Lecture 3: Stream Ciphers

Stephen Huang

Content

1. Terminology

Encryption Basic Review:

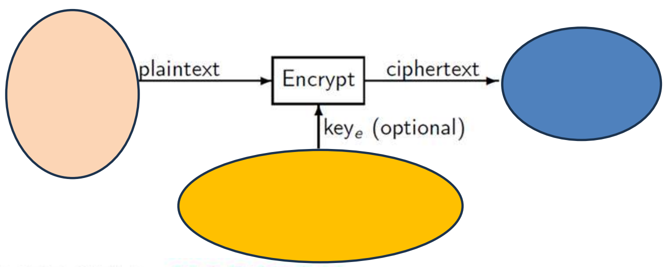
- Plaintext: unencrypted information
  - `Hello World`
- Ciphertext: encrypted information
  - `JNTY67BKJJWX`
- Key: Secret to transform between plaintext and ciphertext.
  - `A0123bc99j98kl87aBc7d`

Desired Properties

- Unique Decryption: Decrypting the encryption of a message always results in the original message.?
- Confidential: Difficult to extract information about the plaintext without the key.
- Why not impossible?
- Efficient: Encryption, encryption, and key generation are efficiently computable.
- Secure: It should be hard to guess the key, even with the knowledge of a plaintext/ciphertext pair.

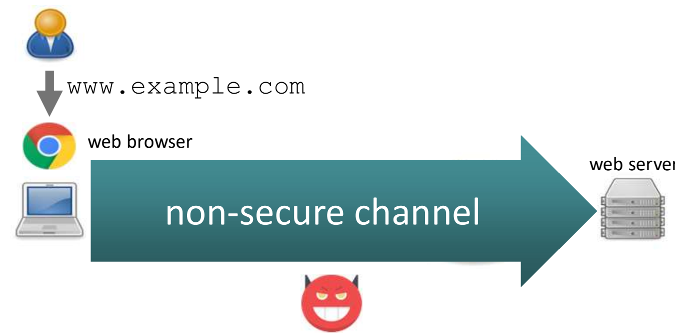
Spaces

- Plaintext Space: Set of valid plaintexts.
- Ciphertext Space: Set of valid ciphertexts.
- Key Space: Set of valid keys.
  - 128-bit key -> 2^{128} possible keys

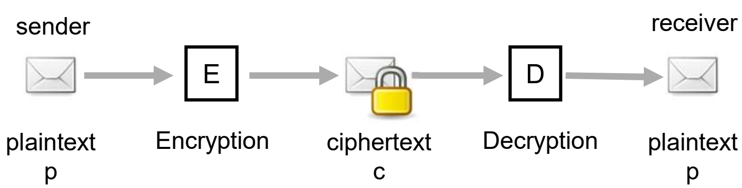


1. Perfect Security

Motivating Example: Web Security



Reminder: Encryption



- Plaintext p
- Encryption E
- Ciphertext c
- Decryption D

Kerckhoffs' Principle

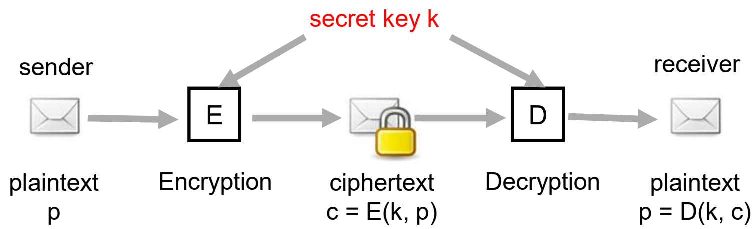
- The design requirement for ciphers (1883):

  "A cryptographic system should be secure even if all of its details, except for the key, are publicly known."

- Rejection of security by obscurity for cryptography.
- Obscuring security leads to a false sense of security, which is often more dangerous than not addressing security at all.


Auguste Kerckhoff

Reminder: Encryption

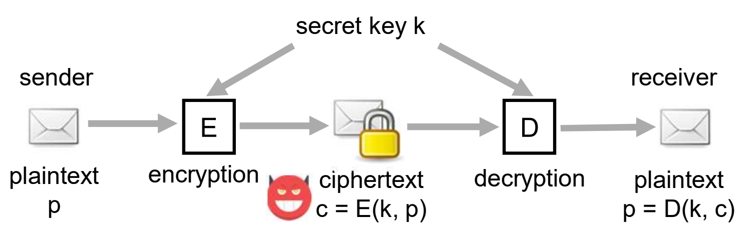


- Model
  - Plaintext p
  - Secret key k
  - Encryption E
  - Ciphertext c = E(k, p)
  - Decryption D

Truly Random Keys

- Clock drift
  - Modern computers often have more than one crystal oscillator for timing (e.g., for real-time clock, for CPU).
  - Clock crystals are not precise, and there can be random variations in their speeds.
- Disk drives
  - hard disk drives have small random fluctuations in their rotational speeds due to chaotic air turbulence
  - measuring disk seek-time captures this randomness
- Linux: /dev/random
  - multiple sources (e.g., mouse and keyboard activity, disk I/O operations, specific interrupts)
  - similar sources on other operating systems

Reminder: Encryption



- Attacker's goal: recover the secret key or the plaintext
- None of the classic cryptographic algorithms are secure

How to define security?

How to define security?

- First idea: "Attacker cannot recover the secret key"
  - E(k, p) = p would be considered secure under this definition.
- Second idea: "Attacker cannot recover the complete plaintext"
  - Suppose that E(k, p) is secure, then,
    E(k, p1 ‖ p2) = p1 ‖ E(k, p2) would also be secure.
- Third idea: "Attacker cannot recover any part of the plaintext"
  - What if some part of the plaintext is deterministic (e.g., "`GET / HTTP/1.1 …`")?
- None of these ideas define security correctly.

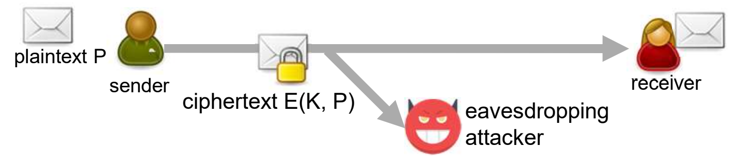
2. Perfect Security

- Published by Claude Shannon in 1949, based on his classified report from 1945.
- Perfect Secrecy: "*After a cryptogram is intercepted by the enemy, the a posteriori probabilities of this cryptogram representing various messages be identically the same as the a priori probabilities of the same messages before the interception.*"

Claude Shannon

≈ The attacker gains no information about the plaintext from observing the ciphertext.

Perfect Security Formally



- Perfect security

$$\Pr[P = p] \quad = \quad \Pr[P = p \mid E(K, P) = c]$$

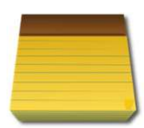
probability that plaintext P is a particular message p (without observing the ciphertext)

probability that plaintext P is a particular message p given that the ciphertext E(K, P) is c

- Equivalent to saying that plaintext P and ciphertext E(K, P) are independent.

One-Time Pad

- Works for binary data as well as alphabetic text.
  - Plaintext: sequence of bits or letters
- Key: sequence of bits or letters
  1. at least as long as the plaintext
  2. chosen uniformly at random
  3. used to encrypt only one message



- Encryption/decryption: modulo add/subtract each bit (or letter) of the key to the corresponding bit (or letter) of the plaintext/ciphertext
  - binary: $c_i = p_i \oplus k_i$ (XOR operation, i.e., modulo 2 addition)
  - alphabetic: $c_i = p_i + k_i \bmod 26$ and $p_i = c_i - k_i \bmod 26$ (for 26 letters)

One-Time Pad: Binary Example

- XOR operation:

  same as x + y mod 2 addition (or subtraction since -1 = 1 mod 2)

X	Y	X ⊕ Y
0	0	0
0	1	1
1	0	1
1	1	0

- Encryption
  - plaintext: 0 1 0 1
  - key: 0 1 1 0
  - ciphertext: 0 0 1 1

⇩

- Decryption
  - ciphertext: 0 0 1 1
  - key: 0 1 1 0
  - plaintext: 0 1 0 1

One-Time Pad: Alphabets

- Encode space as the 27th letter and perform addition modulo 27

a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	SPACE
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26

- plaintext: `mr mustard with the candlestick in the hall`
  key: `pxlmvmsydofuyrvzwc tnlebnecvgdupahfzzlmnyih`
  ciphertext: `ANKYODKYUREPFJBYOJDSPLREYIUNOFDOIUERFPLUYTS`
- plaintext: `miss scarlet with the knife in the library`
  key: `pftgpmiydgaxgoufhklllmhsqdqogtewbqfgyovuhwt`
  ciphertext: `ANKYODKYUREPFJBYOJDSPLREYIUNOFDOIUERFPLUYTS`
- Both keys are equally likely: an attacker cannot learn which key and plaintext are the correct ones from observing the ciphertext

Security of One-Time Pad

- One-time pad is perfectly secure.
- Proving perfect security: We have to show that for any plaintext p and ciphertext c,

  $$\Pr[P = p \mid E(K, P) = c] = \Pr[P = p]$$

  Proof for special case: plaintext is 1-bit long (either 0 or 1)

  key and ciphertext are also 1-bit long

Proof

- Given that the ciphertext is 0, the probability that the plaintext is 0:

$$\begin{aligned} &\Pr[P=0 \mid E(K,0)=0] \\ =\ &\Pr[P=0 \mid K \oplus 0=0] \\ =\ &\Pr[P=0 \mid K=0] \\ =\ &\Pr[P=0] \end{aligned}$$

- Given that the ciphertext is 0, the probability that the plaintext is 1.

$$\begin{aligned} &\Pr[P=1 \mid E(K,1)=0] \\ =\ &\Pr[P=1 \mid K \oplus 1=0] \\ =\ &\Pr[P=1 \mid K=1] \\ =\ &\Pr[P=1] \end{aligned}$$

- Similar argument works for the general case.

It's okay if you don't understand the proof.

One-Time Pad in Practice

- One-time pad has been used in special applications since the early 20th century
  - early systems used pencil, paper, and mental arithmetic
  - reportedly used by various intelligence agencies and diplomatic services
- During World War II and the Cold War, Soviet security agencies made heavy use of one-time pads
  - keys were often printed on miniaturized pads made of highly flammable paper

- Washington-Moscow hotline
  - established in 1963 between the Pentagon and the Kremlin
  - each country delivered keys on tape via its embassy abroad

Limitations of Perfect Security

Why is a one-time pad not the "end of cryptography?"

- For perfect security, we always need that

  length of key ≥ length of the plaintext

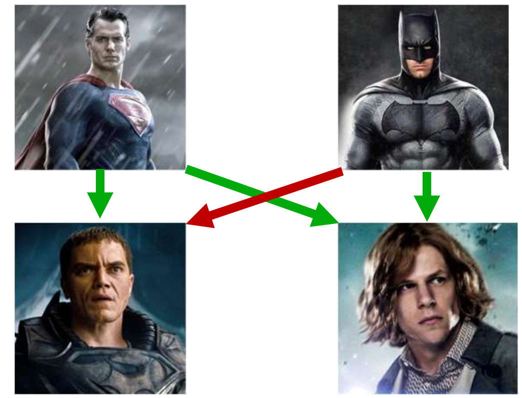
If you want to encrypt your 100 GB disk, you will need another 100 GB disk to store the key.

- Practical problems
  - distributing and storing long keys
  - generating long truly random keys

Key for OTP is uniform and cannot be compressed.

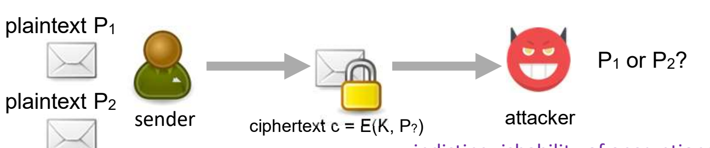
- Practical protocols, such as SSL (HTTPS), IPSec, or SSH, do not use a one-time pad.

Perfect vs Practical Security



Semantic Security

- Proposed by Goldwasser and Micali in 1982.
- Definition for chosen plaintext attacks (simplified):
  - attacker chooses two plaintexts
  - sender encrypts one of the two plaintexts (chosen at random)
  - attacker observes ciphertext and must guess which plaintext was encrypted
- Encryption is semantically secure if the attacker's advantage for any efficiently computable guess is negligible over random guessing.



indistinguishability of encryptions

Many-Time Pad: Bad Idea

- Idea: simply reuse the key for multiple plaintexts
- key: k = 01010011
  plaintexts: p_1 = 00111101
  ciphertexts: c_1 = 01101110

- attacker can get:
  $c_1 \oplus c_2$ = 10001101
  $c_1 \oplus c_2 = (p_1 \oplus k) \oplus (p_2 \oplus k) = p_1 \oplus p_2 \oplus k \oplus k = p_1 \oplus p_2$
- Problem: if the attacker knows some parts of a plaintext, it can recover the same parts of another plaintext
  - example: attacker knows p_1 = 00111101 (e.g., standard protocol format)
  - attacker can compute $p_1 \oplus c_1 \oplus c_2 = p_1 \oplus p_1 \oplus p_2 = 0 \oplus p_2 = p_2$

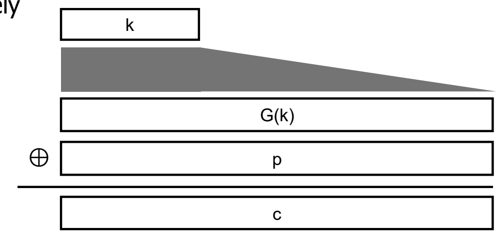
Many-Time Pad: Bad Idea

- Idea: simply reuse the key for multiple plaintexts
- key: `k = 01010011`
  plaintexts: `p1 = 00111101`
  ciphertexts: `c1 = 01101110`

- attacker can get:
  $c1 \oplus c2$ = `10001101`
  $c1 \oplus c2 = (p1 \oplus k) \oplus (p2 \oplus k) = p1 \oplus p2 \oplus k \oplus k = p1 \oplus p2$
- Problem: if the attacker knows some parts of a plaintext, it can recover the same parts of other plaintexts
  - example: p1 = "GET …" (HTTP GET request), p2 = ???
  - attacker learns from $p1 \oplus p2$

3. Stream Ciphers

- Idea: make one-time pad practical by extending the key securely

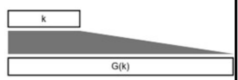


- Deterministic pseudorandom number generator G(k)
  - takes a fixed length seed k (i.e., the key) and produces a sequence of bits
- Decryption: generate the same sequence and XOR to the ciphertext

Pseudorandom Number Generator

Pseudorandom Number Generator (PRNG): takes a fixed-length seed and generates a sequence of bits using a deterministic algorithm.

- Performance requirement



  - PRNG must generate sequences that are as long as the plaintexts, so it must be computationally efficient.
  - many algorithms are tailored for hardware architectures or implementations.
- Security requirement: for an attacker who does not know the key, the bit sequence must be indistinguishable from true randomness.
- Statistical randomness tests (NIST "Statistical Test Suite") such as
  - frequency of 0s and 1s must be roughly equal
  - distribution of the lengths of uninterrupted "runs" of 0s and 1s (e.g., 10001 or 011110) must be roughly the same as for a true source of randomness

⚠ passing these tests does not guarantee security!

Practical Randomness Tests

NIST SP 800-22 "Statistical Test Suite" lists 15 different tests

- Frequency test
  - Number of 0s and 1s must be approximately equal
- Runs test
  - Run: uninterrupted sequence of identical bits (e.g., 1000001 or 01111110)
  - The distribution of the run lengths must be roughly the same as the distribution for a true randomness source
- Maurer's universal statistical test
  - Considers the distance (i.e., number of bits) between matching patterns (i.e., between identical subsequences)
  - Distribution must be approximately the same as for a true randomness source
  - Detects if a sequence can be compressed, which would contradict randomness

Linear Congruential Generator

- Generates a sequence of numbers X_1, X_2, ... from a seed number X_0

$$X_{n+1} = a \cdot X_n + b \mod m$$

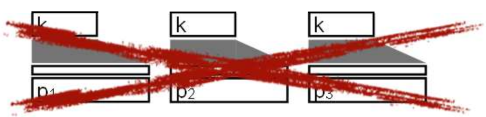
- Parameters a, b, and m must be carefully chosen
  - a = b = 1 → sequence 0, 1, 2, 3, …
  - a = 7, b = 0, m = 32, and X_0 = 7 → sequence 7, 17, 23, 1, **7**, …
  - if m is prime and b = 0 (and a is properly chosen), then the sequence length is m – 1.
    - *example*: $X_{n+1} = 7^5 \cdot X_n \mod (2^{31} - 1)$
- Passes many statistical tests, but provides no security
  - if the attacker knows the parameters, it can predict the sequence from one observed value.
  - parameters can be efficiently computed from three observed values.

Designing Secure Stream Ciphers

- Cryptanalytic attacks
  - The attacker may know some bits of the plaintext → may know some bits of the sequence.
  - *Goal:* The bit sequence must be indistinguishable from true randomness.
  - Security requirements
    1. Uniform distribution: frequency of 0s and 1s is approximately equal in the generated sequence
    2. Independence: no subsequence can be inferred from another, disjoint subsequence
- Brute-force attacks
  - N-bit long key can take 2^N different values → attacker may try all of them.
  - Since 2014, NIST has recommended at least **112-bit** keys for encryption.
  - As computers get faster over time, key sizes must be increased.
- Key re-use: *can we use the same key to encrypt multiple plaintexts?*

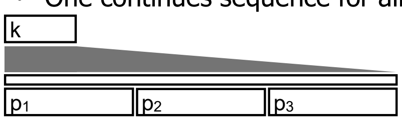
Key Reuse

- Naive approach – Many-time pad



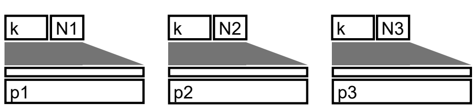
  - attacker learns p1 ⊕ p2, p2 ⊕ p3, p1 ⊕ p3, etc.
  - if the attacker knows one plaintext, it can recover the others

- One continues sequence for all plaintexts



  - stream cipher must support seeking to any position in the sequence
  - otherwise, full sequence must generated from beginning to decrypt p3

- Nonce



  - nonce: number used once
  - mix (*e.g.*, prefix or XOR) key with each nonce → effectively different key for each plaintext

Integrity Attacks

Original Plaintext:	Y	E	S
Binary Representation:	01011010	01000101	01010011
A Pseudorandom Seq.:	11010010	00100000	11110101
Original Ciphertext:	10001011	01100101	10100110
Modified Ciphertext:	10011100	01101111	11010100
Pseudorandom Seq.:	11010010	00100000	11110101
Binary Representation:	01001110	01001111	00100001
Modified Plaintext:	N	O	!

Next Topic

- Stream Cipher
- Stream and Block Ciphers
- Block Cipher Modes of Operation