Recap: perfect secrecy

Informal description

An encryption method is perfectly secret if it is unbreakable

- no matter how much time you have
- no matter how powerful resources you have

One-time pad

- message and key: equal length
- throw away key after use
- the key is unknown to the attacker
- **key: true random bits**

Random sequences

What to expect from a random sequence?

- distribution of 0 and 1 (or other alphabet): uniform
- inability to predict future values based on the past/present
- no correlation between characters at different positions

Question

How do we define the randomness of a seqeunce?

Do random sequences actually exist?

- s_1=111111111111111111111111111111111111111111
- s_2=111111111111111111111000000000000000000000
- s_3=101010101010101010101010101010101010101010
- s_4=100011101011000011101010101110001110111110

Do random sequences actually exist?

- s_1=111111111111111111111111111111111111111111
- s_2=111111111111111111111000000000000000000000
- s_3=101010101010101010101010101010101010101010
- s_4=100011101011000011101010101110001110111110

Do random sequences actually exist?

- s_1=111111111111111111111111111111111111111111
- s_2=111111111111111111111000000000000000000000
- s_3=101010101010101010101010101010101010101010
- s_4=100011101011000011101010101110001110111110

Do random sequences actually exist?

- s_1=111111111111111111111111111111111111111111
- s_2=111111111111111111111000000000000000000000
- s_3=101010101010101010101010101010101010101010
- s_4=100011101011000011101010101110001110111110

Random sequence

Definition

Denote by $d(s)$ the minimal description of sequence s using some universal language $\mathcal{L}$. The length of this description is called the Kolmogorov complexity of s: $K(s) = |d(s)|$.

Definition

Sequence s is ***algorithmically random*** *if $|s| < K(s)$.*

Interpretation

Think of Kolmogorov complexity as the length of the shortest (python/C++/whatever) program that generates s as output. Randomness: no „easy” description/compression/program available.

Do random sequences actually exist?

- s_1=111111111111111111111111111111111111111111

  Short description of s_1: e.g. $[1] * 42$

- s_4=100011101011000011101010101110001110111110

  s_4 Looks random, *looks* uncompressible.

Kolmogorov complexity: the caveat

Theorem

No program exists that would compute $K(s)$ for any input s.

Proof by contradiction (sketch) Suppose a program KolmogorovComplexity(s) does the job. Suppose it has length 100000. Now, what's the shortest program that generates the output of the following algorithm?

```
for i = 1 → ∞ do
    for each sequence s with length ∀i do
        if KolmogorovComplexity(s) ≥ 200000 then
            return s
        end if
    end for
end for
```

Generating random sequences

Quote

„Anyone who attempts to generate random numbers by deterministic means is, of course, living in a state of sin." – John von Neumann

Corollary of impossibility theorem

No mathematical algorithm to be expected that generates provably random sequences.

Properties of randomness needed

- physical source
- poor randomness implies software vulnerailities

Computational security

Kerckhoffs's principles

1. Make the cryptosystem secure in *practice* if not mathematically.
2. ...

Idea

Relax the conditions on perfect secrecy. No breaking of cipher

- in „reasonable” time
- with „reasonable” probability of success

Computational secrecy

Reasonable time = ???

- efficient adversary
- efficient algorithms

Definition

*An algorithm A has **polynomial running time** if $\exists\, p(.)$, a polynomial s.t. $\forall x \in \{0,1\}^*$ computation of $A(x)$ terminates in $\leq p(|x|)$ steps.*

Definition

*An algorithm A is **probabilistic** if it has access to a source generating uniformly random and independent bits.*

Efficient adversary

Using only probabilistic polynomial time (**PPT**) algorithms

Computational secrecy

Reasonable time = ???

- efficient adversary
- efficient algorithms

Definition

An algorithm A has ***polynomial running time*** *if $\exists\ p(.)$, a polynomial s.t. $\forall x \in \{0,1\}^*$ computation of $A(x)$ terminates in $\leq p(|x|)$ steps.*

Definition

An algorithm A is ***probabilistic*** *if it has access to a source generating uniformly random and independent bits.*

Efficient adversary

Using only probabilistic polynomial time (**PPT**) algorithms

Computational secrecy

Reasonable success

- very small probability
- negligible probability

Definition

*Function f is **negligible** if $\forall p(.)$, a polynomial $\exists N \in \mathbb{R}^+$ with* $\forall n \in \mathbb{N}, n > N : f(n) < \frac{1}{p(n)}$.

Reasonable probability of success

Negligible function as the probability of successful break.

Computational secrecy: two approaches

Concrete approach

A scheme is (t, ε)-secure if $\forall$ adversaries with at most t time, the prob. of success is at most ε.

Asymptotic approach

A scheme is secure if $\forall$ PPT adversaries have only negligible probability of successfully breaking the scheme.

Security proofs by reduction

- Unconditional security: has its limits
- we need some assumption for computational security
- a „basic” problem being hard
- based on that, we can argument for the difficulty of breaking the scheme

Reduction pattern

- If we suppose a PPT adversary: $\exists \mathcal{A}$ breaks the scheme with non-negl. prob.
- then there’s an algorithm $\exists \mathcal{A}'$ solving the (by assumption) hard problem

(Computationally) secure encryption scheme

Definition

An encryption scheme is a triple $\Pi = (Gen, Enc, Dec)$ *where :*

- Gen *is key generation, a probabilistic algorithm that returns a key* $k \in_R \mathcal{K}$ *(maybe using an input called the security parameter)*
- Enc *is encryption, a probabilistic algorithm that returns a ciphertext* $c \in \mathcal{C}$ *on inputs* $k \in \mathcal{K}$ *and* $m \in \mathcal{M}$*, i.e.* $c := Enc_k(m)$*.*
- Dec *is decryption a deterministic algorithm that returns a plaintext upon inputs* k *and* $c \in \mathcal{C}$*: the return value is* $Dec_k(c)\ in \mathcal{M}$*.*

Threat model

Passive attacker: eavesdropping, has acces to a single encrypted text.

(Computationally) secure encryption scheme

Attack

- $\mathcal{A}$: eavesdropping
- a single instance of an encrypted message
- passive attack
- goal: $\mathcal{A}$ learns nothing about the plaintext m
  - *semantic security*
  - hard to handle
- instead: **indistinguishability**

(Computationally) secure encryption scheme

Definition (Indistinguishablity experiment with eavesdropper $PrivK^{eav}_{\mathcal{A},\Pi}(n)$)

1. *Adversary $\mathcal{A}$ returns two messages m_0, m_1 with $|m_0| = |m_1|$ upon input 1^n.*
2. $k = Gen(1^n), b \in_R \{0,1\} : c = Enc_k(m_b)$*. The ciphertext c is sent to $\mathcal{A}$*
3. *$\mathcal{A}$ outputs $b' \in \{0,1\}$*
4. $PrivK^{eav}_{\mathcal{A},\Pi}(n) = 1$*, if $b = b'$, otherwise 0.*

Definition

The scheme $\Pi = (Gen, Enc, Dec)$ has the indistinguishability property aginst one eavesdropping if all PPT adversaries $\mathcal{A}$, a negligible function $e(.)$ exists for which

$$P(PrivK^{eav}_{\mathcal{A},\Pi}(n) = 1) \leq \frac{1}{2} + e(n).$$

Pseudorandomness

Idea

- one-time pad idea
- replace perfect security by computational security
- replace random key by ???

Pseudorandom sequence

- PR for short
- relaxed, computational version of randomness
- *looks* random to a PPT observer
- generated from a short truly random sequence (seed)

Pseudorandomness

Idea

- one-time pad idea
- replace perfect security by computational security
- replace random key by ???

Pseudorandom sequence

- PR for short
- relaxed, computational version of randomness
- *looks* random to a PPT observer
- generated from a short truly random sequence (seed)

Pseudorandomness

Intuition

- has some physical randomness in it, but the sequence is much longer
- in reasonable time: indistinguishable from true randomness

Definition

Let $l(.)$ be a polynomial (called expansion factor) and $G : \{0,1\}^n \to \{0,1\}^{l(n)}$ a DPT (deterministic PT) algorithm. Then G is a pseudorandom generator if

1. $\forall n : l(n) > n$
2. $\forall D$ *PPT distinguisher,* $\exists e(.)$, *a negligible function with* $\forall s \in_R \{0,1\}^n, \forall r \in_R \{0,1\}^{l(n)}$*:*

$$|Pr(D(r) = 1) - Pr(D(G(s)) = 1)| \leq e(n)$$

Pseudorandomness

Intuition

- has some physical randomness in it, but the sequence is much longer
- in reasonable time: indistinguishable from true randomness

Definition

Let $l(.)$ be a polynomial (called expansion factor) and $G : \{0,1\}^n \to \{0,1\}^{l(n)}$ a DPT (deterministic PT) algorithm. Then G is a pseudorandom generator if

1. $\forall n : l(n) > n$
2. $\forall D$ *PPT distinguisher,* $\exists e(.)$*, a negligible function with* $\forall s \in_R \{0,1\}^n, \forall r \in_R \{0,1\}^{l(n)}$*:*

$$|Pr(D(r) = 1) - Pr(D(G(s)) = 1)| \leq e(n)$$

Pseudorandomness

Properties

- PRG no „real” randomness
- Brute force always works in principle
- Seed:
  - true randomness
  - secret
  - not TOO short
- PRG exists, assuming . . . is hard
- statistical tests

Example: next-bit test

Next-bit test

A sequnece passes the next-bit test if forall positions i, the attacker

- knowing the first i bits
- can NOT guess the bit at position $(i+1)$ with probability higher than $50\% + \epsilon$.

Statistical tests

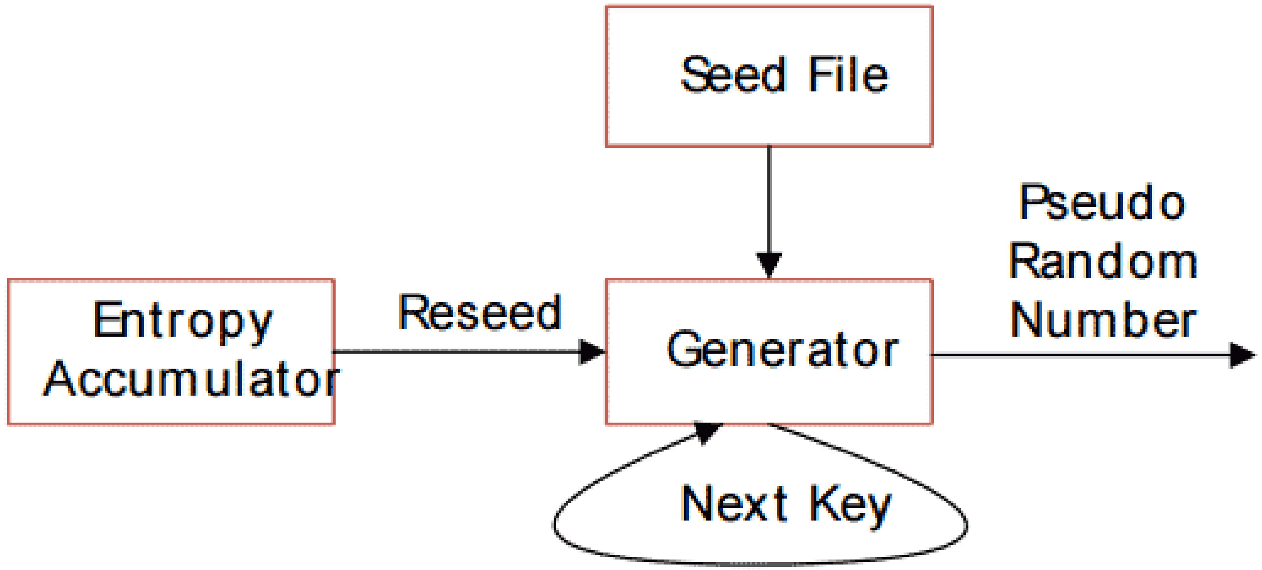
- NIST test (National standards intitute of US)
- DIEHARD test (academic origins – Marsaglia '95)

PRG example: Fortuna

Fortuna

- Schneier, Fergusson 2003
- PRG family with 3 main components:

1. **Generator**: generate PR stream after seeding
2. **Entropy accumulator**: collect randomness
3. **Seeding**: ensure randomness in a bootstrapping phase



A secure scheme (against 1 eavsedropping)

Scheme using PRG

Let G be a PRG with expansion factor l, and

- **Gen** $k \in_R \{0,1\}^n$
- **Enc** For $k \in \{0,1\}^n$ and $m \in \{0,1\}^{l(n)}$, let $c = Enc_k(m) = G(k) \oplus m$.
- **Dec** For k, $c \in \{0,1\}^{l(n)}$, let $Dec_k(c) = c \oplus G(k)$.

Theorem

If G has the PRG properties, the $\Pi = (Gen, Enc, Dec)$ is secure in the presence of an eavesdropper (with one intercepted message).

A secure scheme (against multiple eavesdropping)

Definition (Indistinguishability experiment $PrivK^{meav}_{\mathcal{A},\Pi}(n)$)

Same as above, except:

1. *Adversary $\mathcal{A}$ issues a sequence* $M_0 = (m_{01}, \ldots, m_{0t}), M_1 = (m_{11}, \ldots, m_{1t})$ *with* $\forall i : |m_{0i}| = |m_{1i}|$
2. $k = Gen(1^n), b \in_R \{0, 1\} : C = (c_1, \ldots, c_t) : c_i = Enc_k(m_{bi})$ *received by $\mathcal{A}$*

- $\exists$ scheme secure for exactly 1 eavesdropping attempt
- deterministic algo never good $\Rightarrow$ randomization needed (IV)
- synchronization issues
- $Enc_k(m) = (IV, G(k, IV) \oplus m)$

A secure scheme (against multiple eavesdropping)

Definition (Indistinguishability experiment $PrivK^{meav}_{\mathcal{A},\Pi}(n)$)

Same as above, except:

1. *Adversary $\mathcal{A}$ issues a sequence* $M_0 = (m_{01}, \ldots, m_{0t}), M_1 = (m_{11}, \ldots, m_{1t})$ *with* $\forall i : |m_{0i}| = |m_{1i}|$
2. $k = Gen(1^n), b \in_R \{0, 1\} : C = (c_1, \ldots, c_t) : c_i = Enc_k(m_{bi})$ *received by $\mathcal{A}$*

- $\exists$ scheme secure for exactly 1 eavesdropping attempt
- deterministic algo never good $\Rightarrow$ randomization needed (IV)
- synchronization issues
- $Enc_k(m) = (IV, G(k, IV) \oplus m)$

Chosen plaintext attack

Attack

Active adversary: has access to arbitraty pairs (c, m)

Definition (CPA indistinguishability experiment $PrivK^{cpa}_{\mathcal{A},\Pi}(n)$)

1. $k = Gen(1^n)$
2. *Adversary $\mathcal{A}$ has oracle (black box) access to $Enc_k(.)$-hez, issues two plaintexts m_0, m_1 with $|m_0| = |m_1|$*
3. $b \in_R \{0, 1\} : c = Enc_k(m_b)$ *given to $\mathcal{A}$*
4. *$\mathcal{A}$ has further oracle access $Enc_k(.)$, outputs $b' \in \{0, 1\}$*
5. $PrivK^{cpa}_{\mathcal{A},\Pi}(n) = 1$, *if $b = b'$, 0 otherwise.*

Chosen plaintext attack

Definition

A scheme $\Pi = (Gen, Enc, Dec)$ *is CPA-secure if for any PPT adversary* $\mathcal{A}$ $\exists e(.)$, *a negligible function with*

$$P(PrivK^{cpa}_{\mathcal{A},\Pi}(n) = 1) \leq \frac{1}{2} + e(n).$$

- Cannot be deterministic
- Secure against one eavesdropping $\Rightarrow$ secure against multiple eavesdropping
- Can be fulfilled by PRG

Wrap-up

Summary

- Sequence is random if hard to describe/compress
- Relax perfect secrecy: computational security
- PPT adversary + negligible success probability
- Pseudorandom sequence: computational difficulty formulation