Block and Stream Ciphers

Stjepan Picek; TU Delft, The Netherlands

Faculty of Electrical Engineering and Computing, Zagreb

Outline

Outline

Basics

- Confusion the ciphertext statistics should depend on the plaintext statistics in a manner too complicated to be exploited by the cryptanalyst.
- Diffusion - each digit of the plaintext and each digit of the secret key should influence many digits of the ciphertext.
- A block cipher should behave like a random permutation.
- Product ciphers.
- Round, network.

Substitution-Permutation Networks, SPN

- The "substitution" portion refers to small random functions that is actually a random 1-1 and onto function of the alphabet, and "permutation" refers to the mixing of the outputs of the random functions.
- Key can specify S-boxes and permutations or key can be mixed into computation.
- Each cipher consists of a number of rounds.
- Each round consists of XOR with the key, S-box layer, and mixing layer.

Substitution-Permutation Networks, SPN

- Important: for SPNs, S-boxes need to be invertible.
- In a substitution-permutation network F in which the S-boxes are all 1-1 and onto (and of polynomial-size), there exists an efficient procedure for computing $F^{-1}(y)$. Furthermore, for every key k and every input x, $F_k^{-1}(F_k(x)) = x$.
- Important: avalanche effect.
- The S-boxes are designed so that any change of at least a single bit to the input to an S-box results in a change of at least two bits in the output.
- The mixing permutations are designed so that the output bits of any given S-box are spread into different S-boxes in the next round.

Substitution-Permutation Networks, SPN

- Pseudorandomness of substitution-permutation networks: there is no formal justification for why such a design yields a pseudorandom permutation.

Feistel Networks

- An alternative way of constructing a block cipher.
- The low-level building blocks (S-boxes, mixing permutations and key schedule) are the same; the difference is in the high-level design.
- The advantage of Feistel networks over substitution permutation networks is that they enable the use of S-boxes that are not necessarily invertible.
- A Feistel network is thus a way of constructing an invertible function from non-invertible components.

Feistel Networks

- A Feistel network refers to an internal f-function that does not need to be invertible.
- This function receives a subkey and typically contains components like S-boxes and mixing permutations.
- The framework of a Feistel network can deal with any internal f-function, irrespective of its design.
- The input x to a Feistel network is separated into two halves, x_1 and x_2, and each half is passed separately through the f-function.

Outline

DES

- Developed in IBM in 1974 (based on cipher Lucifer).
- Lucifer is a Feistel cipher which encrypts blocks of 64 bits using a key size of 128 bits.
- DES is insecure but still used in legacy applications.
- Already broken in 1998: DES Challenge II: a computer worth 250k USD that breaks DES message in less than 3 days.
- Biggest problem is the small key size.
- 3DES is secure cipher.
- Since DES is by far the best-studied symmetric algorithm, its design principles have inspired many current ciphers.
- Data Encryption Standard (FIPS PUB 46) in 1977.

DES Operations

- For each block of plaintext, encryption is handled in 16 rounds which all perform the identical operation.
- In every round a different subkey is used and all subkeys k_i are derived from the main key k.

DES Operations

- After the initial bitwise permutation IP of a 64-bit plaintext x, the plaintext is split into two halves L_0 and R_0.
- These two 32-bit halves are the input to the Feistel network, which consists of 16 rounds.
- The right half R_i is fed into the function f.
- The output of the f function is XORed with the left 32-bit half L_i.
- Finally, the right and left half are swapped.

DES Rounds

- This process repeats in the next round and can be expressed as:

$$L_i = R_{i-1} \tag{1}$$

$$R_i = L_{i-1} \oplus f(R_{i-1}, k_i). \tag{2}$$

- After round 16, the 32-bit halves L_{16} and R_{16} are swapped again, and the final permutation IP^1 is the last operation of DES.

DES Rounds

- The Feistel structure really only encrypts (decrypts) half of the input bits per each round, namely the left half of the input.
- The right half is copied to the next round unchanged.
- In particular, the right half is not encrypted with the f function.

S-boxes Design Principles

- Each S-box has six input bits and four output bits.
- No single output bit should be too close to a linear combination of the input bits.
- If the lowest and the highest bits of the input are fixed and the four middle bits are varied, each of the possible 4-bit output values must occur exactly once.
- If two inputs to an S-box differ in exactly one bit, their outputs must differ in at least two bits.

S-boxes Design Principles

- If two inputs to an S-box differ in the two middle bits, their outputs must differ in at least two bits.
- If two inputs to an S-box differ in their first two bits and are identical in their last two bits, the two outputs must be different.
- For any nonzero 6-bit difference between inputs, no more than 8 of the 32 pairs of inputs exhibiting that difference may result in the same output difference.
- A collision (zero output difference) at the 32-bit output of the eight S-boxes is only possible for three adjacent S-boxes.

Key Schedule

- The key schedule derives 16 round keys k_i, each consisting of 48 bits, from the original 56-bit key.
- First, note that the DES input key is often stated as 64-bit, where every eighth bit is used as an odd parity bit over the preceding seven bits.
- The eight parity bits are not actual key bits and do not increase the security.
- DES is a 56-bit cipher, not a 64-bit one.

Decryption

- Essentially the same function as encryption.
- Compared to encryption, only the key schedule is reversed.
- The basic idea is that the decryption function reverses the DES encryption in a round-by-round manner.
- That means that decryption round 1 reverses encryption round 16, decryption round 2 reverses encryption round 15, etc.

Outline

Internatonal Data Encription Algorithm, IDEA

- Finished in 1992.
- Secure.
- Key 128 bits, blocks of 64 bits.
- Encryption in 9 (8.5) steps.
- In total, 52 subkeys used.
- Lai-Massey scheme.

Outline

NIST Competition

- On January 2, 1997, NIST announced that they wished to choose a successor to DES to be known as AES.
- Like DES, this was to be "an unclassified, publicly disclosed encryption algorithm capable of protecting sensitive government information well into the next century."
- NIST asked for input from interested parties on how the successor should be chosen.
- Block cipher with 128 bit block size.
- Three key lengths must be supported: 128, 192 and 256 bit.
- Security relative to other submitted algorithms.
- Efficiency in software and hardware.

NIST Competition

- 15 candidates: CAST-256, CRYPTON, DEAL, DFC, E2, FROG, HPC, LOKI97, MAGENTA, MARS, RC6, Rijndael, SAFER+, Serpent, and Twofish.
- AES finalists: Rijndael: 86 positive, 10 negative
  Serpent: 59 positive, 7 negative
  Twofish: 31 positive, 21 negative
  RC6: 23 positive, 37 negative
  MARS: 13 positive, 84 negative

AES Number of Steps

- Key size 128: 10 rounds.
- Key size 192: 12 rounds.
- key size 256: 14 rounds.

Decryption

- All layers need to be inverted.
- Reversed key schedule.
- Since the last encryption round does not perform the MixCol operation, the first decryption round also does not contain the corresponding inverse layer.

Outline

Basics

- We can divide symmetric key cryptography into block and stream ciphers if we consider encryption primitives.
- But we can also consider block and stream ciphers as parts of some other cryptographic mechanisms.
- Then, we can divide symmetric key cryptography into primitives and modes.
- Modes of operation provide a way to encrypt messages of arbitrary length using block or stream ciphers

Basics

- Modes of operation can be divided into confidentiality modes, authenticity modes and authenticated-encryption modes.

Basics

- After the AES selection process, the US National Institute of Standards and Technology (NIST) supported the process of evaluating new modes of operations in a series of special publications and workshops.
- Currently, there are eight approved block cipher modes: five for confidentiality (ECB, CBC, CFB, OFB, CTR), one for authentication (CMAC) and two combined modes for confidentiality and authentication (CCM, GCM).

Authenticity Modes

- When block cipher should be used as MAC.
- CBC-MAC, OMAC, CMAC.

Authenticated-encryption modes

- A mode of operation that adds the authentication feature to a block algorithm.
- The Galois Countermode (GCM).

Confidentiality Modes

- How to encrypt long plaintexts with a block cipher.
- Electronic Code Book mode (ECB),
- Cipher Block Chaining mode (CBC),
- Cipher Feedback mode (CFB),
- Output Feedback mode (OFB),
- Counter mode (CTR).

Confidentiality Modes

- All of the five modes have one goal: they encrypt data and thus provide confidentiality for a message sent from Alice to Bob.
- The ECB and CFB modes require that the length of the plaintext be an exact multiple of the block size of the cipher used, e.g., a multiple of 16 bytes in the case of AES.
- If the plaintext does not have this length, it must be padded.
- There are several ways of doing this padding in practice.

Confidentiality Modes

- One possible padding method is to append a single “1” bit to the plaintext and then to append as many “0” bits as necessary to reach a multiple of the block length.
- Should the plaintext be an exact multiple of the block length, an extra block consisting only of padding bits is appended.
- The latter three modes use the block cipher as a building block for a stream cipher.

ECB

- The most straightforward way of encrypting a message.
- Let us assume that the block cipher encrypts (decrypts) blocks of size b bits.
- Messages which exceed b bits are partitioned into b-bit blocks. If the length of the message is not a multiple of b bits, it must be padded to a multiple of b bits prior to encryption.

ECB Advantages

- Block synchronization between the encryption and decryption parties Alice and Bob is not necessary, i.e., if the receiver does not receive all encrypted blocks due to transmission problems, it is still possible to decrypt the received blocks.
- Bit errors, e.g., caused by noisy transmission lines, only affect the corresponding block but not succeeding blocks.
- Block ciphers operating in ECB mode can be parallelized, e.g., one encryption unit encrypts (or decrypts) block 1, the next one block 2, and so on.

ECB Disadvantages

- The main problem of the ECB mode is that it encrypts highly deterministically.
- Identical plaintext blocks result in identical ciphertext blocks, as long as the key does not change.
- An attacker can recognize if the same message has been sent twice simply by looking at the ciphertext.
- If an attacker reorders the ciphertext blocks, this might result in valid plaintext and the reordering might not be detected.

CBC

- There are two main ideas behind the Cipher Block Chaining (CBC) mode.
- First, the encryption of all blocks are "chained together" such that ciphertext y_i depends not only on block x_i but on all previous plaintext blocks as well.
- Second, the encryption is randomized by using an initialization vector (IV).

CBC

- The ciphertext y_i, which is the result of the encryption of plaintext block x_i, is fed back to the cipher input and XORed with the succeeding plaintext block x_{i+1}.
- This XOR sum is then encrypted, yielding the next ciphertext y_{i+1}, which can then be used for encrypting x_{i+2}, etc.
- For the first plaintext block x_1 there is no previous ciphertext.
- For this an IV is added to the first plaintext, which also allows us to make each CBC encryption nondeterministic.

CBC

- The first ciphertext y_1 depends on plaintext x_1 (and the IV).
- The second ciphertext depends on the IV, x_1 and x_2.
- The last ciphertext is a function of all plaintext blocks and the IV.

CBC

- When decrypting a ciphertext block y_i in CBC mode, we have to reverse the two operations we have done on the encryption side.
- First, we have to reverse the block cipher encryption by applying the decryption function.
- After this we have to undo the XOR operation by again XORing the correct ciphertext block.

CBC

- If we choose a new IV every time we encrypt, the CBC mode becomes a probabilistic encryption scheme.
- If we encrypt a string of blocks $x_1, \ldots, x_t$ once with a first IV and a second time with a different IV, the two resulting ciphertext sequences look completely unrelated to each other for an attacker.
- We do not have to keep the IV secret.
- In most cases, we want the IV to be a nonce, i.e., a number used only once.

OFB

- In the Output Feedback (OFB) mode a block cipher is used to build a stream cipher encryption scheme.
- Note that in OFB mode the key stream is not generated bitwise but instead in a blockwise fashion.
- The output of the cipher gives us b key stream bits, where b is the width of the block cipher used, with which we can encrypt b plaintext bits using the XOR operation.
- We start with encrypting an IV with a block cipher.
- The cipher output gives us the first set of b key stream bits.
- The next block of key stream bits is computed by feeding the previous cipher output back into the block cipher and encrypting it.

OFB

- The OFB mode forms a synchronous stream cipher as the key stream does not depend on the plain or ciphertext.
- Using the OFB mode is quite similar to using a standard stream cipher such as RC4 or Trivium.
- Since the OFB mode forms a stream cipher, encryption and decryption are exactly the same operation.
- The receiver does not use the block cipher in decryption mode to decrypt the ciphertext.

OFB

- This is because the actual encryption is performed by the XOR function, and in order to reverse it, i.e., to decrypt it, we simply have to perform another XOR function on the receiver side.
- This is in contrast to ECB and CBC mode, where the data is actually being encrypted and decrypted by the block cipher.

OFB

- As a result of the use of an IV, the OFB encryption is also nondeterministic: encrypting the same plaintext twice results in different ciphertexts.
- As in the case for the CBC mode, the IV should be a nonce.
- One advantage of the OFB mode is that the block cipher computations are independent of the plaintext.
- One can precompute one or several blocks si of key stream material.

CFB

- The Cipher Feedback (CFB) mode also uses a block cipher as a building block for a stream cipher.
- It is similar to the OFB mode but instead of feeding back the output of the block cipher, the ciphertext is fed back.
- As in the OFB mode, the key stream is not generated bitwise but instead in a blockwise fashion.
- To generate the first key stream block, we encrypt an IV. For all subsequent key stream blocks, we encrypt the previous ciphertext.

CFB

- Since the CFB mode forms a stream cipher, encryption and decryption are exactly the same operation.
- The CFB mode is an example of an asynchronous stream cipher since the stream cipher output is also a function of the ciphertext.

CFB

- As a result of the use of an IV, the CFB encryption is also nondeterministic: encrypting the same plaintext twice results in different ciphertexts.
- As in the case for the CBC and OFB modes, the IV should be a nonce.

CTR

- Another mode which uses a block cipher as a stream cipher.
- As in the OFB and CFB modes, the key stream is computed in a blockwise fashion.
- The input to the block cipher is a counter which assumes a different value every time the block cipher computes a new key stream block.

CTR

- We have to be careful how to initialize the input to the block cipher.
- We must prevent using the same input value twice.
- If an attacker knows one of the two plaintexts that were encrypted with the same input, he can compute the key stream block and thus immediately decrypt the other ciphertext.
- We can use $IV||CTR$.

CTR

- Why do we need so many modes?
- One attractive feature of the CTR mode is that it can be parallelized because, unlike the OFB or CFB mode, it does not require any feedback.

GCM

- The Galois Counter Mode (GCM) is an encryption mode which also computes a message authentication code (MAC).
- A MAC provides a cryptographic checksum that is computed by the sender, Alice, and appended to the message.
- Bob also computes a MAC from the message and checks whether his MAC is the same as the one computed by Alice.
- This way, Bob can make sure that (1) the message was really created by Alice and (2) that nobody tampered with the ciphertext during transmission.
- These two properties are called message authentication and integrity, respectively.

GCM

- GCM protects the confidentiality of the plaintext x by using an encryption in counter mode.
- Additionally, GCM protects not only the authenticity of the plaintext x but also the authenticity of a string AAD called additional authenticated data.
- This authenticated data is, in contrast to the plaintext, left in clear in this mode of operation.

GCM

- The GCM consists of an underlying block cipher and a Galois field multiplier with which the two GCM functions authenticated encryption and authenticated decryption are realized.
- On the sender side, GCM encrypts data using the Counter Mode (CTR) followed by the computation of a MAC value.
- For encryption, first an initial counter is derived from an IV and a serial number.
- hen the initial counter value is incremented, and this value is encrypted and XORed with the first plaintext block.

GCM

- For subsequent plaintexts, the counter is incremented and then encrypted.
- Note that the underlying block cipher is only used in encryption mode.
- GCM allows for precomputation of the block cipher function if the initialization vector is known ahead of time.

GCM

- For authentication, GCM performs a chained Galois field multiplication.
- For every plaintext x_i an intermediate authentication parameter g_i is derived.
- g_i is computed as the XOR sum of the current ciphertext y_i and g_i, and multiplied by the constant H.
- The value H is a hash subkey which is generated by encryption of the all-zero input with the block cipher.

Outline

Basics

- We can divide symmetric key cryptography into block and stream ciphers if we consider encryption primitives.
- But we can also consider block and stream ciphers as parts of some other cryptographic mechanisms.
- We already introduced one stream cipher and a number of ways how to obtain stream cipher from block cipher.

Basics

- In practice, block ciphers are used more often than stream ciphers.
- Since stream ciphers tend to be small and fast, they are particularly relevant for applications with little computational resources.
- Traditionally, it was assumed that stream ciphers tended to encrypt more efficiently than block ciphers.
- Efficient for software-optimized stream ciphers means that they need fewer processor instructions (or processor cycles) to encrypt one bit of plaintext.
- For hardware-optimized stream ciphers, efficient means they need fewer gates (or smaller chip area) than a block cipher for encrypting at the same data rate.

Advantages

- An advantage of stream ciphers is that they avoid error propagation, which gives them an advantage in applications where errors may occur during the transmission.
- Another advantage which makes them well adapted to military use is that only the PRG needs to be protected; the devices which receive the keystream as input and perform the encryption do not require particular environment.

Practical Stream Ciphers

- Practical stream ciphers use a stream of key bits $s_1, s_2, \ldots$ that are generated by the keystream generator, which should have certain properties.
- A standard way is to use Linear Feedback Shift Registers (LFSRs).

LFSRs

- LFSRs are small circuits containing a number of memory cells (flip-flops), each of which holds one bit of information.
- The set of such cells form a register.
- In each cycle a certain predefined set of cells are "tapped" and their value is passed through a functions, called the feedback function.
- The register is then shifted down by one bit, with the output bit of the feedback shift register being the bit that is shifted out of the register.
- The combination of the tapped bits is then fed into the empty cell at the top of the register.

LFSRs

- A plain LFSR produces a sequence with good statistical properties but it is cryptographically weak.
- Combinations of LFSRs can make secure stream ciphers.

Mathematical Description of LFSRs

- There are m flip-flops and m possible feedback locations, all combined by the XOR operation.
- Whether a feedback path is active or not, is defined by the feedback coefficient $p_0, p_1, \ldots, p_{m-1}$.
- If $p_i = 1$ the feedback is active.
- If $p_i = 0$, the corresponding flip-flop is not used for feedback.
- The output values are given through a combination of some previous output values.

LFSRs

- Due to the finite number of recurring states, the output sequence of an LFSR repeats periodically.
- An LFSR can produce output sequences of different lengths, depending on the feedback coefficients.
- The maximum sequence length generated by an LFSR of degree m is $2^m 1$.
- LFSRs are often specified by polynomials using the following notation: an LFSR with a feedback coefficient vector $(p_{m-1}, \ldots, p_1, p_0)$ is represented by the polynomial.

LFSRs

$$P(x) = x^m + p_{m-1}x^{m-1} + \ldots + p_1 x + p_0. \tag{3}$$

- Maximum-length LFSRs have what is called primitive polynomials.
- Primitive polynomials can relatively easily be computed.

Known Plaintext Attack on LFSR

- LFSRs are linear.
- Linear systems are governed by linear relationships between their inputs and outputs.
- If we use an LFSR as a stream cipher, the secret key k is the feedback coefficient vector $(p_{m-1}, \ldots, p_1, p_0)$.
- Attack is possible if the attacker knows some plaintext and the corresponding ciphertext.
- Additionally, attacker knows the degree m of LFSR.
- Since he knows the ciphertext and plaintext, he can reconstruct part of the keystream bits.

Known Plaintext Attack on LFSR

- Attacker can construct a set of linear equations in m unknowns, which can be solved with Gaussian elimination.
- As soon as the attacker knows $2m$ output bits of an LFSR of degree m, the p_i coefficients can exactly be constructed by solving a system of linear equations.

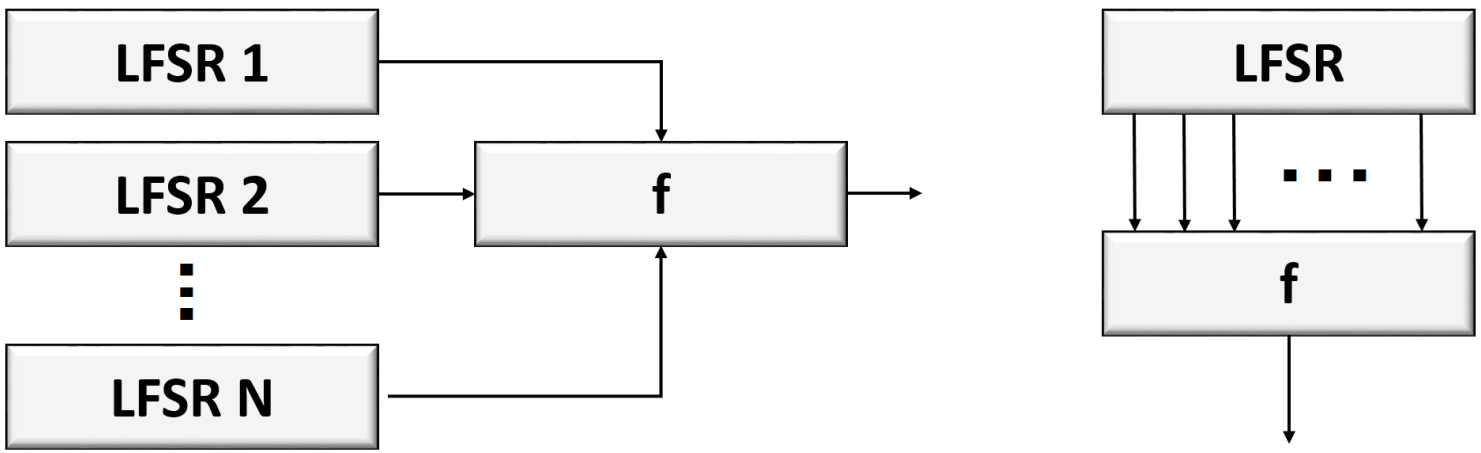
Linear Complexity

- For an infinite binary sequence $s_0, s_1, \ldots,$, the linear complexity of s is $L(s)$ where
  1. $L(s) = 0$ if s is the zero sequence.
  2. $L(s) = \infty$ if no LFSR generates s.
  3. $L(s)$ is the length of the shortest LFSR to generate s.

Combining LFSRs

- To obtain greater security, a common practice is to use a number of LFSRs, each producing a different output sequence.
- The key is the initial state of all LFSRs and the keystream is produced by using a nonlinear combination function.

Combining LFSRs



Figure

Combiner Generators

- We need to protect against correlation attack.
- We need a smart choice of nonlinear function.
- Boolean functions need to be balanced, with high nonlinearity, large algebraic degree, large algebraic immunity, large fast algebraic immunity, and large correlation immunity.

Filter Generators

- Take single LFSR with internal state $s_1, \ldots, s_L$ and then make the output of the stream cipher a nonlinear function of the whole state.

Alternating Step Generator

- Take 3 LFSRs of sizes L_1, L_2, and L_3, which are pairwise coprime and approximately the same size.
- Denote the output sequence x_1, x_2, x_3.
- The first LFSR is clocked on every iteration.
- If its output is equal to 1, then the second LFSR is clocked and the output of the third LFSR is repeated from its last value.
- If the output of first LFSR is 0, then the third LFSR is clocked and the output of the second LFSR is repeated from the last value.
- The output of the generator is $x_2 \oplus x_3$.

Shrinking Generator

- Take 2 LFSRs with output sequences x_1, x_2.
- Throw some of the x_2 stream under the control of x_1 stream.
- Both LFSRs are clocked at the same time and if x_1 is equal to 1 then the output of the generator is the value of x_2.
- If x_1 is equal to 0, then generator clocks again.
- As a consequence, the generator does not produce a bit at each iteration.

eSTREAM

- The eSTREAM project had the explicit goal to advance the state-of-the-art knowledge about stream cipher design.
- As part of this objective, new stream ciphers that might become suitable for widespread adoption were investigated.
- The ciphers were divided into two "profiles", depending on the intended application:
  1. Stream ciphers for software applications with high throughput requirements.
  2. Stream ciphers for hardware applications with restricted resources such as limited storage, gate count, or power consumption.

eSTREAM

- A total of 34 candidates were submitted to eSTREAM.
- At the end of the project four software-oriented ("Profile 1") ciphers were found to have desirable properties: HC-128, Rabbit, Salsa20/12, and SOSEMANUK.
- With respect to hardware-oriented ciphers ("Profile 2"), the following three ciphers were selected: Grain v1, MICKEY, and Trivium.

Trivium

- Trivium is a relatively new stream cipher which uses an 80-bit key.
- It is based on a combination of three shift registers.
- Even though these are feedback shift registers, there are nonlinear components used to derive the output of each register, unlike the LFSRs that we studied in the previous section

Description

- Three shift registers, A, B and C.
- The lengths of the registers are 93, 84 and 111.
- The XOR-sum of all three register outputs forms the keystream s_i.
- The output of each register is connected to the input of another register.
- Thus, the registers are arranged in circle-like fashion.
- The cipher can be viewed as consisting of one circular register with a total length of $93+84+111 = 288$.

Description

- The input of each register is computed as XOR of two bits:
  1. The output of register A is part of the input of register B.
  2. One register bit at a specific location is fed back to the input.
- The output of each register is computed as the XOR-sum of three bits:
  1. The rightmost register bit.
  2. One register bit at a specific location is fed forward to the output.
  3. The output of a logical AND function whose input is two specific register bits.

Security

- Note that the AND operation is equal to multiplication in modulo 2 arithmetic.
- If we multiply two unknowns, and the register contents are the unknowns that an attacker wants to recover, the resulting equations are no longer linear as they contain products of two unknowns.
- Thus, the feedforward paths involving the AND operation are crucial for the security of Trivium as they prevent attacks that exploit the linearity of the cipher.

Encryption

- Two input parameters: a key k and an initialization vector IV.
- Initially, an 80-bit IV is loaded into the 80 leftmost locations of register A, and an 80-bit key is loaded in the 80 leftmost locations of register B. All other register bits are set to zero with the exception of the three rightmost bits of register C, i.e., bits c_{109}, c_{110} and c_{111}, which are set to 1.
- In the first phase, the cipher is clocked $4 \cdot 288 = 1152$ times. No cipher output is generated.
- The bits produced hereafter, i.e., starting with the output bit of cycle 1153, form the keystream.

RC4

- Up to recently, widely deployed in browsers to secure traffic to websites using TLS.
- Very simple and very fast.
- Yet, recently discovered also very insecure.
- Especially vulnerable when the beginning of the output keystream is not discarded, or when nonrandom or related keys are used.

Description

- Take array S, indexed from 0 to 255, consisting of integers, permuted in key-dependent way.
- The output of RC4 is keystream of bytes K XORed with the plaintext in bytewise fashion.

RC4 Algorithms

- Let $i = 0, j = 0$.
- $i \leftarrow (i + 1) mod 256$.
- $j \leftarrow (j + S_i) mod 256$.
- Swap (S_i, S_j).
- $t \leftarrow (S_i + S_j) mod 256$.
- $K \leftarrow S_t$.

RC4 Key Schedule

- for $i = 0$ to 255 do $S_i = i$.
- Initialize K_i, for $i = 0, \ldots, 255$, with the key, repeating if necessary.
- $j \leftarrow 0$.
- for $i = 0$ to 255 do
  - $j \leftarrow (j + S_i + K_i) mod 256$.
  - Swap (S_i, S_j).

- Used to encrypt on-air traffic in the 2nd generation mobile networks.
- Developed in 1987, proprietary design.
- Reverse engineered in 1999.
- There is A5/2, which is a weakened version of A5/1.

Description

- Use 3 LFSRs with lengths 19, 22, and 23.
- Characteristic polynomials are:

$$\begin{aligned} x^{18} + x^{17} + x^{16} + x^{13} + 1 \\ x^{21} + x^{20} + 1 \\ x^{22} + x^{21} + x^{20} + x^{7} + 1. \end{aligned} \tag{4}$$

- The output of the cipher is XOR of 3 output bits of LFSRs.

Description

- To clock the registers, there is a "clocking bit" for each register.
- They are in positions 10, 11, 12 of LFSRs.
- Call these positions c_1, c_2, c_3.
- At each step, 3 bits are computed and the "majority bit" is determined as:

$$(c_1 \cdot c_2) \oplus (c_2 \cdot c_3) \oplus (c_1 \cdot c_3). \tag{5}$$

- The ith LFSR is clocked if the majority bit equals c_i.