Lecture notes: Elliptic curves in cryptography

Lorenz Panny

TU/e, 2DMI10 'Applied Cryptography'

November 27, 2018

1 Why?

- Many popular public-key cryptosystems operate on a *group*.
- Core underlying assumption: *Discrete Logarithm Problem* is hard.
- How do we actually *instantiate* these schemes?

1.1 Reminders

Definition 1. An *abelian group* is a set G with a fixed element $0 \in G$, a binary operation $\oplus\colon G \times G \to G$, an unary operation $\ominus\colon G \to G$, and such that for all $x, y, z \in G$:

$$(x \oplus y) \oplus z = x \oplus (y \oplus z) \qquad 0 \oplus x = x \qquad (\ominus x) \oplus x = 0 \qquad x \oplus y = y \oplus x$$

Sometimes also written using multiplicative notation: $(H, 1, \otimes, \square^{-1})$.

Definition 2. For any group G and $x \in \mathbb{Z}$, let $[x]\colon G \to G$ denote the multiplication-by-n map (in multiplicative groups: exponentiation). For $g \in G$, the element $[x]g$ is *defined* as

$$\underbrace{g + g + \cdots + g}_{x \text{ copies}}$$

if $x \geq 0$ and as $[-x](\ominus g)$ if $x < 0$.

Warning. This is not how $[x]g$ is *computed*, since the cost is linear in x. Instead, one takes time linear in $\log x$ by writing $x = 2x' + b$ with $b \in \{0, 1\}$ and making use of the property

$$[x]g = \begin{cases} [x'](g \oplus g) & \text{if } b = 0; \\ g \oplus [x'](g \oplus g) & \text{if } b = 1. \end{cases}$$

(This is known as 'double-and-add' or 'square-and-multiply' or 'fast binary exponentiation', and so on. This is just the idea; there are better variants.)

Definition 3. Let G be a cyclic group of order n and $g \in G$ a generator. The *Discrete Logarithm Problem* (DLP) is: Given any $h \in \langle g \rangle \subseteq G$, compute $x \in \mathbb{Z}$ such that $[x]g = h$.

1.2 Cryptographic groups

Non-example. Let $p \in \mathbb{Z}$ be a prime and $G = (\mathbb{Z}/p, +)$. Then $[x]$ is just multiplication by $x \in \mathbb{Z}/p$ and we can recover $x \bmod p$ from $[x]g$ as $g^{-1} \cdot [x]g$, where g^{-1} can be computed using Euclid's algorithm.

Example (Diffie–Hellman '76)**.** Let $p \in \mathbb{Z}$ be a prime and $G = (\mathbb{F}_p^*, \cdot)$. Then $[x]$ is exponentiation $g \mapsto g^x$ and x can be recovered in *subexponential* time using *index calculus* methods. These algorithms rely crucially on prime factorization of integers, which finite fields (being quotients of $\mathbb{Z}$ and their extensions) inherit. The impact is that p must be big (multiple thousand bits) for security, which makes everything (comparably) slow.

Example (Koblitz '85, Miller '85)**.** Elliptic curves. The topic of this lecture!

Warning. *Every* finite cyclic group is abstractly isomorphic to some $(\mathbb{Z}/n, +)$. The difficulty of DLP therefore does not (only) lie in the group *structure*, but first and foremost in its *representation*!

2 Elliptic curves

2.1 Reminder: 'Clock crypto'

Let $C = \{x \in \mathbb{R}^2 : |x| = 1\}$ be the unit circle. We can turn this into an abelian group by adding angles relative to the y-axis:

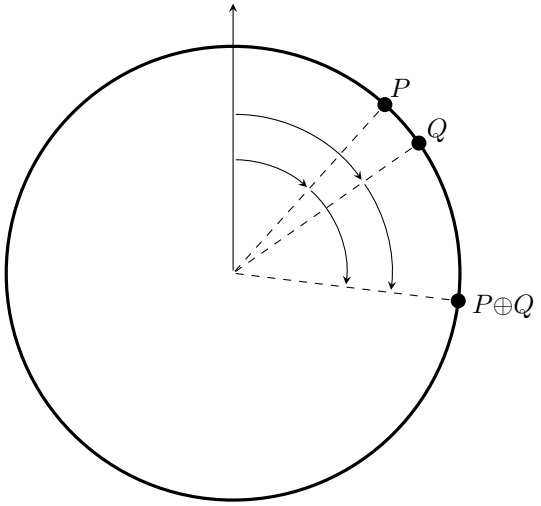

Figure 1: 'Clock crypto'

The neutral element is $\mathcal{O} = (0, 1) \in C$, and negating means mirroring an element over the y-axis, i.e., $(x, y) \mapsto (-x, y)$. To see how addition works, note that every point on C is of the form $(\sin\alpha, \cos\alpha)$, where α is the angle from the y-axis. The 'sum' of two points $P = (\sin\alpha, \cos\alpha)$ and $Q = (\sin\beta, \cos\beta)$ is given by the point $P \oplus Q = (\sin(\alpha+\beta), \cos(\alpha+\beta))$. Since

$$\begin{aligned}\sin(\alpha+\beta) &= \sin\alpha\cos\beta + \cos\alpha\sin\beta \\ \cos(\alpha+\beta) &= \cos\alpha\cos\beta - \sin\alpha\sin\beta\,,\end{aligned}$$

this means our addition law is

$$\oplus\colon\ (x_1, y_1), (x_2, y_2) \ \longmapsto\ (x_1y_2 + y_1x_2,\ y_1y_2 - x_1x_2)\,. \tag{1}$$

The same formulas work over finite fields $\mathbb{F}_p$! Is this good for crypto?

Lemma 1. Let $C = \{(x, y) \in \mathbb{F}_p^2 : x^2 + y^2 = 1\}$ and define a group on C with neutral element $\mathcal{O} = (0, 1) \in C$, negation $(x, y) \mapsto (-x, y)$, and addition as in (1). Then the map

$$\varphi\colon\quad C \to \mathbb{F}_{p^2}^*, (x, y) \mapsto \sqrt{-1} \cdot x + y$$

is an injective group homomorphism.
$\Longrightarrow$ DLP in $(C, \oplus)$ is no more secure than DLP in $(\mathbb{F}_{p^2}^*, \cdot)$ or even $(\mathbb{F}_p^*, \cdot)$ if $\sqrt{-1} \in \mathbb{F}_p$.

2.2 Edwards curves

Definition 4. Let k be a field with $\operatorname{char} k \neq 2$ and $d \in k \setminus \{0, 1\}$. Then

$$E_d\colon\ x^2 + y^2 = 1 + dx^2y^2$$

is an *Edwards curve* over k.
A *point* on E_d is a tuple $(x, y) \in k^2$ that satisfies the curve equation above.[1]
The set of points (in this sense) on E_d is denoted by $E_d(k)$.

Over $\mathbb{R}$ and for $d < 0$, Edwards curves look very similar to a 'clock' with a dent.

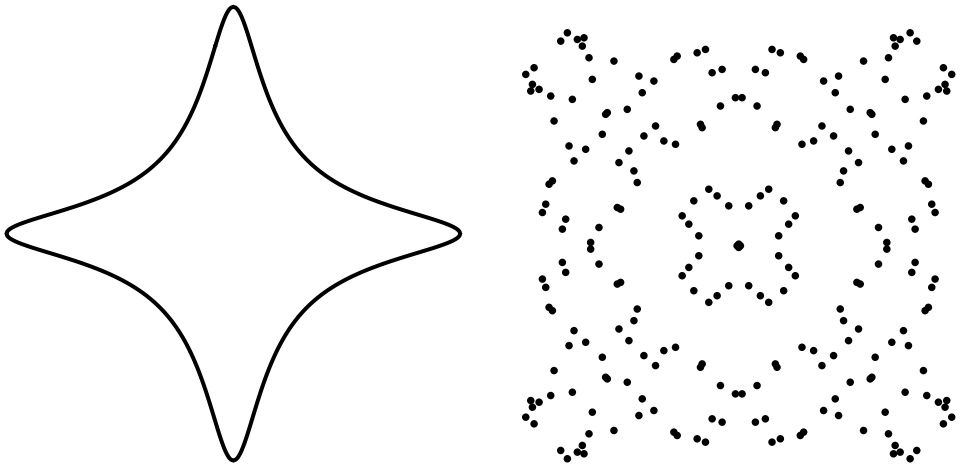

Figure 2: The curve $x^2 + y^2 = 1 - 71x^2y^2$ over $\mathbb{R}$ and over $\mathbb{F}_{257}$.

Theorem 1. If d is a non-square in k, then $E_d(k)$ forms an abelian group with neutral element $(0, 1)$, negation $(x, y) \mapsto (-x, y)$, and addition

$$(x_1, y_1) \oplus (x_2, y_2) := \left(\frac{x_1y_2 + y_1x_2}{1 + dx_1x_2y_1y_2},\ \frac{y_1y_2 - x_1x_2}{1 - dx_1x_2y_1y_2} \right).$$

Proof. **Exercise** in case you're bored.

Remark. The curve E_d as defined above is not an elliptic curve according to standard definitions. However, it is so closely related to an elliptic curve that it is usually still considered to *be* one. (To be precise, the desingularized projectivization of E_d is an elliptic curve. This pathology does not show up over k if d is a non-square.)

2.2.1 Why again?

Without knowing quite a bit of mathematical background in number theory and algebraic geometry, using elliptic curves may seem pretty 'random': Why not some other form of equation? Are these the only objects where this works?

The answer is that it is pretty difficult to 'just write down' an equation whose solutions satisfy a meaningful group law, such that the group law is not just addition (terrible) or

[1] Note there are other definitions! In particular, most *mathematical* literature takes $x, y \in \overline{k}$ when talking about a 'point' and uses the terminology 'rational point' for a point with $x, y \in k$. In cryptographic contexts, a 'point' often means 'rational point' because this is a finite set and easier to deal with on computers. For now, rational points are all we need.

multiplication (suboptimal) on the underlying field in disguise. Elliptic curves are, in a sense, the *simplest* of several classes of objects at once:

- They are exactly the 1-dimensional *abelian varieties*, i.e., 'shapes' cut out of a space by polynomial equations, with a natural group law on the points (which is also given by polynomials).
- They are smooth projective curves of *genus 1*. Genus-g curves have a natural group law on multisets of g points, hence elliptic curves (of the lowest 'non-trivial' genus) have a group law on *points* rather than just *collections* of points.

One can 'do crypto' on both of these classes of objects in general, but the situation becomes messier both computationally and security-wise. Usually the extra effort is not worth it (disclaimer: opinions about this vary). This is why elliptic curves are great: simplicity, performance, and security.

2.3 Other curve forms

There are various other forms of elliptic curves in common use:

- Short Weierstraß curves: $y^2 = x^3 + ax + b$ and a made-up 'point at infinity' $\mathcal{O}$
  - $-$ Annoying addition law (case distinctions).
  - $+$ Every elliptic curve in $\operatorname{char} k \notin \{2, 3\}$ can be written like this.
  - $-$ Basically no other benefits (except historical relevance).
- Montgomery curves: $By^2 = x^3 + Ax^2 + x$ and a made-up 'point at infinity' $\mathcal{O}$
  - $-$ Annoying addition law (case distinctions).
  - $-$ Always has a point $(0, 0)$ of order 4.
  - $+$ Many curves can be written like this.
  - $+$ Fast and side-channel resistant one-coordinate scalar multiplications.
    Curve25519 with $(p, A, B) = (2^{255} - 19, 486662, 1)$ is all over the internet.
    (See RFC 7748 at `https://tools.ietf.org/html/rfc7748`.)

Lemma 2. Any Edwards curve can be transformed to a Montgomery curve, and any elliptic curve over a field of characteristic $\notin \{2, 3\}$ can be transformed to short Weierstraß form.

Example: Let E_d be an Edwards curve and define the Montgomery curve

$$M_{A,B}\colon\ Bv^2 = u^3 + Au^2 + u$$

where $A = 2(1 + d)/(1 - d)$ and $B = 4/(1 - d)$. Then

$$\psi\colon\ (x, y) \longmapsto \left(\frac{1 + y}{1 - y}, \frac{1 + y}{(1 - y)x}\right)$$

is a map from E_d to $M_{A,B}$ which is defined almost everywhere. Moreover, wherever defined ψ is bijective and commutes with the group laws on E_d and $M_{A,B}$.

Remark. One might ask if there are no nicer maps ψ without points where they are not defined. However, every such map must have exceptional points: After all, $(0, 1) \in E_d$ must go to $\mathcal{O} \in M_{A,B}$, which does not have coordinates in (u, v)-space.

Remark. One might ask whether there is always a map in the other direction: For E_d as we have defined it this is false, but there is a slightly more general form of Edwards curves ('twisted' Edwards curves) into which any Montgomery curve can be transformed. The equation becomes

$$ax^2 + y^2 = 1 + dx^2y^2.$$

Many properties of elliptic curves are independent of the concrete coordinate system and apply to any way of writing the curve. We and everyone else thus typically only restrict to a specific form when necessary.

Write $\mathcal{O}$ for the neutral element, i.e., for Edwards curves $\mathcal{O} = (0, 1) \in E_d$.

2.4 Point counting

For cryptographic applications, we need to know how big our group really is. However, counting algorithms for generic groups are too slow as they essentially work by solving DLP, which should be infeasible. Similarly, plugging in all values for x and checking if there exists a corresponding y takes even longer.

The following classic result gives lower and upper bounds on the group size:

Theorem 2 (Hasse)**.** Let E be an elliptic curve over $\mathbb{F}_q$. Then the number of points (over $\mathbb{F}_q$) on E is bounded by

$$q + 1 - 2\sqrt{q} \;\leq\; \#E(\mathbb{F}_q) \;\leq\; q + 1 + 2\sqrt{q}.$$

Hasse's theorem implies that the group on E is about the size of the underlying field (which makes sense, as it is a one-dimensional object over that field). However, due to the Pohlig–Hellman algorithm, not only the *size* of the group is critical, but also the *prime factorization* of the group order. Hence we need to know the exact number of points, rather than just an estimate obtained from the Hasse bounds.

Theorem 3 (Schoof '85)**.** There exists an algorithm that computes the number of points on a given elliptic curve over $\mathbb{F}_q$ in time polynomial in $\log q$.

The algorithm and its improved versions (by Elkies and Atkin) are non-trivial but extremely interesting; **exercise** if you're bored: Look into it![2]

Using these ingredients, finding elliptic curves for use in cryptography is easy: They are usually constructed by brute-force search, picking new curves and counting points via (variants of) Schoof's algorithm until one with a big prime-order subgroup (which ensures security against all generic algorithms) is found.
(The prime number theorem states that for large x, there are approximately $x/\ln x$ primes below x, hence modelling $\#E(\mathbb{F}_q)$ as a random number of size $\approx q$ shows that we can expect a prime-order curve after approximately $\ln q$ tries.)

Remark. There are also methods to construct a curve with a *chosen* number of points, but they are only efficient in rather specific circumstances. (If you're bored, search for 'CM method'.)

3 Security

As usual in cryptography, all of these statements reflect the current public knowledge at the time of writing. For all we know, somebody could break DLP on elliptic curves (ECDLP) in polynomial time tomorrow, though this seems unlikely given how many capable people have tried (and failed) in the past decades.

In this very unmathematical, highly practical sense, the following facts are facts:

Fact. *Well-chosen elliptic curves are as close to generic groups as it gets.*

For 'good' E, the *only* improvement over generic algorithms is that P and $\ominus P$ share a coordinate (y if E is in Edwards form), hence can be identified in some DLP algorithms. This loses at most a single bit of security.

[2] Shamelessly advertising my B.Sc. thesis at `https://yx7.cc/docs/tum/thesis_schoof.pdf` as an introductory reference.

Fact. Solving the discrete-logarithm problem in a subgroup of prime size ℓ of a well-chosen elliptic curve over a prime field $\mathbb{F}_p$ requires, on average, $\sqrt{\pi\ell/4}$ steps using the best known (memory-efficient) method.

For comparison, the same method on a *generic* group of the same size, i.e., without making use of any special properties of elliptic curves, takes $\sqrt{\pi\ell/2}$ steps.

Warning. The difficulty of the DLP is *required* for the security of a cryptographic system relying on elliptic curves, but it is not *sufficient*.

For instance, there are active attacks making use of insufficient input validation: Send a point that is on a *different* curve, learn information about the private key from the resulting leakage. See `https://safecurves.cr.yp.to` for many more examples of such practical pitfalls and elliptic curves designed to protect you from shooting yourself in the foot. Quote from the webpage:

> There are many attacks that break real-world ECC without solving ECDLP. The core problem is that if you implement the standard curves, chances are you're doing it wrong. — D. J. Bernstein & T. Lange

3.1 Weak classes of elliptic curves

There exist several families of elliptic curves that are known to be (much) weaker than generic groups. Basically all of these examples are well-understood and the underlying reasons do not generalize to larger classes of, or even all, elliptic curves. (There are a few 'known unknowns,' where the community does not have strong confidence in the current security estimates, but these cases are easily avoided.)

- Slightly unclear: Curves over non-prime fields $\mathbb{F}_{p^k}$. (Not broken per sé, but shaky security history. The conservative choice is $\mathbb{F}_p$ for a large prime p.)
- Curves over $\mathbb{F}_q$ with q points (reduces DLP in $E(\mathbb{F}_q)$ to DLP in $(\mathbb{F}_q, +)$).
- Curves of low 'embedding degree' k (reduces DLP in $E(\mathbb{F}_q)$ to DLP in $(\mathbb{F}_{p^k}^*, \cdot)$). k is the smallest integer such that $\#E(\mathbb{F}_q) \mid p^k - 1$.
  In particular: *Supersingular* elliptic curves.
  More about this when we discuss pairings in the next lecture!
- Slightly weaker: Curves with small $t^2 - 4q$, where $t = q + 1 - \#E(\mathbb{F}_q)$.

4 Fast arithmetic on elliptic curves

Naïvely implementing the Edwards formulas

$$(x_1, y_1), (x_2, y_2) \longmapsto \left(\frac{x_1y_2 + y_1x_2}{1 + dx_1x_2y_1y_2}, \frac{y_1y_2 - x_1x_2}{1 - dx_1x_2y_1y_2} \right)$$

gives (all operations in $\mathbb{F}_q$):

$$\begin{aligned}
C &\leftarrow x_1 \cdot x_2 \\
D &\leftarrow y_1 \cdot y_2 \\
E &\leftarrow d \cdot C \cdot D \\
x_3 &\leftarrow ((x_1 + y_1) \cdot (x_2 + y_2) - C - D) \cdot (1 + E)^{-1} \\
y_3 &\leftarrow (D - C) \cdot (1 - E)^{-1}
\end{aligned}$$

This costs a bunch of additions (usually cheap), six multiplications, one multiplication with a (small?) constant d, and two inversions. In short, ignoring additions:

$$6\mathbf{M} + 1\mathbf{m} + 2\mathbf{I}.$$

Problem: Inversions can be very expensive, especially in constant-time code!

4.1 Projective coordinates

We can split fractions into numerator and denominator to delay divisions until the end of the computation. We can compute on these fractions like we would with integer fractions:

$$\frac{a}{b} + \frac{c}{d} = \frac{ad+bc}{bd}; \qquad \frac{a}{b} \cdot \frac{c}{d} = \frac{ac}{bd}; \qquad 1/\frac{a}{b} = \frac{b}{a}; \qquad \text{etc.}$$

We store (x, y) as fractions X/Z, Y/Z with a common denominator Z, i.e., a point on an Edwards curve is now of the form $[X : Y : Z]$ which represents $(X/Z, Y/Z)$. The conversions are given by

$$(x, y) \longmapsto [x : y : 1] \quad \text{and} \quad [X : Y : Z] \longmapsto (X/Z, Y/Z).$$

In this representation, the Edwards addition formula becomes

$$\begin{aligned}&[X_1 : Y_1 : Z_1], [X_2 : Y_2 : Z_2] \\ &\quad \longmapsto \left[\frac{X_1Y_2Z_1Z_2 + Y_1X_2Z_1Z_2}{Z_1^2Z_2^2 + dX_1X_2Y_1Y_2} : \frac{Y_1Y_2Z_1Z_2 - X_1X_2Z_1Z_2}{Z_1^2Z_2^2 - dX_1X_2Y_1Y_2} : 1\right],\end{aligned}$$

and after cancelling denominators:

$$\begin{aligned}&[X_1 : Y_1 : Z_1] \oplus [X_2 : Y_2 : Z_2] \\ &\quad = \Big[(X_1Y_2Z_1Z_2 + Y_1X_2Z_1Z_2)(Z_1^2Z_2^2 - dX_1X_2Y_1Y_2) \\ &\qquad : (Y_1Y_2Z_1Z_2 - X_1X_2Z_1Z_2)(Z_1^2Z_2^2 + dX_1X_2Y_1Y_2) \\ &\qquad : (Z_1^2Z_2^2 + dX_1X_2Y_1Y_2)(Z_1^2Z_2^2 - dX_1X_2Y_1Y_2)\Big]\end{aligned}$$

While this *looks* much worse than what we started with, it is actually very good for computations! Here is how we can evaluate this expression in a computer:

$$\begin{aligned}A &\leftarrow Z_1 \cdot Z_2 \\ B &\leftarrow A^2 \\ C &\leftarrow X_1 \cdot X_2 \\ D &\leftarrow Y_1 \cdot Y_2 \\ E &\leftarrow d \cdot C \cdot D \\ F &\leftarrow B - E \\ G &\leftarrow B + E \\ X_3 &\leftarrow A \cdot F \cdot ((X_1 + Y_1) \cdot (X_2 + Y_2) - C - D) \\ Y_3 &\leftarrow A \cdot G \cdot (D - C) \\ Z_3 &\leftarrow F \cdot G\end{aligned}$$

The cost is

$$10\mathbf{M} + 1\mathbf{m} + 1\mathbf{S}.$$

(Squarings $\mathbf{S}$ are often a bit cheaper than $\mathbf{M}$.)

4.2 The Montgomery ladder

Note that 'almost all' of the information contained in an elliptic-curve point is determined by one of the coordinates: On an Edwards curve E_d, for a given y, there exist at most two x such that (x, y) is a point on a given Edwards curve E_d. Moreover, the y-coordinate

of $[n]P$ depends only on the y-coordinate of P, since negation preserves y and therefore $[n](\ominus P) = \ominus[n]P$, has the same y-coordinate as $[n]P$. One may therefore wonder if we can avoid computing x in the first place, sacrificing one bit of information (and hence security), for much faster computations. It turns out that this is indeed the case.

Since the underlying ideas of this section really apply to any choice of curve model, we now replace the ad-hoc term 'dropping coordinates' by the mathematically much more meaningful 'identifying a point P and its negative $\ominus P$'. It turns out that the set $E/\pm$ of such equivalence classes still supports a well-defined *scalar multiplication*, whereas the *group* structure gets lost.

To build an algorithm that computes $[n](\pm P)$ without computing $[n]P$ on the way, we will need the operation of *differential addition*:

xDBLADD:

- **Input:** The elements $\pm P$, $\pm Q$, and $\pm(P \ominus Q)$ for some $P, Q \in E$.
- **Output:** The elements $\pm[2]P$ and $\pm(P \oplus Q)$.

On a Montgomery curve (modulo $\pm$) using $[X : Z]$ coordinates, one `xDBLADD` costs $5\mathbf{M} + 4\mathbf{S} + 1\mathbf{m}$. Using this operation, we can build the *Montgomery ladder*:

xMUL:

- **Input:** A scalar $n \geq 0$ and an element $\pm P$.
- **Output:** The element $\pm[n]P$.
  1. Write $n = \sum_{i=0}^{\ell-1} b_i 2^i$ with each $b_i \in \{0, 1\}$.
  2. Let $\pm Q \leftarrow \pm\mathcal{O}$ and $\pm R \leftarrow \pm P$.
  3. For $i \in (\ell - 1, \ell - 2, \ldots, 1, 0)$:
     1) If $b_i = 0$: Let $(\pm Q, \pm R) \leftarrow$ `xDBLADD`$(\pm Q, \pm R, \pm P)$.
     2) If $b_i = 1$: Let $(\pm R, \pm Q) \leftarrow$ `xDBLADD`$(\pm R, \pm Q, \pm P)$.
- Return $\pm Q$.

Notice that the algorithm above can be implemented such that the only conditional operations are swaps, which are inexpensive and easy to get right in constant-time implementations. This makes the Montgomery ladder highly efficient and robust against side-channel attacks.

Correctness. Ad-hoc notation: Write (k) for $\pm[k]P$. The variables $(x) := \pm Q$ and $(y) := \pm R$ are thus initialized as $(x) = (0)$ and $(y) = (1)$. By induction, we have $y - x = 1$ at all times, and hence the two possible steps in each loop are

$$(x), (y) \longmapsto \begin{cases} (2x), (x + y) & \text{if } b_i = 0; \\ (x + y), (2y) & \text{if } b_i = 1. \end{cases}$$

Define $u_i = \sum_{j=i}^{\ell-1} b_j 2^{j-i}$ and $v_i = u_i + 1$. The loop invariant at the end of the loop in the Montgomery ladder algorithm is

$$x = u_i; \quad y = v_i.$$

To see this, do induction. After the final iteration, we have $x = u_0 = n$. □

The bottom line is a practical result: With well-designed curves and algorithms, you can do *thousands* of Diffie–Hellman key exchanges *per second* on a single core!