Elliptic Curve Cryptography vs. RSA

Aaron Blumenfeld
University of Arizona
blumenfeld@email.arizona.edu

Elliptic Curve Cryptography vs. RSA

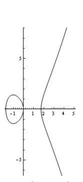
As factoring techniques and computational power have evolved, RSA key sizes have increased to the point that at least 3072-bit keys are required to match the security of AES.

Elliptic curve cryptography (ECC) is a promising invention which allows us to match the security of AES with keys as small as 256 bits.

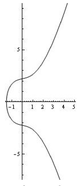
Elliptic Curves

An elliptic curve E is the graph of $y^2 = x^3 + ax + b$. We will focus on elliptic curves mod p instead of over $\mathbb{R}$, since that is the typical setting in cryptography.

We can map a (plaintext) message to a point on a curve, which lets us do cryptography.



$E(\mathbb{R}) : y^2 = x^3 - 3x$



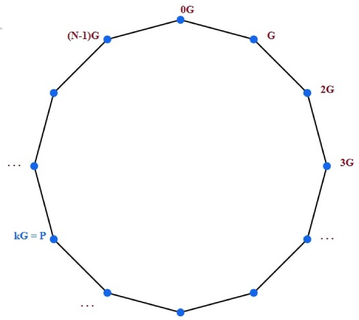
$E(\mathbb{R}) : y^2 = x^3 + x + 5$

Elliptic Curve Discrete Logarithms

Given an elliptic curve $E \pmod p$ with generator G, consider solving $kG = P$ for k. This is the discrete logarithm problem for elliptic curves.

$$kG = \underbrace{G + G + \ldots + G}_{\text{k terms}}$$

Compare to solving $g^k \equiv y \pmod p$ for k — we're still performing discrete exponentiation, simply written additively. Thus, we use repeated *doubling* instead of repeated squaring.



Visual representation of multiples of a generator G

Diffie-Hellman Key Exchange

Elliptic curve Diffie-Hellman is very similar to regular Diffie-Hellman:

1. Fix an elliptic curve $E \pmod p$ with N points, and generator G
2. Alice and Bob choose secret integers a and b in $[2, N-1]$
3. Alice sends aG to Bob, and Bob sends bG to Alice

Since $a(bG) = b(aG) = abG$, they each compute the point abG. Alice and Bob can then use some procedure to extract a key out of the point abG.

Integer Factorization Progress

Number field sieve algorithms can be used to factor integers in subexponential time (and are easily parallelizable).

1024-bit RSA could likely be broken in about 2 years on a few million cores with tens of GB of memory per processor.

RSA Modulus Size (in bits)	Year Factored
576	2003
640	2005
704	2012
768	2009

Key Size Comparison between ECC and RSA

Security Bits	RSA Key Size	ECC Key Size
80	1024	160
112	2048	224
128	3072	256
192	7680	384
256	15360	512

Discrete Logarithm Progress

The best algorithm (Pollard's Rho Algorithm) for breaking ECC takes time $O(\sqrt{n})$ (which is still exponential time).

In 2002, over 10,000 people solved an ECC discrete log over a 109-bit prime field after 549 days of computation. It's projected that 160-bit ECC could be broken within a year using 2.5 billion computers (only 117-bit ECC has been broken so far).

EC Discrete Log Size (in bits)	Year Broken
108	2000
109	2002
112	2009
117	2016

Experimental Results

Below are results from several experiments comparing the results between ECC and RSA. One area where RSA is competitive is *verifying* digital signatures. This is because this verification is simply RSA encryption of a short hash of the message, and the public exponent e is frequently quite small.

Table: First Experiment by Singh, Khan, and Singh

	Key Generation	Signature	Verification
ECC-224	30 ms	40 ms	45 ms
RSA-2048	2.8 sec	75 ms	1.7 ms

Table: Second Experiment by Singh, Khan, and Singh

	Key Generation	Signature	Verification
ECC-256	32 ms	47 ms	55 ms
RSA-3072	10.3 sec	185 ms	3.4 ms

Table: Third Experiment by Singh, Khan, and Singh

	Key Generation	Signature	Verification
ECC-384	16 ms	55 ms	71 ms
RSA-7680	61 sec	1.6 sec	13 ms

Elliptic Curve Discrete Logarithms vs. Classical Discrete Logarithms

Why use discrete logarithms on *elliptic curves* instead of discrete logarithms mod p?

Classical Discrete Logarithms:

Discrete logarithms mod p can be broken using an algorithm called *the index calculus*, which, like integer factorization, runs in subexponential time.

Finding a primitive root mod p generally involves factoring $p - 1$, which is hard.

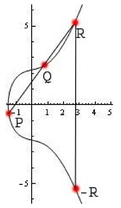
Elliptic Curve Discrete Logarithms:

The best known algorithm is Pollard's Rho algorithm, which still takes exponential time.

Given an elliptic curve mod p with a prime number of points, *any* point will generate the entire curve.

Point Addition

Point addition has several different cases, but here is one case to illustrate the basic idea. To add two points P and Q on E:



1. Draw a line through P and Q, which will intersect the curve in a third point R
2. We define $P + Q$ to be the reflection of R across the x-axis

Patent Issues

One hindrance to ECC moving forward is patent issues, especially by Certicom. But the situation is improving, as patents only last 20 years: many of Certicom's ECC-related patents are expiring by 2019.

Additionally, many of NSA's ECC-related patents have expired (or lapsed due to failure to pay fees).

Other Obstacles for ECC

- The math involved with elliptic curves is very complex compared to RSA
- Many low-level implementation details are difficult to get right
- ECC is fairly new (invented in 1985), whereas people have been factoring integers for thousands of years

Therefore, many people feel more comfortable with RSA, even if it means using 15,000-bit numbers.

References

- Bos, Kaihara, Kleinjung, Lenstra, and Montgomery, On the Security of 1024-bit RSA and 160-bit Elliptic Curve Cryptography
- Maletsky, RSA vs ECC Comparison for Embedded Systems
- Singh, Khan, Singh, A critical review on Elliptic Curve Cryptography
- Singh, Khan, Singh, Performance evaluation of RSA and Elliptic Curve Cryptography
- Trappe and Washington, *Introduction to Cryptography with Coding Theory*
- Washington, *Elliptic Curves: Number Theory and Cryptography*
- Discrete Logarithm Records (Elliptic Curves), Wikipedia
- ECC Patents, Wikipedia
- Elliptic-Curve Cryptography (History), Wikipedia
- RSA Factoring Challenge, Wikipedia