CSC3631 Cryptography

Stream Cipher

Changyu Dong

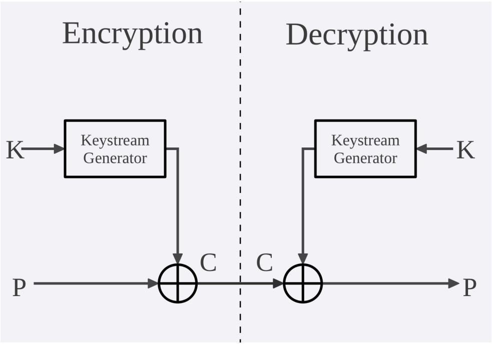
Newcastle University

Modern Cryptography

- Cryptography we are currently using.
- Clearly defined
- Rigorous design
- Provably secure (at least mathematically)

Stream Cipher

- Encrypt individual bits one at a time.
- A key is used to generate a **key stream** – a bit stream.
- To encrypt, plaintext stream is XORed ($\oplus$) with the key stream.
- To decrypt, ciphertext stream is XORed with the (same) key stream.



XOR

- Exclusive OR, or simply XOR, is one of the most widely used operations in cryptography.
- XOR is a Boolean operation that takes two bits as the input and outputs one bit.
- Its truth table is as the following:

A	B	A $\oplus$ B
1	1	0
0	0	0
1	0	1
0	1	1

- If the bits are the same, the result is 0; if the bits are different, the result is 1.

Properties of XOR

- XOR has a few interesting properties that makes it useful in cryptogrpahy:
  - It is associative and commutative
    - Associative: $(A \oplus B) \oplus C = A \oplus (B \oplus C)$
    - Commutative: $A \oplus B = B \oplus A$
  - Anything XORs 1 is its negation: $A \oplus 1 = \neg A$
  - Anything XORs 0 is itself $A \oplus 0 = A$
  - Anything XORs itself is 0 $A \oplus A = 0$

Vernam Cipher

- A stream cipher invented by Vernam in the 1917
- Key is a random bit string that is no shorter than the plaintext
- Bitwise XOR for encryption and decryption
- Plaintext as a bit stream XORed with with the key stream
- $c_i = m_i \oplus k_i$
- To decrypt, XOR the ciphertext stream with the same key stream again
- $c_i \oplus k_i = (m_i \oplus k_i) \oplus k_i = m_i \oplus (k_i \oplus k_i) = m_i \oplus 0 = m_i$

How Secure is the Vernam Cipher

- Intuitively, it is secure
- The key is random, the ciphertext is then also random
- Claude Shannon proved that this cipher is unconditionally secure (perfect secrecy).

Perfect Secrecy

- The first rigorous security definition, still used in literatures nowadays.
- a cipher has perfect secrecy means:
  - It is secure: Given the ciphertext, the attacker's knowledge about the plaintext is as much as not given the ciphertext.
  - The security is unconditional: Even if the attacker has unlimited computational power, the cipher is still secure.
- More formally, for any plaintext m and ciphertext c, $Pr[M = m|C = c] = Pr[M = m]$
  - $Pr[M = m]$: the estimated probability of the plaintext being a particular message m before seeing the ciphertext.
  - $Pr[M = m|C = c]$: the estimated probability of the plaintext being m after knowing that the ciphertext is c.

One-bit Encryption Using the Vernam Cipher

- The plaintext is 1-bit long and the key is 1-bit long
- The plaintext can be either 0 or 1
- The key can be either 0 or 1
- For whatever reason, the attacker thinks the probability of the plaintext being 1 is p before seeing the ciphertext
- Then the probability of the plaintext being 0 is $1 - p$ from the attacker's point of view
- Since the key is chosen randomly, it is 50/50 being 1 or 0
- That is
  - $Pr[M = 1] = p$
  - $Pr[M = 0] = 1 - p$
  - $Pr[K = 0] = 0.5$
  - $Pr[K = 1] = 0.5$

One-bit Encryption Using the Vernam Cipher

$Pr[M = 1] = p$
$Pr[M = 0] = 1 - p$
$Pr[K = 0] = 0.5$
$Pr[K = 1] = 0.5$

- The ciphertext is 1 when the plaintext is 1 and the key is 0, or when the plaintext is 0 and the key is 1, so the probability of the ciphertext being 1 is:

$$
\begin{aligned}
& Pr[C = 1] \\
= \;& Pr[M = 1 \text{ and } K = 0] + Pr[M = 0 \text{ and } K = 1] \\
= \;& Pr[M = 1] \times Pr[K = 0] + Pr[M = 0] \times Pr[K = 1] \\
= \;& p \times 0.5 + (1 - p) \times 0.5 \\
= \;& 0.5p + 0.5 - 0.5p \\
= \;& 0.5
\end{aligned}
$$

- Then the probability of the ciphertext being 0 is $Pr[C = 0] = 1 - 0.5 = 0.5$

One-bit Encryption Using the Vernam Cipher

- Now the attacker sees the ciphertext
- If the ciphertext is 0, there are two cases

$Pr[M = 1] = p$
$Pr[M = 0] = 1 - p$
$Pr[K = 0] = 0.5$
$Pr[K = 1] = 0.5$
$Pr[C = 0] = 0.5$
$Pr[C = 1] = 0.5$

1. The plaintext is 1, the probability of this case

$$
\begin{aligned}
& Pr[M = 1 | C = 0] \\
= & \frac{Pr[M = 1 \textit{ and } C = 0]}{Pr[C = 0]} \\
= & \frac{Pr[M = 1 \textit{ and } K = 1]}{Pr[C = 0]} \\
= & \frac{Pr[M = 1] \times Pr[K = 1]}{Pr[C = 0]} \\
= & \frac{p \times 0.5}{0.5} \\
= & p = Pr[M = 1]
\end{aligned}
$$

2. The plaintext is 0, the probability of this case

$$
\begin{aligned}
& Pr[M = 0 | C = 0] \\
= & \frac{Pr[M = 0 \textit{ and } C = 0]}{Pr[C = 0]} \\
= & \frac{Pr[M = 0 \textit{ and } K = 0]}{Pr[C = 0]} \\
= & \frac{Pr[M = 0] \times Pr[K = 0]}{Pr[C = 0]} \\
= & \frac{(1 - p) \times 0.5}{0.5} \\
= & 1 - p = Pr[M = 0]
\end{aligned}
$$

One-bit Encryption Using the Vernam Cipher

- If the ciphertext is 1, there are two similar cases
  3. The plaintext is 1, $Pr[M = 1|C = 1] = Pr[M = 1]$
  4. The plaintext is 0, $Pr[M = 0|C = 1] = Pr[M = 0]$
- These are the only cases
- In all cases $Pr[M = m|C = c] = Pr[M = m]$
- The proof can be generalised to multiple bits
- The Vernam cipher indeed has perfect secrecy

Limitations of Perfect Secrecy

- Hard to achieve and perfectly secure ciphers are difficult to use in practice
  - Requires truly randomly keys
  - Key-length $\geq$ message-length
  - Keys are never reused

Modern Stream Ciphers

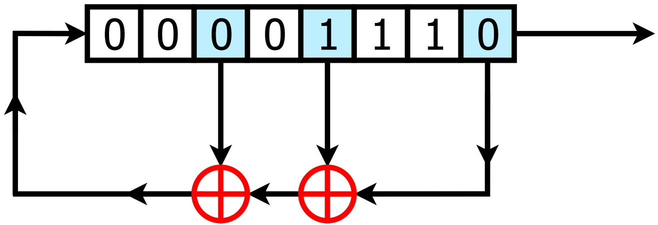
- Main differences:
  - The keystream is not truly random, but pseudorandom
  - The quality of the pseudorandom bit streams generated by a stream cipher must be very similar to that of real random bit streams
  - Keystream generators produce a stream of pseudorandom bits using short secret keys

PRG and Stream Ciphers

- The keystream generators are deterministic: the same key generates the same keystream
- Essentially, stream ciphers are pseudorandom generators (PRGs), the keys are the seeds
- One bit change in the key should change the output completely

Linear Feedback Shift Registers (LFSRs)

- A shift register whose input bit is a linear function of its previous state.
- In each cycle, some bits in the register is XORed together to produce a new bit, then the register shift and output 1 bit, the new bit then takes the empty space of the register.



Recurrence

- It is impossible to generate an infinitely long key stream.
- After some cycles, LFSR will return to a state it has been in, and start to produce the same bit stream it has produced before.
- The period can be short or long, depending on the design.
- The best case: for an L-bit LFSR, the period is $2^L - 1$ (i.e. can produce a $2^L - 1$ bit long stream).
  - $L = 128$, we can get $2^{128} - 1$ bits
  - 1 TB $= 2^{43}$ bits, you need $2^{85} = 38$ trillion trillion 1TB disks if you want to store the whole stream.
- (If you want to know the math about how to design a good LFSR, read Cryptography made simple §12.2 – not required)

Combine LFSRs

- LFSRs, although can produced large number of non-repeating bits, are insecure.
- For an L-bit LFSR, it is sufficient to determine its entire stream if one can obtain $2L$ consecutive bits from the stream.
  - Due to the linearity of the function
  - If a stream cipher is naively built on top of LFSR, it would not secure against a known plaintext attack.
- To solve the problem, use multiple LFSRs and combine the output using a non-linear function.
- (If you want to know more, read Cryptography made simple §12.4 – not required)

RC4

- The most commonly implemented stream ciphers
- Historically used in Secure Sockets Layer (SSL) (to protect Internet traffic) and WEP (to protect Wireless traffic) – still being used in many legacy devices.
- Support variable key size (40 -256 bits)
- Output unbounded number of bytes
- Simple design and easy to implement
- Not secure – should be avoided if possible.

Other Stream Ciphers

- SEAL (Software-optimized Encryption Algorithm): 160-bit key
- Grain, HC-256, MICKEY, Rabbit, Salsa20, SOSEMANUK, Trivium . . .
- See wikipedia for a (possibly incomplete) list.

Advantage of Stream Ciphers

- Fast
- Easy to implement in hardware
- Can encrypt streams
  - You don't want to buffer data before encryption
- However, nowadays, block ciphers can work as a stream cipher (in certain mode) and their speed can be very good with hardware acceleration (e.g. Intel AES instruction set) – in many cases you don't really need a pure stream cipher.

Weaknesses of Stream Ciphers

- If the same key stream is ever used twice, then easy to break
  - $C_1 = A \oplus K$, $C_2 = B \oplus K$
  - $C_1 \oplus C_2 = A \oplus K \oplus B \oplus K = A \oplus B$
  - Reveal partial information about A and B
- Ciphertext can be modified so that plaintext is changed accordingly after decryption
  - Send 1 bit means yes/no
  - An attacker intercepts the ciphertext and flip the bit then passes the modified bit to the receiver
  - Receiver decrypts the bit but no the plaintext is exactly the opposite

How to Counter the Weaknesses

- Key stream reuse
  - Use Initialisation Vector (IV).
  - Essentially a random or pseudorandom bit string
  - For each encryption, generates a new IV and combine it with the key to form a one-time key
  - IV can be sent in clear with the ciphertext,
- Ciphertext modification
  - Need some integrity protection mechanisms
  - Usually Message authentication code (will see later)

RC4 and WEP

- WEP (Wired Equivalent Privacy): used in 802.11 wireless network to provide security
- WEP uses RC4 for confidentiality
  - A long term secret key k (that's fine)
  - An IV is prepended to the key before encrypting a packet (that's also fine)
  - IV is sent in clear with the packet (that's still fine)
  - IV is 24-bit long (that's NOT fine)
  - In most systems, implemented as a counter starting from 0 (that makes things even worse)
- Passive attack: collect enough raw encrypted data and look for plaintext encrypted with the same IV.
- A table-based attack:
  - An insider generates a packet for each IV.
  - Extracts the key stream by xoring the ciphertext with the plaintext.
  - Stores all the key streams in a table indexed by the IV. (Requires 15GB in total.)

RC4 and WEP

- More sophisticated attack
  - Some correlation allows an attacker to recover once a byte in the key given enough key streams.
  - Certain packets used in TCP/IP have fixed structure and predictable content (known-plaintext)
  - You can get key streams without an insider
  - You can send the packet back to get new reply packets
    - You can collect enough key stream with packet injection very fast
- 104-bit key can be recovered in less than 1 minute
- Using less than 40,000 packets with a success probability of 50%. In order to succeed in 95% of all cases, 85,000 packets are needed.
- (Here the attacks are based on implementation and design flaws of WEP, rather than attacking RC4 itself. A recent attack on WPA-TKIP and TLS is actually by attacking RC4. see `https://www.rc4nomore.com/`)

Reading

- Cryptography made simple §9.1,9.2, 10.2, 12
- Cryptography theory and practice $1.1.7, 2.3
- Applied cryptography: §16,17