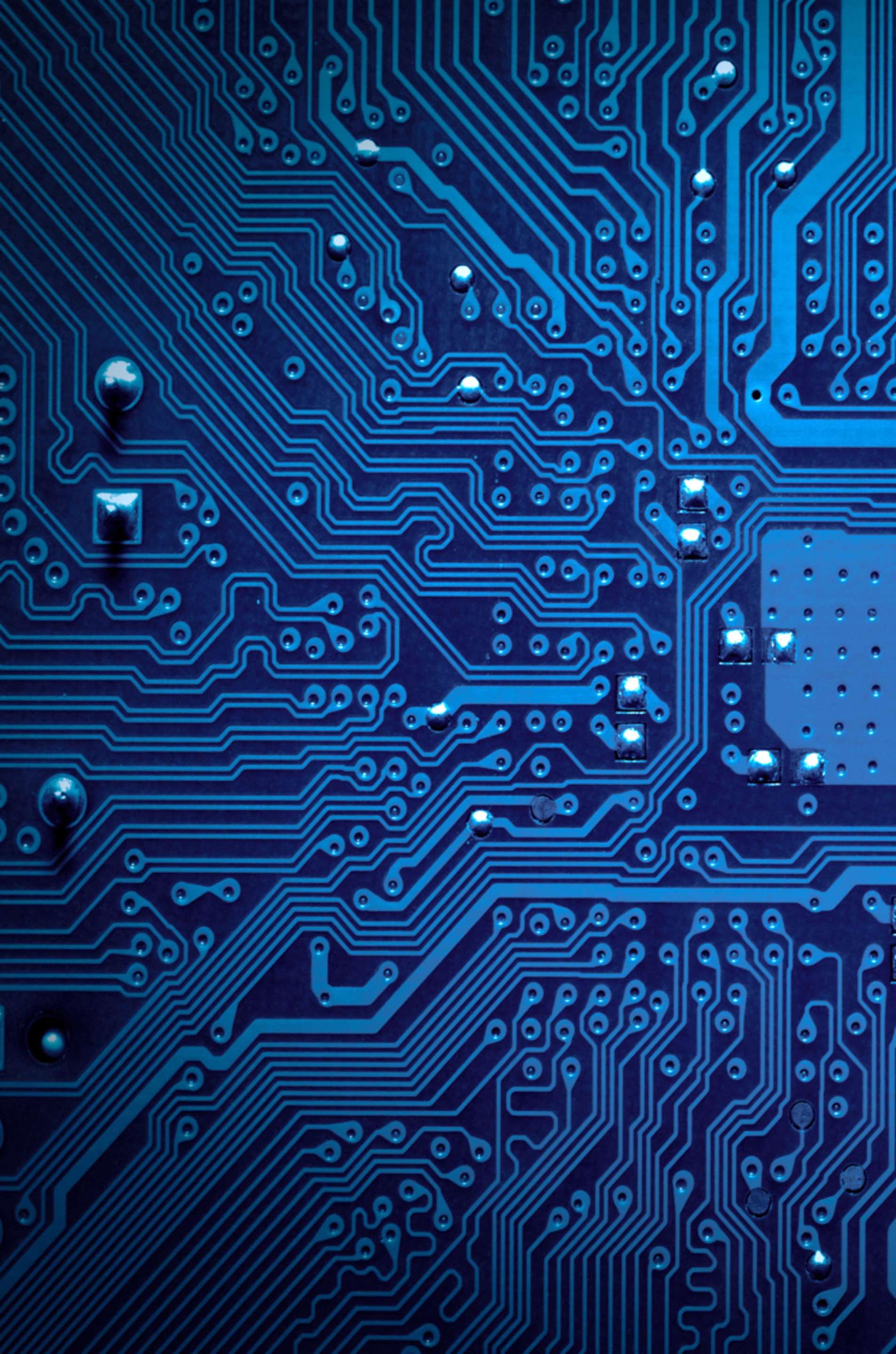


Stream Ciphers

ECE568 – Lecture 9

Courtney Gibson, P.Eng.
University of Toronto ECE

Outline

Introduction to Stream Ciphers

- Keystream

Synchronous & Self-Synchronizing Ciphers

Stream Cipher Properties

- Performance
- Security
- Error Recovery

Stream Cipher Implementations

- RC4
- SEAL

Introduction to Stream Ciphers

Stream Ciphers

- Some applications require encryption or decryption to be performed with low latency
  - *e.g.*, audio or video streams
- **Stream ciphers** are well suited for such applications because they operate a bit or byte at a time
  - Produce ciphertext exactly as long as plaintext
  - Unlike block ciphers that encrypt a block at a time and then use different modes to improve security, stream ciphers have **no modes** because they can be used to encrypt arbitrarily long messages

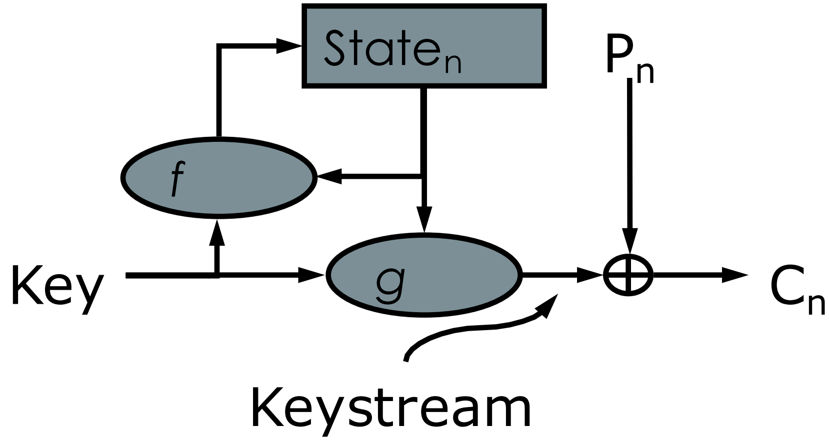
Stream Ciphers

- Stream ciphers are closely related to one time pads
- However, the pad (called the **keystream**) is a pseudo-random sequence of bits generated from a much shorter key
- The stream of random bits is then used in place of the one time pad and XOR'ed with the plain text
- There are two types of stream ciphers
  - Synchronous Stream Ciphers
  - Self-Synchronizing Stream Ciphers

Synchronous and Self-Synchronizing Stream Ciphers

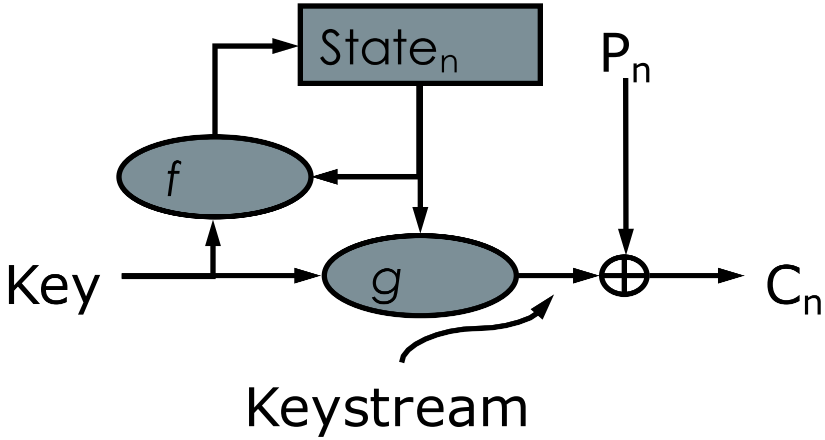
Synchronous Stream Ciphers

- The keystream is **independent** of the message text
- The State is modified by the function **f** and the key
- Each step uses **feedback** in which **f** takes the current state to produce the new state



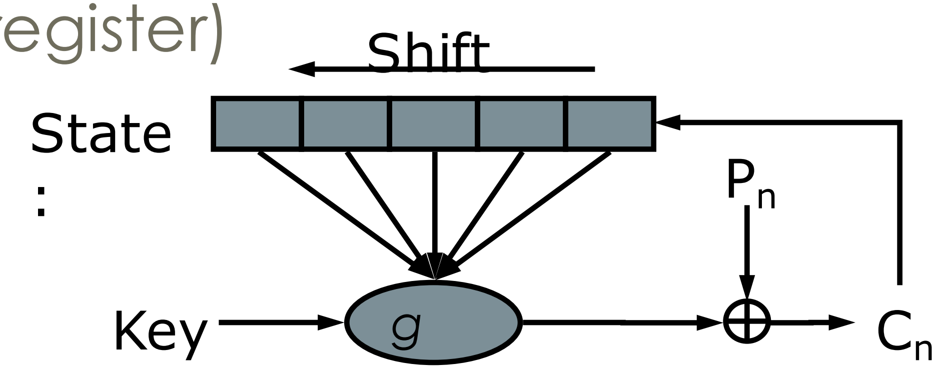
Synchronous Stream Ciphers

- Encryption XORs the keystream with plaintext
- Decryption uses the key to produce the same keystream, and XORs the keystream with the ciphertext to recover plaintext
- The initial state is often referred to as an **IV**, just like in block ciphers



Self-Synchronizing Stream Ciphers

- The keystream depends on the plaintext
- The state consists of a **shift register**
- Every ciphertext bit created is shifted into the shift register and fed back as an input into **g**
- Thus each ciphertext bit has an effect on the next **n** bits (where **n** is the length of the shift register)



Stream Cipher Properties

Stream Cipher Properties

Security

- In general stream ciphers have similar properties to OTP (One-Time Pad)
- It is dangerous to use the same keystream to encrypt two different messages
  - For synchronous ciphers: key or IV must be changed for new message
  - For self-synchronizing ciphers: insert random data at the beginning
    - Why can't we insert random data for sync ciphers?
- They are malleable (*i.e.*, ciphertext can be changed to generate related plaintext)
- With self-synchronizing ciphers, an adversary can **replay** previously-sent ciphertext into a stream, and the cipher will resync

Stream Cipher Properties

Performance

- In general stream ciphers have better performance than block ciphers
- This is especially true for hardware implementations
- The keystream for synchronous stream ciphers can be pre-computed before the message arrives so encryption/decryption is simply an XOR

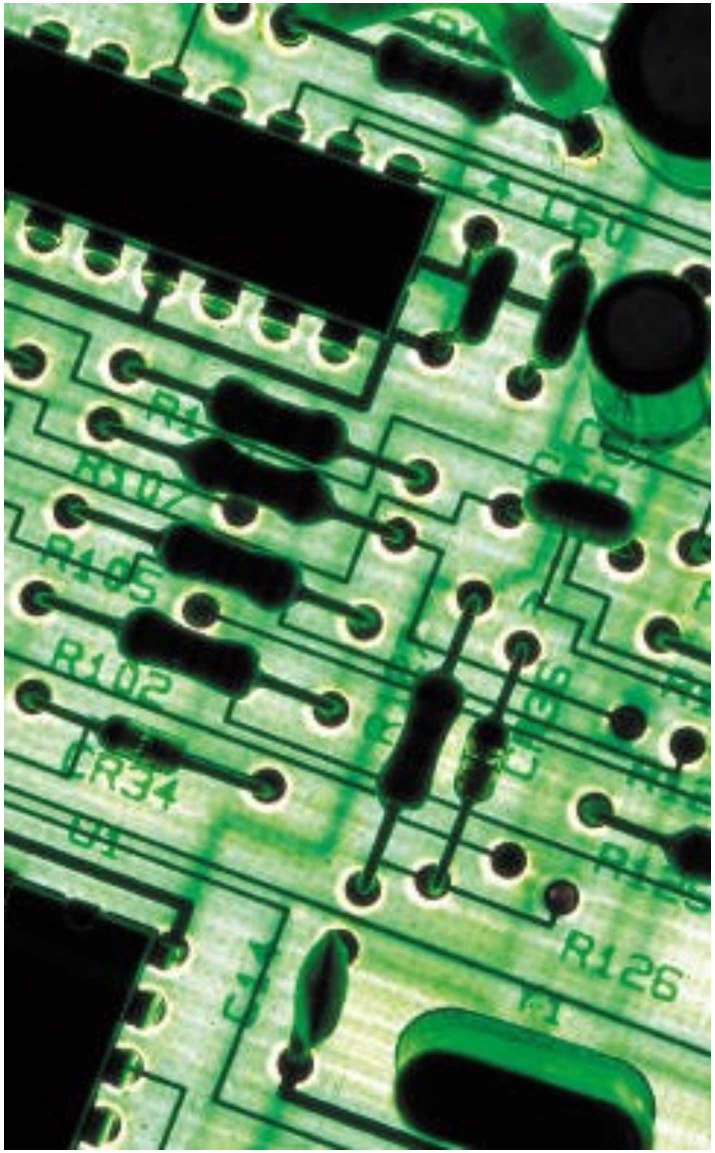
Stream Cipher Properties

Error Propagation

- For synchronous stream ciphers, a transmission error only affects the corresponding plaintext bits
- For self-synchronizing ciphers, the error will affect the next **n** bits (where **n** is the size of the shift register)
  - After that the affected bit gets shifted out of the register

Error Recovery

- For synchronous stream ciphers, if a section of cipher text is lost, the ciphertext stream and keystream become “out of sync”, and recovery is impossible unless we know exactly how much ciphertext is lost
- Self-synchronizing stream ciphers will recover after **n** bits have passed



Stream Cipher Implementations

RC4 and SEAL

Stream Ciphers: RC4 and SEAL

RC4 ("Ron's Code") was a proprietary stream cipher created by Ron Rivest of RSA Labs

- It is probably the most commonly used stream cipher in the public domain
- Its algorithm is now publicly known, but a license from RSA is still required to use it
- Its algorithm is very simple and offers good performance in software

SEAL is a stream cipher owned by IBM

- SEAL is optimized for software performance
- Has an interesting property that it can easily generate arbitrary portions of the keystream without having to start from the beginning

RC4 Implementation

- S is an array of size 256 that contains the state
- Always contains a permutation of 0…255
- keylength is generally 5-16 bytes
- Key scheduling algorithm initializes state S
- PRGA generates keystream

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

Key Scheduling Algorithm

```
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
```

Pseudo-Random Generation Algo

Selecting the Right Cipher

- While stream ciphers offer better performance, they are difficult to use safely
  - Ciphers are either vulnerable to replay or IV's need to be managed never to repeat
    - *e.g.*, WEP used RC4 but the IV was too short
  - Repeating IVs is more damaging than with a block cipher that uses CBC
- Block ciphers are easier and more commonly used
  - There is no reason to use DES anymore except backwards compatibility: use AES
  - CBC is most common encryption mode for arbitrary data
  - ECB is safe to use on short pieces of data where plain text blocks are unlikely to repeat (*e.g.*, passwords, keys, etc…)

Cipher Performance

- Throughput test on a 2.4 GHz Pentium 4

Throughput in kilobytes/second

Block length	16 bytes	64 bytes	256 bytes	1024 bytes	8192 bytes
rc4	80151.38k	87598.19k	89158.83k	89548.97k	89437.53k
aes-128 cbc	49921.99k	51200.70k	52822.53k	53381.46k	53627.56k
aes-192 cbc	44962.74k	45100.27k	46442.23k	46841.17k	47091.67k
aes-256 cbc	40476.10k	40809.45k	41571.00k	41902.08k	41885.70k

- Things to note
  - RC4 has much better throughput
  - Throughput increases with block length
  - AES-192 and AES-256 are slow because of additional rounds

Questions?