Crypto Engineering '20

✦

Elliptic curve cryptography

Pierre Karpman
pierre.karpman@univ-grenoble-alpes.fr
https://www-ljk.imag.fr/membres/Pierre.Karpman/tea.html

2020–11–03/13

References

This part of the course is mostly based on:

- *Curve-based cryptography*, Ben Smith (The Famous Yurt School, 2016)
- *Pairings for beginners*, Craig Costello
- *Montgomery curves and their arithmetic*, Craig Costello & Ben Smith (2018)

Motivation

DLP

Recall that given a group $\mathbb{G} = \langle g \rangle$ of prime order N, the *discrete logarithm problem* in $\mathbb{G}$ asks that given (g, g^x) with $x \xleftarrow{\$} [\![0, N-1]\!]$, find x

If we know a group where the DLP is hard, we can do:

- Public key-exchange (Diffie-Hellman)
- Signatures (Schnorr; DSA...)
- (Semi-Homomorphic) public-key encryption (ElGamal)

In a *generic group model*, solving a DLP instance requires expected $\approx \sqrt{N}$ group operations (Shoup, 1997)

Motivation (cont.): DLP/CDHP/DDHP precisions

- Actual cryptosystems rarely need a hard DLP *per se*, rather
- A hard CDHP (e.g. in Diffie-Hellman)
- A hard DDHP (e.g. in textbook ElGamal)
- But it is possible to solve a DDHP by solving a CDHP and to solve a CDHP by solving a DLP
- In most groups, the hardness of the DLP gives a good indication of the hardness of CDHP and DDHP (but there are counter-examples, cf. TD)

Motivation (cont.)

- Typical instantiation for $\mathbb{G}$: take $\mathbb{F}_p^\times$, where p is a "large" prime
  - It may also be fine to work with non-prime fields of medium/large characteristic
- But $\mathbb{F}_p^\times$ is NOT a generic group. DLP is much easier!
- The *Number field sieve* (NFS) has subexponential "$\mathrm{L}_p(1/3)$" complexity
  - $\mathrm{L}_x(\alpha, c) := \exp((c + o(1))(\log x)^\alpha(\log\log x)^{1-\alpha})$
- E.g. a DLOG computation with $p \approx 2^{795}$ took 3 200 core-years $\lll 2^{398}$ group operations (Boudot et al., 2020)
- Even better NFS variants exist in fields of small characteristic $\rightsquigarrow$ a recent record in characteristic two is a DLOG computation in $\mathbb{F}_{2^{30750}}$, taking 2 900 core-years (Granger et al., 2019)

Exeunt multiplicative groups, enter elliptic curves

- A prime-ordered group of points on a (well-chosen) elliptic curve is a good cryptographic approximation of a generic group
- $\Rightarrow$ the best-known algorithms are generic (e.g. Pollard ρ, Pollard kangaroos $\leftarrow$ cf. TP#3)
- For n-bit security, pick a prime-ordered group of 2^{2n} elements; double security $\Rightarrow$ double the bitlength of the order; gives scalability

So...

- What are these groups like?
- How do you compute in them?
- Can you do fancy stuff?

First: affine/projective spaces

- The points in the n-dimensional *affine space* $\mathbb{A}^n(\mathbb{F}_q)$ are n-tuples of $\mathbb{F}_q$ elements
- The points in the n-dimensional *projective space* $\mathbb{P}^n(\mathbb{F}_q)$ are equivalence classes over $n+1$-tuples of $\mathbb{F}_q$ elements, not all equal to zero, where $(X_0 : \ldots : X_n) \sim (\lambda X_0 : \ldots : \lambda X_n)$, $\lambda \in \mathbb{F}_q^\times$
- Example: $(2:1:0)$ and $(4:2:0)$ define the same point ($\rightsquigarrow$ doesn't make sense to say that $X_0 = 2$, but saying that $X_2 = 0$ or $X_0/X_1 = 2$ does)
- (In the following, we only consider planes, with affine points (x, y) and projective points $(X : Y : Z)$)

Affine/projective spaces (cont.)

- $\mathbb{A}^2$ is included in $\mathbb{P}^2$ via (typically) $(x, y) \mapsto (X : Y : 1)$
- The inverse mapping is $(X : Y : Z) \mapsto (X/Z, Y/Z)$, only defined if $Z \neq 0$
- The projective points of the form $(X : Y : 0)$ are in the *hyperplane at infinity* (here this is a line)

Elliptic curves

An elliptic curve $E/\mathbb{F}_q$ can be defined via a "short Weierstraß" (affine) model: it is the set of points verifying $y^2 = x^3 + ax + b$, $a, b \in \mathbb{F}_q$ under the non-singularity condition $4a^3 + 27b^2 \neq 0$
One often works projectively, using $(x, y) \mapsto (X/Z, Y/Z)$ (and multiplying everything by Z^3 to clear the denominators), giving the *projective* model $Y^2Z = X^3 + aXZ^2 + bZ^3$

- Such a curve has a unique point at infinity: $O_E = (0 : 1 : 0)$ (or simply O; also recall that $(0 : 2 : 0)$ is the same point)
- We are usually only interested in the points lying in $\mathbb{F}_q$, the $\mathbb{F}_q$*-rational poins of* E, written $E(\mathbb{F}_q)$
- There may be different models for the same elliptic curve ($\approx$ different formulas, up to changes of coordinates)

Group orders of elliptic curves

We will shortly define a group law over $E(\mathbb{F}_q)$; for the DLP to be hard therein we need $\# E(\mathbb{F}_q)$ to be large enough $\rightarrow$ how do you pick E? $\mathbb{F}_q$?

- By Hasse's theorem, if E is defined over $\mathbb{F}_q$, $\# E(\mathbb{F}_q) = q + 1 - t$ with $|t| \leq 2\sqrt{q}$
- So to get "n-bit" security, pick $q \approx 2^{2n}$
- Not much restriction on the exact field choice $\rightsquigarrow$ can use one with efficient arithmetic such as $\mathbb{F}_{2^{127}-1}$ or $\mathbb{F}_{2^{448}-2^{224}-1}$
- (Then pick E and check that $\# E(\mathbb{F}_q)$ has a large prime factor, etc.)
- ("Point counting" is not trivial, but it is reasonably efficient)

The group law for points of an E.C.

One can define a group over the ($\mathbb{F}_q$-rational) points of an E.C., best described geometrically. We first define and describe the negation $\ominus$ of a point

- E.C. have a natural symmetry along the X-axis: if $P = (X_P : Y_P : 1) \in E$, then so is $(X_P : -Y_P : 1) \rightsquigarrow$ use this to define $\ominus P$ as $(X_P : -Y_P : 1)$
- The point at infinity $(0 : 1 : 0)$ is reflected to $(0 : -1 : 0)$, which is itself, so $\ominus O = O \leftarrow$ this is going to be the neutral element
- A projective equation for the vertical line "$x = \alpha$" is $X = \alpha Z$; if such a line intersects E, it does so in O (since $0 = \alpha 0$), and possibly in $(\alpha : \pm\beta : 1)$ where $\beta^2 = \alpha^3 + a\alpha + b$

The group law (cont.)

Theorem: A line (a degree-one equation) intersects E (a degree-three equation) in three points, counted with multiplicity

- So knowing P, Q, one can determine the unique other point R of E on the line going through P and Q (and more: if P and Q are in $E(\mathbb{F}_q)$, so will be R)
- Let P, Q, $R \in E$ be colinear; one defines the group law $\oplus$ by $P \oplus Q = \ominus R$, for which O is the identity

Why is this a group law over E (or more useful for us, $E(\mathbb{F}_q)$)?

- Internal-law, commutativity, existence of unique inverse and neutral element come from the above algebraic-geometry arguments
- The harder axiom is associativity... won't do it here...

Discrete logarithms in E.C.

- The group of points in an elliptic curve uses additive notation
- So the DLOG of $Q \in \langle P \rangle$ is m s.t. $[m]P = Q$, where $[m]P = P \oplus ... \oplus P$ m times
- $[m]P$ can be computed in time logarithmic in m using a "double-and-add" ($\equiv$ "square-and-multiply) process
- So we (obviously) need to be able to compute $P \oplus P$ and $P \oplus Q$

How to compute in the group?

Let P, Q be in $E(\mathbb{F}_q)$, how do you compute $P \oplus Q$ in practice?

- Elementary if P or Q is O, or $P = \ominus Q$
- Need explicit formulas when $P = Q \neq O$ (doubling) and $O \neq P \neq (\ominus)Q \neq O$ (regular addition)

(Back to the) Affine case, example when $P \neq Q$:

1. Determine the equation $y = \lambda x + \nu$ of the line passing through P and Q
2. E.g. $\lambda = (y_Q - y_P)/(x_Q - x_P)$; $\nu = (y_Q x_P - y_P x_Q)/(x_P - x_Q)$
3. Solve $(x - x_P)(x - x_Q)(x - x_R) = (x^3 + ax + b) - (\lambda x + \nu)^2$ for $x_R \rightarrow x_R = \lambda^2 - x_P - x_Q$ (i.e. the point is either P, Q or R, and it lies both on the E.C. and on the line between P and Q)
4. Deduce y_R as $-(\lambda x_R + \nu)$

The case $P = Q$ is obtained “similarly” by differentiating E to find the slope of the tangent at P

More on group laws

The implementation of the group laws in ECC is important for:

- Performance (obvs.)
  - Try also to optimise $P \oplus Q$ when P is fixed; tripling $[3]P$ (for doubling/tripling-add chains)...
- Security; need formulas that:
  - are always correct (not so easy, actually), even on (possibly) adversially chosen inputs
  - take uniform time to be computed (no special cases)

Some options:

- Use projective coordinates $\rightsquigarrow$ get rid of costly field inversions
- (Possibly) use alternative models for E $\rightsquigarrow$ different formulas

Example: Twisted Edwards models

Define $E/\mathbb{F}_q$ via $ax^2 + y^2 = 1 + dx^2y^2$; the group law on $E(\mathbb{F}_q)$ is completely defined (e.g. for doubling, simply use $x_P = x_Q$, $y_P = y_Q$ in the below!) by

$$(x_P, y_P) \oplus (x_Q, y_Q) = \left(\frac{x_P y_Q + y_P x_Q}{1 + d x_P x_Q y_P y_Q}, \frac{y_P y_Q - a x_P x_Q}{1 - d x_P x_Q y_P y_Q} \right)$$

and $\ominus(x, y) = (-x, y)$, and $(0, 1)$ is the neutral element

- In practice, use a variant with projective coordinates
- One may use such a curve model even if E was initially defined with a Weierstraß equation (warning: restrictions apply)

Another well-known model is the one of Montgomery curves, defined (in the affine case) via $by^2 = x^3 + ax^2 + x$ (more about that one later)

Caveat: models restrictions

- Not all models are equivalent in terms of the curves they may define
- For instance, if $\# E(\mathbb{F}_q)$ is not divisible by 4, then E *does not* have an Edwards or Montgomery model
  - (Let p be the largest prime that divides $\# E(\mathbb{F}_q) = hp$; we say that $E(\mathbb{F}_q)$ has *cofactor* h)
- But the curves used in some ECC standards are s.t. $\# E(\mathbb{F}_q)$ is prime, i.e. have cofactor 1 $\rightsquigarrow$ cannot use the "nicer" models!
  - (We still know complete formulas, cf. Renes et al., EC 2016, but they're slower than for e.g. Edwards curves)

(For more about models, formulas... cf. the *Explicit-Formulas Database*: `https://hyperelliptic.org/EFD/`)

Curves for multi-dimensional scalar multiplication

Recall that we are eventually interested in computing $[m]P$ s.t. the associated DLP is hard $\rightsquigarrow$ m is large, e.g. 256 bits

- One way to speed-up this computation (beyond fast curve formulas, etc.) is to use a curve with one (or sometimes even more) *efficiently computable endomorphism* $\phi : E(\mathbb{F}_q) \to E(\mathbb{F}_q)$ s.t. the action of ϕ corresponds to the multiplication by a large fixed scalar (an eigenvalue) λ, i.e. $\forall\, P \in E(\mathbb{F}_q)$, $\phi(P) = [\lambda]P$
- To compute $[m]P$, decompose m into (a_1, a_2) s.t. $[m]P = [a_1]P \oplus [a_2]\,\phi(P)$ (i.e. take a_1, a_2 s.t. $a_1 + \lambda a_2 \equiv m \mod N$, where $N = \#\langle P \rangle$) AND a_1, $a_2 \leq\approx \sqrt{m}$ (typically computed using lattice reduction)

Curves for multi-dimensional scalar mult. (cont.)

- Usefulness: one can compute $[a_1]P \oplus [a_2]\,\phi(P)$ faster than by computing $[a_1]P$ and $[a_2]\,\phi(P)$ separately (which would cost $\approx$ the same as computing $[m]P$)
  - Ex.: for the "Four$\mathbb{Q}$" curve (Costello & Longa, 2015) which uses 4-dimensional decomposition, using endomorphisms gives a $\approx 1.8\times$ speed-up
- But: endomorphism-accelerated curves are harder to find, may have more structure, and may be harder to implement than regular ones

Why is multi-dimensional scalar mult. faster?

Say we want to compute $[9]P \oplus [12]\,\phi(P)$

- Naïve (non constant-time): $[8]P \oplus P \oplus [8]\,\phi(P) \oplus [4]\,\phi(P) \rightsquigarrow$ 6 doubles, 3 adds
- Idea: precompute the points P, $\phi(P)$, $P \oplus \phi(P)$ and share the accumulator, that is:
  1. $A := O$
  2. $A := A \oplus (P \oplus \phi(P)) = P \oplus \phi(P)$ (bit 3 of 9 & 12 is 1)
  3. $A := [2]A = [2]P \oplus [2]\,\phi(P)$
  4. $A := A \oplus \phi(P) = [2]P \oplus [3]\,\phi(P)$ (bit 2 of 9 is 0, bit 2 of 12 is 1)
  5. $A := [2]A = [4]P \oplus [6]\,\phi(P)$
  6. $A := A \oplus O$ (do nothing: bit 1 of 9 & 12 is 0)
  7. $A := [2]A = [8]P \oplus [12]\,\phi(P)$
  8. $A := A \oplus P = [9]P \oplus [12]\,\phi(P)$ (bit 0 of 9 is 1, bit 0 of 12 is 0)

  $\rightsquigarrow$ 3 doubles, 3 adds

Actually constant-time scalar multiplication

- When computing $[m]P$, it is important not to leak *anything* about m
- ...for instance its Hamming weight (leaked in the previous example via e.g. timing or DPA)
- We need a way to compute $[m]P$ in (cryptographic) constant-time

The Montgomery ladder

We define the following function, due to Montgomery

```
scalarm(m, n, P) // m = ∑_{i=0}^{n-1} m_i 2^i
{
        A0 = O; A1 = P;
        for (i = n-1; i >= 0; i--)
                mi = (m >> n) & 1;
                if (mi == 0)
                        (A0,A1) = ([2]A0, A0 + A1);
                        ↪  // simultaneous
                else
                        (A1,A0) = ([2]A1, A0 + A1);
        return A0;
}
```

The Montgomery ladder (cont.)

Why does this work?

- We have the invariant $A_1 \ominus A_0 = P$
  - Initially true
  - Then (first branch): $A_0' = [2]A_0 = [2](A_1 \ominus P)$, $A_1' = A_0 \oplus A_1 = (A_1 \ominus P) \oplus A_1 = [2]A_1 \ominus P$
  - And (second branch): $A_1' = [2]A_1 = [2](A_0 \oplus P)$, $A_0' = A_0 \oplus A_1 = A_0 \oplus A_0 \oplus P = [2]A_0 \oplus P$
- We also have that at the end of step i, $A_0 = [m/2^i]P$ (and thence $A_1 = [m/2^i + 1]P$)
  - Initially true
  - Then (first branch): $m_i = 0 \to m/2^i = 2 \times (m/2^{i+1})$ and $A_0' = [2]A_0 = [2]([m/2^{i+1}]P) = [m/2^i]P$
  - And (second branch): $m_i = 1 \to m/2^i = 2 \times (m/2^{i+1}) + 1$ and $A_0' = A_0 \oplus A_1 = [m/2^{i+1}]P \oplus [m/2^{i+1} + 1]P = [m/2^i]P$
- We return the last value $A_0 = [m/1]P$

The Montgomery ladder (cont.)

- Constant-timedness: the two branches are *exactly* the same up to the role of A_0/A_1
- But we dislike branches in cryptography...
- So use a (constant-time) conditional swap instead

```
(T0, T1) = cswap(mi, A0, A1);
(T0, T1) = ([2]T0, T0 + T1);
(A0, A1) = cswap(mi, T0, T1);
```

- ...will be truly constant-time (as long as the group formulas are, cf. above)

A constant-time conditional swap

```
cswap(b,n,x,y) // x, y are n-bit strings to swap if
↪  the bit b == 1
{
        bn = broadcast(b, n); // bn = bbbbb...bbbb
        t = b & (x ^ y);
        x = x ^ t;
        y = y ^ t;
        return (x,y);
}
```

On two's complement architecture, one can implement `broadcast` on words as `(b ^ 1) - 1`

Going beyond groups

In a Diffie-Hellman key-exchange, the group is useful to get:

- commutativity $\rightsquigarrow$ correctness of the protocol
- security (i.e. CDHP is hard, $\approx$ DLP is hard)

But we aren't that much interested in the group *elements* themselves

- Recall that for $P \in E(\mathbb{F}_q) \backslash \{O\}$, $x_P \in \mathbb{F}_q$ determines $(P, \ominus P)$, i.e. "most" of the point
- Can we speed-up computations/improve resilience by "simplifying" P?
- An idea: why not just working with $(X_P : Z_P)$, i.e. working on $E(\mathbb{F}_q)/\langle \ominus \rangle \cong \mathbb{P}^1(\mathbb{F}_q)$?
- We define $\boldsymbol{x} : E \to \mathbb{P}^1$, $P = (X_P : Y_P : Z_P) \mapsto (X_P : Z_P)$

Pseudo-operations on E

We can define $[m]_* : \boldsymbol{x}(P) \mapsto \boldsymbol{x}([m]P)$, but how do we compute it?

- Observe that $\boldsymbol{x}(P)$, $\boldsymbol{x}(Q)$ determine both $\boldsymbol{x}(P \oplus Q) = \boldsymbol{x}(\ominus P \ominus Q)$ and $\boldsymbol{x}(P \ominus Q) = \boldsymbol{x}(\ominus P \oplus Q)$
- We can define $\texttt{xADD} : (\boldsymbol{x}(P), \boldsymbol{x}(Q), \boldsymbol{x}(P \ominus Q)) \mapsto \boldsymbol{x}(P \oplus Q)$ and $\texttt{xDBL} : \boldsymbol{x}(P) \mapsto \boldsymbol{x}([2]P)$
- The Montgomery ladder "differential addition chain" will provide a way to compute $\boldsymbol{x}([m]P)$ using only `xADD`s and `xDBL`s

The Montgomery ladder for $[m]_*$

```
pscalarm(m, n, x(P))
{
        A0 = x(O); A1 = x(P);
        for (i = n-1; i >= 0; i--)
                (T0, T1) = cswap(mi, A0, A1);
                (T0, T1) = (xDBL(T0), xADD(T0, T1,
                ↪  x(P))); // or a fused ``xDBLADD''
                (A0, A1) = cswap(mi, T0, T1);
        return A0;
}
```

The Montgomery ladder for $[m]_*$ (cont.)

Why does this work?

- `xADD` needs as input $\boldsymbol{x}(P)$, $\boldsymbol{x}(Q)$, $\boldsymbol{x}(P \ominus Q)$
- But we have already seen that in the original ladder $A_1 \ominus A_0$ is always equal to P
  - Here: $A_1 \ominus A_0 = \boldsymbol{x}(P)$
- Since $T_0 \ominus T_1 = A_1 \ominus A_0$ or $A_0 \ominus A_1$, it is equal to $\oplus P$ (in the original ladder)
- So $\boldsymbol{x}(T_0 \ominus T_1) = \boldsymbol{x}(P)$
- So here, $T_0 \ominus T_1 = \boldsymbol{x}(P)$ directly

x-line arithmetic on Montgomery curves

We still need to define explicit formulas for `xADD` and `xDBL`. We will show that for curves given in a Montgomery model, which (along with the above ladder) were originally introduced to speed up ECM factorisation (formulas also exist for the more general Weierstraß model, but they're slower)

A Montgomery curve $E/\mathbb{F}_q$ is given by the equation $BY^2Z = X^3 + AX^2Z + XZ^2$ where $B, A \pm 2 \neq 0$. One can then show the formulas:

$\mathtt{xADD}((X_P : Z_P), (X_Q : Z_Q), (X_{P\ominus Q} : Z_{P\ominus Q})) =$
$(Z_{P\ominus Q}(S_P T_Q + T_P S_Q)^2 : X_{P\ominus Q}(S_P T_Q - T_P S_Q)^2)$, where
$S_\alpha := X_\alpha - Z_\alpha$, $T_\alpha := X_\alpha + Z_\alpha$ $\rightsquigarrow$ does not depend on A nor B!

$\mathtt{xDBL}(X : Z) = (UV : W(U + CW))$, where $U := (X + Z)^2$,
$V := (X - Z)^2$, $W := R - S$, $C := (A - 2)/4$ $\rightsquigarrow$ only depends on A!

x-only Diffie-Hellman

We can now do elliptic-curve Diffie-Hellman in two ways

- Take $P \in E(\mathbb{F}_q)$, $\mathcal{A}$ computes and sends $[a]P$, receives $[b]P$ and computes $[ab]P$ (Working in $E(\mathbb{F}_q)$)
- Take $\boldsymbol{x}(P), P \in E(\mathbb{F}_q)$, $\mathcal{A}$ computes and sends $[a]_*P$, receives $[b]_*P$ and computes $[ab]_*P$ (Working in $E(\mathbb{F}_q)/\langle\ominus\rangle$)

In both cases, we must check that P lies on $E(\mathbb{F}_q)$ ($\rightsquigarrow$ possible problems if soemone is lying/injected a fault/made a mistake... Also somewhat expensive)

- Can we define another variant s.t. no check is necessary?

(Quadratic) twists of Montgomery curves

Let $E/\mathbb{F}_q : BY^2Z = X^3 + AX^2Z + XZ^2$,
$E'/\mathbb{F}_q : B'Y^2Z = X^3 + AX^2Z + XZ^2$, be two Montgomery curves;
E and E' are isomorphic via $(X, Y) \mapsto (X, \sqrt{B/B'}Y)$

- If B/B' is a square in $\mathbb{F}_q$ ($\square_{\mathbb{F}_q}(B/B')$), E and E' are isomorphic ("the same") *over* $\mathbb{F}_q$
- Otherwise, $\square_{\mathbb{F}_{q^2}}(B/B')$ since $\mathbb{F}_{q^2} \cong \mathbb{F}_q[\sqrt{R}] = \mathbb{F}_q/\langle X^2 - R\rangle$ for any non-square R in $\mathbb{F}_q$, so E and E' are isomorphic *over* $\mathbb{F}_{q^2}$, but **not** ("are different") over $\mathbb{F}_q$, and E' is said to be a *quadratic twist* of E
- Also $\neg\square_{\mathbb{F}_q}(B/B')$ iff. exactly one of B or B' is a non-square. (If neither is a square and $p := \mathrm{char}(\mathbb{F}_q)$, $b := \left(\frac{\mathrm{N}(B)}{p}\right) = -1$, $b' := \left(\frac{\mathrm{N}(1/B')}{p}\right) = \left(\frac{\mathrm{N}(B')}{p}\right) = -1$, and $\left(\frac{\mathrm{N}(B/B')}{p}\right) = bb' = 1$ since both the field norm and the Legendre symbol are multiplicative) (So all quadratic twists of E are $\mathbb{F}_q$-isomorphic)

Quadratic twists (cont.)

Now let $E/\mathbb{F}_q : B \ldots$, $E'/\mathbb{F}_q : B' \ldots$ be a curve and "its" quadratic twist (unique up to iso.) and $x \in \mathbb{F}_q$, then $\exists\, P \in E(\mathbb{F}_q)$ or $E'(\mathbb{F}_q)$ s.t. $\boldsymbol{x}(P) = x$. Proof (affine case):

- Let $x' := x^3 + Ax^2 + x$, and assume w.l.o.g. that $\square_{\mathbb{F}_q}(B)$, $\neg\,\square_{\mathbb{F}_q}(B')$
- Then if $\square_{\mathbb{F}_q}(x')$, $\square_{\mathbb{F}_q}(x'/B)$ and $(x, \sqrt{x'/B}) \in E(\mathbb{F}_q)$
- Else $\square_{\mathbb{F}_q}(x'/B')$ and $(x, \sqrt{x'/B'}) \in E'(\mathbb{F}_q)$

Exercice: show that this would not be true if E and E' were $\mathbb{F}_q$-isomorphic

Back to x-only Diffie-Hellman

We now have a strategy for avoiding point validation in (x-only) ECDH:

- Find a curve pair $(E/\mathbb{F}_q, E'/\mathbb{F}_q)$ where E' is a quadratic twist of E, and the DLP/CDHP is hard on *both* of them (in that case we say that E is *twist-secure*)
- Pick $x \in \mathbb{F}_q$, $\mathcal{A}$ computes and sends $[a]_* P$, receives $[b]_* P$ and computes $[ab]_* P$, where P is implicitly defined by x and is on $E(\mathbb{F}_q)$ or $E'(\mathbb{F}_q)$ (Working in $E(\mathbb{F}_q)/\langle \ominus \rangle \cup E'(\mathbb{F}_q)/\langle \ominus \rangle$) (One must still check that $\langle P \rangle$ for the induced P has a large order, and sometimes one may require that this group's order is prime, which is not guaranteed here)

$\rightsquigarrow$ The basis of Curve25519 software (Bernstein, 2006)