SOME MATHEMATICAL FOUNDATIONS OF CRYPTOGRAPHY

PARKER HAVIZA

Abstract. This paper introduces three mathematical methods of sharing encrypted information: the Pohlig-Hellman Key Exchange and the ElGamal and RSA cryptosystems. These systems are based on the hardness of integer factorization and the Discrete Logarithm Problem. Definitions of these problems and attacks against these cryptosystems are discussed. This paper assumes no prior experience with cryptography or modular arithmetic and builds up from basic definitions.

Contents

1. Groundwork 1
1.1. General Concepts 1
1.2. Essential Formulas and Algorithms 2
2. Discrete Logarithm Problem 5
2.1. Collision Algorithms 5
2.2. Pohlig-Hellman Algorithm 6
3. Cryptosystems 7
3.1. Diffie-Hellman Key Exchange 7
3.2. ElGamal 8
3.3. RSA 8
4. Finding Prime Numbers 9
4.1. Primality Testing 9
4.2. Factorization 10
Acknowledgements 11
References 11

1. Groundwork

Before explaining the fundamental problems of cryptography or exploring the mathematics of individual cryptosystems, it will be helpful to cover some basic concepts which form a framework for later algorithms.

1.1. General Concepts.

Definition 1.1. A *finite commutative ring* R is a finite set with two operations, denoted $+$ (addition) and $\star$ (multiplication). Both operations are commutative and associative, and are linked by the distribution property

$$a \star (b + c) = a \star b + a \star c \qquad \text{for } a, b, c \in R.$$

The set contains identity elements for both operations. The set also contains additive inverses.

The basis for all calculations in this paper is *modular arithmetic*, also known informally as clock arithmetic for the way it describes the addition of times of day.

Definition 1.2. Let a, b, m be integers with m positive. We call m the *modulus*. We say a and b are *congruent modulo* m if

$$a = b + mk \quad \text{for some } k \geq 0.$$

We express this relation with the notation

$$a \equiv b \pmod{m}.$$

Proposition 1.3. If $a_1 \equiv a_2 \pmod{m}$ and $b_1 \equiv b_2 \pmod{m}$ then

$$a_1 \pm b_1 \equiv a_2 \pm b_2 \pmod{m} \quad \textit{and} \quad a_1 \cdot b_1 \equiv a_2 \cdot b_2 \pmod{m}.$$

Proof. By definition, $a_1 = a_2 + mk$ and $b_1 = b_2 + mj$ for some $m, j \geq 0$. Then we have

$$a_1 \pm b_1 \equiv a_2 \pm b_2 + m(k \pm j) \equiv a_2 \pm b_2 \pmod{m}$$

and

$$a_1 \cdot b_1 \equiv a_2 \cdot b_2 + m(b_2 k + a_2 j + mkj) \equiv a_2 \cdot b_2 \pmod{m}. \qquad \square$$

Definition 1.4. The *ring of integers modulo* m is the set

$$\{0,\ 1,\ ...,\ m-1\}$$

with addition and multiplication operations mod m. It is denoted $\mathbb{Z}/m\mathbb{Z}$.

Definition 1.5. A *finite field* is a finite ring with a multiplicative inverse for every element in the set except zero.

Definition 1.6. The *inverse of a non-zero integer* a *(mod* p*)* is the unique integer denoted a^{-1} that satisfies

$$a \cdot a^{-1} \equiv 1 \pmod{p}.$$

To "divide" both sides of an equation by a is to multiply both sides by a^{-1}.

Definition 1.7. A *unit* is an element of a ring that has a multiplicative inverse. The group of units modulo m is denoted $(\mathbb{Z}/m\mathbb{Z})^*$.

1.2. **Essential Formulas and Algorithms.** The following formulas and algorithms are essential for later proofs.

Algorithm 1. (Euclidean Algorithm) Assume without loss of generality that for integers a and b, $a > b$. Let $r_0 = a$ and $r_1 = b$. Repeat the following until $r_{i+1} = 0$:
(1) Starting with $i = 1$, divide r_{i-1} by r_i and denote the remainder as r_{i+1}
(2) Increment i

Proposition 1.8. The algorithm terminates when $r_i = \gcd(a, b)$.

Proof. For some integers q_i and the remainder after division r_i we can write the equations

$$a = bq_1 + r_2, \quad b = r_2 q_2 + r_3, \quad ..., \quad r_{t-2} = r_{t-1} q_{t-1} + r_t, \quad r_{t-1} = r_t q_t$$

denoting the greatest i as t. Note that $r_i > r_{i+1}$, so eventually we reach an r_{i+1} such that $r_{i+1} = 0 = r_{t+1}$. We can write the iteration step

$$r_{i+1} = r_{i-1} - r_i q_i$$

and see that a common divisor of r_{i-1} and r_i is also a divisor of r_{i+1}. Similarly, a common divisor of r_i and r_{i+1} is a divisor of r_{i-1}. It follows that

$$\gcd(r_{i-1}, r_i) = \gcd(r_i, r_{i+1}) \quad \text{for all } \ i = 1, 2, 3... \tag{1.9}$$

Since $r_{t-1} = r_t q_t + 0$ we have

$$\gcd(r_{t-1}, r_t) = \gcd(r_t q_t, r_t) = r_t.$$

Using $i = 1$ and (1.9) we have

$$\gcd(r_0, r_1) = \gcd(a, b) = r_t.$$

□

Proposition 1.10. (Extended Euclidean Algorithm) For positive integers a and b, there are always integers u and v such that

$$au + bv = \gcd(a, b).$$

Proof. Note that $r_1 = b$ and $r_2 = 1 \cdot a - q_1 \cdot b$ are integers. The iteration step tells us

$$r_{i+1} = r_{i-1} - r_i q_i.$$

By induction we see that r_{i-1} and r_i are linear integer combinations of a and b. In other words, for some integers x_1, x_2, y_1, and y_2 we can write

$$r_i = ax_1 + by_1 \quad \textit{and} \quad r_{i-1} = ax_2 + by_2.$$

Substituting these equations into the iteration step we have

$$r_{i+1} = r_{i-1} - r_i q_i = ax_2 + by_2 - (ax_1 + by_1)q_i = a(x_2 - y_1 q_i) + b(x_2 - y_1 q_i).$$

This gives r_{i+1} as a linear integer combination of a and b. Denote $u_{i+1} = x_2 - y_1 q_i$ and $v_{i+1} = x_2 - y_1 q_i$. This means that $r_t = \gcd(a, b)$ is also a linear integer combination of a and b with integers u_t and v_t. □

Proposition 1.11. Let p be prime. Then the ring of integers modulo p is a field (every non-zero integer has a multiplicative inverse).

Proof. If $a \in \mathbb{Z}/p\mathbb{Z}$ then $\gcd(a, p) = 1$. The Extended Euclidean Algorithm tells us that there are integers u, v such that

$$1 = au + pv.$$

This can be rewritten in the modular form

$$au = 1 - pv \equiv 1 \pmod{p}$$

So there exists an inverse u for every non-zero a in the ring. □

A ring of integers modulo a prime number is called a *field of prime order* and is denoted $\mathbb{F}_p$.

Perhaps the most important theorem underlying modern cryptography is Fermat's Little Theorem, which is the basis for many proofs that will follow in this paper. It states the following:

Theorem 1.12. (Fermat's Little Theorem) For any prime number p and integer a,

$$a^{p-1} \equiv \begin{cases} 1 \pmod{p} & p \nmid a \\ 0 \pmod{p} & p \mid a \end{cases} \tag{1.13}$$

Proof. If $p \mid a$ then $a \equiv 0 \pmod{p}$ so we will look at the case where $p \nmid a$. Then the list

$$a,\ 2a,\ 3a,\ ...,\ (p-1)a \pmod{p}$$

will contain $p-1$ distinct elements. There are $p-1$ distinct numbers between 1 and $p-1$, so this list must contain the same elements as

$$1,\ 2,\ ...,\ p-1 \pmod{p}.$$

These lists may not be in the same order, but the products of their elements are the same. We equate these products to see that

$$a \cdot 2a \cdot ... \cdot (p-1)a \equiv 1 \cdot 2 \cdot ... \cdot (p-1) \pmod{p}.$$

After factoring we see that

$$a^{p-1}(p-1)! \equiv (p-1)! \pmod{p}.$$

And noting that $(p-1)!$ is non-zero and hence invertible mod p by Proposition 1.11 we obtain

$$a^{p-1} \equiv 1 \pmod{p}. \qquad \square$$

We can easily calculate inverses mod p using Theorem 1.12. Note that

$$a \cdot a^{p-2} \equiv a^{p-1} \equiv 1 \pmod{p}.$$

Thus the inverse can be found by computing $a^{p-2} \pmod{p}$, so we can easily perform division in a field of finite prime order.

If we are working modulo a composite number in the ring $\mathbb{Z}/(pq)\mathbb{Z}$ where p, q are distinct primes, we can still make use of Fermat's Little Theorem with Euler's Formula.

Theorem 1.14. (Euler's Formula) Given two distinct primes p and q, let

$$g = \gcd((p-1)(q-1)).$$

Then

$$a^{(p-1)(q-1)/g} \equiv 1 \pmod{pq}.$$

Proof. Considering the equation mod p we have

$$a^{(p-1)(q-1)/g} = (a^{p-1})^{(q-1)/g} \equiv 1^{(q-1)/g} \equiv 1 \pmod{p}$$

Repeating this process mod q shows that $a^{(p-1)(q-1)/g} \equiv 1 \pmod{q}$. Then because $a^{(p-1)(q-1)/g} - 1$ is divisible by both p and q, it is divisible by pq. $\square$

Similar formulas can be derived when the modulus can be factored into more than two primes. This is not necessary for cryptographic purposes, since the modulus is usually chosen as the product of two large primes in order to make factorization computationally difficult.

The Chinese Remainder Theorem is a way of solving a system of equations in modular arithmetic.

Algorithm 2. Let the input of this algorithm be integers $a_1...a_t$ and $m_1...m_t$ where $\gcd(m_i, m_j) = 1$ for $i \neq j$. Let $x_1 = a_1$.
Repeat the following steps for $2 \leq i \leq t$:

(1) Let integer y_i satisfy

$$y_i \equiv (m_1 m_2 \cdots m_{i-1})^{-1}(a_i - x_{i-1}) \pmod{m_i}. \tag{1.15}$$

(2) Let

$$x_i = x_{i-1} + y_i(m_1 m_2 \cdots m_{i-1}). \tag{1.16}$$

Complete the algorithm by outputting x_t.

Proposition 1.17. Denote x as the output of Algorithm 2. Then $x \equiv a_i \pmod{m_i}$ for all $1 \leq i \leq t$ and is uniquely determined mod $m_1 m_2 ... m_t$.

Proof. Plugging (1.15) into (1.16) for some integer q_i we have

$$x_i = x_{i-1} + (a_i - x_{i-1}) + m_i q_i(m_1 m_2 \cdots m_{i-1}) = a_i + m_i q_i(m_1 m_2 \cdots m_{i-1}).$$

This simplifies to

$$x_i \equiv a_i \pmod{m_i}.$$

Note then that (1.16) can be rewritten as

$$x_i \equiv x_{i-1} \equiv a_{i-1} \pmod{m_{i-1}}.$$

Hence, by induction on i, x_i satisfies $x_i \equiv a_j \pmod{m_j}$ for all $j < i$. □

2. Discrete Logarithm Problem

Cryptography relies on operations known as trapdoor functions which are easy to compute but take prohibitively long to invert using modern computing power. One example of a potential trapdoor function is multiplying two large primes and then attempting to recover the product's original factors. Many modern cryptosystems rely on the Discrete Logarithm Problem, or DLP, which is a trapdoor function.

Definition 2.1. A *primitive root* g is a number such that every element of $\mathbb{F}_p$ can be written as g raised to some power.

Thus, since $g^{p-1} \equiv 1 \pmod{p}$ by Fermat's Little Theorem, the list

$$g,\ g^2,\ g^3,\ ...,\ g^{p-1}$$

is a list of every element in $\mathbb{F}_p^*$, the set of units of $\mathbb{F}_p$.

The *Discrete Logarithm Problem* is the question of finding an exponent x such that

$$g^x \equiv h \pmod{N} \tag{2.2}$$

for integers g and N. For cryptographic purposes, we are interested in the Discrete Logarithm Problem where g is a primitive root and N is either prime or the product of primes.

2.1. Collision Algorithms.

A collision algorithm is a brute-force method of solving an equation. The general strategy in collision algorithms is to create two lists and find a match between them. We can apply collision algorithms to the DLP.

Algorithm 3. (Shank's Babystep-Giantstep Algorithm) Choose an integer n such that $\sqrt{N} < n < N$ where N is the modulus of the particular Discrete Logarithm Problem. Generate the lists

$$e,\ g,\ g^2,\ g^3,\ ...,\ g^n \qquad \text{“babystep”}$$

and

$$h,\ h \cdot g^{-n},\ h \cdot g^{-2n},\ ...,\ h \cdot g^{-n^2} \qquad \text{“giantstep”}.$$

Find a match between these lists of the form $g^i = h \cdot g^{-jn}$.

Proposition 2.3. Given a match, $x = i + jn$ solves the discrete logarithm $g^x \equiv h \pmod{N}$.

Proof. For integers q, r let $x = nq + r$ where $0 \leq n < r$. Then because $1 \leq x < N$ and $n > \sqrt{N}$ we have

$$q = \frac{x - r}{n} < \frac{N}{n} < \frac{n^2}{n} < n.$$

Rewriting the Discrete Logarithm Problem we have

$$g^x = g^{r-nq} = h$$

So,

$$g^r = h \cdot g^{nq}$$

where g^r is in the "babystep" list and g^{qn} is in the "giantstep" list. Thus this algorithm will always return a match and have a solution of the form $x = i + jn$. □

Another collision algorithm uses random numbers to create a high probability of finding a match in relatively few steps.

Proposition 2.4. Write the Discrete Logarithm Problem letting $x = y - z$ so that

$$g^y \equiv h \cdot g^z \pmod{p}$$

for prime p and primitive root g. Choose random values of y and z to form the lists

$$g^{y_1},\ g^{y_2},\ ...,\ g^{y_n} \qquad 1 < y_i < p - 1 \tag{2.5}$$

and

$$h \cdot g^{z_1},\ h \cdot g^{z_2},\ ...,\ h \cdot g^{z_n}. \tag{2.6}$$

Then for randomly generated y_i and z_j of a matching pair of numbers, $x = y - z$ has a $1 - e^{-n^2/(p-1)}$ chance of solving the discrete logarithm problem $g^x \equiv h \pmod{p}$.

Proof. The list (2.5) is a size-n subset of the size-$(p-1)$ set $\{g^1, g^2, ..., g^{p-1}\}$ because g is a primitive root. The probability of getting a match between (2.5) and at least one member of (2.6) is

$$\text{1-Pr(No match)=1-}(\frac{p-1-n}{n})^n = 1 - (1 - \frac{n}{p-1})^n.$$

After noting that $e^{-x} \geq 1 - x$ this gives us the lower bound $1 - e^{-n^2/(p-1)}$. □

2.2. **Pohlig-Hellman Algorithm.** The ability to factor $p - 1$ is crucial in the Pohlig-Hellman Algorithm for solving the DLP.

Algorithm 4. Given the discrete logarithm problem $g^x \equiv h \pmod{p}$ for prime p and primitive root g, suppose $L = p - 1$ can be factored in the form

$$L = q_1^{k_1},\ q_2^{k_2},\ ...,\ q_t^{k_t}.$$

For $1 \leq i \leq t$ let g_i and h_i be defined by

$$g_i = g^{L/q_i^{k_i}} \quad \textit{and} \quad h_i = h^{L/q_i^{k_i}}.$$

Solve the t discrete logarithms of the form

$$g_i^{y_i} \equiv h_i \pmod{p}$$

using Shank's Algorithm or by using the Pohlig-Hellman algorithm again. Use the Chinese Remainder Theorem to find x by solving

$$x \equiv y_1 \pmod{q_1^{k_1}},\ x \equiv y_2 \pmod{q_2^{k_2}},\ ...,\ x \equiv y_1 \pmod{q_t^{k_t}}.$$

Proposition 2.7. This x solves the discrete logarithm $g^x \equiv h \pmod{p}$.

Proof. Let $x \equiv y_i \pmod{q_i^{k_i}}$. This means $x = y_i + q_i^{k_i} j_i$ for some j_i. Then we have

$$\begin{aligned}(g^x)^{L/q_i^{k_i}} \pmod{p} &\equiv (g^{L/q_i^{k_i}})^{y_i} \cdot g^{Lj_i} \pmod{p} \\ &\equiv (g_i)_i^y \cdot 1_i^j \pmod{p} \\ &\equiv h_i \pmod{p} \\ &\equiv h^{L/q_i^{k_i}} \pmod{p}.\end{aligned}$$

Writing $(g^x)^{L/q_i^{k_i}} \equiv h^{L/q_i^{k_i}} \pmod{p}$ in logarithmic form we have

$$(L/q_i^{k_i}) \cdot x \equiv (L/q_i^{k_i}) \cdot \log_g(h) \pmod{p}. \tag{2.8}$$

Since the greatest common divisor of all $L/q_i^{k_i}$ is 1, the Extended Euclidean Algorithm tells us there are integers $c_1 ... c_t$ which satisfy

$$L/q_1^{k_1} \cdot c_1 + L/q_2^{k_2} \cdot c_2 + \ ... \ + L/q_t^{k_t} \cdot c_t = 1. \tag{2.9}$$

If we multiply both sides of 2.8 by c_i and sum over $1 \le i \le t$ we get

$$\sum_{i=1}^{t} L/q_i^{k_i} \cdot c_i \cdot x \equiv \sum_{i=1}^{t} L/q_i^{k_i} \cdot c_i \cdot \log_g(h) \pmod{p}.$$

Using (2.9) we have

$$x \equiv \log_g(h) \pmod{p}.$$ □

3. Cryptosystems

Cryptography is the study of privately transmitting a message across potentially open channels. Encryption is the process of obscuring a message from everyone except the intended recipient, who decrypts the message. Together encryption and decryption algorithms form a cryptosystem. The following cryptosystems are based on the difficulty of either factoring a product of large primes or solving the discrete logarithm problem.

3.1. Diffie-Hellman Key Exchange.

Definition 3.1. A *key* is a secret number used in a cryptosystem when encrypting and decrypting messages. Without it, messages are hard to recover from the encrypted text.

One major question in cryptography deals with how to establish a secret key between two people who can only communicate across open channels. One method was proposed by Whitfield Diffie and Martin Hellman in 1978 [3].

Algorithm 5. (Diffie-Hellman Key Exchange) Let prime p and primitive root g be known.
Person A: choose a secret integer α and send $a \equiv g^\alpha \pmod{p}$. Calculate

$$b^\alpha \pmod{p}.$$

Person B choose a secret integer β and send $b \equiv g^\beta \pmod{p}$. Calculate

$$a^\beta \pmod{p}.$$

Proposition 3.2. Persons A and B now share the same key.

Proof. It must be shown that $b^\alpha \pmod{p} \equiv a^\beta \pmod{p}$. This is the shared key.

$$\begin{aligned} b^\alpha \pmod{p} &\equiv (g^\beta)^\alpha \pmod{p} \\ &\equiv (g^\alpha)^\beta \pmod{p} \\ &\equiv a^\beta \pmod{p} \end{aligned}$$

□

To uncover the key, a third person intercepting the publicly sent information would have to be able to find α from $g^\alpha \pmod{p}$ or β from $g^\beta \pmod{p}$. This is the Discrete Logarithm Problem. Numbers in cryptosystems are chosen sufficiently large that it would take decades to find these values with current algorithms. An attacker could also find a way of finding $g^{\alpha\beta}$ from either g^α or g^β, but no efficient method for this is known.

3.2. **ElGamal.** The ElGamal cryptosystem is a means of encrypting and decrypting a secret message m.

Algorithm 6. Encryption Scheme: Let prime p and primitive root g be known.
Person A: Select a random private key α and distribute a public key $a \equiv g^\alpha \pmod{p}$.
Person B: choose a random private key k and private message m and send

$$c_1 \equiv g^k \pmod{p} \quad \textit{and} \quad c_2 \equiv ma^k \pmod{p}.$$

Proposition 3.3. The message can be decrypted by computing

$$m \equiv (c_1^\alpha)^{-1} \cdot c_2 \pmod{p}.$$

Proof. Person A knows α, so they can calculate

$$\begin{aligned} (c_1^\alpha)^{-1} \cdot c_2 &\equiv (g^{ak})^{-1} \cdot (ma^k) \pmod{p} \\ &\equiv (g^{ak})^{-1} \cdot (m(g^a)^k) \pmod{p} \\ &\equiv m \pmod{p} \end{aligned}$$

□

Again, the strategy in breaking this encryption lies in the Discrete Logarithm Problem of discovering α from g^α.

3.3. **RSA.** Another cryptosystem is RSA, introduced by Rivest, Shamir, and Adelman in their 1978 paper [2]. The security of this system is based on the factoring of the product of large primes.

Algorithm 7. Person A: distribute a product of primes $N = pq$ and an *encryption exponent* e such that $\gcd(e, (p-1)(q-1)) = 1$. (We denote the inverse of e as d.)
Person B: choose a secret message m and send

$$c \equiv m^e \pmod{N}.$$

Person A: compute d which by definition satisfies

$$ed \equiv 1 \pmod{(p-1)(q-1)}.$$

Proposition 3.4. Then the message can be decoded with the formula

$$m \equiv c^d \pmod{N}.$$

Proof. By the Euclidean Algorithm there exists integers d and k such that $de = 1 - (p-1)(q-1) \cdot k$, so we know that there exists an inverse for every e.

$$\begin{aligned} c^d &\equiv m^{de} \pmod{N} \\ &\equiv m^{1-(p-1)(q-1)\cdot k} \pmod{N} \\ &\equiv m \cdot (m^{p-1}m^{q-1})^{-k} \pmod{N} \\ &\equiv m \cdot (1 \cdot 1)^{-k} \pmod{N} \\ &\equiv m \pmod{N} \end{aligned}$$

□

It takes a lot of computing power to find d for a large p, q unless the factorization of N is known, making the sent message transparent for the publisher of N but not for other recipients of the encoded text.

4. Finding Prime Numbers

4.1. **Primality Testing.** If large primes are necessary for these cryptosystems, how do we find them? It is necessary to determine whether a randomly chosen large number is prime. It is only necessary to look at large odd numbers because the only even prime is 2.

Definition 4.1. A *witness* a for the compositeness of a number n is an integer such that

$$a^n \not\equiv a \pmod{n}.$$

Indeed, multiplying both sides of (1.13) by a gives $a^n \equiv a \pmod{n}$ for prime n. However, for some integers a Fermat's Little Theorem is still satisfied for composite n. This brings us to the Miller-Rabin Witness Test.

Algorithm 8. (Miller-Rabin Witness Test) Given an n written as $n - 1 = 2^k q$ for odd integer q, choose an integer a such that $a \nmid n$, and check the following conditions:

(1) $a^q \equiv 1 \pmod{n}$
(2) $a^{2^j q} \equiv -1 \pmod{n}$ for some $j < k$

Stop test if a number violating one of these conditions is found.

Proposition 4.2. If n is prime, both conditions are true.

Proof. Consider the list

$$a^q,\ a^{2q},\ a^{4q},\ ...,\ a^{2^k q}.$$

Note that if n is prime $a^{2^k q} \equiv a^{n-1} \equiv 1 \pmod{n}$ by Fermat's Little Theorem. Each number is the square of the previous number with the last term being 1. Thus either the first term in the list is 1, or some number in the list is 1 when squared. In other words, if $a^q \not\equiv 1 \pmod{n}$ there is an integer b such that $b \not\equiv 1 \pmod{n}$ and $b^2 \equiv 1 \pmod{n}$, hence $b = -1$. So for some $j < k$, $a^{2^j q} \equiv -1 \pmod{n}$. □

To test whether a number is prime, the Miller-Rabin test chooses several values for a in the hopes of discovering a witness for the compositeness of n (a number that violates one of the conditions). The chance of n being prime improves with every non-witness. This test does not conclusively prove primality, but can show that n is likely to be prime.

4.2. **Factorization.** Modern cryptography relies on numbers that are hard to factor. If the modulus used in a cryptosystem can be factored, the corresponding cryptosystem can be cracked. The simplest way to factor a large number is divide it by every integer starting with 1, hoping to get an integer. The numbers that are used in cryptosystems are sufficiently large to make this approach impossible with modern computing power. There are other methods to consider.

Proposition 4.3. (Pollard's p-1 Method) Given $N = pq$ for p and q prime, choose L such that $p-1 \mid L$ and $q-1 \nmid L$ and choose a such that $a^k \not\equiv 1 \pmod{q}$ where $k \equiv L \pmod{q-1}$. Then p can be computed by the formula

$$p = \gcd(a^L - 1, N).$$

Proof. By the definition of divisor, $L = i(p-1) = j(q-1) + k$ for nonzero integers i, j, k where $1 \leq k < q-1$. Using Fermat's Little Theorem shows that

$$a^L = a^{i(p-1)} = (a^{p-1})^i \equiv 1^i \equiv 1 \pmod{p}$$

and

$$a^L = a^{j(q-1)+k} = (a^{q-1})^j a^k \equiv 1^j \cdot a^k \equiv a^k \pmod{q}.$$

From the equations above, we see that $p \mid a^L - 1$ and $q \nmid a^L - 1$. Thus $p = \gcd(a^L - 1, N)$. □

If k is small and q is large, there is a good probability that $a^k \not\equiv 1 \pmod{q}$ does not hold for a randomly chosen a. In fact, if $p-1$ is a product of small primes it suffices to check $\gcd(a^{n!} - 1, N)$ for $n = 1, 2, 3...$ because if $p-1$ can be written as a product of small primes it will divide $n!$ for a small value of n. Thus we see that Pollard's Method is a fast factoring method in the case that $p-1$ factors into small primes.

Another method of factoring relies on a very simple formula, $X^2 - Y^2 = (X - Y)(X + Y)$, which we can adapt to factor N.

Algorithm 9. Given $N = pq$ where p, q are primes, calculate

$$c = N + b^2 \quad \text{for} \quad b = 1, 2, 3...$$

check each c to see if it can be written as z^2 (so c is a perfect square). Stop when a perfect square is found.

Proposition 4.4. If such a c is found, N can be factored as $p = z - b$ and $q = z + b$.

Proof. $N = pq = z^2 - b^2 = (z-b)(z+b)$, giving $p = z - b$ and $q = z + b$. □

As b increases, each calculation takes more computing time. We can try using multiples of N to improve the efficiency of the algorithm by only checking small b values. If we look at equations of the form $kN = (z-b)(z+b)$ then some factor of N may divide both $(z-b)$ and $(z+b)$. If so, we can check values of the form

$$c = kN + b^2 \quad \text{for} \quad b = 1, 2, 3...$$

to find $(z-b)$ and $(z+b)$. One of these terms will be divisible by k because $N = pq$, so we find

$$p = \gcd(N, z-b) \quad and \quad q = \gcd(N, z+b)$$

to complete a factorization of N.

Acknowledgements

It is my pleasure to thank my mentor John Wilmes for his wisdom, guidance, and patience. This paper was made possible by his strong desire to see me truly understand the underlying concepts. I would also like to thank Professor Peter May for organizing an amazing REU and Professor Laszlo Babai for his stimulating lectures on linear algebra.

References

[1] Jeffrey Hoffstein, Jill Piper and Joseph H. Silverman. An Introduction to Mathematical Cryptography. Springer, 2008.

[2] R. L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. Comm. ACM, 21(2):120-126, 1978.

[3] W. Diffie and M. E. Hellman. New directions in cryptography. IEEE Trans. Information Theory, IT-22(6):644-654, 1976.