Efficient Application of Countermeasures for Elliptic Curve Cryptography

Vladimir Soukharev, Ph.D.
Basil Hess, Ph.D.

InfoSec Global Inc.

May 19, 2017

Outline

- Introduction
- Brief Summary of ECC Arithmetic
- Coutermeasure Classifications
- Optimization Techniques
- Coutermeasure Techniques
- Summary and Conclusion

INFOSEC GLOBAL

Introduction

- A cryptographic algorithm can be mathematically shown to be secure.
- However, this does not guarantee full practical security.
- Reason: there can also be attacks on the physical layer.
- Two categories: passive and active attacks.
- We need to take extra steps and effort to prevent such attacks and provide full security.

INFOSEC GLOBAL

Elliptic Curves

We assume that F is a *finite field* of characteristic *greater than* 3. "Finite field" is essential, because cryptography uses finite fields. "Characteristic greater than 3" is not essential, but it simplifies matters greatly.

Definition

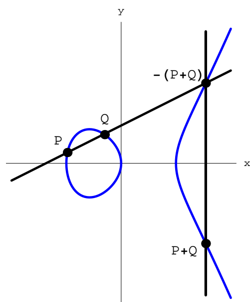
An *elliptic curve* over F is the set of solutions $(x, y) \in F^2$ to an equation

$$y^2 = x^3 + ax + b, \quad a, b \in F,$$

plus an additional point ∞ (at infinity).

INFOSEC GLOBAL

Group Law



Elliptic curves admit an abelian group operation with an identity element ∞. Let $P = (x_1, y_1)$ and $Q = (x_2, y_2)$. Then

$$P+Q = \left(\left(\frac{y_2-y_1}{x_2-x_1}\right)^2 - x_1 - x_2, \right.$$

$$\left. - \left(\frac{y_2-y_1}{x_2-x_1}\right) \left(\left(\frac{y_2-y_1}{x_2-x_1}\right)^2 - 2x_1 - x_2 \right) - y_1 \right)$$

INFOSEC GLOBAL

Elliptic Curve Scalar Multiplication

- A frequent ECC operation is scalar multiplication
- $Q = d \cdot P = \underbrace{P + ... + P}_{d}$
- Often the scalar d is the private key and Q is the public key, with P being the global public parameter.
- Getting d, given P and Q, needs the Elliptic Curve Discrete Logarithm Problem (ECDLP) to be to solved.
- Needs to be implemented efficiently, *but...*
- Depending on the implementation, d can leak through a side channel.

What are *Side-Channel Attacks*?

Side-Channel Attacks (SCAs) are attacks that aim at the physical implementation of a cryptosystem and the emissions of that implementation.

Side channels:

- Power consumption
- Timing
- RF emissions
- Sound
- ...

INFOSEC GLOBAL

Example of a Side-Channel Attack

Consider the *DOUBLE-and-ADD* algorithm for scalar multiplication:

- The secret scalar d is written in binary format.
- The algorithm runs iteratively.
- If the corresponding bit for the given iteration is 0, only *DOUBLE* is performed.
- If the corresponding bit for the given iteration is 1, *DOUBLE* and *ADD* are performed.

DOUBLE and ADD have different complexities. ADD is more expensive than DOUBLE.
Result: An adversary observing power consumption can obtain all of the bits of d!

INFOSEC GLOBAL

Simple Side-Channel Attacks

Definition:

- A simple SCA obtains information from single observed operations.

Example:

- In the case of elliptic curve cryptography (ECC), the operation is a single scalar multiplication.
- ECC is based on the computational hardness of the ECDLP: For a point P on the elliptic curve and a random secret value $d \in \{1, \ldots, n-1\}$, it is hard to derive the discrete logarithm d from $Q = dP$.
- A simple SCA exploits timing or power consumption characteristics of a scalar multiplication algorithm (i.e. the *DOUBLE-and-ADD* algorithm) that depend on the secret scalar d.

INFOSEC GLOBAL

Differential Side-Channel Attacks

Definition:

- Differential SCAs obtain information from multiple observed operations with the same secret.
- Statistical methods are applied to derive the secret from the traces.

Example:

- Operations performed with a static private key.

INFOSEC GLOBAL

Goubin-Type Side-Channel Attacks

Definition:

- Refers to elliptic curve elements that remain unchanged after applying randomization techniques against a differential SCAs.

Example:

- Affine $P = (x, y)$ can be represented in the projective space as $P' = (\theta x, \theta y, \theta z)$ for $\theta \in K^*$.
- Zeros in the points $(X, 0, Z)$ or $(0, Y, Z)$ remain.

INFOSEC GLOBAL

Fault Injection Attacks

Definition:

- Injecting *faults* to hardware or to software algorithms in order to provoke them to behave abnormally.

Example:

- *Padding oracles* by provoking error states (e.g. in AES CBC-MAC).
- "Flipping bits" in hardware using out-of-specification environmental conditions:
  - Voltage, power glitches, clock, temperature, UV, laser, X-rays, etc.

INFOSEC GLOBAL

Purpose of Countermeasures

Resist simple and differential SCAs, and fault-injection attacks.

- Make cryptosystems really secure.
- No need to rely on hardware countermeasures.
- Implement countermeasures in software.

INFOSEC GLOBAL

Optimizations

- Fixed point scalar multiplication
- Variable point scalar multiplication
- Coordinate selection
- Linear combinations

INFOSEC GLOBAL

Fixed-Base Comb Method

Table: Precomputed table for 256-bit curves.

index	bits	point
0	0000	$0 \cdot G$
1	0001	$1 \cdot G$
2	0010	$2^{64} \cdot G$
3	0011	$(2^{64} + 1) \cdot G$
4	0100	$2^{128} \cdot G$
...	...	...

During the scalar multiplication $k \cdot P$, the columns of the above matrix are processed "column by column": Initialize $Q = \infty$, then for each $i \in \{d-1, ..., 0\}$:

- $Q = 2Q$
- $Q = Q + [K_{w-1,i}, ..., K_{1,i}, K_{0,i}] \cdot P$

INFOSEC GLOBAL

Fixed-Base Comb Method - Performance

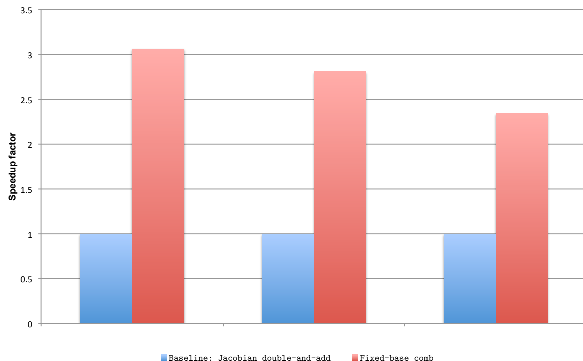


Figure: Fixed-base comb method, double-and-add as baseline (speedup factor 1). 256-bit (left), 384-bit (middle), 521-bit curve (right)

INFOSEC GLOBAL

Fixed-base Comb Method - Side Channels?

Lookup tables are dangerous, because it makes the cryptosystem vulnerable to Cache-Timing attacks.
Solutions:

- Secret-dependent table indices.
  - Example: table-based AES implementations.
- Be careful. Try to preload table to cache.
- The table should not be too large.

Sliding Window Scalar Multiplication

Combines the NAF method for multiplication with a sliding window.

- Use a window of size w.
- Precompute $P_i = iP$ for all odd i up to window size.
- In each iteration, use the greatest subset of odd size $i \leq w$.

INFOSEC GLOBAL

Sliding Window Scalar Multiplication - Performance

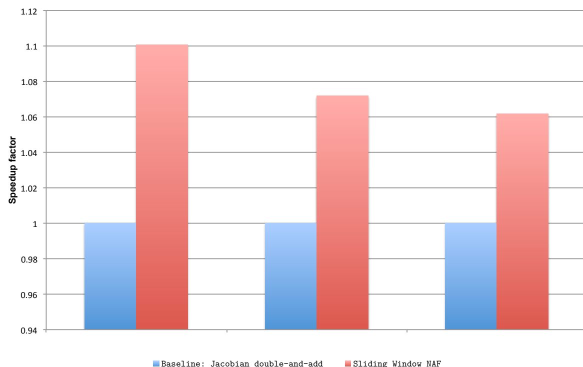


Figure: Sliding window NAF method, Jacobian double-and-add as baseline (speedup factor 1). 256-bit (left), 384-bit (middle), 521-bit curve (right).

INFOSEC GLOBAL

Sliding Window Scalar Multiplication - Side Channels?

- In a sliding window, it is not predetermined what element to process next.
- Depends on the secret scalar.
- FLUSH+RELOAD attack.

How to resist? Use a deterministic, fixed window.

Coordinate Selection - Candidates

Curve in Weierstrass form: $y^2 = x^3 + Ax + B$

- Affine: Points on the curve are represented as $P = (x, y)$.
- Standard projective: $(X : Y : Z), Z \neq 0$ corresponds to affine $(X/Z, Y/Z)$.
- Jacobian projective: $(X : Y : Z), Z \neq 0$ corresponds to affine $(X/Z^2, Y/Z^3)$.

INFOSEC GLOBAL

Coordinate Selection

Selection procedure

1. Obtain modular inversion and multiplication timings in each prime field F_p.
2. For all coordinate candidates, count the number of additions and doublings.
3. Select a representation considering the I/M ratio.

INFOSEC GLOBAL

Coordinate Selection - Operations

We have determined that Jacobian Projective coordinates are most efficient for the general case. There are three major operation types, whose costs we need to consider (assuming field multiplication and squaring have the same cost).

- DOUBLE: 10 field multiplications.
- ADD: 16 field multiplications.
- MIXED ADDITION (Jacobian and Affine): 11 field multiplications.

Linear Combination (New!)

Common ECC operations are *linear combinations*:

- $s \cdot P + r \cdot Q$
- Speedup of 20% with the optimized method

1. Convert P into Jacobian coordinates. Do not discard the affine version of P.
2. Convert Q into Jacobian coordinates. Do not discard the Affine version of Q.
3. Compute $T = s \cdot P$ (in Jacobian coordinates, using the affine coordinates of P for the ADD steps in mixed addition).
4. Compute $L = r \cdot Q$ (in Jacobian coordinates, using the affine coordinates of Q for the ADD steps in mixed addition).
5. Compute $T + L$ (in pure Jacobian coordinates).
6. Take the result and convert it from Jacobian to Affine.

INFOSEC GLOBAL

Linear Combination (2)

Common ECC operations are *linear combinations*:

- $s \cdot (P + r \cdot Q)$

1. Convert Q into Jacobian coordinates. Do not discard the affine version of Q.
2. Compute $T = r \cdot Q$ (in Jacobian coordinates, using the affine coordinates of Q for the ADD steps, in mixed addition).
3. Add to T point P, i.e. compute $L = T + P$. (Note that here, we are doing mixed coordinate addition, as P is in affine coordinates. The result is in Jacobian coordinates.)
4. Convert L to Affine, saving the Jacobian representation.
5. Compute $s \cdot L$.
6. Take the result and convert it from Jacobian to Affine.

Countermeasures

- Parameter selection for elliptic curves
- Constant-time implementations
- Constant-time using Projective Jacobian coordinates
- Blinding for curves in Weierstrass form
- Blinding for Edwards curves
- Preventing fault-injection attacks

INFOSEC GLOBAL

Parameter Selection

Correct parameter selection prevents Goubin-type attacks.
We need to select elliptic curve $E : y^2 = x^3 + Ax + B$ (over finite field F), that does not have points of the form:

1. $(x, 0)$ for some $x \in F$.
2. $(0, y)$ for some $y \in F$.

To satisfy the above conditions, the following will suffice for each:

1. - E should have no points of order 2.
   - $x^3 + Ax + B$ should be irreducible over F.
2. B must not be a square in F.

INFOSEC GLOBAL

Constant Time Implementation

- Implement field arithmetic using 64-bit arithmetic and Boolean operations only.
- For constant flow, avoid any conditional carries and branches.
- Add a dummy operation to the algorithms (e.g. double-and-add). If no addition is necessary, always add the point-at-infinity.
- Lookup tables should have constant access time.

Constant Time - Pitfalls

Beware of compilers!

- Constant-time lookup (C) code might be compiled to conditional branches.
- Example: Long multiplications in 64-nit Windows exploited in Curve25519-donna.
- CPU instructions might be not constant time.

What to do?

- `#pragma GCC optimize ("O0")`. Be careful with portability, and not all optimizations are bad...
- Carefully analyze the ASM output and fix sections if necessary.
- Be aware what platforms and compilers should be supported.

Constant-Time Using Projective Jacobian Coordinates (New!)

Problem for the curves in Weierstrass form: different algorithms for ADD and DOUBLE.

Goal: Make DOUBLE and ADD indistinguishable.

Recall: When using Projective Jacobian Coordinates, DOUBLE needs 10, while MIXED ADDITION needs 11 field multiplications.

Solution: Add one dummy field multiplication in DOUBLE.

Blinding for Weierstrass Curves

Countermeasure against a differential SCA, if a secret d is reused.

Affine version (compute $d \cdot P$):

- Select a random point R.
- Instead of computing $d \cdot P$, compute $d \cdot (R + P)$.
- Subtract $S = d \cdot R$ to obtain $d \cdot P$.
- In the next operation with d, update R and S: $R' = (-1)^b 2R$, $S' = (-1)^b 2S$.

Projective version: chose random $\theta \in F$...

- Standard projective: $(\theta x : \theta y : \theta)$
- Jacobian projective: $(\theta^2 x : \theta^3 y : \theta)$

INFOSEC GLOBAL

Edwards Curves

Curve equation:

- Untwisted: $x^2 + y^2 = 1 + dx^2y^2$, $d \in F$ - e.g. ed448
- Twisted: $ax^2 + y^2 = 1 + dx^2y^2$, $a, d \in F$ - e.g. ed25519

Advantages of Edwards form:

- A complete formula for addition - no doubling formula needed.
- Efficient addition formula
- Used in EdDSA

Blinding usually only discussed for Weierstrass curves.

Blinding for Edwards Curves (New!)

- Homogeneous projective coordinates (1)
- Inverted coordinates (2)
- Extended coordinates (3)

-	(1)	(2)	(3)
X	$X = \theta \cdot x$	$X = \theta \cdot y$	$X = \theta \cdot x$
Y	$Y = \theta \cdot y$	$Y = \theta \cdot x$	$Y = \theta \cdot y$
Z	$Z = \theta$	$Z = \theta \cdot x \cdot y$	$Z = \theta$
T	-	-	$T = \theta \cdot x \cdot y$
P_B	$(X : Y : Z)$	$(X : Y : Z)$	$(X : Y : Z : T)$
Overhead	20%	10%	10%

INFOSEC GLOBAL

Classification: Security and Efficiency (New!)

Four categories:

1. Scalar multiplication with generator G by *secret parameter*.
2. Scalar multiplication with generator G by *publicly known parameter*.
3. Scalar multiplication with point $P \neq G$ by *secret parameter*.
4. Scalar multiplication with point $P \neq G$ by *publicly known parameter*.

INFOSEC GLOBAL

Classification: Security and Efficiency (2)

Cat.	SSCA CM	DSCA CM	Optimizations	Fixed-point
1	✓	✗	Applicable	✓
2	✗	✗	High applicability	✓
3	✓	✓	Medium applicability	✗
4	✗	✗	High applicability	✗

INFOSEC GLOBAL

Preventing Fault-Injection Attacks (New!)

Protection for functions that rely on single-bit outputs or single-bit inputs:

- isZero
- isEqual
- isUnity
- isBitSet
- isPow2
- swapConditional
- copyConditional
- ECC signature verification

INFOSEC GLOBAL

Preventing Fault-Injection Attacks (2)

Functions depending on *single output-bit*.

- The operation $f \in \{0, 1\}$ is performed in two variations: $f_1 \in \{0, 1\}$, $f_2 \in \{0, 1\}$.
- The second operation returns the inverse of the first operation: $f_2 = \neg f_1$.
- If the above property does not hold, the result is "invalid".
- Both operations may be performed in constant-time to avoid timing attacks on cryptographic applications.
- The function values are interpreted in the following way:
  - $f_1 = 1, f_2 = 0$ corresponds to $f = 1$.
  - $f_1 = 0, f_2 = 1$ corresponds to $f = 0$.
  - $f_1 = 0, f_2 = 0$ corresponds to "invalid".
  - $f_1 = 1, f_2 = 1$ corresponds to "invalid".

INFOSEC GLOBAL

Example: isZero

```
int isZero(int* i1, int iLen) {
  int f1 = isZeroF1(i1, iLen);
  int f2 = isZeroF2(i1, iLen);
  if (f1 == f2) return ERROR_CODE;
  return f1 & !f2;
}

int isZeroF1(int* i1, int iLen) {
  int res = 0;
  for (int i = 0; i < iLen; ++i) {
    res |= i1[i];
  }
  return !res;
}

int isZeroF2(int* i1, int iLen) {
  int res = 1;
  for (int i = 0; i < iLen; ++i) {
    res &= !i1[i];
  }
  return !res;
}
```

INFOSEC GLOBAL

Preventing Fault-Injection Attacks (3)

Functions depending on *single input-bit*.

- The operation $g(i), i \in \{0,1\}$ is performed twice: $g_1(i)$, $g_2(\neg i)$. g_2 operates on the inverted input bit of g_1.
- If $g(i) \neq g(\neg i)$, both g_1 and g_2 are suppose to return a different value on different inputs.
- Both operations may be performed in constant-time to avoid timing attacks on cryptographic applications.
- If the return values of g_1 and g_2 are equal, the result is interpreted as "valid", and "invalid" otherwise.

INFOSEC GLOBAL

Example: Conditional Swap

```
int swapConditional(int* i1, int* i2, int bit) {
  int i1Bak = *i1;
  int i2Bak = *i2;
  swapConditionalF1(i1, i2, bit);
  swapConditionalF2(&i1, &i2, bit);
  if (!(*i1 == i2Bak & *i2 == i1Bak)) return ERROR_CODE;
}

void swapConditionalF1(int* i1, int* i2, int bit) {
  int mask = ~bit;
  int ret = mask & (*i1 ^ *i2);
  *i1 = *i1 ^ ret;
  *i2 = *i2 ^ ret;
}

void swapConditionalF2(int* i1, int* i2, int bit) {
  int mask = ~(!bit);
  int ret = mask & (*i1 ^ *i2);
  *i1 = *i1 ^ ret;
  *i2 = *i2 ^ ret;
}
```

Summary, Remarks, and Future Development

- Optimizations and Countermeasures for ECC.
- Simple SCAs and differential SCAs.
- Often a trade-off between efficient and secure code.
- Countermeasures are not always necessary. This opens potential for optimization, without reducing security.
- It is advisable to categorize cryptography operations.
- Future work: optimizations and countermeasures for isogeny-based cryptography and other quantum-resistant schemes.

Thank You!

INFOSEC
GLOBAL

INFOSEC
GLOBAL