S 6160 ryptology Lectu
Pseudorandom Generato

Maria Francis

September 3, 2025

omputational Security

- We had defined the notions of one time computa for encryption schemes and also adversarial indist
- In both cases we ask the adversary to distinguish encryptions of any two messages even if these me chosen maliciously.
- In computational security we require that an effic can distinguish the encryptions of two different m
- This is a weaker notion compared to adversarial indistinguishability/perfect secrecy.
- Does computational security give us more flexibil statistical notions of security?
- More precisely, Can we have a computationally se encryption scheme s.t. $|\mathcal{K}| < |\mathcal{M}|$?
- If $P = NP$, we have no hope!

03/09/202

omputational Security

Theorem

uppose $P = NP$, then for any n there is no one time computationally secure encryption scheme with $\mathcal{K} = \{$
$\mathcal{M} = \{0, 1\}^{n+1}$.

Proof

Let $(Enc, \quad ec)$ be an encryption scheme with key
message space $\mathcal{M}$ and let $\mathcal{A}$ be an attacker for t
computational security game defined as follows:

$\mathcal{A}$ chooses two random messages m_0, m_1 unforml
independently in $\mathcal{M} = \{0, 1\}^{n+1}$.

fter receiving a ciphertext c from $\mathcal{C}$ which encry
$OneSec^b$ for $b \in \{0, 1\}$, $\mathcal{A}$ checks if $\exists k \in \mathcal{K} \quad ec($
and outputs 1 if that is the case and 0 otherwise

omputational Security

Claim : $\mathcal{A}$ is efficient, i.e. probabilistic polynomia

To prove that define $\mathcal{L} := \{(c, m) : \exists k \in \mathcal{K}, \quad ec$

Then $\mathcal{L} \in NP$ as k is a valid witness for any $(c,$
construction.

But by assumption $NP = P$ which implies $\mathcal{L}$ can
efficiently (c, m) in the language and therefore $\mathcal{A}$
check $\exists k \in \mathcal{K} \quad ec(k, c) = m_1$ efficiently.

omputational Security

Claim : $\mathcal{A}$ distinguishes games $OneSec^0$ and One
constant probability.

If $\mathcal{A}$ plays $OneSec^1$ he always outputs 1 because
correctness of the scheme,

$$Pr[OneSec^1_{\mathcal{A}} = 1] = 1$$

Now suppose $\mathcal{A}$ plays $OneSec^0$. Then for any c,
$S := \{ \quad ec(k, c), k \in \mathcal{K}\}$ has size at most $|\mathcal{K}| =$
$\mathcal{A}$ will output 1 if and only if $m_1 \in S$ and since
independently of m_0 and m_1 does not depend on

$$Pr[OneSec^0_{\mathcal{A}} = 1] = Pr[m_1 \in S] \leq |\mathcal{K}|/|\mathcal{M}$$

which concludes the proof.

Pseudorandom Generators

Pseudorandom Generator is a (family of) funct
stretches a random input string (the seed) and o
longer string which looks uniform.

efinition
family of deterministic and efficient to compute fun
$G : (\{0,1\}^n \to \{0,1\}^{\ \ n)})_{n\in}$ s.t. $\forall \ell(n) > n$ is a pseu
generator if

$$G(U_n) \approx U_{\ \ n)}$$

$G(\{0,1\}^n)$ has size at most 2^n whereas $\{0,1\}^{\ \ n)}$
greater than 2^{n+1}. So an unbounded adversary c
between the two distributions. However if G is a
efficient adversary can do the same.

PRGs for omputationally Secure Encryption

ssume a PRG G. Define

$$Enc(k, m) := G(k) \oplus m$$
$$ec(k, c) := G(k) \oplus c,$$

where $\mathcal{K} = \{0,1\}^n$ and $\mathcal{M} = \{0,1\}^{\ n)}$.

Since $\ell(n) > n$, $|\mathcal{K}| < |\mathcal{M}|$.

G is a one time pad.

Suppose $G : (\{0,1\}^n \rightarrow \{0,1\}^{\ n)})_{n \in}$ is a PRG.
$(Enc,\ ec)$ defined above is one time computatio

PRGs for omputationally Secure Encryption

The proof proceeds with a sequence of hybrids: l
$b \in \{0, 1\}$ be the game defining one time compu
security.

We define a sequence a of intermediate games, w
differ in the way c is computed by the challenger

Game $OneSecR^b$, $b \in \{0, 1\}$ – same as $OneSec^b$
computes $c = R \oplus m$ where $R \leftarrow U\ \ _{n)}$.

Game $OneSecR$ – here $\mathcal{C}$ picks c uniformly in

Claim : $OneSec^b \approx OneSecR^b$

$neSec \quad \approx \quad neSecR$

Consider that we have an efficient distinguisher
distinguishes between $OneSec^b$ and $OneSecR^b$ fo
Then this implies a distinguisher R between $G(U$
How?

R receives a string $z \in \{0,1\}$ $^{n)}$ and sends $z \oplus$
's output will be R's output. Clearly R is effic
If z is from $G(U_n)$ then receives samples fron
if z is from U $_{n)}$ then receives samples from
Therefore if can distinguish efficiently, then R
distinguish which is a contradiction to the secur
of PRG.

y adversarial indistinguishability we have
$OneSecR^0 \approx OneSecR$ and therefore
$OneSec^0 \approx OneSecR^0 \approx OneSecR \quad \approx OneSec$

03/09/202

Stretching PRGs

Let $G : (\{0,1\}^n \to \{0,1\}^{n+1})_{n\in}$ be a PRG, i.e.

Then there exists a PRG G : $(\{0,1\}^n \to \{0,1\}$

polynomial $\ell(n) > n$.

Let G be a PRG with a 1 bit stretch, i.e. G outp

We need to iterate G to get an extra bit of pseu

at every step:

$$x_0 \to \quad x_1(b_1 \text{ gets added }) \to \quad \cdots \to \quad x_k(b_k$$

where $x_i \in \{0,1\}^n$, $b_i \in \{0,1\}$ are defined as x_0

and $G(x_i) = (x_{i+1}||b_{i+1})$ for $i < k$ and set

$$G\ (x_0) = (x_k||b_1|| \quad ||b_k) \in \{0,1\}^{n+}$$

Here $\ell(n) = n + k$.

03/09/202

Stretching PRGs

To prove that G is a PRG. Note that the definit
gives guarantees w.r.t. uniform inputs.

We define the following distributions: $H_i = (x_k||$
where

$b_1, \quad , b_i \in \{0, 1\}$

$x_i \leftarrow \{0, 1\}^n \rightarrow \quad x_{i+1}^{b_{i+1}}$ gets

$\cdots \rightarrow \quad x_k{}^{b_k}$gets dded

Claim $H_i \approx H_{i+1}, \forall i < k$.

Similar reduction argument as before since we ca
distinguisher that would imply a distinguisher

Stretching PRGs - Proof of Securit

distinguisher between H_i and H_{i+1} for any i
distinguisher R for the PRG G with the same suc
probability.

More precisely, R receives some sample
$y = (x_{i+1}||b_{i+1}) \in \{0,1\}^{n+1}$ and computes k
iterations of G with input x_{i+1}.

For the first i bits, R picks $b_1,$ $, b_i$ uniformly it
sends $(x_k||b_1||\cdots||b_k)$ to and forwards the ou

Such a reduction R is efficient as G is efficient to

Stretching PRGs - Proof of Securit

If y comes from $G(U_n)$ then receives samples
H_i.

If y comes from U_{n+1} then receives samples th
H_{i+1}.

Note that H_0 corresponds to $G(U_n)$ and H_k corre
$U\quad_{n)}$, a hybrid argument concludes the proof for

One has to be careful if $k(n)$ is polynomial in n f

03/09/202

Examples or not of PRGs

Suppose $G(s)$ is a secure PRG that outputs bit-s
$\{0,1\}^n$. In each of the following cases, say wheth
necessarily a psuedorandom generator. If yes, giv
not then show a counterexample.

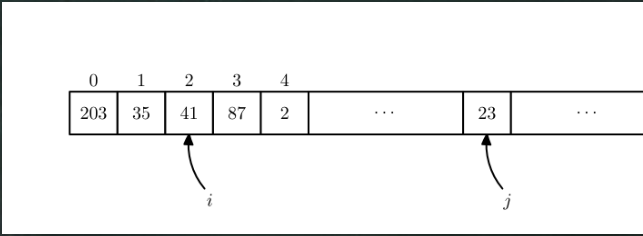
1. $G\ (s) := G(s\ ) \wedge G(s_2)$
2. $G\ (s) := G(s) \oplus 1^n$

ryptanalysis of R 4 stream ciphe

- RC4 stream cipher designed by Ron Rivest in 198
  used for securing Web traffic in the SSL/TLS pro
- It is designed to operate on 8-bit processors with
  memory.
- While RC4 is still in use, it has been shown to be
  a number of significant attacks and should not b
  projects.
- The heart of RC4 is the RC4 PRG.
  - RC4 PRG maintains an internal state consisting
    256 bytes plus two additional bytes i, j used as

R 4 stream cipher

S contains all the numbers 0, ,255 and each numb
exactly once.



R 4 stream cipher - Initializing S

RC4 stream cipher key s is a seed for the PRG and is
initialize the array S to a pseudo-random permutation
numbers $0, \quad , 255$ by using the setup algorithm.

```
input string of bytes $s$
for $i \leftarrow 0$ to 255 do: $S[i] \leftarrow i$
$j \leftarrow 0$
for $i \leftarrow 0$ to 255 do
    $k \leftarrow s[i \bmod |s|]$ (Extracting one byte from the s
    $j \leftarrow (j + S[i] + k) \bmod 256$
    Swap$(S[i], S[j])$
```

During the loop, i runs linearly through the array
around. You are swapping at each iteration the e
i with the entry at index j.

R 4 stream cipher - Stream Gener

The PRG generates a pseudo random output one byte
using the following stream generator:

```
$i, j \leftarrow 0$
repeat
    $i \leftarrow (i + 1) \bmod 256$
    $j \leftarrow (j + S[i]) \bmod 256$
    swap $(S[i], S[j])$
    Outputs $S[(S[i] + S[j]) \bmod 256]$
```

Repeats the above till needed. gain the index i
the array and j runs around and swapping $S[i]$ ar
continuously shuffles the array S.

Security of R 4

Let n be the size of the array S, $n = 256$ for RC4

RC4 setup algorithm initializes the array S to a p
0 255 generated from the given random seed.

Even if we assume the setup algorithm is perfect
a uniform permutation from the set of all 256 pe
Mantin and Shamir showed that the output of R

Suppose the array S is set to a random permutat
0 n 1 and that i, j are set to 0. Then the pr
the second byte of the output of RC4 is equal to

The lemma shows that the probability that the se
the output of RC4 is 0 is twice what it should be

Security of R 4

simple distinguisher for the RC4 PRG can be c

Given a string $x \in \{0, \quad , 255\}$, for $\ell \geq 2$, the c outputs 0 if the second byte of x is 0 and output

The lemma says the distinguisher has an advanta (approx) which is 0 39 for RC4.

The second byte bias was generalized to all bytes et al. though the bias is not as noticeable as in t byte.

They show, for example, that given the encryptio plaintext encrypted under 2^{30} random keys, it is recover the first 128 bytes of the plaintext with p close to 1.

Not as impossible as we think! In response, RS recommendation – discard the first 1024 bytes, t this attack, but not other attacks.