CS 4770: Cryptography

CS 6750: Cryptography and Communication Security

Alina Oprea
Associate Professor, CCIS
Northeastern University

January 25 2018

Review

- Pseudorandom generators (PRG)
  - Computationally indistinguishable output from random
  - Security definition
  - Examples
- EAV-secure encryption
  - Construction from PRG
  - Shorten key in OTP
  - Reduction proof

Outline

- Stream cipher definition
- Constructions
  - LFSR
  - RC4
  - Salsa20
- Attacks on implementations and protocols
  - Two-time pad
  - Malleability

Stream ciphers vs Block ciphers

- Stream ciphers
  - Encrypt variable-length messages to variable-length ciphertexts
  - Used in practice to instantiate PRG
  - Produce a deterministic string of output bits (encrypt messages on demand)
- Block ciphers
  - Map n-bit plaintext to n-bit ciphertext
  - Random permutation
  - Fixed length

Stream ciphers

- Produce random bits on demand
- Algorithms (Init, GetBits)
- Init
  - Input seed s and optionally initialization vector (IV)
  - Output state s_0
- GetBits
  - Input state s_i
  - Outputs bit y and new state s_{i+1}

Stream ciphers

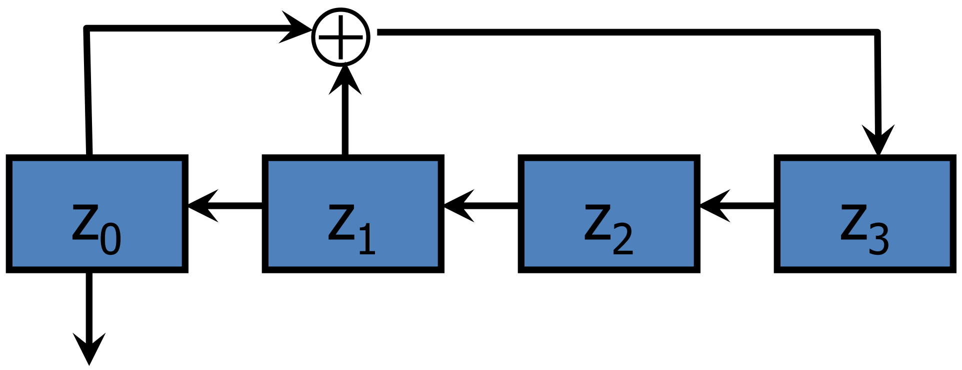
- **Input**: seed s
- **Output:** $y_1, \ldots y_\ell$

- s_0 = Init(s,IV)
- For i = 1 to ℓ
  - (y_i, s_i) = GetBits(s_{i-1})
- Return $y_1, \ldots y_\ell$

- Requirement: output is a pseudorandom generator for any $\ell > n$

Linear Feedback Shift Register (LFSR)

Example:
4-bit LFSR

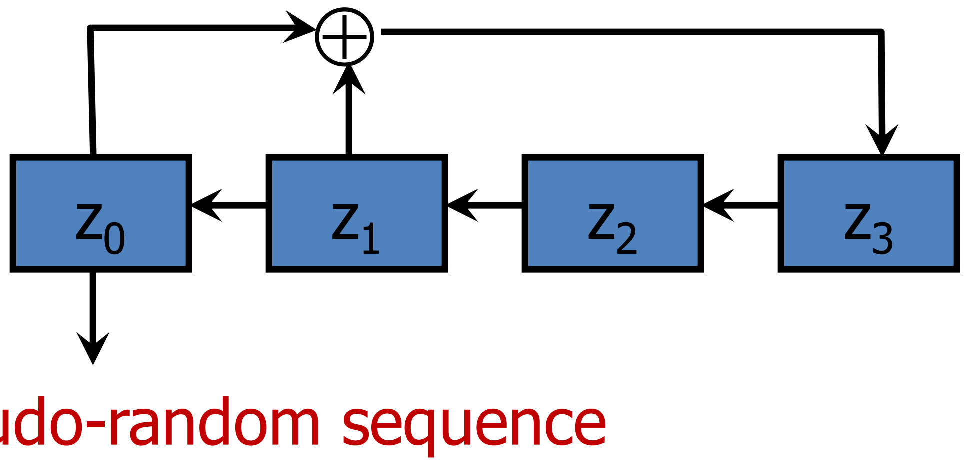


add to pseudo-random sequence

- Key is used as the seed
  - For example, if the seed is 1001, the generated sequence is 1001101011110001001…
- Repeats after 15 bits (2^4-1)

Linear Feedback Shift Register (LFSR)

<u>Example</u>:
4-bit LFSR



- $z_i \quad = z_{i-4} + z_{i-3} \mod 2$

  $= 0 \cdot z_{i-1} + 0 \cdot z_{i-2} + 1 \cdot z_{i-3} + 1 \cdot z_{i-4} \mod 2$
- We say that cells 0 & 1 are selected.
- An L-cell LFSR is *maximum-length* if some initial state will produce a sequence that repeats every $2^L - 1$ bits

Cryptanalysis of LFSR

- Given a 4-stage LFSR, we know
  - $z_4=z_3c_3+z_2c_2+z_1c_1+z_0c_0 \mod 2$
  - $z_5=z_4c_3+z_3c_2+z_2c_1+z_1c_0 \mod 2$
  - $z_6=z_5c_3+z_4c_2+z_3c_1+z_2c_0 \mod 2$
  - $z_7=z_6c_3+z_5c_2+z_4c_1+z_3c_0 \mod 2$
- Knowing $z_0,z_1,\ldots,z_7$, one can compute c_0,c_1,c_2,c_3 by solving the linear system
- In general, knowing 2n output bits, one can solve an n-stage LFSR

Reconstruction attack

LSFR in practice

Typically implemented in hardware

Applications

DVD encryption (CSS):   2 LFSRs

GSM encryption (A5/1,2):   3 LFSRs

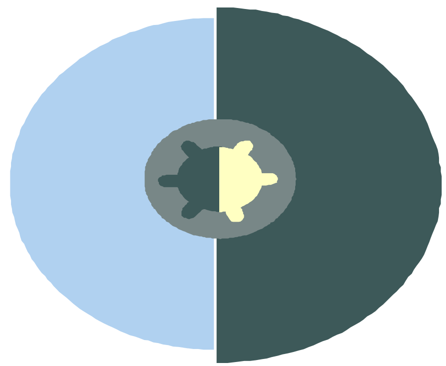
Bluetooth (E0):   4 LFSRs

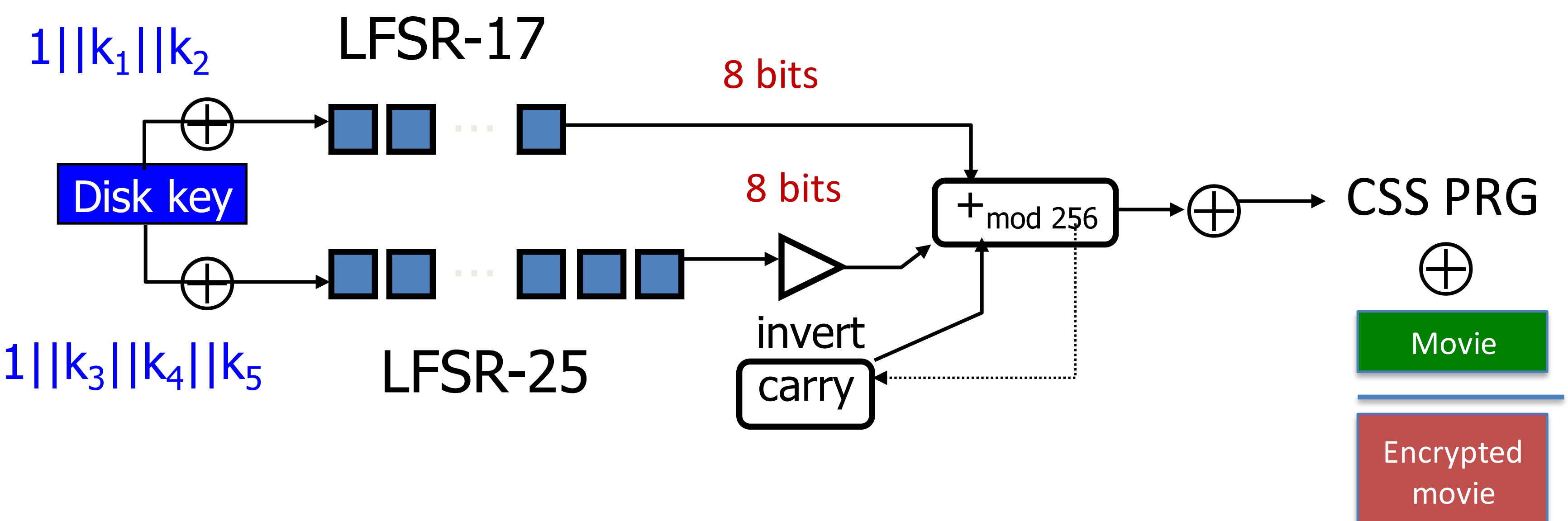
All broken!

Content Scrambling System (CSS)

DVD encryption scheme from Matsushita and Toshiba

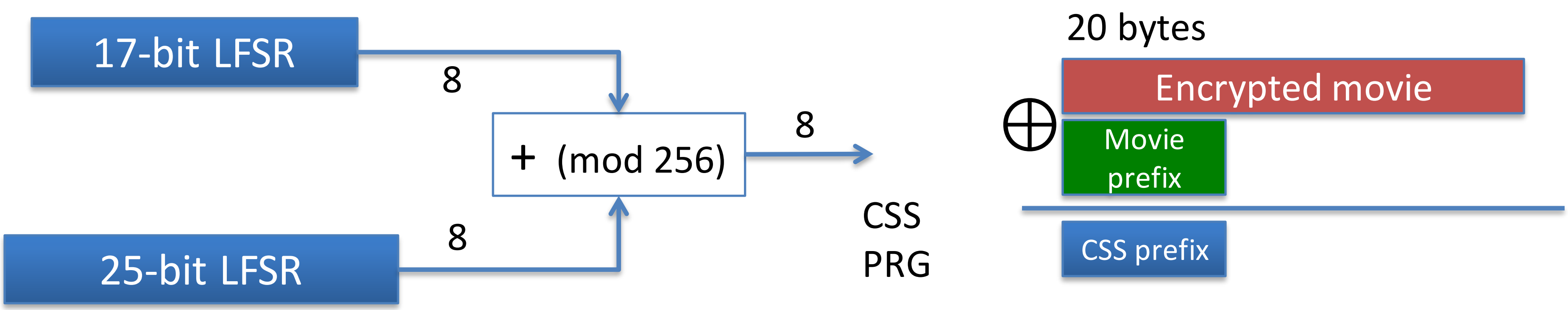
Seed = 5 bytes = 40 bits





Cryptanalysis of CSS

Mpeg files have fixed prefix!



For all possible initial settings of 17-bit LFSR do:

- Run 17-bit LFSR to get 20 bytes of output
- Subtract from CSS prefix $\Rightarrow$ candidate 20 bytes output of 25-bit LFSR
- If consistent with 25-bit LFSR (easy to test), found correct initial settings of both !!

Using key, generate entire CSS output

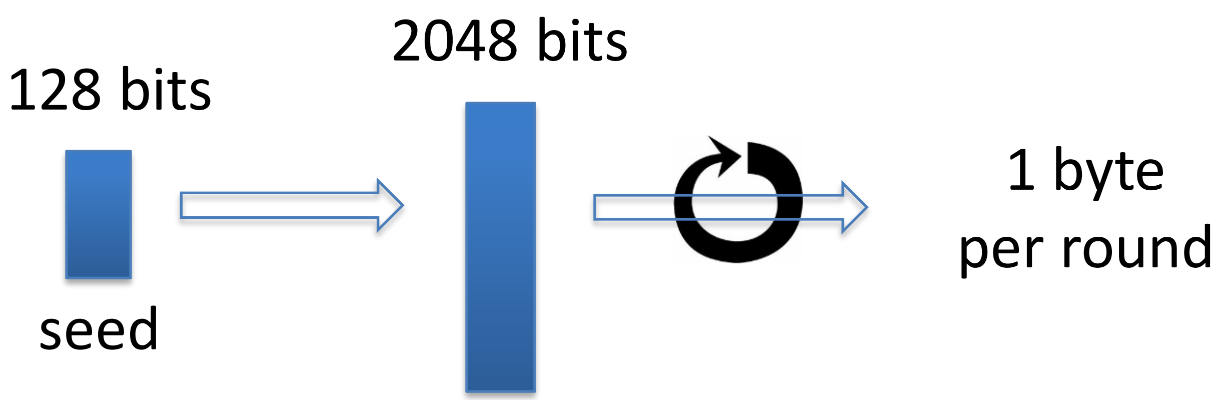
2^{17} time attack

LSFR review

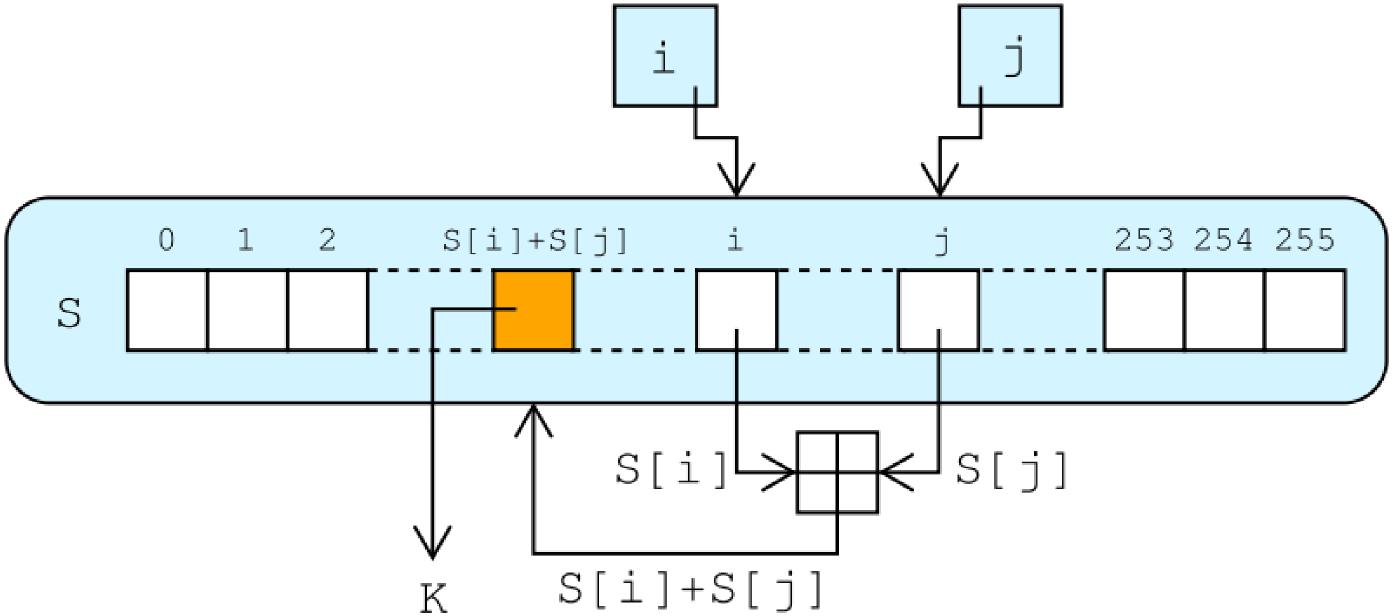
- Usually implemented in hardware
- Very fast, efficient, can generate as many bits as necessary
- Good statistical properties
  - Output of generated bits uniformly distributed
- Not cryptographically secure
  - Reconstruction attack
  - Streams ciphers based on LSFR broken
- Can be used as a primitive in other cryptographic constructions

RC4

- Designed by Ron Rivest in 1987
- Simple, fast, widely used
  - SSL/TLS for Web security, WEP for wireless



RC4 Encryption




```
i = j = 0;
While (more_byte_to_encrypt)
    i = (i + 1) (mod 256);
    j = (j + S[i]) (mod 256);
    swap(S[i], S[j]);
    k = (S[i] + S[j]) (mod
    256);
    C_i = M_i XOR S[k];  PRG
```

RC4 Initialization

```
Divide key K into L bytes
for i = 0 to 255 do
    S[i] := i
j := 0
for i = 0 to 255 do
        j := (j+S[i]+K[i mod L]) mod 256
        swap(S[i],S[j])
```

Key 128 bits
L = 16
Can be longer

Generate initial permutation from key K

- To use RC4, usually prepend initialization vector (IV) to the key
- Weaknesses
  - Bias in initial output: Pr[2^{nd} byte = 0] = 2/256
  - Prob. of (0,0) is $1/256^2 + 1/256^3$
  - Related key attacks
- To use RC4, discard first 256 bytes, but today RCA is considered insecure

Modern stream ciphers: eStream

PRG: $\{0,1\}^s \times R \longrightarrow \{0,1\}^n$

seed nonce

Nonce: a non-repeating value for a given key.

$$E(k, m ; r) = m \oplus PRG(k ; r)$$

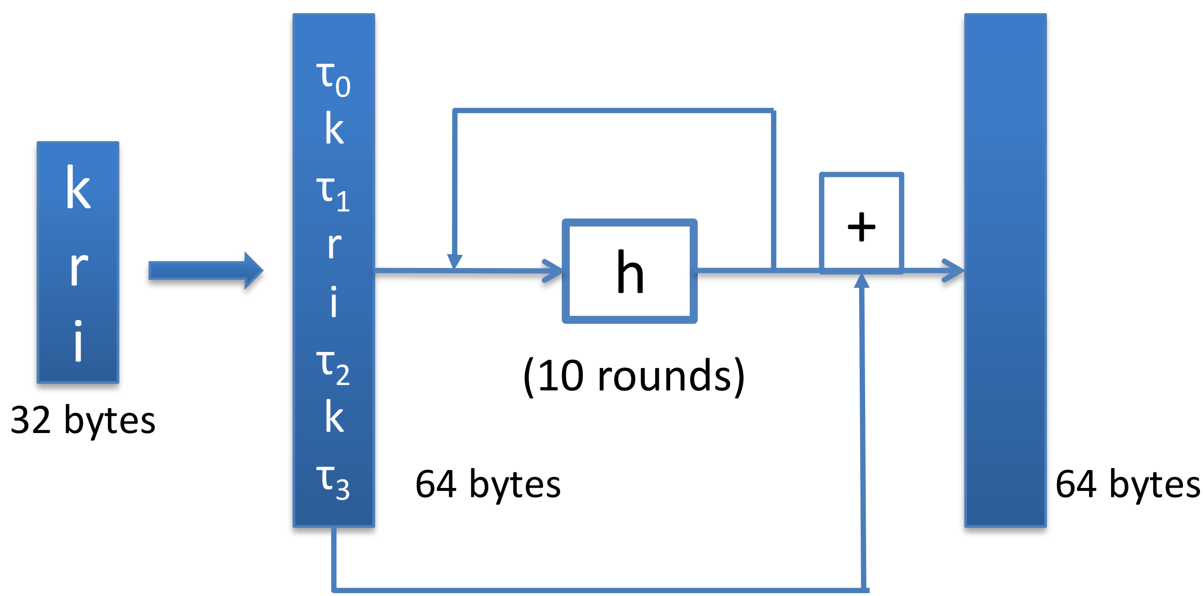
The pair (k,r) is never used more than once.

eStream: Salsa 20 (SW+HW)

nonce

Salsa20: $\{0,1\}^{128 \text{ or } 256} \times \{0,1\}^{64} \longrightarrow \{0,1\}^{n}$ (max n = 2^{73} bits)

Salsa20(k ; r) := H(k , (r, 0)) ll H(k , (r, 1)) ll …



h: invertible function designed to be fast on x86

Is Salsa20 secure ?

- Unknown: no known **provably** secure PRGs

- In reality: no known attacks better than exhaustive search

Performance: Crypto++ 5.6.0 [Wei Dai]

AMD Opteron, 2.2 GHz (Linux)

	PRG	Speed (MB/sec)
	RC4	126
eStream	Salsa20/12	643
eStream	Sosemanuk	727

Outline

- Stream cipher definition
- Constructions
  - LFSR
  - RC4
  - Salsa20
- Attacks on implementations and protocols
  - Two-time pad
  - Malleability

Attack 1: **two time** pad is insecure !!

Never use stream cipher key more than once !!

$C_1 \leftarrow m_1 \oplus PRG(k)$

$C_2 \leftarrow m_2 \oplus PRG(k)$

Eavesdropper does:

$C_1 \oplus C_2 \rightarrow m_1 \oplus m_2$

Enough redundancy in English text that:

$m_1 \oplus m_2 \rightarrow m_1, m_2$

[A Natural Language Approach to Automated Cryptanalysis of Two-time Pads]

Real world examples

- MS-PPTP (windows NT):
  - Microsoft Point-to-Point Tunneling Protocol

k

$\longrightarrow m_1$

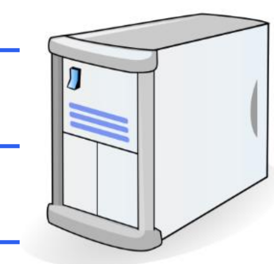
$\longrightarrow m_2$

$\longrightarrow m_3$

$u_1 \longleftarrow$

$u_2 \longleftarrow$

$u_3 \longleftarrow$



k

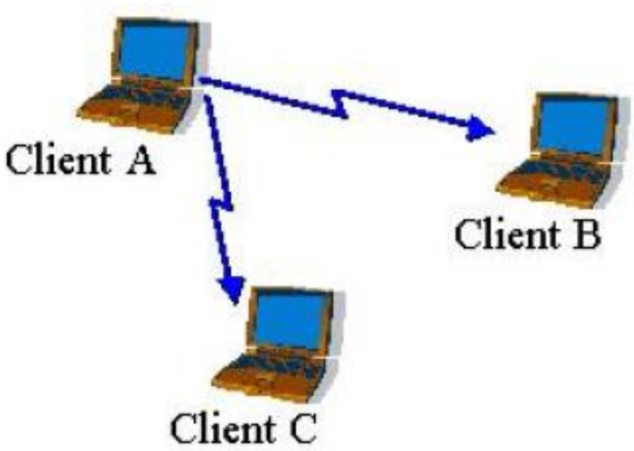
$[m_1 || m_2 || m_3] \oplus \mathrm{PRG}(k)$

$[u_1 || u_2 || u_3] \oplus \mathrm{PRG}(k)$

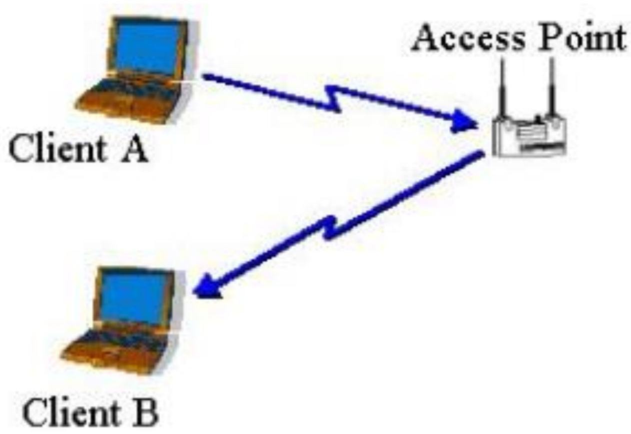
Need different keys for $C \longrightarrow S$ and $S \longrightarrow C$

802.11b Overview

- Standard for wireless networks (IEEE 1999)
- Two modes: infrastructure and ad hoc



IBSS (ad hoc) mode



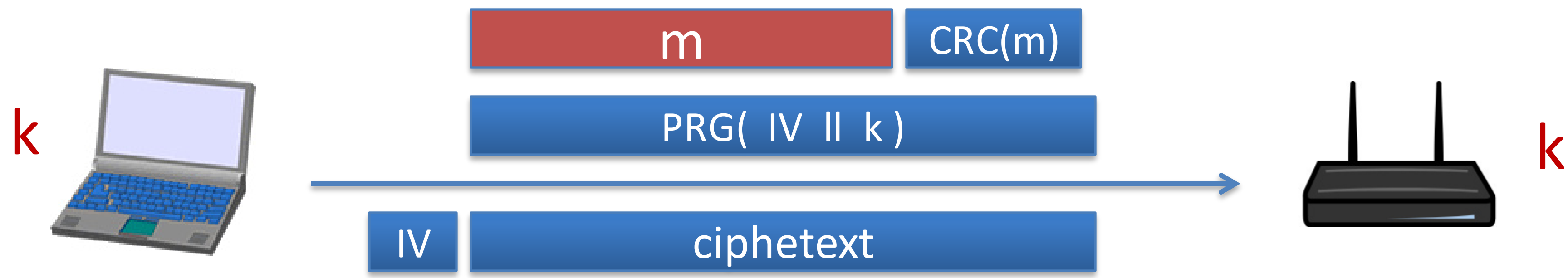
BSS (infrastructure) mode

WEP: Wired Equivalent Privacy

- Special-purpose protocol for 802.11b
  - Intended to make wireless as secure as wired network
- Goals: confidentiality, integrity, authentication
- Assumes that a secret key is shared between access point and client
- Uses RC4 stream cipher seeded with 24-bit initialization vector and 40-bit key

Real world examples

802.11b WEP:

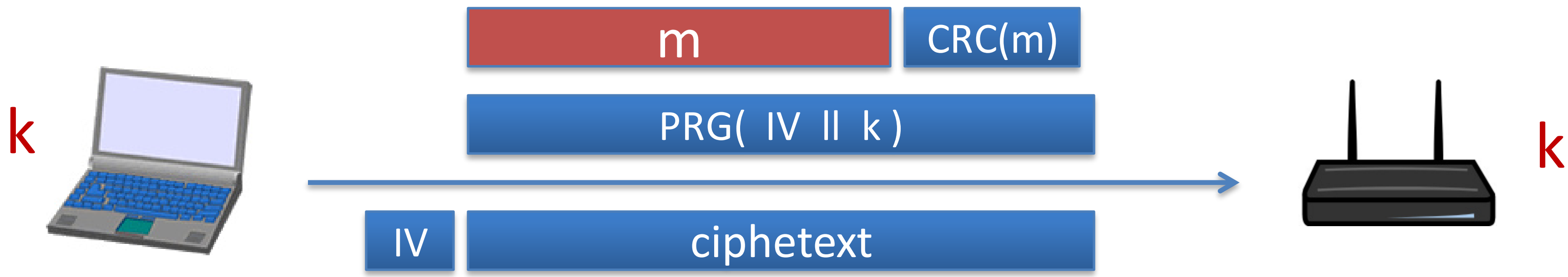


Length of IV: 24 bits

- Repeated IV after 2^{24} ≈ 16M frames
- On some 802.11 cards: IV resets to 0 after power cycle

Avoid related keys

802.11b WEP:



key for frame #1: (1 ll k)

key for frame #2: (2 ll k)

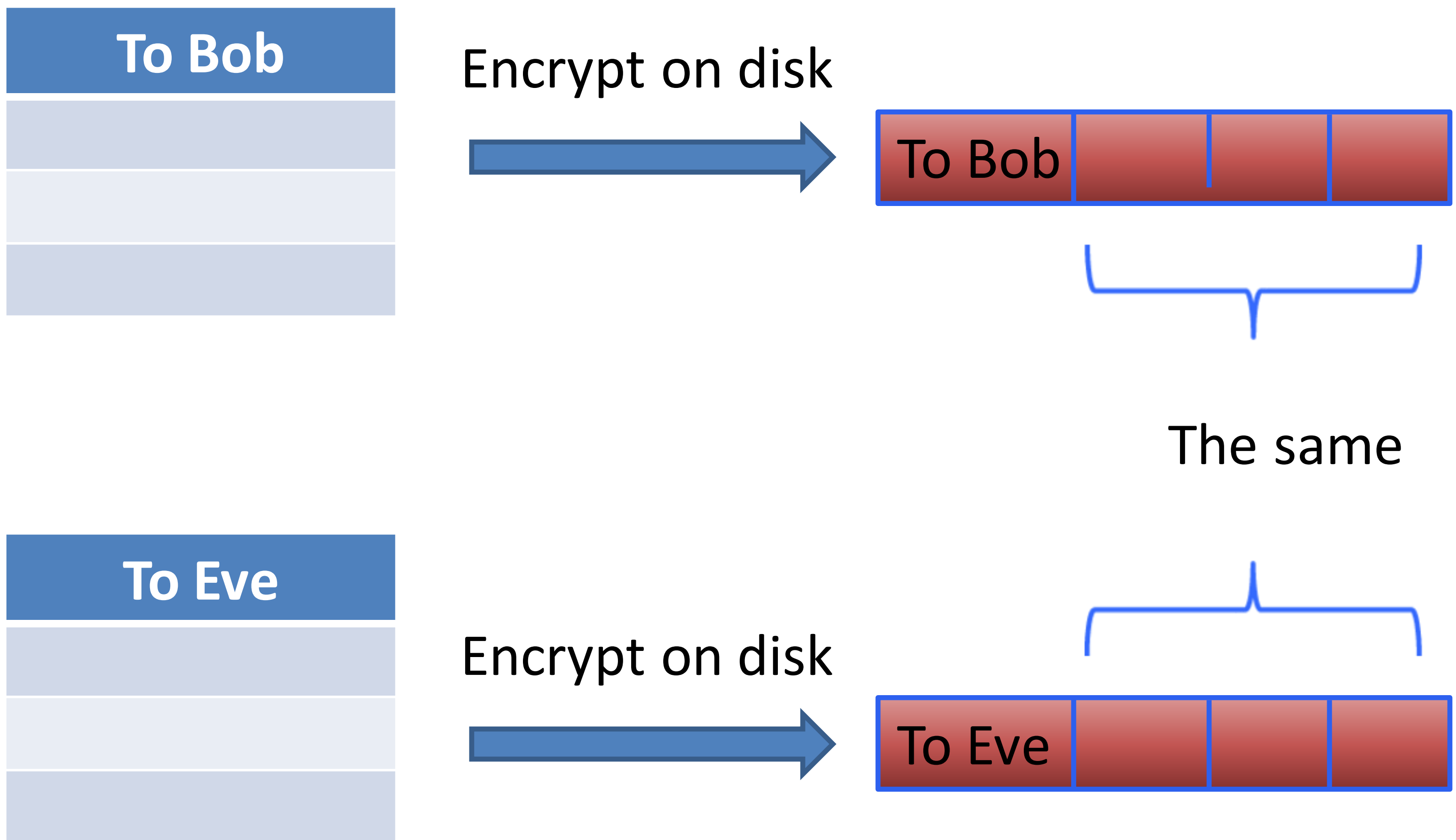
24 bits 104 bits

For RC4 cipher
- Fluhrer-Mantin-Shamir can recover k after 10^6 frames
- Recent attack: 10,000 frames

Better design

- How to fix related key attacks?
  - $k_i = i \mid\mid k$
- Microsoft PPTP
  - Use PRG with single key and long output
  - $[m_1 \mid\mid m_2 \mid\mid m_3] \oplus PRG(k)$
- Generate pseudorandom keys
  - Use second PRG: $PRG'(k) = k_1 \ldots k_n$
  - Encrypt each frame m_i with different key k_i
    - $c_i = PRG(k_i) \oplus m_i$
  - The pseudorandom keys are not related!

Yet another example: disk encryption



Adversary learns access patterns (which blocks changed)

Two-time pad attack on modified block

Two time pad attack: summary

Never use stream cipher key more than once !!

- *Network traffic*: negotiate new key for every session (e.g. TLS)
  - Different key for client and server

- Disk encryption: typically do not use a stream cipher

- Network protocols have been broken!
  - WEP
  - 802.11

Acknowledgement

Some of the slides and slide contents are taken from
http://www.crypto.edu.pl/Dziembowski/teaching
and fall under the following:

We have also used slides from Prof. Dan Boneh online cryptography course at Stanford University:

http://crypto.stanford.edu/~dabo/courses/OnlineCrypto/