Stream ciphers

The One Time Pad
PRG

Symmetric Ciphers: definition

<u>Def</u>: a **cipher** defined over $(\mathcal{K}, \mathcal{M}, \mathcal{C})$

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

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

s.t. $\forall m \in \mathcal{M}, k \in \mathcal{K}$: $D(k, E(k,m)) = m$

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

The One Time Pad (Vernam 1917)

First example of a "secure" cipher

$$\mathcal{M} = \mathcal{C} = \{0,1\}^n \quad , \qquad \mathcal{K} = \{0,1\}^n$$

key = (random bit string as long the message)

The One Time Pad (Vernam 1917)

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

$$D(k,c) = k \oplus c$$

msg:	0	1	1	0	1	1	1	$\oplus$
key:	1	0	1	1	0	1	0	
CT:	1	1	0	1	1	0	1	
	0	1	1	0	1	1	1	

Indeed:

$$D(k, E(k,m)) = D(k, k \oplus m) = k \oplus (k \oplus m) = (k \oplus k) \oplus m = 0 \oplus m = m$$

You are given a message (m) and its OTP encryption (c).

Can you compute the OTP key from m and c ?

Yes, the key is $k = m \oplus c$.

The One Time Pad (Vernam 1917)

Very fast enc/dec !!

… but long keys (as long as plaintext)

Is the OTP secure? What is a secure cipher?

What is a secure cipher?

Attacker's abilities: **CT only attack** (for now)

Possible security requirements:

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

$E(k,m) = m$ would be secure

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

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

Shannon's idea:

CT should reveal no "info" about PT

Information Theoretic Security

(Shannon 1949)

Def: A cipher $(\boldsymbol{E}, \boldsymbol{D})$ over $(\mathcal{K}, \mathcal{M}, \mathcal{C})$ has **perfect secrecy** if

$\forall m_0, m_1 \in \mathcal{M}$ $\quad (\mathrm{len}(m_0) = \mathrm{len}(m_1))$ $\quad$ and $\quad \forall c \in \mathcal{C}$

$$\boxed{\Pr[E(k, m_0) = c] = \Pr[E(k, m_1) = c]}$$

where k is uniform in $\mathcal{K}$ $\quad (k \xleftarrow{R} \mathcal{K})$

m_0

m_1 $\xrightarrow{E}$ c

$$\frac{1}{|\mathcal{K}|}$$

Information Theoretic Security

Def: A cipher ***(E,D)*** over (K,M,C) has **perfect secrecy** if

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

$$Pr[\,E(k,m_0)=c\,] \;=\; Pr[\,E(k,m_1)=c\,] \qquad \text{where } k \xleftarrow{R} K$$

⇒ Given CT can't tell if msg is m_0 or m_1 (for all m_0, m_1)

⇒ most powerful adv. learns nothing about PT from CT

⇒ no CT only attack!! (but other attacks possible)

已知密文攻击

Lemma: OTP has perfect secrecy.

Proof:

$\forall m, c$: $\Pr_k [E(k,m)=c] = \dfrac{\#\text{keys } k \in \mathcal{K} \text{ s.t. } E(k,m)=c}{|\mathcal{K}|}$

So: if $\forall m, c$: $\#\{k \in \mathcal{K} : E(k,m)=c\} = \text{const.}$

$\Longrightarrow$ cipher has perfect secrecy

Let $m \in \mathcal{M}$ and $c \in \mathcal{C}$.

How many OTP keys map $\boldsymbol{m}$ to $\boldsymbol{c}$?

- 2
- 1
- Depends on the message

Lemma: OTP has perfect secrecy.

Proof:

For OTP: $\forall m, c$: if $E(k,m) = c$

$\Longrightarrow k \oplus m = c \Longrightarrow k = m \oplus c$

$\Longrightarrow \boxed{\#\{k \in \mathcal{K} : E(k,m) = c\} = 1}$

$\Longrightarrow$ OTP has perfect secrecy ∎

The bad news …

Thm: perfect secrecy $\Rightarrow$ $|\mathcal{K}| \geq |\mathcal{M}|$

i.e. perfect secrecy $\Longrightarrow$ key-len $\geq$ msg-len

$\Longrightarrow$ hard to use in practice !!

Review

Cipher over (K,M,C): a pair of "efficient" algs $(\boldsymbol{E}, \boldsymbol{D})$ s.t.

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

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 ⇒ key-len ≥ msg-len

Stream Ciphers: making OTP practical

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

PRG is a function $G: \underbrace{\{0,1\}^s}_{\text{seed space}} \rightarrow \{0,1\}^n$ $n \gg s$

(eff. computable by a deterministic algorithm)

s s

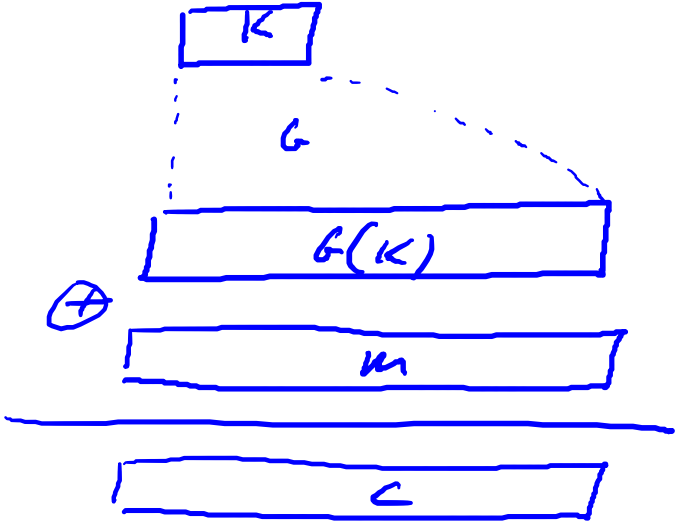
k k

Sender Receiver

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?

A Yes, if the PRG is really “secure”

B No, there are no ciphers with perfect secrecy

C Yes, every cipher has perfect secrecy

D 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

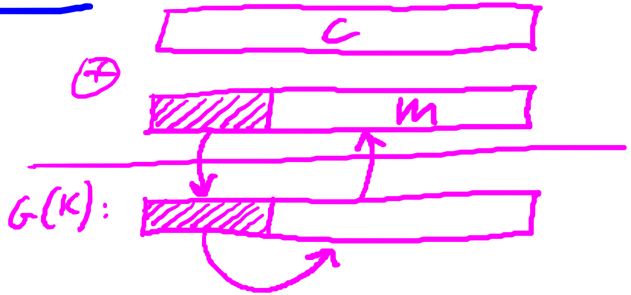
- Security will depend on specific PRG

PRG must be unpredictable

Suppose PRG is predictable:

$$\exists i: \quad G(k)\big|_{1,\dots,i} \xrightarrow{\text{alg}} G(k)\big|_{i+1,\dots,n}$$

Then:



even $G(k)\big|_{1,\dots,i} \to G(k)\big|_{i+1}$

is a problem!

PRG must be unpredictable

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

$\exists$ "eff" alg. A and $\exists\, 0 \le i \le n-1$ s.t.

$$\Pr_{k \xleftarrow{R} \mathcal{K}}\left[A\left(G(k)\big|_{1,\dots,i}\right) = G(k)\big|_{i+1} \right] > \frac{1}{2} + \varepsilon$$

For non-negligible ε (e.g. $\varepsilon = 1/2^{30}$)

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:

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

output bits of $r[i]$

i++

seed = 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)

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: $\exists d\text{:}\ \varepsilon(\lambda) \geq 1/\lambda^{d}$ inf. often (ε ≥ 1/poly, for many λ)
  - ε negligible: $\forall d, \lambda \geq \lambda_d\text{:}\ \varepsilon(\lambda) \leq 1/\lambda^{d}$ (ε ≤ 1/poly, for large λ)

Few Examples

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

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

$$\varepsilon(\lambda) = \begin{cases} 1/2^{\lambda} & \text{for odd } \boldsymbol{\lambda} \\ 1/\lambda^{1000} & \text{for even } \boldsymbol{\lambda} \end{cases}$$

Negligible

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 $\lambda=1,2,3,\ldots$ 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.

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

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

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 !!

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

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

Eavesdropper does:

$$C_1 \oplus C_2 \rightarrow$$

Enough redundancy in English and ASCII encoding that:

$$m_1 \oplus m_2 \rightarrow m_1 , m_2$$

Real world examples

- Project Venona

- MS-PPTP (windows NT):

$k = (k_{C \to S}, k_{S \to C})$

k

m_1

m_2

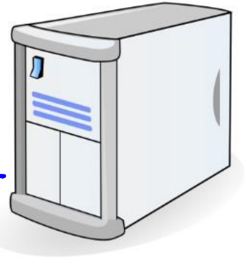
m_3

$[m_1 \| m_2 \| m_3] \oplus G(k)$

s_1

s_2

s_3



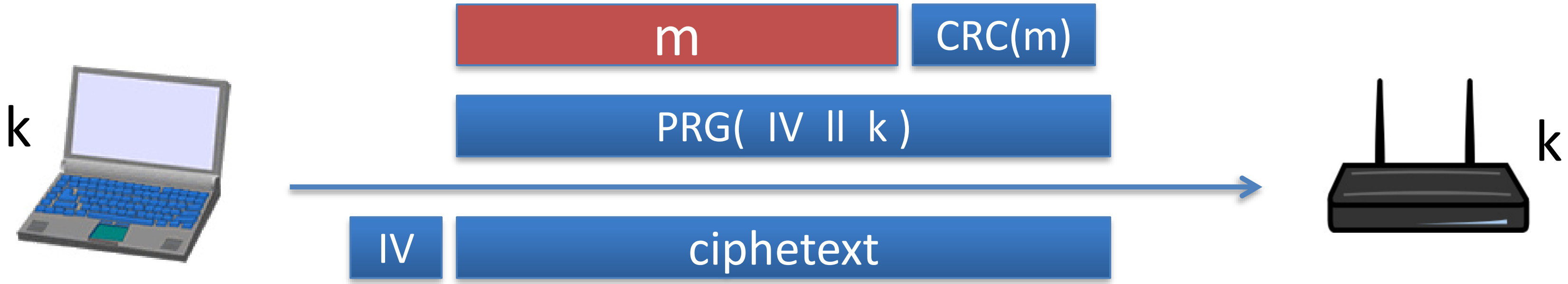
k

$[s_1 \| s_2 \| s_3] \oplus G(k)$

Need different keys for C⟶S and S⟶C

Real world examples

802.11b WEP:

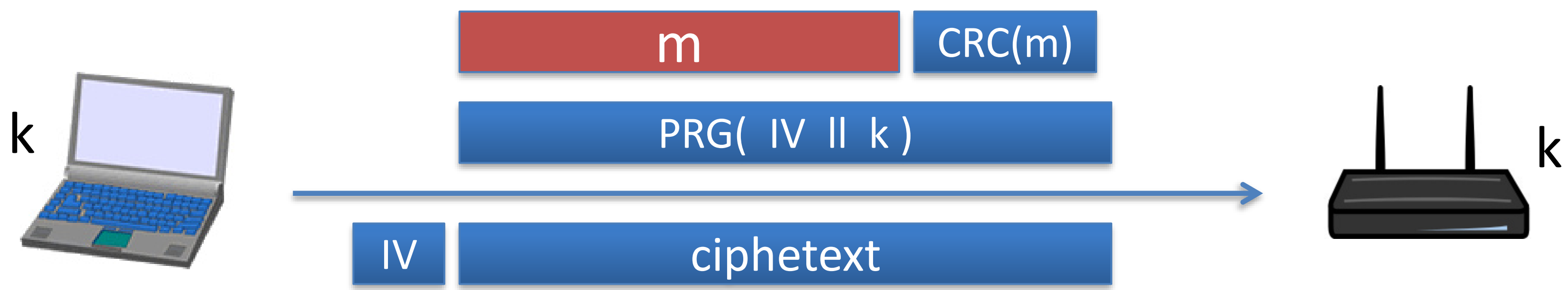


Length of IV: 24 bits

- 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)

⋮

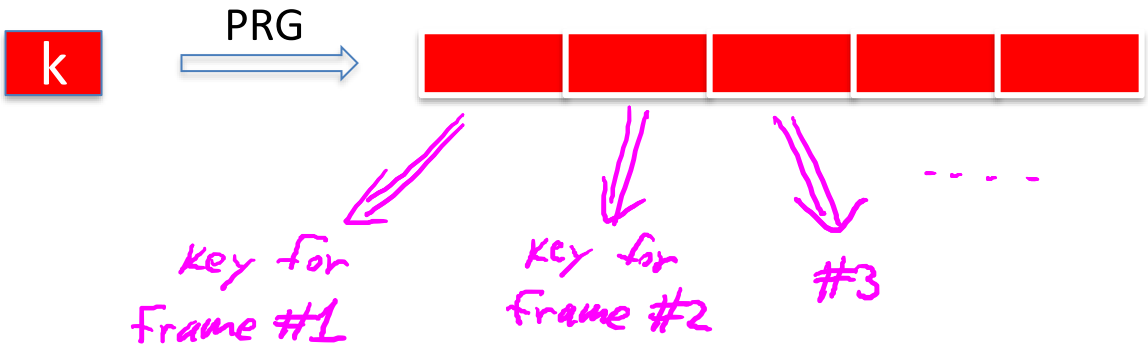
24 bits   1024 bits

For the RC4 PRG:

FMS2001 $\Rightarrow$ can recover k after 10^6 frames

Recent attacks $\approx$ 40,000 frames

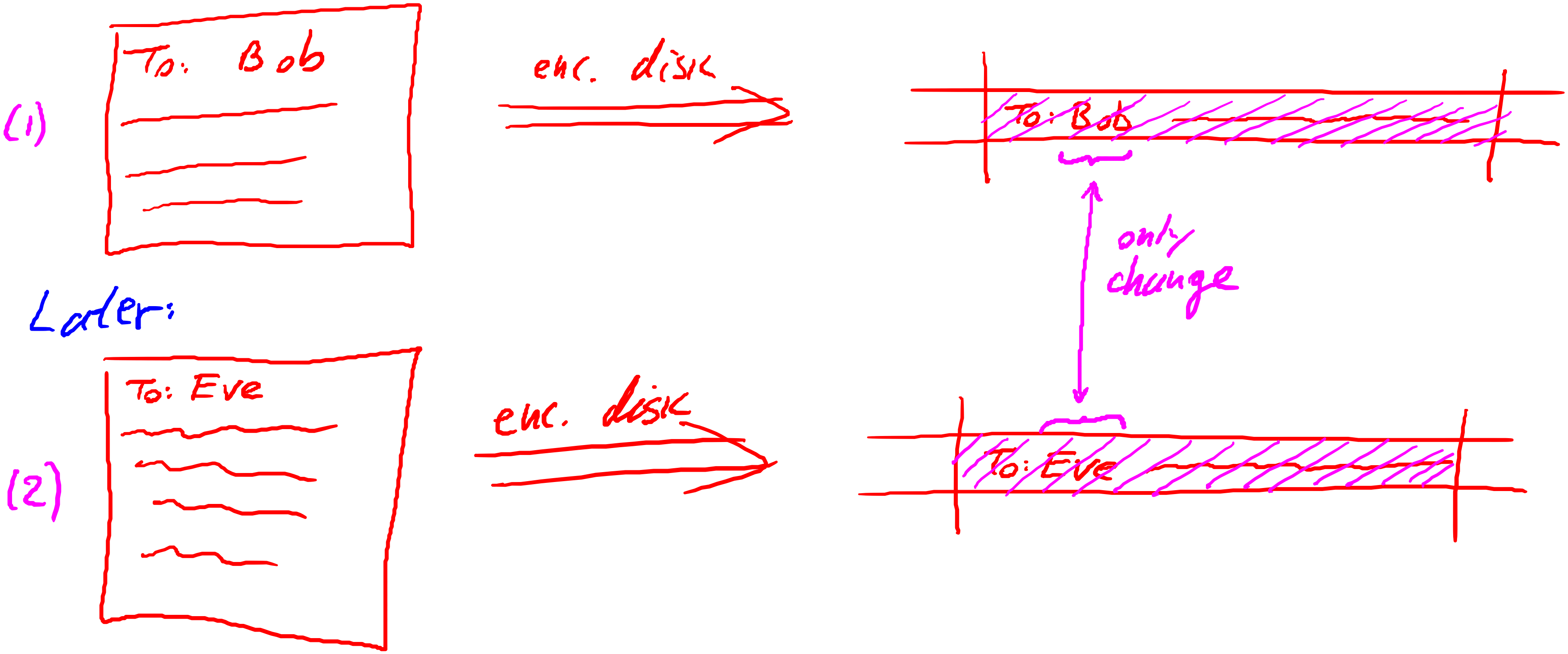
A better construction



⇒ now each frame has a pseudorandom key

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

Yet another example: disk encryption

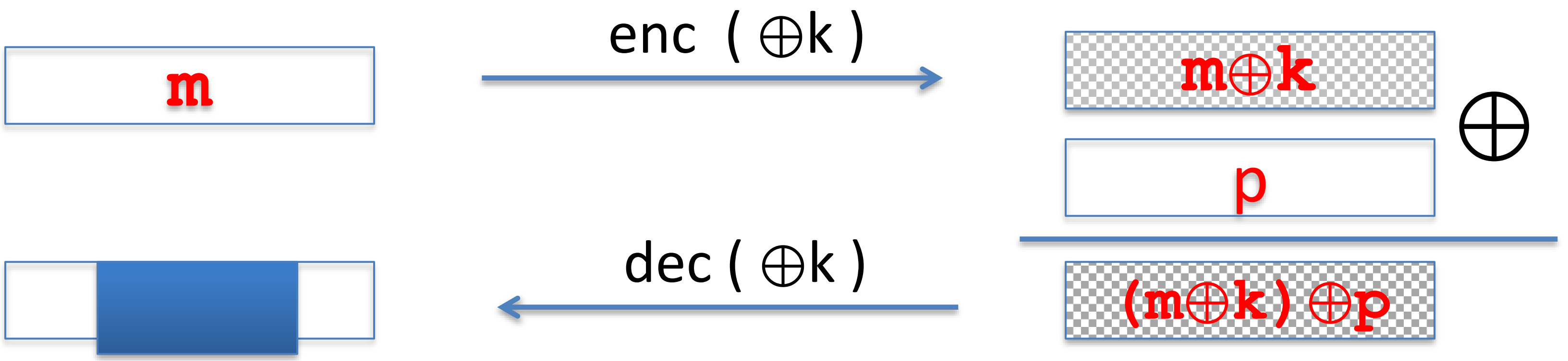


Two time pad: summary

Never use stream cipher key more than once !!

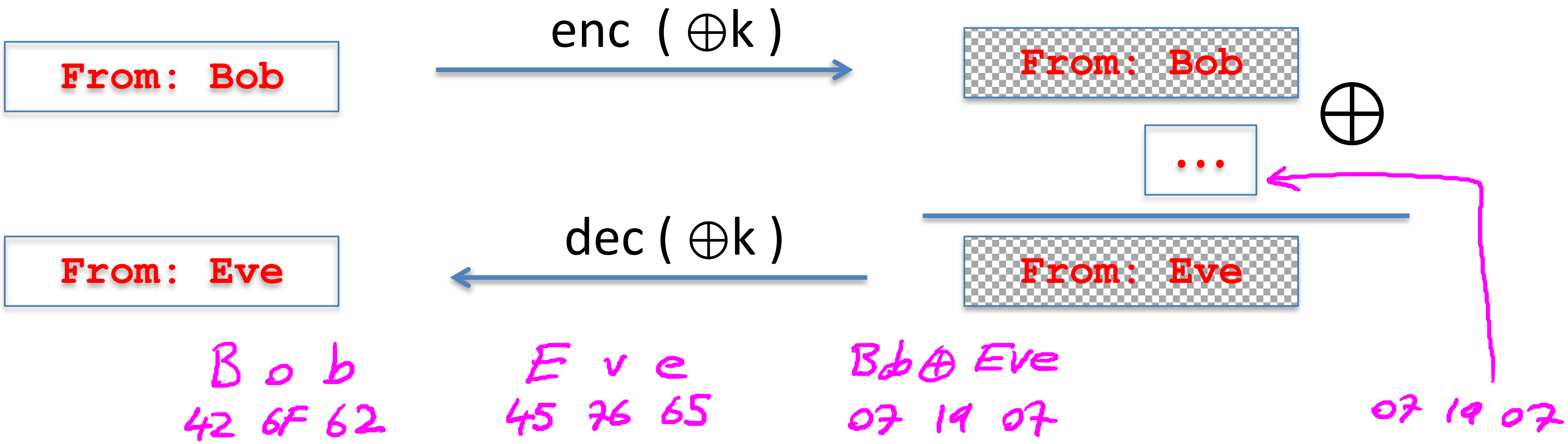
- Network traffic: negotiate new key for every session (e.g. TLS)
- Disk encryption: typically do not use a stream cipher

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)

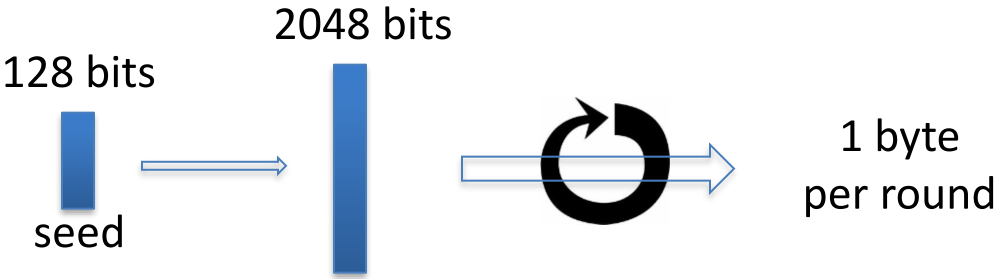


Modifications to ciphertext are undetected and have predictable impact on plaintext

Stream ciphers

Real-world Stream Ciphers

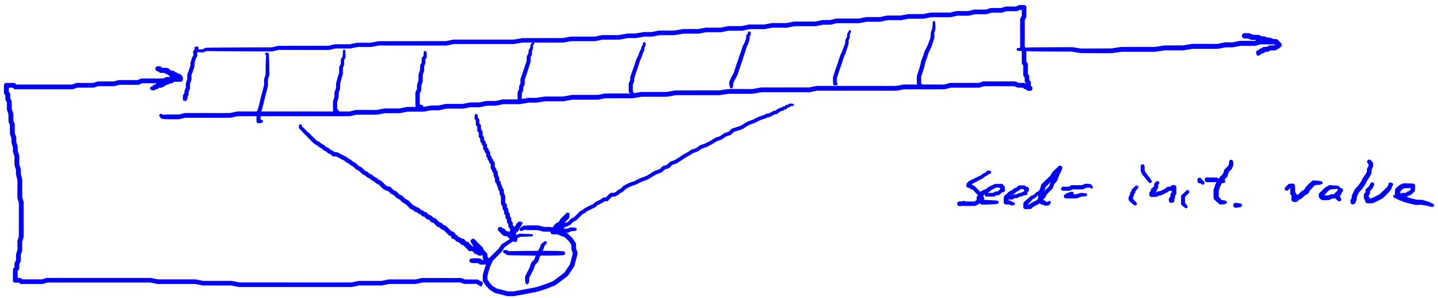
Old example (software): RC4 (1987)



- 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

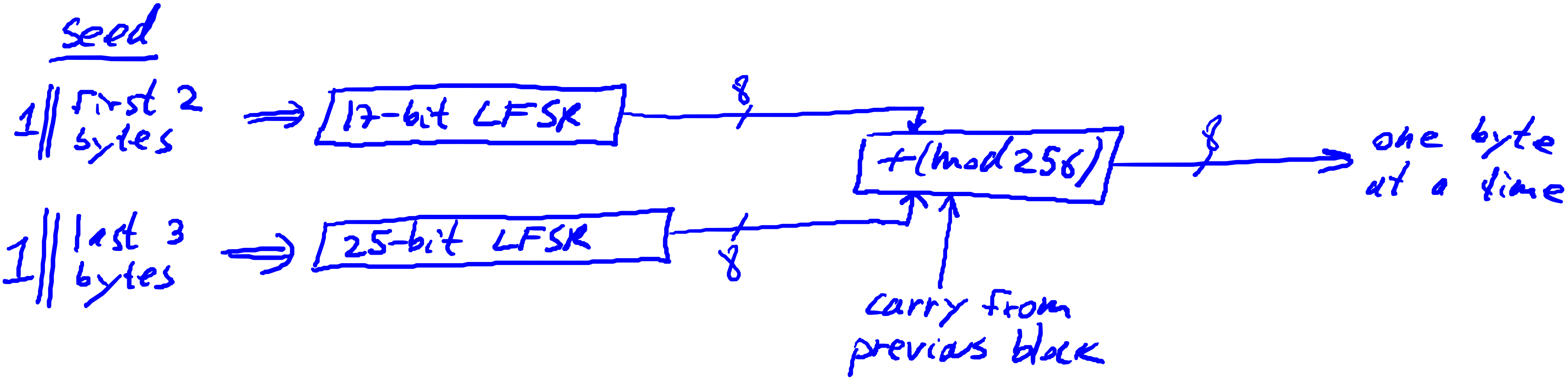
Bluetooth (E0): 4 LFSRs

all broken

参考 教材2.2

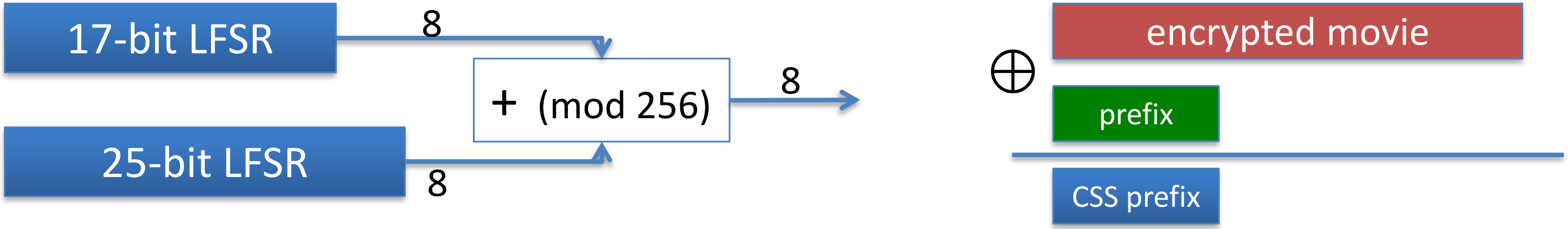
Old example (hardware): CSS (badly broken)

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

https://www.cs.cmu.edu/~dst/DeCSS/FrankStevenson/analysis.html

Modern stream ciphers: eStream

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

nonce

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

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

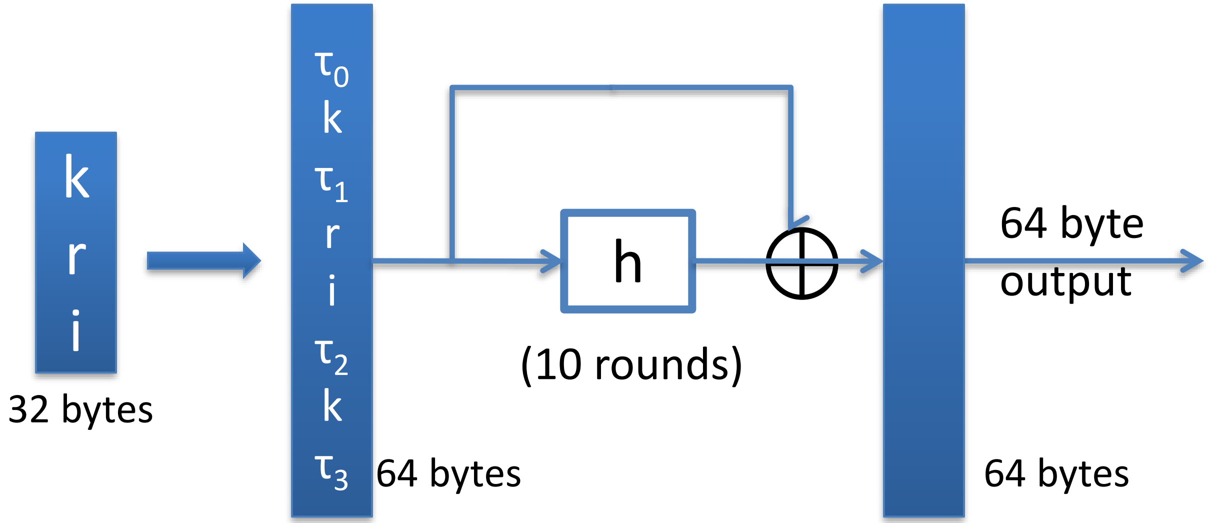
The pair (k,r) is never used more than once.

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)) || H(k , (r, 1)) || …



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

chacha20

Coutinho, Murilo, and Tertuliano C. Souza Neto. "Improved linear approximations to ARX ciphers and attacks against chacha." EUROCRYPT 2021.

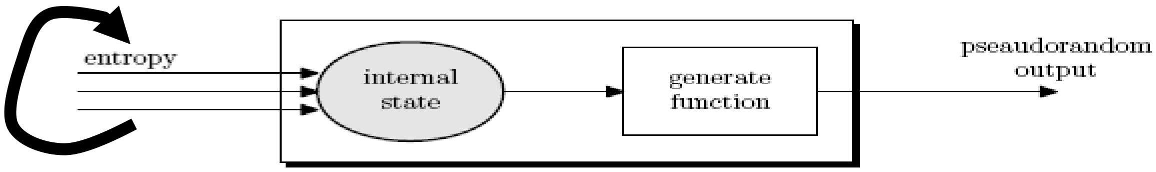
Performance:

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

Stream ciphers

PRG Security Defs

Let $G: K \longrightarrow \{0,1\}^n$ be a PRG

Goal: define what it means that

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

is "indistinguishable" from

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

$\{0,1\}^n$

Statistical Tests

not random

random

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

an alg. A s.t. A(x) outputs "0" or "1"

Examples:

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

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

Statistical Tests

More examples:

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

$\vdots$

Advantage

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

Define:

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

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

Adv close to 0 $\Longrightarrow$ A cannot

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

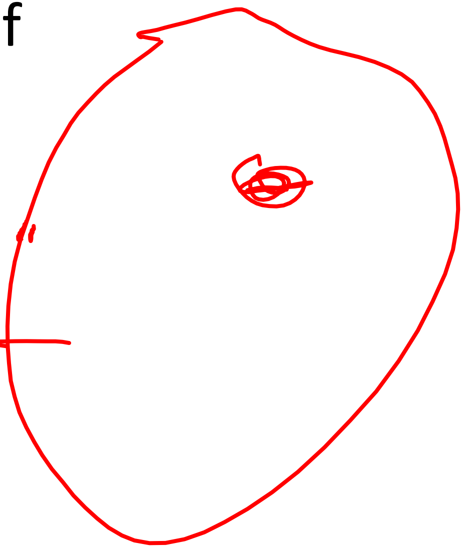
$$\text{Adv}_{\text{PRG}}[A,G] = \Big| \overset{2/3}{\Pr[A(G(k))=1]} - \overset{1/2}{\Pr[A(r)=1]} \Big| =$$

Secure PRGs: crypto definition

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

$\forall$ ~~"eff"~~ stat. tests A:

$Adv_{PRG}[A, G]$ is "negligible"



Are there provably secure PRGs? $\Rightarrow P \neq NP$

but we have heuristic candidates.

Easy fact: a secure PRG is unpredictable

We show: PRG predictable $\Rightarrow$ PRG is insecure

Suppose A is an efficient algorithm s.t.

$$\Pr_{k \xleftarrow{R} \mathcal{K}}\left[A(G(k)|_{1,\dots,i}) = G(k)|_{i+1} \right] > \frac{1}{2} + \varepsilon$$

for non-negligible ε (e.g. ε = 1/1000)

Easy fact: a secure PRG is unpredictable

$x|_{i+1}$

Define statistical test B as:

$$\Longrightarrow B(x) = \boxed{\begin{array}{l} \text{if } A(x|_{1,\dots,i}) = x_{i+1} \text{ output } 1 \\ \text{else output } 0 \end{array}}$$

$r \xleftarrow{R} \{0,1\}^n : \quad \Pr[B(r)=1] = \frac{1}{2}$

$r \xleftarrow{R} \mathcal{K} : \quad \Pr[B(G(k))=1] > \frac{1}{2} + \varepsilon$

$$\Longrightarrow \text{Adv}_{PRG}[B,G] = \left| \Pr[B(r)=1] - \Pr[B(G(k))=1] \right| > \varepsilon$$

Thm (Yao'82): an unpredictable PRG is secure

Let $G: K \longrightarrow \{0,1\}^n$ be PRG

"Thm": if $\forall\, i \in \{0, \ldots, n\text{-}1\}$ PRG G is unpredictable at pos. i

then G is a secure PRG.

If next-bit predictors cannot distinguish G from random

then no statistical test can !!

Let $G: K \longrightarrow \{0,1\}^n$ be a PRG such that

from the last n/2 bits of G(k)

it is easy to compute the first n/2 bits.

Is G predictable for some $i \in \{0, \dots, n-1\}$?

Yes

No

Stream ciphers

Semantic security

Goal: secure PRG ⇒ "secure" stream cipher

Review

Stream Cipher: making OTP practical using a PRG

Linear feedback shift register(LFSR)

Bad constructions: CSS, 802.11b WEP, MS-PPTP

Good construction: Salsa20

Security def. of PRG: unpredictable <=> security [no “effi” sta. test]

Notation

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. tests 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 $\{ k \xleftarrow{R} K : G(k) \} \approx_p \text{uniform}(\{0,1\}^n)$

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

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

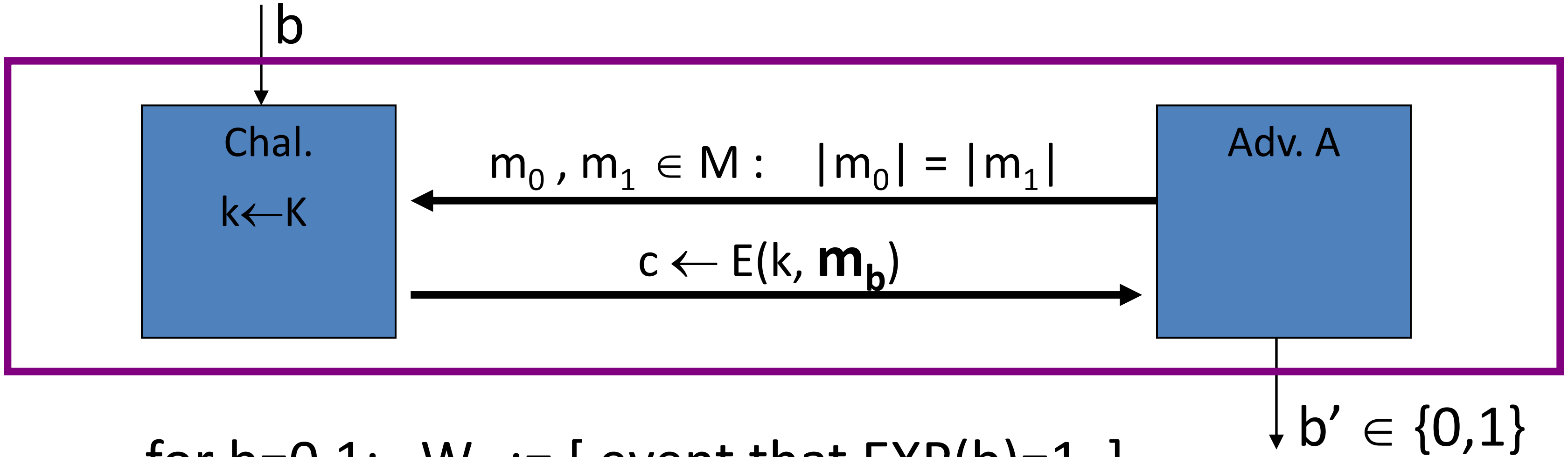
(E,D) has perfect secrecy if $\forall\, m_0, m_1 \in M \quad (\ |m_0| = |m_1|\ )$

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

… 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]

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

Semantic Security (one-time key)

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

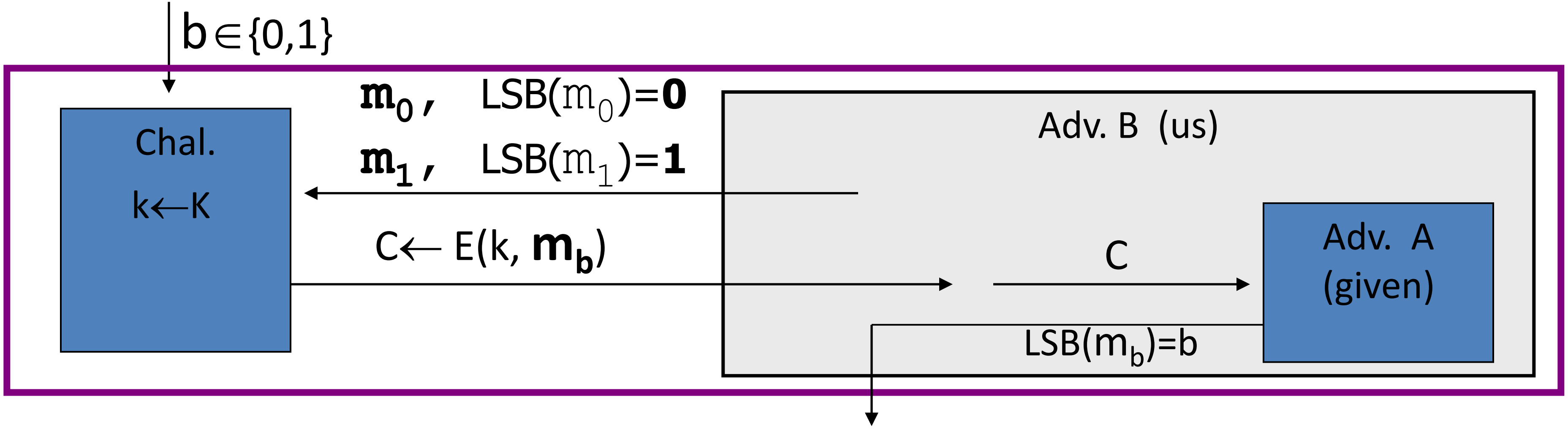
$\mathrm{Adv}_{SS}[A,\mathbb{E}]$ is negligible.

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

Examples

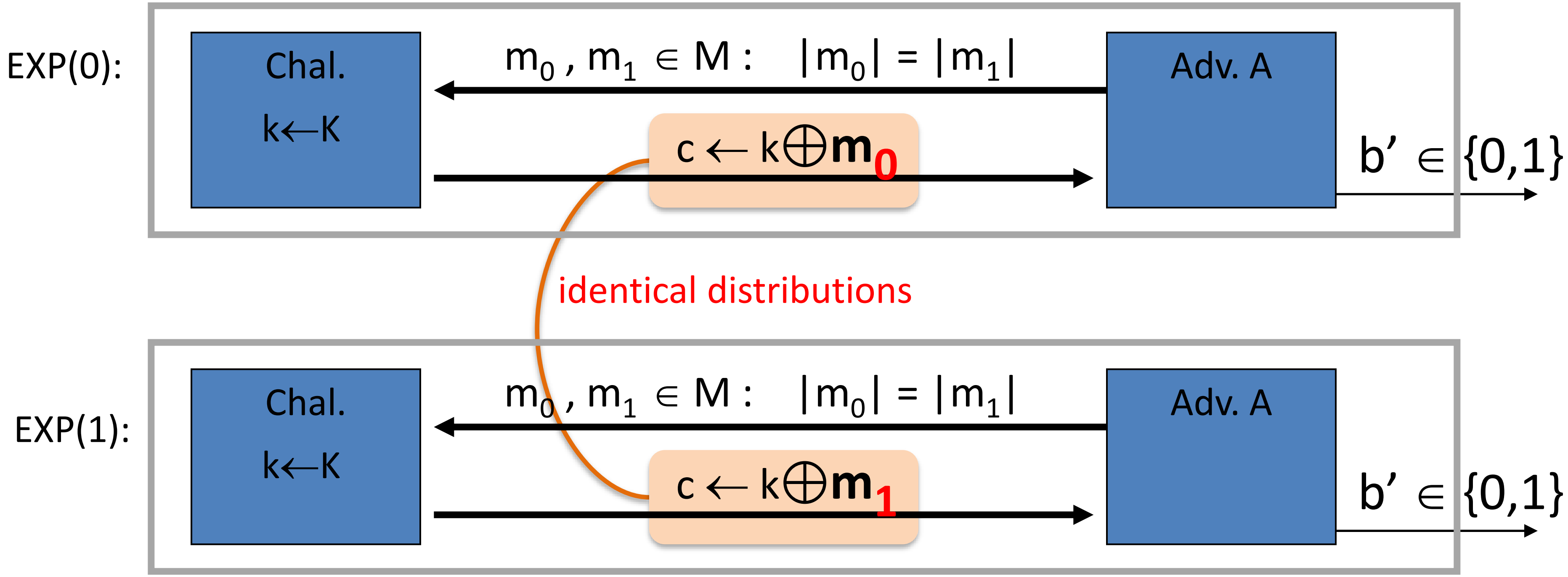
Suppose efficient A can always deduce LSB of PT from CT.

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



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

OTP is semantically secure



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

Stream ciphers

Stream ciphers are semantically secure

Goal: secure PRG ⇒ semantically secure stream cipher

Stream ciphers are semantically secure

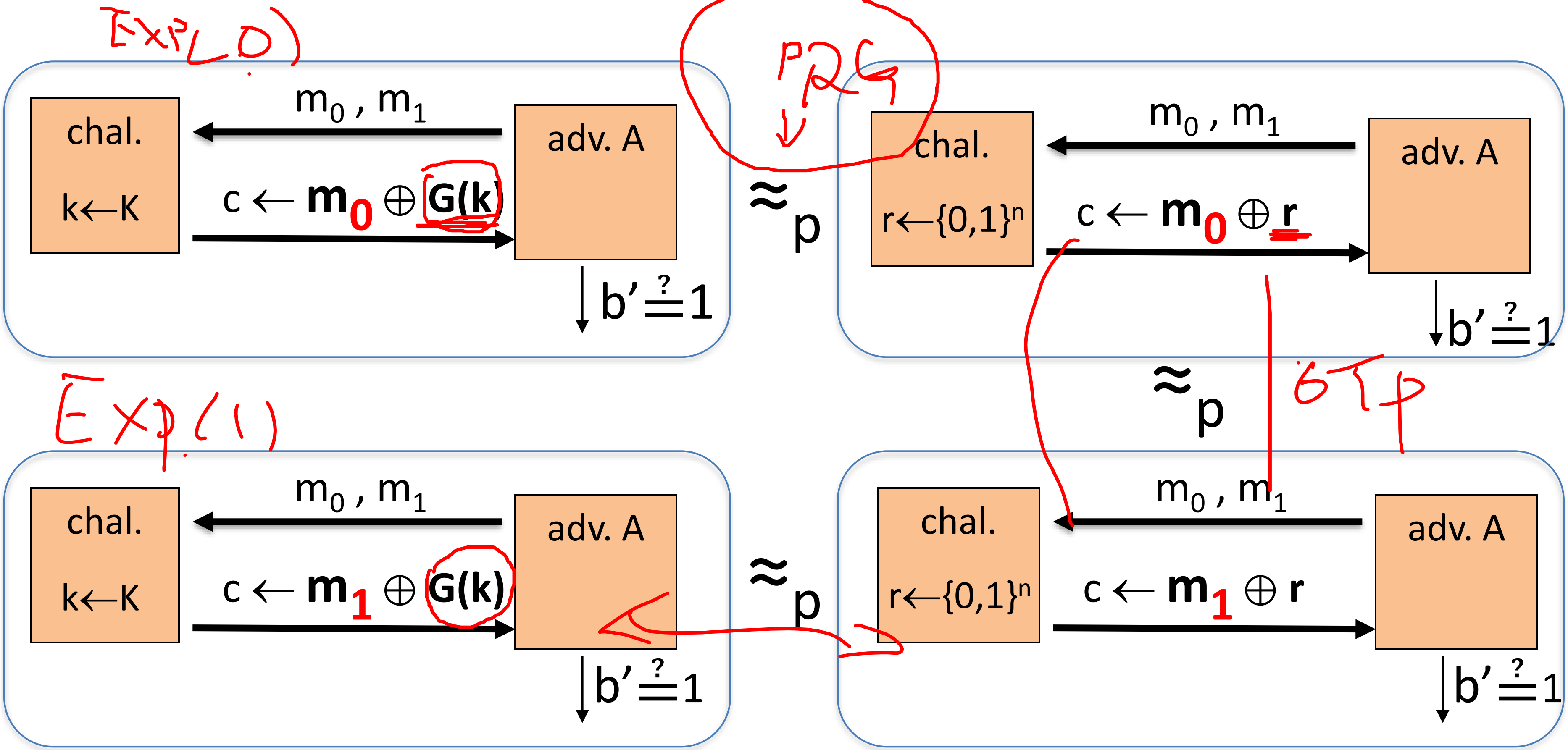
Thm: $G: K \longrightarrow \{0,1\}^n$ is a secure PRG $\Rightarrow$

stream cipher E derived from G is sem. sec.

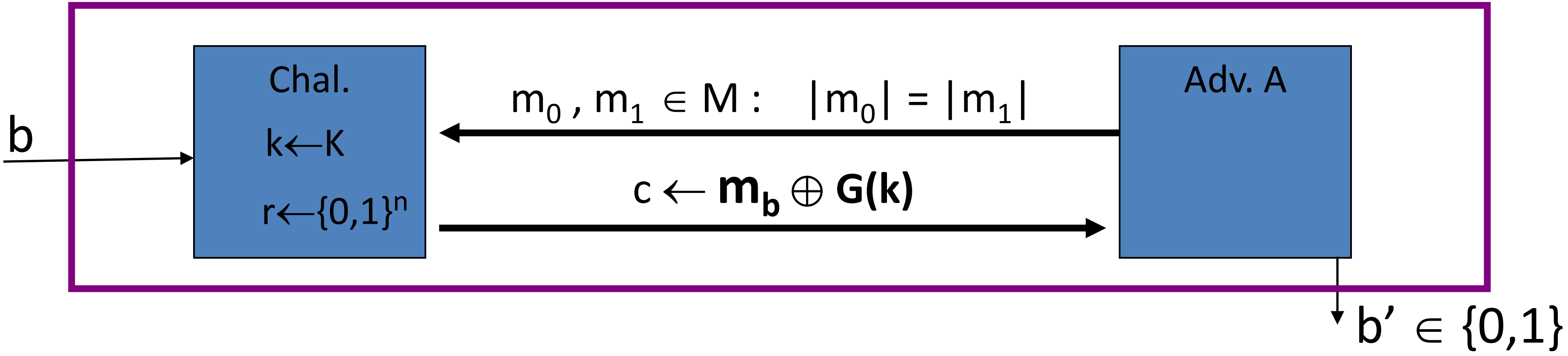
$\forall$ sem. sec. adversary A , $\exists$a PRG adversary B s.t.

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

Proof: intuition



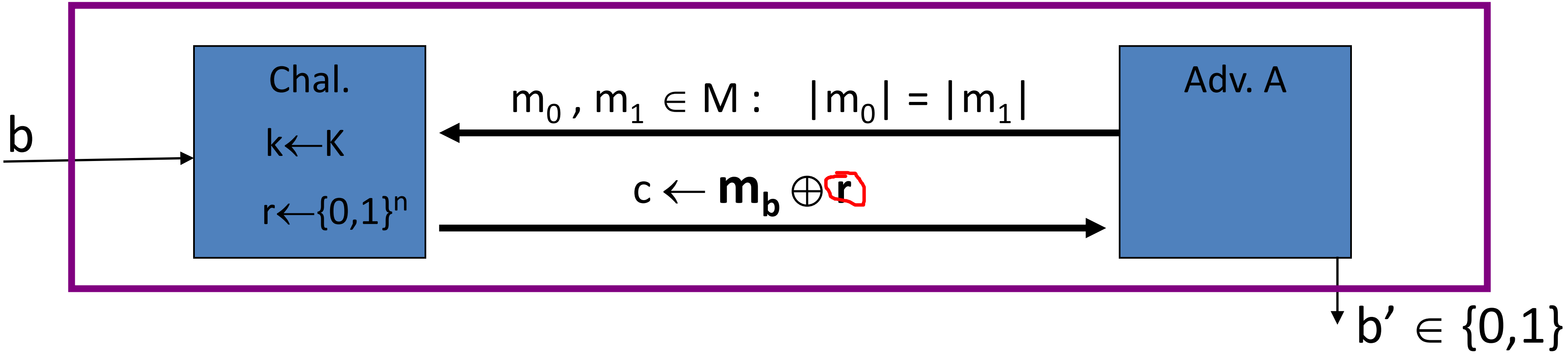
Proof: Let A be a sem. sec. adversary.



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

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

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



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

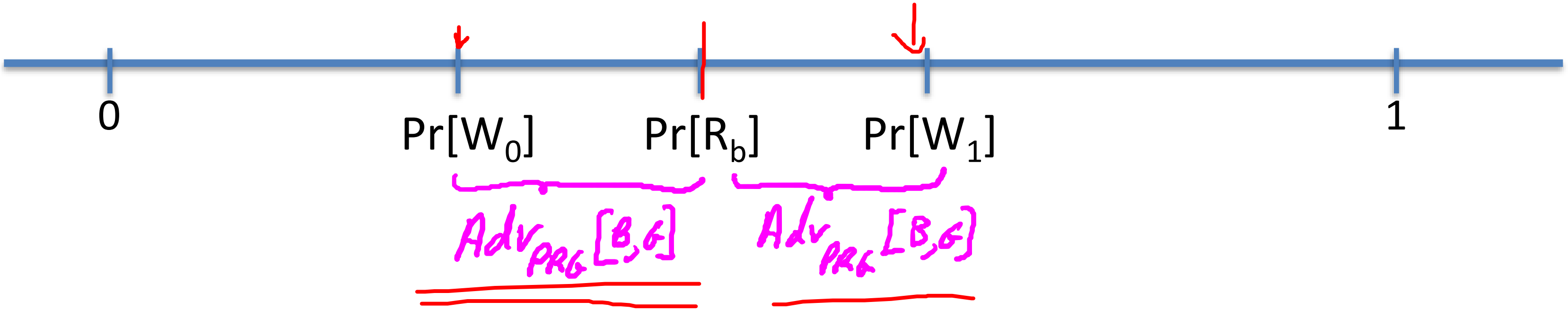
$$\mathrm{Adv}_{SS}[A,E] = \left| \Pr[W_0] - \Pr[W_1] \right|$$

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

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

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

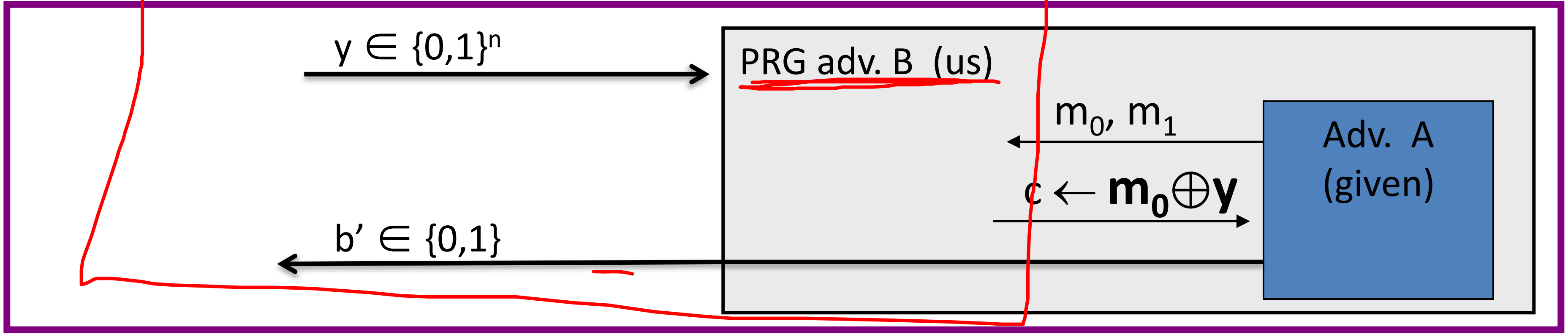
Claim 2: $\exists B$: $\left|\Pr[W_b] - \Pr[R_b]\right| = Adv_{PRG}[B, G]$ for $b = 0, 1$



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

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