Self Reading Project

Cryptography Algorithms and Principles

Abhilasha Sharma Suman

23B1011

Contents

1 Algorithms

1.1 Substitution Cipher

Basic bread and butter cipher. All characters are mapped to another character by shifting the whole set of characters by n positions. Caesar cipher, named after Julius Caesar, is a specific case where n=3. Happen to be symmetric, that is, the same key is required to encrypt and decrypt the cipher text.

Easy to break since only 26 different settings to try which means it is vulnerable to brute force attacks. You just try every possible shift and keep doing it until you get a meaningful message.

- Implemented Caesar and Affine cipher for example (both encryption and decryption done and to be frank there is no difference between them except the shift amount).

1.2 Vigenere Cipher

This, in my humble opinion, is the most genius cryptography technique of them all. It is essentially a combination of multiple Caesar ciphers but every alphabet comes from a different shift series. We have a text to encrypt and a key string. Say that the text to encrypt is 'mynameisketchup' and the key is 'social'. So we extend the key to 'socialsocialsoc' so that it matches the length of the input string to encrypt. Each alphabet of the modified key represents a different shift series, so A represent 0 shift, B represents 1 shift and so on.

Then we see the character corresponding to each character in the string to encrypt and replace it by it's corresponding letter in the shift series corresponding to the character in the extended key. The enigma machine from World War 2 used 3 rotating Vigenere cipher on 5 different rotors with different starting points.

Cracking vigenere cipher through brute force attack is something I wouldn't advise. There are a few methods to break vigenere cipher but most of them rely on frequency analysis of letters in the language and then matching them with the frequency of characters in the message. This doesn't work very well for shorter texts.

1.3 RSA

RSA is a public key cryptography method. Public key cryptography methods. In these methods there are two parts to the key, one of which is public meaning anyone who has the key can encrypt a message and the other one is private. Both the receiver and the sender have their pair of keys that match (not identical, match in the sense that one's public key encryption can be decrypted by the other's private key).

The RSA method relies on the difficulty in factorizing a large integer. In this method, the public key consists of two numbers where one number is a multiplication of two large prime numbers and private key is also derived from the same two prime numbers. So if somebody can factorize the public key, the private key is compromised, therefore the strength of encryption is based on the key size. If we were to double or triple it, this security would increase exponentially. RSA keys are typically 1024 or 2048 bits long (128 or 256 bytes big) factorising whom remains an infeasible task for now. The process of RSA is has 2 main parts: key generation, encryption/decryption.

1. Key Generation
   Choose two large prime numbers p and q. The first part of the key, the public key n, is given as the product of p and q and the other part (the private key) is generated using the totient function of n. Then we select a small integral exponent e that is not a factor of the totient function of n $\phi(n)$. The private key $p_k = \frac{(k*\phi(n)+1)}{e}$ where k is some integer.
2. Encryption and Decryption
   Now, to encrypt the text we will just make a concatenated string from their positions in the language alphabet that is a$\longrightarrow$1, b$\longrightarrow$2 and so on. Let this number obtained be z. The encrypted message is public key $n|(z^d)$.
   To encrypt a message m, we need to convert it to an integer between 0 and n-1. This can be done using a reversible encoding scheme, such as ASCII or UTF-8. Once we have the integer representation of the message, we compute the ciphertext c as c =me (mod n). This can be done efficiently using modular exponentiation algorithms, such as binary exponentiation.
   Here is the code implementation of RSA encryption and decryption

```
   import random
import math

prime = set()

public_key = None
private_key = None
n = None


def addPrimes():
 sieve = [True] * 250
 sieve[0] = False
 sieve[1] = False
 for i in range(2, 250):
  for j in range(i * 2, 250, i):
   sieve[j] = False

 for i in range(len(sieve)):
  if sieve[i]:
   prime.add(i)


def pickAPrime():
 global prime
 k = random.randint(0, len(prime) - 1)
 it = iter(prime)
 for _ in range(k):
  next(it)

 ret = next(it)
 prime.remove(ret)
```

```
 return ret


def setkeys():
 global public_key, private_key, n
 prime1 = pickAPrime()
 prime2 = pickAPrime()

 n = prime1 * prime2
 fi = (prime1 - 1) * (prime2 - 1)

 e = 2
 while True:
  if math.gcd(e, fi) == 1:
   break
  e += 1

 public_key = e

 d = 2
 while True:
  if (d * e) % fi == 1:
   break
  d += 1

 private_key = d

def encrypt(message):
 global public_key, n
 e = public_key
 encrypted_text = 1
 while e > 0:
  encrypted_text *= message
  encrypted_text %= n
  e -= 1
 return encrypted_text

def decrypt(encrypted_text):
 global private_key, n
 d = private_key
 decrypted = 1
 while d > 0:
  decrypted *= encrypted_text
  decrypted %= n
  d -= 1
 return decrypted
```

```
def code(message):
 encoded = []
 # Calling the encrypting function in encoding function
 for letter in message:
  encoded.append(encrypt(ord(letter)))
 return encoded


def decode(encoded):
 s = ''
 # Calling the decrypting function decoding function
 for num in encoded:
  s += chr(decrypt(num))
 return s


if __name__ == '__main__':
 addPrimes()
 setkeys()
 message = "SoS was Fun"
 coded = code(message)

 print("Original message",message)
 print("Encoded Message",''.join(str(p) for p in coded))
 print("Message decoded!",''.join(str(p) for p in decode(coded)))
```

2 Public Key Cryptographic Systems

Public key cryptography, also called asymmetric cryptography, is a cryptographic system that relies on a pair of keys: a public key, which is known to everyone, and a private key, which is known only by select parties. This cryptographic system is the backbone of modern digital communication, enabling secure data transmission, digital signatures, and key exchange mechanisms.

In this report, we will explore the principles behind public key cryptography, the mathematical foundations, popular algorithms, and real-world applications.

2.1 Overview of Cryptographic Systems

- Symmetric Key Cryptography

  In symmetric key cryptography, the same key is used for both encryption and decryption. The main challenge with symmetric key systems is the secure exchange of the key between the communicating parties. Examples of symmetric key algorithms include the Data Encryption Standard (DES) and the Advanced Encryption Standard (AES).

- Public Key Cryptography

  Public key cryptography solves the key exchange problem by using two keys: a public key for encryption and a private key for decryption. Only the private key holder can decrypt the messages

encrypted with the public key. This system enables secure communication without the need for the sender and receiver to share a secret key beforehand.

2.2 Mathematical Foundations of Public Key Cryptography

Public key cryptography relies on mathematical functions that are easy to compute in one direction but hard to reverse without special knowledge (such as a private key). These functions are often based on number theory and include:

1. Modular Arithmetic

   Modular arithmetic is a system of arithmetic for integers where numbers "wrap around" upon reaching a certain value, called the modulus. Given a modulus n, for any integer a, we have:

   $$a \bmod n = r \quad \text{where} \quad 0 \leq r < n$$

   Modular arithmetic is the basis for many cryptographic algorithms, such as RSA.

2. Prime Factorization

   The security of public key systems like RSA depends on the difficulty of prime factorization. Given a large number N, it is computationally hard to factor N into its prime factors p and q. This property is used in the key generation process for RSA.

3. Discrete Logarithm Problem

   The discrete logarithm problem is another mathematical foundation used in public key cryptography. In systems like Diffie-Hellman key exchange and ElGamal encryption, it is easy to compute powers of a number modulo n, but difficult to compute the inverse operation, i.e., finding x such that:

   $$g^x \bmod n = y$$

   This problem is assumed to be computationally infeasible, providing the basis for security.

2.3 RSA Algorithm

This has already been discussed but I want this report to be longer, so here we go again. RSA (Rivest–Shamir–Adleman) is one of the most widely used public key cryptosystems. It enables secure data transmission through encryption and digital signatures. The security of RSA is based on the difficulty of factoring large integers.

1. Key Generation

   The key generation process in RSA involves the following steps:

   (a) Choose two large prime numbers p and q.

   (b) Compute $N = p \times q$ and $\phi(N) = (p-1) \times (q-1)$, where $\phi(N)$ is Euler's Totient function.

   (c) Choose an integer e such that $1 < e < \phi(N)$ and $\gcd(e, \phi(N)) = 1$ (i.e., e is coprime with $\phi(N)$).

   (d) Compute the private key d such that $d \times e \equiv 1 \pmod{\phi(N)}$. This means d is the modular multiplicative inverse of e modulo $\phi(N)$.

The public key is (e, N) and the private key is (d, N). The values of p and q must be kept secret.

2. Encryption

   To encrypt a message m (where m is a number such that $0 \leq m < N$), the sender uses the recipient's public key (e, N) and computes the ciphertext c as:

$$c = m^e \mod N$$

3. Decryption

   To decrypt the ciphertext c, the recipient uses their private key (d, N) and computes the original message m as:

$$m = c^d \mod N$$

   Since $d \times e \equiv 1 \pmod{\phi(N)}$, the decryption process correctly retrieves m from c.

2.3.1 Security of RSA

The security of RSA relies on the computational difficulty of factoring the modulus N, which is the product of two large primes p and q. Without knowledge of p and q, it is infeasible to compute $\phi(N)$ and, therefore, the private key d.

2.4 Diffie-Hellman Key Exchange

The Diffie-Hellman key exchange protocol is a method for two parties to securely share a secret key over an insecure channel. Unlike RSA, Diffie-Hellman is used for key exchange, not for encryption or digital signatures.

2.4.1 Protocol Overview

The Diffie-Hellman protocol works as follows:

(a) Both parties agree on a large prime number p and a generator g (a primitive root modulo p).

(b) Alice chooses a private key a and sends $A = g^a \mod p$ to Bob.

(c) Bob chooses a private key b and sends $B = g^b \mod p$ to Alice.

(d) Alice computes the shared secret $s = B^a \mod p$.

(e) Bob computes the shared secret $s = A^b \mod p$.

Since $s = g^{ab} \mod p$, both Alice and Bob now share the same secret value s, which can be used as a key for symmetric encryption.

2.4.2 Security of Diffie-Hellman

The security of the Diffie-Hellman protocol is based on the hardness of the *Discrete Logarithm Problem*. Given $g^a \mod p$, it is computationally difficult to determine a. Thus, even if an attacker intercepts A and B, they cannot easily compute the shared secret s.

2.5 Elliptic Curve Cryptography (ECC)

Elliptic Curve Cryptography (ECC) is another form of public key cryptography based on the algebraic structure of elliptic curves over finite fields. ECC offers equivalent security to RSA with much smaller key sizes, making it highly efficient for modern applications.

2.5.1 Elliptic Curves

An elliptic curve is defined by an equation of the form:

$$y^2 = x^3 + ax + b$$

where a and b are constants. The points on the curve, along with a special point at infinity, form an abelian group under a well-defined addition operation.

2.5.2 Key Generation in ECC

Key generation in ECC involves selecting a base point G on the elliptic curve, which is publicly known. The private key is a random integer d, and the public key is the point $Q = dG$, where dG denotes scalar multiplication of G.

2.5.3 Encryption and Decryption in ECC

The encryption and decryption process in elliptic curve cryptography is similar to that in RSA. A message is encrypted using the recipient's public key and can be decrypted using the corresponding private key.

2.5.4 Security of ECC

ECC is based on the difficulty of the *Elliptic Curve Discrete Logarithm Problem* (ECDLP), which is believed to be much harder than the integer factorization problem on which RSA is based. This allows ECC to achieve high security with shorter keys.

2.6 Applications of Public Key Cryptography

Public key cryptography has numerous applications in modern computing including but not limited to digital signatures (for the verification of the source of a file), SSL and TLS (Secure Socket Layer and Trasnport Layer Security), Cryptocurrency and message encryption (eg: while texting and mailing, the messages are encrypted to maintain privacy)

3 Zero Knowledge Proof

Zero-knowledge proofs are those that prove a statement without yielding any additional information, for example:

(a) In the example of colour blind friend and two balls, we have May and June out of which May is color blind.June has two balls and needs to prove that the balls are two different colors. The latter switches the balls randomly behind her and shows them to the former who must tell her if the balls have been switched. The possibility of June answering correctly is 50%. When this experiment is carried out over and over again, the possibility of June being able to answer correctly with false information is pretty low. So we verify June's (**prover**) claim without the colors ever being revealed to May(**verifier**).

(b) Imagine a cave with a single entrance but two pathways (path A and B) that connect at a common door locked by a passphrase. Alice wants to prove to Bob she knows the code to the door but without revealing the code to Bob. To do this, Bob stands outside of the cave and Alice walks inside the cave taking one of the two paths (without Bob knowing which path was taken). Bob then asks Alice to take one of the two paths back to the entrance of the cave (chosen at random). If Alice originally chose to take path A to the door, but then Bob asks her to take path B back, the only way to complete the puzzle is for Alice to have knowledge of the code for the locked door. This process can be repeated multiple times to prove Alice has knowledge of the door's code and did not happen to choose the right path to take initially with a high degree of probability. After multiple repetitions of this, Bob can be sufficiently confident that Alice knows the code without ever knowing the code himself.

These examples demonstrate the properties of zero knowledge of the fact to the verifier, completeness of the proof and the soundness of the proof (that is if the prover isn't truthful they won't be able to convince the verifier). So what is the significance of ZKPs in cryptography? The use of ZKP protocols allows us to cryptographically prove to someone that they know about a piece of information without actually revealing the information thus maintaining privacy. It's use is on the rise in areas like public blockchain networks. The two types of zero knowledge proofs are discussed below.

3.1 Interactive Zero Knowledge Proofs

Interactive ZKP requires the verifier to constantly ask a series of questions about the "knowledge" the prover possess. Like in the first example, the May asked June about the switch status of the balls. A real life example of this was seen in Nuclear Disarmament drive in 2016, when Princeton University and one of it's departments demonstrated a technique which might allow inspectors to verify whether an object is nuclear or not without revealing sensitive information that might compromise the details of the internal workings or recording and sharing any information.

Some problems with this method of cryptography are that the transfer of the encrypted information becomes problematic because you essentially need to provide the proof to every single verifier which means the process has to repeated over and over too many times and also, both the prover and verifier need to be available at the same time for the process to be able to work.

3.2 Non Interactive Zero Knowledge Proofs

NIZKPs have emerged as the main concept in the world of cryptography and engineering mathematics. These proofs enable one party to prove the validity of a statement to another party without revealing any additional information beyond the validity of the statement itself.Fiat and Shamir invented Fiat-Shamir heuristic and changed interactive zero-knowledge proof to non-interactive zero-knowledge proof. Fiat–Shamir heuristic is technique for taking an interactive proof of knowledge and creating digital signature based on it. This way ' witness' or fact can be verified publicly without prover being online all the time.

NIZKs do not require interaction between the prover and the verifier, making them highly efficient for various applications in secure communications, blockchain, and privacy-preserving technologies. The way they work is that a trusted third party generates and distributes a common reference string (CRS) to both the prover and the verifier (**Setup Phase**). Then the prover uses the CRS to create a proof that demonstrates the truth of the statement without revealing any additional information which is sent to the verifier (Proof generation; The generation process can be computationally intensive, but it eliminates the need for interactive communication). Finally in the verification stage, the verifier uses the CRS and the proof received from the prover to check the validity of the statement. If the proof is valid, the verifier is convinced that the statement is true, without learning anything else.

This method is transferable easily, it is efficient, maintains privacy and is easily scalable. It's applications include digital signatures by allowing the signer to prove the authenticity of their signature without exposing the signed message. This is crucial in scenarios where the content of the message must remain confidential.