Stream Ciphers

"making OTP practical"

idea: replace "random" key $k \xleftarrow{R} K$ by "pseudorandom" key

Pseudorandom generator (PRG): fn $G: \{0,1\}^s \to \{0,1\}^n$, $n \gg s$

($\{0,1\}^s$: seed space; $\{0,1\}^n$: output)

G is "efficiently" computable by a deterministic algorithm

Using a PRG: $G = M = \{0,1\}^n$, $K = \{0,1\}^s$

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

Secrecy $|k| \ll |m| \Rightarrow$ no perfect secrecy

$\Rightarrow$ Need different def. of security

$\Rightarrow$ Security depends on PRG

PRG properties

→ PRG G must be unpredictable

Def: PRG $G: \{0,1\}^s \to \{0,1\}^n$ is unpredictable if

$\forall$ "eff" alg A, $\forall i = 1 \ldots n$:

A cannot compute $G(k)[i]$ (ith bit of $G(k)$) just given $G(k)[0 \ldots i-1]$

More precisely:

$$\left| \Pr\left[A(G(k)[0 \ldots i-1]) = G(k)[i]\right] - \tfrac{1}{2} \right| \text{ is negligible}$$

where $k \xleftarrow{R} \{0,1\}^s$

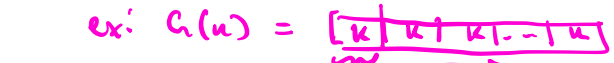
(prob of guessing next bit is not better than coin toss)

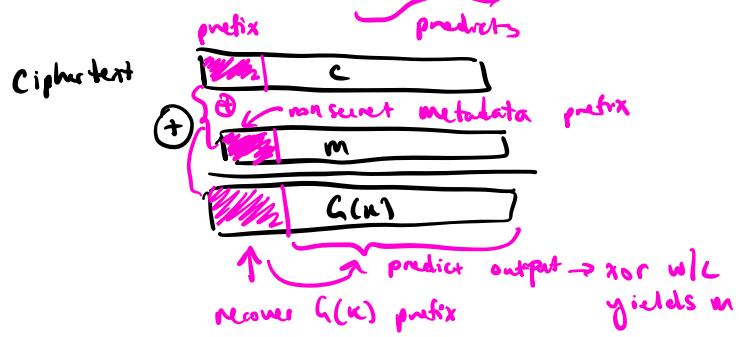
(negligible: $\leq 2^{-80}$)

Why do we need unpredictability?

Suppose G was perfectly predictable: encryption can be broken

$G(k)[0 \ldots i-1]$ can predict $G(k)$

ex: $G(k) =$ 



Weak generators: don't use for crypto

1. Glibc rand()
2. Java Math.random()
3. MS VB RND()

} linear sequential generator

4. /dev/random

What to use?

openSSL cryptorand() function
(when properly seeded)

→ Secure PRG

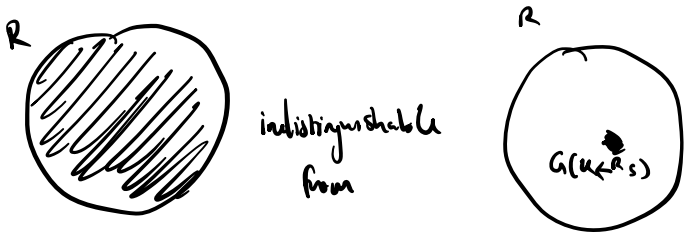
Def: PRG $G: S \to R$ is secure if for all "eff" algs A:

$$Adv[A, G] := \left| Pr[\underbrace{k \xleftarrow{R} S : A(G(k)) = 1}_{\text{event}}] - Pr[\underbrace{r \xleftarrow{R} R : A(r) = 1}_{\text{event}}] \right| \leq 2^{-80}$$

Want to make sure that A outputs 1 at same probability over $G(k)$ and true random as inputs

⇒ behaves the same on PRG's and true random
⇒ cannot distinguish between PRG and true random

PRG security



If $k \xleftarrow{R} S$, we can treat $G(k)$ as if it were uniform in R

Lemma: PRG secure ⇒ PRG unpredictable (converse true, but harder to prove)

Proof: contra positive: unpredictable → adversary predicts entirely
→ if can predict, know it's PRG
→ not secure

Fact: PRG secure ⇒ derived stream cipher "secure"
↑
semantic security, to be defined

Indistinguishability: Distributions P_1, P_2 (over finite set Ω) are indistinguishable ($P_1 \stackrel{c}{\approx} P_2$) if for all "eff" alg A:

$$\left| Pr[x \xleftarrow{R} P_1 : A(x) = 1] - Pr[x \xleftarrow{R} P_2 : A(x) = 1] \right| \leq 2^{-80}$$

For a PRG:

$$P_1 = \{k \xleftarrow{R} S : G(k)\}$$
$$P_2 = \{r \xleftarrow{R} R : r\}$$

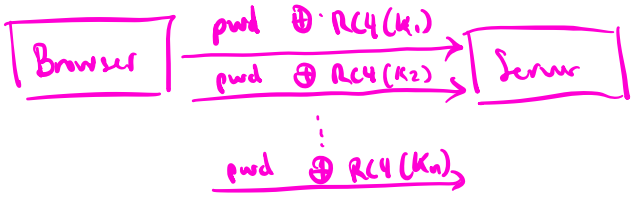
Real-world PRGs (+ derived stream cipher) — see eStream

① Broken: RC4 : used by 802.11b WEP

→ first 256 bytes of output are biased

$P[2^{nd} \text{ byte of } RC4(k) = 0] = \frac{2}{256}$ (twice as big as it should be)

Why is this a problem?



After 10,000 samples, most common byte of CT = 2nd byte of pwd

Eventually, can get all the bytes

Note: any leak (including 1 byte) means the system is broken!

② Good: ChaCha20 : Fast on 64-bit CPUs
widely used on Android

$\pi : \{0,1\}^{512} \to \{0,1\}^{512}$ (64-byte blocks)
fixed permutation

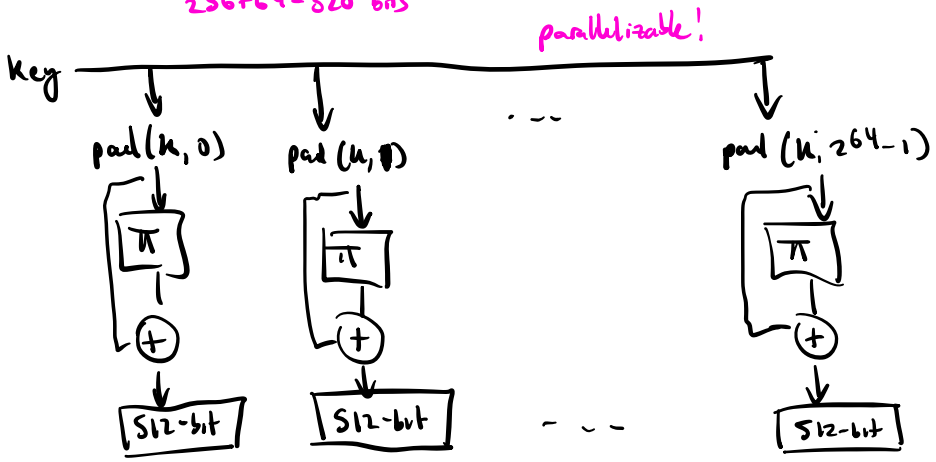
Cannot prove security ⇒ implies $P \neq NP$

$k \in \{0,1\}^{256}$, $j \in \{0, 1, \ldots 2^{64}-1\}$

$pad(k, j)$ → 512-bit output

$256+64 = 320$ bits

input key: 256 bits
Max output: $2^{64} \times 32$ bytes



③ Broken: CSS PRG (DVD encryption)

Attacks on OTP and stream ciphers (HW #0)

① Two-time pad attack!

two-time pad:

$$C_1 \leftarrow M_1 \oplus k$$

$$C_2 \leftarrow M_2 \oplus k$$

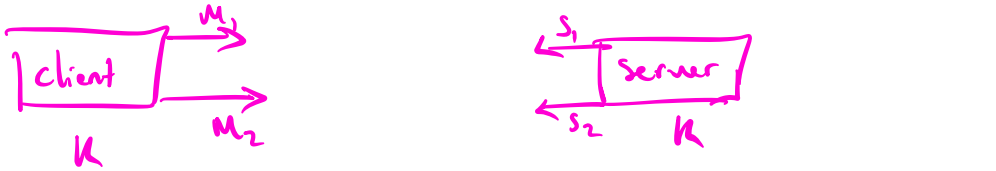
$$C_1 \oplus C_2 = (M_1 \oplus k) \oplus (M_2 \oplus k)$$

$$= M_1 \oplus M_2$$

$\Rightarrow$ often, enough to recover M_1 and M_2 (project Venona)

Stream cipher key should only be used once!

ex. MS PPTP

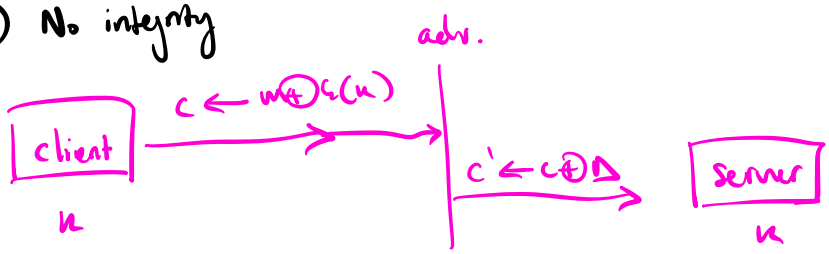


$[M_1, M_2, \ldots] \oplus PRG(k)$ — ok alone, this is OK

$[S_1, S_2, \ldots] \oplus PRG(k)$ — two-time pad!

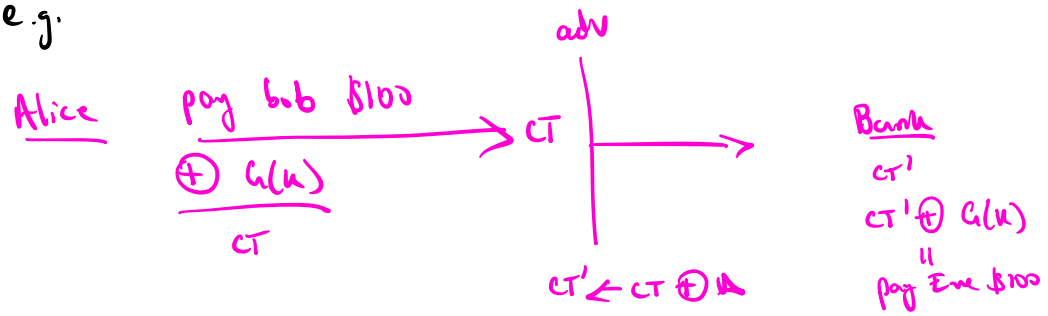
Protocols like TLS generate keys for each direction

② No integrity



$c' \oplus G(k) = m \oplus \Delta$

e.g.



Bob = [42 CF 62]
Eve = [45 26 65]

$\Rightarrow$ Δ = [0000 071407 0000]

↑

Bob $\oplus$ Eve

$\Rightarrow$ OTP + stream ciphers provide no integrity!