Applications of Group Theory and Abstract Algebra to Cryptography

Joey Peeters
Chris Brantner
Math 425
April 30, 2023

Abstract

An overview of different methods in cryptography and how they relate to abstract algebra. We will investigate error-correction through use of Reed-Solomon codes, as well as cryptographic methods for secure information transfer. Security methods include Diffie-Hellman key exchange and RSA encryption. We then investigate the impact of Quantum Computers to the future of cryptography, looking at a possible method of encryption involving lattices.

Introduction

Many topics in computer science rely heavily on abstraction, generalizing complex concepts into more broad features. The use of abstraction is observable within most areas of computing; from abstraction of analogue signals into discrete, then applying a binary system to represent these signals, then using Boolean logic to perform arithmetic and logical operations which underlie all the architecture of modern software development. From machine learning to operating systems to web page development, abstraction is a powerful tool which computer science relies upon. Given the pervasiveness of abstraction in the field, it should come as no surprise that abstract algebra has a wide variety of applications. In this, we narrow our focus to applications in cryptography.

Generally, cryptography is about encoding messages. From Scientific American in 1866, it is "the art of reading and writing dispatches, messages, etc., in such a way that only those who possess the key can decipher them." (*Scientific American*). This is a rather broad definition, and has evolved over time to include encoding information for reasons other than secrecy, as was the

origin of cryptography. The broad patterns used to encode information will be examined through the lens of the mathematics which make them possible. In this, we will look at two general problems: data storage integrity and communications security.

Beginning with the problem of data integrity, the underlying problem is transfer of information. When information is transmitted across a channel there is always risk of the information becoming corrupted and damaged. The goal of error correction is twofold; recognize when an error has occurred, and correct the error. While the general problem is simple, the solutions we desire are those which minimize computational overhead. The simplest solution, of course, would be to simply transmit the information multiple times. Then, each could be checked against the others to correct any errors. However, this is extremely computationally inefficient, as we are increasing our information length by many-fold. It is this trade off between overhead and ability to correct errors which codes seek to find a balance in.

Code Definitions

According to Judy Walker, group theory and combinatorics are the traditional tools of math for studying codes, while more recently algebraic geometry has made a big impact on the field (Walker 3). Before delving into details about types of codes, a background on the jargon associated with them is important.

Formally, a code C over an alphabet A is simply a subset of $A^n := A \times \cdots \times A$(n copies). (Walker). Notice that a code here is defined as a set. Elements of this set are called codewords, usually denoted with a lowercase c, and a codeword's length is usually denoted by n. Often, the alphabet of interest is a finite field. A code C is said to be linear if $C$is a vector subspace of A^n. (Walker). The dimension of a code, k, is another useful parameter to talk about, and is an important variable when calculating metrics of a code. The dimension is the dimension of the vector space over the alphabet.

Metrics

When talking about codes generally, it is important to establish some metrics with which we can compare the efficacy of codes. The general goal of error correction is to find a balance between overhead (the length added to the original message) with utility (ability to find and correct errors). One important feature besides dimension is the minimum distance of a code, denoted d.

Minimum distance is defined as elements of the code which are not equal to each other. Abstractly, the goal is to have two "points" (codewords of the code) which are "far enough away" in a sense to be useful. If two codewords are "close", then their ability to correct errors is limited. This is how minimum distance comes to be useful as a metric. Distances here are not defined in terms

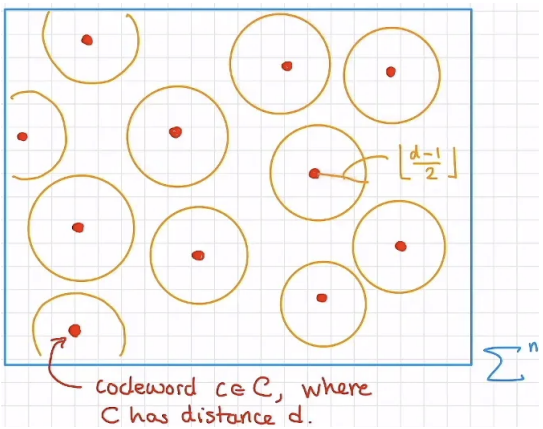

Figure 1: Representation of distance, courtesy of Rex Wang

of Euclidean distance, but rather as a more abstract metric space (Wootters). Again, the usefulness of abstraction in this field can not be overstated. In the above example, we can see an illustration of the Hamming distance of a code, the floor of $\frac{d-1}{2}$ as illustration of how points are "far enough away".

Another useful metric is the information rate, $R = \frac{k}{n}$ where k is the dimension and n is the code length (Walker) . Equivalently, this is $\frac{\log_{\text{alphabet}} |C|}{n}$(Wooters). More metrics are defined for codes, and are useful in establishing bounds on performance. This allows for quantifying how well a code can possibly work for recovery. Beyond the need for metrics, we also need metrics for efficiency in the encoding and decoding algorithms of a code. Complexity analysis is an entire branch of computer science on its own, and will not be covered here.

Reed-Solomon Codes

Armed with the lexical necessities to talk of linear codes, we can now delve into the world of Reed-Solomon codes. Reed-Solomon codes rely on a useful property of polynomials that low degree polynomials do not have too many roots. That is, a nonzero univariate polynomial of at most degree a has at most a roots (Wooters). This ensures that the distance of our code will be acceptable. The points of interest will be compiled into what's known as an evaluation set, consisting of points with which we can recover local information about our original message.

Reed-Solomon codes are defined over a finite field of order q, denoted $\mathbb{F}_q$. Then, the Reed-Solomon code with dimension k defined over $\mathbb{F}_q$ with $q \leq n \leq k$ and evaluation points $\bar{\alpha}(\alpha_1, \alpha_2 \ldots \alpha_n)$ is defined as $RS_q(\bar{\alpha}, n, k) := (f(\alpha_1), f(\alpha_2), \ldots, f(\alpha_n)) | f \in \mathbb{F}_q[X], deg(f) \leq k-1$ (Walker). As inputs, the code takes in the evaluation set, code length, and dimension of the code. The points in the evaluation set are then mapped to polynomials in $\mathbb{F}_q[X]$ such that the degree of each polynomial does not exceed one less than the dimension

of the code. This describes a natural encoding map, upon which algorithms can be made, that takes points to polynomials. While the theory behind these codes was proposed in the 1960s, there were no algorithms to implement these codes until many years later (Wootters). This is a good example of how often a pure mathematical concept can later translate to useful applications, even when it's unknown at the time of the concepts inception how this could work. Further, it's demonstrative of the powers of abstraction, and how abstract algebra has been useful to cryptography.

Today, Reed-Solomon codes are in use in many data storage methods. Their low overhead combined with their quick encoding algorithms make for an effective tool to recover errors generated in data transmission across channels. Abstraction in cryptography makes heavy use of concepts in abstract algebra, such as finite fields, mappings between sets, and linearity.

Information Transfer Security

Moving from the sphere of data integrity, another main branch (perhaps more well known of the two) of cryptography deals with information security across channels. The need to securely communicate was the original motivation behind cryptography and predates the invention of the computer by a fair amount. The field began as ciphers, or encryptions of messages, to obfuscate message contents from prying eyes. From the Caesar cipher to the Enigma Machine, mathematics is a tool which can be used to better understand how to make useful encryptions (Singh).

One such type of encryption involves the use of what are known as "public keys". This is a model in which two end users want to transfer information across a channel with potential adversaries. To do so securely, users must pass only encrypted information in the channel, so as to prevent adversarial agents from intercepting sensitive information. Here, a "public key" is employed to aid in this. We will discuss the Diffie-Hellman method, in which the public key model is employed (Koblitz et al.).

Diffie-Hellman Key Exchange

In this protocol, two end users have their own private "key", a string of binary code. The goal is to verify that each user is legitimate without giving away their own private key. This is achieved using public key exchange, in which each user combines their private key with a public one, and the other user is able to verify, mathematically, that the other is legitimate. The original implementation of this is one which uses the multiplicative group of integers modulo p, where p is prime. In this, p is public, as well as a base g. These are the public codes, available to anybody.

As an example, say two users have private keys a and b. They want to verify that the other's is legitimate. The way they do this, is user A would combine

$g^a \pmod p$ and send that to user B. Similarly, B would combine $g^b \pmod p$ and send that to user A. Then, user A would take the g^b they received, and take it to their secret code a, resulting in $(g^b)^a$, and B would take the g^a they received as $(g^a)^b$. Then, they would verify that these two values match, as $(g^a \bmod p)^b = (g^b \bmod p)^a$. The result is they have a shared encryption key without ever giving away their private keys. The strength of this is that it is incredibly computationally difficult (near impossible) to figure out what a and b are given only g, g^a, g^b, and p (Meijer).

Generally, this relies on group theory, specifically finite cyclic groups. If G is a finite cyclic group of order n, then two users agree upon n, a natural number, and g, a generating element of G. One user picks their secret key, a, between 1 and n. The other user picks their secret key b, between 1 and n. Then, each sends g^a, g^b to the other. Then, they commute $(g^b)^a$, $(g^a)^b$ respectively. Now, they both have a common key without giving away publicly their secret key. This can be applied over any finite cyclic group with varying degrees of security. These keys are very long in length to make decryption by guessing highly unlikely and computationally time consuming.

Problems Arising from Quantum Computing

Going through all these forms of cryptography are great for sending messages along that you do not want to be encoded by someone these messages aren't meant for. A problem posed to cryptography that drives the field to keep pushing are quantum computers. These quantum computers are potentially able to solve classic cryptographic methods much more efficiently than a classic computer because of Shor's algorithm. Shor's algorithm is based on using quantum Fourier transformations to factor down a number more efficiently (Politi). These computers are able to factor large numbers and find discrete logarithms at a much quicker pace because of this algorithm.

Intro to Quantum Computers

Quantum computers are a type of computer that use quantum mechanical phenomena, such as superposition and entanglement, to perform operations on data. Unlike classical computers, which use bits that can be either 0 or 1, quantum computers use quantum bits, or qubits, which can be in a superposition of both 0 and 1 at the same time. For a clear example of this, a classical computer can represent $0 - 255$ by eight bits, whereas a quantum computer is able to do the same 256 with eight qubits (J. Cheng). This allows quantum computers to perform certain calculations much faster than classical computers, particularly in areas such as cryptography, optimization, and simulation. However, building and operating quantum computers is challenging due to the fragility of qubits and the need for precise control and isolation from the environment.

Summarizing quantum computers, we see that this new advance in technology is able to solve some of our pre-existing encoding techniques and offers a more efficient route than classical computers. The only problem in the way of these operating systems is the fact that they are not available commercially. These computers still are in the process of needing more stable qubits, better control systems and improved error correction. (RF Wireless World)

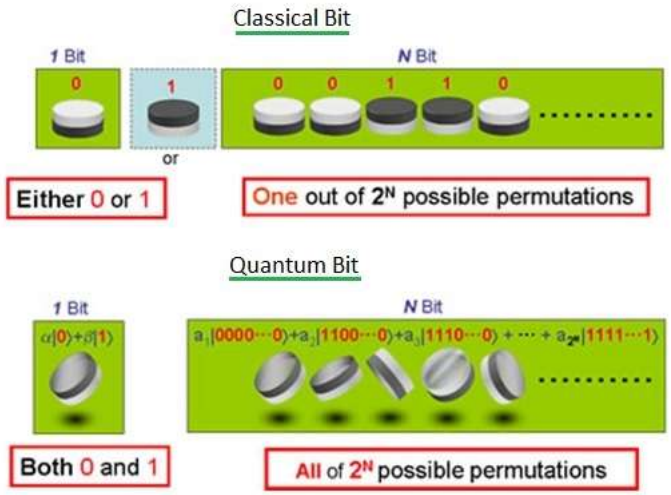

Figure 2: Classical bit vs. Quantum bit courtesy of *www.rfwireless-world.com*

Lattice-Based Cryptographic Problems

After reviewing the risk that quantum computers pose to breaking many pre-existing cryptographic protocols, we're now able to take a look into lattice-based cryptography. The alternative mathematical problems provided from lattice-based cryptography include the Shortest Vector Problem (SVP) and the Learning with Errors (LWE) problem. These two problems are what makes this type of cryptography one of the strongest against quantum computers. SVP is one of the most well-known studied lattice problems in cryptography. The main point of SVP is to find the shortest nonzero vector in a given lattice where a classical computer would have issues solving for how large the instance is (Daniele). This is known as a "hard" problem in the field since you would need another piece of information (what is reasonably orthogonal to a vector) and using a linear combination in order to find this shortest vector.

Lattice-based cryptography uses the hardness of SVP to design cryptographic schemes that are believed to be secure against classical and quantum computers. For example, the LWE problem is a generalization of SVP, where the vector is not exactly in the lattice but perturbed by some noise.In the LWE problem, a set of linear equations is formed where the variables are integers modulo some large number q, and the coefficients are chosen from a small set of random values (Regev). However, a small amount of "error" is added to the equations by randomly perturbing the coefficients. The task is to solve for the

variables given the equations with errors.

The LWE problem is important in cryptography because it provides a way to create secure and efficient public-key encryption schemes, key exchange protocols, and digital signature schemes. These schemes have several advantages over traditional cryptography methods like RSA, including resistance to quantum attacks and smaller key sizes, which makes them more efficient to use in practice.

Conclusion

Cryptography is a fascinating field of study that has played a critical role in securing information transmission over the internet, as well as preserving data in storage. Reed-Solomon codes are a type of error-correcting code that is able to detect and correct errors in data transmission. A large branch of cryptography deals with information transfer over unsecured channels, which Diffie-Hellman key exchange helps to mediate. Abstract algebra plays a significant role in cryptography, particularly in the development of lattice-based cryptography as of late. These studies are being pushed due to the risk quantum computers now impose on classical cryptography. While cryptographic problems are challenging, abstract algebra and other branches of mathematics offer insight as to how to tackle these problems, and re-framing a question using abstraction is what has brought forth much of the technology we use today.

Works Cited

Daniele Micciancio. (n.d.). Shortest Vector Problem (SVP). Retrieved April 30, 2023, from https://cseweb.ucsd.edu/ daniele/LatticeLinks/SVP.html

D'Anvers, Jan-Pieter. "CO6GC: Introduction to Lattice-Based Cryptography (Part 2): Lwe Encryption." COSIC, 8 June 2022

Politi, A., Matthews, J. C. F., and O'Brien, J. L. (2009). Shor's quantum factoring algorithm on a photonic chip. Science, 325(5945), 1221. https://doi.org/10.1126/science.1173731

J. Cheng, Y. Xie, Y. Shi, X. Liu, Z. Wang, "Secure decentralized access control for blockchain-based medical data sharing with privacy protection," Expert Systems with Applications, vol. 190, p. 114984, 2022. doi: 10.1016/j.eswa.2022.114984.

Regev, Oded. "Learning with errors, twenty years later." Journal of the ACM (JACM) 65, no. 6 (2018): 38. https://cims.nyu.edu/ regev/papers/lwesurvey.pdf

RF Wireless World. (n.d.). Difference between bit and qubit. Retrieved April 30, 2023, from https://www.rfwireless-world.com/Terminology/Difference-between-Bit-and-Qubit.html

Wootters, Mary, director. Algebraic Coding Theory, 30 Mar. 2021, https://youtu.be/yQkEnd e2lNg. Accessed 20 Apr. 2023.

Singh, Simon. "The History of Cryptography: How the History of Code-breaking Can Be Used in the Mathemathics Classroom with Resources on a New CD-ROM." Mathematics in School, vol. 32, no. 1, 2003, pp. 2–6. JSTOR, http://www.jstor.org/stable/30212224. Accessed 1 May 2023.

Meijer, A. R. "Groups, Factoring, and Cryptography." Mathematics Magazine, vol. 69, no. 2, 1996, pp. 103–09. JSTOR, http://www.jstor.org/stable/2690663. Accessed 1 May 2023.

Walker, Judy. "Codes and Curves." The Student Mathematical Library, 2000, https://doi.org/10.1090/stml/007. Accessed 20 Apr. 2023.

"CRYPTOGRAPHY." Scientific American, vol. 14, no. 19, 1866, pp. 295–96. JSTOR, http://www.jstor.org/stable/24974248. Accessed 30 Apr. 2023.

Wang, Rex. "Distance." JuliaLang.org, 26 Nov. 2022, https://forem.julialang.org/rexwzh/coder-decoder-for-qr-codes-ospp22-work-product-3gn4. Accessed 20 Apr. 2023.

Koblitz, Neal, and Alfred J. Menezes. "A Survey of Public-Key Cryptosystems." SIAM Review, vol. 46, no. 4, 2004, pp. 599–634. JSTOR, http://www.jstor.org/stable/20453567.