Private-Key Encryption and Pseudorandomness (Part I)

Yu Zhang

Harbin Institute of Technology

Cryptography, Autumn, 2022

Outline

Content

Idea of Computational Security

Computational security vs. Information-theoretical security

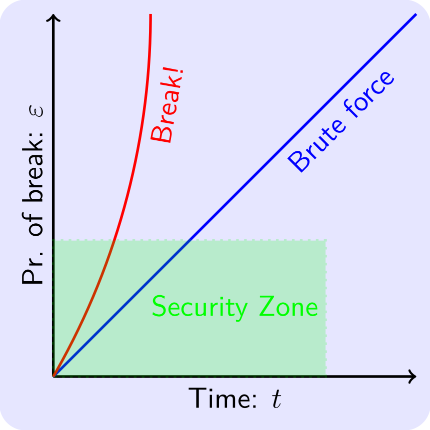
Kerckhoffs's Another Principle

A [cipher] must be practically, if not mathematically, indecipherable.

- Information-theoretical security: Perfect secrecy.
  Q: what's the limitation of perfect secrecy?
- Computational security:
  - Only preserved against adversaries that run in a **feasible amount of time**.
  - Adversaries can succeed with some **very small probability**.

Necessity of the Relaxations

Limit the power of adversary (against brute force with pr. 1 in time linear in $|\mathcal{K}|$) and allow a negligible probability (against random guess with pr. $1/|\mathcal{K}|$).



Concrete Approach and Asymptotics

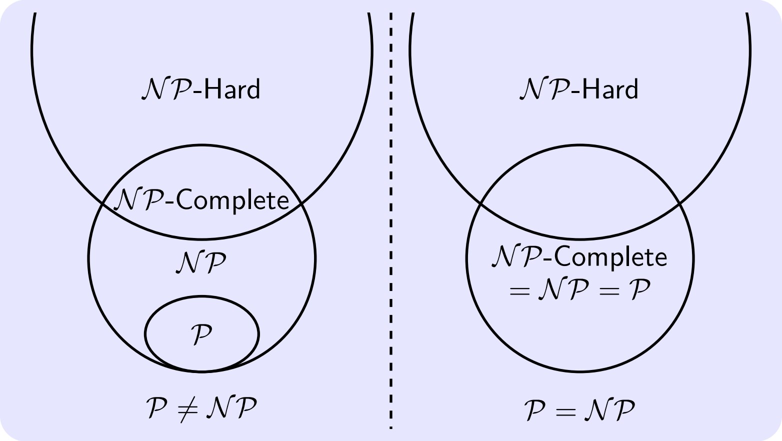
Concrete Approach: A scheme is (t, ε)-**secure** if every adversary running for time at most t succeeds in breaking the scheme with probability at most ε.

Example

Optimal security: when the key has length n, an adversary running in time t can succeed with probability at most $t/2^n$.

$t = 2^{80}$	2^{20} 1GHz CPUs run 35 years.
$n = 128$	2^{170} atoms in the planet.
$\varepsilon = 2^{-48}$	once every 100 years with probability 2^{-30}/sec.

Asymptotics: A method of describing limiting behavior. Given the input size of a problem, n, the time complexity is $f(n)$.
For example, the time complexity of quick sort for n numbers is $O(n \cdot \log n)$.



The majority of computer scientists believe $\mathcal{P} \neq \mathcal{NP}$.

This is very dangerous!

Efficient Computation

- An algorithm A runs in **polynomial time** if there exists a polynomial $p(\cdot)$ such that, for every input $x \in 0, 1^*$, $A(x)$ terminates within at most $p(|x|)$ steps.
  Q: is $n!$ polynomial? is $\log n$ polynomial?
- A can run another PPT A' as a sub-routine in polynomial-time.
  Q: $f(x) = x^2$, is $g(x) = \frac{x^3}{f(x)}$ polynomial?
- A **probabilistic** algorithm has the capability of "tossing coins". Random number generators should be designed for cryptographic use, not `random()` in C.
- Open question: Does probabilistic adversaries are more powerful than deterministic ones? $\mathcal{P} = \mathcal{BPP}$?

Negligible Success Probability

- A function f is **negligible** if for every polynomial $p(\cdot)$ there exists an N such that for all integers $n > N$ it holds that $f(n) < \frac{1}{p(n)}$.
- Q: is $\left(\frac{3}{n}\right)^9$ negligible? is $\frac{n^2}{2^n}$ negligible?
- Q: is $\mathsf{negl}_1(n) + \mathsf{negl}_2(n)$ negligible?
- Q: is $poly(n) \cdot \mathsf{negl}(n)$ negligible?

Asymptotic Approach

Problem X (breaking the scheme) is *hard* if X cannot be solved by any polynomial-time algorithm for time t except with negligible probability ε.

- t, ε are described as functions of **security parameter** n (usually, the length of key).
- Caution: 'Security' for large enough values of n.

Example

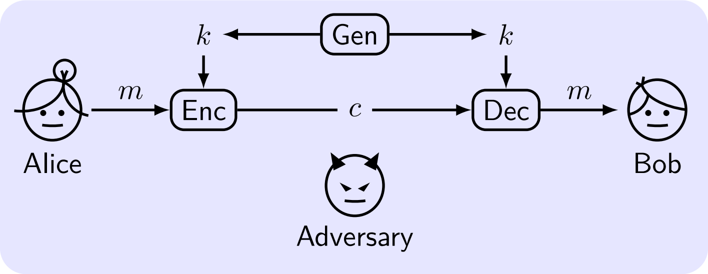
"Breaking the scheme" with probability $2^{40} \cdot 2^{-n}$ in n^3 minutes.

$n \leq 40$	6 weeks with probability 1.
$n = 50$	3 months with probability $1/1000$.
$n = 500$	more than 200 years with probability 2^{-500}.

Q: Under Moore's Law, who has more advantages? Adversary or Alice?

Content

Defining Private-key Encryption Scheme



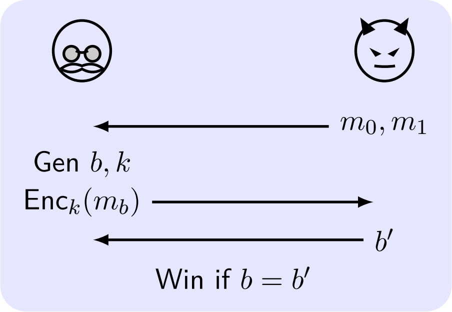
A **Private-key encryption scheme** Π is a tuple of $\textsc{ppt}$ $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$

- $k \leftarrow \mathsf{Gen}(1^n), |k| \geq n$ (security parameter).
  $\mathsf{Gen}(1^n)$ chooses $k \leftarrow \{0,1\}^n$ uniformly at random (***u.a.r***)
- $c \leftarrow \mathsf{Enc}_k(m), m \in \{0,1\}^*$ (all finite-length binary strings).
  Fixed-length if $m \in \{0,1\}^{\ell(n)}$
- $m := \mathsf{Dec}_k(c)$
- $\mathsf{Dec}_k(\mathsf{Enc}_k(m)) = m$

Eavesdropping Indistinguishability Experiment

The eavesdropping indistinguishability experiment $\mathsf{PrivK}^{\mathsf{eav}}_{\mathcal{A},\Pi}(n)$:

1. $\mathcal{A}$ is given input 1^n, outputs m_0, m_1 of the same length
2. $k \leftarrow \mathsf{Gen}(1^n)$, a random bit $b \leftarrow \{0,1\}$ is chosen. Then $c \leftarrow \mathsf{Enc}_k(m_b)$ (challenge ciphertext) is given to $\mathcal{A}$
3. $\mathcal{A}$ outputs b'. If $b' = b$, $\mathsf{PrivK}^{\mathsf{eav}}_{\mathcal{A},\Pi} = 1$, otherwise 0



Defining Private-key Encryption Security

Definition 1

Π has **indistinguishable encryptions in the presence of an eavesdropper** if $\forall$ PPT $\mathcal{A}$, $\exists$ a negligible function negl such that

$$\Pr\left[\mathsf{PrivK}^{\mathsf{eav}}_{\mathcal{A},\Pi}(n) = 1\right] \leq \frac{1}{2} + \mathsf{negl}(n),$$

where the probability it taken over the random coins used by $\mathcal{A}$.

Understanding Definition of Indistinguishability

If an adversary always fails in the experiments, is the scheme secure?

What's the probability of using the same key in two successive eavesdropping indistinguishability experiments?

If the lowest bit of message can be guessed from the ciphertext with probability $\frac{3}{4}$, is the scheme secure?

If the lowest 3 bits of message can be guessed from the ciphertext with probability $\frac{3}{8}$, is the scheme secure?

Correlation: If the distributions of (X and Z) and (Y and Z) are indistinguishable, are X and Y also indistinguishable?

Semantic Security

Intuition: No partial information leaks.

Definition 2

Π is **semantically secure in the presence of an eavesdropper** if $\forall$ PPT $\mathcal{A}$, $\exists \mathcal{A}'$ such that $\forall$ distribution $X = (X_1, \dots)$ and $\forall f, h$,

$$\left|\Pr[\mathcal{A}(1^n, \mathsf{Enc}_k(m), h(m)) = f(m)] - \Pr[\mathcal{A}'(1^n, h(m)) = f(m)]\right|$$

$$\leq \mathsf{negl}(n).$$

where m is chosen according to X_n, $h(m)$ is external information.

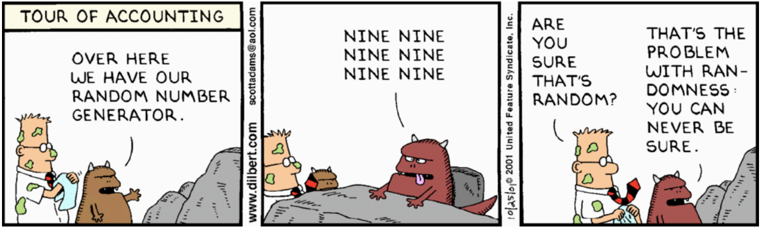
Theorem 3

*A private-key encryption scheme has **indistinguishable** encryptions in the presence of an eavesdropper $\iff$ it is **semantically secure** in the presence of an eavesdropper.*

Content

Conceptual Points of Pseudorandomness

- True randomness can not be generated by a describable mechanism
- Pseudorandom looks truly random for the observers who don't know the mechanism
- No fixed string can be "random" or "pseudorandom" which refers to the properties of the process used to generate a string
- Q: is it possible to definitively prove randomness?



Distinguisher: Statistical Tests

The pragmatic approach is to take many sequences of random numbers from a given generator and subject them to a battery of statistical tests.[1]

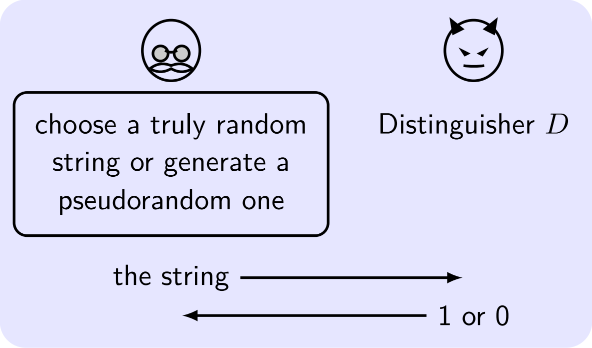
- $D(x) = 0$ if $|\#0(x) - \#1(x)| \leq 10 \cdot \sqrt{n}$
- $D(x) = 0$ if $|\#00(x) - n/4| \leq 10 \cdot \sqrt{n}$
- $D(x) = 0$ if max-run-of-$0(x) \leq 10 \cdot \log n$

Pseudorandomness means being **next-bit unpredictable**,
G passes all next bit tests $\iff$ G passes all statistical tests. How many tests shall we need?

[1] State-of-the-art: NIST Special Publication 800-22 "*A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications*"

Intuition for Defining Pseudorandom

Intuition: Generate a long string from a short truly random seed, and the pseudorandom string is indistinguishable from truly random strings.



Definition of Pseudorandom Generators

Definition 4

A deterministic polynomial-time algorithm $G : \{0,1\}^n \rightarrow \{0,1\}^{\ell(n)}$ is a **pseudorandom generator (PRG)** if

1. (Expansion:) $\forall n, \ell(n) > n$.
2. (Pseudorandomness): $\forall$ PPT distinguishers D,

$$|\Pr[D(r) = 1] - \Pr[D(G(s)) = 1]| \leq \mathsf{negl}(n),$$

where r is chosen *u.a.r* from $\{0,1\}^{\ell(n)}$, the **seed** s is chosen *u.a.r* from $\{0,1\}^n$. $\ell(\cdot)$ is the **expansion factor** of G.

- **Existence**: Under the weak assumption that *one-way functions* exists, or $\mathcal{P} \neq \mathcal{NP}$

Real World Cases

glibc random()

$r[i] = (r[i-3] + r[i-31])\%2^{32}$

Netscape (by reverse-engineering)

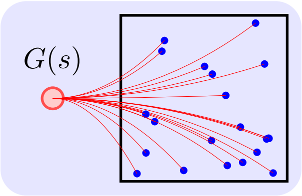
```
global variable seed;
RNG_CreateContext();
    (seconds, microseconds) = time of day;
    pid = process ID; ppid = parent process ID;
    a = mklcpr(microseconds);
    b = mklcpr(pid + seconds + (ppid << 12));
    seed = MD5(a, b);
RNG_GenerateRandomBytes()
    x = MD5(seed);
    seed = seed + 1;
    return x;
```

Problems On PRG

F is PRG. Is G PRG?

- $G(s)$ is such that $XOR(G(s)) = 1$
- $G(s) = F(0)$
- $G(s) = F(s) \| 0$
- $G(s) = F(s \oplus 1^{|s|})$
- $G(s) = F(s) \| F(s)$
- $G(s \| s') = F(s) \| F(s')$
- $G(s) = F(s \| 0)$
- $G : s \leftarrow \{0,1\}^{20}, G(s) = F(s)$

Sufficient Seed Space



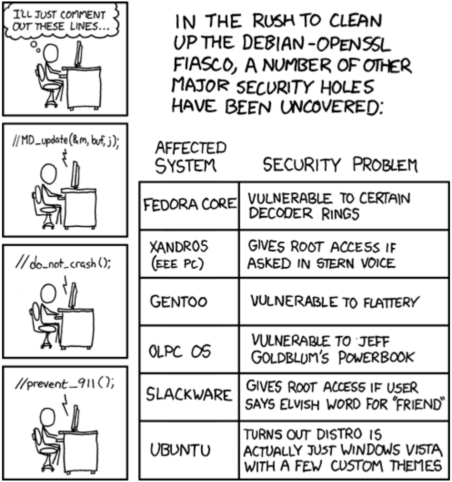
- **Sparse outputs**: In the case of $\ell(n) = 2n$, only 2^{-n} of strings of length $2n$ occurs.
- **Brute force attack**: Given an unlimited amount of time, one can distinguish $G(s)$ from r with a high probability by generating all strings with all seeds.

$$|\Pr[D(r) = 1] - \Pr[D(G(s)) = 1]| \geq 1 - 2^{-n}$$

- **Sufficient seed space**: s must be long enough against brute force attack.

Bad Randomness [xkcd:424]

In 2008, the Debian project announced that a vulnerability in the OpenSSL package. The bug was caused by the removal of the line of code from md_rand.c. (CVE-2008-0166)



Content

Reduction (Complexity)

A **reduction** is a transformation of one problem A into another problem B.

Reduction $A \leq_m B$ [2] : A is **reducible** to B if solutions to B exist and whenever given the solutions A can be solved.

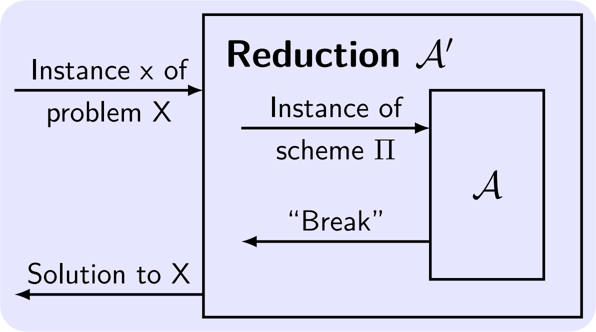
Solving A **cannot be harder** than solving B.

Example

- "measure the area of a rectangle" $\leq_m$ "measure the length and width of rectangle"
- "calculate x^2" $\leq_m$ "calculate $x \times y$"

[2] m means the mapping reduction.

Proofs of Reduction



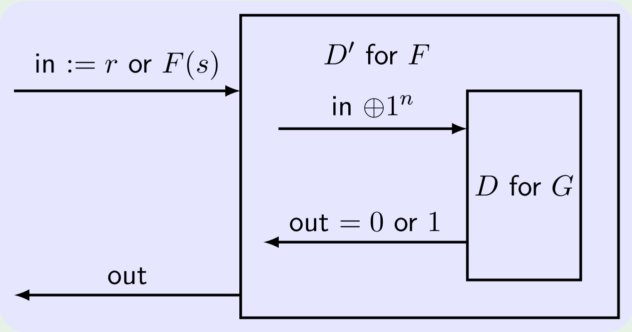
- A PPT $\mathcal{A}$ can break Π with probability $\varepsilon(n)$.
- **Assumption**: Problem X is *hard* to solve.
- **Reduction**: Reduce $\mathcal{A}'$ to $\mathcal{A}$. $\mathcal{A}'$ solves x efficiently with probability $1/p(n)$, running $\mathcal{A}$ as a sub-routine.
- **Contradiction**: If $\varepsilon(n)$ is non-negligible, then $\mathcal{A}'$ solves X efficiently with non-negligible probability $\varepsilon(n)/p(n)$.

An Example of Proof By Reduction

If $F(s)$ is PRG, so is $G(s) = F(s) \oplus 1^{|n|}$?

- Problem A (Assumption): to distinguish $F(s)$ from r
- Problem B (Break the scheme): to distinguish $G(s)$ from r

Idea: Reduce A to B. As $F(s)$ is distinguishable, so is $G(s)$.



An Example of Proof By Reduction (Cont.)

If $F(s)$ is PRG, so is $G(s) = F(s) \oplus 1^{|n|}$?

$$\Pr[D'(F(s)) = 1] = \Pr[D(G(s) = F(s) \oplus 1^n) = 1]$$

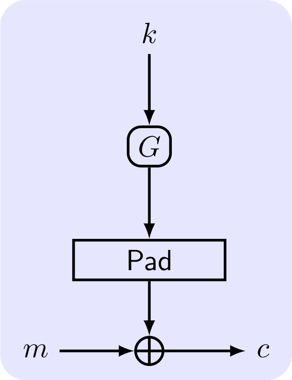
$$\Pr[D'(r) = 1] = \Pr[D(r \oplus 1^n) = 1] = \Pr[D(r) = 1]$$

$$\begin{aligned} \mathsf{negl} &\geq \Pr[D'(F(s)) = 1] - \Pr[D'(r) = 1] \\ &= \Pr[D(G(s)) = 1] - \Pr[D(r) = 1] \end{aligned}$$

According to the definition of PRG, $G(s)$ is a PRG.

Content

A Secure Fixed-Length Encryption Scheme



Construction 5

- $|G(k)| = \ell(|k|)$, $m \in \{0,1\}^{\ell(n)}$.
- Gen: $k \in \{0,1\}^n$.
- Enc: $c := G(k) \oplus m$.
- Dec: $m := G(k) \oplus c$.

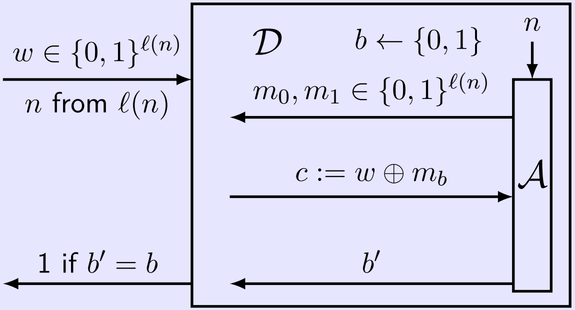
Theorem 6

This fixed-length encryption scheme has indistinguishable encryptions in the presence of an eavesdropper.

Proof of Indistinguishable Encryptions

Idea: Use $\mathcal{A}$ to construct D for G, so that D distinguishes G when $\mathcal{A}$ breaks $\tilde{\Pi}$. Since D cannot distinguish G, so that $\mathcal{A}$ cannot break $\tilde{\Pi}$.

Proof.



$$\Pr[D(w) = 1] = \Pr[\mathsf{PrivK}^{\mathsf{eav}}_{\mathcal{A},\tilde{\Pi}}(n) = 1]$$

□

Proof of Indistinguishable Encryptions (Cont.)

Proof.

To prove $\varepsilon(n) \stackrel{\text{def}}{=} \Pr[\mathsf{PrivK}^{\mathsf{eav}}_{\mathcal{A},\Pi}(n) = 1] - \frac{1}{2}$ is negligible.

(1) If w is r chosen *u.a.r*, then $\tilde{\Pi}$ is OTP.

$$\Pr[D(r) = 1] = \Pr[\mathsf{PrivK}^{\mathsf{eav}}_{\mathcal{A},\tilde{\Pi}}(n) = 1] = \frac{1}{2};$$

(2) If w is $G(k)$, then $\tilde{\Pi} = \Pi$.

$$\Pr[D(G(k)) = 1] = \Pr[\mathsf{PrivK}^{\mathsf{eav}}_{\mathcal{A},\Pi}(n) = 1] = \frac{1}{2} + \varepsilon(n).$$

Use Definition 4:

$$|\Pr[D(r) = 1] - \Pr[D(G(k)) = 1]| = \varepsilon(n) \le \mathsf{negl}(n).$$

$\square$

Handling Variable-Length Messages (homework)

Definition 7

A **deterministic** polynomial-time algorithm G is a **variable output-length pseudorandom generator** if

1. $G(s, 1^\ell)$ outputs a string of length $\ell > 0$, where s is a string.
2. $G(s, 1^\ell)$ is a prefix of $G(s, 1^{\ell'})$, $\ell' > \ell$.[3]
3. $G_\ell(s) \stackrel{\text{def}}{=} G(s, 1^{\ell(|s|)})$. Then $\forall \ell(\cdot)$, G_ℓ is a PRG with expansion factor ℓ.

Both Construction 5 and Theorem 6 hold here.

[3] for technical reasons to prove security.

Computational Security vs. Info.-theoretical Security

	Computational	**Info.-theoretical**
Adversary	PPT eavesdropping	no limited eavesdropping
Definition	indistinguishable $\frac{1}{2} + \mathsf{negl}$	indistinguishable $\frac{1}{2}$
Assumption	pseudorandom	random
Key	short random str.	long random str.
Construction	XOR pad	XOR pad
Prove	reduction	prob. theory