Singular curve point decompression attack

Peter Günther

joint work with

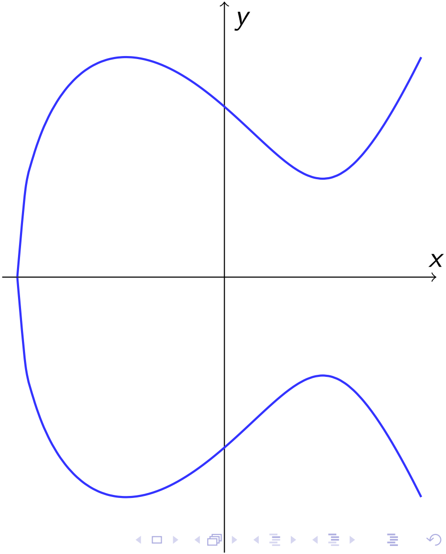
Johannes Blömer

University of Paderborn

FDTC 2015, September 13th, Saint Malo

Elliptic curves

Example: $E(\mathbb{R}) : y^2 = x^3 - 3x + 3$



Elliptic curve $E(\mathbb{K})$

Points $(x, y) \in \mathbb{K}^2$ that fulfill

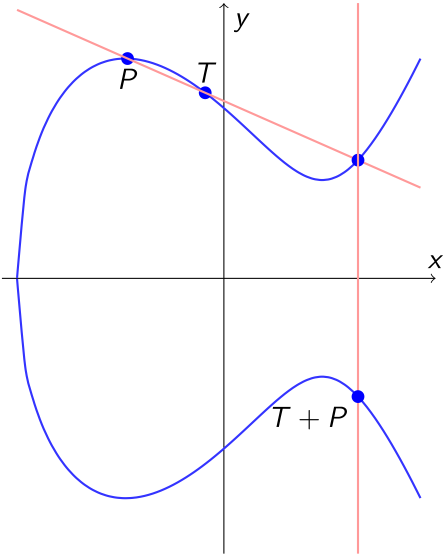
$$y^2 = x^3 + a_4 x + a_6$$

with $a_4, a_6 \in \mathbb{K}$ and discriminant

$$\Delta := -16(4a_4^3 + 27a_6^2) \neq 0.$$

Elliptic curves as additive group

Example: $E(\mathbb{R}) : y^2 = x^3 - 3x + 3$

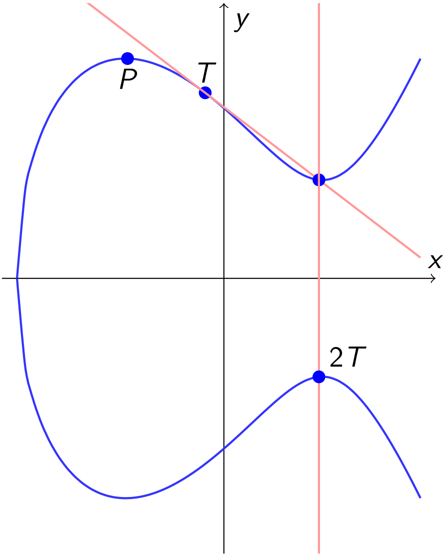


Group operation independent from a_6:

$$\lambda = \frac{y_P - y_T}{x_P - x_T}$$
$$x_{P+T} = \lambda^2 - x_P - x_T$$
$$y_{P+T} = \lambda(x_P - x_{P+T}) - y_P$$

Elliptic curves as additive group

Example: $E(\mathbb{R}) : y^2 = x^3 - 3x + 3$



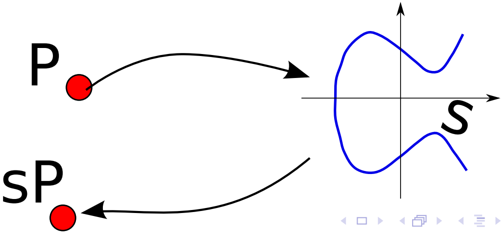
Group operation independent from a_6:

$$\lambda = \frac{3x_T + a_4}{2y_T}$$
$$x_{2T} = \lambda^2 - 2x_T$$
$$y_{2T} = \lambda(x_T - x_{2T}) - y_T$$

Elliptic curve scalar multiplication and DLOG

- Scalar multiplication: $s \in \mathbb{N}, P \in E(\mathbb{F}_q)$:

$$sP := P + P + \cdots + P \text{ (} s \text{ times)}$$

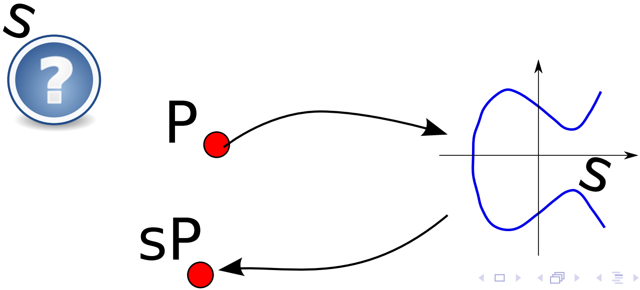


Elliptic curve scalar multiplication and DLOG

- Scalar multiplication: $s \in \mathbb{N}, P \in E(\mathbb{F}_q)$:

$$sP := P + P + \cdots + P \text{ (}s\text{ times)}$$

- Discrete logarithm (DLOG): given $P, Q = sP$, compute s
- Assumption: complexity of DLOG problem exponential on elliptic curve
  $\Rightarrow$ high security already for small $\mathbb{F}_q$ (e.g. 256 bit)

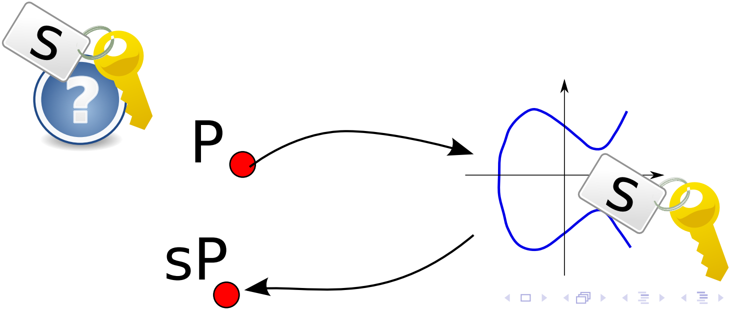


Elliptic curve scalar multiplication and DLOG

- Scalar multiplication: $s \in \mathbb{N}, P \in E(\mathbb{F}_q)$:

$$sP := P + P + \cdots + P \text{ (s times)}$$

- Discrete logarithm (DLOG): given $P, Q = sP$, compute s
- Assumption: complexity of DLOG problem exponential on elliptic curve $\Rightarrow$ high security already for small $\mathbb{F}_q$ (e.g. 256 bit)
- Important cryptographic primitive (ECDH, ECDSA, ...)

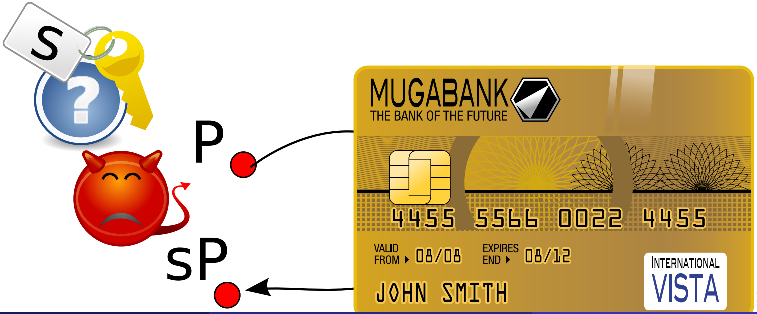


Elliptic curve scalar multiplication and DLOG

- Scalar multiplication: $s \in \mathbb{N}, P \in E(\mathbb{F}_q)$:

$$sP := P + P + \cdots + P \text{ (} s \text{ times)}$$

- Discrete logarithm (DLOG): given $P, Q = sP$, compute s
- Assumption: complexity of DLOG problem exponential on elliptic curve $\Rightarrow$ high security already for small $\mathbb{F}_q$ (e.g. 256 bit)
- Important cryptographic primitive (ECDH, ECDSA, . . .)
- Adversarial environment: Physical protection of s required



Invalid point attack on scalar multiplication

$E : y^2 = x^3 + a_4x + a_6$

Outline of invalid point attacks

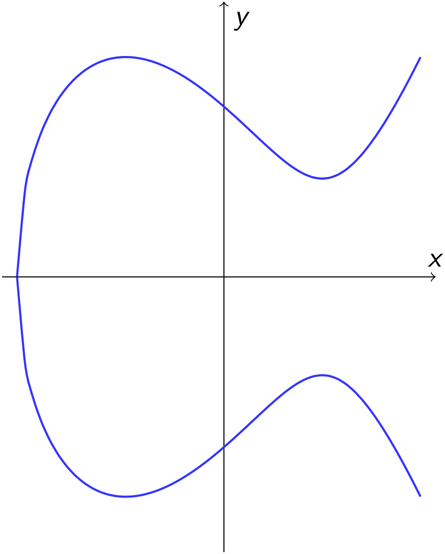
1. Group law does not require a_6
2. Move P to weak curve with same a_4
3. Obtain $Q = sP$ for secret s on weak curve
4. Compute DLOG of Q to base P on weak curve
5. Infer DLOG s on original curve

Examples weak curve attacks

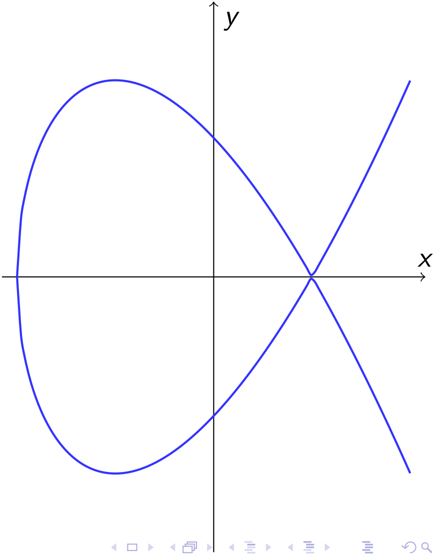
- P on curve with smooth order
- P in small subgroup
- P on singular curve

Singular curves with node ($a_4 \neq 0$)

$E(\mathbb{R}) : y^2 = x^3 - 3x + 3$

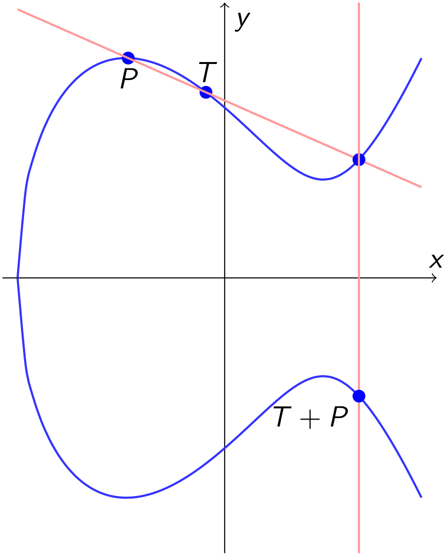


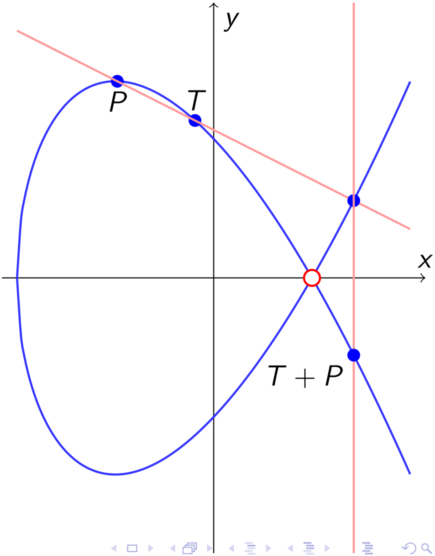
$E(\mathbb{R}) : y^2 = x^3 - 3x + 2,\ \Delta = 0$



Singular curves with node ($a_4 \neq 0$)

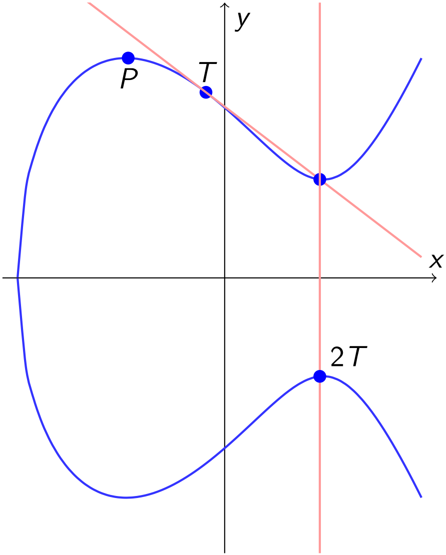
$E(\mathbb{R}) : y^2 = x^3 - 3x + 3$ $E(\mathbb{R}) : y^2 = x^3 - 3x + 2,\ \Delta = 0$



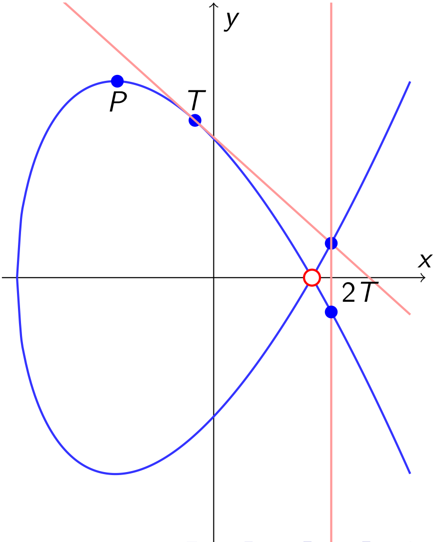


Singular curves with node ($a_4 \neq 0$)

$E(\mathbb{R}) : y^2 = x^3 - 3x + 3$

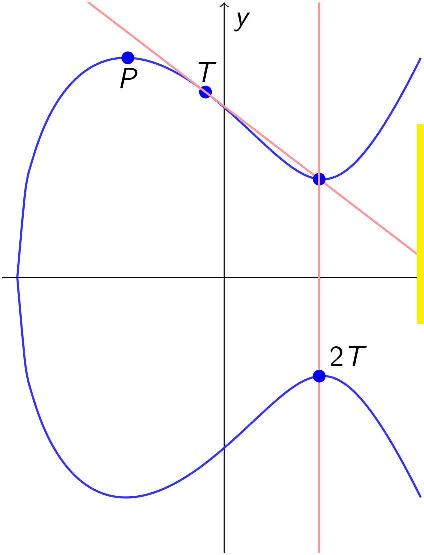


$E(\mathbb{R}) : y^2 = x^3 - 3x + 2,\ \Delta = 0$

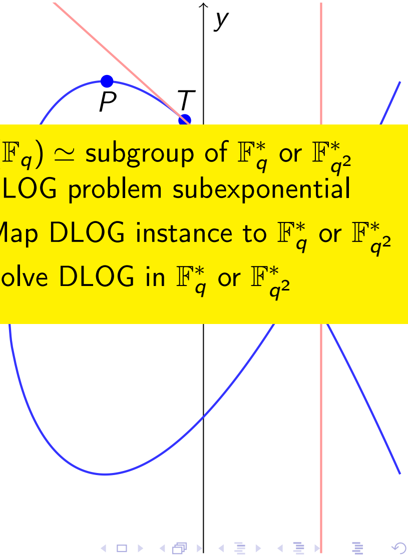


Singular curves with node ($a_4 \neq 0$)

$E(\mathbb{R}) : y^2 = x^3 - 3x + 3$



$E(\mathbb{R}) : y^2 = x^3 - 3x + 2,\ \Delta = 0$

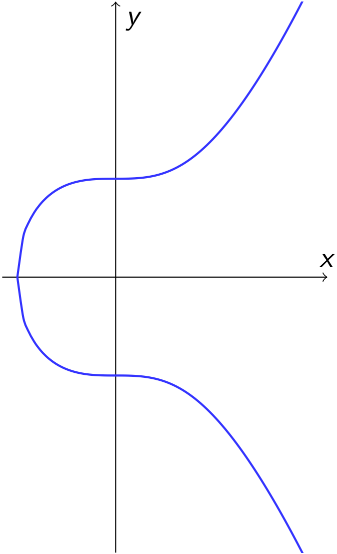


$E_{NS}(\mathbb{F}_q) \simeq$ subgroup of $\mathbb{F}_q^*$ or $\mathbb{F}_{q^2}^*$
$\Rightarrow$ DLOG problem subexponential

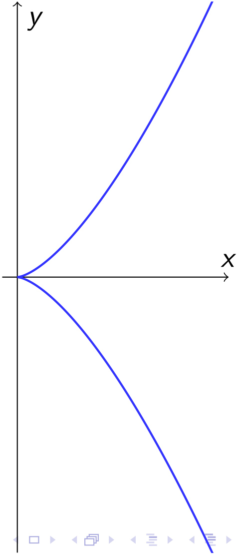
1. Map DLOG instance to $\mathbb{F}_q^*$ or $\mathbb{F}_{q^2}^*$
2. Solve DLOG in $\mathbb{F}_q^*$ or $\mathbb{F}_{q^2}^*$

Singular curves with cusp ($a_4 = 0$)

$E(\mathbb{R}) : y^2 = x^3 + 1$

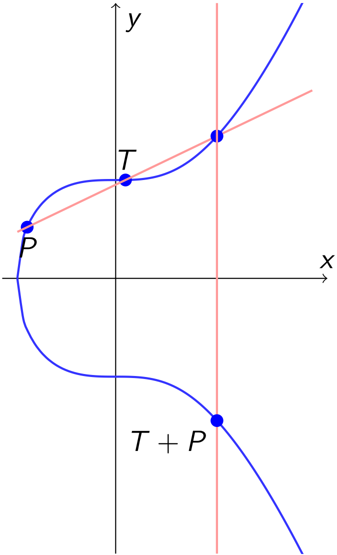


$E(\mathbb{R}) : y^2 = x^3$, $\Delta = 0$

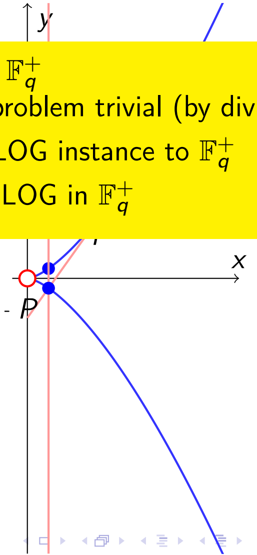


Singular curves with cusp ($a_4 = 0$)

$E(\mathbb{R}) : y^2 = x^3 + 1$



$E(\mathbb{R}) : y^2 = x^3$, $\Delta = 0$

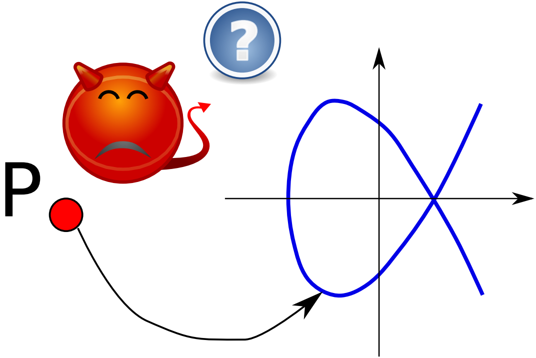


$E_{NS}(\mathbb{F}_q) \simeq \mathbb{F}_q^+$
$\Rightarrow$ DLOG problem trivial (by division)

1. Map DLOG instance to $\mathbb{F}_q^+$
2. Solve DLOG in $\mathbb{F}_q^+$

Singular curve attack on scalar multiplication

- For fixed a_4, there are at most 2 corresponding singular curves
- Random faults will not provide points on singular curve
- How do we get a point onto one of them?



Our approach: Point decompression

Compression

$$\begin{aligned} \text{Compress} : E(\mathbb{F}_q) &\rightarrow \mathbb{F}_q \times \{0,1\} \\ (x,y) &\mapsto (x,b) \text{ where } b = \text{LSB}(y) \end{aligned}$$

- Reduces bandwidth by 50%
- Defined in many standards like IEEE 1363, SEC 1, X9.62
- Decompression prior to scalar multiplication

Point compression

Decompress

```
Require: $E : y^2 = x^3 + a_4x + a_6$, $(x, b) \in \mathbb{F}_q \times \{0, 1\}$
Ensure: $(x, y)$ with $y^2 = x^3 + a_4x + a_6$
$v \leftarrow x^3 + a_4x$                          ▷ $v = x^3 + a_4x$
$v \leftarrow v + a_6$                             ▷ $v = x^3 + a_4x + a_6$
if $\sqrt{v} \in \mathbb{F}_q$ then
    $v \leftarrow (-1)^b\sqrt{v}$                  ▷ $v = (-1)^b\sqrt{x^3 + a_4x + a_6}$
    return $(x, y)$
else
    return $\mathcal{O}$
end if
```

Point compression

Decompress

```
Require: $E$
Ensure: $(x,$
$v \leftarrow x^3$
$v \leftarrow v +$                                                    $+ a_6$
if $\sqrt{v} \in$
    $v \leftarrow (-1)^b\sqrt{v}$                    ▷ $v = (-1)^b\sqrt{x^3 \qquad + a_6}$
    return $(x, y)$
else
    return $\mathcal{O}$
end if
```

- Similar implementations in IEEE 1363, SEC 1, X9.62, OpenSSL
- Implicit (partial) point validation: $\text{Decompress}(x, b) \in E(\mathbb{F}_q)$

Attack on decompression

Decompress

```
Require: $E : y^2 = x^3 + a_4x + a_6$, $(x, b) \in \mathbb{F}_q \times \{0, 1\}$
Ensure: $(x, y)$ with $y^2 = x^3 + a_4x + a_6$
$v \leftarrow x^3 + a_4x$                          $\triangleright\ v = x^3 + a_4x$
$v \leftarrow v + a_6$                             $\triangleright\ v = x^3 + a_4x + a_6$
if $\sqrt{v} \in \mathbb{F}_q$ then
    $v \leftarrow (-1)^b\sqrt{v}$                  $\triangleright\ v = (-1)^b\sqrt{x^3 + a_4x + a_6}$
    return $(x, y)$
else
    return $\mathcal{O}$
end if
```

Attack on decompression

Decompress with $a_4 = 0$

```
Require: $E : y^2 = x^3 + a_4 x + a_6$, $(x, b) \in \mathbb{F}_q \times \{0, 1\}$
Ensure: $(x, y)$ with $y^2 = x^3 \qquad + a_6$
$v \leftarrow x^3$                                  ▷ $v = x^3$
$v \leftarrow v + a_6$                              ▷ $v = x^3 \qquad + a_6$
if $\sqrt{v} \in \mathbb{F}_q$ then
    $v \leftarrow (-1)^b \sqrt{v}$                  ▷ $v = (-1)^b \sqrt{x^3 \qquad + a_6}$
    return $(x, y)$
else
    return $\mathcal{O}$
end if
```

Attack on decompression

Decompress with $a_4 = 0$ and with fault

Require: $E : y^2 = x^3 + a_4x + a_6$, $(x, b) \in \mathbb{F}_q \times \{0, 1\}$

Ensure: (x, y) with $y^2 = x^3 \cancel{+ a_6}$

```
$v \leftarrow x^3$                                   ▷ $v = x^3$
$v \leftarrow v \cancel{+ a_6}$                      ▷ $v = x^3 \cancel{+ a_6}$
if $\sqrt{v} \in \mathbb{F}_q$ then
    $v \leftarrow (-1)^b \sqrt{v}$                   ▷ $v = (-1)^b \sqrt{x^3 \cancel{+ a_6}}$
    return $(x, y)$
else
    return $\mathcal{O}$
end if
```

Attack on decompression

Decompress with $a_4 = 0$ and with fault

Require: $E : y^2 = x^3 + a_4x + a_6$, $(x, b) \in \mathbb{F}_q \times \{0, 1\}$

Ensure: (x, y) with $y^2 = x^3$ ~~$+ a_6$~~

```
v ← $x^3$                                        ▷ $v = x^3$
v ← $v \cancel{+ a_6}$                           $= x^3 \cancel{+ a_6}$
if $\sqrt{v} \in \mathbb{F}_q$ then
    v ← $(-1)^b\sqrt{v}$                         $^b\sqrt{x^3 \cancel{+ a_6}}$
    return $(x, y)$
else
    return $\mathcal{O}$
end if
```

Observation:
x quadratic residue ⇒
output on singular curve
$y^2 = x^3$

Hash string to curve

Decompress: building block of other algorithms

MapToPoint : $\{0,1\}^* \rightarrow E(\mathbb{F}_q)$

```
Require: $E : y^2 = x^3 + a_4 x + a_6$, $\mathrm{H} : \{0,1\}^* \rightarrow \mathbb{F}_q \times \{0,1\}$, $M \in \{0,1\}^*$,
Ensure: $P \in E(\mathbb{F}_q)$
i ← 0
repeat                                   ▷ until $(x, b)$ is valid compression
    $(x, b) \leftarrow \mathrm{H}(M \parallel i)$
    $P \leftarrow \mathrm{Decompress}(x, b)$
    $i \leftarrow i + 1$
until $P \neq \mathcal{O}$
return $P$
```

Hash string to curve

Decompress: building block of other algorithms

MapToPoint : $\{0,1\}^* \to E(\mathbb{F}_q)$

Require: $E : y^2 = x^3 + a_4 x + a_6$, $\mathrm{H} : \{0,1\}^* \to \mathbb{F}_q \times \{0,1\}$, $M \in \{0,1\}^*$,
Ensure: $P \in E(\mathbb{F}_q)$

```
$i \leftarrow 0$
repeat
    $(x, b) \leftarrow \mathrm{H}(M \parallel i)$
    $P \leftarrow \mathrm{Decompress}(x, b)$
    $i \leftarrow i + 1$
until $P \neq \mathcal{O}$
return $P$
```

pression

Atttack:
Choose M such that
$\mathrm{H}(M \parallel 0) = (x, b)$ with
quadratic residue x.

Properties of the attack

Features of the attack

- Efficient, especially in the case $a_4 = 0$
- One shot: can be applied to exponentiation with nonce
- Applications:
  - Point decompression (encryption schemes)
  - Hashing to curve (special signature schemes)
  - Random point sampling (countermeasures)

Limitations of the attack

- Access to $Q = sP$ required
- For $a_4 \neq 0$: stronger control over (x, b) required
  - Attack on plain Decompress still possible
  - Attack on MapToPoint not possible

Example application: BLS short signatures

Definition (BLS Signatures)

- $\mathbb{G}_1, \mathbb{G}_2 \subseteq E(\mathbb{F}_q)$: cyclic groups of order r with generators P_1 and P_2
- $\mathbb{G}_T \subseteq \mathbb{F}_q^*$ cyclic group of order r
- pairing $\mathrm{e} : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$
- $\mathrm{MapToPoint} : \{0,1\}^* \to E(\mathbb{F}_q)$

- **KeyGen**$(\,\cdot\,)$:
  1. Select s uniformly at random from $[0, r-1]$
  2. Output secret key s and public key $P_s = sP_2$
- **Sign**(M, s):
  1. compute $P = \mathrm{MapToPoint}(M) \in \mathbb{G}_1$
  2. compute and output $\sigma = sP$ as signature for M under s
- **Verify**(M, σ, P_s): output 1 if and only if

$$\mathrm{e}(\sigma, P_2) = \mathrm{e}(\mathrm{MapToPoint}(M), P_s).$$

Example application: BLS short signatures

Definition (BLS Signatures)

- $\mathbb{G}_1, \mathbb{G}_2 \subseteq E(\mathbb{F}_q)$: cyclic groups of order r with generators P_1 and P_2
- $\mathbb{G}_T \subseteq \mathbb{F}_q^*$ cyclic group of order r
- pairing $\mathrm{e} : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$
- $\mathrm{MapToPoint} : \{0,1\}^* \to E(\mathbb{F}_q)$

- **KeyGen**(·):
  1. Select s uniformly at random from $[0, r-1]$
  2. Output secret key s and public key $P_s = sP_2$
- **Sign**(M, s):
  1. compute $P = \mathrm{MapToPoint}(M) \in \mathbb{G}_1$
  2. compute and output $\sigma = sP$ as signature for M under s
- **Verify**(M, σ, P_s): output 1 if and only if

$$\mathrm{e}(\sigma, P_2) = \mathrm{e}(\mathrm{MapToPoint}(M), P_s).$$

Example application: BLS short signatures

Definition (BLS Signatures)

- $\mathbb{G}_1, \mathbb{G}_2 \subseteq E(\mathbb{F}_q)$: cyclic groups of order r with generators P_1 and P_2
- $\mathbb{G}_T \subseteq \mathbb{F}_q^*$ cyc
- pairing e : $\mathbb{G}_1$
- MapToPoint

Very efficient with Barreto-Naehrig (BN) curves:

$$E : y^2 = x^3 + a_6$$

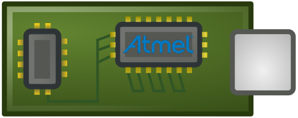
Note: $a_4 = 0$

- **KeyGen**$(\cdot)$:
  1. Select s uniformly at random from $[0, r-1]$
  2. Output secret key s and public key $P_s = sP_2$
- **Sign**(M, s):
  1. compute $P = \text{MapToPoint}(M) \in \mathbb{G}_1$
  2. compute and output $\sigma = sP$ as signature for M under s
- **Verify**(M, σ, P_s): output 1 if and only if

$$\text{e}(\sigma, P_2) = \text{e}(\text{MapToPoint}(M), P_s).$$

Attack: Proof of concept realization

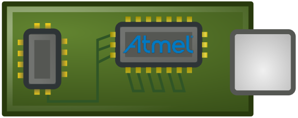
Target: BLS short signatures of Relic toolkit on AVR



- Target hardware: Atmel AVR Xmega A1
- Target software: Relic toolkit
    - Open source
    - Prime and binary field arithmetic
    - NIST and pairing-friendly curves including BN curves
    - Bilinear maps and related extension fields
    - Cryptographic protocols including BLS short signatures
- Attack: Second order instruction skip attack
    - First fault: decompress to singular curve
    - Second fault: remove point validation countermeasure

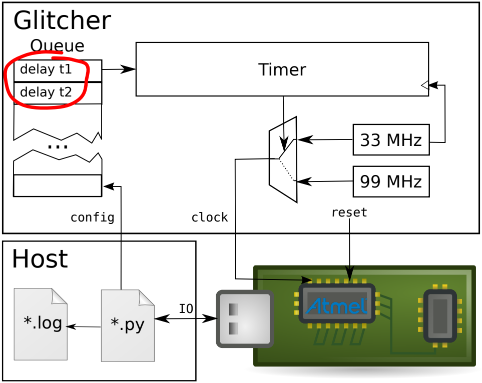
Attack: Proof of concept realization

Target: BLS short signatures of Relic toolkit on AVR

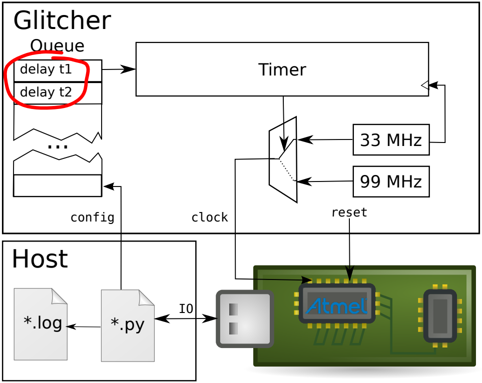


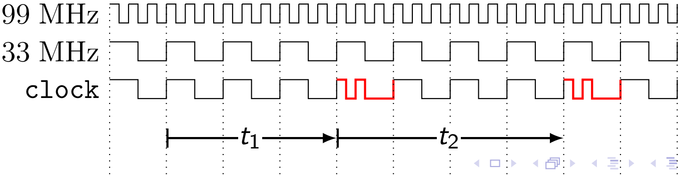
- Target hardware: Atmel AVR Xmega A1
- Target software: Relic toolkit
  - Open source
  - Prime and binary field arithmetic
  - NIST and pairing-friendly curves including BN curves
  - Bilinear maps and related extension fields
  - Cryptographic protocols including BLS short signatures
- Attack: Second order instruction skip attack
  - First fault: decompress to singular curve
  - Second fault: remove point validation countermeasure

Instruction skips via clock glitching



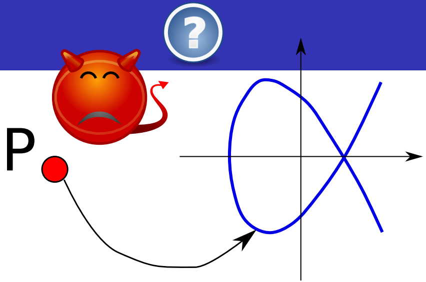
Instruction skips via clock glitching





The RELIC implementation

First fault: move to singular curve

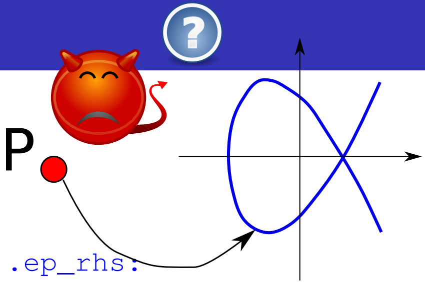


Decompress:

```
Require: $E : y^2 = x^3 + a_4x + a_6$,
    $(x, b) \in \mathbb{F}_q \times \{0, 1\}$
Ensure: $(x, y)$ with $y^2 = x^3 + a_4x + a_6$
v ← $x^3 + a_4x$
v ← $v + a_6$
if $\sqrt{v} \in \mathbb{F}_q$ then
    $v \leftarrow (-1)^b\sqrt{v}$
    return $(x, y)$
else
    return $\mathcal{O}$
end if
```

The RELIC implementation

First fault: move to singular curve



BN-curve: $E : y^2 = x^3 + 17$, note: $a_4 = 0$

Decompress:

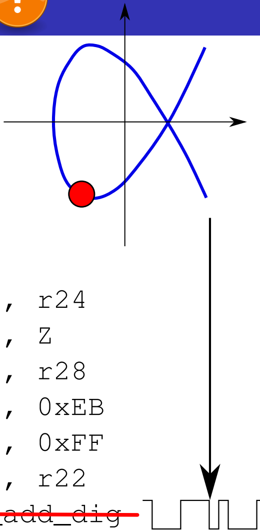
```
Require: $E : y^2 = x^3 + a_4 x + a_6$,
    $(x, b) \in \mathbb{F}_q \times \{0, 1\}$
Ensure: $(x, y)$ with $y^2 = x^3 \qquad + a_6$
v ← $x^3$
v ← $v + a_6$
if $\sqrt{v} \in \mathbb{F}_q$ then
    v ← $(-1)^b \sqrt{v}$
    return $(x, y)$
else
    return $\mathcal{O}$
end if
```

avr-gcc

```
.ep_rhs:
...
movw   r30, r24
ld     r20, Z
movw   r22, r28
subi   r22, 0xEB
sbci   r23, 0xFF
movw   r24, r22
call   fp_add_dig
rjmp   .+18
...
```

The RELIC implementation

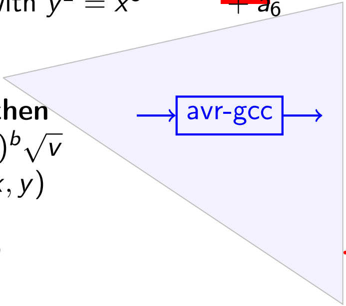
First fault: move to singular curve



BN-curve: $E : y^2 = x^3 + 17$, note: $a_4 = 0$

Decompress:

```
Require: $E : y^2 = x^3 + a_4 x + a_6$,
    $(x, b) \in \mathbb{F}_q \times \{0, 1\}$
Ensure: $(x, y)$ with $y^2 = x^3 \cancel{+ a_6}$
v ← $x^3$
v ← $v \cancel{+ a_6}$
if $\sqrt{v} \in \mathbb{F}_q$ then
    $v \leftarrow (-1)^b \sqrt{v}$
    return $(x, y)$
else
    return $\mathcal{O}$
end if
```



```
.ep_rhs:
...
movw   r30, r24
ld     r20, Z
movw   r22, r28
subi   r22, 0xEB
sbci   r23, 0xFF
movw   r24, r22
~~call   fp_add_dig~~
rjmp   .+18
...
```

References

- Relic toolkit:
  `https://github.com/relic-toolkit`
- Glitcher Die Datenkrake:
  `https://www.usenix.org/conference/woot13/workshop-program/presentation/nedospasov`