DLM IN ELLIPTIC CURVE CRYPTOGRAPHY: AN INTRODUCTION TO CRYPTOGRAPHY

SIMON STEAD

Abstract. This report will concentrate on some of the important ways cryptography is used in the world today. The first half will be concerned with encryption, looking at the RSA encryption and Diffie Hellman Merkle Key Exchange. We then look at ways of breaking these, by considering factoring algorithms and solutions to the discreet logarithm problem.

1. RSA Encryption

RSA [1] Encryption is a very secure way to exchange encoded messages. We begin by giving a brief outline of the steps.

We start by choosing a public key, a modulus n that is the product of two large primes p, q, and what's known as a public key exponent, e, which is coprime to $\phi(n) = (p-1)(q-1)$ since p, q are prime and ϕ is multiplicative. To obtain our cypher text c, we encrypt our message m by computing $c = m^e \mod n$. Now since e is coprime to $\phi(n)$, it has a well defined inverse, that is there exists a d such that $de \equiv 1 \mod n$ [1] . Computing $c^d \equiv m^{ed} \equiv m \mod n$ clearly returns our message to us. The proof is due to Fermat's little theorem, as we can write

$$m^{ed} \equiv m^{ed-1}m \equiv (m^{p-1})^{k(q-1)}m \equiv 1m \mod p$$

as $ed - 1 \equiv (p-1)$. [2]

We can also verify that the encrypted message came from the correct source by means of a digital signature which can also be encrypted using RSA - one simply makes a hash value of the message, which is done algorithmically. This value is very small (typically is only 20-40 bytes long), and the signature is just this value raised to the power d ($\mod n$). Bob receiving the message can therefore decrypt the hash value to check it against the message, to see if either have been tampered with [3].

RSA is now the standard crypto systems used. A benefit is that companies and individuals are able to release their own extremely long public keys and exponents, typically 1024 or 2048 bits, which allows for extremely secure data transfer, as factorising a number this hard takes many years. It is low cost with respect to

programming power, so allows encryption and decryption to happen quickly.

2. Diffie Hellman Merkle Key Exchange [4] [5]

We also begin by a short description of this key exchange algorithm, with the help from the cryptographic crime fighting duo, Alice and Bob. Take a prime p, and a primitive root modulo p, say g. Note, this will generate the whole multiplicative group $\mathbb{Z}/p\mathbb{Z}$. Alice chooses a secret number a and computes $g^a \mod p$, sending it to Bob. Bob does the same with a secret number b, sending $g^b \mod p$ to Alice. Now they are both able to calculate $(g^a)^b \equiv (g^b)^a \equiv g^{ab} \mod p$, thereby obtaining a shared secret [4]. Notice this can take place on any finite group with generator g, which is exactly why we can perform the same feat using an elliptic curve over a finite field, under point addition.

This key exchange algorithm allows us to not only create a shared public key between two people, but for an arbitrary number. For simplicity we show how it works for 2^n people, but the method is exactly the same for any $N \in \mathbb{N}$.

The computationally naive way would be to have the n participants arrange in a circle, pass their secret to the person to their left, and compute $g^{s_1} \mod p$ for the secret they've just been passed. Then repeat by passing the secret they just been handed, computing $g^{s_1 \bullet s_2} \mod n$ and so on until they have $g^{\prod_{i=1}^{n} s_i} \mod p$, the shared secret for the whole group. This requires each participant to perform n exponentiations, which can be improved upon drastically.

Again Label the secret keys s_1 to s_n, and split the n people into two groups. One half applies all their exponents to the generator, obtaining $g^{\prod_{i=1}^{n/2} s_i} \mod p$, while the other half does the same and produces $g^{\prod_{i=n/2}^{n} s_i} \mod p$. They then swap their results, and half their groups again, applying their exponents to what they've been given, swapping again. This algorithm is repeated until all exponents have been multiplied, providing the shared secret $g^{\prod_{i=1}^{n} s_i} \mod p$.

For example, 8 people with secret keys a,b,c,d,e,f,g,h, are first split into two groups of four; g^{abcd} and g^{efgh} are the results at the end of the first pass. On the second, after breaking up into four groups, the results are $g^{abcd \bullet ef}$, $g^{abcd \bullet fg}$, $g^{ab \bullet efgh}$ and $g^{cd \bullet efgh}$. After swapping, dividing and exponentiating again, we get $g^{a \bullet cdefgh}$, $g^{b \bullet cdefgh}$, $g^{c \bullet abefgh}$, $g^{d \bullet abefgh}$, $g^{abcdef \bullet g}$, $g^{abcdef \bullet h}$, $g^{abcdgh \bullet e}$ and $g^{abcdgh \bullet f}$. Then all that is left is to for each participant to perform their own exponentiation so each have the shared secret $g^{abcdefgh}$.

This is much faster, and requires only $1 + \lfloor \log_2(n) \rfloor$ exponentiations.

3. Factoring Algorithms

The reason cryptography works so well, is because multiplication is NP, that is to say it's very easy to check whether the product of two numbers is a given answer, but factorising the product into its constituent factors is extremely difficult for suitably large n. Here we outline the modern algorithms used to factorise numbers, the rational sieve and the special number field sieve, which are both special cases of the general number field sieve.

First, a few definitions that will help us. Definition: A number $n \in \mathbb{N}$ is B-smooth if n factorises into prime numbers less than or equal to B. A factor base for $\mathbb{Z}$ is a set of primes $P = \{2, 3, 5\}$, and for an arbitrary number field $\mathbb{K}$ is just the set of prime ideals whose norm is bounded by a certain value.

3.1. **Rational Sieve.**

This is the simplest case, but illustrates the method very well. We first select a number to factorise, n, a factor base P, containing all primes less than some number B. Clearly if some $p \in P$ divides n then we're done, so assume without loss of generality that they do not.

What we're looking for are numbers $z < n$ such that both z and $z + n$ are B-smooth [8]. This provides a congruence relation between the prime factorisations of z and $z + n$. Let the prime factorisations of z and $z + n$ be $p_1^{a_1}...p_n^{a_n}$ and $p_1^{b_1}...p_n^{b_n}$ respectively, then $p_1^{a_1}...p_n^{a_n} \equiv p_1^{b_1}...p_n^{b_n} \mod n$. Once we find enough of these congruence relations, we can multiply them (as in linear algebra) until the exponents on both sides are even, i.e. both numbers are square; $x^2 \equiv y^2 \mod n$ [8].

This statement is equivalent to n dividing $(x - y)(x + y)$ due to the difference of two squares, so we can find a factorisation for n by performing the division algorithm to find $\gcd(x - y, n)$ and/or $\gcd(x + y, n)$. n will be equal to the lowest common multiple of these two gcds [8].

3.2. **Special Number Field Sieve.**

This sieve is a generalisation on the rational sieve, in the sense that the latter half of the algorithm (finding congruences $\mod p$ such that all powers are even and computing the greatest common divisors) is exactly the same. The difference is in how we choose our congruences.

Here n is the number we wish to factorise. We choose an irreducible polynomial $f \in \mathbb{Z}[x]$ with a root γ, such that there exists an integer $c < n$ for which $f(c) \equiv 0 \mod n$. This all means there exists a homomorphism η from $\mathbb{Z}[\gamma]$ to $\mathbb{Z}/p\mathbb{Z}$ such that $\eta(\gamma) = c$ [6] .

Now, instead of checking z and $z + n$ for B-smoothness as in the rational sieve, we look for integer pairs (a, b) that makes $a+bc$ B-smooth, and $a+b\gamma$ smooth with respect to the set of prime ideals with bounded norm. The latter is equivalent to $N(a + b\gamma)$ being N_{max}-smooth, where N_{max} is the maximum norm over all prime ideals in the factor base [7].

Once we have this criterion satisfied for a number of pairs (a, b) we find the image of $a + b\gamma$ in $\mathbb{Z}/p\mathbb{Z}$, giving us a congruence $a + bc \equiv \eta(a + b\gamma) \mod n$. We can then proceed as in the rational sieve.

3.3. **General Number Field Sieve.** Again, just a more general case of the previous (as the name would suggest), we consider *both* sides of the equation from arbitrary number fields.

For this we need two polynomials f, g of degrees d, e respectively, which are both irreducible over $\mathbb{Q}$ and have a common root modulo n. We take θ and τ as roots of our polynomials f and g respectively, and define the rings $\mathbb{Z}[\theta]$ and $\mathbb{Z}[\tau]$ in which to find our elements.[7]

We'll still be looking for congruent squares, so we can use the same method as before, but we need to further impose that these elements are to be norms of squares of elements in our number fields.

Consider the norm of an element $a-\theta b$ in $\mathbb{Z}[\theta]$, which is by definition an integer:

$$N(a - \theta b) = \prod_{i=1}^{d}(a - \theta_i b) = b^d \prod_{i=1}^{d}(\frac{a}{b} - \theta_i) = b^d f(\frac{a}{b}).$$

Labelling this u, and $N(a - \tau b) = b^d g(\frac{a}{b})$ as v, the problem boils down to finding integer pais (a, b) that make both u and v N_{max}-smooth. Again, exactly the same process occurs, we denote by η_θ and η_τ the homomorphisms from $\mathbb{Z}[\theta]$ and $\mathbb{Z}[\tau]$ to $\mathbb{Z}/p\mathbb{Z}$. [8] We then find a number of congruences of the form $\eta_\theta(u) \equiv \eta_\tau(v) \mod n$, which we combine to have one with even exponents, to then separate using the difference of two squares.

4. Discrete Logarithm Problem

We've discussed how to break RSA by means of a factoring algorithm, but what of Diffie Hellman Merkle key exchange, Is there a way to break that? Any eavesdropper during a key exchange can never see the full shared secret of course, as it is never sent. They would see only $g^{bcdefgh}$ etc for the 8 person example. The problem is then if $g^r = smodp$, and g and s are known for a given p, then what is r? This is known as the discrete logarithm problem, and also has no quick solution.

In 1978 J.M. Pollard created the "Kangaroo Algorithm" [11], [9] in order to solve this problem, and named so for the analogy Pollard used to describe it; a tame kangaroo jumping and setting a trap for a wild kangaroo, who starts at a random position, but them follows the kangaroo's footprints to the trap if it finds them. The algorithm works for any finite cyclic group.

Consider the steps of the kangaroo as a sequence $\{a_i\}_{i=0}^{n}$ where $a_i = g^x$ for some x and choose a starting value for the tame kangaroo, say, $a_0 = g^k$. Define a function f that tells you how far the kangaroo is to jump given it's position.

So $a_{i+1} = a_i g^{f(a_i}$. The 'trap' is set at m_n, and total distance from a_0 to a_n is $\sum_{i=0}^{n-1} f(a_i)$. So $a_n = g^{r+d}$ [11].
We then start another sequence of numbers $\{b_i\}_{i=0}^{n'}$ at the answer, so $b_0 = s$. Following the same function f for its step size, we will get a number $d_j = \sum_{i=0}^{j-1} f(b_i)$ for which the wild kangaroo's position $b_j = a_n$, the trap position.[11] Then we can find the exponent r by noting $a_n = g^{k+d} = b_j = sg^{d_j}$ and so $g^r \equiv g^{k+d-d_j} \mod p$, implying $r \equiv k + d - d_j \mod p$. This is therefore a solution to the discrete logarithm problem, however the complexity grows exponentially with the problem size, so becomes infeasible relatively quickly.[11]

The algorithm can also fail, depending on the choice of starting value, but all this means is we need to run the algorithm again with a different starting value and it will most likely work fine. The probability of success is dependent on the order of the finite cyclic group in question. In the presentation we demonstrated and discussed this card trick based on Pollard's algorithm [10], which had a probability of success of approximately 85%. We managed to increase the effectiveness of the trick by altering $f(x_{picturecard}) = 5$ to $f(x_{picturecard}) = 1$ instead. The probability of success was then calculated to be approximately 94%, verified by Chris Hughes.

References

[1] R.L. Rivest and A. Shamir and L. Adleman, A Method for Obtaining Digital Signatures and Public-Key Cryptosystems, Communications of the ACM, Vol 21 (1978), 120-126.
[2] Burt Kaliski , The Mathematics of the RSA Public-Key Cryptosystem, RSA Laboratories. (2006).
[3] Johannes Buchmann, Erik Dahmen, and Michael Zydlo. Post-Quantum Cryptography, chapter Hash-based Digital Signature Schemes, pages 35-94.
[4] Diffie W. and Hellman, M. New directions in cryptography, IEEE Trans. Inform. Theory IT-22 (Nov. 1976) 644-654.
[5] Diffie W. and Hellman, M. Exhaustive cryptanalysis of the NBS data encryption standard. Computer. 10 (June 1977), 74-84.
[6] A.K. Lenstra and H.W. Lenstra, The development of the number field sieve, Lecture Notes in Mathematics 1554, Springer-Verlag, Berlin (1993).
[7] Matthew E. Briggs, An Introduction to the General Number Field Sieve, Virginia Polytechnic Institute and State University (1998).
[8] Carl Pomerance, The Number Field Sieve, Proceedings of Symposia in Applied Mathematics, Volume 48, (1994).
[9] J. M. Pollard, Monte Carlo methods for index computation (mod p), Math. Comp., 32 (1978), 918924.
[10] J. M. Pollard, Kruskals card trick, Math. Gaz., 84 (July 2000), 4649.
[11] J.M. Pollard, Kangaroos, Monopoly and discrete logarithms, J. Crypt. 13(2000), 437-447.