A gentle introduction to elliptic curve cryptography

Craig Costello

Summer School on Real-World Crypto and Privacy
June 5, 2017
Šibenik, Croatia
Part 1: Motivation

Part 2: Elliptic Curves

Part 3: Elliptic Curve Cryptography

Part 4: Next-generation ECC
Diffie-Hellman key exchange (circa 1976)

\[ q = 1606938044258990275541962092341162602522202993782792835301301 \]
\[ g = 123456789 \]

\[ a = \]
685408003627063 761059275919665 781694368639459 527871881531452

\[ g^a \mod q = 78467374529422653579754596319852702575499692980085777948593 \]

560048104293218128667441021342483133802626271394299410128798 = \[ g^b \mod q \]

\[ b = \]
362059131912941 987637880257325 269696682836735 524942246807440

\[ g^{ab} \mod q = 437452857085801785219961443000845969831329749878767465041215 \]
## Index calculus

Solve \( g^x \equiv h \pmod{p} \)

e.g. \( 3^x \equiv 37 \pmod{1217} \)

- Factor base \( p_i = \{2, 3, 5, 7, 11, 13, 17, 19\} \), \( \#p_i = 8 \)
- Find 8 values of \( k \) where \( 3^k \) splits over \( p_i \), i.e., \( 3^k \equiv \pm \prod p_i \pmod{p} \)

\[
\begin{array}{c|c|c|c}
\text{(mod 1217)} & \text{(mod 1216)} & \text{(mod 1216)} \\
3^1 & 3 & L(2) \equiv 216 \\
3^{24} & -2^2 \cdot 7 \cdot 13 & L(3) \equiv 1 \\
3^{25} & 5^3 & L(5) \equiv 819 \\
3^{30} & -2 \cdot 5^2 & L(7) \equiv 113 \\
3^{34} & -3 \cdot 7 \cdot 19 & L(11) \equiv 1059 \\
3^{54} & -5 \cdot 11 & L(13) \equiv 87 \\
3^{71} & -17 & L(17) \equiv 679 \\
3^{87} & 13 & L(19) \equiv 528 \\
\end{array}
\]
Index calculus

solve $g^x \equiv h \pmod{p}$
e.g. $3^x \equiv 37 \pmod{1217}$

$L(2) \equiv 216$
$L(3) \equiv 1$
$L(5) \equiv 819$
$L(7) \equiv 113$
$L(11) \equiv 1059$
$L(13) \equiv 87$
$L(17) \equiv 679$
$L(19) \equiv 528$

Now search for $j$ such that $g^j \cdot h = 3^j \cdot 37$ factors over $p_i$

$3^{16} \cdot 37 \equiv 2^3 \cdot 7 \cdot 11 \pmod{1217}$

$L(37) \equiv 3 \cdot L(2) + L(7) + L(11) - 16 \pmod{1216}$

$\equiv 3 \cdot 216 + 113 + 1059 - 1$

$\equiv 588$

Subexponential complexity $L_p [1/3, (64/9)^{1/3}] = e^{((64/9)^{1/3} + o(1)) \cdot (\ln(p))^{1/3} \cdot (\ln \ln(p))^{2/3}}$
Diffie-Hellman key exchange

$$g = 123456789$$

$$a = \cdots$$

$$b = \cdots$$

$$g^a = \cdots$$

$$g^b = \cdots$$
Diffie-Hellman key exchange (cont.)

- Individual secret keys secure under Discrete Log Problem (DLP): \( g, g^x \mapsto x \)
- Shared secret secure under Diffie-Hellman Problem (DHP): \( g, g^a, g^b \mapsto g^{ab} \)
- Fundamental operation in DH is group exponentiation: \( g, x \mapsto g^x \) 
  ... done via “square-and-multiply”, e.g., \((x)_2 = (1,0,1,1,0,0,0,1 \ldots)\)
- We are working “mod \( q \)”, but only with one operation: multiplication
- Main reason for fields being so big: (sub-exponential) index calculus attacks!
DH key exchange (Koblitz-Miller style)

If all we need is a group, why not use elliptic curve groups?

Rationale: “it is extremely unlikely that an index calculus attack on the elliptic curve method will ever be able to work” [Miller, 85]
Part 1: Motivation

Part 2: Elliptic Curves

Part 3: Elliptic Curve Cryptography

Part 4: Next-generation ECC
### Some good references

| Elliptic curves | Silverman’s talk: “An Introduction to the Theory of Elliptic Curves”  
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Elliptic curves</td>
<td>Sutherland’s MIT course on elliptic curves:</td>
</tr>
<tr>
<td></td>
<td><a href="https://math.mit.edu/classes/18.783/2015/lectures.html">https://math.mit.edu/classes/18.783/2015/lectures.html</a></td>
</tr>
<tr>
<td>ECC</td>
<td>Koblitz-Menezes: ECC: the serpentine course of a paradigm shift</td>
</tr>
</tbody>
</table>
group \((G, +)\) can do \(+ -\)

ring \((R, +, \times)\) can do \(+ - \times\)

field \((F, +, \times)\) can do \(+ - \times \div\)
If you’ve never seen an elliptic curve before....

Remember: an elliptic curve is a group defined over a field

<table>
<thead>
<tr>
<th>elliptic curve group $(E, \oplus)$</th>
<th>can do $\oplus \ominus$</th>
</tr>
</thead>
<tbody>
<tr>
<td>underlying field $(K, +, \times)$</td>
<td>can do $+ - \times \div$</td>
</tr>
</tbody>
</table>

operations in underlying field are used and combined to compute the elliptic curve operation $\ominus$
Boring curves

\[ f(x, y) = 0 \quad \text{or} \quad f(X, Y, Z) = 0 \]

Degree 1 (lines)

\[ ax + by = c \quad \text{or} \quad ab \neq 0 \]

Degree 2 (conic sections)

\[ ax^2 + bxy + cy^2 + dx + ey + f = 0 \quad \text{or} \quad abc \neq 0 \]

e.g., ellipses, hyperbolas, parabolas

• “Genus” measures geometric complexity, and both are genus 0
• We know how to describe all solutions to these, e.g., over (exts of) \( \mathbb{Q} \)
• Not cryptographically interesting

Not cryptographically interesting
Elliptic curves

• Degree 3 is where all the fun begins...

\[ ax^3 + bx^2y + cxy^2 + dy^3 + ex^2 + fxy + gy^2 + hx + iy + j = 0 \]

\[ ch(K) \neq 2,3 \]

\[ E/K: \quad y^2 = x^3 + ax + b \]

• Elliptic curves ↔ genus 1 curves
• Set is \( \approx \) points \( (x, y) \in K \times K \) satisfying above equation
• Geometrically/arithmetically/cryptographically interesting
• Fermat’s last theorem/BSD conjecture/ ...
Elliptic curves, pictorially

\[ E/\mathbb{R} : y^2 = x^3 + x + 1 \]

\[ E/\mathbb{R} : y^2 = x^3 - x \]
Elliptic curves are groups

• So $E$ is a set, but to be a group we need an operation

• The operation is between points $(x_P, y_P) \oplus (x_Q, y_Q) = (x_R, y_R)$

• Remember: a group $(E, \oplus)$ defined over a field $(K, +, \times)$

• $K$ will be fields we’re used to, e.g., $\mathbb{Q}, \mathbb{C}, \mathbb{R}, \mathbb{F}_p$

• Remember: the (boring) operations $+, -, \times, \div$ in $K$ are used to compute the (exotic) operation $\oplus$ on $E$
Elliptic curve group law is easy

Fun fact: homomorphism between Jacobian of elliptic curve and elliptic curve itself.

Upshot: you don’t have to know what a Jacobian is to understand/do elliptic curve cryptography
The elliptic curve group law $\oplus$

We need $(x_P, y_P) \oplus (x_Q, y_Q) = (x_R, y_R)$

**Question:** Given two points lying on a cubic curve, how can we use their coordinates to give a third point lying on the curve?
The elliptic curve group law ⊕

We need \((x_P, y_P) \oplus (x_Q, y_Q) = (x_R, y_R)\)

**Question:** Given two points lying on a cubic curve, how can we use their coordinates to give a third point lying on the curve?

**Answer:** A line that intersects a cubic twice must intersect it again, so we draw a line through the points \((x_P, y_P)\) and \((x_Q, y_Q)\)
The elliptic curve group law $\oplus$

$R = P \oplus Q$

$R = P \oplus P$
The elliptic curve group law \( \oplus \)

\[ y = \lambda x + \nu \quad \text{intersected with} \quad y^2 = x^3 + ax + b \]

\[ x^3 - (\lambda x + \nu)^2 + ax + b = 0 \]

\[ x^3 - \lambda^2 x^2 + (a - 2\lambda\nu)x + (b - \nu^2) = (x - x_P)(x - x_Q)(x - x_R) \]

\[ x_R = \lambda^2 - x_1 - x_2 \]

\[ y_R = -(\lambda x_R + \nu) \]

\[ \lambda = \frac{y_Q - y_P}{x_Q - x_P} \]

\[ \lambda = \frac{dy}{dx} = \frac{3x_P^2 + a}{2y_P} \]
A toy example

\[ E/\mathbb{R} : y^2 = x^3 - 2x \]

What about \( E/\mathbb{Q} : y^2 = x^3 - 2 \) ?
The (abelian) group axioms

- **Closure**: the third point of intersection must be in the field

- **Identity**: \( E_{a,b}(K) = \{(x, y) : y^2 = x^3 + ax + b\} \cup \{\infty\} \)

- **Inverse**: \( \ominus (x, y) = (x, -y) \)

- **Associative**: proof by picture

- **Commutative**: line through \( P \) and \( Q \) same as line through \( Q \) and \( P \)
A toy example, cont.

\[ E / \mathbb{F}_{11} : y^2 = x^3 - 2x \]

\[ \#E = 12 \]

\[ (7,5) \oplus (8,10) = (10,1) \]
Part 1: Motivation

Part 2: Elliptic Curves

Part 3: Elliptic Curve Cryptography

Part 4: Next-generation ECC
Diffie-Hellman key exchange (circa 2016)

$q = 123456789$

$$g = a^b \pmod{q}$$

$$a = 41166406209599306608228525634418724107797992052797979347972317653086278383724247193766654694079781931294536136393973$

$$b = 4947664818322719328620181452555971990797762353766060481794987575545667054218587810513313821749726890599954928429540667899476$

$$g^a = 19749664818322719328620181452555971990797762353766060481794987575545667054218587810513313821749726890599954928429540667899476$

$$g^{ab} = 301061952941926937617350582464699124119958946561336153294395090886273029798033995180819113987880067639$

$$g^{ab} = 611271095370153907819768258438760711245438952974303230777552605012745155129275073181324945396659656$$

$$g^{ab} = 963256314092659364692150679857599280975466614266688947156342250179614556680778511034977598361086569276265454$

$$g^{ab} = 9053722513692627517434282036219645299720171163129219199961086475744375097361125005143089651201961186$

$$g^{ab} = 6134642765695265642985127359624510504369773198164817070505937388621737425526013433373058423892174946$

$$g^{ab} = 8636666684038313079437099765378482236201541790600007959983054091883878768567495832899520974334$$

$$g^{ab} = 75965893825977919037611838930411219219320000099410879907596869618953515014279426677479425602201308468
RECOMMENDED ELLIPTIC CURVES FOR FEDERAL GOVERNMENT USE

July 1999

This collection of elliptic curves is recommended for Federal government use and contains choices of private key lengths and underlying fields.

§1. Parameter Choices

§2. Curves over Prime Fields

For each prime $p$, a pseudo-random curve

$$E : y^2 = x^3 - 3x + b \pmod{p}$$
ECDH key exchange (1999 – nowish)

\[ p = 2^{256} - 2^{224} + 2^{192} + 2^{96} - 1 \]
\[ p = 1157920892103562487666677138432031875073158121289535519138702894\]

\[ E/\mathbb{F}_p: y^2 = x^3 - 3x + b \]

\[ \#E = 1157920892103562487666677138432031875073158121289535519138702894 \]

\[ P = (4843956129390645175905258525797914202762949526041747995844080717082404635286, 3613425967497957985851279195879819566111066729850150718771982535684144051099) \]

\[ a = 8913064459124603357763977064146285502314502849283525603183721922317324614395 \]

\[ [a]P = (84116208261315898167593067868200525612344221886333785331584793435449501658416, 1028856555421855980267392501728853001096802660585480486219453931280434275765740) \]

\[ b = 100955574639327864188069383169190708032771910919058405391679781082193405190826 \]

\[ [b]P = (10122888292005762667970413154540793024589549154209098899577542687271695288383, 778874181903040229941165950345562577670807185615679689372138134363978498341594) \]

\[ [ab]P = (10122888292005762667970413154540793024589549154209098899577542687271695288383, 778874181903040229941165950345562577670807185615679689372138134363978498341594) \]
The fundamental ECC operation

\[ P, k \mapsto [k]P \]
Scalar multiplications via double-and-add

How to (naively) compute \( k, Q \mapsto [k]Q \)?

\[
\begin{align*}
P &\leftarrow Q \\
\text{for } i \text{ from } n - 1 \text{ downto } 0 \text{ do} \\
&P \leftarrow [2]P \\
&\text{if } k_i = 1 \text{ then} \\
&P \leftarrow P \oplus Q \\
&\text{end if} \\
&\text{end for} \\
\text{return } P \, (= \, [k]Q)
\end{align*}
\]
Scalar multiplications via double-and-add

How to (naively) compute $k, Q \mapsto [k]Q$?

$k = (k_n, k_{n-1}, \ldots, k_0)_2$

1. $P \leftarrow Q$
2. for $i$ from $n - 1$ downto 0 do
   1. if $k_i = 1$ then
      1. $P \leftarrow P \oplus Q$
   2. end if
3. end for
4. return $P (= [k]Q)$
Scalar multiplications via double-and-add

How to compute $k, Q \mapsto [k]Q$ on $y^2 = x^3 + ax + b$?

$k = (k_n, k_{n-1}, ..., k_0)$

$$(x_P, y_P) \leftarrow Q$$

for $i$ from $n - 1$ downto 0 do

$$\lambda \leftarrow (3x_P^2 + a)/(2y_P); \quad v \leftarrow y_P - \lambda x_P;$$
$$x_P \leftarrow \lambda^2 - 2x_P; \quad y_P \leftarrow -(\lambda x_P + v);$$

if $k_i = 1$ then

$$\lambda \leftarrow (y_P - y_Q)/(x_P - x_Q); \quad v \leftarrow y_P - \lambda x_P;$$
$$x_P \leftarrow \lambda^2 - x_P - x_Q; \quad y_P \leftarrow -(\lambda x_P + v)$$

end for

return $$(x_P, y_P) = [k](x_Q, y_Q)$$
Projective space

• Recall we defined the group of $K$-rational points as
  \[ E_{a,b}(K) = \{(x, y) : y^2 = x^3 + ax + b\} \cup \{\infty\} \]

• The natural habitat for elliptic curve groups is in $\mathbb{P}^2(K)$, not $\mathbb{A}^2(K)$

• For (easiest) example, rather than $(x, y) \in \mathbb{A}^2$, take $(X:Y:Z) \in \mathbb{P}^2$ modulo the equivalence $(X:Y:Z) \sim (\lambda X : \lambda Y : \lambda Z)$ for $\lambda \in K^*$

• Replace $x$ with $X/Z$ and $y$ with $Y/Z$, so $E_{a,b}(K)$ is the set of solutions $(X:Y:Z) \in \mathbb{P}^2(K)$ to
  \[ E : \quad Y^2Z = X^3 + aXZ^2 + bZ^3 \]

• So the affine points $(x, y)$ from before become $(x : y : 1) \sim (\lambda x : \lambda y : \lambda)$ and the point at infinity is the unique point with $Z = 0$, i.e., $(0 : 1 : 0) \sim (0 : \lambda : 0)$
Projective space, cont.

• One practical benefit of working over $\mathbb{P}^2$ is that the explicit formulas for computing $\oplus$ become much faster, by avoiding field inversions.

• Thus, the fundamental ECC operation $k, P \mapsto [k]P$ becomes much faster...

\[
\begin{align*}
(x', y') &= [2](x, y) \\
\lambda &\leftarrow (3x^2 + a)/(2y) \\
x' &\leftarrow \lambda^2 - 2x \\
y' &\leftarrow -(\lambda(x' - x) + y);
\end{align*}
\]

$1S + 2M + 1I$

\[
\begin{align*}
(X' : Y' : Z') &= [2](X : Y : Z) \\
X' &= 2XY((3X^2 + aZ^2)^2 - 8Y^2XZ) \\
Y' &= (3X^2 + aZ^2)(12Y^2XZ - (3X^2 + aZ^2)^2) - 8Y^4Z^2 \\
Z' &= 8Y^3Z^3
\end{align*}
\]

$5M + 6S$
Projective scalar multiplications

How to compute $k, Q \mapsto [k]Q$ on $y^2 = x^3 + ax + b$?

$k = (k_n, k_{n-1}, ..., k_0)$

$(X_P:Y_P:Z_P) \leftarrow Q$

for $i$ from $n - 1$ downto 0 do

$(X_P:Y_P:Z_P) \leftarrow [2](X_P:Y_P:Z_P) \quad 5M + 6S$

if $k_i = 1$ then

$(X_P:Y_P:Z_P) \leftarrow (X_P:Y_P:Z_P) \oplus (X_Q:Y_Q:Z_Q) \quad 9M + 2S$

end for

return $(x_P, y_P) \leftarrow (X_P/Z_P, Y_P/Z_P) \quad 1I + 2M$
ECDLP security and Pollard’s rho algorithm

- ECDLP: given $P, Q \in E(\mathbb{F}_p)$ of prime order $N$, find $k$ such that $Q = [k]P$

- Pollard’78: compute pseudo-random $R_i = [a_i]P + [b_i]Q$ until we find a collision $R_i = R_j$ with $b_i \neq b_j$, then $k = (a_j - a_i)/(b_i - b_j)$

- Birthday paradox says we can expect collision after computing $\sqrt{\pi n/2}$ group elements $R_i$, i.e., after $\approx \sqrt{N}$ group operations. So $2^{128}$ security needs $N \approx 2^{256}$

- The best known ECDLP algorithm on (well-chosen) elliptic curves remains generic, i.e., elliptic curves are as strong as is possible
Consider \( E/\mathbb{F}_{1217} : \ y^2 = x^3 - 3x + 139 \)

\[ #E(\mathbb{F}_{1217}) = 1277 \]

\[ P = (3,401) \text{ and } Q = (192,847) \]

ECDLP: find \( k \) such that \([k]P = Q\)

Regardless of factor base, can’t efficiently decompose elements!

e.g., factor base \( R_i = \{(3,401), (5,395), (7,73), (11,252), (13,104), (19,265)\} \)

Writing \( S = \sum [k_i]R_i \) involves solving discrete logarithms, compare this to integers \( \text{mod } p \) where we lift and factorise over the integers
Part 1: Motivation
Part 2: Elliptic Curves
Part 3: Elliptic Curve Cryptography
Part 4: Next-generation ECC
What’s wrong with old school ECC?

• **Side-channel attacks**: starting with Kocher’99, side-channel attacks and their countermeasures have become extremely sophisticated

• **Decades of new research**: we now know much better/faster/simpler/safer ways to do ECC

• **Suspicion surrounding previous standards**: Snowden leaks, dual EC-DRBG backdoor, etc., lead to conjectured weaknesses in the NIST curves
Next generation elliptic curves

• 2014: CFRG receives formal request from TLS working group for recommendations for new elliptic curves
• 2015: NIST holds workshop on ECC standards
• 2015: CFRG announces two chosen curves, both specified in Montgomery (1987) form

\[ E/\mathbb{F}_p : y^2 = x^3 + Ax^2 + x \]

• Bernstein’s Curve25519 [2006]: \( p = 2^{255} - 19 \) and \( A = 486662 \)
• Hamburg’s Goldilocks [2015]: \( p = 2^{448} - 2^{224} - 1 \) and \( A = 156326 \)
• Both primes offer fast software implementations!
• Their group orders are divisible by 8 and 4, but this form offers several advantages.
Montgomery’s fast differential arithmetic

\[ E/\mathbb{F}_p : y^2 = x^3 + Ax^2 + x \]

- drop the \( y \)-coordinate, and work with \( x \)-only.
- projectively, work with \((X : Z) \in \mathbb{P}^1\) instead of \((X : Y : Z) \in \mathbb{P}^2\)
- But (pseudo-)addition of \( x(P) \) and \( x(Q) \) requires \( x(Q \ominus P) \)

Extremely fast pseudo-doubling: \( x\text{DBL} \)

\[
X_{[2]P} = (X_P + Z_P)^2(X_P - Z_P)^2 \\
Z_{[2]P} = 4X_PZ_P((X_P - Z_P)^2 + (A + 2)X_PZ_P)
\]

Extremely fast pseudo-addition: \( x\text{ADD} \)

\[
X_{P+Q} = Z_{P-Q}[(X_P - Z_P)(X_Q + Z_Q) + (X_P + Z_P)(X_Q - Z_Q)]^2 \\
Z_{P+Q} = X_{P-Q}[(X_P - Z_P)(X_Q + Z_Q) - (X_P + Z_P)(X_Q - Z_Q)]^2
\]
Differential additions and the Montgomery ladder

- Given only the \( x \)-coordinates of two points, the \( x \)-coordinate of their sum can be two possibilities
- Inputting the \( x \)-coordinate of the difference resolves ambiguity
- The (ingenious!) Montgomery ladder fixes all differences as the input point: in \( k, x(P) \mapsto x([k]P) \), every \texttt{xADD} is of the form
  \[ \texttt{xADD}(x([n+1]P), x([n]P), x(P)) \]
- We carry two multiples of \( P \) “up the ladder”: \( x(Q) \) and \( x(Q \oplus P) \)
- At \( i^{th} \) step: compute \( x([2]Q \oplus P) = \texttt{xADD}(x(Q \oplus P), x(Q), x(P)) \)
- At \( i^{th} \) step: pseudo-double (\texttt{xDBL}) one of them depending on \( k_i \)
Fast, compact, simple, safer Diffie-Hellman

- Write $k = \sum_{i=0}^{\ell-1} k_i 2^i$ with $k_{\ell-1} = 1$ and $P = (x_P, y_P)$ in $E[n]$ (e.g., on Curve25519 or Goldilocks)

$$(x_0, x_1) \leftarrow (\text{xDBL}(x_P), x_P)$$
for $i = \ell - 2$ downto 0 do
$$(x_0, x_1) \leftarrow \text{cSWAP}(k_{i+1} \otimes k_i, (x_0, x_1))$$
$$(x_0, x_1) \leftarrow (\text{xDBL}(x_0), \text{xADD}(x_0, x_1, x_P))$$
end for
$$(x_0, x_1) \leftarrow \text{cSWAP}(k_0, (x_0, x_1))$$
return $x_0 (= x_{[k]P})$

- $x$-only Diffie-Hellman (Miller’85): $x([ab]P) = x([a][b]P) = x([b][a]P)$

(Elliptic curves for security)
Curve25519 and Goldilocks in the real world


• Both curves integrated into TLS ciphersuites

• In 2014, OpenSSH defaults to Curve25519

• Curve25519 is used in Signal Protocol (Facebook Messenger, Google Allo, WhatsApp), iOS, GnuPG, etc (https://en.wikipedia.org/wiki/Curve25519)
(Twisted) Edwards curves

\[ E : ax^2 + y^2 = 1 + dx^2y^2 \]

- Neutral element is \((0,1)\) - no projective space needed for \(E(K)\)
- Addition law is complete (for well-chosen \(E\))

\[
(x_1, y_1) + (x_2, y_2) = \left( \frac{x_1y_1 + x_2y_2}{y_1y_2 - x_1x_2}, \frac{x_1y_1 - x_2y_2}{y_1y_2 - x_1x_2} \right)
\]

- Extremely fast: 8M! Also works for doubling, inverses, everything
- Fast, simple, exception-free implementations that always compute correctly
- Also birationally equivalent to Montgomery curves!
Elliptic curves: the best of both worlds

attacker: generic vs. us: not generic
ECC is the best of both worlds

attacker’s toolbox vs. our toolbox
Questions?