Radbound University

Elliptic Curve Cryptography, Part 2

Cryptography, Autumn 2021

Lecturers: J. Daemen, B. Mennink

December 7, 2021

Institute for Computing and Information Sciences
Radboud University

Outline

ECC domain parameters

Scalar multiplication

Projective coordinates

Elliptic Curve Cryptosystems

Conclusions

ECC domain parameters

The order of the elliptic-curve group

- To have n bits of security for DL it would be sufficient that:
  (1) $q = \mathrm{ord}(G) \geq 2^{2n}$ and q prime
  (2) $\mathcal{E}$ is chosen so that it *avoids some properties*
- Due to Lagrange: $\mathrm{ord}(G) \mid \#\mathcal{E}$
- So we need $\mathcal{E}$ with an order that is divisible by a prime $\geq 2^{2n}$
- What can we expect for $\#\mathcal{E}$?
  - for roughly half of the values $x \in \mathbb{F}_p$, the expression $x^3 + ax + b$ is a square (mathematical term: *quadratic residue*)
  - if so and if y is a solution, so is $-y$
  - so $\#\mathcal{E}(\mathbb{F}_p) \approx \frac{1}{2} \cdot 2 \cdot p + 1 = p + 1$

Theorem of Hasse (Helmut Hasse, 1922)

For an elliptic curve over $\mathbb{F}_p$: $\#\mathcal{E} = p + 1 + t$ with $-2\sqrt{p} \leq t \leq 2\sqrt{p}$

ECC domain parameters

- We want $\mathcal{E}$ with $\#\mathcal{E} = hq$ with q a large prime and $h \leq 10$ or so
- Technique: repeat until a suitable curve is found:
  - take parameters p, a, b that would give a *good* curve
  - compute $\#\mathcal{E}$ with Schoof's algorithm (see Wikipedia)
- To assure backdoor absence, choice of p, a, b should be *explainable*
- Curves are proposed by experts and standardization bodies

ECC domain parameters

- The prime p (in general, a prime power p^n including $p = 2$)
- The curve parameters a and b (may have a different shape)
- The generator G
- The order q of the generator
- The co-factor: $h = \#\mathcal{E}/q$

Standard elliptic curves [for info only]

- 2000: First ECC domain parameters by company Certicom
- 2004: NIST standardized these and added some more
  - range of target security strength matching key lengths of AES
  - p are *Pseudo-Mersenne* primes for efficient modular reduction
    - $p256 = 2^{256} - 2^{224} + 2^{192} + 2^{96} - 1$
    - $p384 = 2^{384} - 2^{128} - 2^{96} + 2^{32} - 1$
    - $p521 = 2^{521} - 1$ (an actual Mersenne prime!)
  - all have cofactor $h = 1$, gives certain advantages
  - all have $a = -3$, allowing optimizations
  - innuendo of NSA backdoor, up to now without proof

  look it up: https://csrc.nist.gov/publications/detail/sp/800-186/draft
- 2005: German *Brainpool* consortium proposed alternative curves
- 2010: China publishes its own curves
- 2011: la France présente sa propre courbe: *die Französische Kurve*
- etc.

ECC 2.0 curves [for info only]

Efficient curves based on new insights and advanced math, best known:

- 2005: Curve25519 by Dan Bernstein, 126-bit security
- 2015: Curve448-Goldilocks by Mike Hamburg, 224-bit security
- 2015: FourQ by Craig Costello/Patrick Longa, 123-bit security

Their introduction was followed by a fierce battle for adoption

- Technical merit plays a role for adoption but other aspects too
- Lobbying in standardization groups and development community
  - ISO, NIST, BSI, ...
  - Internet standard governing TLS 1.3, SSH, ...: CFRG
  - OpenSSL, OpenVPN, ...
  - Signal, WhatsApp, ...

Scalar multiplication

Efficient scalar multiplication

- Scalar multiplication is the ECC counterpart of exponentiation
- Computing $[a]G$ in naive way takes $a-1$ point additions
- Infeasible if a and the coordinates of G are hundreds of bits long
- ECC counterpart of square-and-multiply is double-and-add
- Example: $[43]G$ with $G = (5,1) \in \mathcal{E}(\mathbb{F}_{23}) : y^2 = x^3 - x - 4$

$$\begin{array}{rccccccc}
10 & [2]G & = & G+G & [2]G = & (2,18) & = & (5,1)+(5,1) \\
100 & [4]G & = & [2]G+[2]G & [4]G = & (14,9) & = & (2,18)+(2,18) \\
1000 & [8]G & = & [4]G+[4]G & [8]G = & (21,17) & = & (14,9)+(14,9) \\
10000 & [16]G & = & [8]G+[8]G & [16]G = & (5,22) & = & (21,17)+(21,17) \\
100000 & [32]G & = & [16]G+[16]G & [32]G = & (2,5) & = & (5,22)+(5,22)
\end{array}$$

working it out:

$$\begin{array}{rccccccc}
11 & [3]G & = & [2]G+G & [3]G = & (20,15) & = & (5,1)+(2,18) \\
1011 & [11]G & = & [8]G+[3]G & [11]G = & (9,7) & = & (21,17)+(20,15) \\
101011 & [43]G & = & [32]G+[11]G & [43]G = & (21,6) & = & (2,5)+(9,7)
\end{array}$$

- Only 5 doublings and 3 additions instead of 42
- Side note: this example can be done in 3 doubling and 1 addition (find out why!)

Pseudocode for double-and-add, left-to-right variant

Input: point $G \in \mathcal{E}$, scalar $a \in \mathbb{Z}/q\mathbb{Z}$
Output: $A \in \mathcal{E}$ with $A = [a]G$

Let $a = a_0 + 2a_1 + 2^2 a_2 + 2^3 a_3 + \ldots + 2^{n-1} a_{n-1}$ and $\forall i : a_i \in \mathbb{Z}/2\mathbb{Z}$
$T \leftarrow G$
for $i \leftarrow n-2$ down to 0 **do**
  $T \leftarrow [2]T$
  if $a_i = 1$ **then** $T \leftarrow T + G$
end for
return $A \leftarrow T$

- there are many other algorithms for scalar multiplication
- for better efficiency, protection against side channel or fault injection attacks, ...
- these are out of scope of this course

Projective coordinates

Projective space

- Remarkable: $\mathcal{O} \in \mathcal{E}$ but **no** solution of the Weierstrass equation
- ...that defines a subset of the affine plane: $\{(x, y) \in \mathbb{F}_p \times \mathbb{F}_p\}$

More *natural*: picture the elliptic curve in the *projective plane*

The projective plane $\mathbb{P}^2$ over a field K

Set of equiv. classes of triplets (X, Y, Z) (all in K) excluding $(0, 0, 0)$
The equivalence relation is defined as
$(X_1, Y_1, Z_1) \sim (X_2, Y_2, Z_2) \Leftrightarrow \exists \lambda \in K \setminus \{0\} : (X_1, Y_1, Z_1) = (\lambda X_2, \lambda Y_2, \lambda Z_2)$

- We write $(X : Y : Z)$ for the equivalence class containing (X, Y, Z)
- Each class $(X : Y : Z)$ corresponds to a point
- If $Z \neq 0$ this is affine point $(x, y) = (X \times Z^{-1}, Y \times Z^{-1})$
- Classes $(X : Y : 0)$ are "points at infinity"

$\mathbb{P}^2$ is the affine plane extended with the (line of) points at infinity

The elliptic curve equation in homogeneous coordinates

Substitution of x by X/Z and y by Y/Z in the Weierstrass equation $y^2 = x^3 + ax + b$ and multiplication by Z^3 yields

$$Y^2Z = X^3 + aXZ^2 + bZ^3$$

In this equation all terms have the same degree (3) $\Rightarrow$ *homogeneous*

Therefore these $(X : Y : Z)$ are called *homogeneous coordinates*

Intersection of curve with line at infinity $Z = 0$ is $(0 : 1 : 0)$

- Neutral element: $\mathcal{O} = (0 : 1 : 0)$ satisfies the homogeneous equation!
- Inverse: $-(X : Y : Z) = (X : -Y : Z)$

Computing with these, we can avoid multiplicative inverses! Intuition:

$$(X/R : 0 : Z) = (X : 0 : Z \times R)$$

Note: there are other types of projective coordinates

On the choice of representation [for info only]

There is a wide variety of representations, **make sure to check**
`https://hyperelliptic.org/EFD/`

- Hardness of ECDLP is independent of the representation
- Affine is most compact and hence used in communication
- Projective avoids inversions and hence used in computation
- Converting projective to affine requires inverting Z
- Best choice of type of projective coordinates depends on
  - protocol: key agreement, signature, encryption, ...
  - platform: CPU instruction set, co-processor presence, ASIC, ...
  - domain parameters: pseudo-Mersenne or not, value of a, ...
  - need for protection against side channel attacks, ...
  - this is a subject of cryptographic engineering

Elliptic Curve Cryptosystems

Elliptic Curve Cryptosystems

Key pair generation in ECC

$a \xleftarrow{\$} \mathbb{Z}/q\mathbb{Z}$
$A \leftarrow [a]G$

ECC variants of classical discrete log schemes:

- ECDH: shared secret is x-coordinate of point on curve
- EC ElGamal encryption: plaintext and ciphertext are points on curve
- EC Schnorr authentication
- EC signature variants:
  - ECDSA (of DSA)
  - EdDSA (Schnorr signature)

All we said about classical discrete-log schemes applies to EC variants too

But there are some specifics ...

Elliptic Curve Diffie-Hellman (ECDH) key exchange

	Alice		Bob
have in advance:	$\mathcal{E}, G, (q)$		$\mathcal{E}, G, (q)$
	$a \xleftarrow{\$} \mathbb{Z}/q\mathbb{Z}$		$b \xleftarrow{\$} \mathbb{Z}/q\mathbb{Z}$
	$A \leftarrow [a]G$		$B \leftarrow [b]G$
		$\xrightarrow{\text{Alice}, A}$	
		$\xleftarrow{\text{Bob}, B}$	
	$P \leftarrow [a]B$		$P \leftarrow [b]A$

Alice and Bob arrive at the same shared secret point P

$$P = [a]B = [a][b]G = [ab]G = [b][a]G = [b]A$$

- As shared secret one takes the x-coordinate of the shared point P
- Does this reduce the security?
  - given $P \in \mathcal{E}$, x_p almost fully determines P
  - y_p has 2 possible values, so carries one more bit of information
- Alice and Bob derive key(s) from secret: $K \leftarrow \mathrm{h}(\text{“KDF”}; x_p)$

EC ElGamal encryption

Alice		Bob
$\mathcal{E}, G, (q), B$		$\mathcal{E}, G, (q), b, B(= [b]G)$
$a \xleftarrow{\$} \mathbb{Z}/q\mathbb{Z}$		
$A \leftarrow [a]G$		
$C \leftarrow M + [a]B$	$\xrightarrow{\text{Alice},(C,A)}$	$M \leftarrow C - [b]A$

- Cryptogram consists of two points on the curve: 4 affine coordinates
- Reduce data overhead by using *compressed representation*:
  - x-coordinate and parity of y: $y \bmod 2$
  - requires reconstruction of y-coordinate by receiver
- Reconstruction: compute $x^3 + ax + b$ and take its square root
- Square root is non-trivial but feasible: [for info only]
  - if $p \equiv 3 \pmod 4$, $\sqrt{x} = \pm x^{(p+1)/4}$
  - for $p \equiv 1 \pmod 4$ it is more complicated

EC Schnorr authentication protocol

Alice		Bob
$\mathcal{E}, G, q, A, a$		$\mathcal{E}, G, q$ (Alice: A)
$v \xleftarrow{\$} \mathbb{Z}/q\mathbb{Z}$		
$V \leftarrow [v]G$	$\xrightarrow{\text{Alice}, V}$	$c \xleftarrow{\$} \mathbb{Z}/q\mathbb{Z}$
	$\xleftarrow{c}$	
$r \leftarrow v - ca$	$\xrightarrow{r}$	$V \stackrel{?}{=} [r]G + [c]A$

- Just a different cyclic group
- Commitment V is now much shorter
- ... and can be shortened more with compressed point representation

EC Digital Signature Algorithm (ECDSA)

- NIST standard FIPS 186 defined DSA
  - This standard is updated regularly
  - FIPS 186-2 (2000) refers to ECDSA in an ANSI standard
  - FIPS 186-3 (2009) specifies ECDSA
  - Currently: draft FIPS 186-5 under revision
- ECDSA is probably the most implemented DL signature algorithm

This thing looks like this [for information only]:

Alice		Bob
$\mathcal{E}, G, q, A, a$		$\mathcal{E}, G, q$ (Alice: A)
$v \xleftarrow{\$} \mathbb{Z}/q\mathbb{Z}$, $V \leftarrow [v]G$		
$c \leftarrow x_v \bmod q$		
$r \leftarrow v^{-1}(\mathrm{h}(m) + ca)$	$\xrightarrow{m,(r,c)}$	$w \leftarrow r^{-1}$
		$P \leftarrow [\mathrm{h}(m)w]G + [cw]A$
		$c \stackrel{?}{=} x_p \bmod q$

EdDSA: the return of Schnorr!

Dan Bernstein proposes EdDSA as ECDSA alternative in 2007

- Ed stands for Edwards curve but maybe also *deterministic*
- It derives ephemeral key v from message
  - for this the private key is extended with a secret k
  - this avoids weaknesses due to bad randomness
  - …but introduces other potential vulnerabilities
- Ed25519: EdDSA using SHA-512 and Curve25519
- Ed448: EdDSA using SHAKE256 and Curve448 (much nicer!)

Specifications are a messy affair, but in our formalism it looks like this:

Alice		Bob
$\mathcal{E}, G, q, A, a, k$		$\mathcal{E}, G, q$ (Alice: A)
$v \leftarrow \mathrm{h}(k; m)$, $V \leftarrow [v]G$		
$c \leftarrow \mathrm{h}(\mathcal{E}; G; A; V; m)$		
$r \leftarrow v + ca$	$\xrightarrow{m,(r,V)}$	$c \leftarrow \mathrm{h}(\mathcal{E}; G; A; V; m)$
		$[r]G \stackrel{?}{=} V + [c]A$

Deployment of elliptic curve cryptography

ECC is probably the most widespread public-key crypto, e.g.,

- Handshake in TLS 1.3 (HTTPS)
- Secure Shell (SSH)
- Key agreement in Signal, Whatsapp
- Software update signatures (Sony, ...)
- Signatures in Bitcoin and other cryptocurrencies

For more examples, see

In search of CurveSwap: Measuring elliptic curve implementations in the wild, L. Valenta et al.
`https://eprint.iacr.org/2018/298.pdf`.

Conclusions

Conclusions

- ECC is probably the most widespread public key crypto
- Elliptic curves provide great groups for discrete log based crypto
  - key exchange, encryption, authentication and signatures
  - short public keys, signatures and shared secrets
  - there is a wide variety of curves and representations
- ECC is efficient
  - projective coordinates for efficient point addition and doubling
  - double-and-add for efficient scalar multiplication
  - point compression for very short public keys and signatures
- Elliptic curves support *pairings* that allow exotic functionality (out of scope of this introductory course)