GUJARAT TECHNOLOGICAL UNIVERSITY

Chandkheda, Ahmedabad

Affiliated

VIHSWAKARMA GOVERNMENT ENGINEERING COLLEGE

A Report on:

DIGITAL CRYPTEX

Under Subject of:

DESIGN ENGINEERING

BE SEM 6: DE 2B

(ELECTRONICS AND COMMUNICATION)

SUBMITTED BY:

1) AARUSHI GANGWAR	150170111001
2) ALOKENDU MAZUMDER	150170111003
3) SARTHAK JAIN	150170111034

PROF. JAGRUTI NAYAK

(FACULTY GUIDE)

ACADEMIC YEAR 2018-2019

Contents

Chapter 1: Cryptography - History, Importance and Origins

1.1) Abstract and Goal of Project

Nowadays security breaches are common I almost every sector. The encryption techniques that are being used are getting old and outdated. Today hackers are able to replicate almost every decrypter keys.

The beauty of cryptography is, that anyone can design a rough algorithm to encrypt data in suitable form without using any predefined set of rules.

With an aim to develop a whole new algorithm, that to without the rigorous use of decryption key which will ultimately minimize the probability of losing key to the third party's hand, a group of three students including me are currently working on that which used the foundation laws of digital electronics.

1.2) Concepts of Cryptography

Cryptography or **cryptology** is the practice and study of techniques for secure communication in the presence of third parties called adversaries. More generally, cryptography is about constructing and analysing protocols that prevent third parties or the public from reading private messages, various aspects in information security such as data confidentiality, data integrity, authentication, and non-repudiation are central to modern cryptography. Modern cryptography exists at the intersection of the disciplines of mathematics, computer science, electrical engineering, communication science, and physics. Applications of cryptography include electronic commerce, chip-based payment cards, digital currencies, computer passwords, and military communications.

Modern cryptography is heavily based on mathematical theory and computer science practice; cryptographic algorithms are designed around computational hardness assumptions, making such algorithms hard to break in practice by any adversary. It is theoretically possible to break such a system, but it is infeasible to do so by any known practical means. These schemes are therefore termed computationally secure; theoretical advances, e.g., improvements in integer factorization algorithms, and faster computing technology require these solutions to be continually adapted. There exist information-theoretically secure schemes that probably cannot be broken even with unlimited computing power—an example is the one-time pad—but these schemes are more difficult to implement than the best theoretically breakable but computationally secure mechanisms.

1.3) Classical Cryptography

The main classical cipher types are transposition ciphers, which rearrange the order of letters in a message (e.g., 'hello world' becomes 'ehlol owrdl' in a trivially simple rearrangement scheme), and substitution ciphers, which systematically replace letters or groups of letters with other letters or groups of letters (e.g., 'fly at once' becomes 'gmz bu podf' by replacing each letter with the one following it in the Latin alphabet). Simple versions of either have never offered much confidentiality from enterprising opponents. An early substitution cipher was the Caesar cipher, in which each letter in the plaintext was replaced by a letter some fixed number of positions further down the alphabet. Suetoniusreports that Julius Caesar used it with a shift of three to communicate with his generals. Atbash is an example of an early Hebrew cipher. The earliest known use of cryptography is some carved ciphertext on stone in Egypt (ca 1900 BCE), but this may have been done for the amusement of literate observers rather than as a way of concealing information.

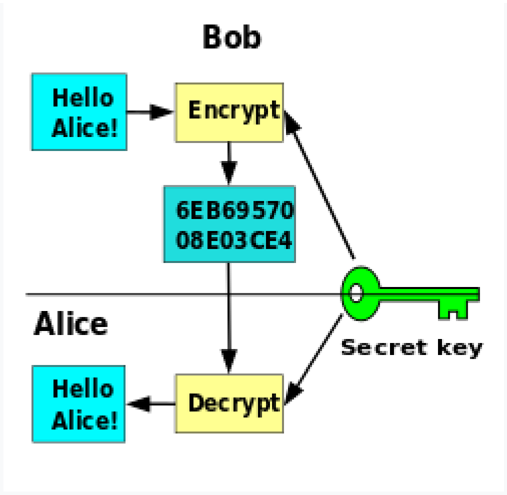
The Greeks of Classical times are said to have known of ciphers (e.g., the scytale transposition cipher claimed to have been used by the Spartan military).[17] Steganography (i.e., hiding even the existence of a message so as to keep it confidential) was also first developed in ancient times. An early example, from Herodotus, was a message tattooed on a slave's shaved head and concealed under the regrown hair.[11] More modern examples of steganography include the use of invisible ink, microdots, and digital watermarks to conceal information.

In India, the 2000-year-old Kamasutra of Vātsyāyana speaks of two different kinds of ciphers called Kautiliyam and Mulavediya. In the Kautiliyam, the cipher letter substitutions are based on phonetic relations, such as vowels becoming consonants. In the Mulavediya, the cipher alphabet consists of pairing letters and using the reciprocal ones.[11]

In Sassanid Persia, there were two secret scripts, according to the Muslim author Ibn al-Nadim: the *šāh-dabīrīya* (literally "King's script") which was used for official correspondence, and the *rāz-saharīya* which was used to communicate secret messages with other countries.

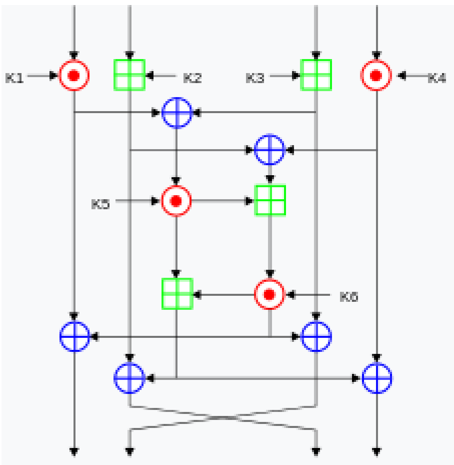
1.4) Modern Cryptography

1.4.1) Symmetric-Key Cryptography



Symmetric-key cryptography, where a single key is used for encryption and decryption

Symmetric-key cryptography refers to encryption methods in which both the sender and receiver share the same key (or, less commonly, in which their keys are different, but related in an easily computable way). This was the only kind of encryption publicly known until June 1976.



One round (out of 8.5) of the IDEAcipher, used in some versions of PGPfor high-speedencryption of, for instance, e-mail

Symmetric key ciphers are implemented as either block ciphers or stream ciphers. A block cipher enciphers input in blocks of plaintext as opposed to individual characters, the input form used by a stream cipher.

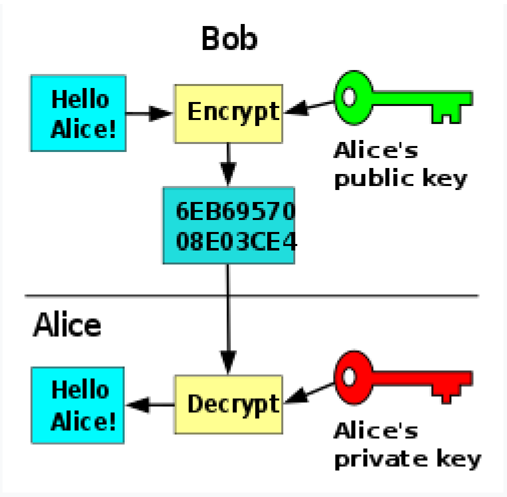
The Data Encryption Standard (DES) and the Advanced Encryption Standard (AES) are block cipher designs that have been designated cryptography standards by the US government (though DES's designation was finally withdrawn after the AES was adopted). Despite its deprecation as an official standard, DES (especially its still-approved and much more secure triple-DES variant) remains quite popular; it is used across a wide range of applications, from ATM encryptionto e-mail privacy[29] and secure remote access. Many other block ciphers have been designed and released, with considerable variation in quality. Many have been thoroughly broken, such as FEAL.

Stream ciphers, in contrast to the 'block' type, create an arbitrarily long stream of key material, which is combined with the plaintext bit-by-bit or character-by-character, somewhat like the one-time pad. In a stream cipher, the output stream is created based on a hidden internal state that changes as the cipher operates. That internal state is initially set up using the secret key material. RC4 is a widely used stream cipher; see Category:Stream ciphers. Block ciphers can be used as stream ciphers; see Block cipher modes of operation.

Cryptographic hash functions are a third type of cryptographic algorithm. They take a message of any length as input, and output a short, fixed length hash, which can be used in (for example) a digital signature. For good hash functions, an attacker cannot find two messages that produce the same hash. MD4 is a long-used hash function that is now broken; MD5, a strengthened variant of MD4, is also widely used but broken in practice. The US National Security Agency developed the Secure Hash Algorithm series of MD5-like hash functions: SHA-0 was a flawed algorithm that the agency withdrew; SHA-1 is widely deployed and more secure than MD5, but cryptanalysts have identified attacks against it; the SHA-2 family improves on SHA-1, but it isn't yet widely deployed; and the US standards authority thought it "prudent" from a security perspective to develop a new standard to "significantly improve the robustness of NIST's overall hash algorithm toolkit." Thus, a hash function design competition was meant to select a new U.S. national standard, to be called SHA-3, by 2012. The competition ended on October 2, 2012 when the NIST announced that Keccak would be the new SHA-3 hash algorithm. Unlike block and stream ciphers that are invertible, cryptographic hash functions produce a hashed output that cannot be used to retrieve the original input data. Cryptographic hash functions are used to verify the authenticity of data retrieved from an untrusted source or to add a layer of security.

Message authentication codes (MACs) are much like cryptographic hash functions, except that a secret key can be used to authenticate the hash value upon receipt; this additional complication blocks an attack scheme against bare digest algorithms, and so has been thought worth the effort.

1.4.2) Public-Key Cryptography



Public-key cryptography, where different keys are used for encryption and decryption

Symmetric-key cryptosystems use the same key for encryption and decryption of a message, though a message or group of messages may have a different key than others. A significant disadvantage of symmetric ciphers is the key management necessary to use them securely. Each distinct pair of communicating parties must, ideally, share a different key, and perhaps each ciphertext exchanged as well. The number of keys required increases as the square of the number of network members, which very quickly requires complex key management schemes to keep them all consistent and secret. The difficulty of securely establishing a secret key between two communicating parties, when a secure channel does not already exist between them, also presents a chicken-and-egg problem which is a considerable practical obstacle for cryptography users in the real world.

The Diffie–Hellman and RSA algorithms, in addition to being the first publicly known examples of high quality public-key algorithms, have been among the most widely used. Others include the Cramer–Shoup cryptosystem, ElGamal encryption, and various elliptic curve techniques. See Category:Asymmetric-key algorithms.

To much surprise, a document published in 1997 by the Government Communications Headquarters (GCHQ), a British intelligence organization, revealed that cryptographers at GCHQ had anticipated several academic developments. Reportedly, around 1970, James H. Ellis had conceived the principles of asymmetric key cryptography. In 1973, Clifford Cocks invented a solution that essentially resembles the RSA algorithm. And in 1974, Malcolm J. Williamson is claimed to have developed the Diffie–Hellman key exchange.

Public-key cryptography can also be used for implementing digital signature schemes. A digital signature is reminiscent of an ordinary signature; they both have the characteristic of being easy for a user to produce, but difficult for anyone else to forge. Digital signatures can also be permanently tied to the content of the message being signed; they cannot then be 'moved' from one document to another, for any attempt will be detectable. In digital signature schemes, there are two algorithms: one for *signing*, in which a secret key is used to process the message (or a hash of the message, or both), and one for *verification,* in which the matching public key is used with the message to check the validity of the signature. RSA and DSA are two of the most popular digital signature schemes. Digital signatures are central to the operation of public key infrastructures and many network security schemes (e.g., SSL/TLS, many VPNs, etc.).

Public-key algorithms are most often based on the computational complexity of "hard" problems, often from number theory. For example, the hardness of RSA is related to the integer factorization problem, while Diffie–Hellman and DSA are related to the discrete logarithm problem. More recently, elliptic curve cryptography has developed, a system in which security is based on number theoretic problems involving elliptic curves. Because of the difficulty of the underlying problems, most public-key algorithms involve operations such as modular multiplication and exponentiation, which are much more computationally expensive than the techniques used in most block ciphers, especially with typical key sizes. As a result, public-key cryptosystems are commonly hybrid cryptosystems, in which a fast high-quality symmetric-key encryption algorithm is used for the message itself, while the relevant symmetric key is sent with the message, but encrypted using a public-key algorithm. Similarly, hybrid signature schemes are often used, in which a cryptographic hash function is computed, and only the resulting hash is digitally signed.

1.5) The Concept of Cryptanalysis

The goal of cryptanalysis is to find some weakness or insecurity in a cryptographic scheme, thus permitting its subversion or evasion.

It is a common misconception that every encryption method can be broken. In connection with his WWII work at Bell Labs, Claude Shannonproved that the one-time pad cipher is unbreakable, provided the key material is truly random, never reused, kept secret from all possible attackers, and of equal or greater length than the message. Most ciphers, apart from the one-time pad, can be broken with enough computational effort by brute force attack, but the amount of effort needed may be exponentially dependent on the key size, as compared to the effort needed to make use of the cipher. In such cases, effective security could be achieved if it is proven that the effort required (i.e., "work factor", in Shannon's terms) is beyond the ability of any adversary. This means it must be shown that no efficient method (as opposed to the time-consuming brute force method) can be found to break the cipher. Since no such proof has been found to date, the one-time-pad remains the only theoretically unbreakable cipher.

There are a wide variety of cryptanalytic attacks, and they can be classified in any of several ways. A common distinction turns on what Eve (an attacker) knows and what capabilities are available. In a ciphertext-only attack, Eve has access only to the ciphertext (good modern cryptosystems are usually effectively immune to ciphertext-only attacks). In a known-plaintext attack, Eve has access to a ciphertext and its corresponding plaintext (or to many such pairs). In a chosen-plaintext attack, Eve may choose a plaintext and learn its corresponding ciphertext (perhaps many times); an example is gardening, used by the British during WWII. In a chosen-ciphertext attack, Eve may be able to *choose* ciphertexts and learn their corresponding plaintexts. Finally in a man-in-the-middle attack Eve gets in between Alice (the sender) and Bob (the recipient), accesses and modifies the traffic and then forwards it to the recipient. Also important, often overwhelmingly so, are mistakes (generally in the design or use of one of the protocols involved; see Cryptanalysis of the Enigma for some historical examples of this).

Chapter 2: The Algorithm – Digital Cryptex

2.1) Background

Our algorithm is purely based on the very foundation laws of digital electronics. It uses the laws of Boolean algebra and hex codes to encrypt the input message. The common software which we use is Aurdino Uno.

The Aurdino Uno provide the platform for the encryption and decryption.

Once we enter the desired message, the serial monitor of software shows us the encrypted form, same applies for the decrypter.

2.2) Encrypted:

1) First, all spaces in input string is avoided.
2) All input strings are made case sensitive by manipulating ascii values.
3) All letters are subtracted by 64 to get in range from 0-26.
4) Then a 5-variabvle k-map will designed and all the variable will type into one 32-bit string.
5) Those min terms are then converted into hex file.

2.3) Decrypted:

1) All the grouped hex digits are converted into binary min terms(s2).
2) According to min term position an array is designed (s1).
3) Multiply the string to s1 to s2.
4) Then we get the string in random manner

Note: We can use a separate key that arranges the random decrypted message.

2.4) Extras:

Tools Used:

1) Arduino IDE
2) C++

Problems faced till now:

1) Our algorithm can accept only 6 letter string as of now.
2) Less compatible.
3) Power consumption is here as it uses the Arduino board continually.

Applications:

1) Data security
2) Homeland security
3) Encryption in communication channels

Chapter 3: The Code Fragments

3.1) For Encryption

```
#include <SoftwareSerial.h>


  SoftwareSerial mySerial(9, 10);
String encrypted_msg="";


int h,i;
String n="";
char w[5];
void setup() {
Serial.begin(9600);
mySerial.begin(9600);
    for(;;)
    {
        if(Serial.available()>0)
        {
        w[i] =Serial.read();
        n+=String(w[i]);
        h++;


       if(h==6)
       {
        break;
       }


        }
    }
```

```
Serial.print("Orignal_msg  :");
  Serial.print(n);
  Serial.print("\n");

  int i,j,s,f=0,x,g,a,b,c[30],m[100],k[4][8];
  k[0][0]=0;  k[0][1]=1;   k[0][2]=3;  k[0][3]=2;      k[0][4]=16; k[0][5]=17; k[0][6]=19;  k[0][7]=18;
  k[1][0]=4;  k[1][1]=5;   k[1][2]=7;  k[1][3]=6;      k[1][4]=20; k[1][5]=21;  k[1][6]=23;  k[1][7]=22;
  k[2][0]=12; k[2][1]=13;  k[2][2]=15; k[2][3]=14;     k[2][4]=28; k[2][5]=29;  k[2][6]=31;  k[2][7]=30;
  k[3][0]=8;  k[3][1]=9;   k[3][2]=11; k[3][3]=10;     k[3][4]=24; k[3][5]=25;  k[3][6]=27;  k[3][7]=26;


x=n.length();
n.replace(" ","");


for(i=0;i<x;i++)
  {
  a=n[i];
  b=a-32;

  if(a>=97 && a<=122)
  c[i]=b-64;

  else if(a>=65 && a<=90)
  c[i]=a-64;



 }
 for(i=0;i<x;i++)
 {
  for(j=0;j<=3;j++)
  {
   for(g=0;g<=7;g++)
   {
    if(c[i]==k[j][g])
     {
     k[j][g]=1;
   }
   }
   }
   }

  for(i=0;i<=3;i++)
   {
   for(j=0;j<=7;j++)
   {
   if(k[i][j]!=1)
   {
    k[i][j]=0;
   }




   }
  }
/*for(i=0;i<4;i++)
{
  for(j=0;j<8;j++)
  {
    Serial.print(k[i][j]);
    Serial.print("\t");
  }
}*/

for(i=0;i<4;i++)
{
  for(j=0;j<8;j++)
  {
    if(j==0 || j==4)
    {
      k[i][j]*=8;
    }

    else if(j==1 || j==5)
    {
      k[i][j]*=4;
    }

    else if(j==2 || j==6)
    {
      k[i][j]*=2;
    }

    else if(j==3 || j==7)
    {
      k[i][j]*=1;
    }
  }
}
int p[4][2],t;
for(i=0;i<4;i++)
```

```
{
  t=0;
  p[i][t]=0;
  for(j=0;j<4;j++)
  {
    p[i][t]+=k[i][j];
  }
  t=1;
  p[i][t]=0;
  for(j=4;j<8;j++)
  {
    p[i][t]+=k[i][j];
  }
}

for(i=0;i<4;i++)
{
  for(j=0;j<2;j++)
  {
  encrypted_msg+=String(p[i][j],HEX);


}}
Serial.print("encrypted_msg : ");
Serial.print(encrypted_msg);
  }
void loop()
{
  if (Serial.available()>0)
   switch(Serial.read())
  {
    case 's':
      SendMessage();
      break;
    case 'r':
      RecieveMessage();
      break;
  }

 if (mySerial.available()>0)
   Serial.write(mySerial.read());
}



 void SendMessage()
{
  mySerial.println("AT+CMGF=1");    //Sets the GSM Module in Text Mode
  delay(1000);  // Delay of 1000 milli seconds or 1 second
  mySerial.println("AT+CMGS=\"+918264860070\"\r"); // Replace x with mobile number
  delay(1000);
  mySerial.println(encrypted_msg);// The SMS text you want to send
  delay(100);
   mySerial.println((char)26);// ASCII code of CTRL+Z
  delay(1000);
}



 void RecieveMessage()
{
  mySerial.println("AT+CNMI=2,2,0,0,0"); // AT Command to receive a live SMS
  delay(1000);
 }
```

3.2) For Decryption

```
String encrypted_msg="43080020";
String s1="";
int s2[32]={0,1,3,2,16,17,19,18,4,5,7,6,20,21,23,22,12,13,15,14,28,29,31,30,8,9,11,10,24,25,27,26};
int s3[32];
char s4[32];
int i;
void setup() {
Serial.begin(9600);

 for(i=0; i<encrypted_msg.length(); i++)
    {
      switch(encrypted_msg[i])
      {
        case '0':
          s1+="0000";
          break;
        case '1':
          s1+="0001";
          break;
        case '2':
          s1+="0010";
          break;
        case '3':
          s1+="0011";
          break;
        case '4':
          s1+="0100";
          break;
        case '5':
```

```
   s1+="0101";
   break;
case '6':
   s1+="0110";
   break;
case '7':
   s1+="0111";
   break;
case '8':
   s1+="1000";
   break;
case '9':
   s1+="1001";
   break;
case 'a':
case 'A':
   s1+="1010";
   break;
case 'b':
case 'B':
   s1+="1011";
   break;
case 'c':
case 'C':
   s1+="1100";
   break;
case 'd':
case 'D':
   s1+="1101";
   break;
case 'e':
```

```
            case 'E':
               s1+="1110";
               break;
            case 'f':
            case 'F':
               s1+="1111";
               break;
         }
}
//Serial.print(s1);
//Serial.print("\n");


for(i=0;i<32;i++)
{
 if(s1[i]=='0')
 {
   s3[i]=0;
 }
 else if(s1[i]='1')
 {
   s3[i]=s2[i]+64;
 }
// Serial.println(s3[i]);
}
String decrypt_msg="";
 for(i=0;i<32;i++)
 {
  if(s3[i]!=0)
  {
    s4[i]=s3[i];
  }
  decrypt_msg+=String(s4[i]);
 // Serial.print(s4[i]);
 }
 Serial.print("decrypt_msg : ");
Serial.print(decrypt_msg);
}

void loop() {
  // put your main code here, to run repeatedly:

}
```

3.3) The Communication Code

```
String encrypted_msg="43080020";

String s1="";

int s2[32]={0,1,3,2,16,17,19,18,4,5,7,6,20,21,23,22,12,13,15,14,28,29,31,30,8,9,11,10,24,25,27,26};

int s3[32];

char s4[32];

int i;

void setup() {

Serial.begin(9600);


 for(i=0; i<encrypted_msg.length(); i++)

   {

      switch(encrypted_msg[i])

      {

         case '0':

            s1+="0000";

            break;

         case '1':

            s1+="0001";

            break;

         case '2':

            s1+="0010";

            break;

         case '3':

            s1+="0011";

            break;

         case '4':

            s1+="0100";

            break;

         case '5':
```

```
   s1+="0101";
   break;
case '6':
   s1+="0110";
   break;
case '7':
   s1+="0111";
   break;
case '8':
   s1+="1000";
   break;
case '9':
   s1+="1001";
   break;
case 'a':
case 'A':
   s1+="1010";
   break;
case 'b':
case 'B':
   s1+="1011";
   break;
case 'c':
case 'C':
   s1+="1100";
   break;
case 'd':
case 'D':
   s1+="1101";
   break;
case 'e':
```

```
          case 'E':
             s1+="1110";
             break;
          case 'f':
          case 'F':
             s1+="1111";
             break;
       }
}
//Serial.print(s1);
//Serial.print("\n");


for(i=0;i<32;i++)
{
 if(s1[i]=='0')
 {
   s3[i]=0;
 }
 else if(s1[i]='1')
 {
   s3[i]=s2[i]+64;
 }
// Serial.println(s3[i]);
}
String decrypt_msg="";
 for(i=0;i<32;i++)
 {
  if(s3[i]!=0)
  {
    s4[i]=s3[i];
  }
  decrypt_msg+=String(s4[i]);
 // Serial.print(s4[i]);
 }
 Serial.print("decrypt_msg : ");
Serial.print(decrypt_msg);
}


void loop() {
  // put your main code here, to run repeatedly:
```

Conclusion:

We conclude that, the encryption and decryption algorithm uses the founding laws of digital electronics and it converts the input string into a stream of hex codes and the secret key is passed between sender and receiver in order to receive the message successfully.
