

ECC Protections against both Observation and Perturbation Attacks

Audrey LUCAS and Arnaud TISSERAND

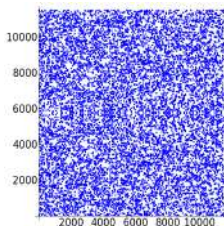
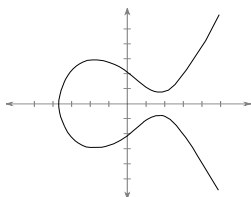
CryptArchi 2017



Outline

- 1 Introduction
- 2 Point Protection using Coordinates Verification
- 3 Scalar Protection using Iteration Counter
- 4 Evaluation of Two Protections
- 5 Conclusion

Elliptic Curves Cryptography (ECC) over \mathbb{F}_p



$$E : y^2 = x^3 + ax + b$$

Point *doubling*:

DBL

\neq

Point *addition*:

ADD

Scalar multiplication



$$[k]P = \underbrace{P + P + \dots + P}_{k \text{ times}}$$

Scalar Multiplication Example: Double and Add

Require: P and $k = (k_{n-1}, \dots, k_0)_2$.

Ensure: $[k]P$

$Q \leftarrow \mathcal{O}$

for $i = n - 1$ **to** 0 **do**

$Q \leftarrow 2 \cdot Q$

if $k_i = 1$ **then**

$Q \leftarrow P + Q$

return Q

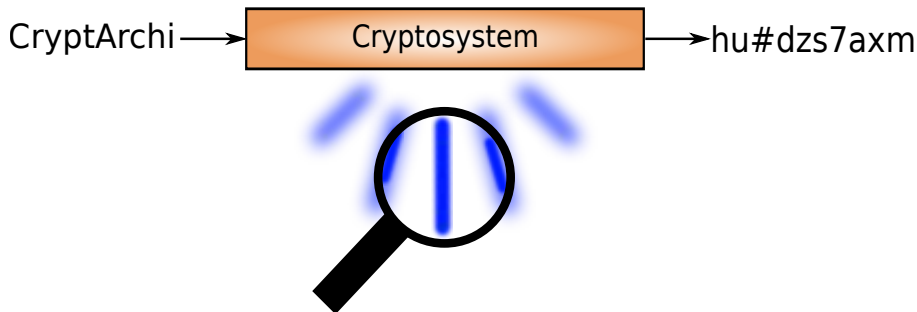
1		0	0	0	1	
DBL	ADD	DBL	DBL	DBL	DBL	ADD

Physical Attacks

Observation: Side Channel Attacks (SCA)

- Computation time, power consumption, ...
- Simple power analysis (SPA), differential power analysis (DPA), ...

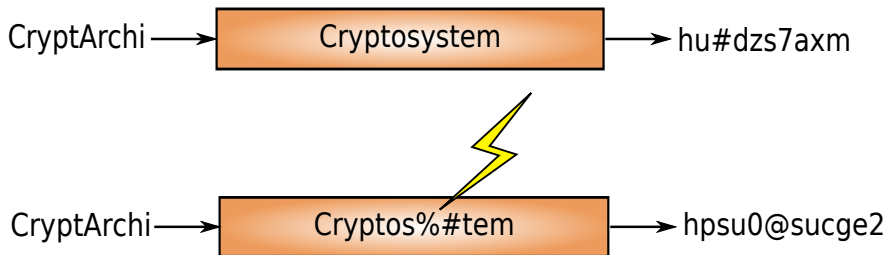
Side channel attacks



Physical Attacks

Perturbation: Fault Attacks (FA)

- Clock, supply voltage, laser, ...
- Bit flip fault, stuck-at fault, ...
- Safe error, differential fault analysis (DFA), ...



Physical Attacks

Countermeasures against SCAs

- Randomization: scalar masking, point blinding, scalar recoding, ...
- Uniformization: uniform curve, regular algorithm, ...
- Hardware: specific logic styles, reconfiguration, ...

Countermeasures against FAs

- Hardware: shielding, sensor, ...
- Redundancy calculation: time, space.
- ECC case: verification of point coordinates at the end of scalar multiplication.

Problem

1		0	0	0	1	
DBL	ADD	DBL	DBL	DBL	DBL	ADD

Double and add

Problem

1		0	0	0	1	
DBL	ADD	DBL	DBL	DBL	DBL	ADD

Double and add

1		0		0		0		1	
DBL	ADD	DBL	ADD	DBL	ADD	DBL	ADD	DBL	ADD

Double and add always

Problem

1		0	0	0	1	
DBL	ADD	DBL	DBL	DBL	DBL	ADD

Double and add

1		0		0		0		1	
DBL	ADD	DBL	ADD	DBL	ADD	DBL	ADD	DBL	ADD

Double and add always - safe error

Problem

Protection for one type of attacks may leave the system vulnerable on other type of attacks.

1		0	0	0	1	
DBL	ADD	DBL	DBL	DBL	DBL	ADD

Double and add

1		0		0		0		1	
DBL	ADD	DBL	ADD	DBL	ADD	DBL	ADD	DBL	ADD

Double and add always - safe error

Problem

Protection for one type of attacks may leave the system vulnerable on other type of attacks.

1		0	0	0	1	
DBL	ADD	DBL	DBL	DBL	DBL	ADD

Double and add

1		0		0		0		1	
DBL	ADD	DBL	ADD	DBL	ADD	DBL	ADD	DBL	ADD

Double and add always - safe error

Protect the system from both types of attacks simultaneously.

Outline

- 1 Introduction
- 2 Point Protection using Coordinates Verification
- 3 Scalar Protection using Iteration Counter
- 4 Evaluation of Two Protections
- 5 Conclusion

Point Verification

Principle:

- Verify if the current/final point is on the curve [BMM].
- Adaptable for many curves and many coordinates.

Verification period:

- At the very end: very low cost but late detection.
- Every ℓ iterations: larger cost but earlier detection.
- At each iteration ($\ell = 1$): important cost but immediate detection and possible uniformization of behaviour.

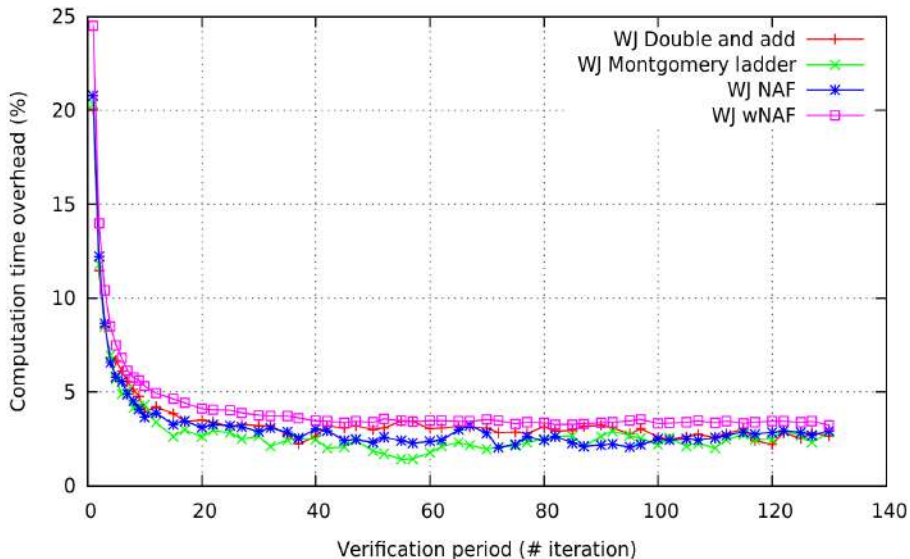


I. Biehl, B. Meyer, and V. Müller.

Differential Fault Attacks on Elliptic Curve Cryptosystems.

In *Proc. Advances in Cryptology - CRYPTO 2000*.

Software Implementation Result of Curve on 521 bits



Uniformization I

Uniformization:

1			0		0		0		1		
DBL	V	ADD	DBL	V	DBL	V	DBL	V	DBL	V	ADD

Behaviour for ADD and for DBL V must be very close.

Example: Weierstrass curve - projective

Verification:

$$V : Y^2Z = X^3 + aXZ^2 + bZ^3$$

\Rightarrow Cost: $4M + 3S + 5A + 1 \times b$

ADD	DBL	DBL+V
$11M + 6S + 18A$	$5M + 6S + 14A$	$9M + 8S + 19A + 1 \times b$

Uniformization II

Example: Weierstrass curve - projective

- New verification:

$$Y^3Z = X^3Y + aXYZ^2 + bYZ^3$$

\Rightarrow Cost: $8M + 3S + 5A + 1 \times b$.

- Verification is included in *DBL*.

\Rightarrow Cost: $13M + 9S + 19A + 1 \times b$.

- Optimization using factorization.

\Rightarrow Cost: $11M + 6S + 18A + 1 \times b$.

- $1 \times b$ is added to *ADD*.

The costs for *ADD* and *DBL* + *V* are equal.

Summary

Good points

- This protection is equivalent to *double and add always* but at for smaller cost.
- Bit flip detection in all field elements.

Weakness

- Scalar is vulnerable to fault attacks.

Outline

- 1 Introduction
- 2 Point Protection using Coordinates Verification
- 3 Scalar Protection using Iteration Counter**
- 4 Evaluation of Two Protections
- 5 Conclusion

Scalar Protection

Point verification does not protect the scalar from fault attack.

Example

1			0		0		0		1		
DBL	V	ADD	DBL	V	DBL	V	DBL	V	DBL	V	ADD

Scalar Protection

Point verification does not protect the scalar from fault attack.

Example

1			0		0		0	1		
DBL	V	ADD	DBL	V	DBL	V	DBL	V	DBL	ADD

Scalar Protection

Point verification does not protect the scalar from fault attack.

Example

1			0			0			0			1		
DBL	V	ADD	DBL	V	DBL	V	DBL	V	DBL	V	DBL	V	ADD	

1			0		0		1			1		
DBL	V	ADD	DBL	V	DBL	V	DBL	V	ADD	DBL	V	ADD

Scalar Protection

Point verification does not protect the scalar from fault attack.

Example

1			0			0			0			1		
DBL	V	ADD	DBL	V	DBL	V	DBL	V	DBL	V	DBL	V	ADD	

1			0		0		1			1		
DBL	V	ADD	DBL	V	DBL	V	DBL	V	ADD	DBL	V	ADD

- Attack is not detected: current point is on curve but is wrong result.
- Proposed countermeasure: *Iteration Counter*.

Scalar Protection - Iteration Counter

Aim: Verify if the operation sequence is equivalent to key k .

Scalar Protection - Iteration Counter

Aim: Verify if the operation sequence is equivalent to key k .

First idea, naive idea:

- Count the number of *ADDs* during computations.
 - Use a *cmp* register and a *ref* reference value.
 - If $k_i = 1$, add 1 to register.
- No detection in case of 2 bit flips at different indexes.

Scalar Protection - Iteration Counter

Aim: Verify if the operation sequence is equivalent to key k .

First idea, naive idea:

- Count the number of *ADDs* during computations.
 - Use a *cmp* register and a *ref* reference value.
 - If $k_i = 1$, add 1 to register.
- No detection in case of 2 bit flips at different indexes.

Second idea:

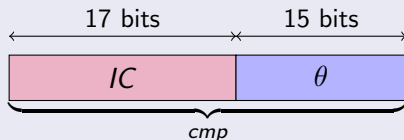
- Count the number of *ADDs* combined with a weight.
 - This weight is the iteration index.
 - If $k_i = 1$, add i to register.
- Problem: if $k_i = 0$, no electric activity.

Scalar Protection - Iteration Counter

Last idea:

Divide the register in 2 parts.

- First part: counter IC .
- Second part: noise θ .
- If $k_i = 1$, add iteration index i to IC .
- If $k_i = 0$, add random number to θ .
- At the end: comparison between IC and the reference counter ref .
- (17, 15) parameters has been selected for first implementation (work in progress).



Example - Double and Add

Require: $P, k = (k_{n-1}, \dots, k_0)_2$. Data: *ref*.

Ensure: $Q = k \cdot P$

for $i = n - 1$ **to** 0 **do**

$Q = 2 \cdot Q$

$\theta = \text{random number}$

if $k_i = 1$ **then**

$Q \leftarrow Q + P$

$cmp \leftarrow cmp + i \cdot 2^{15}$

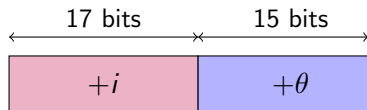
else

$cmp \leftarrow cmp + \theta$

$IC \leftarrow cmp \gg 15$

if $IC \neq ref$ **then**

return *Fault detected*



Summary

Good points

- Cost: n small integer additions + small random number generator.
- Bit flip detection on the scalar.

Weakness

- Late detection.
- Stuck-at fault not detected \Rightarrow safe error vulnerability.

Future work

- Key recoding.

Outline

- 1 Introduction
- 2 Point Protection using Coordinates Verification
- 3 Scalar Protection using Iteration Counter
- 4 Evaluation of Two Protections**
- 5 Conclusion

Evaluation of Two Protections I

What is implemented

- 3 curves:
 - Weierstrass, Edwards and Tripling-oriented Doche–Icart–Kohel (TDIK).
- 4 coordinates:
 - affine, projective, jacobian, standard.
- 4 algorithms:
 - double and add, Montgomery ladder, NAF, w -NAF ($w = 3, 4$).
- 3 point *verification* methods:
 - at the very end;
 - every ℓ iterations;
 - at each iteration ($\ell = 1$).

Evaluation of Two Protections II

Implementation target

- C language with GMP library.
- Intel core i7-5600U CPU @ 2.60GHz

Algorithms	Coordinates		
	Affine	Jacobian	Projective
Double and add	5.7%	25.3%	18.8%
Montgomery ladder	6.0%	26.2%	19.0%
NAF	6.1%	25.5%	18.1%
3-NAF	7.2%	29.8%	24.7%

Computation overheads in the worst case for Weierstrass curves.

Outline

- 1 Introduction
- 2 Point Protection using Coordinates Verification
- 3 Scalar Protection using Iteration Counter
- 4 Evaluation of Two Protections
- 5 Conclusion

Conclusion

What is done

- Bit flip attack detection.
 - on all field elements, on the scalar.
 - low cost.
 - adaptable for many curves.
- SPA resistant.

Future works

- Stuck-at fault and safe error.
- Implementation on small processors (32, 16 bits).

Thank you for your attention.

Questions?