Degenerate Curve Attacks

Extending Invalid Curve Attacks to Edwards Curves and Other Models

Samuel Neves   Mehdi Tibouchi

PKC 2016

What this paper is about

- We look at a certain type of implementation attacks against elliptic curves, known as "invalid curve attacks"

What this paper is about

- We look at a certain type of implementation attacks against elliptic curves, known as “invalid curve attacks”
- Known to apply to curves in Weierstrass form, as well as a few other (Hessian curves, Huff’s model, etc.)

What this paper is about

- We look at a certain type of implementation attacks against elliptic curves, known as “invalid curve attacks”
- Known to apply to curves in Weierstrass form, as well as a few other (Hessian curves, Huff’s model, etc.)
- However, no attack of that type known so far against Edwards curves (when using the more common addition law), Jacobi intersections, Jacobi quartics...

What this paper is about

- We look at a certain type of implementation attacks against elliptic curves, known as "invalid curve attacks"
- Known to apply to curves in Weierstrass form, as well as a few other (Hessian curves, Huff's model, etc.)
- However, no attack of that type known so far against Edwards curves (when using the more common addition law), Jacobi intersections, Jacobi quartics...
- Our main contribution: showing that there is still something you can do to break (sloppy implementations of) those curve models as well

What this paper is about

- We look at a certain type of implementation attacks against elliptic curves, known as "invalid curve attacks"
- Known to apply to curves in Weierstrass form, as well as a few other (Hessian curves, Huff's model, etc.)
- However, no attack of that type known so far against Edwards curves (when using the more common addition law), Jacobi intersections, Jacobi quartics...
- Our main contribution: showing that there is still something you can do to break (sloppy implementations of) those curve models as well
- On the flip side: the attack can be repurposed as a cheap fault attack countermeasure in some settings

A basic model for invalid curve attacks

- Imagine a cryptographic device doing scalar multiplications on a prescribed elliptic curve E

A basic model for invalid curve attacks

- Imagine a cryptographic device doing scalar multiplications on a prescribed elliptic curve E
- You send it a point P on E, and it outputs $s \cdot P$ for some secret scalar s

A basic model for invalid curve attacks

- Imagine a cryptographic device doing scalar multiplications on a prescribed elliptic curve E
- You send it a point P on E, and it outputs $s \cdot P$ for some secret scalar s
- If you follow the rules: cannot learn much about s

A basic model for invalid curve attacks

- Imagine a cryptographic device doing scalar multiplications on a prescribed elliptic curve E
- You send it a point P on E, and it outputs $s \cdot P$ for some secret scalar s
- If you follow the rules: cannot learn much about s
- So you cheat: send a point $\widetilde{P}$ outside of E

A basic model for invalid curve attacks

- Imagine a cryptographic device doing scalar multiplications on a prescribed elliptic curve E
- You send it a point P on E, and it outputs $s \cdot P$ for some secret scalar s
- If you follow the rules: cannot learn much about s
- So you cheat: send a point $\widetilde{P}$ outside of E
- What happens then?

Sending an invalid point

- If the device is properly designed, it might check and reject your input (point validation)

Sending an invalid point

- If the device is properly designed, it might check and reject your input (point validation)
- You might even think most implementations would do so; that would be optimistic

Sending an invalid point

- If the device is properly designed, it might check and reject your input (point validation)
- You might even think most implementations would do so; that would be optimistic
- Without point validation: the device runs the same sequence of operations as for a point on E, and returns the result

Sending an invalid point

- If the device is properly designed, it might check and reject your input (point validation)
- You might even think most implementations would do so; that would be optimistic
- Without point validation: the device runs the same sequence of operations as for a point on E, and returns the result
- What you get depends on the precise way the arithmetic on E is implemented

Example: short Weierstrass/affine

- Say the device does its scalar multiplications
  - using double-and-add
  - on the short Weierstrass curve $E \colon y^2 = x^3 + ax + b$
  - using the affine coordinate addition law $(x_1, y_1) + (x_2, y_2) = (x_3, y_3)$:

$$x_3 = \lambda^2 - x_1 - x_2 \qquad y_3 = \lambda(x_1 - x_3) - y_1$$

where

$$\lambda = \begin{cases} (3x_1^2 + a)/(2y_1) & \text{if } (x_1, y_1) = (x_2, y_2) \text{ (doubling)} \\ (y_1 - y_2)/(x_1 - x_2) & \text{otherwise} \end{cases}$$

Example: short Weierstrass/affine

- Say the device does its scalar multiplications
  - using double-and-add
  - on the short Weierstrass curve $E\colon y^2 = x^3 + ax + b$
  - using the affine coordinate addition law $(x_1, y_1) + (x_2, y_2) = (x_3, y_3)$:

$$x_3 = \lambda^2 - x_1 - x_2 \qquad y_3 = \lambda(x_1 - x_3) - y_1$$

  where

$$\lambda = \begin{cases} (3x_1^2 + a)/(2y_1) & \text{if } (x_1, y_1) = (x_2, y_2) \text{ (doubling)} \\ (y_1 - y_2)/(x_1 - x_2) & \text{otherwise} \end{cases}$$

- Key observation (Biehl et al.): the addition and doubling formulas depend only on curve parameter a. Identical for all curves of the form $\widetilde{E}\colon y^2 = x^3 + ax + \widetilde{b}$.

Example: short Weierstrass/affine

- The invalid point $\widetilde{P}$ that you sent is on one of the curves $\widetilde{E}$

Example: short Weierstrass/affine

- The invalid point $\widetilde{P}$ that you sent is on one of the curves $\widetilde{E}$
- So the sequence of operations carried out by the device exactly ends up being a scalar multiplication on that curve: it outputs $s \cdot \widetilde{P}$ on $\widetilde{E}$

Example: short Weierstrass/affine

- The invalid point $\widetilde{P}$ that you sent is on one of the curves $\widetilde{E}$
- So the sequence of operations carried out by the device exactly ends up being a scalar multiplication on that curve: it outputs $s \cdot \widetilde{P}$ on $\widetilde{E}$
- If the curve $\widetilde{E}$ is weak (almost all curves are!), you can recover plenty of information on s: e.g. you get $s \bmod \ell$ for any small divisor ℓ of the order

How general is this?

- The description above captures the gist of Antipa et al.'s invalid curve attack (PKC 2003)

How general is this?

- The description above captures the gist of Antipa et al.'s invalid curve attack (PKC 2003)
- Also works with other coordinate systems (projective coordinates, etc.), and doesn't really depend on the scalar multiplication algorithm

How general is this?

- The description above captures the gist of Antipa et al.'s invalid curve attack (PKC 2003)
- Also works with other coordinate systems (projective coordinates, etc.), and doesn't really depend on the scalar multiplication algorithm
- Not just Weierstrass: applies as long as the arithmetic is independent of at least one curve parameter (Hessian curves, Huff curves)

How general is this?

- The description above captures the gist of Antipa et al.'s invalid curve attack (PKC 2003)
- Also works with other coordinate systems (projective coordinates, etc.), and doesn't really depend on the scalar multiplication algorithm
- Not just Weierstrass: applies as long as the arithmetic is independent of at least one curve parameter (Hessian curves, Huff curves)
- However, the (preferred) addition and doubling formulas for Edwards curves and a few others depend on all curve parameters. What about them?

Edwards curves and invalid points

- The (complete) addition law on the twisted Edwards curve $E\colon ax^2 + y^2 = 1 + dx^2y^2$ is given by:

$$(x_1, y_1) + (x_2, y_2) = \left(\frac{x_1y_2 + y_1x_2}{1 + dx_1x_2y_1y_2}, \frac{y_1y_2 - ax_1x_2}{1 - dx_1x_2y_1y_2} \right)$$

Edwards curves and invalid points

- The (complete) addition law on the twisted Edwards curve $E: ax^2 + y^2 = 1 + dx^2y^2$ is given by:

$$(x_1, y_1) + (x_2, y_2) = \left(\frac{x_1y_2 + y_1x_2}{1 + dx_1x_2y_1y_2}, \frac{y_1y_2 - ax_1x_2}{1 - dx_1x_2y_1y_2} \right)$$

- If you send an invalid point $\widetilde{P}$, the computations won't make sense: $\widetilde{P}$ lies on different twisted Edwards curves, but none with the same arithmetic formulas

Edwards curves and invalid points

- The (complete) addition law on the twisted Edwards curve $E\colon ax^2 + y^2 = 1 + dx^2y^2$ is given by:

$$(x_1, y_1) + (x_2, y_2) = \left(\frac{x_1 y_2 + y_1 x_2}{1 + d x_1 x_2 y_1 y_2}, \frac{y_1 y_2 - a x_1 x_2}{1 - d x_1 x_2 y_1 y_2} \right)$$

- If you send an invalid point $\widetilde{P}$, the computations won't make sense: $\widetilde{P}$ lies on different twisted Edwards curves, but none with the same arithmetic formulas
- The device will output something, but it won't be the scalar multiplication by s in some group anymore: hard to exploit

Edwards curves and invalid points

- The (complete) addition law on the twisted Edwards curve $E\colon ax^2 + y^2 = 1 + dx^2y^2$ is given by:

$$(x_1, y_1) + (x_2, y_2) = \left(\frac{x_1y_2 + y_1x_2}{1 + dx_1x_2y_1y_2}, \frac{y_1y_2 - ax_1x_2}{1 - dx_1x_2y_1y_2}\right)$$

- If you send an invalid point $\widetilde{P}$, the computations won't make sense: $\widetilde{P}$ lies on different twisted Edwards curves, but none with the same arithmetic formulas
- The device will output something, but it won't be the scalar multiplication by s in some group anymore: hard to exploit
- In fact, Antipa et al. suggest using arithmetic depending on all curve parameters as a possible countermeasure against invalid curve attacks

NTT

Our main observation

- What I just said about computations not being in a group: true for almost all invalid points $\widetilde{P}$...

Our main observation

- What I just said about computations not being in a group: true for almost all invalid points $\widetilde{P}$...
- ...but there are exceptions: take points of the form $(0, y)$. Applying the formula naively we get:

$$(0, y_1) + (0, y_2) = \left(\frac{0 \cdot y_2 + y_1 \cdot 0}{1 + d \cdot 0 \cdot y_1 y_2}, \frac{y_1 y_2 - a \cdot 0}{1 - d \cdot 0 \cdot y_1 y_2} \right) = (0, y_1 y_2)$$

Our main observation

- What I just said about computations not being in a group: true for almost all invalid points $\widetilde{P}$...
- ...but there are exceptions: take points of the form $(0, y)$. Applying the formula naively we get:

$$(0, y_1) + (0, y_2) = \left(\frac{0 \cdot y_2 + y_1 \cdot 0}{1 + d \cdot 0 \cdot y_1 y_2}, \frac{y_1 y_2 - a \cdot 0}{1 - d \cdot 0 \cdot y_1 y_2} \right) = (0, y_1 y_2)$$

- It easily follows that if you send $(0, y)$ to the device, it will output $(0, y^s)$

Our main observation

- What I just said about computations not being in a group: true for almost all invalid points $\widetilde{P}$...
- ...but there are exceptions: take points of the form $(0, y)$. Applying the formula naively we get:

$$(0, y_1) + (0, y_2) = \left(\frac{0 \cdot y_2 + y_1 \cdot 0}{1 + d \cdot 0 \cdot y_1 y_2}, \frac{y_1 y_2 - a \cdot 0}{1 - d \cdot 0 \cdot y_1 y_2} \right) = (0, y_1 y_2)$$

- It easily follows that if you send $(0, y)$ to the device, it will output $(0, y^s)$
- And so you can recover s by solving a discrete log problem in the multiplicative group of the base field: comparatively very easy!

Does this trick generalize?

- After making that observation, we looked at all other models of elliptic curves we could find

Does this trick generalize?

- After making that observation, we looked at all other models of elliptic curves we could find
- In all cases, similar special invalid points for which the original addition law becomes multiplication in a group isomorphic to $\mathbb{F}_p^*$ or to the twisted multiplicative group

Does this trick generalize?

- After making that observation, we looked at all other models of elliptic curves we could find
- In all cases, similar special invalid points for which the original addition law becomes multiplication in a group isomorphic to $\mathbb{F}_p^*$ or to the twisted multiplicative group
- So the attack seems to apply basically to all curve models

Does this trick generalize?

- After making that observation, we looked at all other models of elliptic curves we could find
- In all cases, similar special invalid points for which the original addition law becomes multiplication in a group isomorphic to $\mathbb{F}_p^*$ or to the twisted multiplicative group
- So the attack seems to apply basically to all curve models
- Like the Antipa et al. attack, mostly unaffected by different coordinate systems or scalar multiplication algorithms

NTT

Degenerate curve attacks

- In the Edwards curve setting, the set of points $(0, y)$ (i.e. the axis $x = 0$) can be seen as the "limit" of the curves $E_{a,d}$ when $a \to \infty$, so it should still have a natural group law, which must necessarily be a form of the multiplicative group

Degenerate curve attacks

- In the Edwards curve setting, the set of points $(0, y)$ (i.e. the axis $x = 0$) can be seen as the "limit" of the curves $E_{a,d}$ when $a \to \infty$, so it should still have a natural group law, which must necessarily be a form of the multiplicative group
- However, the group laws in the family are all different, so no a priori reason why the law on the limit should be the same as the addition law you started with

Degenerate curve attacks

- In the Edwards curve setting, the set of points $(0, y)$ (i.e. the axis $x = 0$) can be seen as the "limit" of the curves $E_{a,d}$ when $a \to \infty$, so it should still have a natural group law, which must necessarily be a form of the multiplicative group
- However, the group laws in the family are all different, so no a priori reason why the law on the limit should be the same as the addition law you started with
- But the terms that depend on the curve parameters all cancel out at the limit, so you end up being fine somehow

Degenerate curve attacks

- In the Edwards curve setting, the set of points $(0, y)$ (i.e. the axis $x = 0$) can be seen as the "limit" of the curves $E_{a,d}$ when $a \to \infty$, so it should still have a natural group law, which must necessarily be a form of the multiplicative group
- However, the group laws in the family are all different, so no a priori reason why the law on the limit should be the same as the addition law you started with
- But the terms that depend on the curve parameters all cancel out at the limit, so you end up being fine somehow
- More generally, the set of special invalid points that let you attack are where your curve families degenerate, hence the name

Is this a realistic threat?

- Regarding concrete impact, mainly two aspects to consider

Is this a realistic threat?

- Regarding concrete impact, mainly two aspects to consider
- Are implementers of Edwards curves as likely to mess up point validation?
  - Not by a long shot
  - The main implementations are by notoriously competent people
  - Specifications being written mandate compressed representations for points on the wire, thwarting the attack

Is this a realistic threat?

- Regarding concrete impact, mainly two aspects to consider
- Are implementers of Edwards curves as likely to mess up point validation?
  - Not by a long shot
  - The main implementations are by notoriously competent people
  - Specifications being written mandate compressed representations for points on the wire, thwarting the attack
- Is the model of a device computing scalar multiplications realistic?
  - Not very but close to static DH key exchange
  - More realistic model: don't get the output point, only a hash
  - Addressed in the paper. Recovering all of s possible but more costly than Antipa et al., because only one group to play with

A constructive application

- Quick idea of our constructive use of this attack

A constructive application

- Quick idea of our constructive use of this attack
- Common trick to protect against fault injection in a device doing computations over $\mathbb{F}_p$:
  1. choose a small auxiliary prime r, and compute mod $p \cdot r$
  2. redo the computation mod r
  3. if the two results coincide mod r, decide there was no fault and output the result mod p

A constructive application

- Quick idea of our constructive use of this attack
- Common trick to protect against fault injection in a device doing computations over $\mathbb{F}_p$:
  1. choose a small auxiliary prime r, and compute mod $p \cdot r$
  2. redo the computation mod r
  3. if the two results coincide mod r, decide there was no fault and output the result mod p
- In the case of ECC, free to choose the elliptic curve over $\mathbb{F}_r$ that you will combine using CRT to get a curve over $\mathbb{Z}/pr\mathbb{Z}$

A constructive application

- Quick idea of our constructive use of this attack
- Common trick to protect against fault injection in a device doing computations over $\mathbb{F}_p$:
  1. choose a small auxiliary prime r, and compute mod $p \cdot r$
  2. redo the computation mod r
  3. if the two results coincide mod r, decide there was no fault and output the result mod p
- In the case of ECC, free to choose the elliptic curve over $\mathbb{F}_r$ that you will combine using CRT to get a curve over $\mathbb{Z}/pr\mathbb{Z}$
- Our suggestion: use a degenerate curve instead!
  - Step 2 above becomes a simple base field exponentiation: much faster

Conclusion and perspectives

- We introduced a new attack that applies surprisingly broadly, and fully breaks unprotected ECC implementations with a single invalid point
  - Antipa et al.'s suggestion that alternate addition laws could protect from invalid curve attacks seems incorrect

Conclusion and perspectives

- We introduced a new attack that applies surprisingly broadly, and fully breaks unprotected ECC implementations with a single invalid point
  - Antipa et al.'s suggestion that alternate addition laws could protect from invalid curve attacks seems incorrect
- You should validate your points even when using Edwards curves!
  - though if you were clueful enough to use Edwards curves, you probably already did

Conclusion and perspectives

- We introduced a new attack that applies surprisingly broadly, and fully breaks unprotected ECC implementations with a single invalid point
  - Antipa et al.'s suggestion that alternate addition laws could protect from invalid curve attacks seems incorrect
- You should validate your points even when using Edwards curves!
  - though if you were clueful enough to use Edwards curves, you probably already did
- Can be used constructively for fault detection

Conclusion and perspectives

- We introduced a new attack that applies surprisingly broadly, and fully breaks unprotected ECC implementations with a single invalid point
  - Antipa et al.'s suggestion that alternate addition laws could protect from invalid curve attacks seems incorrect
- You should validate your points even when using Edwards curves!
  - though if you were clueful enough to use Edwards curves, you probably already did
- Can be used constructively for fault detection
- Question: can we prove that this will work for any elliptic curve model?

Thank you!