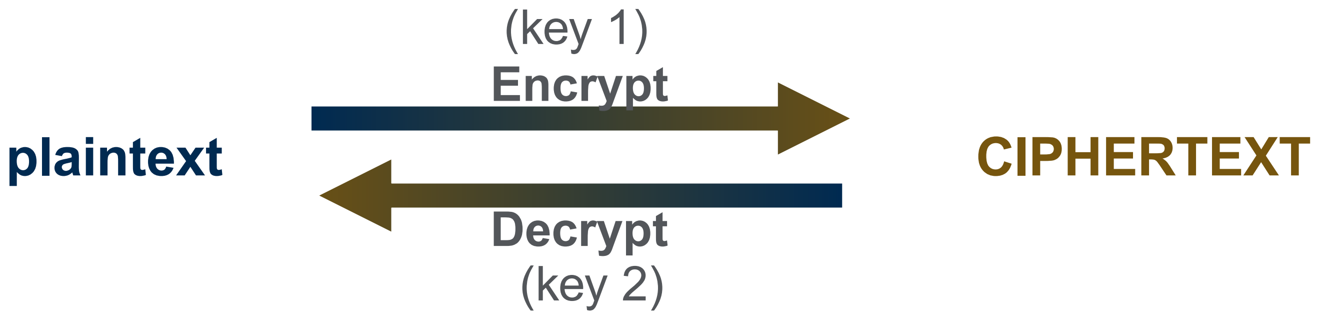
SJSU SAN JOSÉ STATE UNIVERSITY

Lesson 2 – Stream Ciphers

Yan Chen

CS166 Fall 2024

- Terminologies
  - Cryptography: making “secret codes” (secret message)
  - Cryptanalysis: breaking “secret codes”
  - Cryptology: making and breaking “secret codes”
  - Crypto: a synonym for any of the above and more!
- Cipher system (cryptosystem)

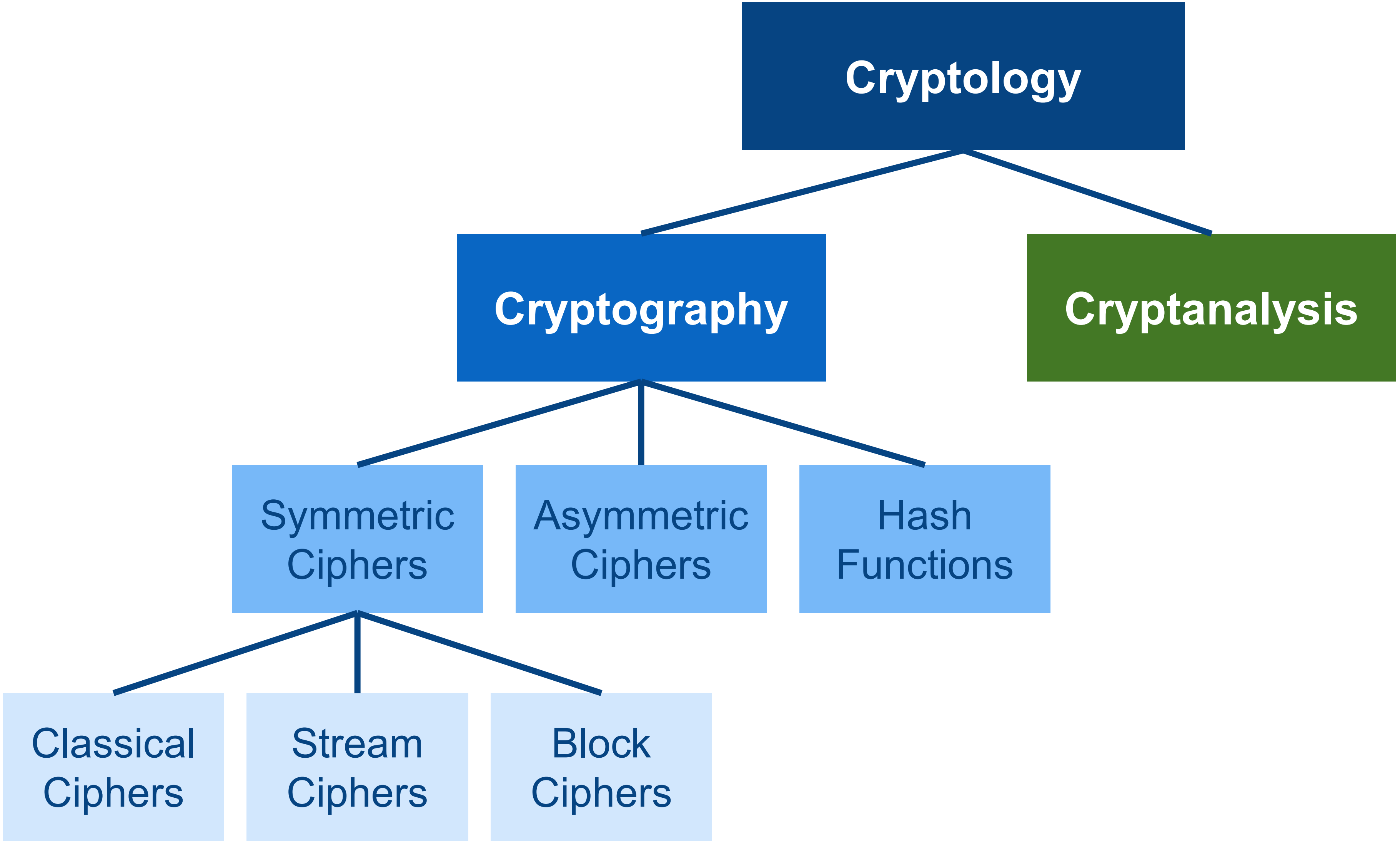


  - Key used for encryption and decryption can be different (Symmetric cipher vs. Asymmetric cipher)

- Kerckhoffs' principle: the strength of a cryptosystem depends ONLY on the key
  - ➢ Trudy only doesn't know the key (and of course, the plaintext)
- More terminologies...
  - ➢ Keyspace: the set of all possible values of the key
  - ➢ Exhaustive key search (brute-force attack): check the whole keyspace – always available to Trudy!
  - ➢ Definition for "secure":

    A cipher system is secure if best know attack is to try all keys

    A cipher system is insecure if any shortcut attack is known


Cryptology
Cryptography
Cryptanalysis
Symmetric Ciphers
Asymmetric Ciphers
Hash Functions
Classical Ciphers
Stream Ciphers
Block Ciphers

- Simple substitution: substitute another character for each character in plaintext
  - (Simplest) Caesar cipher: shift left by alphabet by 3
  - Parameterize the key: key ∊ {1, 2, …, 25}
  - General: any permutation of letters
- Cryptanalysis
  - Caesar: once algorithm known, key is known (too weak!)
  - Parameterized: 25 attempts (worst); 13 attempts (average)
  - General: $26! \approx 2^{88}$ attempts (worst); 2^{87} attempts (average)

    Or statistical attack: English letter frequency!

- (Simplified) Double transposition cipher
  - Put plaintext into a matrix and permutate the rows & columns
  - Key is the matrix size and permutations
  - Pros: hide the statistic ("diffusion")

    Several letters is substituted for the same letter
  - Cons: only shuffle the order, NOT disguise the letters
- One-time pad: key only used once, and use XOR
  - Preparation: encode each letter to a binary string
  - Key generation: random string of bits with the same size of the encoded plaintext

- One-time pad (continued)
  - Encryption: CIPHER = plain ⊕ key, then decode to text
  - Decryption: plain = CIPHER ⊕ key, then decode to text
  - Pros: provably secure (if random & one-time)
  - Cons: not practical (key too large)
- Codebook: dictionary-like book filled with "codewords"
  - Words (plaintext) and corresponding codewords (ciphertext)
  - The code book itself is the key
  - Security depends on the physical security of the codebook
  - Usually use with "additive": new cipher = old cipher + MI

Why Stream Cipher

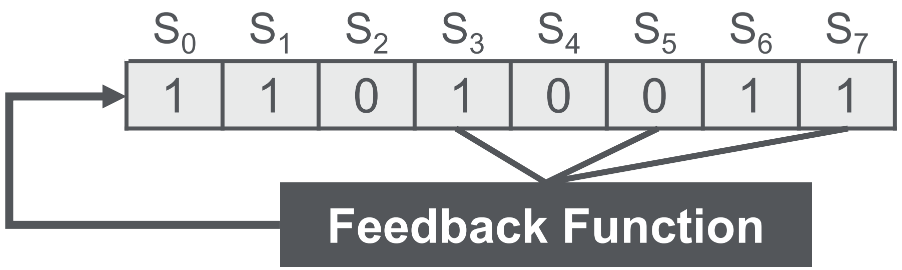
- Recall: Stream ciphers uses the idea of "one-time pad"
  - ➢ Keep the pros: "random" & one-time key
  - ➢ Solve the cons: use a small key instead of large one...
    And "stretch" the small key to the size needed!
- Keystream K: the "stretched" key
  - ➢ Used just like a one-time pad ($C = p \oplus K$, $p = C \oplus K$)
- Algorithm to generate keystream from the key is the "heart" of each stream cipher
  - ➢ More random, more like one-time pad, the better!

Provably secure? Or when it will be insecure?

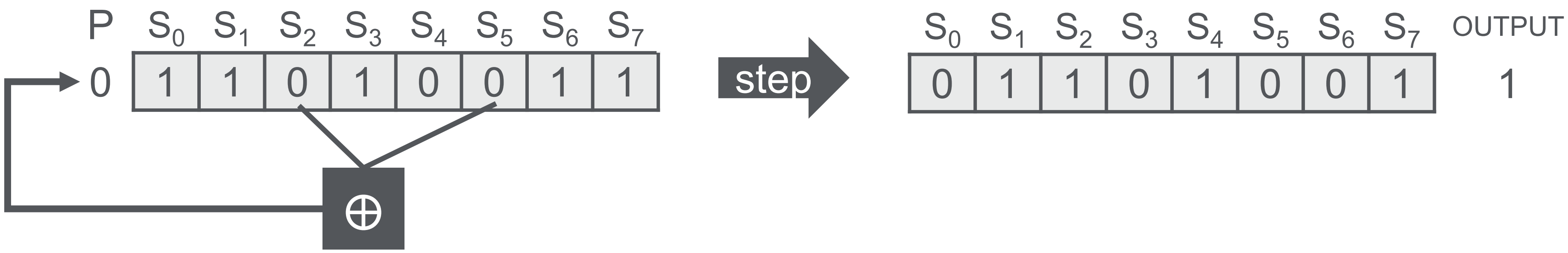
- Taxonomy: Stream ciphers are symmetric-key ciphers
- Stream ciphers were the king of crypto
  - Since it's efficient in hardware
  - Today, hardware gets larger and cheaper...more applications can be handled in software since processors are fast
  - So not as popular as block ciphers now
- Stream ciphers to be covered
  - A5/1: based on shift registers, used in GSM mobile phone
  - RC4: Based on a changing lookup table, used many places
  - Others (reading assignment 1): ORYX & PKZIP

- A5/1: a stream cipher using shift register
  - Used in for GSM (Global System for Mobile Communication), a standard to describe the protocols for 2G cellular networks
  - Efficient in hardware, slow in software (was popular)
- At the beginning, Kerckhoffs' principle was violated...
  - They tried to keep this algorithm secret...
  - But as usual, the secret was reverse-engineered and leaked!
- Uses 3 linear feedback shift registers (LFSR)
  - Key is the initial fill of the registers
  - Keystream generation based on XOR and majority function

- Shift register: a sequence of n bits
  - A temporary memory initialized with an n-bit binary number
  - Can shift left or right
  - Index starts from 0 (can kind of view it as an array)
- Feedback shift register: feedback based on the contents of the shift register and input it into the shift register
  - Leftmost bit calculated by a "feedback function" that takes in bits of the shift register ("taps") and returns a new bit



- Linear feedback shift register (LFSR): the feedback function is linear (only XOR operation is used)
- Step: an operation in feedback shift registers
  - Feedback function f produces a single bit P
  - P becomes the leftmost bit, and all contents shift 1 bit to right
  - Rightmost bit is shifted off (sometimes is taken as output)
  - Example: suppose taps are S_2 & S_5, and $f = S_2 \oplus S_5$



- Application of LFSR: pseudo-random binary generator
  - Pseudo-random: seems random but NOT truly random
- The pattern of the numbers generated are deterministic
  - If initialize the register with the same binary number (“seed”), the same sequence of numbers will be produced
- An n-bits register can have at most 2^n different binaries
  - If at any point, the content is all zero, then ... the register will be all zero at every subsequent step
  - Therefore, the upper-bound of the period length of LFSRs is $2^n – 1$, where n is the number of bits

- A5/1 uses LFSR to generate keystream
  - X: 19 bits, feedback function $F_X = x_{13} \oplus x_{16} \oplus x_{17} \oplus x_{18}$
  - Y: 22 bits, feedback function $F_Y = y_{20} \oplus y_{21}$
  - Z: 23 bits feedback function $F_Z = z_7 \oplus z_{20} \oplus z_{21} \oplus z_{22}$
  - What is the size of the key?
- Whether the register steps or not, based on the output of the majority function m = maj(x, y, z)
  - If more 1's than 0's (2/3 are 1), returns 1
  - If more 0's than 1's (2/3 are 0), returns 0
  - Examples: maj(1, 1, 0) = 1; maj(0, 1, 0) = 0

- To generate one bit in the keystream...
  - Calculate $m = \text{maj}(x_8, y_{10}, z_{10})$
  - X steps only if $x_8 = m$
  - Y steps only if $y_{10} = m$
  - Z steps only if $z_{10} = m$
  - Keystream bit = $x_{18} \oplus y_{21} \oplus z_{22}$ (XOR rightmost bit after step)
- Example:
  - Bold red: used for **maj**; bold and underline: **taps**

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
X	1	0	1	0	1	0	1	0	**1**	0	1	0	1	**0**	1	0	**1**	**0**	**1**				
Y	1	1	0	0	1	1	0	0	1	1	**0**	0	1	1	0	0	1	1	0	0	**0**	**1**	
Z	1	1	1	0	0	0	0	**1**	1	1	**1**	0	0	0	0	1	1	1	1	0	**0**	**0**	**1**

- Example (continued)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
X	1	0	1	0	1	0	1	0	**1**	0	1	0	1	**0**	1	0	**1**	**0**	**1**				
Y	1	1	0	0	1	1	0	0	1	1	**0**	0	1	1	0	0	1	1	0	0	**0**	**1**	
Z	1	1	1	0	0	0	0	**1**	1	1	**1**	0	0	0	0	1	1	1	1	0	**0**	**0**	**1**

- $m = maj(x_8, y_{10}, z_{10}) = maj(1, 0, 1) = 1$, so only X and Z step
- After X & Z step...

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
X	0	1	0	1	0	1	0	1	**0**	1	0	1	0	**1**	0	1	**0**	**1**	**0**				
Y	1	1	0	0	1	1	0	0	1	1	**0**	0	1	1	0	0	1	1	0	0	**0**	**1**	
Z	0	1	1	1	0	0	0	**0**	1	1	**1**	1	0	0	0	0	1	1	1	1	**0**	**0**	**0**

- keystream bit generated $= x_{18} \oplus y_{21} \oplus z_{22} = 0 \oplus 1 \oplus 0 = 1$

- RC4: a stream cipher using lookup table
  - Invented by American cryptographer, Ronald Rivest in 1978
  - Also called “Ron’s Cipher” or “Ron’s Code”
  - Is remarkably simple, lived longer than A5/1
  - Was used in SSL, WEP, and WPA
  - Was broken but may still in use for non-security critical apps
- Each step of RC4 produces a byte (= 8 bits)
  - Efficient in software
  - vs. A5/1 –each step produces a bit: efficient in hardware
- Key size various: from 1 to 255 bytes

- RC4 uses the 2 arrays of unsigned 8-bit numbers
  - key[N]: stores the N bytes key (N from 0 to 255)
  - S[256]: stores the permutations of numbers from 0 to 255

    It's a self-modifying lookup table: change after each step
- Initialization (suppose key has N bytes)
  - Initialize arrays:

```
for i = 0 to 255
        S[i] = i
        K[i] = key[i mod N]
next i
```

  - Initial Permutation of S:

```
j = 0
for i = 0 to 255
        j = (j + S[i] + K[i]) mod 256
        swap(S[i], S[j])
next i
```

- Then, to generate a byte...
  - ➢ Swaps elements in current lookup table (i= 0, j = 0 initially)

```
i = (i + 1) mod 256
j = (j + S[i]) mod 256
swap(S[i], S[j])
```

  - ➢ Selects a keystream byte from table

```
t = (S[i] + S[j]) mod 256
keystreamByte = S[t]
```

- Block ciphers
  - Feistel Cipher
  - DES
  - AES
  - TEA

Concepts

Exercises

- Keystream
- A5/1
  - Shift register, feedback shift register, step, linear feedback shift register
  - Majority function
- RC4

- For majority function maj(x, y, z), ...
  - Write the truth table for this function
  - Derive the boolean function that is equivalent
- Most of the stream ciphers are efficient in hardware only. Why RC4 is also efficient in software?
- Find an upper bound for the number of different states that are possible for...
  - A5/1 cipher (Hint: how many possible states for each LFSR?)
  - RC4 cipher (Hint: it consists of a lookup table S, and two indices i and j. Count the number of possible distinct tables S and the number of distinct indices i and j, then compute the product of these numbers)
  - Why is the size of the state space important to know?

References

- Stamp, Mark and Low, Richard M., "Applied Cryptanalysis: breaking ciphers in the real world," John Wiley & Sons, Inc., New Jersey, USA, 2007
- Stallings, William, "Cryptography and Network Security, Principles and Practice, 6th ed.," Pearson, USA, 2014
- Paar, Christof, "Understanding Cryptography," Faller, Berlin, Germany, 2010
- Stamp, Mark, "Information Security, Principles and Practice, 2nd ed.," Wiley, New Jersey, USA, 2011