Pollard RHO Algorithm for Integer Factorization and Discrete Logarithm Problem

Nagaratna Hegde, PhD
Professor,
Vasavi College of Engineering,
Hyderabad-500031, India

P.Deepthi
Assistant Professor,
Bhoj Reddy Engineering College for Women,
Hyderabad-500059, India

ABATRACT
Security is must everywhere. We have to secure our data at all places like online banking, e-commerce etc., Cryptography doesn't have to be so cryptic. AES, DES, RSA, ECC -- there are so many ways to encrypt our data. Example: company's protecting customer credit card information, securing remote user connections to our network or protecting our intellectual property from digital piracy, we're using encryption every day. In 1980s, there was only one real choice -- the Data Encryption Standard (DES). Today, we have a broad selection of stronger, faster and better-designed algorithms. Now, the problem is to sort out which algorithm to be used. Elliptic curve cryptography (ECC) is one of the most powerful but least understood types of cryptography in wide use today. An increasing number of websites make extensive use of ECC to secure everything from customers HTTPS connections to how they pass data between data centers. So, it's important for end users to understand the technology behind any security system in order to trust it.

General Terms
Cryptography, Security, Elliptic curve cryptography

Keywords
Discrete logarithm problem, elliptic curve, integer factorization, pollard rho.

1. INTRODUCTION
Public key cryptography was invented in 1976 by Whitfield Diffie and Martin Hellman. It is also called as Diffie-Hellman encryption and asymmetric key encryption because it uses two keys instead of one key (symmetric encryption). An asymmetric cryptographic system uses two keys, one is a public key which is known to everyone and other one is a private or secret key known only to the recipient of the message. An important element to the public key system is that the public and private keys are related in such a way that only the public key can be used to encrypt messages and only the corresponding private key can be used to decrypt the message. It is difficult to find the private key if you know the public key.

Public-key systems, such as Pretty Good Privacy (PGP), are becoming popular for transmitting information via the Internet. They are extremely secure and relatively simple to use. The only difficulty with public-key systems is that you need to know the recipient's public key to encrypt a message for him or her.

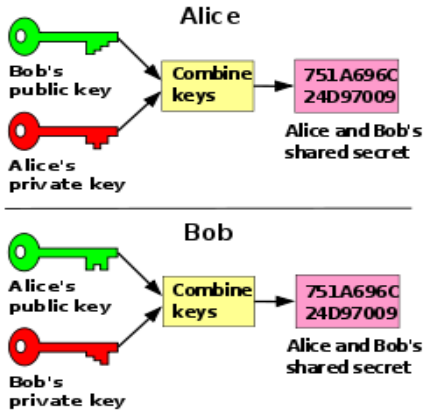

Figure: 1 Public Key Cryptography

2. RSA
The RSA algorithm is the most popular and best known public key cryptography system. Its security relies on the fact that factoring is slow and multiplication is fast. In general, a public key encryption system has two components, a public key and a private key. Encryption works by taking a message and applying a mathematical operation to it to get a random-looking number. Decryption takes the random looking number and applies a different operation to get back to the original number. Encryption with the public key can only be undone by decrypting with the private key.

Computers don't do well with arbitrarily large numbers. We can make sure that the numbers we are dealing with do not get too large by choosing a maximum number and only dealing with numbers less than the maximum. Any calculation that results in a number larger than the maximum gets wrapped around to a number in the valid range.

In RSA, this maximum value is obtained by multiplying two random prime numbers. The public and private keys are two specially chosen numbers that are greater than zero and less than the maximum value, call them public (pub) and private (priv). To encrypt a number you multiply it by itself pub times, making sure to wrap around when you hit the maximum. To decrypt a message, you multiply it by itself priv times and you get back to the original number. It sounds surprising, but it works. When discovered this property was a big breakthrough.

To generate a RSA key pair, first randomly pick any two prime numbers to obtain the maximum (max). Then pick a number to be the public key pub. If you know the two prime numbers, you can compute a corresponding private key priv from this public key. This is how factoring relates to breaking RSA — factoring the maximum number into its component primes allows you to compute someone's private key from the public key and decrypt their private messages.

Let's see this with an example. Take the prime numbers 13 and 7, their product gives us our maximum value of 91. Let's take our public encryption key to be the number 5. Then using the fact that we know 7 and 13 are the factors of 91 and applying an algorithm called the Extended Euclidean Algorithm, we get that the private key is the number 29.

These parameters (max: 91, pub: 5; priv: 29) define a fully functional RSA system. You can take a number and multiply it by itself 5 times to encrypt it, and then take that number and multiply it by itself 29 times and you get the original number back.

These factoring algorithms get more efficient as the size of the numbers being factored get larger. As the number gets larger (i.e. the key's bit length) the gap between the difficulty of factoring large numbers and multiplying large numbers is shrinking. As the resources available to decrypt numbers increase, the size of the keys needs to grow even faster. This is not a sustainable situation for mobile and low-powered devices that have limited computational power. The gap between factoring and multiplying is not sustainable in the long term.

3. ELLIPTIC CURVE

After the introduction of RSA and Diffie-Hellman, researchers explored other mathematics-based cryptographic solutions looking for other algorithms beyond factoring that would serve as good Trapdoor Functions. In 1985, cryptographic algorithms were proposed based on an esoteric branch of mathematics called elliptic curves.

What is an elliptic curve exactly: Unfortunately, unlike factoring something we all had to do for the first time in middle school most people aren't as familiar with the math around elliptic curves.

An elliptic curve is the set of points that satisfy a specific mathematical equation. The equation for an elliptic curve looks something like this:

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

That graphs to something that looks a bit like this:

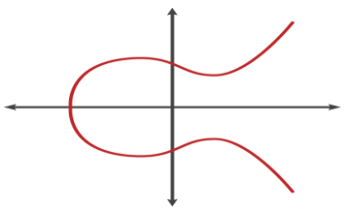

Figure: 2 Elliptic Curve

There are other representations of elliptic curves, but technically an elliptic curve is the set of points satisfying an equation in two variables with degree two in one of the variables and three in the other. An elliptic curve is not just a pretty picture; it also has some properties that make it a good setting for cryptography.

4. ELLIPTIC CURVE CRYPTO GRAPHY (ECC)

The properties and functions of elliptic curves have been studied in mathematics for 150 years. The use of elliptic curve within cryptography was first proposed in 1985, (separately) by Neal Koblitz from the University of Washington, and Victor Miller at IBM. An elliptic curve is not an ellipse (oval shape), but is represented as a looping line intersecting two axes (means lines on a graph used to indicate the position of a point). ECC is based on properties of a particular type of equation created from the mathematical group (a set of values for which operations can be performed on any two members of the group to produce a third member) derived from points where the line intersects the axes. Multiplying a point on the curve by a number will produce another point on the curve, but it is very difficult to find what number was used, even if you know the original point and the result. Equations based on elliptic curves have a characteristic that is very valuable for cryptographic purposes: they are relatively easy to perform, and extremely difficult to reverse.

5. ELLIPTIC CURVE DISCRETE LOGARITHM

The elliptic curve discrete logarithm is the hard problem under elliptic curve cryptography. Despite almost three decades of research, mathematicians still haven't found an algorithm to solve this problem that improves upon the naive approach. This means that for numbers of the same size, solving elliptic curve discrete logarithms is significantly harder than factoring. Since a more computationally intensive hard problem means a stronger cryptographic system, it follows that elliptic curve cryptosystems are harder to break than RSA and Diffie-Hellman.

To visualize how much harder it is to break, Lenstra recently introduced the concept of "Global Security." You can compute how much energy is needed to break a cryptographic algorithm, and compare that with how much water that energy could boil. This is a kind of cryptographic carbon footprint. By this measure, breaking a 228-bit RSA key requires less energy to than it takes to boil a teaspoon of water. Comparatively, breaking a 228-bit elliptic curve key requires enough energy to boil all the water on earth. For this level of security with RSA, you'd need a key with 2,380-bits.

With ECC, you can use smaller keys to get the same levels of security. The key size should be small because more and more cryptography is done on less powerful devices like mobile phones. Multiplying two prime numbers together is easier than factoring the product into its component parts, when the prime numbers start to get very long even just the multiplication step can take some time on a low powered device. If you continue to keep RSA secure by increasing the key length the cost will increase and slower cryptographic performance on the client. ECC appears to offer a better cryptography: high security with short, fast keys.

5.1. Elliptic curves in action: Applications

After a slow start, elliptic curve based algorithms are gaining popularity and the pace of adoption is accelerating. Elliptic curve cryptography is now used in a wide variety of applications: the U.S. government uses it to protect internal communications, the Tor project uses it to help assure anonymity, it is the mechanism used to prove ownership of bitcoins, it provides signatures in Apple's iMessage service, it is used to encrypt DNS information with DNS Curve, and it is the preferred method for authentication for secure web browsing over SSL/TLS. Cloud Flare uses elliptic curve cryptography to provide perfect forward secrecy which is essential for online privacy. First generation cryptographic algorithms like RSA and Diffie-Hellman are still the norm in most areas, but elliptic curve cryptography is quickly becoming the go-to solution for privacy and security online.

6. THE ELLIPTIC CURVE DISCRETE LOGARITHM PROBLEM

In the multiplicative group Zp*, the discrete logarithm problem is: given elements r and q of the group, and a prime p, find a number k such that $r = q^k \bmod p$. If the elliptic curve groups is described using multiplicative notation, then the elliptic curve discrete logarithm problem is: given points P and Q in the group, find a number that $P^k = Q$; k is called the discrete logarithm of Q to the base P. When the elliptic curve group is described using additive notation, the elliptic curve discrete logarithm problem is: given points P and Q in the group, find a number k such that $P^k = Q$

Example:
In the elliptic curve group defined by
$y^2 = x^3 + 9x + 17$ over F23,
What is the discrete logarithm k of Q = (4, 5) to the base P = (16, 5)?

One way to find k is to compute multiples of P until Q is found. The first few multiples of P are:

P = (16,5) 2P = (20,20) 3P = (14,14) 4P = (19,20) 5P = (13,10) 6P = (7,3) 7P = (8,7) 8P = (12,17) 9P = (4,5)

Since 9P = (4, 5) = Q, the discrete logarithm of Q to the base P is k = 9.

In a real application, k would be large enough such that it would be infeasible to determine k in this manner.

As far as this problem is concerned it is very hard to solve quickly. Most people have tried hard to solve discrete logarithm problem but not succeeded. It's easy to write a slow program to solve the discrete log problem.

7. POLLARD RHO

Pollard's rho algorithm is a special-purpose integer factorization algorithm. It was invented by John Pollard in 1975. It is particularly effective for composite numbers having a small prime factor.

The ρ algorithm is based on Floyd's cycle-finding algorithm and on the observation that (as in the birthday problem) t random numbers $x_1, x_2, \ldots, x_t$ in the range [1, n] will contain a repetition with probability P > 0.5.

Pollard's Rho Algorithm is a very interesting and quite accessible algorithm for factoring numbers. It is not the fastest algorithm by far but in practice it outperforms trial division by many orders of magnitude. It is based on very simple ideas that can be used elsewhere.

Pollard's rho factoring algorithm is a special-purpose algorithm for finding small non-trivial factors of an integer. Have a look at following proposition:

Let $f: S \rightarrow S$ be a random function, where $|S| = n$. Let further be $x_0 \in S$, at random. Consider the sequence $x_0, x_1, x_2, \ldots$ defined by $x_{i+1} = f(x_i)$. Since S is finite, the sequence must eventually cycle and be composed of a tail.

For finding out the length of the cycle, Floyd's cycle-finding algorithm is used:

In this method one starts with the pair (x_1, x_2) and iteratively computes (x_i, x_{2i}) from the previous pair (x_{i-1}, x_{2i-2}) until a duplicate $x_m = x_{2m}$ appears for some m. If the tail of the sequence has length λ and the cycle has length μ, so the first time when $x_m = x_{2m}$ is when $m = \mu (1 + \lambda / \mu)$.

Now Floyd's algorithm is utilized by Pollard's rho algorithm to find such a duplicate in the sequence of integers $y_0, y_1, y_2, \ldots, y_i \in Z \; \forall$ i. That sequence is defined by: $y_0 = 2$, $y_{i+1} = f(y_i) = (y_i^2 + 1) \bmod p$, $i \geq 0$. Now Floyd's algorithm is used to find y_m and y_{2m} with $y_m \equiv y_{2m} \pmod p$. Since p divides n, but is unknown, this is done by computing the terms $y_i \bmod n$ and testing whether is $\gcd(y_m - y_{2m}, n) > 1$. If such an m is found and if $\gcd(y_m - y_{2m}, n) < n$, then a non-trivial factor of n is obtained.

7.1. Algorithm

The algorithm takes as its inputs n, the integer to be factored and g(x), a polynomial p(x) computed modulo n. This will ensure that if p|n, and $x \equiv y \bmod p$, then $g(x) \equiv g(y) \bmod p$. In the original algorithm, $g(x) = x^2 - 1 \bmod n$, but nowadays it is more common to use $g(x) = x^2 + 1 \bmod n$. The output is either a non-trivial factor of n, or failure.

It performs the following steps:
1. $x \leftarrow 2$; $y \leftarrow 2$; $d \leftarrow 1$;
2. While d = 1:
   a. $x \leftarrow g(x)$
   b. $y \leftarrow g(g(y))$
   c. $d \leftarrow \gcd(|x - y|, n)$
3. If d = n, return failure.
4. Else, return d.

Note that this algorithm may fail to find a non-trivial factor even when n is composite. In that case, you can try again, using a starting value other than 2 or a different g(x). The name ρ algorithm comes from the fact that the values of x (mod d) eventually repeat with period d, resulting in a ρ shape when you graph the values.

8. POLLARD RHO ALGORITHM FOR DISCRETE LOGARITHM

Pollard's rho algorithm for logarithms is an algorithm introduced by John Pollard in 1978 for solving the discrete logarithm problem analogous to Pollard's rho algorithm for solving the Integer factorization problem.

Algorithm

Let G be a cyclic group of order p, and given

$$\alpha, \beta \in G$$

in G,
and a partition

$$G = S_0 \cup S_1 \cup S_2$$

Let

$$f : G \to G$$

be a map

$$f(x) = \begin{cases} \beta x & x \in S_0 \\ x^2 & x \in S_1 \\ \alpha x & x \in S_2 \end{cases}$$

and define maps

$$g : G \times \mathbb{Z} \to \mathbb{Z}$$

and

$$h : G \times \mathbb{Z} \to \mathbb{Z}$$

by

$$g(x,n) = \begin{cases} n & x \in S_0 \\ 2n \pmod p & x \in S_1 \\ n+1 \pmod p & x \in S_2 \end{cases}$$

Inputs 'a' a generator of G, 'b' an element of G
Output An integer x such that $a^x = b$, or failure

1. Initialize
   $a_0 \leftarrow 0$
   $b_0 \leftarrow 0$
   $x_0 \leftarrow 1 \in G$
   $i \leftarrow 1$

2. $x_i \leftarrow f(x_{i-1})$, $a_i \leftarrow g(x_{i-1},a_{i-1})$,
   $b_i \leftarrow h(x_{i-1},b_{i-1})$

3. $x_{2i} \leftarrow f(f(x_{2i-2}))$, $a_{2i} \leftarrow g(f(x_{2i-2}),g(x_{2i-2},a_{2i-2}))$,
   $b_{2i} \leftarrow h(f(x_{2i-2}),h(x_{2i-2},b_{2i-2}))$
4. If $x_i = x_{2i}$ then
   $r \leftarrow b_i - b_{2i}$
   If r = 0 return failure
   $x \leftarrow r^{-1} (a_{2i} - a_i) \bmod p$
   return x
   If $x_i \neq x_{2i}$ then $i \leftarrow i+1$, and go to step 2.

9. CONCLUSION

Many symmetric and asymmetric algorithms can be used for encryption, decryption, key exchange and digital signature. To break the prime factors of RSA algorithm we can use pollard rho integer factorization algorithm. As we have moved from RSA to elliptic curve cryptography because of its small key sizes we are trying to use pollard rho algorithm for discrete logarithms, which can be used to break the points on the elliptic curve.
In future a pollard rho algorithm can be modified to break elliptic curve cryptography.

10. REFRENCES

[1] The Discrete Logarithm Problem. Available online: http://modular.math.washington.edu/edu/124/lectures/lecture8/html/node5.html

[2] G Wojtenko, "Statistical properties of ECC-point and its impact on ECDLP", https://eprint.iacr.org/2007/092.pdf

[3] Mandy Zandra Seet, "ELLIPTIC CURVE CRYPTOGRAPHY Improving the Pollard-Rho Algorithm", https://www.maths.unsw.edu.au/sites/default/files/mandyseetthesis.pdf

[4] Pollard's rho algorithm. Available online: http://en.wikipedia.org/wiki/Pollard_rho_algorithm

[5] Factoing Large Numbers, A Great Way to Spend a Birthday. Available online: http://www4.ncsu.edu/lrbosko/Publications/Rho.pdf

[6] Connelly Barnes "Integer Factorization Algorithms" http://www.connellybarnes.com/documents/factoring.pdf

[7] Pollard rho Factorization Method. Available online: http://mathworld.wolfram.com/PollardRhoFactorizationMethod.html

[8] Computational Number Theory and Algebra. Available online: http://people.mpi-inf.mpg.de/~csaha/lectures/lec18.pdf

[9] A Quick Tutorial on Pollard's Rho Algorithm. http://www.cs.colorado.edu/srirams/classes/doku.php/pollard_rho_tutorial

[10] ELLIPTIC CURVE CRYPTOGRAPHY (ECC) https://www.certicom.com/ecc

[11] Nick Sullivan, "A (relatively easy to understand) primer on elliptic curve cryptography", http://arstechnica.com/security/2013/10/a-relatively-easy-to-understand-primer-on-elliptic-curve-cryptography/