An Exploration of Methods of Cracking Encryption Schemes

Alexander Agruso
Texas State University

Brandon Howell
Texas State University

Abstract

Blah blah blah abstract here

Introduction

In the age of digital communication, it is more important than ever that we safeguard our information online. With so much of our sensitive information being sent over the internet, we must find ways to ensure that unauthorized parties aren't able to view this information. The most common way of achieving this goal is through an encryption scheme.

One of the earliest known encryption schemes date back to the ancient Greeks, where Polybius invented the "Polybius Square", a primitive encryption method that encoded letters in a five by five square. Another well known encryption method from antiquity is the Julius Cipher, which shifts the letters in the alphabet so that the corresponding letters in a given sentence are scrambled in such a way as to make the text unreadable unless deciphered.

While these schemes may have worked for people in the past, they are rendered essentially useless by the ubiquity of powerful computers in our modern age. As a result, much more sophisticated encryption schemes have been developed that can resist computational efforts to crack them. In this paper, we will focus on two widely used public-key encryption schemes: RSA and Elliptic Curve Encryption. These methods use advanced concepts in math, specifically number theory, to ensure that even the most powerful computers are unable to decipher a message unless they possess the private key, with which deciphering a message becomes a significantly easier task.

Both of these schemes, and all public-key encryption schemes in general, rely on intractable problems, which are problems whose solutions are difficult, if not virtually impossible to find, but are nonetheless easily verifiable once found. So, if one wants to crack these encryption schemes, one must find solutions to these intractable problems. This paper explores various methods of finding solutions to the intractable problems at the core of RSA and Elliptic Curve Encryption, as well as possible improvements that can be made upon existing methods.

RSA Encryption

The RSA encryption scheme, invented by Ron Rivest, Adi Shamir, and Leonard Adleman in 1977, is a widely used public-key encryption system that relies on the difficulty of factoring certain large integers. We provide a brief explanation of how public-private key pairs are generated, and how they are used to encrypt and decrypt messages.

The first step in creating a public-private key pair is to randomly choose two large prime numbers p and q, then calculate their product $n = pq$. Next, the value of $\lambda(n)$ is calculated, where λ is Carmichael's totient function. We define $\lambda(n)$ as the smallest integer m where $a^m \equiv 1 \pmod{n}$ for all integers a that are relatively prime to n. [1] It should be noted

that $\lambda(n)$ is very difficult to calculate if only given n. However, if one has the prime factorization of n, in this case pq, the calculation becomes considerably easier, with it reducing to calculating the least common multiple of $p-1$ and $q-1$. We then choose an integer e such that $2 < e < \lambda(n)$ and e and $\lambda(n)$ are relatively prime. Finally, we find the multiplicative inverse d of $e \pmod{\lambda(n)}$. The public key consists of the modulus n and the exponent e, while the private key consists of the exponent d.

Having calculated our public-private key pair, we are now ready to start encrypting messages. Let M be the message we wish to encrypt. Using n and e, we calculate the ciphertext c by evaluating the following:

$$c \equiv M^e \pmod{n}.$$

The encrypted message c can then be sent to the recipient.

When the message is received, it can be decrypted using our private exponent d. Since d and e are multiplicative inverses, we have that $de \equiv 1 \pmod{n}$, thus we can obtain M as follows:

$$c^d \equiv (M^e)^d \equiv M^{ed} \equiv M^1 \equiv M \pmod{n},$$

thus by calculating c^d, we obtain the original message M, and since d is kept secret, only the recipient can decrypt the ciphertext.

Methods of Cracking RSA

To crack RSA encryption, we must find a way to derive the private key d from the public key values n and e. The most straightforward way of doing this is to find the prime factors p and q of n, which, as previously noted, can be used to easily calculate $\lambda(n)$, and thus calculate d.

Elliptic Curve Encryption

Elliptic curve cryptography, henceforth referred to as ECC, describes several cryptographic schemes that rely on the intractability of the elliptic curve discrete logarithm problem (ECDLP). Before we can discuss how public-private key pairs are generated and how they are used to encrypt and decrypt messages, we will briefly introduce the mathematics behind elliptic curves and the ECDLP.

An elliptic curve is a cubic polynomial in two variables that takes the form of

$$y^2 = x^3 + ax^2 + bx + c,$$

combined with a "point at infinity", which we will denote as O. For the purposes of ECC, we take this curve over a finite field $\mathbb{F}_p$, where p is a carefully chosen prime number. In abstract algebra, a field simply represents a set where the usual notions of addition and multiplication hold true, and in our case, this field has finitely many elements, namely p elements.

Given the set of points on this curve over $\mathbb{F}_p$, along with O, we can endow it with a group structure by equipping it with a special "addition" operation. A group is a mathematical structure that describes the symmetries of a certain object, in this case, the points on our curve. Additionally, it allows us to define the "discrete logarithm" of points on the curve. Given two elements a and b in our group, we define the discrete logarithm $\log_b(a)$ as the integer k such that $b^k = a$. In this case, a and b are points on our curve, Despite the notations being similar, the discrete logarithm and regular logarithms should not be confused with one another.

Future Threats to RSA and ECC

Blah blah blah quantum

References

[1] Wolfram Alpha. Carmichael function. https://mathworld.wolfram.com/CarmichaelFunction.html.