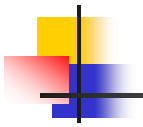
Encryption
Stream Cipher

Dr.Talal Alkharobi

Stream Cipher

- A symmetric cipher in which the plaintext digits are encrypted one at a time, and in which the transformation of successive digits varies during the encryption.
- An alternative name is a state cipher, as the encryption of each digit is dependent on the current state.
- In practice, the digits are typically single bits or bytes.
- Typically execute at a higher speed than block ciphers and have lower hardware complexity.
- Can be susceptible to serious security problems if used incorrectly: The same starting state must never be used twice.

3



OTP

- A one-time pad uses a keystream of completely random digits.
- The keystream is combined with the plaintext digits one at a time to form the ciphertext.
- OTP was proved to be theoretically secure by Shannon in 1949.
- The keystream must be (at least) the same length as the plaintext, and generated completely at random.
- This makes the system very cumbersome to implement in practice,
- OTP has been used for the most critical applications.

4

Example: ciphering

Message	H	E	L	L	O
Key	X	M	C	K	L
Message	7	14	11	11	14
Key	23	12	2	10	11
M+K	30	16	13	21	25
Mod 26	4	16	13	21	25
Ciphered msg	E	Q	N	V	Z

5

Example: Deciphering

Ciphered Msg	E	Q	N	V	Z
Key	X	M	C	K	L

Ciphered Msg	4	16	13	21	25
Key	23	12	2	10	11
CM-K	-19	4	11	11	14
Mod 26	7	4	11	11	14
Recovered Text	H	E	L	L	O

6

Stream Cipher vs OTP

- Stream ciphers can be viewed as approximating the OTP
- A stream cipher makes use of a much smaller and more convenient key — 128 bits,
- Based on the key, it generates a pseudorandom keystream which can be combined with the plaintext digits in a similar fashion to the one-time pad.
- This comes at a cost: because the keystream is now pseudorandom, and not truly random,
- The proof of security associated with the OTP no longer holds.
- It is quite possible for a stream cipher to be completely insecure.

7

Stream Cipher Types

- A stream cipher generates successive elements of the keystream based on an internal state.
- This state is updated in essentially two ways:
  - Synchronous: the state changes independently of the plaintext or ciphertext messages,
  - Self-synchronising: update their state based on previous ciphertext digits.

8

Synchronous stream ciphers

- In a synchronous stream cipher a stream of pseudo-random digits is generated independently of the plaintext and ciphertext messages, and then combined with the plaintext (to encrypt) or the ciphertext (to decrypt).
- In the most common form, binary digits are used (bits), and the keystream is combined with the plaintext using the exclusive or operation (XOR).
- This is termed a binary additive stream cipher.
- In a synchronous stream cipher, the sender and receiver must be exactly in step for decryption to be successful.

9

Synchronous stream ciphers

- If digits are added or removed from the message during transmission, synchronization is lost.
- To restore synchronization, various offsets can be tried systematically to obtain the correct decryption.
- Another approach is to tag the ciphertext with markers at regular points in the output.
- If, however, a digit is corrupted in transmission, rather than added or lost, only a single digit in the plaintext is affected and the error does not propagate to other parts of the message.

10

Synchronous stream ciphers

- This property is useful when the transmission error rate is high;
- It makes it less likely the error would be detected without further mechanisms.
- Because of this property, synchronous stream ciphers are very susceptible to active attacks — if an attacker can change a digit in the ciphertext, he might be able to make predictable changes to the corresponding plaintext bit; for example, flipping a bit in the ciphertext causes the same bit to be flipped in the plaintext.

Self-synchronizing stream ciphers

11

- Also known as asynchronous stream ciphers or ciphertext autokey (CTAK).
- This approach uses several of the previous N ciphertext digits to compute the keystream.
- The idea of self-synchronization was U.S patented in 1946
- It has the advantage that the receiver will automatically synchronise with the keystream generator after receiving N ciphertext digits, making it easier to recover if digits are dropped or added to the message stream.

Self-synchronizing stream ciphers

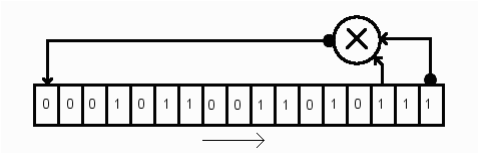
12

- Single-digit errors are limited in their effect, affecting only up to N plaintext digits.
- It is somewhat more difficult to perform active attacks on self-synchronising stream ciphers by comparison with their synchronous counterparts.

13

Linear feedback shift register-based stream ciphers

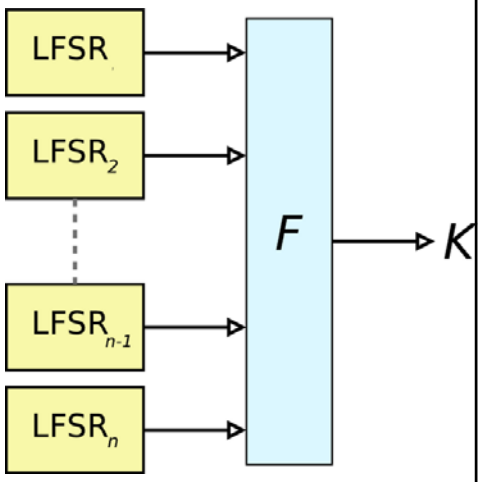
- Linear feedback shift registers (LFSRs) are popular components in stream ciphers as they can be implemented cheaply in hardware, and their properties are well-understood.
- The use of LFSRs on their own, however, is insufficient to provide good security.
- Various schemes have been proposed to increase the security of LFSRs.



14

Non-linear combining functions

- Because LFSRs are inherently linear, one technique for removing the linearity is to feed the outputs of several parallel LFSRs into a non-linear Boolean function to form a combination generator.
- Various properties of such a combining function are critical for ensuring the security of the resultant scheme, for example, in order to avoid correlation attacks



15

Clock-controlled generators

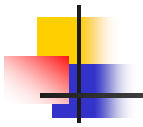
- Normally LFSRs are stepped regularly.
- One approach to introducting non-linearity is to have the LFSR clocked irregularly, controlled by the output of a second LFSR.
- Such generators include the
  - stop-and-go generator,
  - alternating step generator
  - shrinking generator.

16

Security

- To be secure, the period of the keystream, that is, the number of digits output before the stream repeats itself, needs to be sufficiently large.
- If the sequence repeats, then the overlapping ciphertexts can be aligned against each other "in depth", and there are techniques which could allow the plaintext to be extracted.

17



Usage

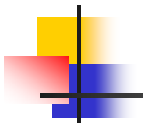
- Stream ciphers are often used in applications where plaintext comes in quantities of unknowable length—for example, a secure wireless connection.
- If a block cipher were to be used in this type of application, the designer would need to choose either transmission efficiency or implementation complexity, since block ciphers cannot directly work on blocks shorter than their block size.
- For example, if a 128-bit block cipher received separate 32-bit bursts of plaintext, three quarters of the data transmitted would be padding.

18

Usage

- Block ciphers must be used in ciphertext stealing or residual block termination mode to avoid padding, while stream ciphers eliminate this issue by naturally operating on the smallest unit that can be transmitted (usually bytes).
- Another advantage of stream ciphers in military cryptography is that the cipher stream can be generated in a separate box that is subject to strict security measures and fed to other devices, e.g. a radio set, which will perform the xor operation as part of their function.
- The latter device can then be designed and used in less stringent environments.

19



Algorithms

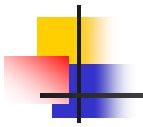
- RC4 is the most widely used stream cipher in software;
- Others include: A5/1 | A5/2 | E0 | FISH | Grain | HC-256 | ISAAC | LILI-128 | MUGI | Panama | Phelix | Pike | Py | Rabbit | RC4 | Salsa20 | Scream | SEAL | SOBER | SOBER-128 | SOSEMANUK | Trivium | VEST | WAKE | Chameleon | Helix

20

RC4

- Also known as ARC4 or ARCFOUR
- The most widely-used software stream cipher and is used in popular protocols such as Secure Sockets Layer (SSL) (to protect Internet traffic) and WEP (to secure wireless networks).
- While remarkable in its simplicity, RC4 falls short of the high standards of security set by cryptographers, and some ways of using RC4 can lead to very insecure cryptosystems (including WEP).
- It is not recommended for use in new systems.
- However, some systems based on RC4 are secure enough for practical use.

21



RC4

- RC4 was designed by Ron Rivest of RSA Security in 1987;
- RC4 was initially a trade secret, but in 1994 a description of it was anonymously posted to the Cypherpunks mailing list.
- It was soon posted on the sci.crypt newsgroup, and from there to many sites on the Internet.
- Because the algorithm is known, it is no longer a trade secret.
- The name "RC4" is trademarked. RSA has never officially released the algorithm), to avoid possible trademark problems.
- The current status seems to be that "unofficial" implementations are legal, but cannot use the RC4 name.

22

RC4 - description

- RC4 generates a pseudorandom keystream which is combined with the plaintext using XOR; decryption is performed the same way.
- To generate the keystream, the cipher makes use of a secret internal state which consists of two parts:
  - A permutation of all 256 possible bytes
  - Two 8-bit index-pointers
- The permutation is initialized with a variable length key, typically 40 and 256 bits, using the key-scheduling algorithm (KSA).
- Once this has been completed, the stream of bits is generated using the pseudo-random generation algorithm (PRGA)

RC4 – description key-scheduling algorithm (KSA)

23

- The key-scheduling algorithm is used to initialize the permutation in array "S".
- "keylength" is defined as the number of bytes in the key and can be in the range $1 \leq \text{keylength} \leq 256$, typically between 5 and 16, corresponding to a key length of 40 – 128 bits.
- First, the array "S" is initialised to the identity permutation.
- S is then processed for 256 iterations in a similar way to the main PRGA algorithm, but also mixes in bytes of the key at the same time.

RC4 – description key-scheduling algorithm (KSA)

24

```
for i from 0 to 255
    S[i] := i
j := 0
for i from 0 to 255
    j := (j + S[i] + key[i mod keylength]) mod 256
    swap(S[i],S[j])
```

The pseudo-random generation algorithm (PRGA)

25

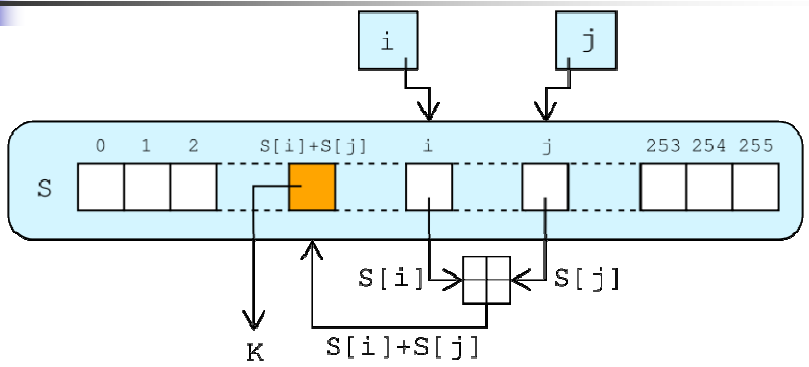
- For as many iterations as are needed, the PRGA modifies the state and outputs a byte of the keystream.
- In each iteration, the PRGA
  - increments i,
  - adds the value of S pointed to by i to j,
  - exchanges the values of S[i] and S[j],
  - outputs the value of S at the location S[i] + S[j] (modulo 256).
- Each value of S is swapped at least once every 256 iterations.

The pseudo-random generation algorithm (PRGA)

26

- `i := 0`
- `j := 0`
- `while GeneratingOutput:`
  - `i := (i + 1) mod 256`
  - `j := (j + S[i]) mod 256`
  - `swap(S[i],S[j])`
  - `output S[(S[i] + S[j]) mod 256]`
- `endwhile`

27

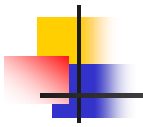


28

RC4 Implementation

- Many stream ciphers are based on linear feedback shift registers (LFSRs), which while efficient in hardware are less so in software.
- The design of RC4 avoids the use of LFSRs, and is ideal for software implementation, as it requires only byte manipulations.
- It uses
  - 256 bytes of memory for the state array, S[0] through S[255],
  - k bytes of memory for the key, key[0] through key[k-1],
  - integer variables, i, j, and k.
- Performing a modulus 256 can be done with a bitwise AND with 255 (on most platforms, simple bytes addition ignoring overflow)

29



RC4 Security

- RC4 falls short of the standards set by cryptographers for a secure cipher in several ways, and thus is not recommended for use in new applications.
- The keystream generated by RC4 is slightly biased in favor of certain sequences of bytes.
- The best attack based on this bias is due to Fluhrer and McGrew, which will distinguish the keystream from a random stream given a gigabyte of output.

30

RC4 Security

- RC4 does not take a separate nonce alongside the key.
- Such a nonce is, in general, a necessary requirement for security, so that encrypting the same message twice produces a different ciphertext each time.
- One approach to addressing this is to generate a "fresh" RC4 key by hashing a long-term key with a nonce.
- However, many applications that use RC4 simply concatenate key and nonce;
- RC4's weak key schedule then gives rise to a variety of serious problems.

Fluhrer, Mantin and Shamir attack

31

- In 2001 a new and surprising discovery was made by Fluhrer, Mantin and Shamir:
- Over all possible RC4 keys, the statistics for the first few bytes of output keystream are strongly non-random, leaking information about the key.
- If the long-term key and nonce are simply concatenated to generate the RC4 key, this long-term key can be discovered by analyzing large number of messages encrypted with this key.
- This and related effects were then used to break the WEP ("wired equivalent privacy") encryption used with 802.11 wireless net

Fluhrer, Mantin and Shamir attack

32

- This caused a scramble for a standards-based replacement for WEP in the 802.11 market, and led to the IEEE 802.11i effort and WPA.
- Cryptosystems can defend against this attack by discarding the initial portion of the keystream (say the first 1024 bytes) before using it.

33

Combinatorial problem

- A combinatorial problem related to the number of inputs and outputs of the RC4 cipher was first posed by Itsik Mantin and Adi Shamir in 2001,
- The total 256 elements in the typical state of RC4, if x number of elements ($x \leq 256$) are only known (all other elements can be assumed empty), then the maximum number of elements that can be produced deterministically is also x in the next 256 rounds.
- This conjecture was put to rest in 2004 with a formal proof given by Souradyuti Paul and Bart Preneel.

34

RC4-based cryptosystems

- WEP
- WPA
- CipherSaber
- BitTorrent protocol encryption
- Microsoft Point-to-Point Encryption
- Secure Sockets Layer (an option)
- Secure shell (an option)
- Kerberos (an option)