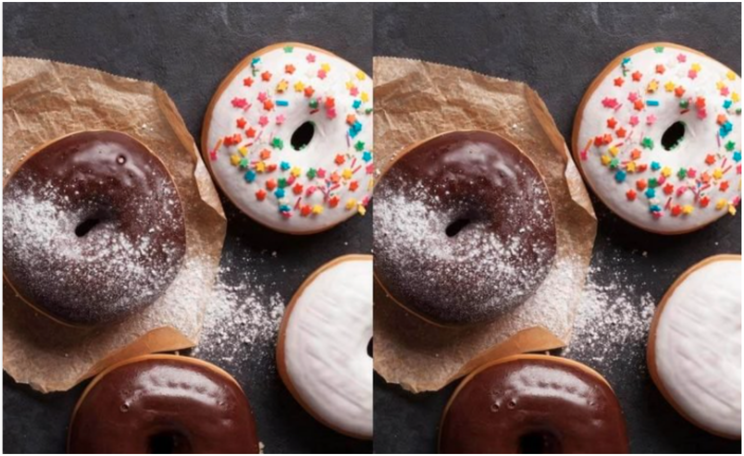
CS4236 Cryptography Theory and Practice Topic 3 - Private key topics: Key length and Computational Security

Hugh Anderson

National University of Singapore
School of Computing

August, 2022

Indistinguishability...

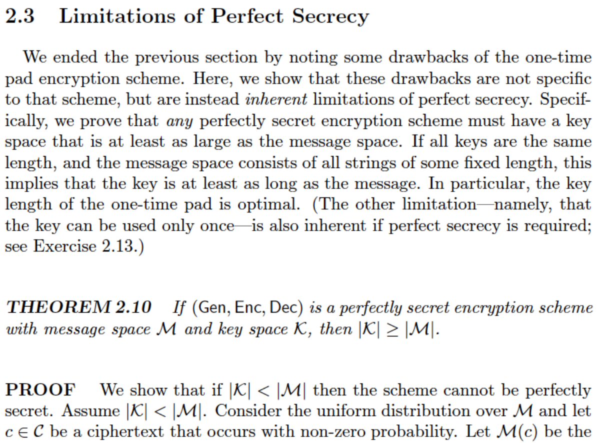


Spot the difference(s)...

Outline

1. **The “length” of the key in “perfect secrecy”**
   - JIT mathematics: “Information theory”
2. **Computational Indistinguishability**
   - Negligable functions
   - Formulating computational indistinguishability
   - Constructing computational secrecy with a PRG

Limitation - need Looooonnnnggggg keys...



2.3 Limitations of Perfect Secrecy

We ended the previous section by noting some drawbacks of the one-time pad encryption scheme. Here, we show that these drawbacks are not specific to that scheme, but are instead *inherent* limitations of perfect secrecy. Specifically, we prove that *any* perfectly secret encryption scheme must have a key space that is at least as large as the message space. If all keys are the same length, and the message space consists of all strings of some fixed length, this implies that the key is at least as long as the message. In particular, the key length of the one-time pad is optimal. (The other limitation—namely, that the key can be used only once—is also inherent if perfect secrecy is required; see Exercise 2.13.)

THEOREM 2.10 *If* (Gen, Enc, Dec) *is a perfectly secret encryption scheme with message space* $\mathcal{M}$ *and key space* $\mathcal{K}$*, then* $|\mathcal{K}| \geq |\mathcal{M}|$.

PROOF We show that if $|\mathcal{K}| < |\mathcal{M}|$ then the scheme cannot be perfectly secret. Assume $|\mathcal{K}| < |\mathcal{M}|$. Consider the uniform distribution over $\mathcal{M}$ and let $c \in \mathcal{C}$ be a ciphertext that occurs with non-zero probability. Let $\mathcal{M}(c)$ be the

Information theory example 1

Now lets hope you all know which country has the best rugby team in the known universe...

The term **information** is commonly understood. Consider these sentences:

1. The sun will rise tomorrow.
2. The Fiji rugby team will win against the All Blacks (New Zealand rugby team) the next time they play.

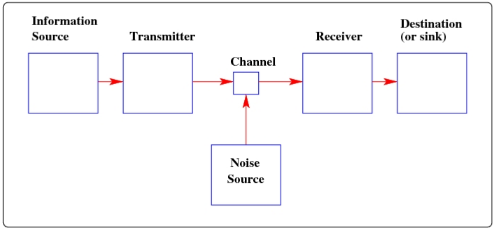
Question: Which sentence contains the most information?

Foundations of information theory

Nyquist (1924) and Hartley (1928) laid the foundations. Hartley showed that the information content is proportional to the *logarithm* of the number of possible messages. Integers between 1 and n need $\log_2 n$ bits.

Shannon developed the mathematical treatment of communication and information leading to “perfect secrecy”, we saw last week.

Information theory model and entropy



Probability and information

In our communication model, the units of transmission (*messages*), are constructed from an alphabet of (say) n symbols $x \in \{x_1, \ldots, x_n\}$ each with a probability of transmission P_x.
We associate with each symbol x a quantity H_x which is a measure of the *information* associated with that symbol.

$$H_x = P_x \times \log_2 \frac{1}{P_x}$$

Entropy - a measure of randomness

Entropy of a source = Amount of information in one symbol

$$H_x = P_x \times \log_2 \frac{1}{P_x}$$

If the probability of occurence of each symbol is the same, we can derive Hartley's result, that the average amount of information transmitted in a single symbol (the source *entropy*) is

$$H(X) = \log_2 n$$

where X is a label referring to each of the source symbols $x_1, \ldots, x_n$.

Units for entropy

- Our units for entropy can be *bits/symbol*, and
- we also sometimes use unit-less entropy measures (relative to the entropy of the system if all symbols were equally likely).

To avoid confusion - we will call this *unit-less* entropy.

Entropy - same probability

Bits needed for equi-probable symbols

Symbols	Entropy of each symbol	Bits needed
2	$H_x = \frac{1}{2}\log_2 2 = \frac{1}{2}$	$2 * \frac{1}{2} = 1$
8	$H_x = \frac{1}{8}\log_2 8 = \frac{3}{8}$	$8 * \frac{3}{8} = 3$
21	$H_x = \frac{1}{21}\log_2 21 = \frac{4.39}{21}$	$21 * \frac{4.39}{21} = 4.39$

What if the symbols are not equally probable/likely?

However, if the probability of occurence of each symbol is not the same, we derive the following result, that the source *entropy* is

$$H(X) = \sum_{i=1}^{n} P_{x_i} \times \log_2 \frac{1}{P_{x_i}}$$

Shannon's paper shows that H determines the channel capacity required to transmit the desired information with the *most* efficient coding scheme.

Entropy - different probability

Example with two symbols and different probabilities

If we had a source emitting two symbols, 0 and 1, with probabilities of 1 and 0, then the entropy of the source is

$$\begin{aligned} H(X) &= \sum_{i=1}^{n} P_{x_i} \times \log_2 \frac{1}{P_{x_i}} \\ &= 1 \times \log_2 1 + 0 \times \log_2 \frac{1}{0} \qquad \text{(Note that } \lim_{y\to 0} y \times \log_2 \frac{1}{y} = 0\text{)} \\ &= 0 \;\; \text{bits/symbol} \end{aligned}$$

Another example: 6 symbols with different probabilities

If we were transmitting a sequence of letters *A,B,C,D,E* and *F* with probabilities $\frac{1}{2}, \frac{1}{4}, \frac{1}{16}, \frac{1}{16}, \frac{1}{16}$ and $\frac{1}{16}$, the entropy for the system is

$$\begin{aligned} H(X) &= \frac{1}{2}\log_2 2 + \frac{1}{4}\log_2 4 + \frac{(1+1+1+1)}{16}\log_2 16 \\ &= 0.5 + 0.5 + 1.0 \\ &= 2 \;\; \text{bits/symbol} \end{aligned}$$

Encoding the letters - bad and good

Two encodings for this situation

Code	A $\Pr[A] = \frac{1}{2}$	B $\Pr[B] = \frac{1}{4}$	C $\Pr[C] = \frac{1}{16}$	D $\Pr[D] = \frac{1}{16}$	E $\Pr[E] = \frac{1}{16}$	F $\Pr[F] = \frac{1}{16}$
3-bit	000	001	010	011	100	101
Complex	0	10	1100	1101	1110	1111

Average number of bits needed for the 3-bit code

$$\begin{aligned}\ell(x) &= \sum_{i=1}^{n} P_{x_i} \times \text{sizeof}(x_i) \\ &= \frac{1}{2} \times 3 + \frac{1}{4} \times 3 + \frac{(1+1+1+1)}{16} \times 3 \\ &= 1.5 + 0.75 + 0.75 \\ &= 3 \;\text{ bits/symbol}\end{aligned}$$

Analysis of encoding

Analysis for the “complex” - variable length encoding

The average length of the bits needed is

$$\begin{aligned}\ell(X) &= \textstyle\sum_{i=1}^{n} P_{x_i} \times \text{sizeof}(x_i) \\ &= \tfrac{1}{2} \times 1 + \tfrac{1}{4} \times 2 + \tfrac{(1+1+1+1)}{16} \times 4 \\ &= 2 \ \text{bits/symbol}\end{aligned}$$

i.e. it is more efficient, averaging only 2 bits for each symbol transmitted.

Relevance to the key size for perfect secrecy

In the textbook, perfect secrecy is defined over uniform distributions - in other words all messages equally likely. In this situation, $|\mathcal{K}| \geq |\mathcal{M}|$. But we have just seen that if messages are not equally likely, they may be encoded in $H(\mathcal{M}) \leq |\mathcal{M}|$ where $H(\mathcal{M})$ is the entropy of the message in bits/message.

Given this, we can imagine another formulation for non-uniform message distributions, where $|\mathcal{K}| \geq H(\mathcal{M})$ or more precisely $H(\mathcal{K}) \geq H(\mathcal{M})$, for (in addition) non-uniform key distributions.

Motivation

Relaxing requirements

We know that perfect secrecy is impossible for short keys. We have been using encryption with short keys for years. Are they not “secure”? Certainly, they are not perfectly secure. However, they might meet weaker (relaxed) and yet practical requirements. What are those requirements?

1. In practice, an adversary doesn’t have unlimited time T to break a scheme.
2. In practice, when it is not perfect (i.e. equality in definition 2.3 doesn’t hold) but close enough, adversary can only meet the attack goal with extremely low probability e (e.g. 1 in 2^{100}). Such low probability practically can be treated as non-occurrence in practice.

Note that there are two components: Attack time T, and Attack success probability e. It is desired to measure security based on (T, e). We want T to be large, e to be small. In a certain sense, perfect secrecy achieves $(T = \infty, e = 0)$.

An asymptotic view

Dealing with (T, e)

For T, similarly to the formulation of an algorithm's running time, we can't use the absolute time. Some asymptotic notion like big-O notation - $\mathcal{O}(m)$ - is required. Same for the probability e.

When comparing the asymptotic performance of two systems M_1 and M_2, we are not comparing two values. We are comparing two functions.
We are asking this question: as the size of the input increases, which system would eventually outperform the other?

Let $F_1(m)$ be the performance of M_1 on input of size m. Likewise, define $F_2(m)$. Now, we want to know which function approaches infinity faster. Alternatively, we may want to know which function approach 0 faster.

Two notions of interest

1. PPT: Polynomial time algorithm.
2. Probability bounded above by a *negligible* function.

Polynomial time algorithms

Definition

An algorithm is polynomial time (PT, polytime) if its expected running time is in $\mathcal{O}(m^t)$ for some t, where m is the size of the input. Specifically:

> Consider an algorithm $\mathcal{A}$. Let us write $\mathcal{A}(m)$ be the worst expected running time on input of size m. We say that $\mathcal{A}$ is *polynomial* time if there exists a t, M such that for all $m > M$, $\mathcal{A}(m) < \mathcal{O}(m^t)$

This idea we will write in shorthand as $p(m)$ indicating polynomial-time.

Notation

PPT algorithm: probabilistic polynomial time algorithms can also flip-a-coin a polynomial number of times.
Most algorithms have different running times on different inputs of the same size. By ”worst” running time, we are taking the max of the expected time over all input of size m.

Background, terms, notation

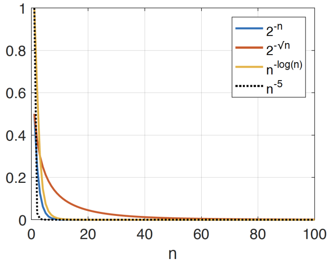
Remarks

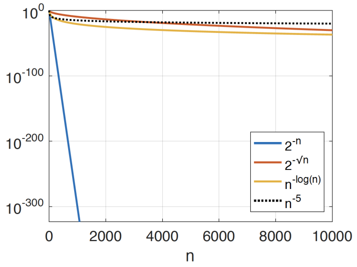
Security parameter. This usually refers to the size of the key, and in the textbook, the notation is n. However, sometimes n refers to the size of the message - Yuch! Some algorithms take in two or more inputs, e.g, $\mathrm{Enc}_k(m)$ takes in the key and the message. There are big-O notations for multiple inputs but they are a bit tedious, so here we take a simple approach: we just use the total input size. That is, if the size of key is $|k|$, and the size of the message is $|m|$, then the polynomial is w.r.t. $|k| + |m|$.

Infeasability: We assume a polynomial time algorithm is feasible, and (say) exponential time algorithms are infeasible. So, *all* adversaries are polynomial time.

Rates of (T, e): In the idea, the computational strength of the adversary is limited to feasible (PPT) power. Computations generally increase in size with larger keys, but for efficiency this increase is practically limited to $T \leq p(n)$. However, the probability of success e should shrink quickly as the security parameter n grows.

Negligable functions





Definition 3.4 - negligable function definition

A non-negative function f from natural to real numbers $f : [0, 1 \ldots] \rightarrow [0, \infty)$, is *negligible* if there is an N s.t. for all $n > N$, $f(n) < \frac{1}{p(n)}$. It is asymptotically smaller than any inverse polynomial function.

Properties of negl

1. Proposition 3.6. $\mathrm{negl}_1 + \mathrm{negl}_2$ is *negligible*
2. $p(n) \times \mathrm{negl}$, is *negligible* where $p(n)$ is any polynomial.

Negligable functions

Our definitions rely on polytime...

Polynomial time algorithms (for both defense and attack) are considered feasible.
And hence the definition of the negligible function involves polynomial time algorithms.

Concrete worked example:

A (PPT) adversary running for $T = n^3$ seconds can succeed in breaking a scheme with probability $e = 2^{40} \times 2^{-n}$. Is this asymptotically secure? **Yes!**

Watch what happens with changing n:

1. When $n = 40$, adversary running for 40^3 seconds (17 hours), can break with probability 1.
2. When $n = 50$, adversary running for 50^3 seconds (34 hours), can break with probability $\frac{1}{1000}$.
3. When $n = 500$, adversary running for 500^3 seconds (nearly 4 years), can break with probability 2^{-460}.

The security parameter is a knob for *tuning* a desired security level

Discussion in the abstract

Discussion will be next week (I think)

Suppose an attack $\mathcal{A}$ will succeed with probability negl. Now consider another attack $\mathcal{B}$ which does the following:

> *On input of size n, carries out $\mathcal{A}$, n^3 times independently. As long as one of the n^3 attacks succeeds, then we consider the attack $\mathcal{B}$ is successful.*

Question: Is the probability of the success of $\mathcal{B}$

1. still negligible? (if so, prove it)
2. no longer negligible? (if so, prove it)
3. depends on the function negl? (if so, give counter examples).

What if $\mathcal{B}$ repeats $\mathcal{A}$ $p(n)$ times, where $p()$ is some polynomial?

Asymptotic security formulation

Main definition approach

It is difficult to incorporate a notion of a polynomial time adversary and less-than-perfect non-leakage into the original definition of perfect secrecy (Definition 2.3).

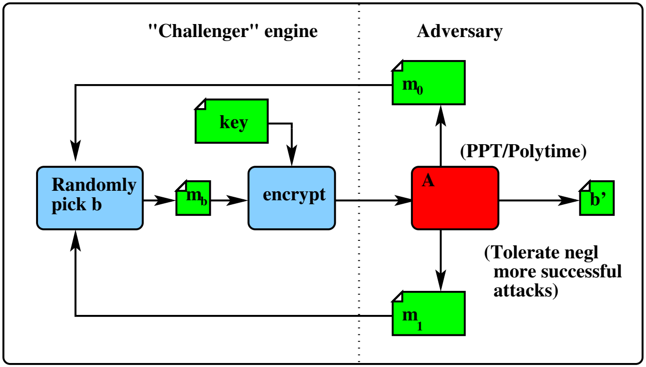
The $\mathrm{PrivK}^{\mathrm{eav}}_{\mathcal{A},\Pi}$ game/experiment definition (Definition 2.5) is equivalent and yet can incorporate notions of time and likelihood. From 2.5, we can derive a reasonable formulation of computational indistinguishability - the main definition 3.8.

General framework of security definitions

Roughly, we say that a system $\mathcal{S}$ is secure if for any PPT adversary $\mathcal{A}$, the probability of its successful game/experiment (compared with a random guess) is *negligible*.

In this approach, we play the same eav/eavesdropper adversarial game $\mathrm{PrivK}^{\mathrm{eav}}_{\mathcal{A},\Pi}$ as before, but now with the adversary $\mathcal{A}$ assumed to be PPT, i.e. $T = \mathcal{O}(p(n))$, and with the probability of success within $e = \mathrm{negl}(n)$. The new game is written $\mathrm{PrivK}^{\mathrm{eav}}_{\mathcal{A},\Pi}(n)$, indicating that it is parameterized by n.

Encryption security...



Definition 3.8 - EAV-security

A private-key encryption Π is EAV-secure if for all PPT adversary $\mathcal{A}$, there exists a negl, s.t.

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

Encryption security...

Definition 3.9 (Equivalent to 3.8)

In this we use $\mathrm{PrivK}^{\mathrm{eav}}_{\mathcal{A},\Pi}(n, b)$ as notation for the game using the bit b, and $\mathrm{out}_{\mathcal{A}}(\mathrm{PrivK}^{\mathrm{eav}}_{\mathcal{A},\Pi}(n, b))$ as the output bit in the game.

A private-key encryption Π is EAV-secure if for all PPT adversary $\mathcal{A}$, there exists a negl, s.t.

$$\left|\Pr[\mathrm{out}_{\mathcal{A}}(\mathrm{PrivK}^{\mathrm{eav}}_{\mathcal{A},\Pi}(n, 0)) = 1] - \Pr[\mathrm{out}_{\mathcal{A}}(\mathrm{PrivK}^{\mathrm{eav}}_{\mathcal{A},\Pi}(n, 1)) = 1]\right| \leq \mathrm{negl}(n)$$

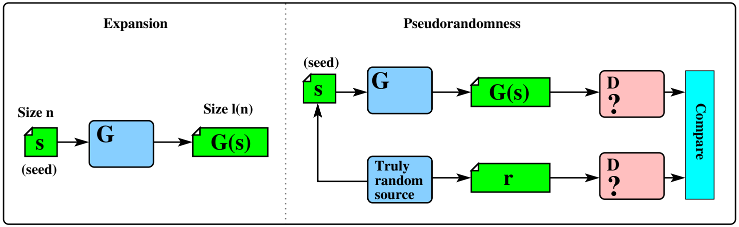
Remarks

Why restrict polytime? As mentioned before, non-poly is practically infeasible.

Why tolerate negl success? In practice, negl success is as good as not successful.

There is a very simple attack that will succeed with negl probability. (read textbook, or think about it).

Definition of a PRG



Definition 3.14

G is deterministic polytime. It is a pseudorandom generator if

1. **Expansion:** Output is longer than the (seed) input $(\ell(n) > n)$.
2. **Pseudorandomness:** For any PPT distinguisher D, there is a negl s.t.

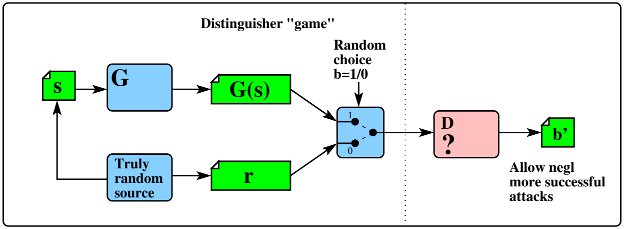
$$|\Pr[D(G(s)) = 1] - Pr[D(r) = 1]| \leq \mathrm{negl}(n) \ |$$

Definition of a PRG

Remark

The previous definition can be read as: For any PPT D, the difference of its behavior on a truly random r and a generated $G(s)$ is *negligible*.

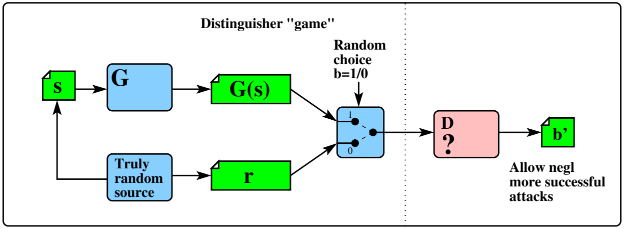
Why is D PPT? Give an example of (no restriction on running time) D s.t. its behavior is different on truly random r and generated $G(s)$.



Discussion

One could give an alternative equivalent definition of pseudorandom generator using a game/experiment in the definition of indistinguishability.

PRG as a game



The game:

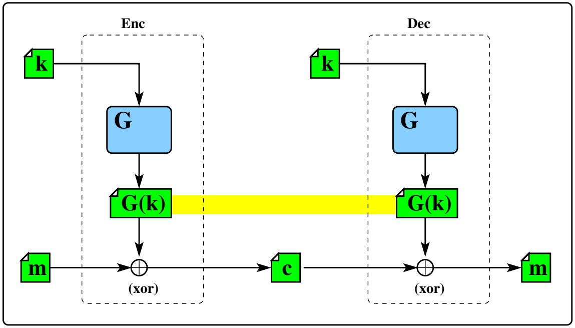
1. Challenger generates a seed s, and constructs $G(s)$ of length $\ell(n)$. Challenger also generates r of length $\ell(n)$, and then randomly chooses a bit b. If $b = 1$ then emit $G(s)$ else emit r. (i.e $g = G(s)$ or $g = r$)
2. Given g, the distinguisher D outputs $b' = 1$ if it determines it is a PRG.

Game-based definition of PRG

It is a PRG if for any PPT distinguisher D, there is a negl s.t.

$$\Pr[D(g) = 1] \leq \frac{1}{2} + \mathrm{negl}(n)$$

Construction of encryption



Construction 3.17: fixed length encryption

Given a pseudorandom generator G with expansion ℓ. Construct the following encryption scheme:

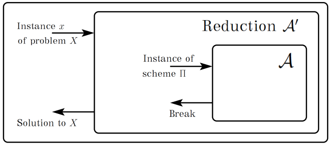
$\mathrm{Gen}(1^n)$**:** on input 1^n, output a truly random key k of size n.

$\mathrm{Enc}_k(m)$**:** on input k and message m of size $\ell(n)$, output $c = G(k) \oplus m$.

$\mathrm{Dec}_k(c)$**:** on input k and ciphertext c of size $\ell(n)$, output $m = G(k) \oplus c$.

Note that this is not a stream cipher. The input size is fixed when n is fixed.

Proof that it is secure? A "reduction proof".



Outline of reduction proofs (p65)

Proof that some new construction Π is secure if some underlying cryptographic primitive X is secure. We often have this situation - for example Diffie-Hellman and factoring large numbers. We start with "Suppose not".

Assume that X is secure but Π is insecure, so this means that a PPT adversary $\mathcal{A}$ attacks Π with success probability $\epsilon(n)$, not $\mathrm{negl}(n)$.

- Make a PPT algorithm $\mathcal{A}'$ that tries to solve instance x of X using $\mathcal{A}$.
- If $\mathcal{A}$ succeeds in breaking Π, $\mathcal{A}'$ solves x with probability $\frac{1}{p(n)}$.
- $\mathcal{A}$ and $\mathcal{A}'$ solve X with probability $\frac{\epsilon(n)}{p(n)}$. $\mathcal{A}$ and $\mathcal{A}'$ are both PPT.

X is solved by PPT $\mathcal{A}'$ with non-negligable probability. A contradiction, so

$$X \text{ is secure} \longrightarrow \Pi \text{ is secure} \qquad \blacksquare$$

Security claim proof

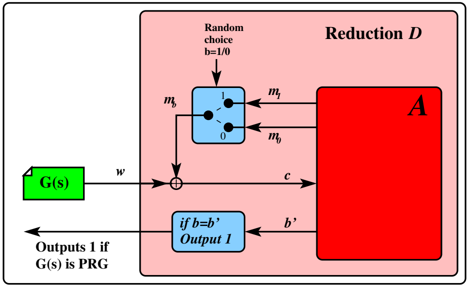
Theorem 3.18

If G is a PRG (3.14, computationally "secure") , then 3.17 is EAV-secure.

Outline of proof

Suppose not! (i.e 3.17 is *not* EAV-secure). This means that there exists an adversary $\mathcal{A}$ (in definition 3.8) that can distinguish between a perfectly secret scheme (where we use r) and the construction (where we use $G(s)$) .

We use this $\mathcal{A}$ to build a D in definition 3.14 that distinguishes $G(s)$. This contradicts that G is a pseudorandom generator.



Security claim proof

Proof

Given $\mathcal{A}$, construct a D that uses $\mathcal{A}$ as a “sub-routine” in the following way:

Distinguisher D: input is a string w of size $\ell(n)$. (main idea: Simulates the game $\mathrm{PrivK}^{\mathrm{eav}}_{\mathcal{A}\Pi}$ (see lecture 2 for steps in the game))

- Simulates step 1 of the game by simulating $\mathcal{A}$ to get m_0, m_1.
- Simulates step 2 of the game by, simulating the challenger. This is done by randomly choosing b. Obtain the ciphertext $c = m_b \oplus w$
- Simulates $\mathcal{A}$. If $\mathcal{A}$ correctly predicts b, then output 1. Otherwise, output 0.

Final step - evaluate the two probabilities

$$\mathrm{Pr}[D(G(s)) = 1] \quad \ldots \quad \mathrm{Pr}[D(r) = 1]$$

The difference is $\frac{\epsilon(n)}{p(n)}$ (not negligible), and so leads to the contradiction. So

$$\text{PRG is secure} \longrightarrow \text{3.17 is secure}$$

■

Conditional security

From Theorem 3.18

The theorem states that G is pseudorandom $\Rightarrow$ Construction is EAV-secure
So, the security of the construction is "conditional"
In contrast, the security of the one-time pad is "unconditional".

Million dollars question: Do we have a pseudorandom generator?
Currently there is no proof. Nevertheless, there are many constructions that are believed to be pseudorandom.
If one can prove the existence of a pseudorandom generator, this implies that $\mathrm{NP} \neq \mathrm{P}$. This is a very big question.

Are we done? Can we go home now?

We have proved that our encryption scheme is EAV-secure, but this only considers ciphertext-only (eavesdropper) attackers. What about situations where we have an attacker who can choose plaintext? In addition, what about multiple observations at the same time?
For this we need next week. More power to the Empire!