Side-channel attacks on PKC and countermeasures
(Tutorial @SPACE2016)
with contributions from PhD students

Lejla Batina

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

December 14, 2016
Outline

SPA on PKC: intro
Template Attacks
Template Attacks basics
Online Template Attacks
OTA with EM
Conclusions and Future work
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 → scalar randomization
Basic algorithm for modular exponentiation

Square-and-multiply algorithm

**Input:** \( x, d = (d_{t-1}, d_{t-2}, ..., k_0) \)

**Output:** \( y = x^d \mod N \)

1. \( R_0 \leftarrow 1 \)
2. for \( i = t - 1 \) downto 0 do
3. \( R_0 \leftarrow R_0^2 \mod N \)
4. 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

\[
\text{Input: } x, \ d = (d_{t-1}, d_{t-2}, \ldots, k_0)_2 \\
\text{Output: } y = x^d \mod N
\]

1. \( R_0 \leftarrow 1, R_1 \leftarrow 1, R_2 \leftarrow x \)
2. \text{for } i = t - 1 \ \text{downto} \ 0 \ \text{do}
3. \( R_0 \leftarrow R_0^2 \mod N \)
4. \( b \leftarrow 1 - d_i; R_b \leftarrow R_b \cdot R_2 \mod N \)
5. \text{end for}
6. \text{return } R_0

When \( d_i = 0 \) there is a dummy multiplication!
SPA on ECC double-and-add

Slide credit: L. Chmielewski.
Randomizing message

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

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

**ECDLP**

Let $E$ be an elliptic curve over a finite field $\mathbb{F}_q$, $G = \langle P \rangle$ 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. $kP = P + P + \ldots + P$ \(k\)-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.
- The choice of attack varies for different protocols e.g. the protocol determines scenario.
  → 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 $G \in E(\mathbb{F}_q)$ ($\text{ord}(G) = r$ 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.
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
- **Scalar-mult** level:
  - Random scalar-splitting, randomizing scalar and points
  - Special scalar-indistinguishable group operations: double-and-add always, 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)
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**
  → when processing the same data, the same computations will result in the same 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
Template motivation example

- Key $k$ is refreshed before every encryption
- Does classical CPA work?
- No! It requires a constant key
Template Attack: requirements

- Attacking a well-protected device directly is hard
- So we use an (unprotected device) of the same model

Figure: protected device (left), unprotected device (right)

- We **profile** the unprotected device i.e. we make **templates**
- We use the templates to break the protected device
Template Attack Procedure

1. Choose a **model** that describes the power consumption
2. Profile the **unprotected device** to create the templates (Template Building)
3. Use the templates to break the **protected** device (Template Matching)
Reduced Templates - Template Building

The procedure:

1. Force the transmitter to send bit 0, e.g. 1000 times
2. Measure the voltage in the receiver: \(0v_1, 0v_2, \ldots, 0v_{1000}\)
3. Compute the mean: \(0\bar{v} = \frac{1}{1000} \sum_{i=1}^{1000} 0v_i\)
4. Similarly, force the transmitter to send bit 1, e.g. 1000 times
5. Measure the voltage in the receiver: \(1v_1, 1v_2, \ldots, 1v_{1000}\)
6. Compute the mean: \(1\bar{v} = \frac{1}{1000} \sum_{i=1}^{1000} 1v_i\)

We have computed the templates \(T_0\) and \(T_1\)
Reduced Templates - Template Matching

The procedure:

1. Observe a measurement $v$ in the receiver
2. Match $v$ to template $T_0$, i.e. compute $\text{score}_0 = |v - 0 \bar{v}|$
3. Match $v$ to template $T_1$, i.e. compute $\text{score}_1 = |v - 1 \bar{v}|$
4. if $\text{score}_0 \geq \text{score}_1$ then bit$=0$ else bit$=1$
1. Model the voltage measurements in the receiver using a random variable $V$ and the bit $B$

2. Create a template for the random variable $T_0 = (V|B = 0)$
   Create a template for the random variable $T_1 = (V|B = 1)$

3. Match a measurement $v$ to $T_0$ or $T_1$ using score
• Side-channel analysis focuses on modelling traces
• Previously we modelled a measurement as random variable $V$ and a *realization* of $V$ is a measurement $v$, e.g. $v = 5.1$ Volts
• A side-channel trace consists of several measured samples that we model as a vector of random variables $L = [L^1, L^2, \ldots, L^{no\text{-}samples}]$
• A *realization* of the random vector $L$ is a trace $l$ e.g. $l = [0.41, 0.10, 0.12, 0.17, 0.36]$
Reduced Templates - Template Building

1. Force the cryptographic device to encrypt $n$ times with key $K = key_0$

2. Measure $n$ traces $l_i$ with $K = key_0$

3. The template for $T_{key_0} = (L|K = key_0)$ is the mean vector $key_0 \overline{l} = 1/n \ast \sum_{i=1}^{n} key_0 l_i$

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 $T_{key_1} = (L|K = key_1)$ is the mean vector $key_1 \overline{l} = 1/n \ast \sum_{i=1}^{n} key_1 l_i$
Reduced Templates - Template Matching

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

4. Decide $key_0$ or $key_1$
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](image-url)
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).

![Diagram showing correlation between target and template traces](image)

**Figure:** Correlation of \((i + 1)\)-iteration of target with \(i\)-iteration of template
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
Optimized double-add-always on twisted Edwards curve

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

**Output:** $Q = kP$

1. $R_0 \leftarrow P$
2. **for** $i = x - 2$ **downto** 0 **do**
3. $R_0 \leftarrow 2R_0$
4. $R_1 \leftarrow R_0 + P$
5. $R_0 \leftarrow R_{k_i}$
6. **end for**
7. **return** $R_0$

---

$k = 100$

$R_0 = P$

$R_0 = 2P, R_1 = 3P$, return $2P$

$k = 110$

$R_0 = 4P, R_1 = 5P$, return $4P$
OTA on Montgomery Ladder

Montgomery ladder on twisted Edwards curve

**Input:** \( P, k = (k_{x-1}, k_{x-2}, ..., k_0) \)

**Output:** \( Q = k \cdot P \)

1. \( R_0 \leftarrow P \)
2. \( R_1 \leftarrow 2 \cdot P \)
3. for \( i = x - 2 \) downto 0 do
   4. \( b = 1 - k_i \)
   5. \( R_b = R_0 + R_1 \)
   6. \( R_{k_i} = 2 \cdot R_{k_i} \)
4. end for
5. return \( R_0 \)

---

<table>
<thead>
<tr>
<th>( k = 100 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( R_0 = P, R_1 = 2P )</td>
</tr>
<tr>
<td>( b = 1 ) ( R_1 = 3P, R_0 = 2P ), return ( 2P )</td>
</tr>
<tr>
<td>( b = 1 ) ( R_1 = 5P, R_0 = 4P ), return ( 4P )</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>( k = 110 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( R_0 = P, R_1 = 2P )</td>
</tr>
<tr>
<td>( b = 0 ) ( R_0 = 3P, R_1 = 4P ), return ( 3P )</td>
</tr>
<tr>
<td>( b = 1 ) ( R_1 = 7P, R_0 = 6P ), return ( 6P )</td>
</tr>
</tbody>
</table>
Setup

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

Acquisition Setup with EM Analysis

Figure: SCA equipment at ParisTech, Acquisition with smart-card or STM32f4 platform
Doubling Formulas for point $P = (X, Y, Z)$

$$\text{www.hyperelliptic.org/EFD/g1p/auto-shortw-jacobian.html}$$:

PolarSSL v1.3.7

$$D_1 \leftarrow X \times X \mod p$$
$$D_2 \leftarrow Y \times Y \mod p$$
$$D_3 \leftarrow D_2 \times D_2 \mod p$$
$$D_4 \leftarrow Z \times Z \mod p$$
$$\vdots$$

mbedTLS v2.2.0

$$D_1 \leftarrow X \times X \mod p$$
$$D_2 \leftarrow Y \times Y \mod p$$
$$D_3 \leftarrow Z \times Z \mod p$$
$$D_4 \leftarrow 4X \times D_2 \mod p$$
$$\vdots$$
Finite Field Multiplication in mbedTLS

**Input:** $A$ and $B_7..B_0$ two elements of 256-bits long.

**Output:** $X = A \times B$

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

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

<table>
<thead>
<tr>
<th>Number of average traces</th>
<th>1</th>
<th>10</th>
<th>50</th>
<th>100</th>
</tr>
</thead>
<tbody>
<tr>
<td>Success Rate</td>
<td>69%</td>
<td>80,70%</td>
<td>91,60%</td>
<td>99,80%</td>
</tr>
</tbody>
</table>

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

- Horizontal techniques including OTA are serious issues for ECC implementers
- Leakages propagate in all dimensions
- Countermeasures such as input point randomization, random isomorphism could prevent this kind of attacks
- It is unclear how many key-bits can leak
Future work

- Recent complete formulas could be tested against OTA
- Hardware implementations: how feasible are OTA-like attacks
- Location-based attacks bring in a new dimension
- Randomizing memory locations: what is the impact on horizontal attacks