Stream ciphers

Stream ciphers

Semantic security

Goal: secure PRG $\Rightarrow$ “secure” stream cipher

What is a secure cipher?

Attacker's abilities: **obtains one ciphertext** (for now)

Possible security requirements:

attempt #1: **attacker cannot recover secret key**

$$E(k,m) = m$$

attempt #2: **attacker cannot recover all of plaintext**

$$E(k, m_0 \| m_1) = m_0 \| m_1 \oplus k$$

Recall Shannon's idea:

CT should reveal no "info" about PT

Recall Shannon's perfect secrecy

Let (E,D) be a cipher over (K,M,C)

(E,D) has perfect secrecy if $\forall\ m_0, m_1 \in M \quad (|m_0| = |m_1|)$

$$\{ E(k,m_0) \} \quad = \quad \{ E(k,m_1) \} \qquad \text{where} \quad k \leftarrow K$$

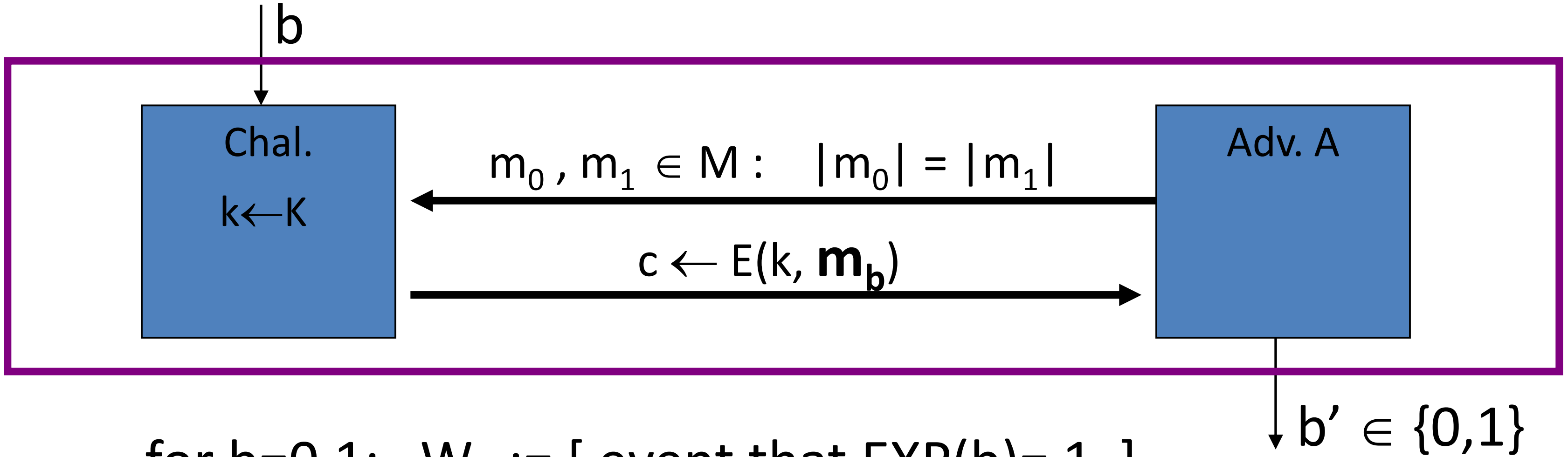
(E,D) has perfect secrecy if $\forall\ m_0, m_1 \in M \quad (|m_0| = |m_1|)$

$$\{ E(k,m_0) \} \ \approx_p \ \{ E(k,m_1) \} \qquad \text{where} \quad k \leftarrow K$$

... but also need adversary to exhibit $m_0, m_1 \in M$ explicitly

Semantic Security (one-time key)

For $b=0,1$ define experiments EXP(0) and EXP(1) as:



for $b=0,1$: W_b := [event that EXP(b)= 1]

$$\text{Adv}_{SS}[A,E] := \Big|\ \Pr[\,W_0\,] - \Pr[\,W_1\,]\ \Big| \quad \in [0,1]$$

Semantic Security (one-time key)

Def: $\mathbb{E}$ is **semantically secure** if for all efficient A

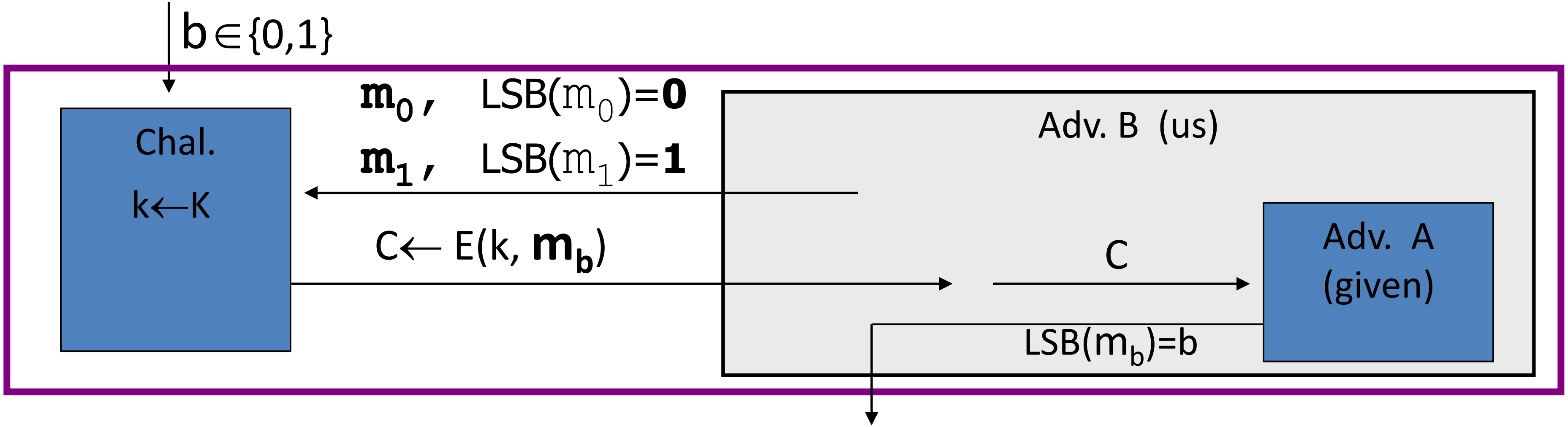
$\mathrm{Adv}_{SS}[A,\mathbb{E}]$ is negligible.

$\Rightarrow$ for all explicit m_0 , $m_1 \in M$: $\{ E(k,m_0) \} \approx_p \{ E(k,m_1) \}$

Examples

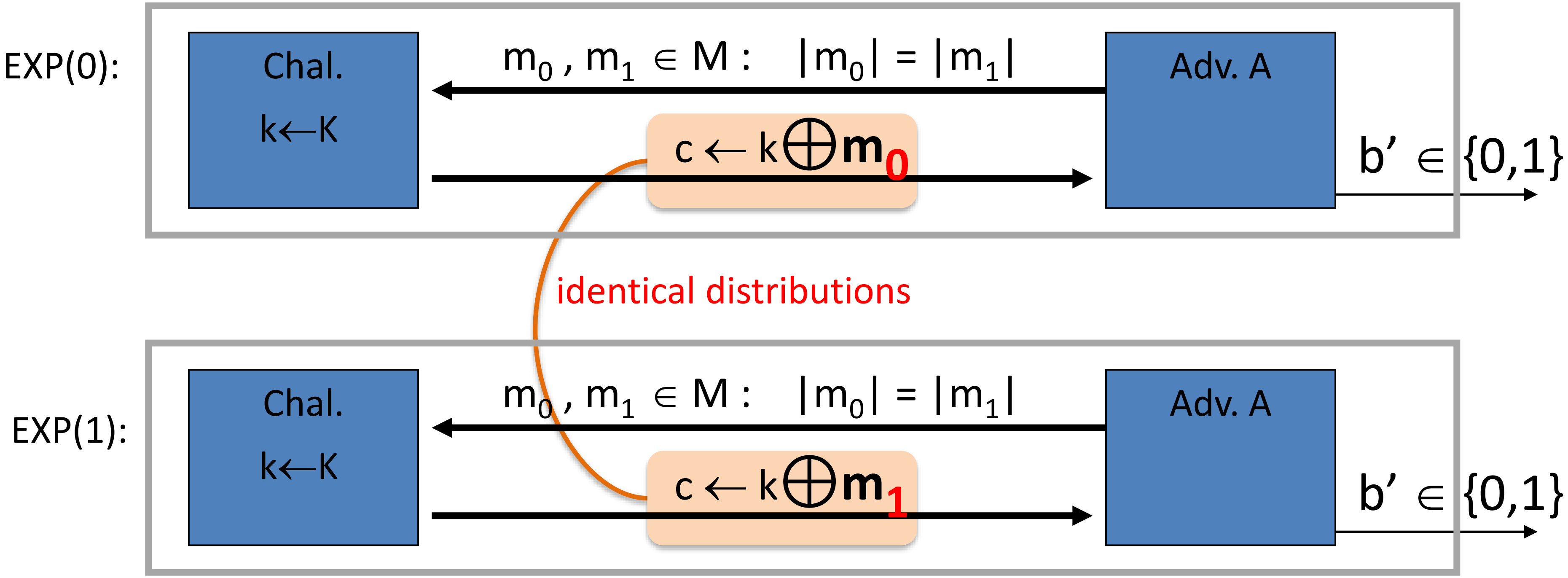
Suppose efficient A can always deduce LSB of PT from CT.

$\Rightarrow$ $\mathbb{E}$ = (E,D) is not semantically secure.



Then $\text{Adv}_{SS}[B, \mathbb{E}]$ = $\Big|$ Pr[**EXP(0)**=1] − Pr[**EXP(1)**=1] $\Big|$ = |0 – 1| = 1

OTP is semantically secure



For **all** A: $\quad \text{Adv}_{SS}[A,\text{OTP}] = \Big| \Pr[\, \mathbf{A(k \oplus m_0)}=1 \,] - \Pr[\, \mathbf{A(k \oplus m_1)}=1 \,] \Big| = 0$

End of Segment

Stream ciphers

Stream ciphers are semantically secure

Goal: secure PRG ⇒ semantically secure stream cipher

Stream ciphers are semantically secure

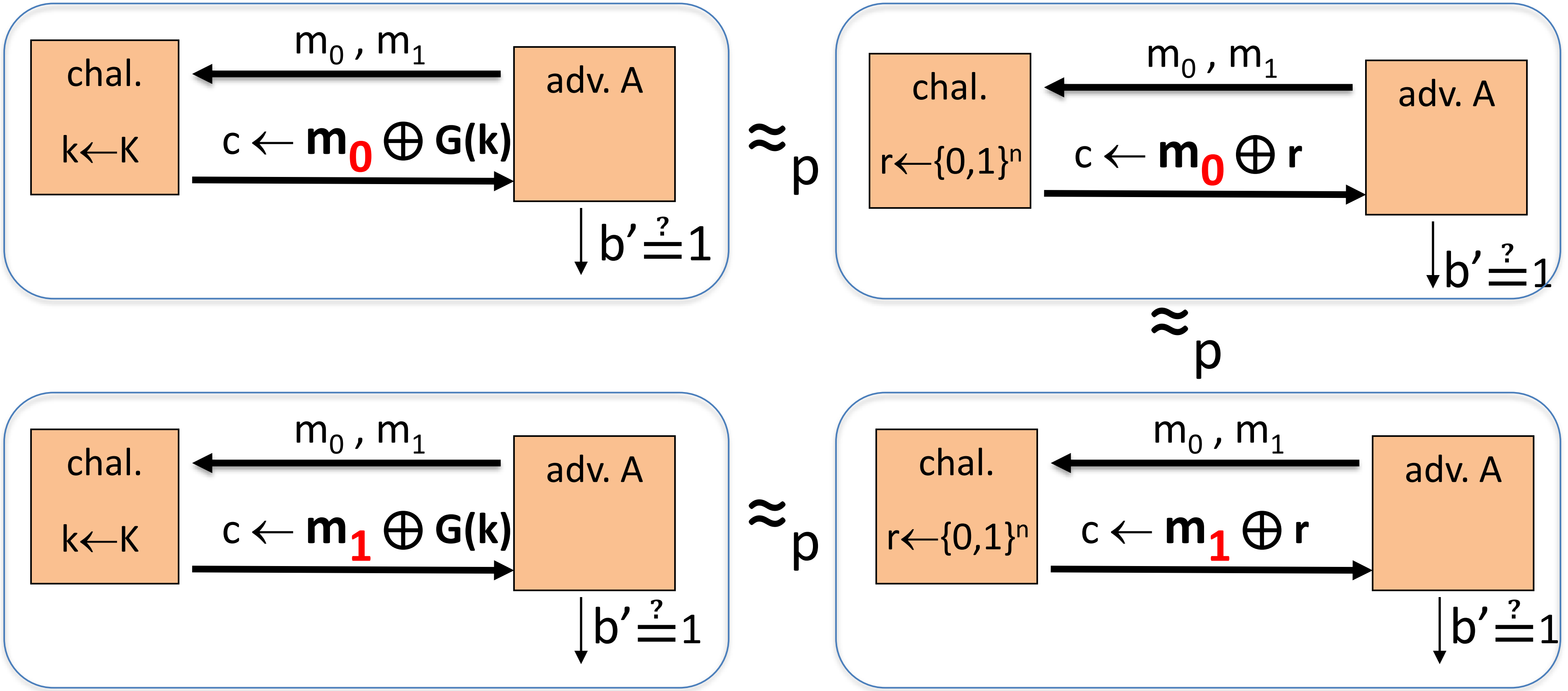
Thm: $G: K \longrightarrow \{0,1\}^n$ is a secure PRG $\Rightarrow$

stream cipher E derived from G is sem. sec.

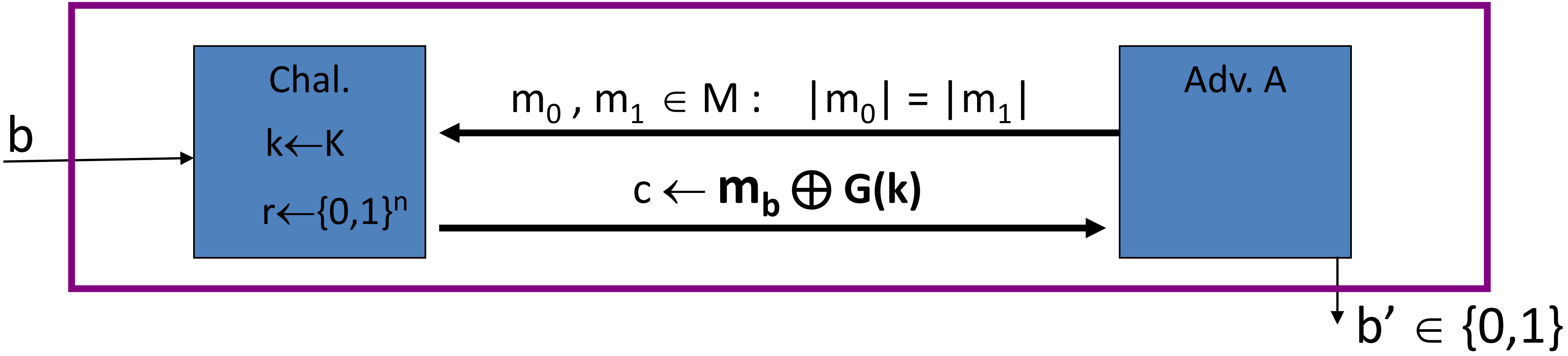
$\forall$ sem. sec. adversary A , $\exists$a PRG adversary B s.t.

$$\mathrm{Adv}_{SS}[A,E] \le 2 \cdot \mathrm{Adv}_{PRG}[B,G]$$

Proof: intuition



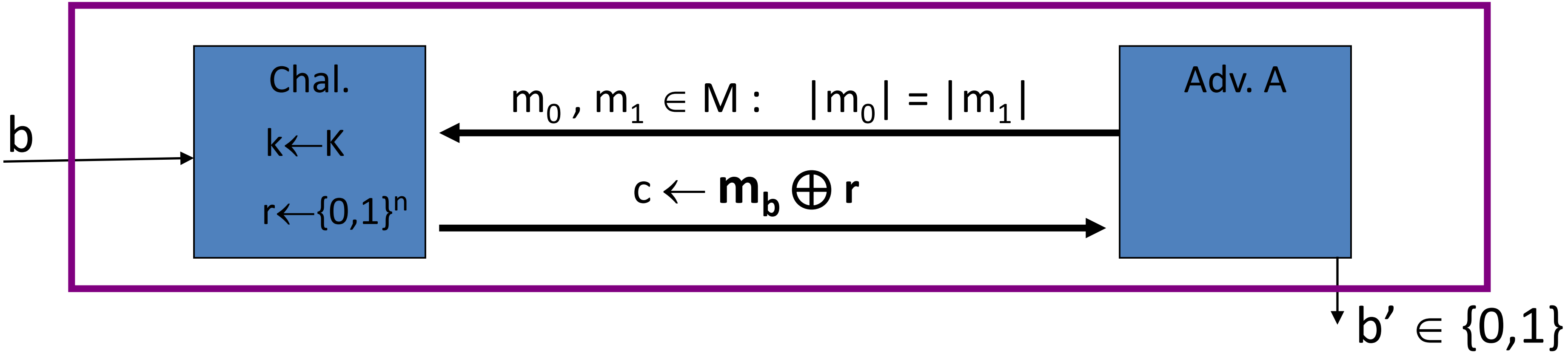
Proof: Let A be a sem. sec. adversary.



For b=0,1: W_b := [event that b'=1].

$$\mathrm{Adv}_{SS}[A,E] = \big|\ \Pr[\, W_0\,] - \Pr[\, W_1\,]\ \big|$$

Proof: Let A be a sem. sec. adversary.



For b=0,1: W_b := [event that b'=1].

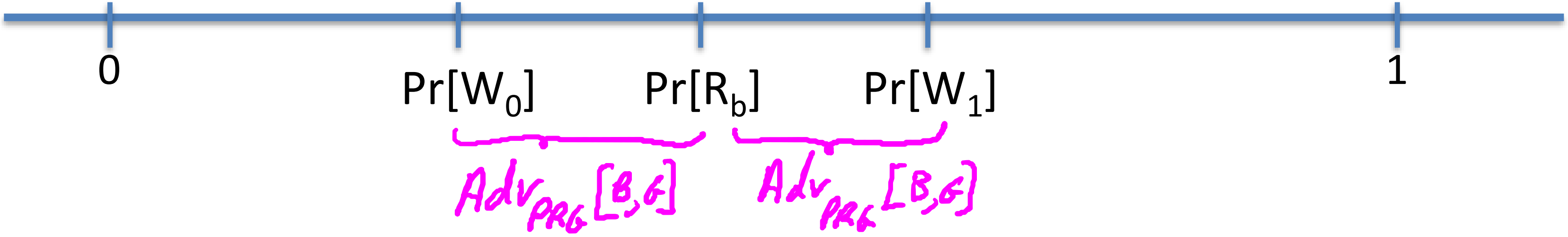
$$\mathrm{Adv}_{SS}[A,E] = \left|\ \Pr[W_0] - \Pr[W_1]\ \right|$$

For b=0,1: R_b := [event that b'=1]

Proof: Let A be a sem. sec. adversary.

Claim 1: $\left|\Pr[R_0] - \Pr[R_1]\right| = Adv_{SS}[A, OTP] = 0$

Claim 2: $\exists B$: $\left|\Pr[W_b] - \Pr[R_b]\right| = Adv_{PRG}[B, G]$ For $b=0,1$

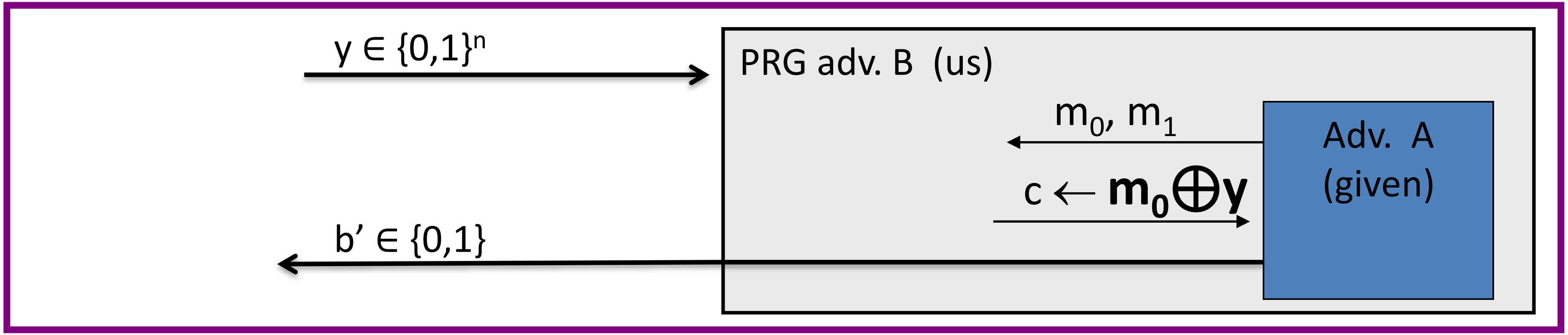


$\Rightarrow$ $Adv_{SS}[A,E] = \left|\Pr[W_0] - \Pr[W_1]\right| \le 2 \cdot Adv_{PRG}[B,G]$

Dan Boneh

Proof of claim 2: $\exists B: \left|\Pr[W_0] - \Pr[R_0]\right| = \mathrm{Adv}_{PRG}[B,G]$

Algorithm B:



$$\mathrm{Adv}_{PRG}[B,G] = \left| \Pr_{r \xleftarrow{R} \{0,1\}^n}[B(r)=1] - \Pr_{k \xleftarrow{R} \mathcal{K}}[B(G(k))=1] \right| = \left| \Pr[R_0] - \Pr[W_0] \right|$$

End of Segment

Stream ciphers

Real-world Stream Ciphers

Modern stream ciphers: eStream

$$\text{PRG:} \quad \underbrace{\{0,1\}^s}_{\text{seed}} \times R \longrightarrow \{0,1\}^n$$

nonce

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

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

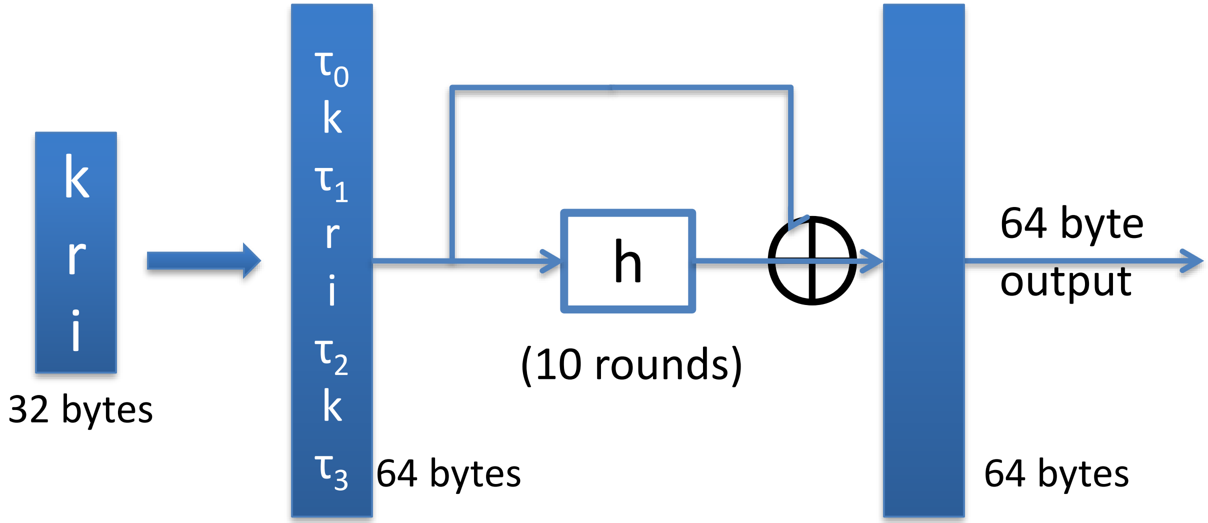
The pair (k,r) is never used more than once.

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 (SSE2)

Is Salsa20 secure (unpredictable) ?

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

- In reality: no known attacks better than exhaustive search

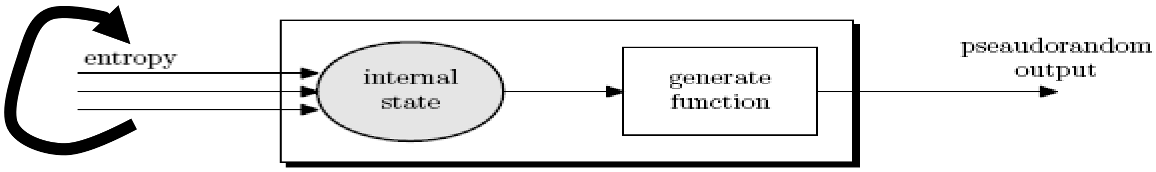
Performance:

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

Generating Randomness (e.g. keys, IV)



Pseudo random generators in practice: (e.g. /dev/random)

- Continuously add entropy to internal state
- Entropy sources:
  - Hardware RNG: Intel **RdRand** inst. (Ivy Bridge). 3Gb/sec.
  - Timing: hardware interrupts (keyboard, mouse)

NIST SP 800-90: NIST approved generators

End of Segment

Additional Slides

Weak PRGs (do not use for crypto)

Lin. Cong. generator with parameters a, b, p:

$r[i] \leftarrow a \cdot r[i-1] + b \mod p$

output bits of $r[i]$

$i{+}{+}$

seed $\equiv r[0]$

glibc random():

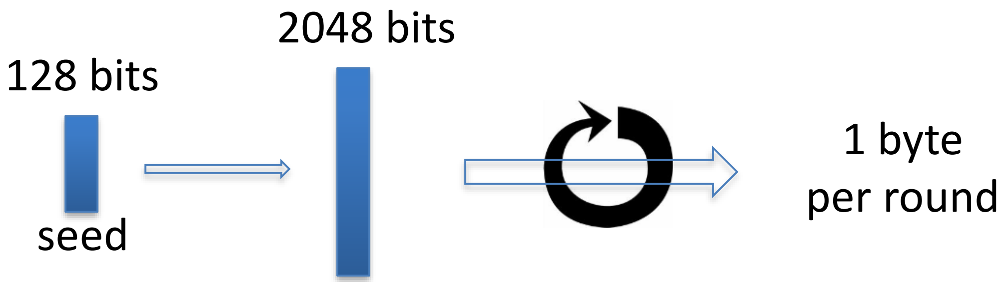
$r[i] \leftarrow (\ r[i-3] + r[i-31]\ )\ \%\ 2^{32}$

output $r[i] >> 1$

never use random() for crypto !!

(e.g. Kerberos V4)

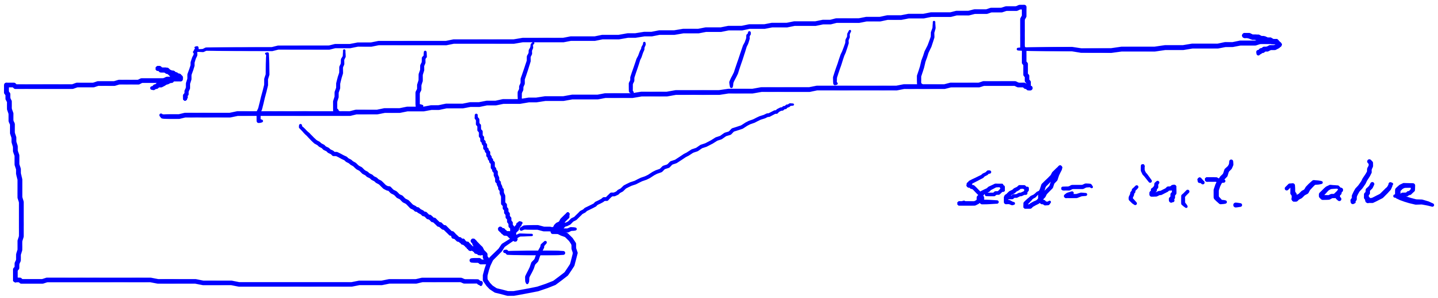
Old example (software): RC4 (1987)



- Used in HTTPS and WEP
- Weaknesses:
  1. Bias in initial output: $\Pr[\, 2^{nd} \text{ byte} = 0 \,] = 2/256$
  2. Prob. of (0,0) is $1/256^2 + 1/256^3$
  3. Related key attacks

Old example (hardware): CSS (badly broken)

Linear feedback shift register (LFSR):



DVD encryption (CSS): 2 LFSRs

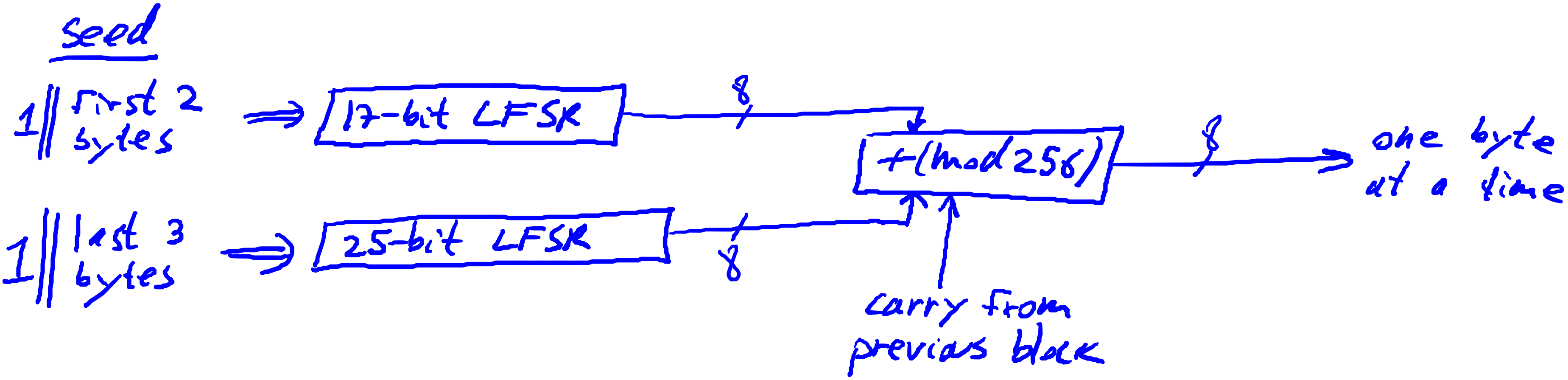
GSM encryption (A5/1,2): 3 LFSRs

Bluetooth (E0): 4 LFSRs

all broken

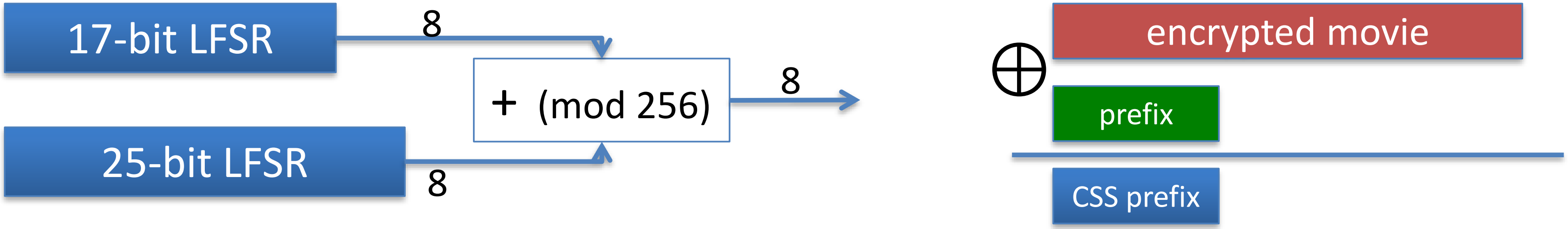
Old example (hardware): CSS (badly broken)

CSS: seed = 5 bytes = 40 bits



Easy to break in time $\approx 2^{17}$

Cryptanalysis of CSS (2^{17} time attack)



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, found correct initial settings of both !!

Using key, generate entire CSS output