Introduction to Cryptology

3.1 - Pseudorandom Generators

Federico Pintore

Mathematical Institute, University of Oxford (UK)

UNIVERSITY OF OXFORD

Michaelmas term 2020

Current State of Affairs

- strong security definition for symmetric-key encryption schemes: perfect secrecy

Current State of Affairs

- strong security definition for symmetric-key encryption schemes: perfect secrecy

- the One Time Pad scheme is perfectly secret, but it requires keys of the same size of messages

Current State of Affairs

- strong security definition for symmetric-key encryption schemes: perfect secrecy

- the One Time Pad scheme is perfectly secret, but it requires keys of the same size of messages

- same issue with all perfectly-secret encryption schemes

Current State of Affairs

- strong security definition for symmetric-key encryption schemes: perfect secrecy
- the One Time Pad scheme is perfectly secret, but it requires keys of the same size of messages
- same issue with all perfectly-secret encryption schemes
- relaxed security definition: computational indistinguishability (or secrecy)

Current State of Affairs

- strong security definition for symmetric-key encryption schemes: perfect secrecy
- the One Time Pad scheme is perfectly secret, but it requires keys of the same size of messages
- same issue with all perfectly-secret encryption schemes
- relaxed security definition: computational indistinguishability (or secrecy)

If the uniformly random key in the OTP scheme is replaced by a *random-looking* key, is computational secrecy achieved?

Pseudorandom Generators (PRGs)

PRGs are used to efficiently produce, from short uniform bit strings, longer bit strings that appear uniform.

A PRG determines a distribution X on bit strings.

Pseudorandomness: sampling from X should be indistinguishable from sampling from the uniform distribution.

Pseudorandom Generators (PRGs)

Definition

Let $\ell(n) \in \mathbb{Z}[n]$ *be a polynomial s.t.* $\ell(n) > n$ *for every* n. *Consider a deterministic polynomial-time algorithm* G *s.t., for any* $n \in \mathbb{N}$ *and* $s \in \{0,1\}^n$*, the output* $G(s)$ *belongs to* $\{0,1\}^{\ell(n)}$.

G *is a pseudorandom generator if, for every PPT statistical test (or distinguisher)* D*, there is a negligible function* negl *s.t.*

$$\mathrm{Adv}^{\mathrm{PRG}}_{G,\mathrm{D}}(n) = |\Pr(\mathrm{D}(r) = 1) - \Pr(\mathrm{D}(G(s)) = 1)| \leq \mathrm{negl}(n)$$

where the probabilities are taken over uniform choice of $r \in \{0,1\}^{\ell(n)}$, $s \in \{0,1\}^n$ *and the randomness used by* D.

Pseudorandom Generators (PRGs)

Definition

Let $\ell(n) \in \mathbb{Z}[n]$ be a polynomial s.t. $\ell(n) > n$ for every n. Consider a deterministic polynomial-time algorithm G s.t., for any $n \in \mathbb{N}$ and $s \in \{0,1\}^n$, the output $G(s)$ belongs to $\{0,1\}^{\ell(n)}$.

G is a pseudorandom generator if, for every PPT statistical test (or distinguisher) D, there is a negligible function negl s.t.

$$\mathrm{Adv}_{G,\mathrm{D}}^{\mathrm{PRG}}(n) = |\Pr(\mathrm{D}(r) = 1) - \Pr(\mathrm{D}(G(s)) = 1)| \leq \mathrm{negl}(n)$$

where the probabilities are taken over uniform choice of $r \in \{0,1\}^{\ell(n)}$, $s \in \{0,1\}^n$ and the randomness used by D.

- D outputs either 1 or 0
- $\ell(n)$ is called expansion factor of G

Pseudorandom Generators (PRGs)

...where the probabilities are taken over uniform choice of $r \in \{0,1\}^{\ell(n)}$, $s \in \{0,1\}^n$ *and the randomness used by* D.

- Rand_n: set of all possible randomness used by D on input an $\ell(n)$-bit string.
- the uniform distributions over $\{0,1\}^{\ell(n)}$ and Rand_n induce a distribution over the event space $\mathcal{E} = \mathcal{P}(\{0,1\})$, where

$$\Pr(D(r) = 1) = \sum_{r,\mathrm{rand}} \frac{1}{2^{\ell(n)}} \frac{1}{|\mathrm{Rand}_n|} D(r, \mathrm{rand})$$

$$\Pr(D(r) = 0) = \sum_{r,\mathrm{rand}} \frac{1}{2^{\ell(n)}} \frac{1}{|\mathrm{Rand}_n|} (1 - D(r, \mathrm{rand}))$$

Fixed-length Encryption Scheme using a PRG

Let G be a PRG with expansion factor $\ell(n)$. Define an encryption scheme

$$E = (\mathrm{KeyGen}, \mathrm{Enc}, \mathrm{Dec})$$

with $\mathcal{M} = \{0,1\}^{\ell(n)}$, as follows:

- $k \leftarrow \mathrm{KeyGen}(n)$: it uniformly samples $k \in \{0,1\}^n$.
- $c \leftarrow \mathrm{Enc}(k,m)$: on input a key $k \in \{0,1\}^n$ and a message $m \in \{0,1\}^{\ell(n)}$, it outputs $c = G(k) \oplus m$.
- $m \leftarrow \mathrm{Dec}(k,c)$: on input a key $k \in \{0,1\}^n$ and a ciphertext $c \in \{0,1\}^{\ell(n)}$, it outputs $m = G(k) \oplus c$.

Computational indistinguishability of E

Theorem

If G is a PRG, then the encryption scheme E derived from G is computationally indistinguishable.

Computational indistinguishability of E

Theorem

If G is a PRG, then the encryption scheme E derived from G is computationally indistinguishable.

The proof is by reduction.

The reduction turns an adversary $\mathcal{A}$ against the computational indistinguishability of E into a distinguisher D for G.

The steps of the proof will be similar also for the other proofs by reduction we will encounter.

Computational indistinguishability of E

Proof.

Let $\mathcal{A}$ be a PPT adversary in $\mathrm{PrivK}^{\mathrm{eav}}_{\mathcal{A},E}$ (the Adversarial Indistinguishability Experiment).

$\mathcal{A}$ is exploited as a subroutine to construct a distinguisher D, defined as follows:

- D receives a bit string $w \in \{0,1\}^{\ell(n)}$;
- D runs $\mathcal{A}$, and obtains two messages $m_0, m_1 \in \{0,1\}^{\ell(n)}$;
- D samples a uniformly random bit $b \in \{0,1\}$, and sends $c = w \oplus m_b$ to $\mathcal{A}$;
- upon reception of b' from $\mathcal{A}$, D outputs 1 if $b = b'$, 0 otherwise.

Computational indistinguishability of E

We have:

$$
\begin{aligned}
&|\Pr(\mathrm{D}(G(s)) = 1) - \Pr(\mathrm{D}(r) = 1)| = \\
&|\Pr(\Pr(\mathrm{PrivK}^{\mathrm{eav}}_{\mathcal{A},E} = 1) - \mathrm{PrivK}^{\mathrm{eav}}_{\mathcal{A},\mathrm{OTP}} = 1)| = \\
&|\Pr(\mathrm{PrivK}^{\mathrm{eav}}_{\mathcal{A},E} = 1) - 1/2| \leq \mathrm{negl}(n)
\end{aligned}
$$

Therefore E is computationally indistinguishable. $\square$

Do PRGs exist?

Pseudorandom Generators (PRGs)

Do PRGs exist?

- not known how to unconditionally prove their existence;

Pseudorandom Generators (PRGs)

Do PRGs exist?

- not known how to unconditionally prove their existence;
- their existence can be proven under the assumption that one-way functions exist;

Pseudorandom Generators (PRGs)

Do PRGs exist?

- not known how to unconditionally prove their existence;
- their existence can be proven under the assumption that one-way functions exist;
- informally, a function is one-way if it is easy to compute but hard to invert;

Pseudorandom Generators (PRGs)

Do PRGs exist?

- not known how to unconditionally prove their existence;
- their existence can be proven under the assumption that one-way functions exist;
- informally, a function is one-way if it is easy to compute but hard to invert;
- the existence of one-way functions implies NP$\neq$ P.

Pseudorandom Generators (PRGs)

What are the PRGs used in cryptographic schemes?

- the algorithms G proven to PRGs (from the existence one-way functions) are not efficient;

Pseudorandom Generators (PRGs)

What are the PRGs used in cryptographic schemes?

- the algorithms G proven to PRGs (from the existence one-way functions) are not efficient;
- in practice, candidate PRGs are used, i.e. no *successfull* distinguishers are known;

Pseudorandom Generators (PRGs)

What are the PRGs used in cryptographic schemes?

- the algorithms G proven to PRGs (from the existence one-way functions) are not efficient;
- in practice, candidate PRGs are used, i.e. no *successfull* distinguishers are known;
- with an abuse of terminology, they are equally called PRGs;

Pseudorandom Generators (PRGs)

What are the PRGs used in cryptographic schemes?

- the algorithms G proven to PRGs (from the existence one-way functions) are not efficient;
- in practice, candidate PRGs are used, i.e. no *successfull* distinguishers are known;
- with an abuse of terminology, they are equally called PRGs;
- practical constructions use stream ciphers.

Further Reading I

Nadhem J AlFardan, Daniel J Bernstein, Kenneth G Paterson, Bertram Poettering, and Jacob CN Schuldt.
On the security of RC4 in TLS.
In 22nd USENIX Security Symposium (USENIX Security 13), pages 305–320, 2013.

Boaz Barak and Shai Halevi.
A model and architecture for pseudo-random generation with applications to/dev/random.
In Proceedings of the 12th ACM conference on Computer and communications security, pages 203–212. ACM, 2005.

Daniel J Bernstein.
The Salsa20 Family of Stream Ciphers.
In New stream cipher designs, pages 84–97. Springer, 2008.

Further Reading II

- Lenore Blum, Manuel Blum, and Mike Shub.
  A simple unpredictable pseudo-random number generator.
  SIAM Journal on computing, 15(2):364–383, 1986.
- Christian Cachin.
  Entropy measures and unconditional security in cryptography.
  PhD thesis, ETH Zurich, 1997.
- Scott Fluhrer, Itsik Mantin, and Adi Shamir.
  Weaknesses in the key scheduling algorithm of RC4.
  In Selected areas in cryptography, pages 1–24. Springer, 2001.

Further Reading III

Christina Garman, Kenneth G Paterson, and Thyla Van der Merwe.
Attacks only get better: Password recovery attacks against RC4 in TLS.
In 24th USENIX Security Symposium (USENIX Security 15), pages 113–128, 2015.

Itsik Mantin and Adi Shamir.
A practical attack on broadcast RC4.
In Fast Software Encryption, pages 152–164. Springer, 2002.