P$
$k = 110$	
$R_0 = P, R_1 = 2P$	
$b = 0$	$R_0 = 3P, R_1 = 4P$, return $3P$
$b = 1$	$R_1 = 7P, R_0 = 6P$, return $6P$

Setup

- ATMega163 microcontroller
- NaCl implementation of twisted Edwards curve with unified formulas
- $\mathcal{E}_p : x^2 + y^2 = 1 + dx^2y^2$, with $d = -(121665/121666)$, $p = 2^{255} - 19$
- High security level (at least 128−bits of security) and constant time implementation

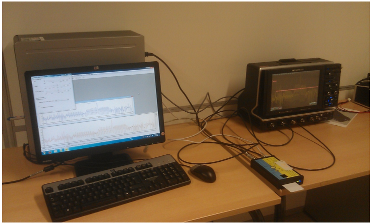


Figure: Our lab setup

OTA on twisted Edwards curve with Power Analysis

- Choose input point $P = \{P_x, P_y, P_z, P_t\}$ for the target trace.
- Compute $2P$ or $3P$ with the same addition formulas.
- Correct bit assumptions have $84 - 88\%$ matching patterns, wrong bit assumptions drops to $50 - 72\%$. Pattern matching threshold: 80%.

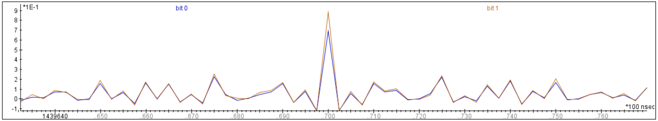


Figure: Pattern match of P on card 1 to $2P$ on card 2 (blue) and to $3P$ on card 2 (brown) for MSB of scalar 1100

[L. Batina, L. Chmielewski, L. Papachristodoulou, P. Schwabe and M. Tunstall. Online Template Attacks. In INDOCRYPT 2014 - 15th International Conference on Cryptology in India, pages 21-36, 2014.]

Acquisition Setup with EM Analysis

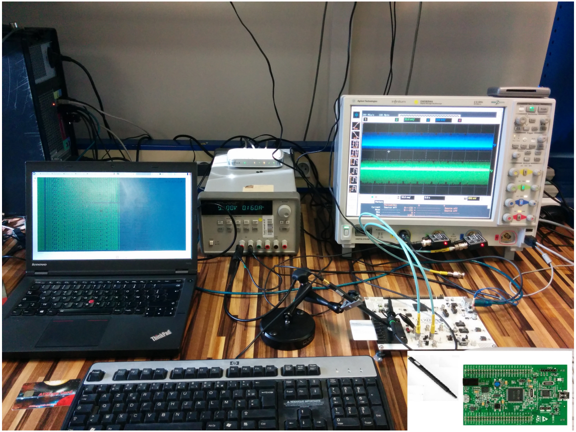


Figure: SCA equipment at ParisTech, Acquisition with smart-card or STM32f4 platform

Doubling Formulas for point $P = (X, Y, Z)$

$www.hyperelliptic.org/EFD/g1p/auto-shortw-jacobian.html$:

PolarSSL v1.3.7

$$
\begin{array}{lllll}
D_1 & \leftarrow & \mathrm{X} & \times & X \mod p \\
D_2 & \leftarrow & \mathrm{Y} & \times & Y \mod p \\
D_3 & \leftarrow & D_2 & \times & D_2 \mod p \\
D_4 & \leftarrow & \mathrm{Z} & \times & Z \mod p \\
 & & \vdots & &
\end{array}
$$

mbedTLS v2.2.0

$$
\begin{array}{lllll}
D_1 & \leftarrow & \mathrm{X} & \times & X \mod p \\
D_2 & \leftarrow & \mathrm{Y} & \times & Y \mod p \\
D_3 & \leftarrow & \mathrm{Z} & \times & Z \mod p \\
D_4 & \leftarrow & 4\mathrm{X} & \times & D_2 \mod p \\
 & & \vdots & &
\end{array}
$$

Finite Field Multiplication in mbedTLS

Input: A and $B_7..B_0$ two elements of 256-bits long.
Output: $X = A \times B$

$X \leftarrow 0$
for i from 7 down to 0 **do**
    $(C, X_{i+7}, X_{i+6}, \ldots, X_i) \leftarrow (X_{i+7}, \ldots, X_i) + A \times B_i$
    $j \leftarrow i + 8$
    repeat
        $(C, X_j) \leftarrow X_j + C$
        $j \leftarrow j + 1$
    until $C \neq 0$
end for
return X

Brainpool curve in mbedTLS

Multiplication of two 32-bit words in mbedTLS.

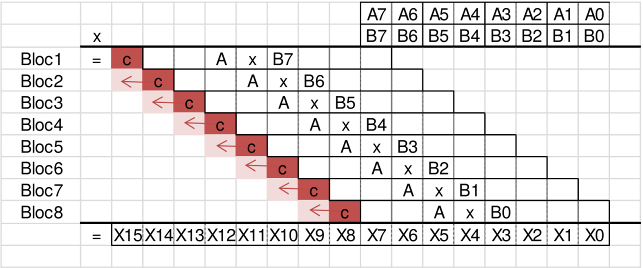


Figure: Propagation of carry during multiplication

Pre-processing phase

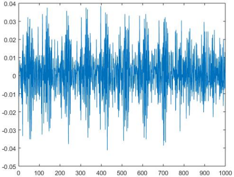


Figure: Pattern of multiplication before reduction

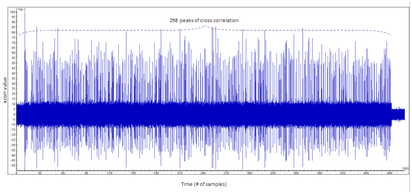


Figure: Cross correlation of multiplication with target trace

Practical OTA on BP256r1 with EM Analysis-Horizontal

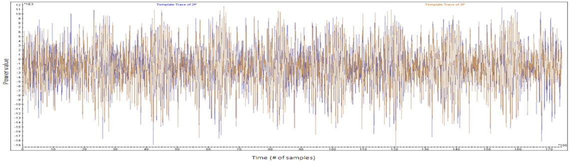


Figure: Same propagation of carry

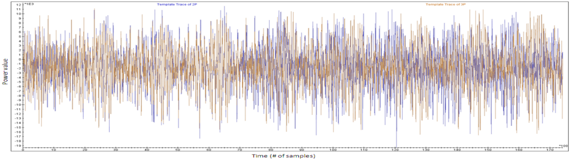


Figure: Different propagation of carry

[M. Dugardin, L. Papachristodoulou, Z. Najm, L. Batina, J.C. Courrege, J.L. Danger, S. Guilley and C. Therond. Dismantling real-world ECC with Horizontal and Vertical Template Attacks. eprint.iacr.org/2015/1001]

Success Rates for 1 key-bit

- Horizontal: 100% success rate with one template trace per bit
- Vertical: Average template traces
- Use 2 averaged templates per key-bit
- Error detection and correction

Number of average traces	1	10	50	100
Success Rate	69%	80,70%	91,60%	99,80%

Table: Different success rates according to the number of average template traces on BP curve.

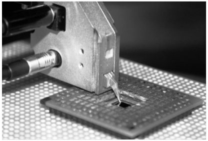
Attacking “SPA-resistant” algorithms

Montgomery ladder for ECC scalar multiplication

Input: P, $k = (k_{x-1}, k_{x-2}, ..., k_0)_2$
Output: $Q = k \cdot P$

$R_0 \leftarrow P$
$R_1 \leftarrow 2 \cdot P$
for $i = x - 2$ downto 0 **do**
    $b = 1 - k_i$
    $R_b = R_0 + R_1$
    $R_{k_i} = 2 \cdot R_{k_i}$
end for
return R_0

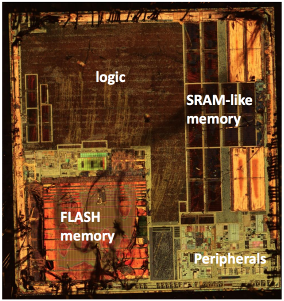
Location-based attacks



- Locating the registers for PKC implementations $\Longrightarrow$ key recovery
- Lookup-table based implementations of the AES cipher can reveal the location of row and column accessed

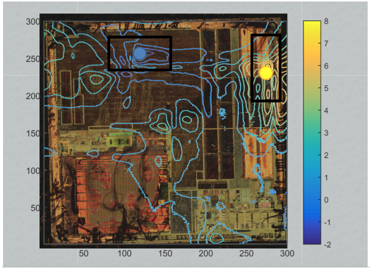
The setup

- modern microcontroller: ARM Cortex M4
- Langer microprobe with spatial resolution of 75 μm (measuring EM emanations)
- scan performed using the XYZ table with step 30 μm, grid size 300x300



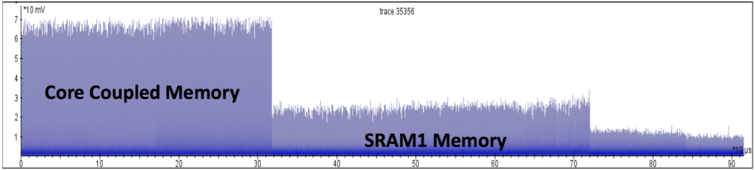
First results: CC memory vs SRAM1

- Blue-coloured dot: strong CC memory emission
- Yellow-coloured dot: strong SRAM1 emission

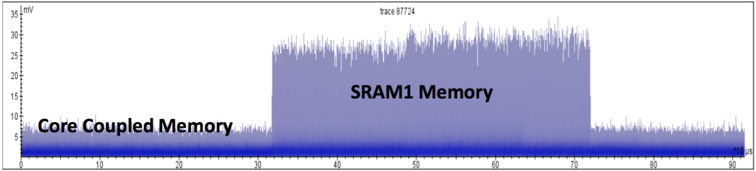


SCA: CC memory vs SRAM1

- Blue-colored dot: strong CC memory emission

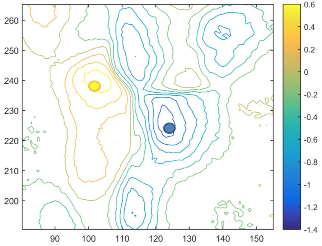


- Yellow-colored dot: strong SRAM1 emission



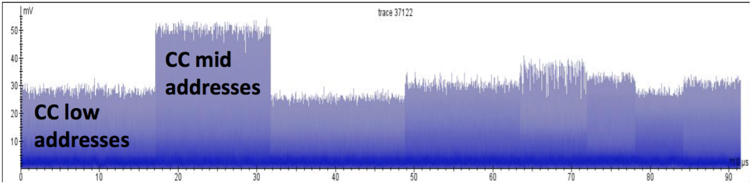
First results: CC memory low addresses vs mid adresses

- Blue-colored dot: stronger emission from mid addresses
- Yellow-coloured dot: stronger emission from low addresses

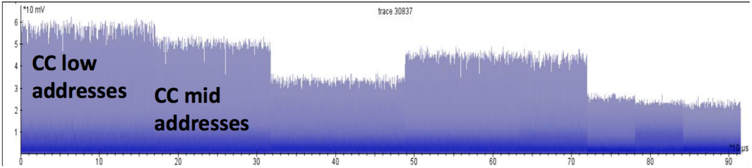


SCA: CC memory low addresses vs mid adresses

- Blue-colored dot: stronger emission from mid addresses



- Yellow-colored dot: stronger emission from low addresses



Conclusions

- Horizontal techniques including OTA are serious issues for ECC implementers
- Countermeasures: input point randomization, random isomorphism, etc.
- Location-based attacks bring in a new dimension
- Ramdomizing memory locations