Internet Security: a Survey on Cryptography

Zhang XiaoTing *SJTU,* Li Yu *SJTU,* Wang Hao *SJTU,*

Abstract

Cryptography (or cryptology) C practice and study of hiding information. Modern cryptography intersects the disciplines of mathematics, computer science, and engineering. Applications of cryptography include ATM cards, computer passwords, and electronic commerce.

Index Terms

keyword

CONTENTS

- *We read hundreds of articles about cryptography to finfish this survey, cost about 5 weeks. Finished: 2016/6/17*
- *This is the first time for us to do the survey. So even the survey may be terrible, we have done our best.*

This is the final project for Computer Network

1 Introduction

Cryptography or cryptology is the practice and study of techniques for secure communication in the presence of third parties called adversaries. [1] More generally, cryptography is about constructing and analyzing protocols that prevent third parties or the public from reading private messages; [2] various aspects in information security such as data confidentiality, data integrity, authentication, and non-repudiation [3] are central to modern cryptography. Modern cryptography exists at the intersection of the disciplines of mathematics, computer science, and electrical engineering. Applications of cryptography include ATM cards, computer passwords, and electronic commerce.

Cryptography prior to the modern age was effectively synonymous with encryption, the conversion of information from a readable state to apparent nonsense. The originator of an encrypted message (Alice) shared the decoding technique needed to recover the original information only with intended recipients (Bob), thereby precluding unwanted persons (Eve) from doing the same. The cryptography literature often uses Alice ("A") for the sender, Bob ("B") for the intended recipient, and Eve ("eavesdropper") for the adversary. [4] Since the development of rotor cipher machines in World War I and the advent of computers in World War II, the methods used to carry out cryptology have become increasingly complex and its application more widespread.

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 provably cannot be broken even with unlimited computing powerłan example is the one-time padłbut these schemes are more difficult to implement than the best theoretically breakable but computationally secure mechanisms.

The growth of cryptographic technology has raised a number of legal issues in the information age. Cryptography's potential for use as a tool for espionage and sedition has led many governments to classify it as a weapon and to limit or even prohibit its use and export. [5] In some jurisdictions where the use of cryptography is legal, laws permit investigators to compel the disclosure of encryption keys for documents relevant to an investigation. [6] [7] Cryptography also plays a major role in digital rights management and copyright infringement of digital media. [8]

2 Symmetric-key cryptography

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).(see Figure 1) This was the only kind of encryption publicly known until June 1976 [9]. Symmetric key ciphers are implemented as either block ciphers or stream ciphers.

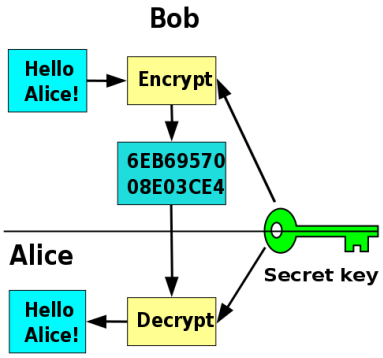


Fig. 1: Symmetric-key cryptography, where a single key is used for encryption and decryption

2.1 Block 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). [10] 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 encryption [11] to e-mail privacy [12] and secure remote access [13]. Many other block ciphers have been designed and released, with considerable variation in quality. Many have been thoroughly broken, such as FEAL. [14] [15]

2.1.1 Feistel cipher

Many modern and also some old symmetric block ciphers are based on Feistel networks (e.g. GOST 28147-89 block cipher), and the structure and properties of Feistel ciphers have been extensively explored by cryptographers. Specifically, Michael Luby and Charles Rackoff analyzed the Feistel cipher construction, and proved that if the round function is a cryptographically secure pseudorandom function, with Ki used as the seed, then 3 rounds are sufficient to make the block cipher a pseudorandom permutation, while 4 rounds are sufficient to make it a "strong" pseudorandom permutation (which means that it remains pseudorandom even to an adversary who gets oracle access to its inverse permutation). [16]

Because of this very important result of Luby and Rackoff, Feistel ciphers are sometimes called LubyCRackoff block ciphers. Further theoretical work has generalized the construction somewhat, and given more precise bounds for security. [17]

Let **F** be the round function and let $K_0, K_1, \ldots, K_n$ be the sub-keys for the rounds $0, 1, \ldots, n$ respectively.
Then the basic operation is as follows:
Split the plaintext block into two equal pieces, (L_0, R_0)

For each round $i = 0, 1, \ldots, n$, compute

$$L_{i+1} = R_i$$
$$R_{i+1} = L_i \oplus F(R_i, K_i)$$

Then the ciphertext is (R_{n+1}, L_{n+1}).

Decryption of a ciphertext (R_{n+1}, L_{n+1}) is accomplished by computing for $i = n, n-1, \ldots, 0$

$$R_i = L_{i+1}$$
$$L_i = R_{i+1} \oplus F(L_{i+1}, K_i)$$

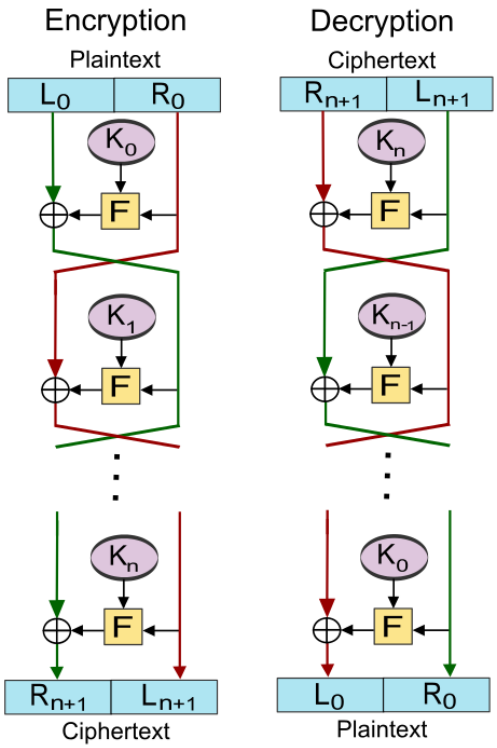
Then (L_0, R_0) is the plaintext again.

One advantage of the Feistel model compared to a substitution-permutation network is that the round function **F** does not have to be invertible.

The diagram illustrates both encryption and decryption. Note the reversal of the subkey order for decryption; this is the only difference between encryption and decryption.

Unbalanced Feistel ciphers use a modified structure where L_0 and R_0 are not of equal lengths. [18] The Skipjack cipher is an example of such a cipher. The Texas Instruments Digital Signature Transponder uses a proprietary unbalanced Feistel cipher to perform challenge-response authentication. [19]

The Thorp shuffle is an extreme case of an unbalanced Feistel cipher in which one side is a single bit. This has better provable security than a balanced Feistel cipher but requires more rounds. [20]



2.1.2 Advanced Encryption Standard

The Advanced Encryption Standard (AES), also known as Rijndael [21] [22] (its original name), is a specification for the encryption of electronic data established by the U.S. National Institute of Standards and Technology (NIST) in 2001. [23]

AES is based on the Rijndael cipher [22] developed by two Belgian cryptographers, Joan Daemen and Vincent Rijmen, who submitted a proposal to NIST during the AES selection process. [24] Rijndael is a family of ciphers with different key and block sizes.

AES is based on a design principle known as a substitution-permutation network, combination of both substitution and permutation, and is fast in both software and hardware. [25] Unlike its predecessor DES, AES does not use a Feistel network. AES is a variant of Rijndael which has a fixed block size of 128 bits, and a key size of 128, 192, or 256 bits. By contrast, the Rijndael specification per se is specified with block and key sizes that may be any multiple of 32 bits, both with a minimum of 128 and a maximum of 256 bits.

AES operates on a 4×4 column-major order matrix of bytes, termed the state, although some versions of Rijndael have a larger block size and have additional columns in the state. Most AES calculations are done in a special finite field.

For instance, if there are 16 bytes, $b_0, b_1, ..., b_{15}$, these bytes are represented as this matrix:

$$\begin{bmatrix} b_0 & b_4 & b_8 & b_{12} \\ b_1 & b_5 & b_9 & b_{13} \\ b_2 & b_6 & b_{10} & b_{14} \\ b_3 & b_7 & b_{11} & b_{15} \end{bmatrix}$$

The key size used for an AES cipher specifies the number of repetitions of transformation rounds that convert the input, called the plaintext, into the final output, called the ciphertext. The number of cycles of repetition are as follows:

- 10 cycles of repetition for 128-bit keys.
- 12 cycles of repetition for 192-bit keys.
- 14 cycles of repetition for 256-bit keys.

Each round consists of several processing steps, each containing four similar but different stages, including one that depends on the encryption key itself. A set of reverse rounds are applied to transform ciphertext back into the original plaintext using the same encryption key.

In the SubBytes step, each byte $a_{i,j}$ in the state matrix is replaced with a SubByte $S(a_{i,j})$ using an 8-bit substitution box, the Rijndael S-box. This operation provides the non-linearity in the cipher. The S-box used is derived from the multiplicative inverse over GF(2^8), known to have good non-linearity properties. To avoid attacks based on simple algebraic properties, the S-box is constructed by combining the inverse function with an invertible affine transformation. The S-box is also chosen to avoid any fixed points (and so is a derangement), i.e., $S(a_{i,j}) \neq a_{i,j}$, and also any opposite fixed points, i.e., $S(a_{i,j}) \oplus a_{i,j} \neq$ 0xFF. While performing the decryption, Inverse SubBytes step is used, which requires first taking the affine transformation and then finding the multiplicative inverse (just reversing the steps used in SubBytes step). See Figure 2.

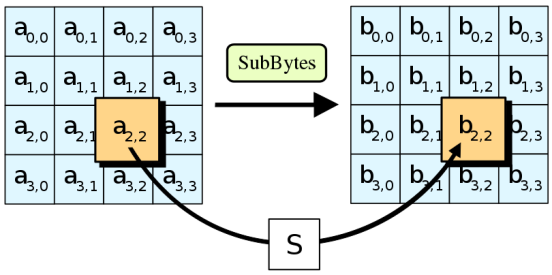


Fig. 2: In the SubBytes step, each byte in the state is replaced with its entry in a fixed 8-bit lookup table, $S : b_{ij} = S(a_{ij})$.

The ShiftRows step operates on the rows of the state; it cyclically shifts the bytes in each row by a certain offset. For AES, the first row is left unchanged. Each byte of the second row is shifted one to the left. Similarly, the third and fourth rows are shifted by offsets of two and three respectively. For blocks of sizes 128 bits and 192 bits, the shifting pattern is the same. Row n is shifted left circular by n-1 bytes. In this way, each column of the output state of the ShiftRows step is composed of bytes from each column of the input state. (Rijndael variants with a larger block size have slightly different offsets). For a 256-bit block, the first row is unchanged and the shifting for the second, third and fourth row is 1 byte, 3 bytes and 4 bytes respectivelyłthis change only applies for the Rijndael cipher when used with a 256-bit block, as AES does not use 256-bit blocks. The importance of this step is to avoid the columns being linearly independent, in which case, AES degenerates into four independent block ciphers. See Figure 3.

In the MixColumns step, the four bytes of column of the state are combined using an invertible linear transformation. The MixColumns function takes four bytes as input and outputs four bytes, where each input byte affects all four output bytes. Together with ShiftRows, MixColumns provides diffusion in the cipher.

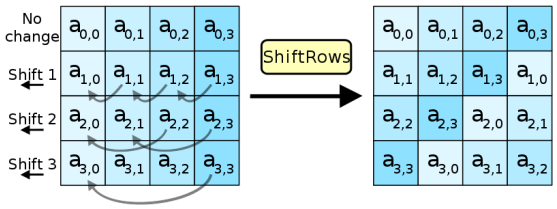


Fig. 3: In the ShiftRows step, bytes in each row of the state are shifted cyclically to the left. The number of places each byte is shifted differs for each row

During this operation, each column is transformed using a fixed matrix (matrix multiplied by column gives new value of column in the state):

$$\begin{bmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{bmatrix}$$

Matrix multiplication is composed of multiplication and addition of the entries. Entries are 8 bit bytes treated as coefficients of polynomial of order x^7. Addition is simply XOR. Multiplication is modulo irreducible polynomial $x^8 + x^4 + x^3 + x + 1$. If processed bit by bit then after shifting a conditional XOR with 0x1B should be performed if the shifted value is larger than 0xFF (overflow must be corrected by subtraction of generating polynomial). These are special cases of the usual multiplication in $\mathbf{GF}(2^8)$.

In more general sense, each column is treated as a polynomial over $\mathbf{GF}(2^8)$ and is then multiplied modulo $x^4 + 1$ with a fixed polynomial $c(x) = 0\text{x}03 \cdot x^3 + x^2 + x + 0\text{x}02$. The coefficients are displayed in their hexadecimal equivalent of the binary representation of bit polynomials from $\mathbf{GF}(2)[x]$. The MixColumns step can also be viewed as a multiplication by the shown particular MDS matrix in the finite field $\mathbf{GF}(2^8)$. This process is described further in the article Rijndael mix columns. See Figure 4.

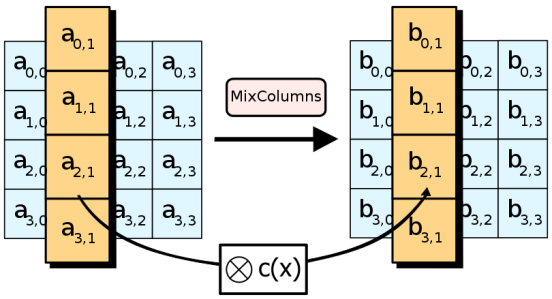


Fig. 4: In the MixColumns step, each column of the state is multiplied with a fixed polynomial $c(x)$.

In the AddRoundKey step, the subkey is combined with the state. For each round, a subkey is derived from the main key using Rijndael's key schedule; each subkey is the same size as the state. The subkey is added by combining each byte of the state with the corresponding byte of the subkey using bitwise XOR. See Figure 5.

On systems with 32-bit or larger words, it is possible to speed up execution of this cipher by combining the SubBytes and ShiftRows steps with the MixColumns step by transforming them into a sequence of table lookups. This requires four 256-entry 32-bit tables, and utilizes a total of four kilobytes (4096 bytes) of memory ł one kilobyte for each table. A round can then be done with 16 table lookups and 12 32-bit exclusive-or operations, followed by four 32-bit exclusive-or operations in the AddRoundKey step. [26]

If the resulting four-kilobyte table size is too large for a given target platform, the table lookup operation can be performed with a single 256-entry 32-bit (i.e. 1 kilobyte) table by the use of circular rotates.

Using a byte-oriented approach, it is possible to combine the SubBytes, ShiftRows, and MixColumns steps into a single round operation.[12]

Until May 2009, the only successful published attacks against the full AES were side-channel attacks on some specific implementations. The National Security Agency (NSA) reviewed all the AES finalists, including Rijndael, and stated that all of them were secure enough for U.S. Government non-classified data. In June 2003, the U.S. Government announced that AES could be used to protect classified information:

The design and strength of all key lengths of the AES algorithm (i.e., 128, 192 and 256) are sufficient to protect classified information up to the SECRET level. TOP SECRET information will require use of either the 192 or 256 key lengths. The implementation of AES in products intended to protect national security systems and/or information must be reviewed and certified by NSA prior to their acquisition and use. [27]

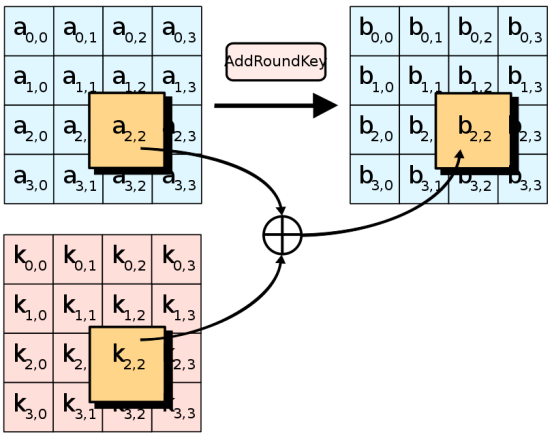


Fig. 5: In the AddRoundKey step, each byte of the state is combined with a byte of the round subkey using the XOR operation ($\oplus$).

AES has 10 rounds for 128-bit keys, 12 rounds for 192-bit keys, and 14 rounds for 256-bit keys. By 2006, the best known attacks were on 7 rounds for 128-bit keys, 8 rounds for 192-bit keys, and 9 rounds for 256-bit keys. [28]

2.1.3 Blowfish

Blowfish has a 64-bit block size and a variable key length from 32 bits up to 448 bits. [29] It is a 16-round Feistel cipher and uses large key-dependent S-boxes. In structure it resembles CAST-128, which uses fixed S-boxes.

The diagram to the left shows Blowfish's encryption routine. Each line represents 32 bits. There are five subkey-arrays: one 18-entry P-array (denoted as K in the diagram, to avoid confusion with the Plaintext) and four 256-entry S-boxes (S0, S1, S2 and S3).

Every round r consists of 4 actions: First, XOR the left half (L) of the data with the r th P-array entry, second, use the XORed data as input for Blowfish's F-function, third, XOR the F-function's output with the right half (R) of the data, and last, swap L and R. See Figure 6

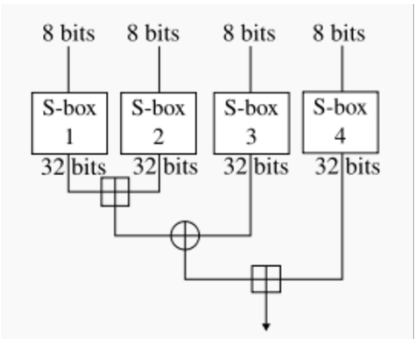


Fig. 6: The round function (Feistel function) of Blowfish

The F-function splits the 32-bit input into four eight-bit quarters, and uses the quarters as input to the S-boxes. The S-boxes accept 8-bit input and produce 32-bit output. The outputs are added modulo 232 and XORed to produce the final 32-bit output (see image in the upper right corner). [29]

After the 16th round, undo the last swap, and XOR L with K18 and R with K17 (output whitening).

Decryption is exactly the same as encryption, except that P1, P2,..., P18 are used in the reverse order. This is not so obvious because xor is commutative and associative. A common misconception is to use inverse order of encryption as decryption algorithm (i.e. first XORing P17 and P18 to the ciphertext block, then using the P-entries in reverse order).

Because the P-array is 576 bits long, and the key bytes are XORed through all these 576 bits during the initialization, many implementations support key sizes up to 576 bits. While this is certainly possible, the 448 bits limit is here to ensure that every bit of every subkey depends on every bit of the key, [29] as the last four values of the P-array don't affect every bit of the ciphertext. This point should be taken in consideration for implementations with a different number of rounds, as even though it increases security against an exhaustive attack, it weakens the security guaranteed by the algorithm. And given the slow initialization of the cipher with each change of key, it is granted a natural protection against brute-force attacks, which doesn't really justify key sizes longer than 448 bits.

Blowfish in pseudocode:

```
uint32_t P[18];
uint32_t S[4][256];

uint32_t f (uint32_t x) {
   uint32_t h = S[0][x >> 24] + S[1][x >> 16 & 0xff];
   return ( h ^ S[2][x >> 8 & 0xff] ) + S[3][x & 0xff];
}

void encrypt (uint32_t & L, uint32_t & R) {
   for (int i=0 ; i<16 ; i += 2) {
      L ^= P[i];
      R ^= f(L);
      R ^= P[i+1];
      L ^= f(R);
   }
   L ^= P[16];
   R ^= P[17];
   swap (L, R);
}

void decrypt (uint32_t & L, uint32_t & R) {
   for (int i=16 ; i > 0 ; i -= 2) {
      L ^= P[i+1];
      R ^= f(L);
      R ^= P[i];
      L ^= f(R);
   }
   L ^= P[1];
   R ^= P[0];
   swap (L, R);
}

{
   // ...
   // initializing the P-array and S-boxes with values derived from pi; omitted in the example
   // ...
   for (int i=0 ; i<18 ; ++i)
      P[i] ^= key[i % keylen];
   uint32_t L = 0, R = 0;
   for (int i=0 ; i<18 ; i+=2) {
      encrypt (L, R);
      P[i] = L; P[i+1] = R;
   }
   for (int i=0 ; i<4 ; ++i)
      for (int j=0 ; j<256; j+=2) {
         encrypt (L, R);
         S[i][j] = L; S[i][j+1] = R;
      }
}
```

2.1.4 Data Encryption Standard

The Data Encryption Standard (DES) was once a predominant symmetric-key algorithm for the encryption of electronic data. It was highly influential in the advancement of modern cryptography in the academic world. Developed in the early 1970s at IBM and based on an earlier design by Horst Feistel, the algorithm was submitted to the National Bureau of Standards (NBS) following the agency's invitation to propose a candidate for the protection of sensitive, unclassified electronic government data. See Figure 7.

DES is the archetypal block cipherłan algorithm that takes a fixed-length string of plaintext bits and transforms it through a series of complicated operations into another ciphertext bitstring of the same length. [30] In the case of DES, the block size is 64 bits. DES also uses a key to customize the transformation [31], so that decryption can supposedly only be performed by those who know the particular key used to encrypt. The key ostensibly consists of 64 bits; however, only

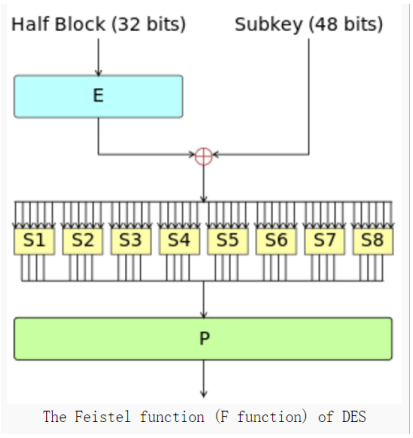


Fig. 7: General

56 of these are actually used by the algorithm. Eight bits are used solely for checking parity, and are thereafter discarded. Hence the effective key length is 56 bits. See Figure 8.

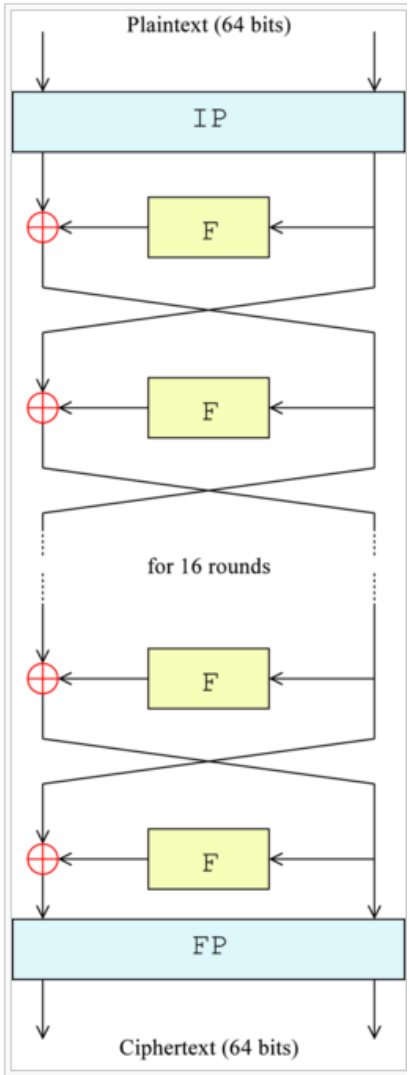


Fig. 8: General

Like other block ciphers, DES by itself is not a secure means of encryption but must instead be used in a mode of operation. FIPS-81 specifies several modes for use with DES. [32] Further comments on the usage of DES are contained in

FIPS-74. [33]

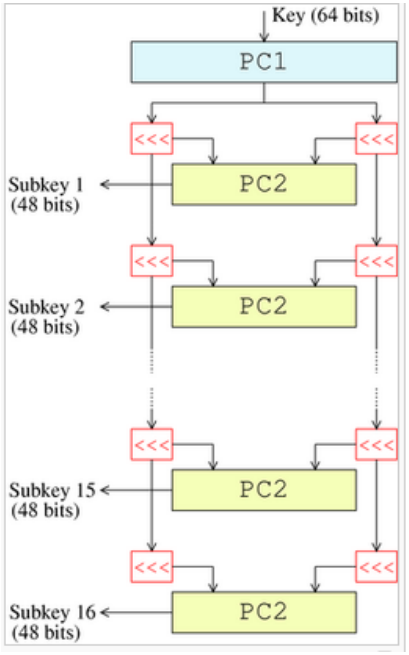


Fig. 9: General

Figure 9 illustrates the key schedule for encryption - the algorithm which generates the subkeys. Initially, 56 bits of the key are selected from the initial 64 by Permuted Choice 1 $(PC-1)$ the remaining eight bits are either discarded or used as parity check bits. The 56 bits are then divided into two 28-bit halves; each half is thereafter treated separately. In successive rounds, both halves are rotated left by one or two bits (specified for each round), and then 48 subkey bits are selected by Permuted Choice $2(PC-2)$ 24 bits from the left half, and 24 from the right. The rotations (denoted by "<<<" in the diagram) mean that a different set of bits is used in each subkey; each bit is used in approximately 14 out of the 16 subkeys. [34]

The key schedule for decryption is similarłthe subkeys are in reverse order compared to encryption. Apart from that change, the process is the same as for encryption. The same 28 bits are passed to all rotation boxes.

2.2 Stream ciphers

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.

2.2.1 A5/1 & A5/2

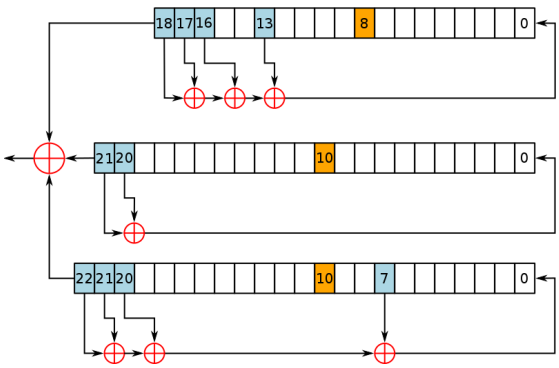


Fig. 10: The A5/1 stream cipher uses three LFSRs.

A GSM transmission is organised as sequences of bursts [35]. In a typical channel and in one direction, one burst is sent every 4.615 milliseconds and contains 114 bits available for information. A5/1 is used to produce for each burst a 114 bit sequence of keystream which is XORed with the 114 bits prior to modulation. A5/1 is initialised using a 64-bit key together with a publicly known 22-bit frame number. Older fielded GSM implementations using Comp128v1 for key generation, had 10 of the key bits fixed at zero, resulting in an effective key length of 54 bits. This weakness was rectified with the introduction of Comp128v2 which yields proper 64 bits keys. When operating in GPRS / EDGE mode, higher bandwidth radio modulation allows for larger 348 bits frames, and A5/3 is then used in a stream cipher mode to maintain confidentiality.

Figure 10 show that a register is clocked if its clocking bit (orange) agrees with the clocking bit of one or both of the other two registers.

A5/1 is based around a combination of three linear feedback shift registers (LFSRs) with irregular clocking. The three shift registers are specified as follows:

LFSR number	Length in bits	Feedback polynomial	Clocking bit	Tapped bits
1	19	$x^{19} + x^{18} + x^{17} + x^{14} + 1$	8	13, 16, 17, 18
2	22	$x^{22} + x^{21} + 1$	10	20, 21
3	23	$x^{23} + x^{22} + x^{21} + x^{8} + 1$	10	7, 20, 21, 22

The bits are indexed with the least significant bit (LSB) as 0.

The registers are clocked in a stop/go fashion using a majority rule. Each register has an associated clocking bit. At each cycle, the clocking bit of all three registers is examined and the majority bit is determined. A register is clocked if the clocking bit agrees with the majority bit. Hence at each step at least two or three registers are clocked, and each register steps with probability 3/4.

Initially, the registers are set to zero. Then for 64 cycles, the 64-bit secret key is mixed in according to the following scheme: in cycle $0 \leq i < 64$, the ith key bit is added to the least significant bit of each register using XOR ł:$R[0] = R[0] \oplus K[i]$. Each register is then clocked.

Similarly, the 22-bits of the frame number are added in 22 cycles. Then the entire system is clocked using the normal majority clocking mechanism for 100 cycles, with the output discarded. After this is completed, the cipher is ready to produce two 114 bit sequences of output keystream, first 114 for downlink, last 114 for uplink.

A5/2 was used for export instead of the relatively stronger (but still weak) A5/1 [36].

2.2.2 FISH

The FISH (FIbonacci SHrinking) stream cipher is a fast software based stream cipher using Lagged Fibonacci generators [37], plus a concept from the shrinking generator cipher. It was published by Siemens in 1993. FISH is quite fast in software and has a huge key length. However, in the same paper where he proposed Pike, Ross Anderson showed that FISH can be broken with just a few thousand bits of known plaintext. [38]

We consider two pseudo random generators A and S. A produces a sequence $a_0, a_l, ..,$ of elements of $GF(2)^{nA}$. S produces a sequence $s_o, s_l, ..$, of elements of $GF(2)^{ns}$. We apply a mapping $d : GF(2)^{nA} \rightarrow GF(2)^{ns}$ to the elements of $s_o, s_l, ..$ to decide which elements are accepted and which arc discarded. In the original shrinking generator only elements genererated by A are accepted or discarded, in our generalization the results of S are treated the same. Another difference of our scheme is that the accepted elements are not yet the final result, another stage of processing is needed. We define the shrinking procedure as follows: If $d(s_i) = 1$ then a_i and s_i are accepted, otherwise they are discarded. That is, we define a sequence $i_l, i_2, ..., i_k, ..,$ where i_k is the k-th position in $s_0, s_l, ...$ with $d(s_i) = 1$. Wc have $d(s_{i_k}) = 1$ and $\#\{j \in 0...i_k - 1 | d(s_j) = 1\} = k - 1$.

We consider the shrunk sequences $z_0, z_l, ...,$ which is $a_{i_1}, a_{i_2}, ..,$ and $h_0, h_1, ...$ which is $s_{i_l}, s_{i_2}, ...$. For all elements $h_j d(h_j) = 1$ holds. The principle of the generalized shrinking generator is illustrated in Figure 11.

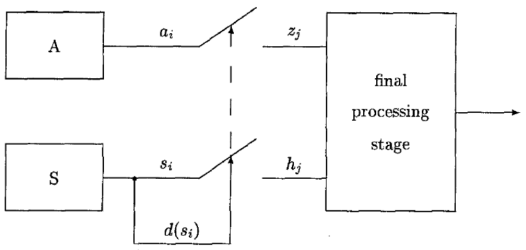


Fig. 11: Principle of the generalized shrinking generator.

In the original shrinking generator there was $n_A = 1$ and $n_s = 1$. The mapping $d()$ was the identity, $z_0, z_l, ...,$ were used as the output bits of the generator.

In order to make full use of the 32 bit wordlength of most popular processors, we choose $n_A = 32$ and $n_s = 32$.

For both A and S we use the fastest software pseudo random number generator we know, namely the additive generator which is also called the lagged Fibonacci generator. We define

$$a_i = a_{i-55} + a_{i-24} \; mod \; 2^{32}$$

and

$$s_i = s_{i-52} + s_{i-19} \; mod \; 2^{32}$$

where + stands for the arithmetical addition operation with carry, and the binary vectors are interpreted as unsigned numbers in the usual way [37]. The values $a_{-55}, a_{-54}, ..., a_{-1} and s_{-52}, s_{-51}, ..., s_{-1}$ are initial values of the generators and must be derived from the key. The sequence of the least significant bits of a lagged Fibonacci generator is generated by a linear feedback shift register (LFSR) where the feedback polynomial is a trinomial.

The mapping $d : GF(2)^{nA} \rightarrow GF(2)^{ns}$ maps a 32 bit vector to its least significant bit, $d((b_{31}, b_{30}, ..., b_0)) = b_0$.

It would be unsecure to use the shrunk sequence $z_0, z_l, ...$, as the result like in the original shrinking generator, since the underlying linear structure could be detected. With probability 1/8 a triple of elements a_i, a_{i-55}, and a_{i-24} is accepted as elements of $z_0, z_l, ...$. An attacker could try to identify such triples by adding elements of $z_0, z_l, ...$, with a suitable distance and checking whether the sum turns up some elements later. Therefore we have to hide the linear structure of $z_0, z_l, ...$

We split the sequences $z_0, z_l, ...$, and $h_0, h_1, ...$ up into pairs (z_{2i}, z_{2i+l}) and (h_{2i}, h_{2i+1}) and derive the two 32 bit output words r_{2i} and r_{2i+1} from these. We define

$$\begin{aligned} c_{2i} &= z_{2i} \oplus (h_{2i} \wedge h_{2i+1}) \\ d_{2i} &= h_{2i+1} A (c_{2i} \oplus z_{2i+1}) \\ r_{2i} &= c_{2i} \oplus d_{2i} \\ r_{2i+1} &= z_{2i+1} \oplus d_{2i} \end{aligned}$$

where $\oplus$ stands for the bitwise logical XOR operation and $\wedge$ for the bitwise logical AND. The last three equations achieve an exchange of those bits of c_{2i} and z_{2i+1} which are 1 in h_{2i+1}. The operations are visualized in 12.

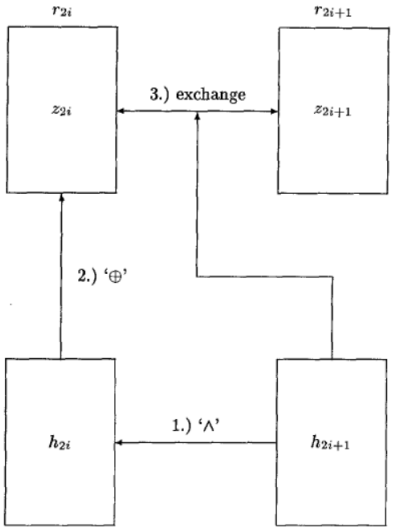


Fig. 12: Final processing stage: The output words r i and r2i+l are derived by executing the indicated operations. [38]

The least significant bits of h_{2i} and h_{2i+1} are 1 because of our choice of the function d. Therefore it is possible to reconstruct the least significant bits of z_{2i} and z_{2i+1} from r_{2i} and r_{2i+1}, and vice versa the least significant bits of r_{2i} and r_{2i+1} follow from r_{2i} and r_{2i+1}. This implies that the least significant bits of the output words of Fish are the bits of the underlying LFSR shrinking generator which has a feedback trinomial. [38]

2.2.3 ISAAC

ISAAC (indirection, shift, accumulate, add, and count) is a cryptographically secure pseudorandom number generator and a stream cipher designed by Robert J. Jenkins Jr. in 1993. [39]

IA (Indirection, Addition) is slightly biased but it appears to be secure. It is immune to Gaussian elimination. IBAA (Indirection, Barrelshift, Accumulate and Add) eliminates the bias in IA without damaging security. ISAAC (Indirection, Shift, Accumulate, Add, and Count) is faster than IBAA, guarantees no bad seeds or short cycles, and makes orderly states disorderly faster.

IA was designed to satisfy these goals:

- Deducing the internal state from the results should be intractable.
- The code should be easy to memorize.
- It should be as fast as possible.

More requirements were added for IBAA:

- It should by cryptographicaUy secure [40] [41].
- No biases should be detectable for the entire cycle length.
- Short cycles should be astronomically rare.

A generator was found that had the appropriate levels of bias. It used an accumulator and barrelshifts. IBAA was formed by combining it with IA without introducing bias or reducing the security of IA. (Any unbreakable unbiased generator which has long cycles must be cryptographically secure.)

ISAAC took away the requirement of easy memorization but added more:

- The C code should be optimized for speed.
- Orderly states should become disorderly quickly.
- There should be no short cycles at all.

ISAAC is similar in form and function to the alleged RC4, although the generators were developed independently. ISAAC is three times faster, less biased, and has longer minimum and average cycle lengths. ISAAC requires an amortized 18.75 machine instructions to produce a 32-bit value. ISAAC should be useful as a stream cipher, for simulations, and as a general purpose pseudorandom number generator.

ISAAC is in fact a family of algorithms indexed by parameter m, which is a positive integer. The internal state of ISAAC at time t consists of a table $S_t = \{s_t[0], ..., s_t[m-1]\}$ of $m = 2^n$ K-bit words and of two K-bit words a_t and i_t . Let z_t denote the output K-bit word of ISAAC at time t. Let initially $i_0 = a_0 = 0$. $K = 2n + \Delta$, $\Delta > 0$. The key of ISAAC is the initial table S_0.

Jenkins takes $m = 256, n = 8, K = 32, p0 = 13, p1 = 6, p2 = 2, p3 = 16, \theta_1 = \theta_2 = 2$.

Let $\theta_1, \theta_2 < n$.

$$G(a_{t-1}, t, p(t)) = \begin{cases} ((a_{t-1} << p0) \oplus a_{t-1}) \text{ if t} = 0 \pmod 4 \\ ((a_{t-1} >> p1) \oplus a_{t-1}) \text{ if t} = 1 \pmod 4 \\ ((a_{t-1} << p2) \oplus a_{t-1}) \text{ if t} = 2 \pmod 4 \\ ((a_{t-1} >> p3) \oplus a_{t-1}) \text{ if t} = 3 \pmod 4 \end{cases}$$

where $>>$ and $<<$ indicate rotation to the right and left, and

$$p(t) = \begin{cases} p0 \text{ if t} = 0 \pmod 4 \\ p1 \text{ if t} = 1 \pmod 4 \\ p2 \text{ if t} = 2 \pmod 4 \\ p3 \text{ if t} = 3 \pmod 4 \end{cases}$$

The next-state function F

1) $i_t = i_{t-1} + 1$ (mod m).
2) $a_t = (G(a_{t-1}, t, p(t)) + s_t[(t + m/2)(mod\ m)])(mod\ 2^K)$.
3) $s_t[i_t] = (s_{t-1}[(s_{t-1}[i_t] >> \theta_1)(mod\ m)] + a_t + z_{t-1})(mod\ 2^K)$.

The output function f

Output: $z_t = (s_t[(s_t[i_t] >> (n + \theta_2))(mod\ m)] + s_{t-1}[i_t])(mod\ 2^K)$. [42]

3 Preliminary of public-key cryptography: Number Theory Reivew

3.1 Modular Arithmetic

Definition 1. An integer $a \in \mathbb{Z}$ is congruent modulo n to another integer $b \in \mathbb{Z}$, $a \equiv b$ (mod n), if and only if $n|a - b$.

This relation is an equivalence relation, and the set of congruence classes under this equivalence relation forms a ring $(\mathbb{Z}_n, +, \times)$. The operations on congruence classes are defined as follows:

1) Addition $[a] + [b] = [a + b]$
2) Multiplication $[a] * [b] = [a * b]$

These definitions are justified and are consistent by the definition of equivalence modulo n.

The set of integers modulo n under just addition forms a group, i.e, every element has an inverse.

If n is a prime number p then $\mathbb{Z}_p$ is a field with p elements. Every element has a multiplicative inverse in $\mathbb{Z}_p$ for some prime p as every element is coprime to the modulus. It turns out that the necessary and sufficient conditions for an element

to have a multiplicative inverse modulo n is such that $gcd(x \in \mathbb{Z}_n, n) = 1$, and the inverse can be found by using the Extended Euclidean Algorithm. This is accomplished by solving a certain instances of Bezout's Identity which will always have a solution over the integers, i.e $ax + by = gcd(a, b)$ always has a solution $x, y \in \mathbb{Z}$. When $gcd(a, b) = 1$ is the special case when finding inverses.

3.2 Fermat's Little Theorem

Theorem 2 (Fermat's Little Theorem)**.** *Suppose that p is a prime number, then* $\forall x[p \nmid x] \Rightarrow x^{p-1} \equiv 1$ *(mod p). This also states, by simple manipulation of the terms, that* $\forall x : x^p \equiv x$ *(mod p).*

This is then generalized by Euler's Theorem.

Theorem 3 (Euler's Theorem)**.** $\forall n$ *and* $gcd(a, n) = 1$*, then* $a^{\phi(n)} \equiv 1$ *(mod n).*

Where $\phi(n)$ counts the number of numbers coprime to n between 1 and n. In particular, if p is a prime number, then $\phi(p) = p-1$. If $y = pq$, where p and q are prime numbers, then $\phi(y) = \phi(p)\phi(q) = (p-1)(q-1)$. In general if $gcd(a, b) = 1$, i.e if a and b are coprime, then $\phi(ab) = \phi(a)\phi(b)$.

Nota Bene: Beware the Carmichael Numbers. They pretend to be primes in the sense that if c is a Carmichael Number then $a^c = a$ (mod c) for every a such that $gcd(a, c) = 1$. They screw up some basic primality testing algorithms. There are infinitely many of these things.

3.3 The Group $\mathbb{Z}_n^*$

For every Ring $\mathbb{Z}_n$ there exists a subgroup of elements that have multiplicative inverses modulo n. This group is called the group of units modulo n. In symbols we write $\mathbb{Z}_n^*$. This group consists of all elements of $\mathbb{Z}_n$ that are coprime to the modulus n An immediate consequence of this fact is that if p is a prime, then $\mathbb{Z}_p$ is a field, i.e. every nonzero element has a multiplicative inverse as every element between 1 and p is coprime to p.

Why? Well, if $a \in \mathbb{Z}_n$ and $gcd(a, n) = 1$, then the Bezout Identity $ax + ny = 1$ has a solution over the integers. So we get that $ny = 1 - ax \Rightarrow 1 \equiv ax$ (mod n). The proof of the other direction is basically the same and is very simple.

Thus, the number of elements (also called the order of the group) is equal to $\phi(n)$, i.e $|\mathbb{Z}_n^*| = \phi(n)$. If p is a prime, then $|\mathbb{Z}_p^*| = p - 1$.

These groups are **cyclic**, i.e, $\exists x \in \mathbb{Z}_n^*$ such that $\langle x \rangle \approx \mathbb{Z}_n^*$. This means that there is a generator of the group, i.e an element of order $\phi(n)$. An element generates a group if raising the element to powers eventually produces every element of the group. This fact is important to us as many cryptosystems depend on using a generator of the group.

3.4 Quadratic Residues: $\mathcal{QR}_n$

Given a prime p, then from the previous section we know that $\mathbb{Z}_p^* = \{1, \ldots, p-1\}$. We wish to study the elements that are squares of other elements.

Example: in $\mathbb{Z}_7^*$ we get that $\mathcal{QR}_7 = \{1, 2, 4\}$. We see this because:

$$1\ 2\ 3\ 4\ 5\ 6$$

all square to

$$1\ 4\ 2\ 2\ 4\ 1$$

Empirically, we see that $|\mathcal{QR}_7| = 3 = \frac{7-1}{2}$. Is it true in general that $|\mathcal{QR}_p| = \frac{p-1}{2}$? We see that $\forall x \in \mathbb{Z}_p^*$, x has two square roots.

Theorem 4. *Given* $a \in \mathbb{Z}_p^*$. $a \in \mathcal{QR}_p \iff a^{\frac{p-1}{2}} \equiv 1$ *(mod p).*

Proof. if $a \equiv x^2$ (mod p) we get:

$$\begin{aligned} a^{\frac{p-1}{2}} &\equiv (x^2)^{\frac{p-1}{2}} \text{ (mod p)} \\ &\equiv x^{p-1} \text{ (mod p)} \\ &\equiv 1 \text{ (mod p)} \end{aligned}$$

Where the last line holds by Fermat's Little Theorem. □

It is clear that if $a \in \mathbb{Z}_p^*$ then a has a square root modulo p if and only if a is a quadratic residue modulo p. When searching for square roots, one has to be sure that the number is a quadratic residue. If it is not a quadratic residue, then it simply does not have a square root modulo p. If p is an odd prime, then p must be either one of the following form:

1) $4n + 1$
2) $4n + 3$

If p is of the form of case 2, then there exist a polynomial time algorithm, but it is an open problem for case 1. (WHY? I think this is for finding square roots of a number modulo p).

If we're looking to solve the equation $x^2 \equiv b$ mod n, where $n = pq$ for p, q are prime numbers. Then solving this equation is equivalent to factoring n. If $p \equiv 3$ mod 4 then let $x = y^{\frac{p+1}{4}}$ mod p.

1) If y has a square root modulo p, then the square roots of y mod p are $\pm x$.
2) if y has no square roots mod p, then $-y$ has square roots mod p and are $\pm x$.

4 PUBLIC-KEY CRYPTOGRAPHY

Public-key cryptography is also known as asymmetric cryptography, which is a kind of cryptographic system that uses pairs of keys: public keys and private keys. Public keys are keys that may be disseminated widely while private keys are known only to the owner.

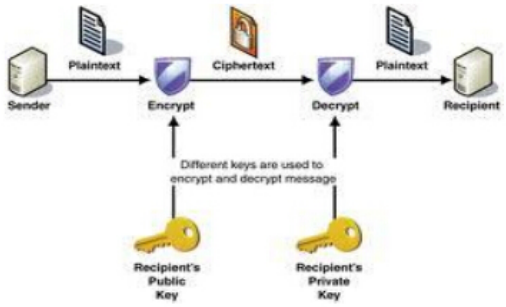


Fig. 13: Principle of the asymmetric cryptography.

These two keys are related mathematically and always being needed to be very large. Thus, even if you know one of them, it is nearly impossible to calculate another one.

The idea of asymmetric cryptography is proposed firstly by Ralph C. Merkle in 1974. Whitfield Diffie and Martin Hellman later in 1976 published a protocol to create keys for the sending and receiving party based on the idea of one-way function.

If the public key is the encryption key, the cryptosystem is often used to upload encrypted data to the owner of the private, for example, communication of online bank between clients and their managers which is under cipher's protection.

If the private key is the encryption one, the cryptosystem is often used to check the integrity and accuracy of the data and in order to check the identification of the send party, for example, a digital signature system.

Unlike the symmetric cryptography, which use the complexity of the algorithms to assure the failure of decryption, asymmetric cryptography construct some difficult problems in mathematics to assure the information security. So the operation in Public-key cryptography can always be easily described as following:

Suppose there are two users:A and B. A message X is sent from A to B. The public key is c, whose coresponding private key is d,which is hold only by B. Then: Before A send the message,he will calculate the ciphertext:

$$M = c(X)$$

After B receive the ciphertext,he can calculate the message:

$$X = d(M) = d(c(X))$$

Although the description of the algorithm can be very easy, the computation complexity may be still considerably large just because the difficulty of the math problems. Thus we can find that the math problem is very import in designing a public-key cryptosystem.

So in the rest part of this section, some public-key cryptosystem and their corresponding math problem will be introduced.

4.1 Prime factorization problem of very large integers : RSA cryptosystem

RSA is one of the first practical public-key cryptosystems and is widely used for secure data transmission even currently. In such a cryptosystem, the encryption key is public while the decryption key is kept secret. RSA is made of the initial letters of the surnames of Ron Rivest, Adi Shamir, and Leonard Adleman, who first publicly described the algorithm in 1977. However, Clifford Cocks, an English mathematician working for the UK intelligence agency GCHQ, had developed an equivalent system in 1973, but which was not declassified until 1997. [43]

A user of RSA creates and then publishes a public key based on two large prime numbers, along with an auxiliary value. The prime numbers must be kept secret. Anyone can use the public key to encrypt a message, but with currently

published methods, if the public key is large enough, only someone with knowledge of the prime numbers can feasibly decode the message. [44]

The system is as follows:

Two large primes p, q are chosen such that $p \neq q$. Then choose an encryption exponent e such that $gcd(e, \phi(pq)) = 1$. This implies that $e \in \mathbb{Z}_{pq}^*$, i.e. e has an inverse modulo n, let this inverse be d, i.e. $de \equiv 1 \bmod pq$.

The numbers $n = pq$ and e are assumed to be public, and d, p, q are kept secret. If M is the encoding of the message to encrypt, then $M \mapsto M^e = C \bmod n$. One has to be sure, however, that $M \in \mathbb{Z}_n$ as information would be lost otherwise. The decryption is carried out by raising C to the decryption exponent: $C^d \bmod n$. The private key is d the inverse of the encrypting exponent. The public key is the encrypting exponent itself. Public keys are obviously known to the public, and thus if Alice wants to send a secure message to Bob, she encrypts the message with Bob's public key, and then transmits the message. Only Bob can decrypt the message as only d is known to him. [45]

The mechanism of RSA is as follows:

For very large primes p and q, there is:

$$n = pq$$

Suppose the ciphertext is c and the message we want to send is m, then in the encryption step:

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

And in the decryption step, we get:

$$(m^e \mod n)^d \mod n = m^{ed} \mod n$$

In the number theory, if there is :

$$\begin{aligned} &\text{p and q are primes} \\ &n = pq \\ &z = (p-1)(q-1) \end{aligned}$$

then:

$$x^y \mod n = x^{(y \mod z)} \mod n$$

Let $x = m$,$y = ed$, we can get:

$$m^{ed} \mod n = m^{(ed \mod z)} \mod n$$

And at beginning we choose $eandd$ which satisfies $ed \mod z = 1$, which means:

$$m^{ed} \mod n = m^1 \mod n = m$$

The security of this protocol lies in the fact that finding the inverse of e is a hard problem as one needs to know the factorization of $n = pq$. Which means this asymmetry is based on the practical difficulty of factoring the product of two large prime numbers, the factoring problem. However, if someone were able to find a fast algorithm (which demands the algorithm runs in less than exponential time) then the security of RSA would be lost, and the system would be broken.

RSA is a relatively slow algorithm, and because of this although it is able to directly encrypt user data, it is less commonly used to do such things. More often, RSA passes encrypted shared keys for symmetric key cryptography which in turn can perform bulk encryption-decryption operations at much higher speed. [46]
citeboneh1999twenty

4.2 Discrete Log Based Protocols

4.2.1 The Discrete Logarithm Problem

In mathematics, a discrete logarithm is an integer k solving the equation $b^k = g$, where b and g are elements of a finite group. Discrete logarithms are thus the finite-group-theoretic analogue of ordinary logarithms, which solve the same equation for real numbers b and g, where b is the base of the logarithm and g is the value whose logarithm is being taken.

Thus the discrete logarithm problem is as follows: Give α, β and n. Find an x such that $\alpha^x \equiv \beta \bmod n$. Usually, in cryptosystems anyway, n is a prime number and α is a generator of $\mathbb{Z}_n^*$.

No efficient general method for computing discrete logarithms on conventional computers is known. Several important algorithms in public-key cryptography base their security on the assumption that the discrete logarithm problem over carefully chosen groups has no efficient solution.

4.2.2 Diffie-Hellman Key Exchange

An application of the discrete logarithm is the Diffie-Hellman Key Exchange. This protocol is used to establish a shared secret over an insecure channel. Pick a large prime p and g a generator of $\mathbb{Z}_p^*$. These are assumed to be public. Then:

$$A \text{ : Picks a secret } \alpha$$
$$B \text{ : Picks a secret } \beta$$
$$A \xrightarrow{g^\alpha} B \text{ and B computes } (g^\alpha)^\beta$$
$$B \xrightarrow{g^\beta} A \text{ and A computes } (g^\beta)^\alpha$$

Then both A and B have a shared secret $g^{\alpha\beta}$. How is this safe? What if Eve is watching the wire and intercepting the communication? This is the "DH-Problem":

$$\text{Input: } g^\alpha, g^\beta$$
$$\text{Output: } g^{\alpha\beta}$$

Empirically this seems always to come down to computing the discrete log, which is empirically hard. [47]

4.2.3 ElGamal

The ElGamal encryption protocol is as follows (All computations are done over $\mathbb{Z}_p$ where p is a large prime). The information assumed to be public are: the prime p, a generator g of $\mathbb{Z}_p^*$, $h = g^\alpha$, and random $r \in \mathbb{Z}_p^*$. The information kept private is α, the exponent. Thus, one sees that this depends pretty much entirely on the empirical hardness of the discrete log problem. To summarize, the public key is the collection: (p, g, h) and the private key is α.

Let M be the encoding of the message to encrypt. Then the sender picks the random value r (a different r must be chosen for each message or the system can be hacked easily, thus r is an ephemeral key) and transmits the tuple: $(Mh^r, g^r) = C = (C_1, C_2)$. Then, to decrypt, one sees that $(g^r)^\alpha = (g^\alpha)^r = h^r$. Then the message M can be recovered as $M = C_1(C_2^\alpha)^{-1}$.

If α is compromised, then the system is hacked, but this always seems to come down to solving a discrete log problem (though it hasn't been proven that the two are equivalent).

A cool aspect of ElGamal is the fact that is has the property of being homomorphic. This comes into play during the voting scheme outlined in class.

4.2.4 Elliptic curve cryptography

Elliptic curve cryptography (ECC) is an approach to public-key cryptography based on the algebraic structure of elliptic curves over finite fields. ECC requires smaller keys compared to non-ECC cryptography (based on plain Galois fields) to provide equivalent security.

Elliptic curves are applicable for encryption, digital signatures, pseudo-random generators and other tasks. They are also used in several integer factorization algorithms that have applications in cryptography, such as Lenstra elliptic curve factorization. In field Fp: $r = kq(\mod p)$. It is very hard to calculate k from known r, p, q.

It is also demanded that p is a very big number, however compared with non-ECC discrete logarithm cryptography, the bit of p can be much less, since the discrete logarithm problem based on ECC is much more difficult than discrete logarithm problem based on exponential modulus.

For current cryptographic purposes, an elliptic curve is a plane curve over a finite field (rather than the real numbers) which consists of the points satisfying the equation:

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

along with a distinguished point at infinity, denoted ∞. (The coordinates here are to be chosen from a fixed finite field of characteristic not equal to 2 or 3, or the curve equation will be somewhat more complicated.)

This set together with the group operation of elliptic curves is an Abelian group, with the point at infinity as identity element. The structure of the group is inherited from the divisor group of the underlying algebraic variety.

$$Div^0(E) -> Pic^0(E)$$

4.2.5 ElGamal - Elliptic Curve Style

We want to use ElGamal with elliptic curves. For an element $p \in E$, where E is the graph of an elliptic curve with the standard group structure defined. Then the order of p is the smallest such k such that $kp = \infty$, which is the identity element in E.

In general, E is not cyclic! What we want is a point P on the curve such that $|P|$ = large prime. We then "pretend" that P is a generator. The secret key, like in the classical ElGamal protocol, is a secret scalar α. We then compute $H = \alpha P$. We then encode our message M such that it is a point on the curve E. Then, $encrypt(M) =$ Pick random r and send $(rP, rH + M)$.

To decrypt find $-(rH)$ and then add to the payload.

4.3 Comparison of discrete logarithm with integer factorization

While computing discrete logarithms and factoring integers are distinct problems, they share some common properties:

1) Both problems are difficult (no efficient algorithms are known for non-quantum computers)
2) For both problems efficient algorithms on quantum computers are known
3) Algorithms from one problem can often adapted to the other

4.4 Advanced Cryptographic Engine(ACE Encrypt)

the collection of units, implementing both a public key encryption scheme and a digital signature scheme. Corresponding names for these schemes : "ACE Encrypt" and "ACE Sign". Schemes are based on Cramer-Shoup public key encryption scheme and Cramer-Shoup signature scheme. Introduced variants of these schemes are intended to achieve a good balance between performance and security of the whole encryption system.

5 SIGNATURES

There exists protocols based on the above encryption protocols to digitally sign a message so that the source can be verified or the message authenticated.

5.1 RSA

The message M is public, and I want to fix my signature to it. Let $n = pq$ where p, q are two large, secret, differing primes. Then let e such that $gcd(e, \phi(n)) = 1$, and let d be the inverse of e modulo n. Then $sigRSA(M) = (M, M^d = S)$. Then e is public, so verifying the message is simply $verifyRSA(M) = S^e = M^{ed} = M$.

For long messages, M is broken into blocks of certain lengths. Obviously, one cannot use the same information for both encrypting and signing as the information that should be kept private in one situation is made public in the other, i.e the public and private keys are switched. [43]

If the message has been tampered with, or if the signature has been tampered with, then the verify step will fail to produce the correct message.

5.2 ElGamal

For a large prime p choose a generator of $\mathbb{Z}_p^*$, g. Then α is secret, and compute $H = g^\alpha$ To sign a message M pick a random k and compute $x = g^k$. Then the signature $y = (M - x\alpha)k^{-1}$, so $sigELGAMAL(M) = (M, y)$. The verification step is done as follows:

$$\begin{aligned} ky &= M - x\alpha \in \mathbb{Z}_p^* \\ M &= x\alpha + ky \in \mathbb{Z}_p^* \\ g^M &= g^{x\alpha} g^{ky} \in \mathbb{Z}_p \\ &= H^x x^y \in \mathbb{Z}_p \end{aligned}$$

5.3 Blind Signatures

Suppose someone has to sign a message M, but the message cannot be revealed to this person. Then there is a clever way of doing it. The question is really, how does one get M^d without revealing M?

Chose random r which is kept secret. Then have the person sign Mr^e i.e computes $(Mr^e)^d = M^d r^{rd} = M^d r$. Then the other party just computes r^{-1} and then $M^d r r^{-1} = M^d$. Message signed.

6 CRYPTOGRAPHIC HASH FUNCTIONS

Used in cryptography firstly in late seventies, hash functions have been proven very useful tools to solve security problems. The arise of this method is with the development of authenticity protection. At a time in military world, authenticity protection was provided by the secret key added to the plaintext each of which was used only once; and in banking problems, the message sender would put the transaction totals and a secret key (known by the receiver) into a function to generate a test key. Although both solutions are not suited for a wider and less restrictive environment, they form the embryonal stadium of the concept of hash functions [48]. Three approaches are developed for the research of hash function, respectively based on information theory, complexity theory and system, are included. In the meantime, hash functions are widely used in the Internet security, working together with other technologies like digital signature.

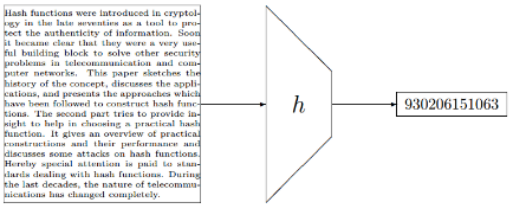


Fig. 14: Basic structure of hash functions

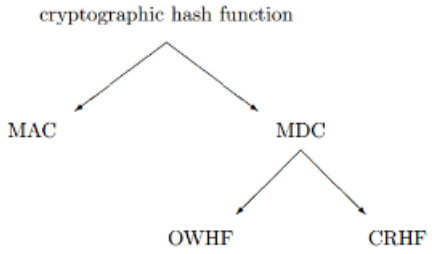


Fig. 15: Taxonomy of hash functions

6.1 Taxonomy: two classes

Cryptographic hash functions can be divided into two main classes. One of them uses secret keys to protect information; such a hash function is called Message Authentication Codes (MAC). On the contrary, a hash function without adding a secret key is called Manipulation Detection Codes (MDC). Furthermore, there are two subclasses of MDC: One-Way Hash Functions (OWHF) and Collision Resistant Hash Functions (CRHF).

In the following the hash function will be denoted with h, and its argument, i.e., the information to be protected with X. The image of X under the hash function h will be denoted with $h(X)$. The secret key will be denoted with$K$.

6.1.1 Message Authentication Code (MAC)

Definition: A MAC is a function satisfying the following conditions:

1) The argument X can be of arbitrary length and the result $h(K, X)$ has a fixed length of n bits (with $n \geq 32...64$).
2) Given h and X, it is hard to determine $h(K, X)$ with a probability of success significantly higher than $1/2^n$. Even when a large number of pairs $(X_i, h(K, X_i))$ are known, where the Xi have been selected by the opponent, it is "hard" to determine the key K or to compute $h(K, X_0)$ for any $X_0 \neq X_i$. This last attack is called an adaptive chosen text attack.

In MAC, the secrecy and authenticity of a short key serves the security and authenticity of the information. The basic idea of MAC protection is to add redundancy to the information. As the definition stressed, a MAC function should be both one-way and collision resistant (if enemy does not know K). It should be noticed that a redundancy can be appended to the hashcode generated by a hash function; although this approach has similarities with MAC, it actually uses a MDC function.

6.1.2 One-way hash function (OWHF)

Definition: A one-way hash function is a function h satisfying the following conditions:

1) The argument X can be of arbitrary length and the result $h(X)$ has a fixed length of n bits (with $n \geq 64$).
2) The hash function must be one-way in the sense that given a Y in the image of h, it is "hard" to find a message X such that $h(X) = Y$, and given X and $h(X)$ it is "hard" to find a message $X_0 \neq X$ such that $h(X_0) = h(X)$.

The last property is a strong condition of the one-way trait of OWHF. It is "hard" to go back to get the information of X given Y. To be specific, producing a (second) X for a certain Y needs 2^n operations.

6.1.3 Collision resistant hash function (CRHF)

Definition: A collision resistant hash function is a function h satisfying the following conditions:

1) The argument$X$ can be of arbitrary length and the result $h(X)$ has a fixed length of n bits (with $n \geq 128$).

2) The hash function must be one-way in the sense that given a Y in the image of$h$, it is "hard" to find a message X such that $h(X) = Y$, and given X and $h(X)$ it is "hard" to find a message $X_0 \neq X$ such that $h(X_0) = h(X)$.
3) The hash function must be collision resistant: this means that it is "hard" to find two distinct messages that hash to the same result.

Although looked similar to the definition of OWHF, CRHF contains a stronger restriction: oother than the properties of OWHF, in CRHF producing a collision requires $O(2^{n/2})$ operations (in the case of "ideal security" [49]). Designing OWHF is easier, and the length of hashcode is shorter (with a smaller space complexity), while CRHF is safer.

6.2 Taxonomy: three approaches

In this section we will follow the the taxonomy for stream ciphers of R. Rueppel [50]. The approaches to research is classified into three kind, each of which will be only introduced briefly.

6.2.1 Information theoretic approach

The characteristic of this approach is unconditionally security, which means the system will not change with the power of the enemys computing, which is also the main advantage of this approach. Its disadvantage is that its key can only be used once (or a finite number of times).

6.2.2 Complexity theoretic approach

The approach taken here sets the model of computation firstly (like a Turing machine [51] or a Boolean circuit) and parameterize them by a security parameter (only algorithms or circuits that require asymptotically polynomial time and space in terms of the size of the input are considered feasible), and then designs cryptographic systems that are provably secure with respect to this model.

6.2.3 System based or practical approach

The efficiency of software and hardware implementations is the major concern in this approach. The goal of this approach is to ensure that breaking a cryptosystem is a difficult problem for the cryptanalyst. In practice it concentrates on the generation of the system and the efficiency against attacks.

6.3 Taxonomy: four generations

6.3.1 Hash functions based on a block cipher

1) Single block hash functions
Hash functions based on a block cipher mainly includes a round function taking one block cipher (of the plaintext) and the output of the last cycle of the system as input until the last block is used.

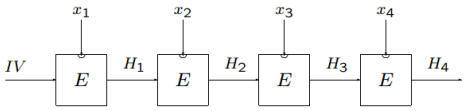


Fig. 16: Process of hash functions based on a block cipher [52]

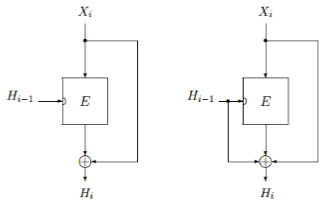


Fig. 17: The round function of the scheme by Matyas et al. [53] (left) and of the scheme by Preneel et al. and Miyaguchi et al. [54](right).

Generating hash functions in this way is often easy to design and trustworthy.
$B_{k,n}$: set of all block ciphers with k-bit keys and n-bit blocks
the cardinality of this set is:
$|B_{k,n}| = \binom{2^n!}{2^k}$

an ideal (block) cipher is a block cipher selected according to the uniform distribution from the set $B_{k,n}$ [55] [56].

However, it is also a slow and export restricted approach. If it takes 2^s steps to invert the compression function, finding a second preimage requires $2^{1+(n+s)/2}$ steps [57].

2) Double block hash functions
This type of functions has been proposed to construct a collision resistant hash function based on a block cipher with a block length of 64 bits. As a representative, MDC-2 proposed by B. Brachtletal [58], also known as the Meyer-Schilling hash functions,after the two co-authors who published them at Securicom88 [59], can be described as follows:

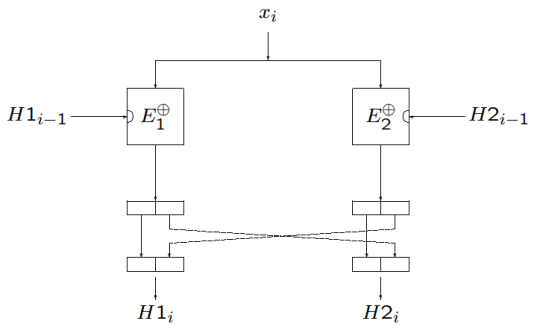


Fig. 18: The round function of the MDC-2 hash function

$T1_i = E \oplus (H1_{i?1}, X_i) = LT1_i||RT1_i$
$T2_i = E \oplus (H2_{i?1}, X_i) = LT2_i||RT2_i$
$H1_i = LT1_i||RT2_i$
$H2_i = LT2_i||RT1_i$

Here $H1_0$ and $H2_0$ are initialized with IV_1 and IV_2 respectively, and the hashcode is equal to $H1_t||H2_t$. In order to protect these schemes against attacks based on semi-(weak) keys [60] the second and third key bits are fixed to 10 and 01 for the first and second encryption.

6.3.2 Hash functions based on modular arithmetic

Generating hash functions by modular arithmetic means using the information mod some (certain) number to decide the hashcode. Their security is uncertain (influenced by the hardness of certain number theoretic problems), so is their time consumption. They are vulnerable to trapdoors.
Through its development history, the algebraic structures experienced many changes: from a small modulus to a large modulus, and then to modular squaring to develop a very efficient scheme. It is in such a form:
$f = (X_i \oplus H_{i?1}^2 mod N \oplus H_i$
To make the system less vulnerable to attacks, stronger schemes have been proposed. One example is that using two levels of squaring:
$f = (H_{i?1} \oplus (X_i)^2)^2 mod N$
and/or increase the exponent number. These schemes are apparently much safer, but the sophistication of them makes them less efficient. [48]

6.3.3 Hash functions based on a knapsack

knapsack problem of dimensions n and $l(n)$:
given a set of nl-bit integers $a_1, a_2, ..., a_n$, and an l-bit integer S
find a vector X with components x_i equal to 0 or 1 such that:
$\sum_{i=1}^{n} a_i * x_i = S mod 2^{l(n)}$
for hashing, one needs $n\ l(n)$.

6.3.4 Dedicated hash functions

Except what have been illustrated above, some dedicated hash functions are also used. MD2(Md4,MD5...), FFT-Hash and Snefru are some instances, but here we will not introduce them in detail.

6.4 Attacks on hash functions and the security requisites

6.4.1 Random attack

The opponent selects a random message and hopes that the change will remain undetected. In case of a good hash function, his probability of success equals $1/2^n$ with n the number of bits of the hashcode. The feasibility of this attack depends on the action taken in case of detection of an erroneous result, on the expected value of a successful attack, and on the number of attacks that can be carried out. For most application this implies that $n = 32$ bits is not sufficient

6.4.2 Birthday attack

This attack can only be used to produce collisions. The idea behind thebirthday attack [61] is that for a group of 23 people the probability that at least two people have a common birthday exceeds $1/2$. Intuitively one would expect that the group should be significantly larger. This can be exploited to attack a hash function in the following way: an adversary generates $r1$ variations on a bogus message and $r2$ variations on a genuine message. The probability of finding a bogus message and a genuine message that hash to the same result is given by:
$1 - exp(-\frac{r_1 * r_2}{2^n})$
which is about 63% when $r = r1 = r2 = 2^{n/2}$. Note that in case of a MAC the opponent is unable to generate the MAC of a message. He could however obtain these MACs with a chosen plaintext attack. A second possibility is that he collects a large number of messages and corresponding MACs and divides them into two categories, which corresponds to a known plaintext attack. The involved comparison problem does not require r^2 operations: after sorting the data, which requires $O(r\log r)$ operations, comparison is easy. Jueneman has shown in 1986 [62] that for $n = 64$ the processing and storage requirements were feasible in reasonable time with the computer power available in every large organization. A timememory-processor trade-off is possible.
If the function can be called as a black box, one can use the collision search algorithm proposed by J.-J. Quisquater [63], that requires about $2\sqrt{\pi/2} * 2^{n/2}$ operations and negligible storage. To avoid this attack with a reasonable safety margin, n should be at least 128 bits. This explains the second condition in Definition 2 of a CRHF.

6.4.3 Exhaustive key search

This attack is only relevant in case of a MAC. It is a known plaintext attack, where an attacker knows$M$plaintext-MAC pairs for a given key and will try to determine the key by trying all possible keys. The expected number of trials equals $2^{k?1}$, with k the size of the key in bits. In order to determine the key uniquely, M has to be slightly larger than k/n.

7 BRANCHES OF CRYPTOGRAPHY

Here contains some mew directions and some branches which are not very widely used or studied up to now.

7.1 Quantum cryptography

Quantum cryptography is the science of exploiting quantum mechanical properties to perform cryptographic tasks.

Since currently used popular public-key encryption and signature schemes (e.g., RSA and ElGamal) are based on the practical difficulty of math problems whose attacker need to a lot of computation which is nearly impossible by classical computers, they can be broken by quantum adversaries. The advantage of quantum cryptography lies in the fact that it allows the completion of various cryptographic tasks that are proven or conjectured to be impossible using only classical (i.e. non-quantum) communication. For example, it is impossible to copy data encoded in a quantum state and the very act of reading data encoded in a quantum state changes the state. This is used to detect eavesdropping in quantum key distribution. [64] [65]

The most well known and developed application of quantum cryptography is quantum key distribution (QKD), which is the process of using quantum communication to establish a shared key between two parties (Alice and Bob, for example) without a third party (Eve) learning anything about that key, even if Eve can eavesdrop on all communication between Alice and Bob. This is achieved by Alice encoding the bits of the key as quantum data and sending them to Bob; if Eve tries to learn these bits, the messages will be disturbed and Alice and Bob will notice. The key is then typically used for encrypted communication using classical techniques. For instance, the exchanged key could be used as the seed of the same random number generator both by Alice and Bob.

The security of QKD can be proven mathematically without imposing any restrictions on the abilities of an eavesdropper, something not possible with classical key distribution. This is usually described as ”unconditional security”, although there are some minimal assumptions required including that the laws of quantum mechanics apply and that Alice and Bob are able to authenticate each other, i.e. Eve should not be able to impersonate Alice or Bob as otherwise a man-in-the-middle attack would be possible.

One aspect of QKD is that it is secure against quantum computers, as its strength does not depend on mathematical complexity, like post-quantum cryptography, but on physical principles. [66]

7.2 DNA based cryptography

In secure encryption schemes, the legitimate user is able to decipher the messages (using some private information available), yet for an adversary (not having this private information) the task of decrypting the cipher text (i.e., breaking" the encryption) should be infeasible. But today, the breaking task can be performed by a non-deterministic polynomial-time machine.

DNA based cryptography is based on DNA computing skills, which are very competitive computing skills compared with traditional computers.

Research work is being done on DNA Computing either using test tubes (biologically) or simulating the operations of DNA using computers (Pseudo or Virtual DNA computing). Gehani et. al., introduced the first trial of DNA based Cryptography in which a substitution method using libraries of distinct one time pads, each of which defines a specific, randomly generated, pair-wise mapping and an XOR scheme utilizing molecular computation and indexed, random key strings are used for encryption. [67] [68]

After that, many other algorithms on DNA based cryptography has been developed, which had used plenty of different technology, like: One-Time-Pad (OTP), DNA XOR OTP and DNA chromosomes indexing [69],DNA digital coding PCR primers - A message is converted to DNA template in which primers are used as key to encode and decode the message [70],DNA binary strands-Molecular checksum, PCR, gel electrophoresis [71],Transcription, Splicing, Translation -mRNA form of data into protein according to genetic code table and key send to the receiver in a secure channel [72],DNA Sequence Addition Operation DNA sequence Matrix, DNA sequence addition using Logistic maps and complementarity [73],DNA sequence matching - data converted into pointers according to DNA strand taken and key send to the receiver in a secure channel [74],Base triplet substitution and DNA binary strands [75]

DNA binary strands support feasibility and applicability of DNA-based Cryptography. The security and the performance of the DNA based cryptographic algorithms are satisfactory for multi-level security applications of todays network. Certain DNA algorithms can resist exhaustive attack, statistical attack and differential attack.

The field of DNA computing is still in its infancy and the applications for this technology have not yet been fully understood. DNA computing is viable and DNA authentication methods have shown great promise in the marketplace of today and it is hoped that its applications will continue to expand. DNA Cipher is the beneficial supplement to the existing mathematical cipher. If the molecular word can be controlled at will, it may be possible to achieve vastly better performance for information storage and security. [76]

7.3 Visual cryptography

The possibility of integrating human visual intelligence into the process of encrypting sensitive information by presenting certain visual information to the recipient's eye is discussed. This adds a new dimension to the cryptocomplexity of such a process.

Two implementations are based on this principle are described. [77]

The first shows how keys used for encryption can be randomly generated by the transmitter, without the necessity of exchanging them with the legitimate recipient. The keys are 'embedded' in a master key and are recovered from it by the intelligence of the legitimate recipient after he or she uses the master key. No human intelligence can be helpful to a user who does not possess the master key. The second implementation concerns the possibility of creating a secret connection between a numerical key and a specific image (e.g. a face). Such a scheme can be used, for example, in validating the identity of the users of credit cards.

7.4 Network steganography

All information hiding techniques that may be used to exchange steganograms in telecommunication networks can be classified under the general term of network steganography. This nomenclature was originally introduced by Krzysztof Szczypiorski in 2003. [78]

Contrary to typical steganographic methods that use digital media (images, audio and video files) to hide data, network steganography uses communication protocols' control elements and their intrinsic functionality. As a result, such methods are harder to detect and eliminate.

Typical network steganography methods involve modification of the properties of a single network protocol. Such modification can be applied to the PDU (Protocol Data Unit),to the time relations between the exchanged PDUs [79], or both (hybrid methods).

Moreover, it is feasible to utilize the relation between two or more different network protocols to enable secret communication. These applications fall under the term inter-protocol steganography.

Network steganography covers a broad spectrum of techniques, which include, among others:

Steganophony : the concealment of messages in Voice-over-IP conversations, e.g. the employment of delayed or corrupted packets that would normally be ignored by the receiver (this method is called LACK : Lost Audio Packets Steganography), or, alternatively, hiding information in unused header fields. WLAN Steganography : transmission of steganograms in Wireless Local Area Networks. A practical example of WLAN Steganography is the HICCUPS system (Hidden Communication System for Corrupted Networks)

8 Conclusion

The conclusion goes here.

References

[1] A. K. Ekert, "Quantum cryptography based on bells theorem," *Physical review letters*, vol. 67, no. 6, p. 661, 1991.
[2] M. Bellare and P. Rogaway, "Introduction to modern cryptography," *UCSD CSE*, vol. 207, p. 207, 2005.
[3] A. J. Menezes, P. C. Van Oorschot, and S. A. Vanstone, *Handbook of applied cryptography*. CRC press, 1996.
[4] W. Diffie and M. E. Hellman, "Privacy and authentication: An introduction to cryptography," *Proceedings of the IEEE*, vol. 67, no. 3, pp. 397–427, 1979.
[5] B.-J. Koops, "Overview per country," *Crypto Law Survey*, 1999.
[6] Y. Akdeniz, N. Taylor, and C. Walker, "Regulation of investigatory powers act 2000 (1): Bigbrother. gov. uk: State surveillance in the age of information and rights,[2001]," *Criminal Law Review*, pp. 73–90, 2001.
[7] T. Engelhardt, *Shadow government: Surveillance, secret wars, and a global security state in a single-superpower world*. Haymarket Books, 2014.
[8] M. A. C. Dizon, "Participatory democracy and information and communications technology: A legal pluralist perspective," *European Journal of Law and Technology*, vol. 1, no. 3, 2010.
[9] W. Diffie and M. E. Hellman, "New directions in cryptography," *Information Theory, IEEE Transactions on*, vol. 22, no. 6, pp. 644–654, 1976.
[10] M. Blaze, W. Diffie, R. L. Rivest, B. Schneier, and T. Shimomura, "Minimal key lengths for symmetric ciphers to provide adequate commercial security. a report by an ad hoc group of cryptographers and computer scientists," tech. rep., DTIC Document, 1996.
[11] A. E. Standard, "Federal information processing standards publication 197," *FIPS PUB*, pp. 46–3, 2001.
[12] A. Saunders, M. M. Cornett, and P. A. McGraw, *Financial institutions management: A risk management approach*, vol. 8. McGraw-Hill/Irwin, 2006.
[13] J. Callas, L. Donnerhacke, H. Finney, and R. Thayer, "Rfc 2440: Openpgp message format," *Status: PROPOSED STANDARD*, 1998.
[14] B. Odiyo and M. Dwarkanath, "Virtual private network," *Uppsala universitet (accessed on November 2011)*, 2011.
[15] B. Schneier, "Applied cryptography, 1996," *Cover and title pages*, pp. 125–147, 1997.
[16] M. Luby and C. Rackoff, "How to construct pseudorandom permutations from pseudorandom functions," *SIAM Journal on Computing*, vol. 17, no. 2, pp. 373–386, 1988.
[17] J. Patarin, "Luby-rackoff: 7 rounds are enough for 2 n (1- ε) security," in *Advances in Cryptology-CRYPTO 2003*, pp. 513–529, Springer, 2003.
[18] http://www.schneier.com/paper-unbalanced-feistel.html.
[19] S. Bono, M. Green, A. Stubblefield, A. Juels, A. D. Rubin, and M. Szydlo, "Security analysis of a cryptographically-enabled rfid device.," in *USENIX Security*, vol. 5, pp. 1–16, 2005.
[20] B. Morris, P. Rogaway, and T. Stegers, "How to encipher messages on a small domain," in *Advances in Cryptology-CRYPTO 2009*, pp. 286–302, Springer, 2009.
[21] M. Rouse, "Block cipher." http://www.searchsecurity.techtarget.com/definition/block-ciphe, 2015.
[22] J. Daemen and V. Rijmen, "Aes proposal: Rijndael," 1999.
[23] N.-F. Standard, "Announcing the advanced encryption standard (aes)," *Federal Information Processing Standards Publication*, vol. 197, pp. 1–51, 2001.
[24] J. Schwartz, "Us selects a new encryption technique," *New York Times*, vol. 3, 2000.
[25] B. Schneier, J. Kelsey, D. Whiting, D. Wagner, C. Hall, N. Ferguson, T. Kohno, and M. Stay, "The twofish teams final comments on aes selection," *AES round*, vol. 2, 2000.
[26] G. Bertoni, L. Breveglieri, P. Fragneto, M. Macchetti, and S. Marchesin, "Efficient software implementation of aes on 32-bit platforms," in *Cryptographic Hardware and Embedded Systems-CHES 2002*, pp. 159–171, Springer, 2002.
[27] L. Hathaway, "National policy on the use of the advanced encryption standard (aes) to protect national security systems and national security information," *National Security Agency*, vol. 23, 2003.
[28] N. Ferguson, J. Kelsey, S. Lucks, B. Schneier, M. Stay, D. Wagner, and D. Whiting, "Improved cryptanalysis of rijndael," in *Fast software encryption*, pp. 213–230, Springer, 2000.
[29] B. Schneier, "Description of a new variable-length key, 64-bit block cipher (blowfish)," in *Fast Software Encryption*, pp. 191–204, Springer, 1993.
[30] T. R. Johnson, "American cryptology during the cold war, 1945-1989. book iii: Retrenchment and reform, 1972-1989. fort meade, md.: National security agency," *Center for Cryptologic History. Available at http://www. nsa. gov/publicJnfo/_files/cryptologic_ histories/cold_warJii. pdf*, 1998.
[31] C. Pracana, "International psychological applications conference and trends (inpact) proceedings (ljubljana, slovenia, may 2-4, 2015).," *Online Submission*, 2015.
[32] P. FIPS, "81, des modes of operation," *Issued December*, vol. 2, p. 63, 1980.
[33] A. M. Comeau, F. J. Accurso, T. B. White, P. W. Campbell, G. Hoffman, R. B. Parad, B. S. Wilfond, M. Rosenfeld, M. K. Sontag, J. Massie, *et al.*, "Guidelines for implementation of cystic fibrosis newborn screening programs: Cystic fibrosis foundation workshop report," *Pediatrics*, vol. 119, no. 2, pp. e495–e518, 2007.
[34] V. Shoup, "Information technology-security techniques-encryption algorithms-part 2: Asymmetric ciphers," *ISO/IEC 18033-2*, 2004.
[35] E. Biham and O. Dunkelman, "Cryptanalysis of the a5/1 gsm stream cipher," in *Progress in CryptologyłINDOCRYPT 2000*, pp. 43–51, Springer, 2000.
[36] J. Quirke, "Security in the gsm system," *AusMobile, May*, pp. 1–26, 2004.
[37] D. E. Knuth, *The art of computer programming: sorting and searching*, vol. 3. Pearson Education, 1998.
[38] U. Blöcher and M. Dichtl, "Fish: A fast software stream cipher," in *Fast Software Encryption*, pp. 41–44, Springer, 1993.
[39] R. J. Jenkins Jr, "Isaac," in *Fast Software Encryption*, pp. 41–49, Springer, 1996.
[40] M. Blum and S. Micali, "How to generate cryptographically strong sequences of pseudorandom bits," *SIAM journal on Computing*, vol. 13, no. 4, pp. 850–864, 1984.
[41] A. C. Yao, "Theory and application of trapdoor functions," in *Foundations of Computer Science, 1982. SFCS'08. 23rd Annual Symposium on*, pp. 80–91, IEEE, 1982.
[42] M. Pudovkina, "A known plaintext attack on the isaac keystream generator.," *IACR Cryptology ePrint Archive*, vol. 2001, p. 49, 2001.
[43] N. Ferguson and B. Schneier, *Practical cryptography*, vol. 23. Wiley New York, 2003.
[44] T. Kleinjung, K. Aoki, J. Franke, A. K. Lenstra, E. Thomé, J. W. Bos, P. Gaudry, A. Kruppa, P. L. Montgomery, D. A. Osvik, *et al.*, "Factorization of a 768-bit rsa modulus," in *Advances in Cryptology–CRYPTO 2010*, pp. 333–350, Springer, 2010.
[45] P. W. Shor, "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer," *SIAM review*, vol. 41, no. 2, pp. 303–332, 1999.
[46] J. F. Kurose, *Computer Networking: A Top-Down Approach Featuring the Internet, 3/E*. Pearson Education India, 2005.
[47] D. Adrian, K. Bhargavan, Z. Durumeric, P. Gaudry, M. Green, J. A. Halderman, N. Heninger, D. Springall, E. Thomé, L. Valenta, *et al.*, "Imperfect forward secrecy: How diffie-hellman fails in practice," in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pp. 5–17, ACM, 2015.

[48] S. on State, R. Progress of Research in Cryptography (3, 1993, and W. Wolfowicz, *Proceedings of the 3rd Symposium on State and Progress of Research in Cryptography: SPRC'93, Rome, Italy-February 15-16, 1993*. 1993.
[49] X. Lai and J. L. Massey, "Hash functions based on block ciphers," in *Advances in CryptologyłEUROCRYPT92*, pp. 55–70, Springer, 1992.
[50] G. J. Simmons, *Contemporary cryptology: The science of information integrity*. IEEE press, 1994.
[51] B. E_K, "The design and analysis of computer algorithms," 1974.
[52] F. L. Bauer and R. Steinbrüggen, *Foundations of secure computation*, vol. 175. Ios Press, 2000.
[53] S. M. Matyas, C. H. Meyer, and J. Oseas, "Generating strong one-way functions with cryptographic algorithm," *IBM Technical Disclosure Bulletin*, vol. 27, no. 10A, pp. 5658–5659, 1985.
[54] S. Miyaguchi, M. Iwata, and K. Ohta, "New 128-bit hash function," in *Proc. 4th International Joint Workshop on Computer Communications, Tokyo, Japan*, pp. 279–288, 1989.
[55] R. S. Winternitz, "Producing a one-way hash function from des," in *Advances in Cryptology*, pp. 203–207, Springer, 1984.
[56] J. Black, P. Rogaway, and T. Shrimpton, "Black-box analysis of the block-cipher-based hash-function constructions from pgv," in *Advances in CryptologyłCRYPTO 2002*, pp. 320–335, Springer, 2002.
[57] X. Lai, *On the design and security of block ciphers*. PhD thesis, Diss. Techn. Wiss ETH Zürich, Nr. 9752, 1992. Ref.: JL Massey; Korref.: H. Bühlmann, 1992.
[58] B. O. Brachtl, D. Coppersmith, M. M. Hyden, S. M. Matyas Jr, C. H. Meyer, J. Oseas, S. Pilpel, and M. Schilling, "Data authentication using modification detection codes based on a public one way encryption function," Mar. 13 1990. US Patent 4,908,861.
[59] C. H. Meyer and M. Schilling, "Secure program load with manipulation detection code," in *Proc. Securicom*, vol. 88, pp. 111–130, 1988.
[60] J. H. Moore and G. J. Simmons, "Cycle structure of the des for keys having palindromic (or antipalindromic) sequences of round keys," *IEEE Transactions on Software Engineering*, vol. 13, no. 2, p. 262, 1987.
[61] G. Yuval, "How to swindle rabin," *Cryptologia*, vol. 3, no. 3, pp. 187–191, 1979.
[62] R. R. Jueneman, "A high speed manipulation detection code," in *Advances in CryptologyłCRYPTO86*, pp. 327–346, Springer, 1986.
[63] J.-J. Quisquater and J.-P. Delescaille, "How easy is collision search? application to des," in *Advances in CryptologyłEUROCRYPT89*, pp. 429–434, Springer, 1989.
[64] M. S. Sharbaf, "Quantum cryptography: An emerging technology in network security," in *Technologies for Homeland Security (HST), 2011 IEEE International Conference on*, pp. 13–19, IEEE, 2011.
[65] N. Gisin, G. Ribordy, W. Tittel, and H. Zbinden, "Quantum cryptography," *Reviews of modern physics*, vol. 74, no. 1, p. 145, 2002.
[66] M. S. Sharbaf, "Quantum cryptography: a new generation of information technology security system," in *Information Technology: New Generations, 2009. ITNG'09. Sixth International Conference on*, pp. 1644–1648, IEEE, 2009.
[67] A. Gehani, T. LaBean, and J. Reif, "Dna-based cryptography," in *Aspects of Molecular Computing*, pp. 167–188, Springer, 2003.
[68] G. Xiao, M. Lu, L. Qin, and X. Lai, "New field of cryptography: Dna cryptography," *Chinese Science Bulletin*, vol. 51, no. 12, pp. 1413–1420, 2006.
[69] M. Borda and O. Tornea, "Dna secret writing techniques," in *IEEE conference*, 2010.
[70] G. Cui, L. Qin, Y. Wang, and X. Zhang, "An encryption scheme using dna technology," in *Bio-Inspired Computing: Theories and Applications, 2008. BICTA 2008. 3rd International Conference on*, pp. 37–42, IEEE, 2008.
[71] A. Leier, C. Richter, W. Banzhaf, and H. Rauhe, "Cryptography with dna binary strands," *Biosystems*, vol. 57, no. 1, pp. 13–22, 2000.
[72] K. Ning, "A pseudo dna cryptography method," *arXiv preprint arXiv:0903.2693*, 2009.
[73] Q. Zhang, L. Guo, X. Xue, and X. Wei, "An image encryption algorithm based on dna sequence addition operation," in *Bio-Inspired Computing, 2009. BIC-TA'09. Fourth International Conference on*, pp. 1–5, Ieee, 2009.
[74] S. T. Amin, M. Saeb, and S. El-Gindi, "A dna-based implementation of yaea encryption algorithm.," in *Computational Intelligence*, pp. 120–125, 2006.
[75] C. T. Clelland, V. Risca, and C. Bancroft, "Hiding messages in dna microdots," *Nature*, vol. 399, no. 6736, pp. 533–534, 1999.
[76] G. Jacob, "Dna based cryptography: An overview and analysis," *International Journal of Emerging Sciences*, vol. 3, no. 1, p. 36, 2013.
[77] B. Arazi, I. H. Dinstein, and O. Kafri, "Intuition, perception, and secure communication," *Systems, Man and Cybernetics, IEEE Transactions on*, vol. 19, no. 5, pp. 1016–1020, 1989.
[78] K. Szczypiorski, "Steganography in tcp/ip networks," in *State of the Art and a Proposal of a New System–HICCUPS, Institute of Telecommunications' seminar, Warsaw University of Technology, Poland*, Citeseer, 2003.
[79] K. Ahsan and D. Kundur, "Practical data hiding in tcp/ip," in *Proc. Workshop on Multimedia Security at ACM Multimedia*, vol. 2, 2002.