CPSC/PMAT 669

Elliptic Curve Cryptography

Mike Jacobson

Department of Computer Science
University of Calgary

Topic 9

Outline

Motivation

Recall: El Gamal PKC and DSA signatures are *generic* in the sense that they can work with any finite abelian group.

The most promising implementations of El Gamal and DSA signatures is to use for the group G the set of points on an elliptic curve defined over a finite field.

The corresponding discrete logarithm problem appears to be very difficult (best known algorithms have exponential complexity).

- can use smaller parameters than RSA for the same security level
- shorter keys, possibly faster protocols

Key Sizes for Elliptic Curve Cryptography

NIST's Recommendations for security level bit sizes (SP 800-57 part 1):

Security level	80	112	128	192	256
Hash function size	160	224	256	384	512
Elliptic curve group size	**160**	**224**	**256**	**384**	**512**
RSA modulus	1024	2048	3072	8192	15360

Elliptic Curve Cryptography was proposed in 1985 independently by N. Koblitz and V. Miller.

Elliptic Curves

An *elliptic curve* is a curve with an equation

$$y^2 = x^3 + Ax + B$$

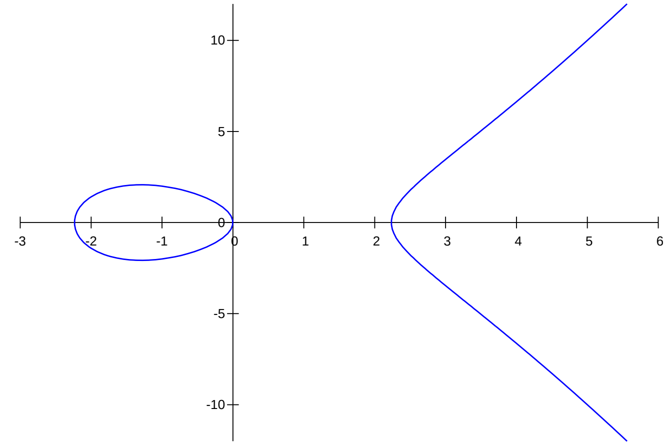
for quantities A, B in a field K with $4A^3 + 27B^2 \neq 0$.

- Equivalent to the polynomial $x^3 + Ax + B$ having three distinct roots.
- As a result, there is a unique tangent line to every point on the curve.

Elliptic curves make numerous appearances throughout math: in geometry, analysis, topology, number theory (e.g. proof of Fermat's Last Theorem), crypto, ...

An Example

The curve $y^2 = x^3 - 5x$ over $\mathbb{R}$



Geometry versus Algebra

Elliptic curves are *geometric* objects.

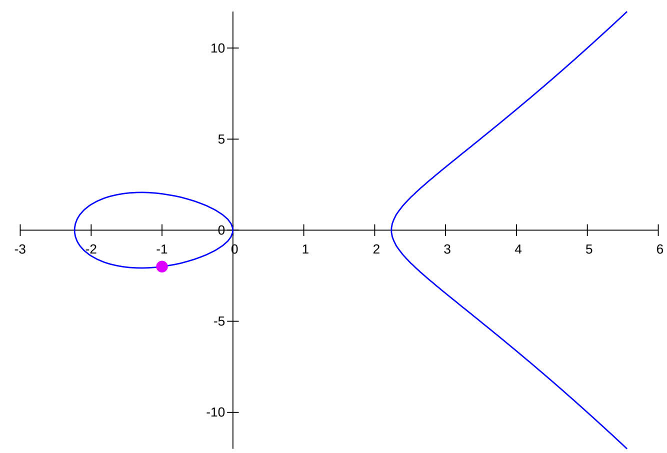
For cryptography, we need to be able to do *algebra*, so we need to perform *arithmetic* on *points* on elliptic curves.

- On $\mathbb{Z}_p^*$, the arithmetic operation was multiplication
- On an elliptic curve, the operation is *addition of points* on the curve
- Inverses in $\mathbb{Z}_p^*$ are replaced by *negatives* of points
- Squaring in $\mathbb{Z}_p^*$ is replaced by point *doubling*
- Exponentiation will be replaced by *scalar multiplication*:

$$nP = \underbrace{P + P + \cdots + P}_{a \text{ times}} \quad \text{for } n \in \mathbb{N}.$$

Elliptic Curve Arithmetic: Negation

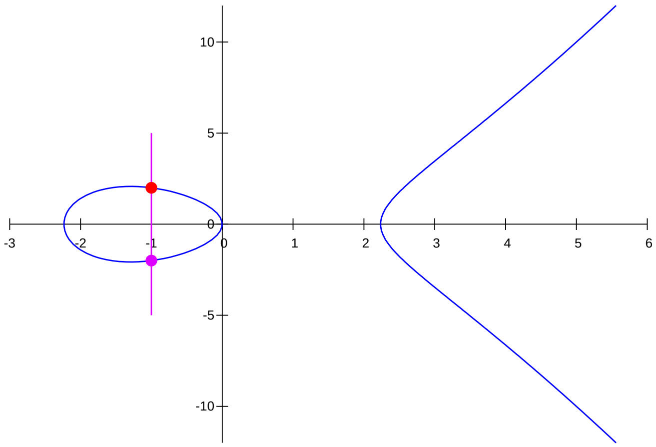
The negative of a point $P = (x, y)$ is its reflection on the x-axis:



$$-(\bullet) = ?$$

Elliptic Curve Arithmetic: Negatives (cont'd)

Negative of $P = (x, y)$ is $-P = (x, -y)$.



$-(\bullet) = \bullet$

Intersections

Any line intersects an elliptic curve in exactly three points.

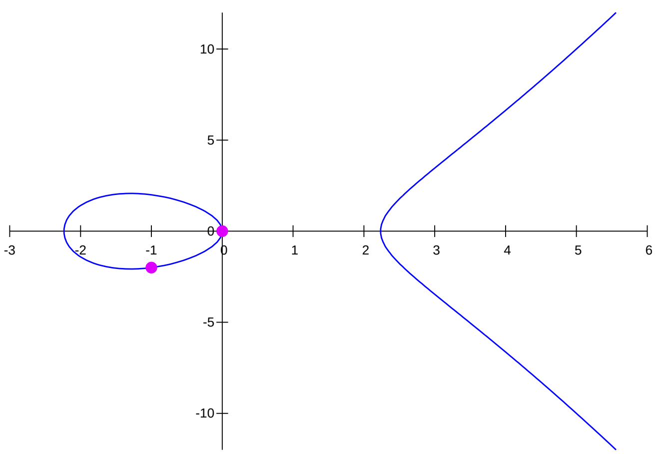
- The intersection may not be "visible" (e.g. for an elliptic curve over $\mathbb{R}$, we would need to draw a picture over $\mathbb{C}$)
- For a vertical line, the third point of intersection is the "point at infinity" which acts like 0.
- For a tangent line, the tangent point needs to be double-counted.

Addition of points is done according the "cord & tangent law": any three collinear points on the elliptic curve sum to zero.

Hence the sum of two points is the negative of the third point of intersection:

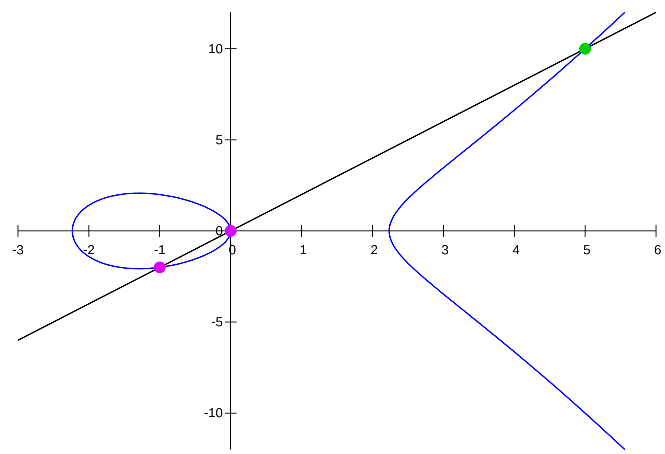
$$P + Q + R = 0 \implies P + Q = -R$$

Elliptic Curve Arithmetic: Addition



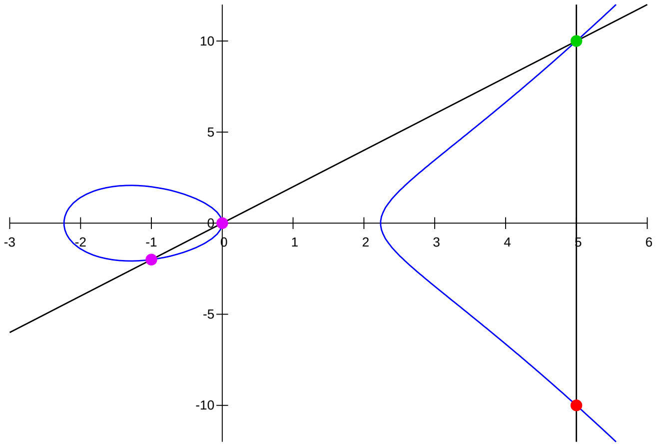
$\bullet + \bullet = ?$

Elliptic Curve Arithmetic: Addition (cont'd)



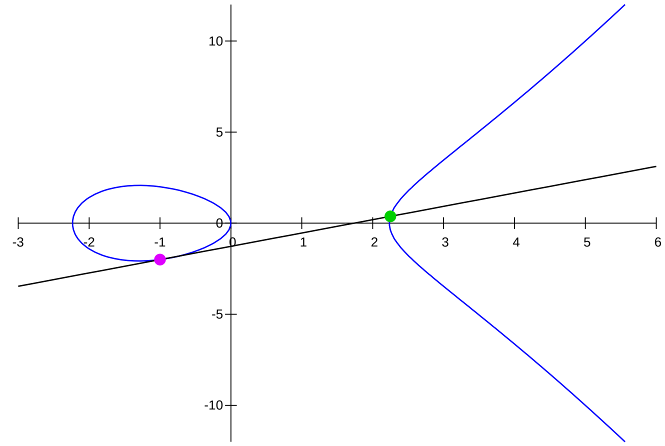
$\bullet + \bullet + \bullet = 0$

Elliptic Curve Arithmetic: Addition (cont'd)



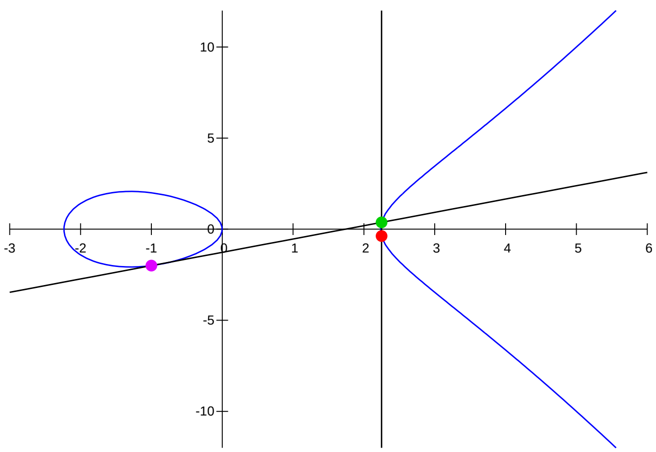
$$\bullet + \bullet + \bullet = 0 \quad \Rightarrow \quad \bullet + \bullet = \bullet$$

Elliptic Curve Arithmetic: Doubling



$$2 \times \bullet = ?$$

Elliptic Curve Arithmetic: Doubling (cont'd)



$$2 \times \bullet + \bullet = 0 \quad \Rightarrow \quad 2 \times \bullet = \bullet$$

Addition Formulas

Let

$$P_1 = (x_1, y_1),\ P_2 = (x_2, y_2) \qquad (P_1 \neq 0, P_2 \neq,\ P_1 \neq P_2)\ .$$

Then

$$\begin{aligned} -P_1 &= (x_1, -y_1) \\ P_1 + P_2 &= (\lambda^2 - x_1 - x_2,\ -\lambda^3 + \lambda(x_1 + x_2) - \mu) \end{aligned}$$

where

$$\boxed{\lambda = \begin{cases} \dfrac{y_2 - y_1}{x_2 - x_1} & \text{if } P_1 \neq P_2 \\[2ex] \dfrac{3x_1^2 + A}{2y_1} & \text{if } P_1 = P_2 \end{cases}} \qquad \boxed{\mu = \begin{cases} \dfrac{y_1 x_2 - y_2 x_1}{x_2 - x_1} & \text{if } P_1 \neq P_2 \\[2ex] \dfrac{-x_1^3 + A x_1 + 2B}{2y_1} & \text{if } P_1 = P_2 \end{cases}}$$

Properties of Point Addition

Let P, Q, R be arbitrary points on an elliptic curve. Then point addition satisfies the following properties:

- *Closure:* $P + Q$ is a point on the curve.
- *Existence of an identity:* adding the "point at infinity" to P leaves P unchanged.
- *Existence of inverses:* $-P$ is a point on the curve.
- *Associativity:* $(P + Q) + R = P + (Q + R)$.
- *Commutativity:* $P + Q = Q + P$.

Theorem 1

The points on an elliptic curve form a finite abelian group under point addition.

Elliptic Curves over Finite Fields

Consider $E(\mathbb{F}_q)$ where $\mathbb{F}_q$ is the finite field of q elements.

- Then $|E(\mathbb{F}_q)|$ is finite, as there are only q possible values for each point coordinate.
- A theorem of Hasse states

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

  i.e., $|E(\mathbb{F}_q)|$ is roughly as large as q.
- Can compute $|E(\mathbb{F}_q)|$ in polynomial time (Schoof, Kedlaya, etc...). In practice, can handle q of several thousand digits.

The geometric analogue of point addition does not carry over to the finite field case, but the algebraic formulas still work.

- Thus, $E(\mathbb{F}_q)$ is a finite abelian group under point addition.

Which Finite Fields?

Elliptic curves over $\mathbb{F}_p$ where p is a large prime admit efficient software implementations. The formulas are the same as above if $p > 3$.

Example 1

Let $E : Y^2 = X^3 + X + 1$. Then $P = (3, 10)$ and $Q = (9, 7)$ are both points in $E(\mathbb{F}_{23})$. We have $P + Q = (17, 20)$ and $2P = (7, 12)$.

Elliptic curves over $GF(2^n)$ are also attractive because they admit efficient hardware implementations.

- slightly different formulas required

Elliptic Curve Discrete Logarithm Problem

Since $E(\mathbb{F}_q)$ is a finite abelian group under point addition, it can be used in any generic protocol like Diffie-Hellman or El Gamal.

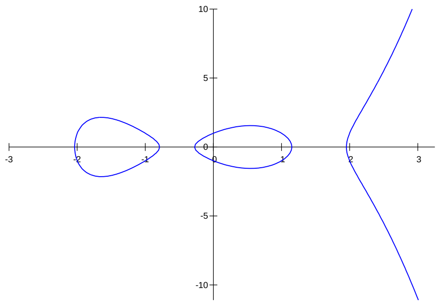
- The additive variant of g^x is computing xP, which can also be done efficiently with the binary exponentiation algorithm.
- The corresponding discrete logarithm problem is to compute x given points P and xP.

Except for a few special cases, the best-known algorithms for solving the elliptic curve discrete logarithm problem are exponential in $\lg q$, namely $O(\sqrt{q})$.

- To achieve 80, 112, 128, 192, and 256 bit security, we choose q with 160, 224, 256, 394, or 512 bits, respectively.
- Significantly smaller than corresponding sizes requried for RSA or El Gamal over $\mathbb{Z}_p^*$

Hyperelliptic Curves

An equally secure setting for discrete log based crypto is that of *genus 2 hyperelliptic curves*: $y^2 = x^5 + ax^3 + bx^2 + cx + d$. (Koblitz 1989)



- Arithmetic is conducted on *pairs* of points, with any three pairs of points lying on a *cubic* summing to 0.
- More complicated, but can choose p of half size (e.g. 128 bits) for the same level of security.

Examples of Cryptosystems Using Elliptic Curves

NSA Suite B endorses elliptic curve cryptography:

- ECDSA: the DSA signature scheme using group of points on an elliptic curve
- ECDH: Diffie-Hellman

ECMQV

- elliptic curve based authenticated key agreement protocol (authenticated version of Diffie-Hellman)
- named after Menezes, Qu, Vanstone
- dropped from Suite B, but used in many other standards and applications (eg. BlackBerry)

Elliptic Curve Key Agreement (ECMQV)

Definition 2

A *key establishment* protocol provides two or more entities communicating over an open network with a shared secret.

- key transport: send key via public-key encryption
- key agreement: Diffie-Hellman

Fundamental security goals:

1. Implicit key authentication (of B to A): A is sure the only person who can construct the key is B.
2. Explicit key authentication (key confirmation): A is assured that B has computed or can compute the key

Together, these provide *explicit key authentication*. To provide this to both A and B requires three message exchanges.

Security Goals for Key Agreement

Other desirable attributes:

1. Forward secrecy: long-term private key being compromised does not affect the security of previous session keys established by honest entities.
2. Key-compromise impersonation resilience: if A's private key becomes compromised, no one can use it to impersonate other people to A
3. Unknown key-share resilience: an entity cannot be tricked into sharing a key with someone to whom he doesn't intend

The Station-to-station protocol is one example of key agreement with explicit key authentication. Another is ECMQV, presented below.

ECMQV: System Parameters

Domain parameters $D = (q, FR, S, a, b, P, n, h)$, where

- q : size of the finite field $\mathbb{F}_q$, representation FR
- S : seed for the random number generator used to find the curve
- $a, b \in \mathbb{F}_q$: coefficients of the curve equation
- $P = (x_P, y_P) \in E(\mathbb{F}_q)$: *base point* of finite, prime order.
- n : order of P ($nP = O$), $h = |E(\mathbb{F}_q)|/n$ (called the *cofactor*)

Key pairs (Q_A, d_A), (Q_B, d_B) with $Q_A = d_A P$ and $Q_B = d_B P$

Key derivation function (KDF): outputs symmetric keys k_1, k_2 given a point

Message authentication code (MAC)

Given a point R, define $\overline{R}$ to be the integer $(\overline{x} \bmod 2^{\lceil f/2 \rceil}) + 2^{\lceil f/2 \rceil}$ where $\overline{x}$ is the integer representation of the x-coordinate of R and $f = \lfloor \log_2 n \rfloor + 1$ is the bitlength of n.

ECMQV Protocol

Goal: A and B establish a shared secret key with mutual entitiy authentication

Protocol messages:

- $A \to B : A, R_A$
- $B \to A : B, R_B, t_B = MAC_{k_1}(2, B, A, R_B, R_A)$
- $A \to B : t_A = MAC_{k_1}(3, A, B, R_A, R_B)$

Steps:

1. A selects $k_A \in [1, n-1]$ at random, computes $R_A = k_A P$ and sends A, R_A to B.

Step 2

B does the following:

1. Perform an *embedded public key validation* of R_A, i.e., check that $R_A \neq O$, the coordinates of R_A are properly-represented elements of $\mathbb{F}_q$, and that $R_A \in E(\mathbb{F}_q)$.
2. Select $k_B \in [1, n-1]$ at random and compute $R_B = k_B P$.
3. Compute $s_B = (k_B + \overline{R_B} d_B) \bmod n$ and $Z_B = h s_B (R_A + \overline{R_A} Q_A)$ and verify that $Z_B \neq O$.
4. $(k_1, k_2) = KDF(x_{Z_B})$, where x_{Z_B} is the x-coordinate of Z_B.
5. Compute $t_B = MAC_{k_1}(2, B, A, R_B, R_A)$.
6. Send B, R_B, t_B to A.

Remaining Steps

A does the following:

1. Perform an embedded public key validation of R_B.
2. Compute $s_A = (k_A + \overline{R_A} d_A) \bmod n$ and $Z_A = h s_A (R_B + \overline{R_B} Q_B)$ and verify that $Z_A \neq O$.
3. $(k_1, k_2) = KDF(x_{Z_A})$, where x_{Z_A} is the x-coordinate of Z_A.
4. Compute $t = MAC_{k_1}(2, B, A, R_B, R_A)$ and verify that $t = t_B$.
5. Compute $t_A = MAC_{k_1}(3, A, B, R_A, R_B)$ and send t_A to B.

B computes $t = MAC_{k_1}(3, A, B, R_A, R_B)$ and verifies that $t = t_A$.

The session key is k_2.

Note 1

The strings “2” and “3” in the MAC inputs distinguish tags from A and B.

Why This Works

The protocol works if $Z_A = Z_B$.

- A computes $Z_A = hs_A(R_B + \overline{R_B}Q_B)$
- B computes $Z_B = hs_B(R_A + \overline{R_A}Q_A)$.

Recall: $R_A = k_A P$, $Q_A = d_A P$, and $s_A = k_A + \overline{R_A}d_A$. Then

$$R_A + \overline{R_A}Q_A = k_A P + \overline{R_A}d_A P = (k_A + \overline{R_A}d_A)P = s_A P$$

Similarly, $s_B P = R_B + \overline{R_B}Q_B$.

Thus, we have $Z_A = hs_A s_B P = Z_B$.

Security of ECMQV

No proven results, but has the following properties:

1. s_A mod n is an implicit signature of the ephemeral public key R_A. "Signature" in the sense that only A can compute s_A, implicitly verified because B uses $s_A P$ to compute Z_B (thereby verifying the signature once A and B have the same shared key). Similarly for s_B mod n, giving implicit key authentication to both parties.
2. Successful verification of t_A and t_B provides key confirmation (both parties require shared secret Z to compute the MACs).
3. Session key k_2 is different each time (ephemeral), gives forward secrecy.
4. Provides "proof" if communications have been tampered with (MACs don't verify correctly).
5. Each party knows the identity of their partner, because IDs are included in MACs.

Summary

Security of ECMQV still subject to debate:

- some attacks, competition (HMQV)

Utility of elliptic curve cryptography widely accepted

- used in practice for many applications (eg. Blackberry, BluRay, etc...)
- Maps between curves (called *isogenies*) are the basis of a *quantum-resistant* cryptosystem.

Huge field of mathematical study in their own right.

For more on elliptic curves and applications to cryptography, take CPSC 629!