Public-key Cryptography and elliptic curves

Dan Nichols
University of Massachusetts Amherst
nichols@math.umass.edu

WINRS Research Symposium
Brown University

March 4, 2017
Cryptography basics

Cryptography is the study of secure communications. Here are some important terms:

- Alice wants to send a message (called the plaintext) to Bob.
- To hide the meaning of the message from others, she encrypts it, transforming the plaintext into the ciphertext.
- Bob can decrypt the ciphertext and reveal the plaintext, but a third party (Eve) cannot.
- A cipher is an algorithm for performing encryption and/or decryption.
Symmetric cryptography

- In a symmetric cipher, the same secret key is used for both encryption and decryption.
- Alice and Bob must share the same key and keep it secret from everyone else.
- This is difficult – how do they exchange keys securely?
- Analogy: a locked safe.
  - Alice, Bob have copies the key to open it
  - Each can leave messages there for the other to find
Here’s a cipher used by Julius Caesar: to encrypt a message, shift each letter \( N \) steps forward in the alphabet.

- if \( N = 3 \), replace every letter with the letter three steps after it in the alphabet.
  - \( a \to d \), \( b \to e \), etc.
  - ’winrs’ \( \to 'zlquv' \)
- Decrypt by shifting each letter back \( N \) steps
- The secret key is \( N \)
Why is this cipher so easy to break?

- The **key space** is small: only 26 possible keys
  - key size $n = \log_2(\text{number of possible keys}) \approx 5$
  - You could easily break this cipher with a **brute force attack**: try every key until you find the right one.
- The cipher does not hide all the statistical properties of the message
  - Check the frequency with which each letter appears in the ciphertext, compare to the expected frequencies of letters in English language.
  - This is an example of an **analytic attack**.
Security of a cipher depends on the best known attacks against it and on parameters like key size

- Tradeoff between security and convenience/efficiency
- Assume every practical cipher can be broken given enough time and resources
- If the best known attack is brute force...
  - Key length $n$ bits means $2^n$ possible keys to try. Impractical for reasonable $n$
- But if there’s a more sophisticated attack with running time polynomial in $n$, this is probably unsafe regardless of key size
  - Moore’s law: computing power per $\$ grows exponentially over time (for now)
- If a new attack is discovered, the cipher may not be completely ruined; just means bigger keys are necessary
Symmetric Cryptography

• Today we have much stronger symmetric ciphers available such as AES (Advanced Encryption Standard)
  • Large key space ($n = 128$ or 256). Brute force attacks are effectively impossible
  • Carefully designed to prevent analytic attacks
• But all symmetric ciphers share two inherent weaknesses
  • Alice and Bob must first communicate to share a key, which requires an already secure channel
  • In a network of $\geq 3$ people, each pair (e.g. Alice, Bob) needs their own shared key.
    • With $N$ people, that’s $N(N - 1)/2$ keys in total.
Public-key cryptography solves these problems

Basic idea: each person has their own public key and (secret) private key

Invented* in 1976 by Whitfeld Diffie, Martin Hellman, and Ralph Merkle
  - Invented much earlier by GCHQ (and probably NSA), but not published...

Analogy: each person has their own locked mailbox with a slot to accept incoming messages

The mailbox is the public key; the key to open the mailbox is the private key.
1. Bob generates both a public key and a private key
   (a) Makes his public key visible to everyone
   (b) Keeps his private key secret
2. Alice encrypts a message using Bob’s public key, sends it to Bob
3. Bob can decrypt the message using his private key
   • Everyone can send encrypted messages to Bob. Only Bob has the private key to decrypt these messages.
   • No secure channel necessary. Alice can send Bob a message without first sharing a secret key.
   • In a network of $N$ people, just need $N$ public keys and $N$ private keys.
How are public-key algorithms used?

- Public-key ciphers are slower and less efficient than symmetric ciphers.
- Modern secure communication usually works like this:
  1. First use a public-key cipher to securely share a secret key for a symmetric cipher like AES.
  2. Then use the symmetric cipher to actually exchange messages.
- This way we get the best of both worlds!
- Based on mathematical trapdoor functions: easy computations that are hard to reverse.
  - easy-to-compute bijection $f$ with hard-to-compute inverse $f^{-1}$
- Example: RSA (Rivest, Shamir, Adelman) is based on the problem of factoring a large integer into two primes
  - Easy to multiply $pq = N$
  - But given $N$, very hard to find $p$ and $q$
Here’s another trapdoor problem. Let \( p \) be an odd prime and let \( b \) be a generator (primitive root) of the cyclic group \((\mathbb{Z}/p\mathbb{Z})^\times\).

- Given \( x \), it’s easy to compute \( y = b^x \pmod{p} \) (use “square and multiply” algorithm)
- But given \( y \), it’s very hard to compute \( x = \log_b y \)
  - Different from logarithms in \( \mathbb{R} \) where we can use numerical techniques e.g. Newton’s method
  - Is there a better way than just trying every value of \( x \)?
- The problem of finding \( x \) such that \( b^x \equiv y \pmod{p} \) is called the discrete logarithm problem (DLP)
The discrete logarithm

Example: Each number in $\mathbb{Z}/31\mathbb{Z}$ appears as $3^x$ for some $x$. But there’s no easy way to tell when a particular value will appear.

<table>
<thead>
<tr>
<th>$c$</th>
<th>$3^x \mod 31$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>9</td>
</tr>
<tr>
<td>3</td>
<td>27</td>
</tr>
<tr>
<td>4</td>
<td>19</td>
</tr>
<tr>
<td>5</td>
<td>26</td>
</tr>
<tr>
<td>6</td>
<td>16</td>
</tr>
<tr>
<td>7</td>
<td>17</td>
</tr>
<tr>
<td>8</td>
<td>20</td>
</tr>
<tr>
<td>9</td>
<td>29</td>
</tr>
<tr>
<td>10</td>
<td>25</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$c$</th>
<th>$3^x \mod 31$</th>
</tr>
</thead>
<tbody>
<tr>
<td>11</td>
<td>13</td>
</tr>
<tr>
<td>12</td>
<td>8</td>
</tr>
<tr>
<td>13</td>
<td>24</td>
</tr>
<tr>
<td>14</td>
<td>10</td>
</tr>
<tr>
<td>15</td>
<td>30</td>
</tr>
<tr>
<td>16</td>
<td>28</td>
</tr>
<tr>
<td>17</td>
<td>22</td>
</tr>
<tr>
<td>18</td>
<td>4</td>
</tr>
<tr>
<td>19</td>
<td>12</td>
</tr>
<tr>
<td>20</td>
<td>5</td>
</tr>
<tr>
<td>21</td>
<td>15</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$c$</th>
<th>$3^x \mod 31$</th>
</tr>
</thead>
<tbody>
<tr>
<td>22</td>
<td>14</td>
</tr>
<tr>
<td>23</td>
<td>11</td>
</tr>
<tr>
<td>24</td>
<td>2</td>
</tr>
<tr>
<td>25</td>
<td>6</td>
</tr>
<tr>
<td>26</td>
<td>18</td>
</tr>
<tr>
<td>27</td>
<td>23</td>
</tr>
<tr>
<td>28</td>
<td>7</td>
</tr>
<tr>
<td>29</td>
<td>21</td>
</tr>
<tr>
<td>30</td>
<td>1</td>
</tr>
<tr>
<td>31</td>
<td>3</td>
</tr>
</tbody>
</table>
Suppose Alice and Bob want to communicate using a symmetric cipher like AES.

- They need to share a secret key without anyone else seeing it.
- Plan: simultaneously create a key over an insecure channel without sharing private info. This is called key exchange.
- Diffie-Hellman key exchange (DHKE) uses the difficulty of the discrete logarithm problem to keep the key safe from attackers.
**Diffie-Hellman key exchange**

DH key exchange algorithm:

- Alice and Bob choose a large prime number $p$ and a primitive root $g \in (\mathbb{Z}/p\mathbb{Z})^\times$. These numbers will be shared publicly.
- Alice chooses a random integer $a$ modulo $p$ to be her private key. She calculates $A = g^a \mod p$, which is her public key.
- Bob chooses a random integer $b$ modulo $p$ to be his private key. He calculates $B = g^b \mod p$, which is his public key.
- Alice and Bob both publish their public keys so everyone can see them. They keep their private keys hidden.

Only Alice knows $a$

Everyone knows $p, g, A, B$

Only Bob knows $b$
Diffie-Hellman key exchange

<table>
<thead>
<tr>
<th>Only Alice knows</th>
<th>Everyone knows</th>
<th>Only Bob knows</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a$</td>
<td>$p, g, A, B$</td>
<td>$b$</td>
</tr>
</tbody>
</table>

Now it’s time to create a shared secret symmetric key.

- Alice calculates $k = B^a \equiv (g^b)^a \equiv g^{ab} \mod m$
- Bob calculates $k = A^b \equiv (g^a)^b \equiv g^{ab} \mod m$
- Now Alice and Bob both know $k = g^{ab}$, which they can use as a shared secret key
- For Eve to find $k$, she would have to know either $a$ or $b$, which are the base-$g$ logarithms of $A$ and $B$ modulo $p$. 
A toy example

- $p = 29, g = 10$
- Alice chooses $a = 6$ for her private key. She calculates $A = 10^6 \equiv 22 \mod 29$ for her public key
- Bob chooses $b = 21$ for his private key. He calculates $B = 10^{21} \equiv 12 \mod 29$ for his public key
- Alice computes $k = B^a \equiv 12^6 \equiv 28 \mod 29$
- Bob computes $k = A^b \equiv 22^{21} \equiv 28 \mod 29$
- The shared secret key is $k = 28$. 
• Note that DHKE cannot actually send an arbitrary message; it only generates a shared secret key.
• There is a similar cipher called ElGamal which is a true encryption/decryption algorithm.
Solving the discrete logarithm problem

- In the real world, the key size is probably $n = \log p \approx 1024$ (i.e. $p$ is about 308 digits!)
- Brute force attack: try all $2^{1024}$ values
  - running time $O(p) = O(2^n)$.
  - would take many, many years even for a supercomputer
- But there are some clever algorithms which speed things up:
  - Pollard rho
  - Pollard Kangaroo
  - Shanks / Baby-step giant-step
- This type of algorithm is called a “Birthday attack”
  - Running time $O(\sqrt{p}) = O\left(2^{\frac{n}{2}}\right)$
  - Better than brute force; equivalent to trying $2^{512}$ numbers instead of $2^{1024}$.
  - Still slow, but doubles the necessary key size for a given security level
There's a much better DLP algorithm called index calculus.

- Uses the fact that many integers modulo $p$ are products of lots of small primes (smooth numbers)
  - Create a factor base of small primes $a_1, a_2, \ldots, a_n$
  - Try to factor $b^k = \prod_{i=1}^{n} a_i^{e_i}$ for different values of $k$. Each one gives us a linear equation. Need $n$ independent equations.
  - Solving this system gives you the discrete log $x_i = \log_b a_i$ of each prime in the base
  - Now try to factor $b^m y = \prod_{i=1}^{n} a_i^{f_i}$ for some small $m$
  - $\log y = \sum f_i x_i - m$

- Running time is (more or less) $O \left( e^{\frac{3}{c} \sqrt{\log p}} \right)$
- IC is strong enough that it forces us to use much larger keys for DHKE
The DLP we’ve seen so far is the \((\mathbb{Z}/p\mathbb{Z})^\times\) version.

- Homomorphism \(\mathbb{Z} \rightarrow \mathbb{Z}/p\mathbb{Z}\) lets us use information about \(\mathbb{Z}\) (e.g. prime factorization) to understand \(\mathbb{Z}/p\mathbb{Z}\).
- This is why index calculus works – \((\mathbb{Z}/p\mathbb{Z})^\times\) is too easy.

But we can extend the DLP to other finite abelian groups. For instance, an elliptic curve \(E(\mathbb{F}_q)\)

- Birthday attacks like Pollard rho will work in any group, but index calculus is specific to \((\mathbb{Z}/p\mathbb{Z})^\times\).
- This means we can get away with smaller keys!
What is an elliptic curve?

Definition

An **elliptic curve** $E$ is a smooth plane curve defined by an equation of the form $y^2 = x^3 + ax + b$ for some constants $a$ and $b$. (Or actually the closure of this curve in projective space)

$E(K)$ is the set of points on this curve defined over the field $K$.

- $E(\mathbb{C})$ is a compact genus 1 Riemann surface and a complex Lie group
- $E(\mathbb{R})$ is a curve (see right) and a Lie group
- $E(\mathbb{Q})$ is a finitely generated abelian group (Mordell-Weil)
- $E(\mathbb{F}_q)$ is a finite abelian group (cyclic or product of two cyclics)
For group structure on $E(\mathbb{Q}) = \{(x, y) \in \mathbb{Q}^2 : y^2 = x^3 + ax + b\}$ we need:

1. an associative binary operation $+$ such that for any two elements $P, Q$ in $G$, $P + Q$ is also in $E(\mathbb{Q})$.
2. an identity element $I$ such that $P + I = P$ for all $P$ in $E(\mathbb{Q})$.
3. an inverse $-P$ for each element $P$, such that $P + -P = I$. 

Group structure of an elliptic curve
Elliptic curve group operation

- To add $P + Q$:
  - Draw the line $PQ$
  - $PQ$ intersects the curve at exactly 3 points*
  - Define $P + Q$ to be the reflection across the $x$-axis of the third intersection point (besides $P$ and $Q$).
- Easy to prove the following:
  - $P + Q$ is always rational, so $P + Q$ is in $E(\mathbb{Q})$
  - $+$ is associative (and commutative)
- To add $P + P$, draw the tangent to the curve at $P$
Two questions:

1. What happens if you add two points with the same $x$ coordinate?
   - $\overline{PQ}$ is a vertical line
   - Only intersects the curve at $P$ and $Q$ – there’s no third point!

2. What is the identity element?
   - We need some point $I$ such that for every point $P$ on the curve, $P + I = P$ and $P + -P = I$.

To answer these questions, we need to add a point at infinity.
• Think of $\infty$ as a point that exists infinitely far above (and/or below) the $x$-axis.

• Think of a vertical line $\overline{PQ}$ as passing through three points: $P, Q$ on the curve and $\infty$.

• $\infty$ is the identity element
  • To add $P + \infty$, draw a vertical line through $P$. The line $\overline{P\infty}$ intersects the curve directly above or below $P$, so $P + \infty = -(\neg P) = P$.

• The inverse of $P$ is its reflection across the $x$-axis, $\neg P$.
  • The line $\overline{P\neg P}$ intersects the curve at $P, \neg P, \text{and} \infty$, so $P + \neg P = \infty$. 
The identity element is $\infty$

- $P + \infty = P$
- $P + -P = \infty$
- $Q + \infty = Q$
- $Q + -Q = \infty$
Let $E$ be an elliptic curve with equation $y^2 = x^3 + ax + b$ and let $P(x_1, y_1)$ and $Q(x_2, y_2)$ be points of $E(\mathbb{Q})$.

- If $P \neq Q$ and $x_1 \neq x_2$, let $s = \frac{y_2 - y_1}{x_2 - x_1}$
- If $P = Q$ and $y_1 \neq 0$, let $s = \frac{3x_1^2 + a}{2y_1}$
- Let $x_3 = s^2 - x_1 - x_2$ and let $y_3 = y_1 - s(x_1 - x_3)$
- Then $(x_3, y_3)$ is the third intersection point of $E$ and $\overline{PQ}$
- Therefore $P + Q = (x_3, -y_3)$.

So you don’t have to actually draw lines on a graph to add points. You can just use this formula.
For cryptography we need a finite group, so we use $E(\mathbb{F}_q)$.

- Consider pairs $(x, y) \in \mathbb{F}_q^2$ which satisfy the equation

$$y^2 = x^3 + ax + b$$

- Example: $y^2 \equiv x^3 + x + 6 \mod 7$
  - $(4, 2)$ is a solution because $2^2 \equiv 4^3 + 4 + 6 \equiv 4 \mod 7$

- The group operation still works. (Use the formulas from the previous slide)

- $E(\mathbb{F}_q)$ is either cyclic or a direct product of two cyclic groups

- Hasse’s theorem: $\#E(\mathbb{F}_q) = q + 1 - a_q(E)$ with

$$|a_q(E)| \leq 2\sqrt{q}$$

  - Sato-Tate conjecture: distribution of $a_q(E)$ among curves defined over $\mathbb{F}_q$
The Elliptic Curve Discrete Log Problem

- If you have a number $n$ and a point $P$ on the curve, it’s easy to add $P$ to itself $n$ times and find the point $nP$
  - Fast algorithm “double and add” (analogous to “square and multiply” for exponentiation mod $m$)
- But, if you have $P$ and an arbitrary point $Q$, how do you find a number $n$ such that $P$ added to itself $n$ times is $Q$?
  - If you keep adding $P$ you’ll eventually hit every point on the curve, but in an unpredictable order.
- This is the discrete log problem, in the group $E(\mathbb{F}_q)$ instead of $(\mathbb{Z}/p\mathbb{Z})^\times$. We call it ECDLP
- ECDH is a version of Diffie-Helman key exchange that uses the $E(\mathbb{F}_q)$ version of the discrete logarithm problem.
Alice and Bob want to securely generate a shared secret key

- They agree on an elliptic curve $E$, a prime $p$, and a base point $P$ on $E$. These things are all shared publicly.
- Alice chooses a random positive integer $a$ to be her private key. She adds $P$ to itself $a$ times to get a point $A = aP$ on $E$. This is her public key.
- Bob chooses a random positive integer $b$ to be his private key. He adds $P$ to itself $b$ times to get a point $B = bP$ on $E$. This is his public key.
- Alice and Bob publish their public keys, but keep their private keys secret.
• Alice adds $B$ to itself $a$ times, getting $k = a(bP) = (ab)P$.
• Bob adds $A$ to itself $b$ times, getting $k = b(aP) = (ab)P$.
• Now Alice and Bob both know $k = (ab)P$, which they can use as a shared secret key.
• For a third person to find $k$, they would have to compute $a$ or $b$, i.e. the discrete log of $A$ or $B$ in $E(\mathbb{F}_p)$.
Why use ECC?

- Birthday attacks work for DLP in any group, including $E(\mathbb{F}_p)$: Pollard rho, Kangaroo, etc
- But index calculus (mostly) only works for $(\mathbb{Z}/p\mathbb{Z})^\times$
  - IC relies on using information about $\mathbb{Z}$ (prime factorization)
  - For supersingular elliptic curves there is a version of index calculus. But we avoid this by not using those curves
- Best known attacks (for general curves) are of the Rho/Kangaroo/BSGS type, which are much slower
- Same security level as DH with much smaller keys!
Choosing a curve

Security and efficiency of ECC depends on choice of curve

- Supersingular curves (lots of endomorphisms) are bad
- $E(\mathbb{F}_p)$ where $p$ has small order $k$ modulo $\#E(\mathbb{F}_p)$ are bad
  - Use Tate pairing to reduce to DLP in $(\mathbb{F}_{p^k})^\times$
- Curves over $\mathbb{F}_{2^k}$ have very fast arithmetic, but there are good specialized algorithms for this case
- Some curves may have hidden weaknesses we can’t see
  - 2013: Snowden leaks reveal that Dual_EC_DRBG random number generator has a backdoor created by the NSA
  - 2015: NSA recommends phasing out ECC-based crypto algorithms (why?)
- Example of a good curve: Curve25519 (Daniel Bernstein)

$$y^2 = x^3 + 486662x^2 + x \quad p = 2^{255} - 19$$
ECC in the real world

- HTTPS often uses key exchange with Curve25519
- Sony PS3 used ECDSA to sign executables (oops)
- Online messaging protocols
- And much more!
Further reading:

- *Understanding Cryptography* by C. Paar, J. Pelzl
  - Good simple textbook on modern crypto algorithms. Written for engineers, no hardcore number theory

- *The Arithmetic of Elliptic Curves* by Joseph H. Silverman
  - Standard graduate text on elliptic curves.