CS326 – Systems Security

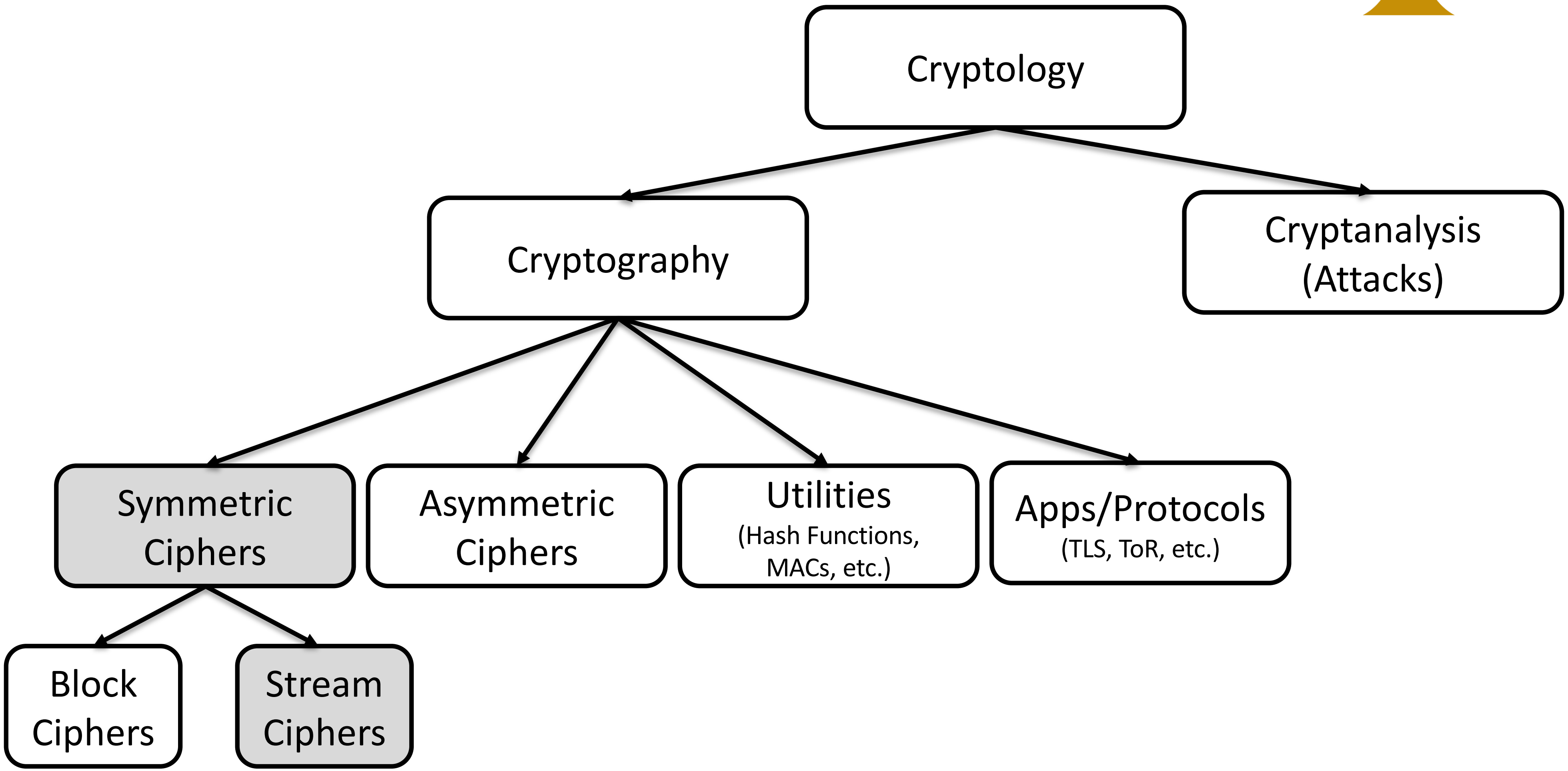
Lecture 6
Stream Ciphers

Elias Athanasopoulos
athanasopoulos.elias@ucy.ac.cy

Sections of this Lecture

- One-Time Pad
- Stream Ciphers
- RC4

Cryptography Roadmap



ONE-TIME PAD

Recall: Vigenère Cipher

```
Key = r, u, n (three Caesar’s keys)

tobeornottobethatisthequestion
runrunrunrunrunrunrunrunrunrun
KIOVIEEIGKIOVNURNVJNUVKHVMGZIA
```

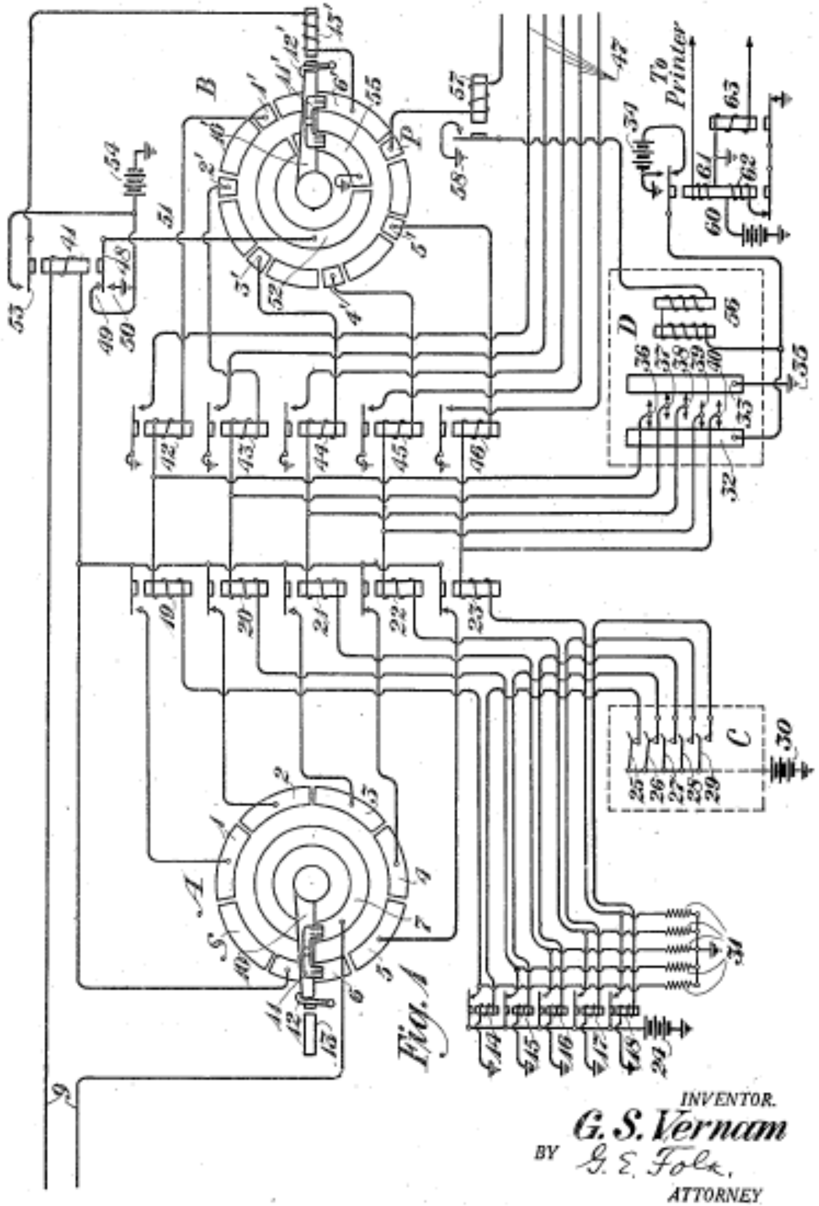
Vernam Cipher

- Use a longer key
- Instead of Caesar cipher, apply a XOR operation between plaintext and key
  $c_i = p_i \oplus k_i$
- For decryption, apply the XOR operation with the same key
  $p_i = c_i \oplus k_i$

G. S. VERNAM.
SECRET SIGNALING SYSTEM.
APPLICATION FILED SEPT. 13, 1918.

1,310,719. Patented July 22, 1919.

2 SHEETS—SHEET 1.



INVENTOR.
G. S. Vernam
BY G. E. Folk,
ATTORNEY

One-Time Pad

- Special case of Vernam cipher
  - The size of the key is the size of the plain text
  - The key is random
  - The key is used only for the transmission of **a single** message

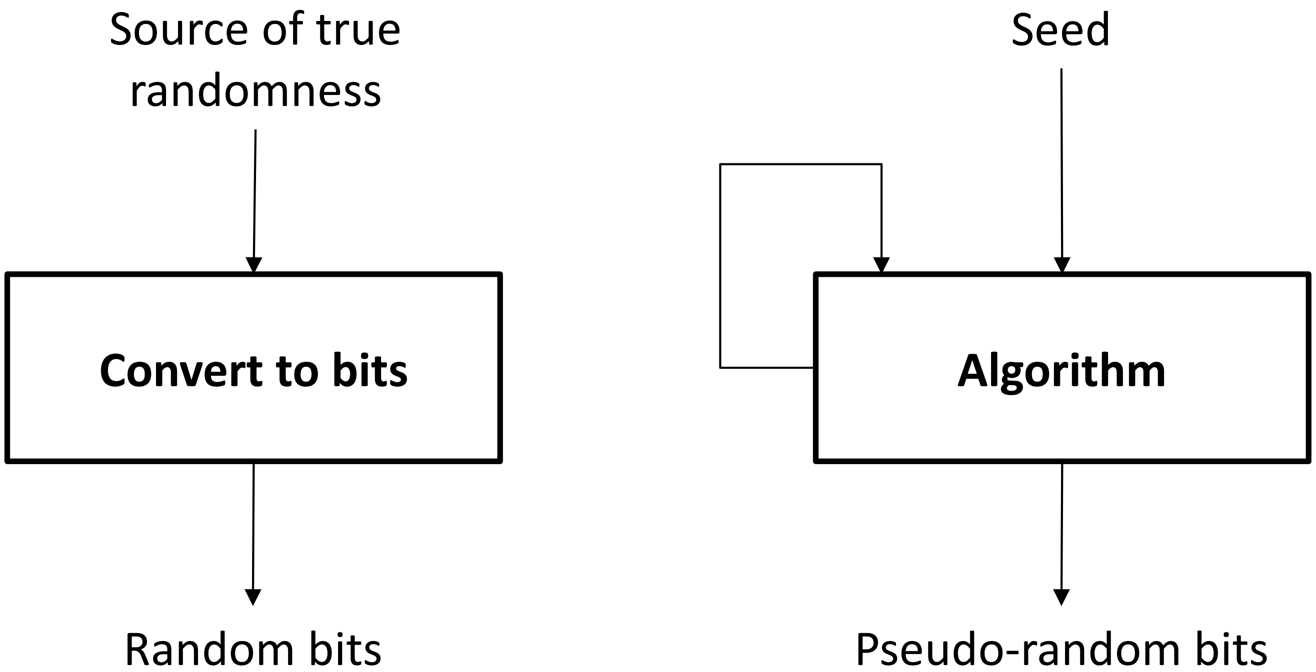
One-time Pad Properties

- Unconditional Security
  - A cryptosystem is unconditionally or information-theoretically secure if it cannot be broken even with infinite computation resources
- Impractical long key
- Key integrity
  - Given a cipher you can select another key that produces a different valid plain text
- Message Integrity
  - Given a key you can select a cipher text that produces the desired plain text

STREAM CIPHERS

Random Number Generator Types

- True Random Number Generators (TRNGs)
- Pseudo-random Number Generators (PRNGs)



True Random Number Generator (TRNG)

- Flipping a coin
- Collecting environmental noise
- Involving non-deterministic activities
  - Users typing in the terminal, I/O timeouts, etc.
- Expensive source of randomness

Pseudo Random Number Generator

- Typically, output stream has good statistical properties
- Output can be reproduced and can be predicted

$X_{n+1} = (aX_n + c) \bmod m$, *$X_0$ is the seed (assume X_0=1)*

Selection of **a**, **c**, and **m**, is critical

- a=7, c=0, m=32

  7, 17, 23, 1, 7, ...
- a=5

  5, 25, 29, 17, 21, 9, 13, 1, 5, ...

Predicting the Output

- Simple PRNG

  $S_0 = seed$
  $S_{i+1} = AS_i + B \bmod m$

- Assumptions
  - Unknown A, B and S_0 as key, but m is known
  - Size of A,B and S_i to be 100 bit
  - 300 bit of output are known, i.e. S_1, S_2 and S_3
  - Attacker knows 300 bit of both plaintext/ciphertext (i.e., 300 bit of a known file's header) e.g. for a pdf, the header is known
- Solution

  $S_2 = AS_1 + B \bmod m$

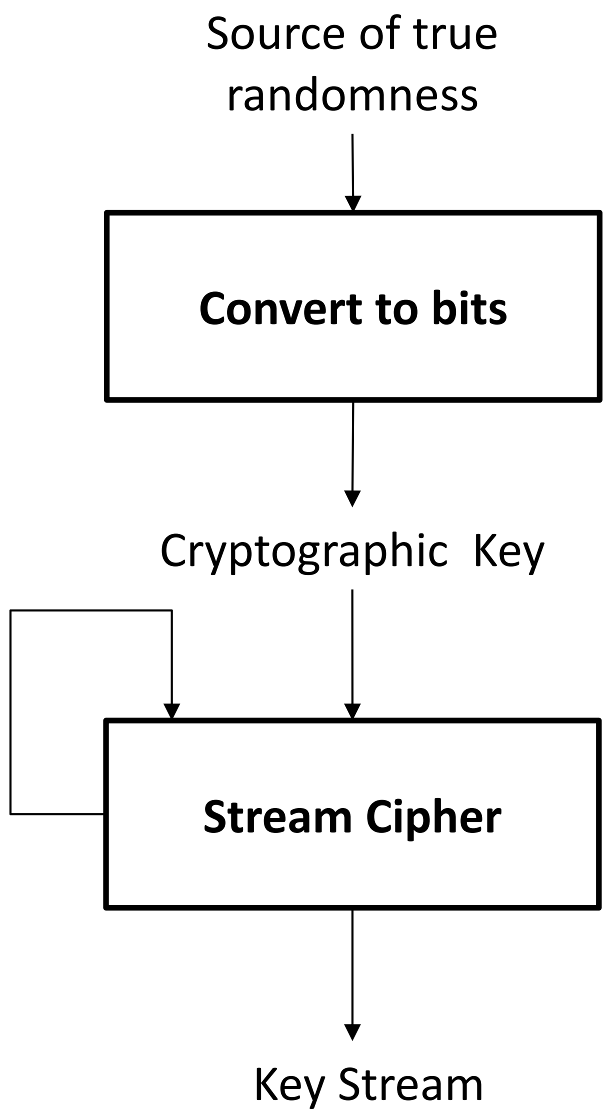
  $S_3 = AS_2 + B \bmod m$

  A and B can now be calculated

Cryptographically Secure Random Number Generators

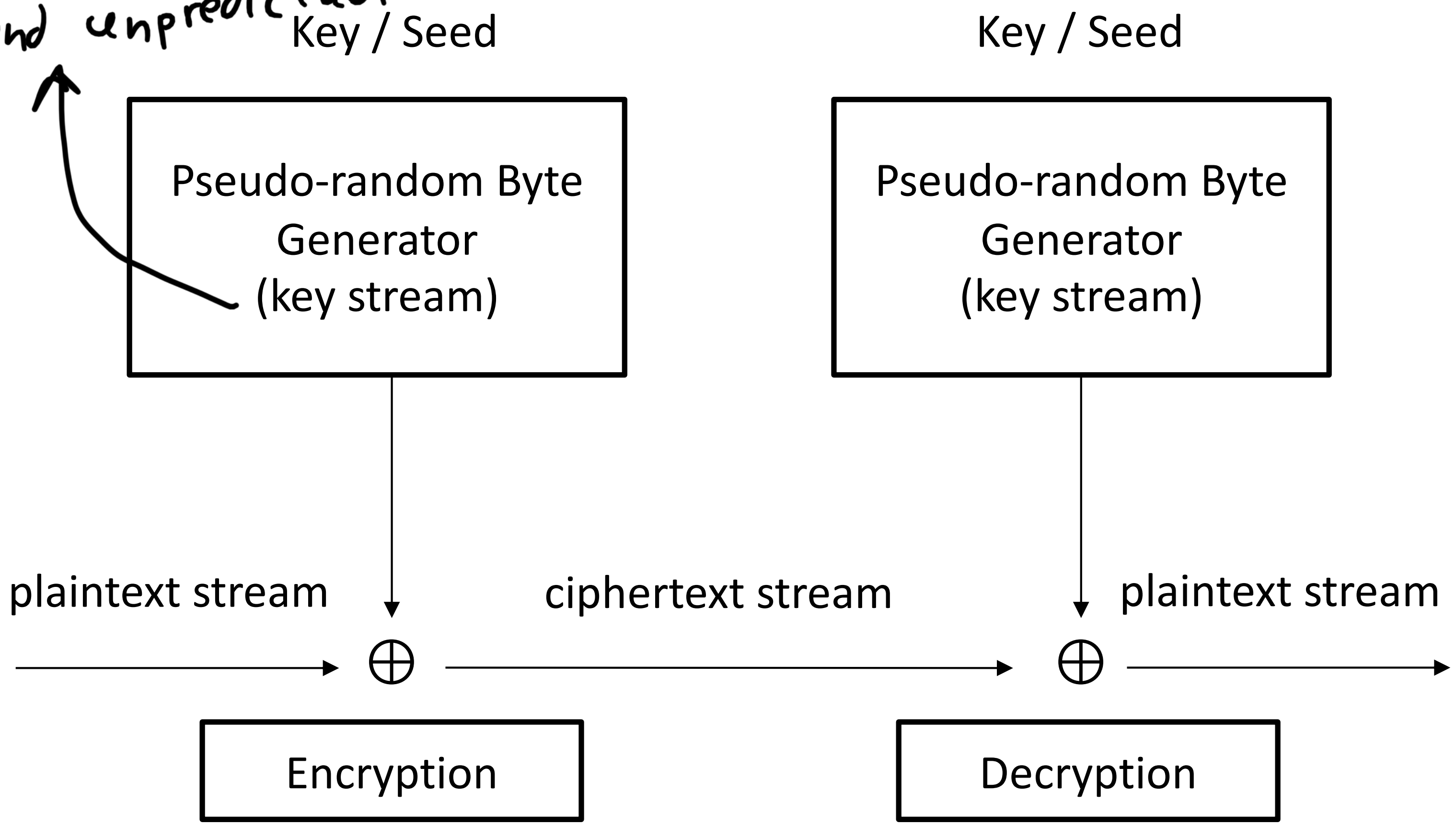
- Special PRNG with additional property
  - Output must be **unpredictable**
- Given *n* consecutive bits of output s_i, the following output bits s_{n+1} cannot be predicted (in polynomial time)
- **Unpredictability** is useful only in cryptography whereas many, many (technical) systems need PRNGs

Key Stream

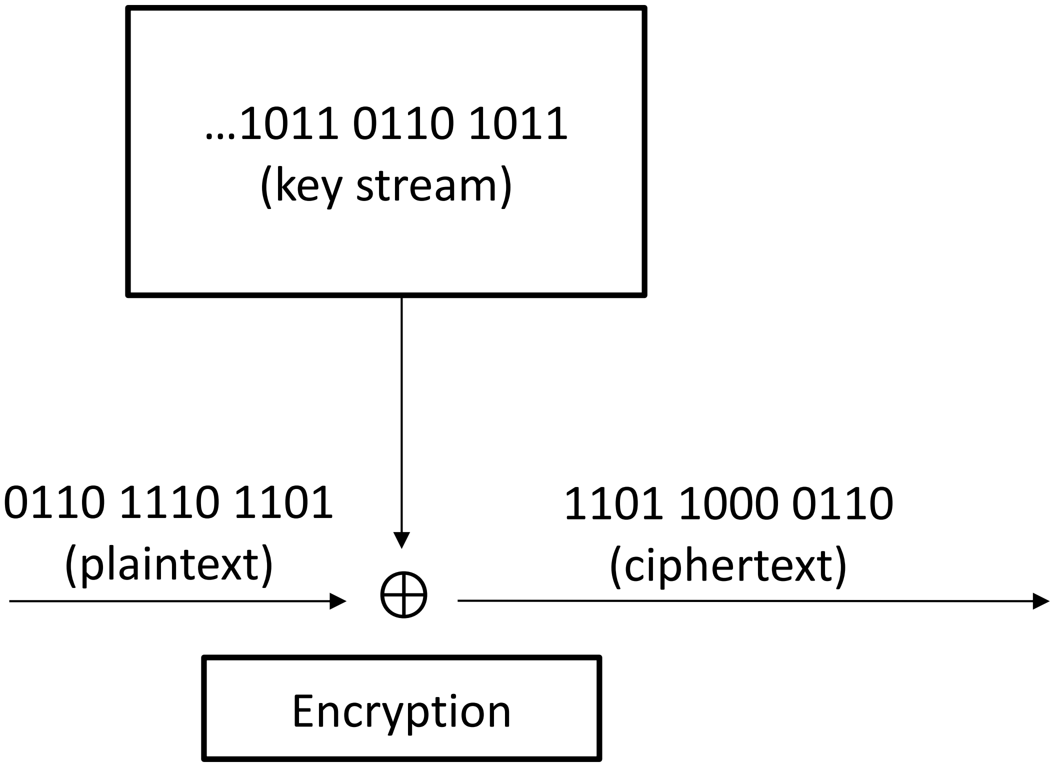


Source of true randomness
Convert to bits
Cryptographic Key
Stream Cipher
Key Stream

Each key stream is different and unpredictable



Example



…1011 0110 1011
(key stream)
0110 1110 1101
(plaintext)
⊕
1101 1000 0110
(ciphertext)
Encryption

Performance

Cipher	Key Length	Mbit/s
DES	56	36.95
3DES	112	13.32
AES	128	51.19
RC4 (Stream Cipher)	(chosen)	**211.34**

Source: Zhao et al., Anatomy and Performance of SSL Processing, ISPASS 2005

Rivest

RC4

RC4

- Designed by Ron Rivest in 1987
- Used today in TLS
  - TLS is the cipher suite behind HTTPS
- Used in WEP
  - Got broken
- There are concerns about the security of RC4
- Based on random permutations
- Period is believed to be greater than 10^{100}
- 8 to 16 machine operations are required per byte of the ciphertext

RC4 – Initialization

```
/* Initialization */
for i = 0 to 255 do
    S[i] = i;
    T[i] = K[i mod keylen];
/* Initial Permutation of S */
j = 0;
for i = 0 to 255 do
    j = (j + S[i] + T[i]) mod 256;
    Swap (S[i], S[j]);
```

is filled with the key (repeated)

key

after loop: T : [K[0], K[1], ...

K[0], K[1]...]

S: [0, 1, ..., 255]

Shuffling S

Produce a random permutation of S (i.e., of all numbers from 0 to 255)

RC4 – Stream Generation

```
i, j = 0;
while (true)
   i = (i + 1) mod 256;
   j = (j + S[i]) mod 256;
   Swap (S[i], S[j]);
   t = (S[i] + S[j]) mod 256;
   k = S[t];
```

Each element of S is swapped with another element at least once every 256 iterations.

k
⊗
p (first byte of plaintext)
= first byte of ciphertext

Then, for all the other bytes repeat the process

Encryption: XOR the next byte of plaintext with **k**
Decryption: XOR the next byte of ciphertext with **k**

RC4 Security

- Optional Reading
  - **On the Security of RC4 in TLS.** Nadhem AlFardan, et al. *In Usenix Security 2013.* https://www.usenix.org/conference/usenixsecurity13/technical-sessions/paper/alFardan

Resources

- This lecture was built using material that can be found at
  - Chapter 5, Handbook of Applied Cryptography, http://cacr.uwaterloo.ca/hac/
  - Chapter 5, Serious Cryptography, https://nostarch.com/seriouscrypto
  - Chapter 2, Understanding Cryptography, http://www.crypto-textbook.com