RSA Encryption

An Application of Abstract Algebra in Cryptography

Angelica Davis
Drew Gottman
Amelia Hartke

April 25, 2016

Background and General Description

Cryptography is used in many everyday functions, mainly online. Before the internet, cryptography was used in sending secret codes. The internet now uses cryptography all the time to send secure codes. Cryptography helps with authentication of signatures, time stamping, electronic money, secure network communications, and disk encryption [11]. These functions all require the receiver to be sure of the sender. Otherwise people could make fake bank accounts and send "money," but with cryptography there's an electronic signature with every transaction certifying the bank account exists from that bank. Basically, cryptography is an internet security badge allowing us to know who belongs and who doesn't.

The earliest extant cipher, the Caesar cipher, makes elementary use of group theory through a simple shift. Specifically, we number each letter in the alphabet according to its cardinality e.g. $A = 00, B = 01, C = 02, \ldots, Z = 25$; thus the alphabet, as the numbers given, forms the group $\mathbb{Z}_{26}$. We encrypt each letter by "shifting" it - i.e. $y = x + e \mod 26$ where x is the number corresponding to the letter we wish to encrypt, e is the encryption shift, and y is the encrypted output. Since $\mathbb{Z}_{26}$ is a group, we are guaranteed closure and a unique inverse for each element; hence, this group yields a unique encryption, and decryption, for each message. Unfortunately, the Caesar Cipher is easily broken if we "fingerprint" the encrypted message, looking for patterns that could correspond to common articles found in the English language such as "the" or "a", enabling us to easily guess the shift. It can even be broken by a brute-force algorithm, by trying each of the 25 possible shifts.

Fortunately, group theory has yielded more sophisticated cryptosystems - the most nebulous of which are public key systems. The advantages of public key systems over traditional cryptographic systems are obvious: security, key size, and efficiency. Three prominent public key systems - Diffie-Helman, RSA, and Elliptic Curve - rely on the use of *trapdoor functions*, which are functions that are easy to process in one direction, but are difficult to process in the other direction [12]. Two public key cryptosystems - Diffie-Hellman and RSA - encrypt messages under the multiplicative group $\mathbb{Z}_n$ where n is the product of two "sufficiently large" primes. Such a cryptographic system forms a trapdoor function because it is easy to multiply two large primes together, but difficult to find the prime factors of a large number.

Elliptic Curve Cryptographic systems form an even more difficult trapdoor function, owing to the subgroup structure of an elliptic curve in a finite field. Elliptic curves owe their advantage over traditional public key systems, such as RSA and Diffie-Hellman, to the difficulty of solving both the discrete-log problem and the Diffie-Hellman problem. At best, elliptic curve algorithms can be solved in exponential time under sufficient conditions. This means we can use a smaller key size that has a comparable level of security to RSA or Diffie-Hellman.

This is hardly a complete survey of abstract algebra's role in cryptography, but the above examples give a flavor for its ubiquitous role in modern cyptosystems. The following sections will provide greater specificity and rigorous underpinnings for the RSA and elliptic curve cryptosystems.

Results from Abstract Algebra

The RSA encryption was born out of a need for a code that was both difficult to break and easy to send. Making a code difficult for an enemy to solve is easy, but this normally involves the code also being only good for one use. The code must also be easy to send to the other party. Sending a huge list of deciphering keys takes too long and is inefficient. Thus using words and/or letters as codes would be exhausting. But, using numbers, which can easily be used by a computer, was an ideal solution. And thus the challenge arises of the code being relatively easy for the right computer to break, but extremely challenging for the wrong computer to solve. Luckily, there's group theory. Using the group of mod (n) under multiplication and large numbers with prime factorization, a cryptosystem is found meeting the criteria [8].

RSA, which stands for the last name initials of the developers of the code, uses group modulo multiplication to convert an integer into an encrypted integer and then the inverse to convert back to the decrypted integer yielding the original message. This is effectively unbreakable because of the use of the large numbers which have a huge number of factorization options making a computer take years to break the code.

The other huge advantage the RSA cryptosystem has over previously existing encryption methods is the use of asymmetric keys. Before the development of asymmetric key cryptography, all forms of symmetric key cryptography faced a serious security issue: key distribution [13]. With a symmetric key system, both the sender and receiver must have the same key to communicate securely. In the context of the worldwide web, it is simply not feasible to generate and securely share a unique key for each and every communication, or even for the most critical communications. At best this could mean thousands of symmetric keys per user, where a new one is generated for each new conversation.

The trapdoor function the RSA algorithm uses is particularly valuable as its computation can be done asymmetrically. That is, a single user can have a hard to crack private key that remains known to that user only. This makes it easy to communicate securely with any number of users, who can all have access to the same public key without ever knowing the private key. So how exactly does the RSA algorithm achieve such a convenient and secure cryptosystem? In general terms, the algorithm proceeds as follows [2]:

User A has a public key pair (e,n) and a private key pair (d,n), where n is the product of two primes, e is relatively prime to $\phi(n) = (p-1)(q-1)$ and d is the multiplicative inverse of e under modular multiplication. Suppose user B wants to send a message to user A. Then user B converts the textual message into a numeric representation, ASCII for instance, that will henceforth be referred to as M for convenience. The message M is then broken into partitions so that $0 \leq |M_i| < n$, where $|M_i|$ represents the numerical value of a partition. Then to encrypt the message, user B accesses user A's public key to compute $E(M_i) = M_i^e \mod n$ for each M_i. This essentially encrypts the message under the multiplicative group $\mathbb{Z}_n$.

The receiver, A, can then easily decrypt the message by computing $E(M_i)^d \mod n$ for each $E(M_i)$. As stated in Fermat's Corollary, this computation yields $E(M_i)^d \mod n = (M_i^e)^d \mod n = M_i \mod n$. Then, by the earlier restriction of having $0 \leq |M_i| < n$, $M_i \mod n = M_i$ and recipient A has effectively decrypted the the message. The method of numerically defining the message is a common, publicly available process, so that mapping the numerical M_i to the textual information is no problem for user A.

RSA Proof of Correctness

Choose $n = pq$ where p and q are distinct primes. Also choose e, where e is relatively prime to $\phi(n) = (p-1)(q-1)$. Find d, the multiplicative inverse of e, with $ed = 1 \mod \phi(n)$. Also note that $ed = 1 \mod \phi(n)$ is equivalent to $ed = k\phi(n) + 1$ for some $k \in \mathbb{Z}$.

The goal is to prove that $(M^e)^d = M \mod n$, where M is defined as being $0 \leq M \leq n-1$. The first step is using Euler's Theorem, specifically the Fermat Corollary. This corollary says that if $(M, p) = 1$, then $M^{p-1} = 1 \mod p$. So then $(M^e)^d = M^{ed} = M^{k\phi+1}$. Also known is that $M^1 = M \mod p$. Using properties with exponents, $M^{k\phi(n)+1} = M^{k\phi(n)}M^1$. And now using modulo properties $M^{k\phi(n)}M^1$ is defined as $M^{k\phi(n)}M = M^{k\phi(n)}M \mod p = M^{k(p-1)(q-1)}M \mod p$. And finally using exponential properties it is found that this equals $(M^{p-1})^{k(q-1)}M \mod p = 1^{k(q-1)}M \mod p = M \mod p$. Thus proving that $M^{ed} = M \mod p$.

This same argument works in showing $M^{ed} = M \mod q$. Which ultimately shows that both distinct primes p and q divide $M^{ed} - M$ and therefore $M^{ed} = (M^e)^d = M \mod n$. ■

Current Status

Although the RSA algorithm for encryption was historically a huge breakthrough in digital security and remains an intellectual triumph, the dazzlingly fast rate of development in high computing power is increasingly exposing serious vulnerabilities in the widely used security system. Because of how universally the public key encryption model is depended upon for information security and privacy, the growing threat brings with it an urgent need for a more secure system.

From a theoretical standpoint, the widely used RSA system remains secure. The standard key size recommended for a truly secure cryptosystem is at least 1024-bits (617 decimal digits) [6]. A key size this large requires years to factor with the best known factorization algorithm. Currently this fastest factoring algorithm is the General Number Field Sieve (GNFS), which has a run time of roughly $e^{n^{1/3}log^{2/3}(n)}$ on an n-bit integer input [1] (an attack that factors the modulus is referred to as a brute force attack).

While in general the RSA algorithm is thought to be secure against brute force attacks, this assessment assumes that any implementation uses a sufficiently large modulus for generating the keys. Unfortunately, there are an alarming number of services that still rely on 512-bit keys even though integers this size have repeatedly been shown to be vulnerable to brute force attacks [3].

Quantum Computing

Seventeen years after the publication of the RSA algorithm, Peter Shor, a professor of mathematics at MIT, published an algorithm for large number factoring on a quantum machine [9]. Unlike familiar, bit driven computers, a quantum computer is powered by qubits. Qubits are a "superposition" bit state, where the value of a qubit is simultaneously 1 or 0 or both. This architectural difference is expected to optimize various computations, in particular the factoring of large numbers.

At the time of Shor's publication, no qubit powered machines existed with which to test the algorithm. However in 2001, a team of Quantalab researchers at MIT lead by physicist and electrical engineer, Isaac Chuang, had successfully created a quantum system and tested Shor's algorithm by successfully factoring the number 15 [9]. This implementation was limited, however, to factoring small numbers only. Yet motivated by this success, Chuang and other scholars were determined to pursue a scalable implementation. This scalable implementation has just recently been realized, with the results published in this year's March issue of Science magazine.

Despite this significant step towards undermining the security of the RSA algorithm, the recent implementation is still limited to the primitive state of the quantum computers currently being built, and as such, the now scalable algorithm has actually yet to be implemented in its full-scale. It is estimated that another 15-30 years of further work in the area will need to be done before a full scale implementation of Shor's algorithm can be achieved [9]. So, it would seem, the RSA encryption model is more or less safe, for the moment, from the threat posed by the advancement of quantum computing. However, Quantalab researchers hope that their efforts will inspire a quick response towards the creation of a quantum-secure key [9].

Elliptical Curve Cryptography

Before the discovery of its useful cryptographic properties, elliptic curves were studied for their aesthetics and importance in number theory. The appearance of elliptic curves in algebra is almost serendipitous: their origins are found in Diophantus' work on cubics, were next studied by Gauss and others as elliptic integrals, were instrumental in solving Fermat's Last Theorem, and now surprisingly, have altered the paradigm of public key cryptography [4].

In 1984, H.W. Lenstra's elliptic curve factoring algorithm was the first use of elliptic curves in cryptography, which, in turn, inspired a different use of elliptic curves in cryptography in a 1985 paper by V. Miller and N. Koblitz [5]. Their cryptographic protocol is based in the discrete logarithm problem of a subgroup of points defined on the elliptic curve in a finite field.

More formally, let E, the elliptic curve, be given by the Weierstrass equation

$$y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6$$

with $a_i \in \mathbb{F}_{ıı}$, a finite field of order q, q a prime or power of a prime. The subgroup used, G, is a prime order subgroup of the points of the $\mathbb{F}_{ıı}$ points of E. Using this, we can formulate the discrete logarithm problem, which asks: given $P, Q \in G$, find $x \mod n$ such that $Q = xP$.

It turns out this problem is incredibly difficult to solve as the best known algorithm to solve the discrete logarithm problem over elliptic curves is exponential [10]. To give a concrete analogy, a 228-bit RSA key requires less energy than it takes to boil less than a teaspoon of water; a 228-bit elliptic curve key requires enough energy to boil all of Earth's water [7]. The advantages here are obvious: smaller key size, lower bandwidth, faster implementation, and higher security. In 2005, the NSA recommended the move from primitive public key cryptography to elliptic curve cryptography [5]. Now, elliptic curves have become pervasive in security protocols involving encryption, digital signature verification, and pseudo-random generation.

Conclusion

Cryptography is but one practical application of abstract algebra, and this subject alone presents many applications of group theory. From Caesar's cipher, to RSA, to elliptic curves, group theory has yielded tremendous improvements in cryptography. With quantum computers looming in the near future, it's certain that group theory will be relied upon to generate safer and more efficient cryptographic protocols.

References

[1] Dan Boneh. Twenty years of attacks on the rsa cryptosystem. *Notices of the American Mathematical Society*, 46:203–213, 1999.

[2] Jean Gallier. Public key cryptography; the rsa system. 2016-03.

[3] Dan Goodin. Breaking 512-bit rsa with amazon ec2 is a cinch. so why all the weak keys? *Ars Technica*, Risk Assessment/Security & Hacktivism, 2015.

[4] Paul Hewitt. A brief history of elliptic curves. Accessed: 2016-04.

[5] Ann Hibner Koblitz, Neal Koblitz, and Alfred Menzes. Elliptic curve cryptography: The serpentine course of a paradigm shift. Accessed: 2016-04.

[6] RSA Laboratories. What key size should be used?

[7] Arjen K. Lenstra. Universal security: From bits and mips to pools, lakes, and beyond.

[8] George MacKiw. *Applications of Abstract Algebra: The RSA Public Key Cryptosystem.* pg.118-137.

[9] Amy Nordrum. Quantum computer comes closer to cracking rsa encryption. *IEEE Spectrum*, March:web, 2016.

[10] Joseph H. Silverman. An introduction to the theory of elliptic curves. Accessed: 2016-04.

[11] Sarah Simpson. Cryptography in everyday life. Accessed: 2016-04.

[12] Nick Sullivan. A relatively easy to understand primer on elliptic curve cryptography.

[13] Qing Yang. Cryptography. CSCI466: Networks, Dec. 2015. Montana State University, Bozeman. Lecture.