Elliptic-curve cryptography VII

Timing attacks and scalar multiplication

Tanja Lange

Eindhoven University of Technology

2MMC10 – Cryptology

Timing attacks are not a new phenomenon

Password recovery if server compares letter by letter:

Try `AAA`,

Timing attacks are not a new phenomenon

Password recovery if server compares letter by letter:

Try `AAA`, `BBB`,

Timing attacks are not a new phenomenon

Password recovery if server compares letter by letter:

Try `AAA`, `BBB`, `CCC`, ...

Timing attacks are not a new phenomenon

Password recovery if server compares letter by letter:

Try `AAA`, `BBB`, `CCC`, ..., `CCC` takes slightly longer to fail.
Try `CAA`,

Timing attacks are not a new phenomenon

Password recovery if server compares letter by letter:

Try `AAA`, `BBB`, `CCC`, ..., `CCC` takes slightly longer to fail.
Try `CAA`, `CBB`,

Timing attacks are not a new phenomenon

Password recovery if server compares letter by letter:

Try `AAA`, `BBB`, `CCC`, . . . , `CCC` takes slightly longer to fail.
Try `CAA`, `CBB`, `CCC`, . . .

Timing attacks are not a new phenomenon

Password recovery if server compares letter by letter:

Try `AAA`, `BBB`, `CCC`, . . . , `CCC` takes slightly longer to fail.
Try `CAA`, `CBB`, `CCC`, . . . , `CRR` takes slightly longer to fail.
Try `CRA`,

Timing attacks are not a new phenomenon

Password recovery if server compares letter by letter:

Try `AAA`, `BBB`, `CCC`, ..., `CCC` takes slightly longer to fail.
Try `CAA`, `CBB`, `CCC`, ..., `CRR` takes slightly longer to fail.
Try `CRA`, `CRB`,

Timing attacks are not a new phenomenon

Password recovery if server compares letter by letter:

Try `AAA`, `BBB`, `CCC`, . . . , `CCC` takes slightly longer to fail.
Try `CAA`, `CBB`, `CCC`, . . . , `CRR` takes slightly longer to fail.
Try `CRA`, `CRB`, `CRC`, . . .

Timing attacks are not a new phenomenon

Password recovery if server compares letter by letter:

Try `AAA`, `BBB`, `CCC`, ..., `CCC` takes slightly longer to fail.
Try `CAA`, `CBB`, `CCC`, ..., `CRR` takes slightly longer to fail.
Try `CRA`, `CRB`, `CRC`, ..., `CRY` takes slightly longer to fail.

$\vdots$

Timing attacks are not a new phenomenon

Password recovery if server compares letter by letter:

Try `AAA`, `BBB`, `CCC`, . . . , `CCC` takes slightly longer to fail.
Try `CAA`, `CBB`, `CCC`, . . . , `CRR` takes slightly longer to fail.
Try `CRA`, `CRB`, `CRC`, . . . , `CRY` takes slightly longer to fail.

$\vdots$

Password is `CRYPTOLOGY`.

1974: Exploit developed by Alan Bell for TENEX operating system.

Reminder: double-and-add method

Compute aP given a and P.

```
a = 44444 # our super secret scalar. No, not that one.
l = a.nbits()
A = a.bits()
R = P
for i in range(l-2,-1,-1):
  R = 2 R
  if A[i] == 1:
    R = R + P
print(R)
```

Reminder: double-and-add method

Compute aP given a and P.

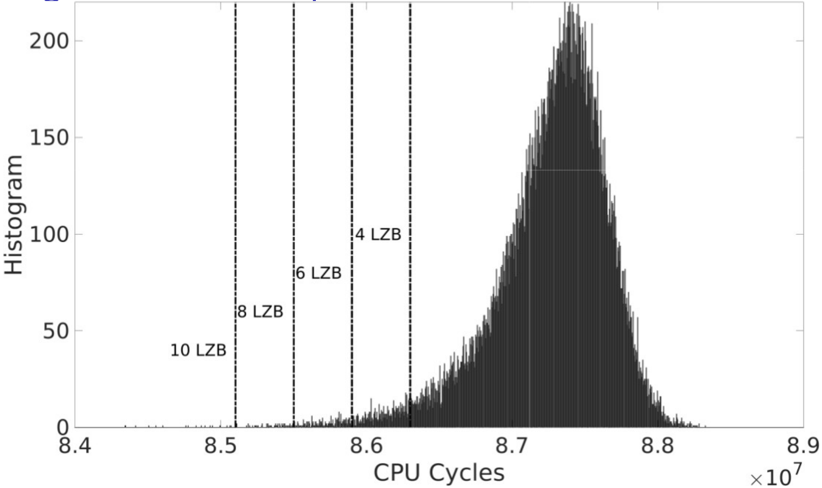
```
a = 44444 # our super secret scalar. No, not that one.
l = a.nbits()
A = a.bits()
R = P
for i in range(l-2,-1,-1):   # loop length depends on a
  R = 2 R
  if A[i] == 1:
    R = R + P
print(R)
```

Reminder: double-and-add method

Compute aP given a and P.

```
a = 44444 # our super secret scalar. No, not that one.
l = a.nbits()
A = a.bits()
R = P
for i in range(l-2,-1,-1):   # loop length depends on a
  R = 2 R
  if A[i] == 1:    # branch depends on a
    R = R + P
print(R)
```

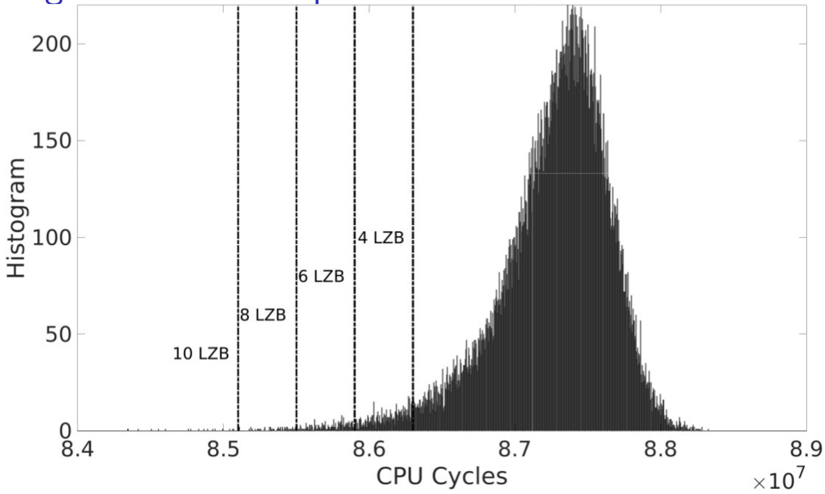
Timings of scalar multiplication on NIST P-256



(Picture from TPM-Fail)

NIST P-256 is an elliptic curve standardized by NIST.
It is a Weierstrass curve modulo a 256-bit prime.

Timings of scalar multiplication on NIST P-256



(Picture from TPM-Fail)

NIST P-256 is an elliptic curve standardized by NIST.

It is a Weierstrass curve modulo a 256-bit prime.

Timing depends strongly on the length of the scalar, also on Hamming weight.

Fixed window method

- Faster methods reduce the number of additions by using windows:
  $14019 =$

Fixed window method

- Faster methods reduce the number of additions by using windows:
  14019 =11   0110 1100 0011

Fixed window method

- Faster methods reduce the number of additions by using windows:
  $14019 = \underbrace{11}_{3}\ \underbrace{0110}_{1\ 2}\ \underbrace{1100}_{3\ 0}\ \underbrace{0011}_{0\ 3}$

Fixed window method

- Faster methods reduce the number of additions by using windows:
  $14019 = \underbrace{11}_{3}\ \underbrace{0110}_{1\ 2}\ \underbrace{1100}_{3\ 0}\ \underbrace{0011}_{0\ 3}$
  Precompute P, $2P$, and $3P$. Left window is innermost coefficient.

  $$14019P = 4\left(4\left(4\left(4\left(4\left(4\left(3P\right)+P\right)+2P\right)+3P\right)\right)\right)+3P.$$

  Same number of doublings, 4 instead of 7 additions.

Fixed window method

- Faster methods reduce the number of additions by using windows:
  $14019 = \underbrace{11}_{3}\ \underbrace{0110}_{1\ 2}\ \underbrace{1100}_{3\ 0}\ \underbrace{0011}_{0\ 3}$
  Precompute P, $2P$, and $3P$. Left window is innermost coefficient.

  $$14019P = 4\left(4\left(4\left(4\left(4\left(4\left(3P\right) + P\right) + 2P\right) + 3P\right)\right)\right) + 3P.$$

  Same number of doublings, 4 instead of 7 additions.
- General case: width-w windows.
  Start from least-significant bit (coefficient of 2^0)
  turn w bits into coefficient in $[2^w - 1, 0]$,
  pad with 0 bits if length is not a multiple of w.

Fixed window method

- Faster methods reduce the number of additions by using windows:
  $14019 = \underbrace{11}_{3}\ \underbrace{0110}_{1\ 2}\ \underbrace{1100}_{3\ 0}\ \underbrace{0011}_{0\ 3}$
  Precompute P, $2P$, and $3P$. Left window is innermost coefficient.

  $$14019P = 4\left(4\left(4\left(4\left(4\left(4\left(3P\right)+P\right)+2P\right)+3P\right)\right)\right)+3P.$$

  Same number of doublings, 4 instead of 7 additions.
- General case: width-w windows.
  Start from least-significant bit (coefficient of 2^0)
  turn w bits into coefficient in $[2^w-1, 0]$,
  pad with 0 bits if length is not a multiple of w.

  E.g. $w = 4$, so coefficients in $[15, 0]$.
  $14019 = \underbrace{0011}_{3}\ \underbrace{0110}_{6}\ \underbrace{1100}_{12}\ \underbrace{0011}_{3}$

Fixed window method

- Faster methods reduce the number of additions by using windows:
  $14019 = \underbrace{11}_{3}\ \underbrace{0110}_{1\ 2}\ \underbrace{1100}_{3\ 0}\ \underbrace{0011}_{0\ 3}$
  Precompute P, $2P$, and $3P$. Left window is innermost coefficient.

  $$14019P = 4\left(4\left(4\left(4\left(4\left(4\left(3P\right)+P\right)+2P\right)+3P\right)\right)\right)+3P.$$

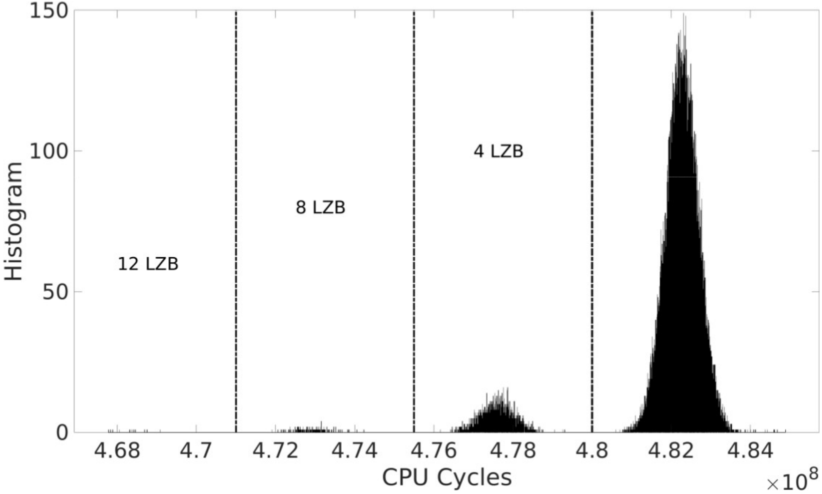
  Same number of doublings, 4 instead of 7 additions.
- General case: width-w windows.
  Start from least-significant bit (coefficient of 2^0)
  turn w bits into coefficient in $[2^w-1,0]$,
  pad with 0 bits if length is not a multiple of w.

  E.g. $w=4$, so coefficients in $[15,0]$.
  $14019 = \underbrace{0011}_{3}\ \underbrace{0110}_{6}\ \underbrace{1100}_{12}\ \underbrace{0011}_{3}$

  $$14019P = 16\left(16\left(16\left(3P\right)+6P\right)+12P\right)+3P.$$

  Same number of doublings, 3 additions.

Timings of scalar multiplication on NIST P-256



Larger windows reduce the variability through branching but accentuate the length.

(Picture from TPM-Fail)