CSCI471/971
Modern Cryptography

PRG, Steam Cipher and Sematic Security

Jiageng Chen
Central China Normal University Wollongong Joint Institute

This slide is made based the online course of Cryptography by Dan Boneh

Symmetric Ciphers: definition

<u>Def</u>: a **cipher** defined over (K, M, C)

is a pair of "efficient" algs (***E***, ***D***) where

$$E : \mathcal{K} \times \mathcal{M} \to \mathcal{C} \qquad D : \mathcal{K} \times \mathcal{C} \to \mathcal{M}$$

for all keys k and all messages m, we have

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

- ***E*** is often randomized. ***D*** is always deterministic.

Pseudorandom Generators

Review

Cipher over (K,M,C): a pair of "efficient" algs (***E***, ***D***) s.t.

$$\forall\, m \in M,\ k \in K: \quad \boldsymbol{D}(k, \boldsymbol{E}(k, m)) = m$$

Weak ciphers: subs. cipher, Vigener, …

A good cipher: **OTP** $M=C=K=\{0,1\}^n$

$$E(k, m) = k \oplus m \quad , \quad D(k, c) = k \oplus c$$

<u>Lemma</u>: OTP has perfect secrecy (i.e. no CT only attacks)

Bad news: perfect-secrecy $\Rightarrow$ key-len $\geq$ msg-len

Stream Ciphers: making OTP practical

idea: replace "random" key by "pseudorandom" key

PRG is a function $G: \{0,1\}^s \to \{0,1\}^n$, $n \gg s$

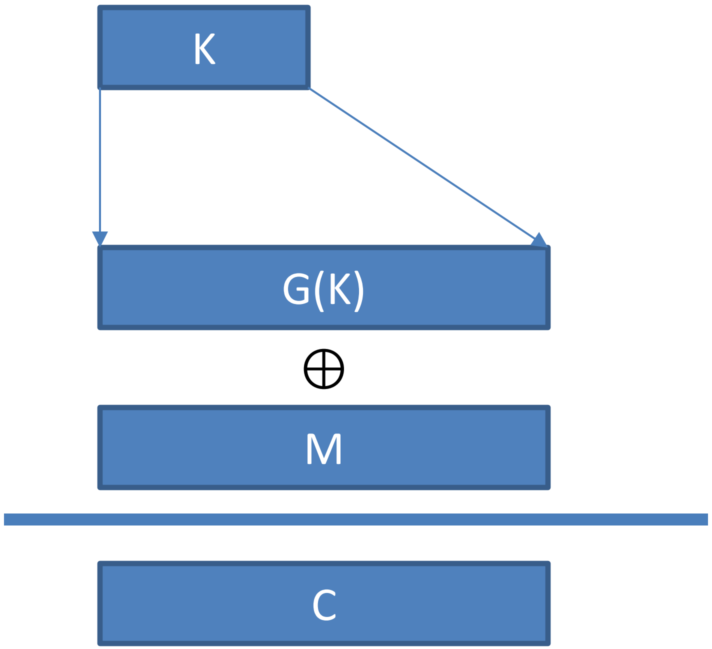
Seed space

Efficiently computable by a deterministic algorithm

Stream Ciphers: making OTP practical

$$C = E(K, m) = m \oplus G(k)$$

$$D(k, C) = C \oplus G(k)$$



Can a stream cipher have perfect secrecy?

- ○ Yes, if the PRG is really “secure”
- ○ No, there are no ciphers with perfect secrecy
- ○ Yes, every cipher has perfect secrecy
- ○ No, since the key is shorter than the message

Stream Ciphers: making OTP practical

Stream ciphers cannot have perfect secrecy !!

- Need a different definition of security

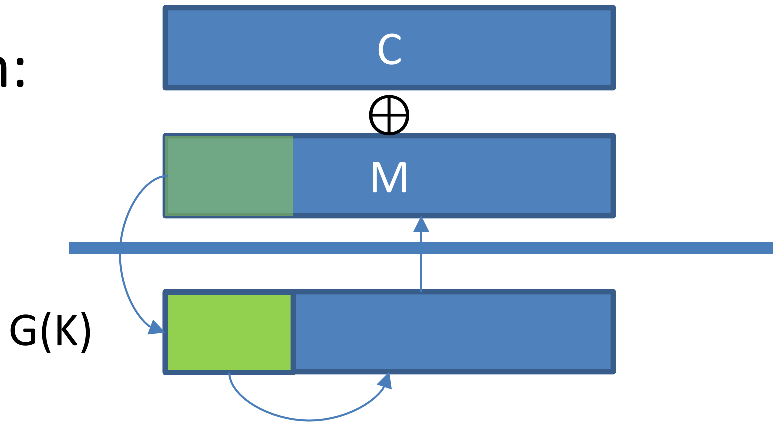
- Security will depend on specific PRG

PRG must be unpredictable

Suppose PRG is predictable.

$$\exists\, i: G(k)|_{1,\ldots,i} \longrightarrow G(k)|_{i+1,\ldots,n}$$

Then:



Even $G(k)|_{1,\ldots,i} \longrightarrow G(k)|_{i+1}$ is a problem!

PRG must be unpredictable

We say that G: K $\longrightarrow \{0,1\}^n$ is **predictable** if:

$\exists$ "eff" alg. A and $\exists\ 0 \leq i \leq n-1$

s.t. Pr[A($G(k)|_{1,\ldots,i}$)=$G(k)|_{i+1}$]> $\frac{1}{2} + \varepsilon$

for non-negligible ε (e.g. ε=1/2^{30})

Def: PRG is **unpredictable** if it is not predictable

$\Rightarrow$ $\forall$i: no "eff" adv. can predict bit (i+1) for "non-neg" ε

Suppose $G:K \longrightarrow \{0,1\}^n$ is such that for all k: **XOR(G(k)) = 1**

Is G predictable ?

- Yes, given the first bit I can predict the second
- No, G is unpredictable
- Yes, given the first (n-1) bits I can predict the n'th bit
- It depends

Weak PRGs (do not use for crypto)

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

r[i] $\longleftarrow$ $a \cdot r[i-1] + b \; mod \; p$ seed=r[0]

output bits of r[i]

i++

glibc random():

r[i] ← (r[i-3] + r[i-31]) % 2^{32}

output r[i] >> 1

never use random() for crypto!

(e.g. Kerberos V4)

Stream ciphers

Negligible vs. non-negligible

Negligible and non-negligible

- In practice: ε is a scalar and
  - ε non-neg: $\varepsilon \geq 1/2^{30}$ (likely to happen over 1GB of data)
  - ε negligible: $\varepsilon \leq 1/2^{80}$ (won't happen over life of key)

- In theory: ε is a function $\varepsilon: Z^{\geq 0} \longrightarrow R^{\geq 0}$ and
  - ε non-neg: $\exists d$: $\varepsilon(\lambda) \geq 1/\lambda^d$ inf. often ($\varepsilon \geq$ 1/poly, for many λ)
  - ε negligible: $\forall d, \lambda \geq \lambda_d$: $\varepsilon(\lambda) \leq 1/\lambda^d$ ($\varepsilon \leq$ 1/poly, for large λ)

Few Examples

$\varepsilon(\lambda) = 1/2^{\lambda}$: negligible

$\varepsilon(\lambda) = 1/\lambda^{1000}$: non-negligible

PRGs: the rigorous theory view

PRGs are "parameterized" by a security parameter λ

- **PRG** becomes "more secure" as λ increases

Seed lengths and output lengths grow with λ

For every λ=1,2,3,… there is a different PRG G_λ:

$$G_\lambda : K_\lambda \longrightarrow \{0,1\}^{n(\lambda)}$$

(in the lectures we will always ignore λ)

An example asymptotic definition

We say that $G_\lambda : K_\lambda \longrightarrow \{0,1\}^{n(\lambda)}$ is **predictable** at position i if:

there exists a polynomial time (in λ) algorithm A s.t.

$$\Pr_{k \leftarrow K_\lambda}\Big[\, A\big(\lambda,\ G_\lambda(k)\big|_{1,\dots,i}\big) = G_\lambda(k)\big|_{i+1} \,\Big] > 1/2 + \varepsilon(\lambda)$$

for some non-negligible function $\varepsilon(\lambda)$

End of Segment

Stream ciphers

Attacks on OTP and stream ciphers

Review

OTP: $E(k,m) = m \oplus k$, $D(k,c) = c \oplus k$

Making OTP practical using a PRG: $G: K \longrightarrow \{0,1\}^n$

Stream cipher: $E(k,m) = m \oplus G(k)$, $D(k,c) = c \oplus G(k)$

Security: PRG must be unpredictable (better def in two segments)

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

Never use stream cipher key more than once !!

$$C_1 \leftarrow m_1 \oplus \text{PRG}(k)$$
$$C_2 \leftarrow m_2 \oplus \text{PRG}(k)$$

Eavesdropper does:

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

Enough redundancy in English and ASCII encoding that:

$$m_1 \oplus m_2 \quad \rightarrow \quad m_1 , \; m_2$$

Real world examples

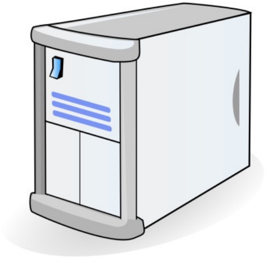
- Project Venona

- MS-PPTP (windows NT):
  - K=(Kc⟶s,Ks⟶c)

k 

m1 ⟶
m2 ⟶
m3 ⟶

⟵ s1
⟵ s2
⟵ s3

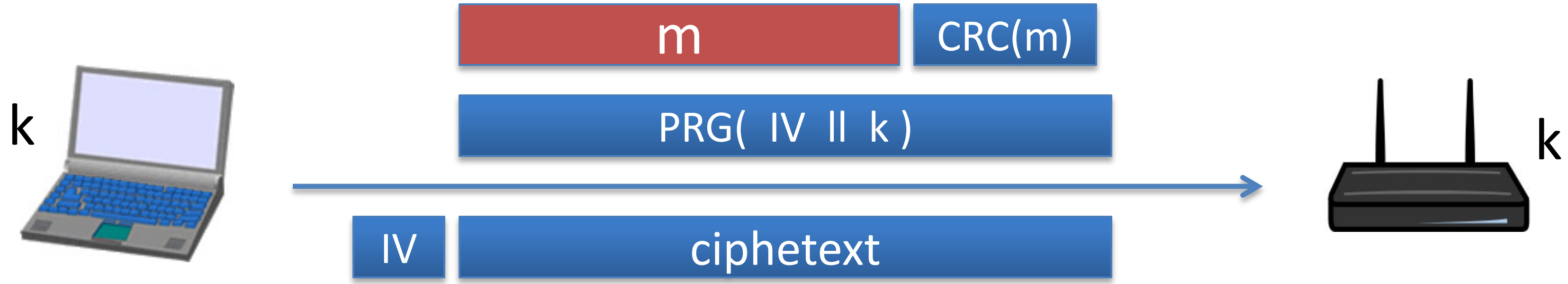
 k

- $[\mathrm{m1} \parallel m2 \parallel m3] \oplus G(k)$ $\quad$ $[s1 \parallel s2 \parallel s3] \oplus G(k)$

Need different keys for C⟶S and S⟶C

Real world examples

802.11b WEP:

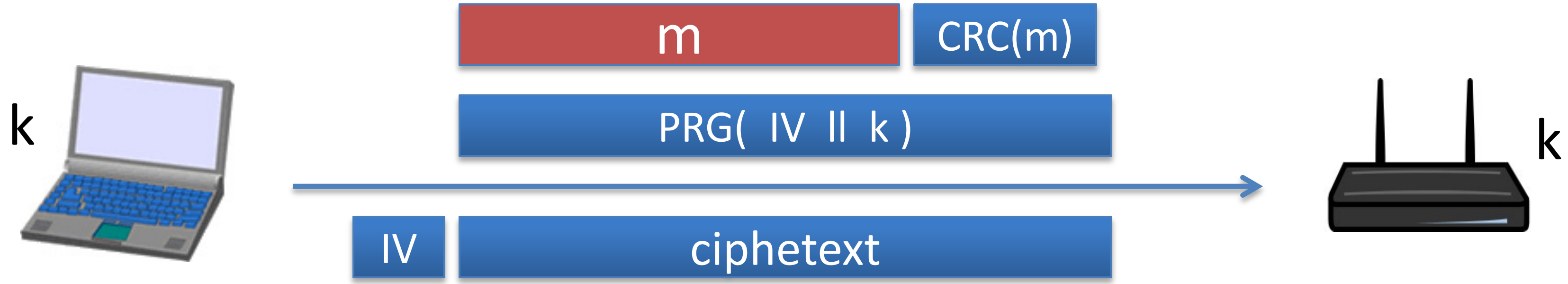


Length of IV: 24 bits

- Repeated IV after $2^{24} \approx$ 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)

⋮

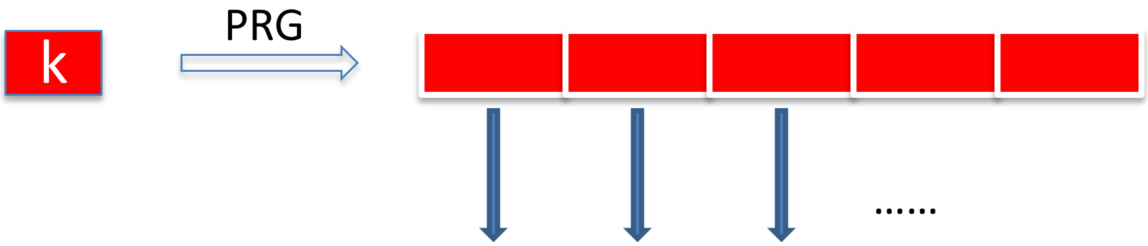
For the RC4 PRG:FMS2001 $\Longrightarrow$

can recover k after 10^6 frames

recent attacks $\approx$ 40000 frames

A better construction

k

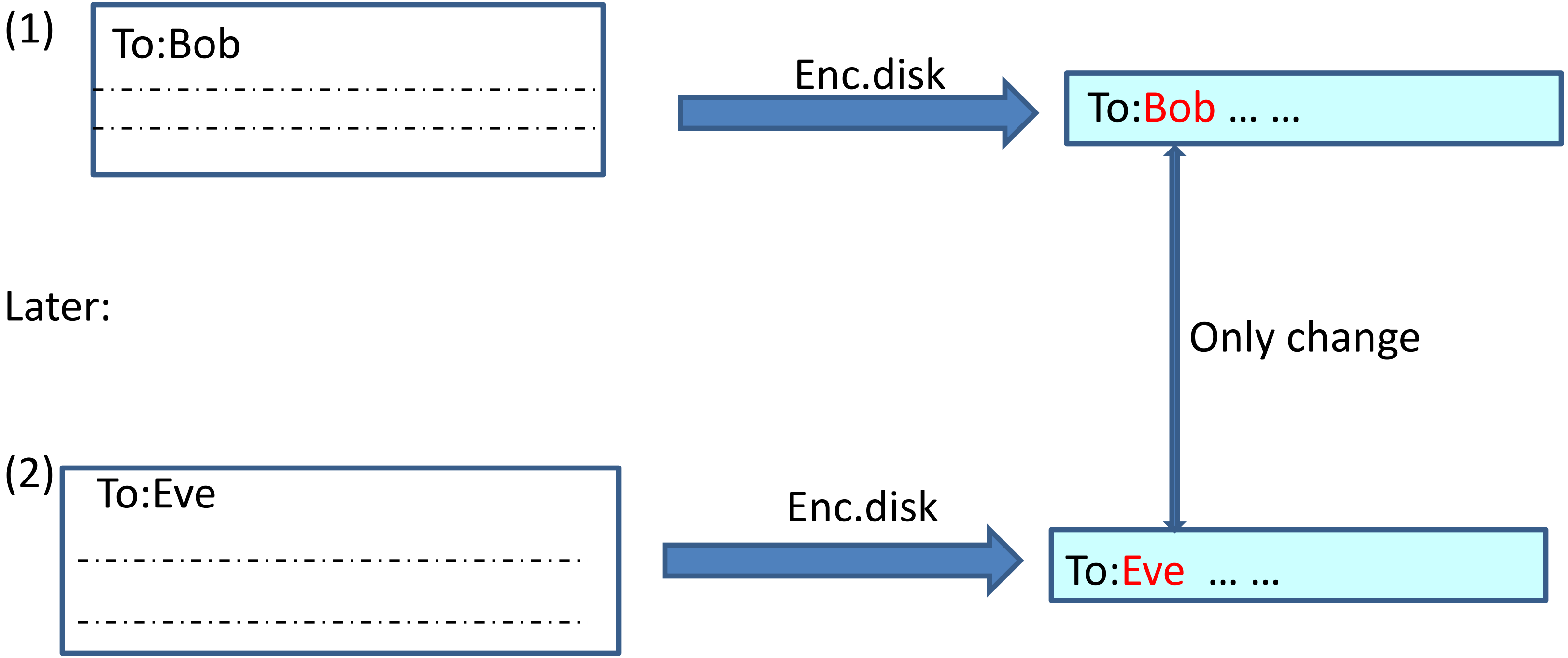


Key for frame #1   #2   #3

⇒ now each frame has a pseudorandom key

better solution: use stronger encryption method (as in WPA2)

Yet another example: disk encryption

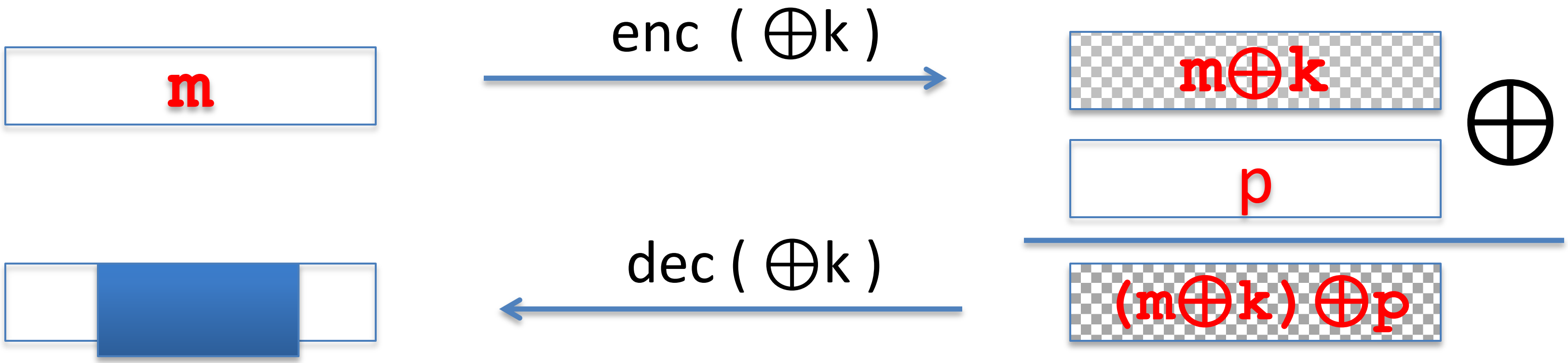


Two time pad: summary

Never use stream cipher key more than once !!

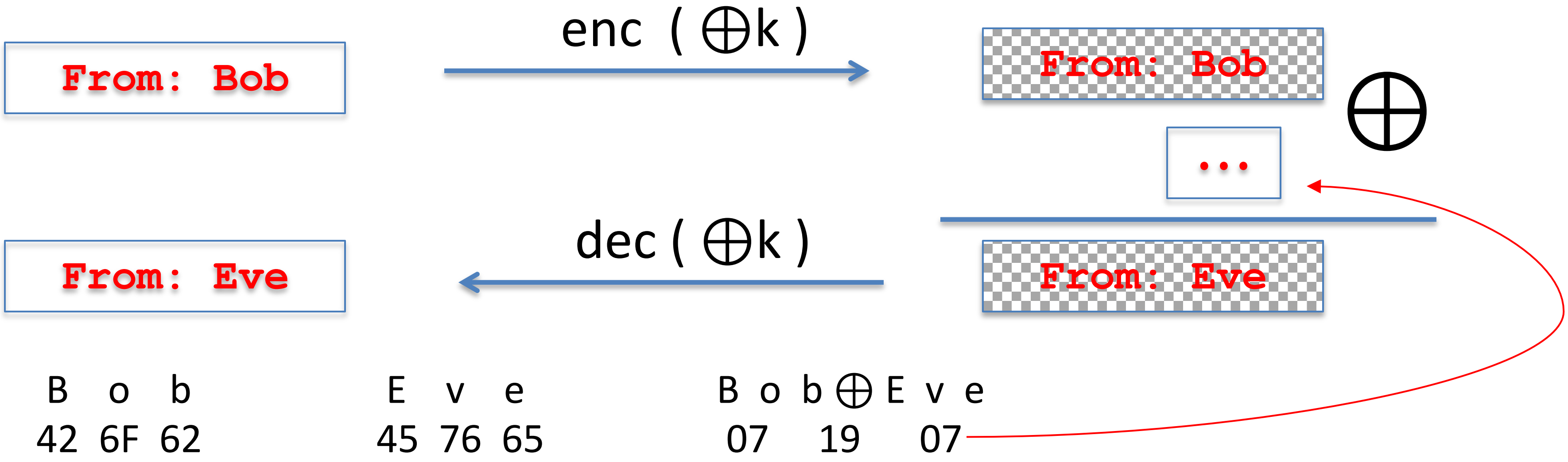
- Network traffic: negotiate new key for every session (e.g. TLS)
- Disk encryption: typically do not use a stream cipher

Attack 2: no integrity (OTP is malleable)



Modifications to ciphertext are undetected and have **predictable** impact on plaintext

Attack 2: no integrity (OTP is malleable)

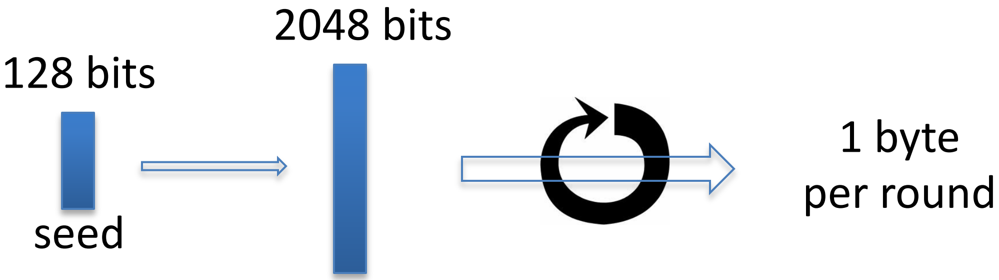


Modifications to ciphertext are undetected and have predictable impact on plaintext

End of Segment

Real-world Stream Ciphers

Old example (software): RC4 (1987)



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

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.

Chacha20 (SW+HW)

Chacha20: $\{0,1\}^{256} \times \{0,1\}^{64} \longrightarrow \{0,1\}^n$

nonce

Padding function pad(s , j, n):

- 256-bit seed s0, s1,…,s7 in $\{0,1\}^{32}$
- 64-bit counter j0, j1 in $\{0,1\}^{32}$
- 64-bit nonce n0,n1 in $\{0,1\}^{32}$
- Output a 512-bit block x0,…,x15 in $\{0,1\}^{32}$

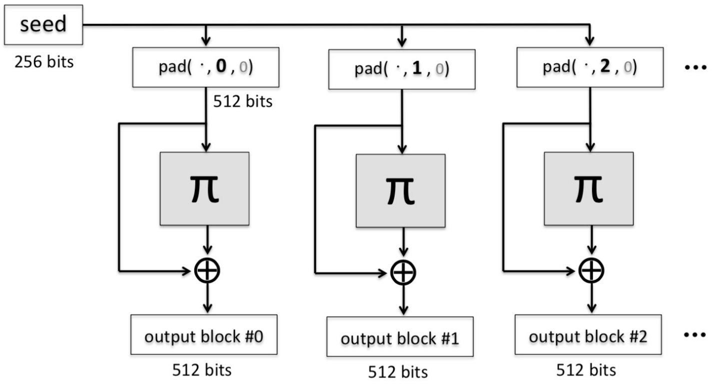
$$\begin{pmatrix} x_0 & x_1 & x_2 & x_3 \\ x_4 & x_5 & x_6 & x_7 \\ x_8 & x_9 & x_{10} & x_{11} \\ x_{12} & x_{13} & x_{14} & x_{15} \end{pmatrix} \longleftarrow \begin{pmatrix} c_0 & c_1 & c_2 & c_3 \\ s_0 & s_1 & s_2 & s_3 \\ s_4 & s_5 & s_6 & s_7 \\ j_0 & j_1 & n_0 & n_1 \end{pmatrix}$$

Permutation function π: $\{0,1\}^{512} \rightarrow \{0,1\}^{512}$

$$\begin{aligned}
&(1)\ \mathrm{QuarterRound}(x_0, x_4, x_8, x_{12}), && (2)\ \mathrm{QuarterRound}(x_1, x_5, x_9, x_{13}),\\
&(3)\ \mathrm{QuarterRound}(x_2, x_6, x_{10}, x_{14}), && (4)\ \mathrm{QuarterRound}(x_3, x_7, x_{11}, x_{15}),\\
&(5)\ \mathrm{QuarterRound}(x_0, x_5, x_{10}, x_{15}), && (6)\ \mathrm{QuarterRound}(x_1, x_6, x_{11}, x_{12}),\\
&(7)\ \mathrm{QuarterRound}(x_2, x_7, x_8, x_{13}), && (8)\ \mathrm{QuarterRound}(x_3, x_4, x_9, x_{14}).
\end{aligned}$$

QuarterRound(a, b, c, d):

```
a += b;  d ^= a;  d <<<= 16;
c += d;  b ^= c;  b <<<= 12;
a += b;  d ^= a;  d <<<= 8;
c += d;  b ^= c;  b <<<= 7;
```



Is Chacha20 secure (unpredictable) ?

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

- In reality: no known attacks better than exhaustive search

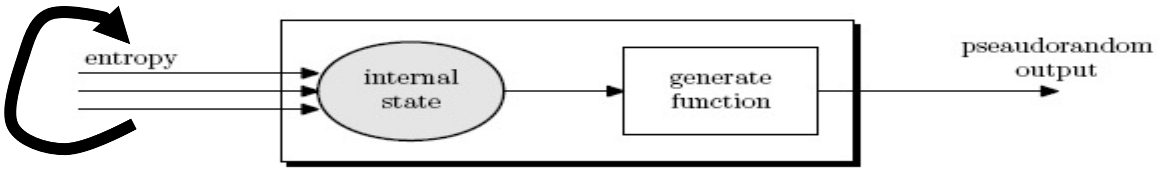
Performance:

Crypto++ 5.6.0 [Wei Dai]

AMD Opteron, 2.2 GHz (Linux)

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

PRG Security Defs

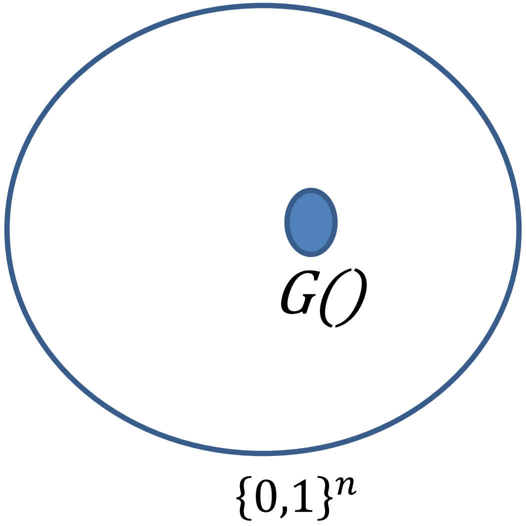
Let G:K ⟶ $\{0,1\}^n$ be a PRG

<u>Goal</u>: define what it means that

$$[k \xleftarrow{R} K, output\ \ G(k)]$$

is "indistinguishable" from

$$[r \xleftarrow{R} \{0,1\}^n, output\ r]$$



Statistical Tests

Statistical test on $\{0,1\}^n$:

an alg. A s.t. A(x) outputs "0"(not random) or "1"(random)

Examples:

(1) A(x)=1 iff $|\#0(x) - \#1(x)| \le 10 \cdot \sqrt{n}$

(2) A(x)=1 iff $|\#00(x) - n/4| \le 10 \cdot \sqrt{n}$

Statistical Tests

More examples:

(3) A(x)=1 iff max-run-of-0(x)<10$\cdot \log_2(n)$

…

…

Advantage

Let G:K $\longrightarrow \{0,1\}^n$ be a PRG and A a stat. test on $\{0,1\}^n$

Define:

$$Adv_{PRG}[A,G] = \left| Pr_{k \xleftarrow{R} K}\left[A\big(G(k)\big) = 1\right] - Pr_{r \xleftarrow{R} \{0,1\}^n}\left[A(r) = 1\right] \right| \in [0,1]$$

Adv close to 1 $\Longrightarrow$ A can dist. G from random

Adv close to 0 $\Longrightarrow$ A cannot dist. G from random

A silly example: A(x) = 0 $\Rightarrow$ Adv_{PRG} [A,G] =

Suppose $G: K \longrightarrow \{0,1\}^n$ satisfies **$msb(G(k)) = 1$** for 2/3 of keys in K

Define stat. test $A(x)$ as:

if [$msb(x)=1$] output "1" else output "0"

Then

$$Adv_{PRG}[A,G] = \big|\, Pr[\, A(G(k))=1] - Pr[\, A(r)=1 \,] \,\big| =$$

Secure PRGs: crypto definition

Def: We say that $G: K \longrightarrow \{0,1\}^n$ is a **secure PRG** if

$\forall$ "eff" stat. test A:

$\text{Adv}_{\text{PRG}}[A,G]$ is "negligible"

Are there provably secure PRGs?

but we have heuristic candidates.

More Generally

Let P_1 and P_2 be two distributions over $\{0,1\}^n$

Def: We say that P_1 and P_2 are

computationally indistinguishable (denoted $P_1 \approx_p P_2$)

if $\forall$ "eff" stat. test A:

$$\left| \Pr_{x \leftarrow P_1}[(A(x)=1] - \Pr_{x \leftarrow P_2}[(A(x)=1] \right| < \text{"negligible"}$$

Example: a PRG is secure if $\{ k \xleftarrow{R} K : G(k) \} \approx_p \text{uniform}(\{0,1\}^n)$

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 $\forall\, m_0, m_1 \in M \quad (\ |m_0| = |m_1|\ )$

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

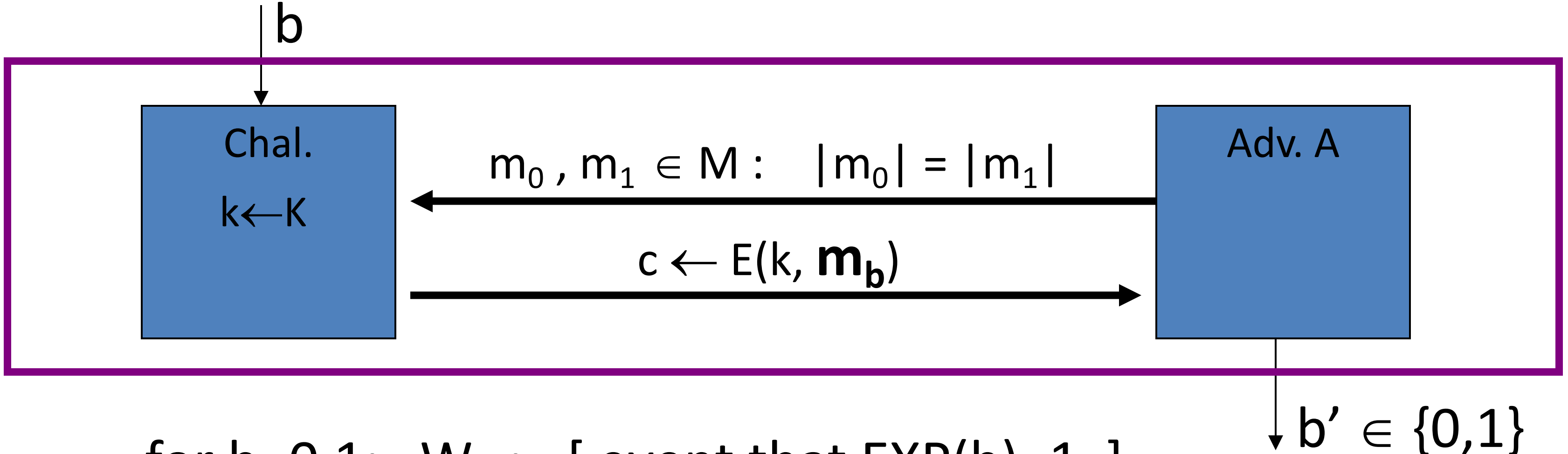
(E,D) has perfect secrecy if $\forall\, m_0, m_1 \in M \quad (\ |m_0| = |m_1|\ )$

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

... 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 $b=0,1$: $\quad W_b :=$ [event that $\text{EXP}(b)=1$]

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

Semantic Security (one-time key)

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

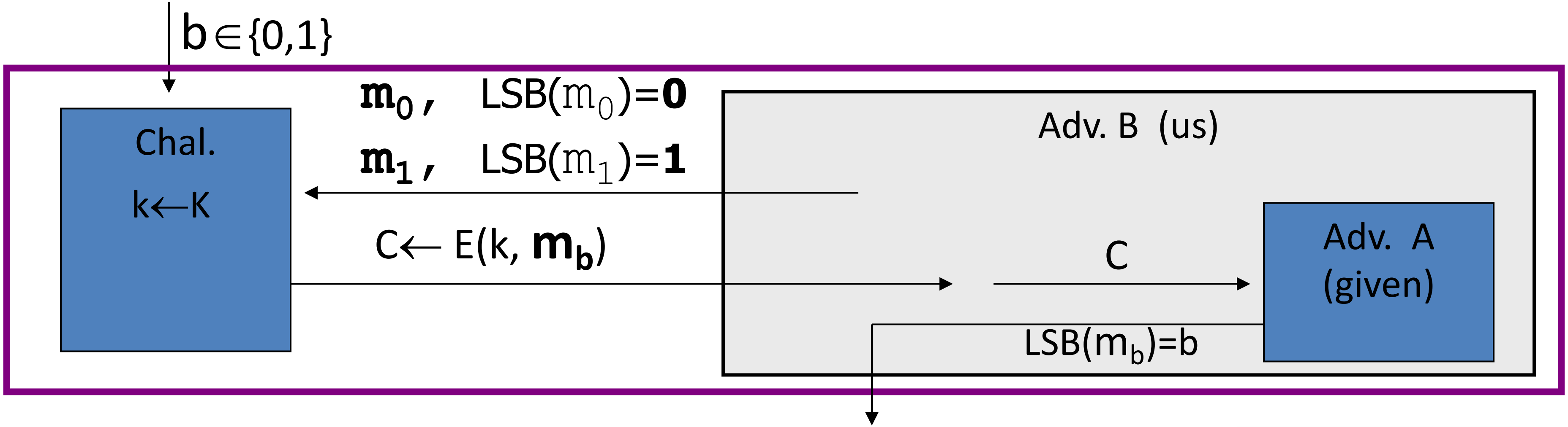
$\mathrm{Adv}_{SS}[A,\mathbb{E}]$ is negligible.

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

Examples

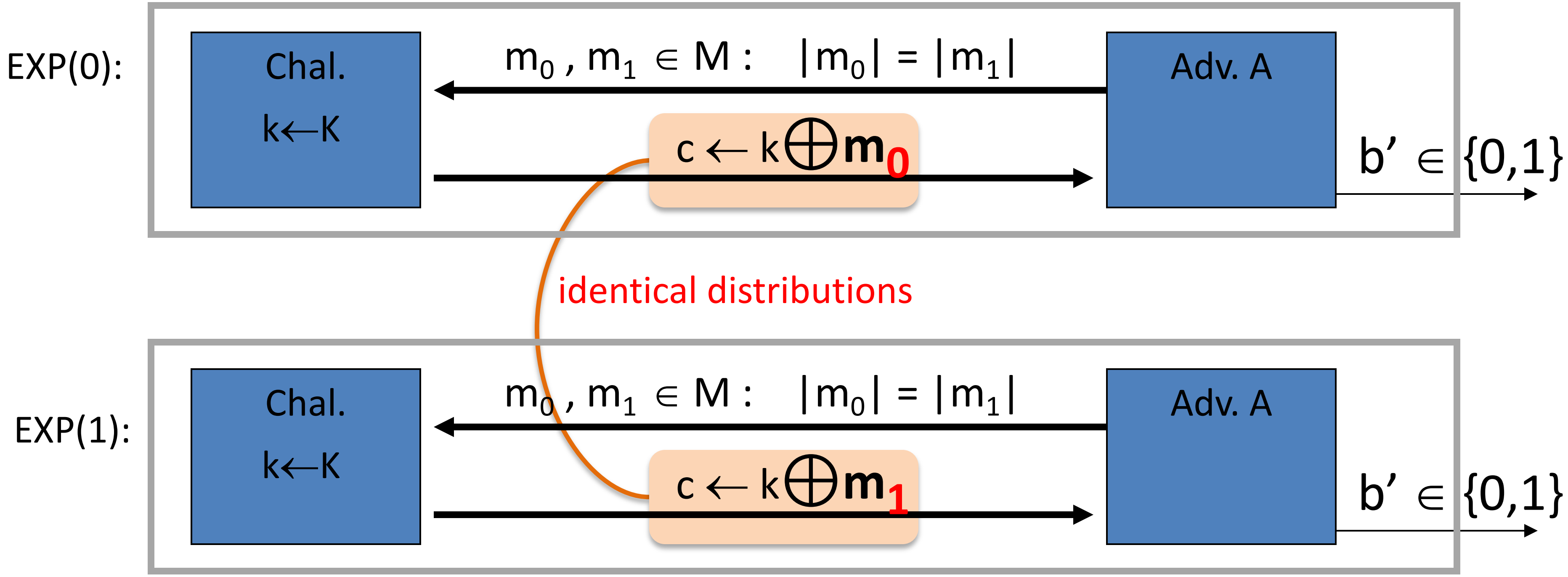
Suppose efficient A can always deduce LSB of PT from CT.

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



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

OTP is semantically secure



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

Stream ciphers are semantically secure

Goal: secure PRG $\Rightarrow$ semantically secure stream cipher

Stream ciphers are semantically secure

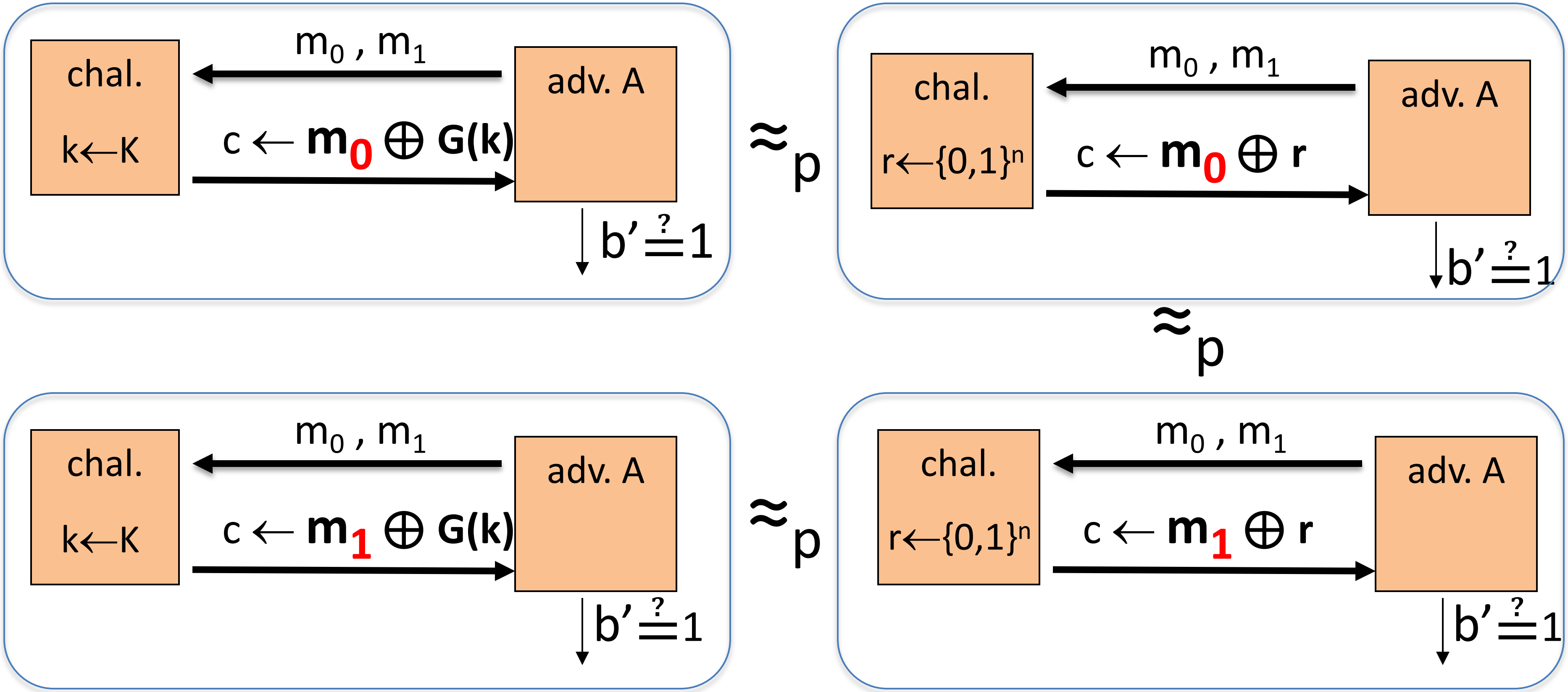
Thm: $G: K \longrightarrow \{0,1\}^n$ is a secure PRG $\Rightarrow$

stream cipher E derived from G is sem. sec.

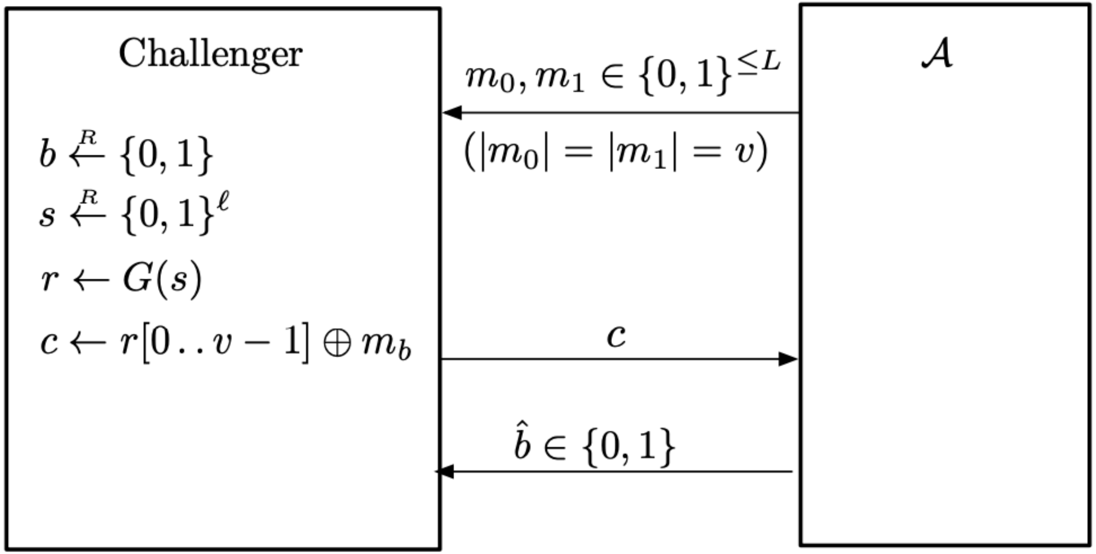
$\forall$ sem. sec. adversary A , $\exists$a PRG adversary B s.t.

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

Proof: intuition



Game 0



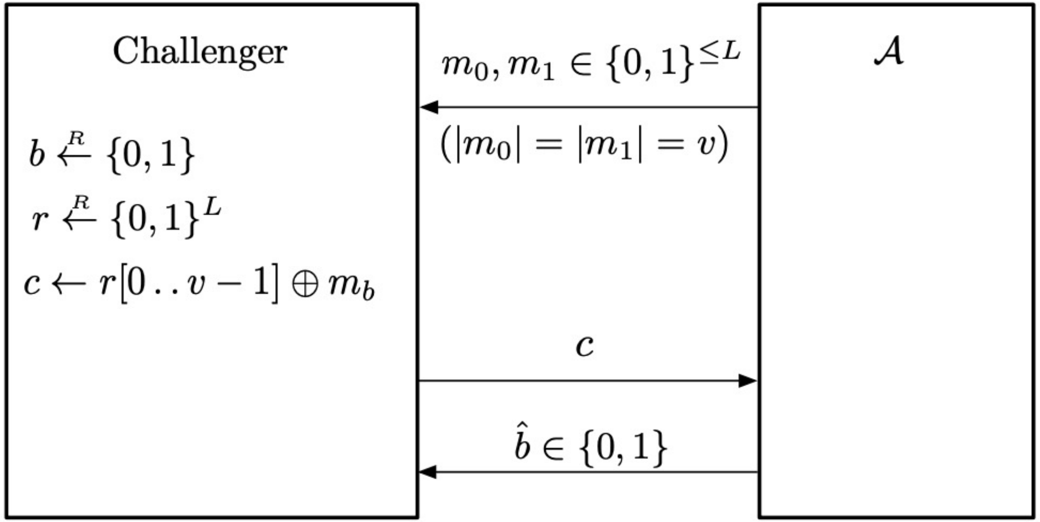
W_0: the event that $\hat{b} = b$ in Game 0

$$\text{SSadv}^*[\mathcal{A}, \mathcal{E}] = |\Pr[W_0] - 1/2|$$

Upon receiving $m_0, m_1 \in \{0,1\}^v$ from $\mathcal{A}$, for some $v \leq L$, do:

$b \xleftarrow{R} \{0,1\}$
$s \xleftarrow{R} \{0,1\}^\ell$, $r \leftarrow G(s)$
$c \leftarrow r[0..v-1] \oplus m_b$
send c to $\mathcal{A}$.

Game 1



Upon receiving $m_0, m_1 \in \{0,1\}^v$ from $\mathcal{A}$, for some $v \leq L$, do:

$b \xleftarrow{R} \{0,1\}$

$r \xleftarrow{R} \{0,1\}^L$

$c \leftarrow r[0..v-1] \oplus m_b$

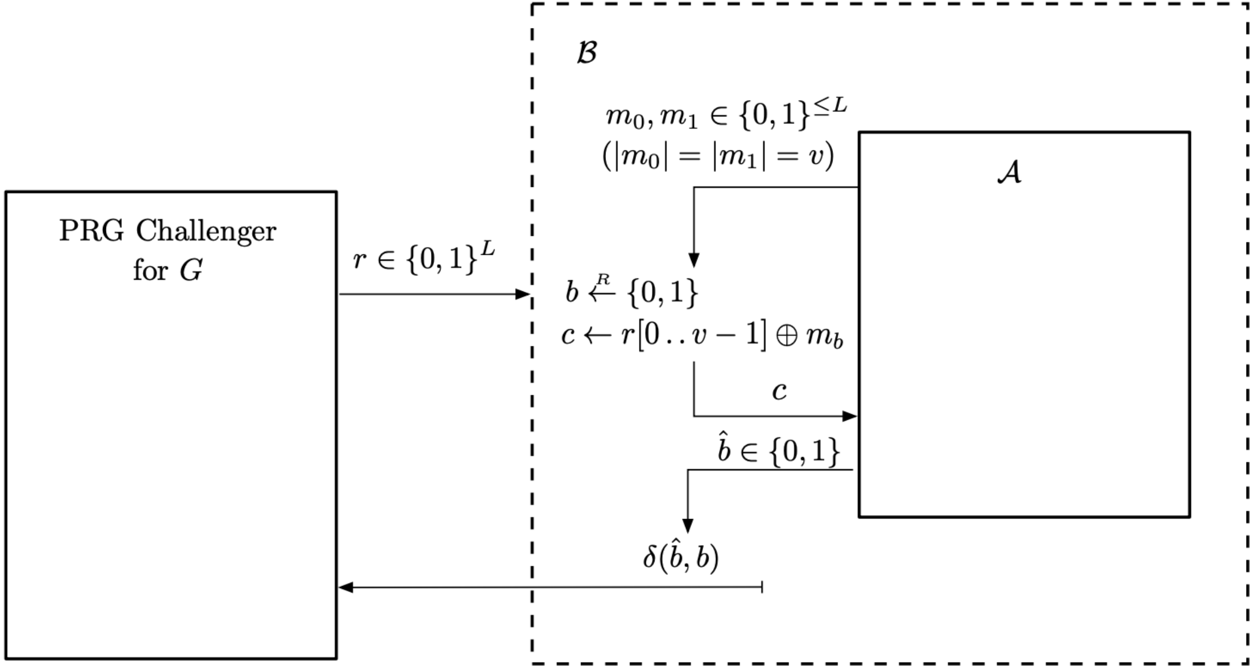
send c to $\mathcal{A}$.

W_1: the event that $\hat{b} = b$ in Game 1

$$\Pr[W_1] = 1/2.$$

Our goal: $|\Pr[W_0] - \Pr[W_1]| = \mathrm{PRGadv}[\mathcal{B}, G].$

Define $\delta(x,y) := \begin{cases} 1 & \text{if } x = y, \\ 0 & \text{if } x \neq y. \end{cases}$



P0: probability that B output 1 when PRG challenger is running exp 0.

P1: probability that B output 1 when PRG challenger is running exp 1.

$$\mathrm{PRGadv}[\mathcal{B}, G] = |p_1 - p_0|.$$

$$p_0 = \Pr[W_0], \quad p_1 = \Pr[W_1].$$

$\Downarrow$

$$|\Pr[W_0] - \Pr[W_1]| = \mathrm{PRGadv}[\mathcal{B}, G].$$

Upon receiving $m_0, m_1 \in \{0,1\}^v$ from $\mathcal{A}$, for some $v \leq L$, do:

$b \xleftarrow{R} \{0,1\}$
$c \leftarrow r[0 \mathinner{..} v-1] \oplus m_b$
send c to $\mathcal{A}$.

Finally, when $\mathcal{A}$ outputs a bit $\hat{b}$, $\mathcal{B}$ outputs the bit $\delta(\hat{b}, b)$.

Recall PRG game