Chapter 2: Encryption

January 28, 2018 (last update: 2018-06-20)

Abstract

This post contains solutions to Exercises in Chapter 2[1], which primarily discuss basic concept/building blocks for cryptography study(e.g. Shannon cipher, Definition of Semantic Security, Attack game etc.). On top of that, we dive deeper on further implications and real-life applications motivated by these questions.

2.1 (multiplicative one-time pad)

Question:

2.1 (multiplicative one-time pad). We may also define a "multiplication mod p" variation of the one-time pad. This is a cipher $\mathcal{E} = (E, D)$, defined over $(\mathcal{K}, \mathcal{M}, \mathcal{C})$, where $\mathcal{K} := \mathcal{M} := \mathcal{C} := \{1, \ldots, p-1\}$, where p is a prime. Encryption and decryption are defined as follows:

$$E(k, m) := k \cdot m \bmod p \qquad D(k, c) := k^{-1} \cdot c \bmod p.$$

Here, k^{-1} denotes the multiplicative inverse of k modulo p. Verify the correctness property for this cipher and prove that it is perfectly secure.

Solution:

To prove *correctness*, it's trivial to notice:
$D(k,\ E(k,m)) = k^{-1} \cdot k \cdot m \ mod\ p = m \ mod\ p.$

To prove *perfect secrecy*, we need to use Theorem 2.1, intuitively, what we want to prove is: for a uniformly distributed and randomly selected $k \in \mathcal{K}$, and the probability across all possible ciphertext output is uniform.[2] More formally, we try to show: For
$k \xleftarrow{R} \mathcal{K},\ m_0, m_1 \in \mathcal{M},\ c \in \mathcal{C}$

$$Pr[\ E(k, m_0) = c\ ] \ = \ Pr[\ E(k, m_1) = c\ ] \tag{1}$$

Equation (1) is equivalent to the following statement:

$$\begin{gathered}\text{For any fixed } c,\ m \in GF(p), \\ \text{there exists a unique key } k \in \{0, 1, \ldots, p-1\}, \text{ s.t.} \\ k \cdot m = c \ mod\ p.\end{gathered} \tag{2}$$

Or namely, $k := m^{-1} \cdot c \ mod\ p$.

Since the inverse of an element in a prime field is unique (if any)[3], statement (2) is true, thus complete the proof.

2.2 (A good substitution cipher)

Question:

2.2 (A good substitution cipher). Consider a variant of the substitution cipher $\mathcal{E} = (E, D)$ defined in Example 2.3 where every symbol of the message is encrypted using an *independent* permutation. That is, let $\mathcal{M} = \mathcal{C} = \Sigma^L$ for some a finite alphabet of symbols Σ and some L. Let the key space be $\mathcal{K} = S^L$ where S is the set of all permutations on Σ. The encryption algorithm $E(k, m)$ is defined as

$$E(k, m) := \big(\ k[0](m[0]),\ k[1](m[1]),\ \ldots,\ k[L-1](m[L-1])\ \big)$$

Show that $\mathcal{E}$ is perfectly secure.

Solution:

The approach is similar to Question 2.1: for $\mathcal{M} = \mathcal{C} = \sum^L$ to achieve perfect secrecy,

$$\begin{cases} c[i] \text{ must be independent of } c[j] \text{ for all } i \neq j \\ \text{for all } e \in \sum,\ i \in \{0, 1, \ldots, L-1\}\ Pr[\ e\ ] \text{ is the same} \end{cases}$$

Since $k[i] \xleftarrow{R} S,$ is randomly selected and uniformly distributed, thus:

$$\text{For any } e \in \sum,\ Pr[\ k[i](m[i]) = e\ ] = \frac{1}{|\sum|}$$

And according to Theorem 2.1, $\mathcal{E}$ is perfectly secure.

2.3 (Chain encryption)

Question:

2.3 (Chain encryption). Let $\mathcal{E} = (E, D)$ be a perfectly secure cipher defined over $(\mathcal{K}, \mathcal{M}, \mathcal{C})$ where $\mathcal{K} = \mathcal{M}$. Let $\mathcal{E}' = (E', D')$ be a cipher where encryption is defined as $E'((k_1, k_2), m) := \big(E(k_1, k_2), E(k_2, m)\big)$. Show that $\mathcal{E}'$ is perfectly secure.

Solution:

The proof is almost identical to Solution 2.2, the only worthy note is that: in this proof, we need to point out k_1, k_2, m are independent of each other.

Application: access control on encrypted files:

One might wonder why and where would anybody ever use this “chained encryption” construction, or more straightforward speaking, why not simply encrypting the message with one key instead? As it turns out, such paradigm can be seen in file access control systems.

In many cases, the k_2 in our construction is called a **master key**, while k_1 is called an **ephemeral key**. Indicated by their names, *ephemeral key* is for “temporary” use and can be changed from time to time, whereas *master key* means to stay consistent and intact for longer period.

One always uses the master key to encrypt the file, and the *access control* feature is handled by further encrypting the master key using separate, possibly multiple ephemeral keys, each known to one particular user and distributing the encrypted master keys to users. Those ephemeral keys are most likely directly derived from a password through a KDF.[4] Now this approach has mostly three advantages:

1. User could frequently change their passwords without decrypting and re-encrypting all files on disk. All we need is re-encrypt the master key using this new *ephemeral key* deduced from the new password.

2. *Master key* usually will be randomly generated with a good source of entropy pool, resulting in a bit string that’s computationally indistinguishable from random, better than a passphrase which is more vulnerable to dictionary attacks. Plus the encrypted files could be computed once and locked up in a physically secure location.

3. Easier distribution of the decryption key. More than one user could be granted access to the data and even if one forgets his/her password, the encrypted files don’t end up Gibberish forever.

For interested readers, you could find a fascinating use case here: Using Trusted Hardware to build a Password Manager. In the fourth post of the series, you could find the architecture inside Intel SGX using a chained encryption.

2.4 (A broken one-time pad)

Question:

2.4 (A broken one-time pad). Consider a variant of the one time pad with message space $\{0,1\}^L$ where the key space $\mathcal{K}$ is restricted to all L-bit strings with an even number of 1’s. Give an efficient adversary whose semantic security advantage is 1.

Solution:

An efficient adversary sends two messages of the same length:

$$m_0 = 0^L$$
$$m_1 = 0^{L-1} \,||\, 1$$

To see why this would completely destroy the one-time pad with advantage of 1, when $b = 0$ or Exp(0), we could expect the number of 1 for the output is even. Compared to when $b = 1$ or Exp(1), there are two cases:

$$\begin{cases} k[L-1] = 0 & \rightarrow \text{ number of 1 in the first L-1 bit of the output is even} \\ & \rightarrow \text{ number of 1 in the final output is odd} \\ k[L-1] = 1 & \rightarrow \text{ number of 1 in the first L-1 bit of the output is odd} \\ & \rightarrow \text{ number of 1 in the final output is odd} \end{cases}$$

As we could see, in both cases, the output turns to have odd number of 1 which enables the adversary to distinguish the two game with probability of 1.

2.5 (impossibility result)

Question:

2.5 (A stronger impossibility result). This exercise generalizes Shannon's theorem (Theorem 2.5). Let $\mathcal{E}$ be a cipher defined over $(\mathcal{K}, \mathcal{M}, \mathcal{C})$. Suppose that $\mathsf{SSadv}[\mathcal{A}, \mathcal{E}] \leq \epsilon$ for *all* adversaries $\mathcal{A}$, even including *computationally unbounded* ones. Show that $|\mathcal{K}| \geq (1 - \epsilon)|\mathcal{M}|$.

Solution:

From Shannon Theorem, we know that if $|\mathcal{K}| \leq |\mathcal{M}|$, then there exists $m_0, m_1 \in \mathcal{M}, c \in \mathcal{C}$, s.t. :

$$Pr[E(k,\, m_0) = c] > 0 \text{ and } Pr[E(k,\, m_1) = c] = 0 \qquad (3)$$

Let $Z :=$ event that (3) happens, assuming $|\mathcal{M}| = (1 + \alpha) \cdot |\mathcal{K}|$:

$$\begin{aligned} \rightarrow \quad & Pr[Z] \leq \tfrac{1}{1+\alpha} \cdot \tfrac{\alpha}{1+\alpha} \leq \tfrac{\alpha}{1+\alpha} \\ \because \quad & \text{SSadv}[\mathcal{A}, \mathcal{E}] \leq Pr[Z] \leq \epsilon \\ \rightarrow \quad & \tfrac{\alpha}{1+\alpha} \leq \epsilon \\ \text{or} \quad & \tfrac{1}{1+\alpha} = \tfrac{|\mathcal{K}|}{|\mathcal{M}|} \geq 1 - \epsilon \end{aligned}$$

Thereby finish the proof.

2.6 (a matching bound)

Question:

2.6 (A matching bound). This exercise develops a converse of sorts for the previous exercise. For $j = 0, \ldots, L-1$, let $\epsilon = 1/2^j$. Consider the L-bit one-time pad variant $\mathcal{E}$ defined over $(\mathcal{K}, \mathcal{M}, \mathcal{C})$ where $\mathcal{M} = \mathcal{C} = \{0,1\}^L$. The key space $\mathcal{K}$ is restricted to all L-bit strings whose first $L - j$ bits are not all zero, so that $|\mathcal{K}| = (1-\epsilon)|\mathcal{M}|$. Show that:

(a) there is an efficient adversary $\mathcal{A}$ such that $\mathsf{SSadv}[\mathcal{A}, \mathcal{E}] = \epsilon/(1-\epsilon)$;

(b) for all adversaries $\mathcal{A}$, even including *computationally unbounded* ones, $\mathsf{SSadv}[\mathcal{A}, \mathcal{E}] \leq \epsilon/(1-\epsilon)$.

Note: Since the advantage of $\mathcal{A}$ in part (a) is non-zero, the cipher $\mathcal{E}$ cannot be perfectly secure.

Solution:

(a) Here is an efficient algorithm for the adversary:

$$v \xleftarrow{R} \{0,1\}^j, \quad u \xleftarrow{R} \{0,1\}^j \backslash \{0\}^j$$
$$w \leftarrow v \oplus u$$
$$\begin{cases} m_0 := v \,||\, \{0,1\}^{L-j} \\ m_1 := w \,||\, \{0,1\}^{L-j} \end{cases}$$

To determine the advantage of the adversary $\mathcal{A}$ above, we observe that: if and only if the adversary *accidentally* pick a u that equals to the randomly selected (by challenger) key k, then will $\mathcal{A}$ be able to distinguish the outputs of the two messages. More precisely, in this extremely lucky yet unlikely case, *if our challenger encrypted m_0, then the ciphertext would be exactly m_1, and vice versa.* Thus:

$$SSadv[\mathcal{A}, \mathcal{E}] = Pr[\mathcal{A} \text{ correctly guesses } u] = \frac{1}{2^j - 1} = \frac{\epsilon}{1-\epsilon}$$

(b) Even with computationally unbounded adversary who could operate under a super-poly time and algorithm, which allows exhaustive search through key space, $\mathcal{A}$ still couldn't distinguish encrypted messages with non-negligible probability. In another word, there is no better way than randomly guessing described in (a), therefore the inequality:

$$SSadv[\mathcal{A}, \mathcal{E}] \leqslant \frac{\epsilon}{1-\epsilon}$$

2.7 (deterministic cipher)

Question:

2.7 (Deterministic ciphers). In this exercise, you are asked to prove in detail the claims made in Example 2.9. Namely, show that if $\mathcal{E}$ is a deterministic cipher that is perfectly secure, then $\mathsf{SSadv}[\mathcal{A}, \mathcal{E}] = 0$ for every adversary $\mathcal{A}$ (bearing in mind that $\mathcal{A}$ may be probabilistic); also show that if $\mathcal{E}$ is the variable length one-time pad, then $\mathsf{SSadv}[\mathcal{A}, \mathcal{E}] = 0$ for all adversaries $\mathcal{A}$.

Solution:

(a) If $\mathcal{A}$ is not probabilistic, it's essentially the same as Theorem 2.3. Thus $SSadv[\mathcal{A}_{non-prob}, \mathcal{E}] = 0$;

Else if $\mathcal{A}$ is probabilistic, then it's easy to see that in "bit-guessing game",$SSadv^*[\mathcal{A}_{prob}, \mathcal{E}] = |\ Pr[b = \hat{b}] - \frac{1}{2}\ | = 0$

Putting it together:

$$SSadv[\mathcal{A}, \mathcal{E}] = SSadv[\mathcal{A}_{non-prob}, \mathcal{E}] + 2 \times SSadv^*[\mathcal{A}_{prob}, \mathcal{E}] = 0$$

(b) For every fixed $l \in 0, \ldots, L$, the semantically secure one-time pad has an advantage of 0. Therefore:

$$SSadv[\mathcal{A}, \mathcal{E}] \leqslant \sum_{i=0}^{L-1} SSadv[\mathcal{A}, \mathcal{E}_i] = 0$$

Note:

In the first part of this question, it might be confusing to make sense of a "*probabilistic adversary*". You may be wondering "couldn't any adversary with random source such as `/dev/random` be probabilistic, why specifically pointing it out?".

Now, intuitively, many describe a probabilistic adversary as the ability to *toss a coin*, and such ability or access to random source might provide *additional power*. Because if, for the sake of the discussion, one adversary could only design deterministic strategies, then there are cases where the strategies are actually worse than just "randomly guess". Therefore in many cases, it's more reasonable and natural to assume a probabilistic adversary who would choose whichever strategy with a higher advantage.

2.8 (roulette)

Question:

2.8 (Roulette). In Section 2.3.4, we argued that if value r is encrypted using a semantically secure cipher, then a player's odds of winning at Internet roulette are very close to those of real roulette. However, our "roulette" game was quite simple. Suppose that we have a more involved game, where different outcomes may result in different winnings. The rules are not so important, but assume that the rules are easy to evaluate (given a bet and the number r) and that every bet results in a payout of $0, 1, \ldots, n$ dollars, where n is poly-bounded. Let μ be the expected winnings in an optimal strategy for a real version of this game (with no encryption). Let μ' be the expected winnings of some (efficient) player in an Internet version of this game (with encryption). Show that $\mu \leq \mu' + \epsilon$, where ϵ is negligible, assuming the cipher is semantically secure.

Hint: You may want to use the fact that if X is a random variable taking values in the set $\{0, 1, \ldots, n\}$, the expected value of X is equal to $\sum_{i=1}^{n} \Pr[\mathsf{X} \geq i]$.

Solution:

The analysis is similar to the elementary wrapper in Section 2.3.4, we use exactly the same elementary wrapper.

Recall that in this attack game, p_0 represents the probability of $\mathcal{B}$ outputs 1 in *Experiment 0*, which equals to the probability of $\mathcal{A}$ correctly guessing the outcome by analyzing encrypted outcomes; while *Experiment 1* (so-called "idealized Internet roulette") is cleverly designed to be equivalent to offline real-life game, thus p_1 is equal to the probability of an off-line player winning the game. Obviously $|p_0 - p_1| = SSadv[\mathcal{B}, \mathcal{E}] \leqslant \epsilon$ where ϵ is negligibly small.

Now the main difference in our exercise, is the varying payouts associated with different outcomes. On top of comparing the probability of winning the game, we need further investigating the expected payout in both situations.

In the real-life game, the optimal strategy, even knowing the payout matrix, is still betting randomly. Just to demonstrate, if Alice always bets on outcomes with the highest payout (i.e. n), then the expected winnings will be $\frac{n}{n+1}$. Let's see how much she could win by just randomly bet:

$$\mu = \text{Exp(X)} = \sum_{i=1}^{n} Pr[X \geqslant i] = \frac{n}{2}$$

Meanwhile, in our Internet version of the game (i.e. *Experiment 0*), the elementary wrapper/adversary $\mathcal{B}$ randomly picked a value from $0, \ldots, n$, which effectively identical to the "random selection of value" described in the last paragraph.

Putting all together:

$$\begin{aligned} |\mu - \mu'| &= |\, p_0 \times \textstyle\sum_{i=1}^{n} Pr[X \geqslant i] - p_1 \times \sum_{i=1}^{n} Pr[X \geqslant i] \,| \\ &= \tfrac{n}{2} \times |p_0 - p_1| \\ &\leqslant \tfrac{n}{2} \times \epsilon \end{aligned}$$

Since ϵ is negligible, n is poly-bounded ,the multiplication of them shall be negligible.

2.9 (Fact 2.6)

Question

2.9. Prove Fact 2.6, using the formal definitions in Section 2.4.

Fact 2.6. *If ϵ and ϵ' are negligible values, and Q and Q' are poly-bounded values, then:*

(i) $\epsilon + \epsilon'$ is a negligible value,

(ii) $Q + Q'$ and $Q \cdot Q'$ are poly-bounded values, and

(iii) $Q \cdot \epsilon$ is a negligible value.

Solution

The proof for these three statements leverages the definition of *negligible* and *poly-bounded*, and the process of which resembles that of proving the limit of a function at certain point in your Calculus class.

(i) Since ϵ is negligible, then for $\forall c \in \mathbb{R}_{\geq \mathbb{0}}$, there exists $n_0 \in \mathbb{Z}_{\geq \mathbb{1}}$, such that for all integer $n \geq n_0$, we have $\epsilon = |f_1(n)| \leq 1/n^c$, which by definition of limit in Calculus, we get:$\lim_{n\to\infty} f_1(n)n^c = 0$

Similarly for ϵ', there exists $n_0' \in \mathbb{Z}_{\geq \mathbb{1}}$, such that for all integer $n \geq n_0'$, we have $\epsilon = |f_2(n)| \leq 1/n^c$ and therefore equivalently $\lim_{n\to\infty} f_2(n)n^c = 0$

$$\epsilon + \epsilon' = \lim_{x\to\infty}(f_1(n) + f_2(n)) \times \frac{1}{n^c} = 0$$

Thus we know $\epsilon + \epsilon'$ is a negligible value.

(ii) Let's take more direct approach in this exercise, instead of using limit, we will be using solely the definition itself.

If Q is poly-bounded, then there exists $c_1, d_1 \in \mathbb{R}_{>\mathbb{0}}$ such that for all integers $n > 0$, we have $Q = |f_1(n)| \leq n^{c_1} + d_1$;

Similarly, there exists $c_2, d_2 \in \mathbb{R}_{>\mathbb{0}}$ such that for all integers $n > 0$, we have $Q' = |f_2(n)| \leq n^{c_2} + d_2$.

Now let $c = max\{c_1, c_2\} + 1, \quad d = d_1 + d_2$, we could derive the following:

$$\begin{gathered} \text{For all integers } n > 0, \\ Q_{sum} = Q + Q' \leq |f_1(n)| \leq n^{c_1} + d_1 + |f_2(n)| \leq n^{c_2} + d_2 \\ \leq n^c + d \end{gathered}$$

By definition of *poly-bounded*, we could say $Q + Q'$ is also poly-bounded.

Proving multiplicative result is similar. Pick $c = 2 \times max\{c_1, c_2\} + 1, \quad d = max\{d_1, d_2\}^2 + \sqrt[max\{c_1,c_2\}]{2max\{d_1, d_2\}}$. It is indeed quite troublesome, without loss of generality, let's assume $c_2 \geq c_1, d_2 \geq d_1$:

$$
\begin{aligned}
\mathcal{Q}_{mul} &= \mathcal{Q} \times \mathcal{Q}' \\
&\leq (n^{c_1} + d_1) \times (n^{c_2} + d_2) \\
&\leq (n^{c_2} + d_2)^2 \\
&\leq n^{2c_2} + 2d_2 n_{c_2} + d_2^2 \\
&\leq n^{2c_2+1} + \sqrt[c_2]{2d_2} + d_2^2 \\
&= n^c + d
\end{aligned}
$$

Thus by definition proves the multiplication of two poly-bounded function is also poly-bounded.

(iii) Almost identical to **(i)**, hint: $\lim_{n\to\infty} f(n)(n^c + d) = 0$

2.10 (definition of semantic security)

Question:

2.10 (Exercising the definition of semantic security). Let $\mathcal{E} = (E, D)$ be a semantically secure cipher defined over $(\mathcal{K}, \mathcal{M}, \mathcal{C})$, where $\mathcal{M} = \mathcal{C} = \{0,1\}^L$. Which of the following encryption algorithms yields a semantically secure scheme? Either give an attack or provide a security proof via an explicit reduction.

(a) $E_1(k, m) := 0 \parallel E(k, m)$

(b) $E_2(k, m) := E(k, m) \parallel \text{parity}(m)$

(c) $E_3(k, m) := \text{reverse}(E(k, m))$

(d) $E_4(k, m) := E(k, \text{reverse}(m))$

Here, for a bit string s, parity(s) is 1 if the number of 1's in s is odd, and 0 otherwise; also, reverse(s) is the string obtained by reversing the order of the bits in s, e.g., reverse$(1011) = 1101$.

Solution:

(a) and **(c)** are both semantically secure scheme. By constructing an elementary wrapper $\mathcal{B}$ for $\mathcal{A}$, we could show that $SSadv[\mathcal{B}, \mathcal{E}'] = SSadv[\mathcal{A}, \mathcal{E}]$.

Specifically, in **(a)**, $\mathcal{B}$ sends two messages of the same length like the usual attack game, and upon receiving from its challenger, it "stripe off" the prepending "0" and send it to $\mathcal{A}$, then it outputs $\hat{b}$ the same as whatever $\mathcal{A}$ outputs;

In **(c)**, we replace "striping off 0" operation with "inverse encrypted ciphertext" operation and get the same efficient adversary $\mathcal{B}$.

(b) is NOT semantically secure: it would be completely broke if $\mathcal{A}$ sends two messages with different parity. For example:
$m_0 = \{0\}^L, \quad m_1 = \{0\}^{L-1} \,||\, 1.$

(d) is semantically secure, but the proof is different from that of the **(a)** or **(c)**. Instead of having an elementary wrapper, $\mathcal{A}$ becomes the wrapper for another adversary $\mathcal{D}$, for instance. And the Experiments start by $\mathcal{D}$ randomly selecting two different bit strings of the same length and send them to $\mathcal{A}$ who reverses both and forwards to its (i.e. $\mathcal{A}'s$) challenger. Upon receiving the ciphertext, $\mathcal{A}$ forward it directly to $\mathcal{D}$ and outputs whatever $\mathcal{D}$ outputs.

It's easy to see the $SSadv[\mathcal{D}, \mathcal{E}'] = SSadv[\mathcal{A}, \mathcal{E}]$, which proves **(d)**'s security.

2.11 (key recovery attacks)

Question:

> ***2.11 (Key recovery attacks).*** Let $\mathcal{E} = (E, D)$ be a cipher defined over $(\mathcal{K}, \mathcal{M}, \mathcal{C})$. A key recovery attack is modeled by the following game between a challenger and an adversary $\mathcal{A}$: the challenger chooses a random key k in $\mathcal{K}$, a random message m in $\mathcal{M}$, computes $c \xleftarrow{R} E(k, m)$, and sends (m, c) to $\mathcal{A}$. In response $\mathcal{A}$ outputs a guess $\hat{k}$ in $\mathcal{K}$. We say that $\mathcal{A}$ wins the game if $D(\hat{k}, c) = m$ and define $\mathrm{KRadv}[\mathcal{A}, \mathcal{E}]$ to be the probability that $\mathcal{A}$ wins the game. As usual, we say that $\mathcal{E}$ is secure against key recovery attacks if for all efficient adversaries $\mathcal{A}$ the advantage $\mathrm{KRadv}[\mathcal{A}, \mathcal{E}]$ is negligible.
>
> (a) Show that the one-time pad is not secure against key recovery attacks.
>
> (b) Show that if $\mathcal{E}$ is semantically secure and $\epsilon = |\mathcal{K}|/|\mathcal{M}|$ is negligible, then $\mathcal{E}$ is secure against key recovery attacks. In particular, show that for every efficient key-recovery adversary $\mathcal{A}$ there is an efficient semantic security adversary $\mathcal{B}$, where $\mathcal{B}$ is an elementary wrapper around $\mathcal{A}$, such that
>
> $$\mathrm{KRadv}[\mathcal{A}, \mathcal{E}] \leq \mathrm{SSadv}[\mathcal{B}, \mathcal{E}] + \epsilon$$
>
> ***Hint:*** Your semantic security adversary $\mathcal{B}$ will output 1 with probability $\mathrm{KRadv}[\mathcal{A}, \mathcal{E}]$ in the semantic security Experiment 0 and output 1 with probability at most ϵ in Experiment 1. Deduce from this a lower bound on $\mathrm{SSadv}[\mathcal{B}, \mathcal{E}]$ in terms of ϵ and $\mathrm{KRadv}[\mathcal{A}, \mathcal{E}]$ from which the result follows.
>
> (c) Deduce from part (b) that if $\mathcal{E}$ is semantically secure and $|\mathcal{M}|$ is super-poly then $|\mathcal{K}|$ cannot be poly-bounded.
>
> ***Note:*** $|\mathcal{K}|$ can be poly-bounded when $|\mathcal{M}|$ is poly-bounded, as in the one-time pad.

Solution:

(a) As one-time pad uses a deterministic algorithm whose inverse operation is easy, namely $\hat{k} = m \oplus c$, which yields $KRadv[\mathcal{A}, \mathcal{E}] = 1$, thus one-time pad is completely broken in key recovery attack games.

(b) We build an elementary wrapper adversary $\mathcal{B}$ as follows:

Upon receiving m from $\mathcal{A}$, does the following:

$$m_0 \leftarrow m, m_1 \xleftarrow{R} \mathcal{M}$$

then, pass m_0 and m_1 to its challenger. Upon receiving ciphertext c from the challenger, directly feed to $\mathcal{A}$, which will return $\hat{k}$. Finally the output algorithm for $\mathcal{B}$ to its semantic security challenger:

$$\text{If } D(\hat{k}, c) = m_0, \text{then outputs 1;}$$
$$\text{otherwise, outputs 0.}$$

Security analysis: in Exp(0), $Pr[W_0] = Pr[\mathcal{A} \text{ wins key recovery game}] = KRadv[\mathcal{A}, \mathcal{E}]$, because challenger of $\mathcal{B}$ encrypted $m_0 = m$ in this case, which makes it exactly the same as the key recovery game for $\mathcal{A}$; Now in Exp(1), since $c \leftarrow E(k, m_1)$ where m_1 is not related to m, thus $\hat{k}$ is independent of actual key of the ciphertext. However, there's still chance $\mathcal{B}$ outputs 1 in this experiment, namely cases where $\mathcal{A}$ guesses a different key which decrypt to the same plaintext m by accident, and based on this blind luck, we have: $Pr[W_1] = \frac{|\mathcal{K}|}{|\mathcal{M}|}$.

Putting all together:

$$\begin{aligned} SSadv[\mathcal{B}, \mathcal{E}] &= |Pr[W_0] - Pr[W_1]| \\ &= KRadv[\mathcal{A}, \mathcal{E}] - \frac{|\mathcal{K}|}{|\mathcal{M}|} \end{aligned}$$

As long as ϵ is negligible, and $\mathcal{E}$ is semantically secure, cipher should be key recovery resistant. [5]

(c) First conclusion to draw from the first two questions is: key recovery security is a sufficient but not necessary condition for semantic security of the underlying cipher. Especially, one needs to come to the realization that semantic security does NOT imply key recovery attack resistance.

If $|\mathcal{M}|$ is super-poly and $|\mathcal{K}|$ is poly-bounded, it's clear to see $KRadv[\mathcal{A}, \mathcal{E}]$ is also negligible, since both $SSadv[\mathcal{A}, \mathcal{E}]$ and ϵ are negligible, which directly contradicts with our realization.

2.12 (security against message recovery)

Question:

2.12 (Security against message recovery). In Section 2.3.3.1 we developed the notion of security against message recovery. Construct a cipher that is secure against message recovery, but is not semantically secure.

Solution:

Consider a substitution cipher, where a random permutation of the message is applied to derive its ciphertext. It is completely secure against message recovery, as brutal forcing the cipher text requires iterating through all possible permutations $P : \{0,1\}^{|M|} \rightarrow \{0,1\}^{|C|}$ [6].

However such substitution cipher is completely broken in SS attack game, for instance:

$$m_0[0] = m_0[1] \rightarrow c[0] = c[1]$$
$$m_1[0] \neq m_1[1] \rightarrow c[0] \neq c[1]$$

2.13 (advantage calculation)

Question:

2.13 (Advantage calculations in simple settings). Consider the following two experiments Experiment 0 and Experiment 1:

- In Experiment 0 the challenger flips a fair coin (probability 1/2 for HEADS and 1/2 for TAILS) and sends the result to the adversary $\mathcal{A}$.
- In Experiment 1 the challenger always sends TAILS to the adversary.

The adversary's goal is to distinguish these two experiments: at the end of each experiment the adversary outputs a bit 0 or 1 for its guess for which experiment it is in. For $b = 0, 1$ let W_b be the event that in experiment b the adversary output 1. The adversary tries to maximize its distinguishing advantage, namely the quantity

$$\left|\Pr[W_0] - \Pr[W_1]\right| \quad \in [0, 1] \ .$$

If the advantage is negligible for all efficient adversaries then we say that the two experiments are indistinguishable.

(a) Calculate the advantage of each of the following adversaries:

(i) $\mathcal{A}_1$: Always output 1.

(ii) $\mathcal{A}_2$: Ignore the result reported by the challenger, and randomly output 0 or 1 with even probability.

(iii) $\mathcal{A}_3$: Output 1 if HEADS was received from the challenger, else output 0.

(iv) $\mathcal{A}_4$: Output 0 if HEADS was received from the challenger, else output 1.

(v) $\mathcal{A}_5$: If HEADS was received, output 1. If TAILS was received, randomly output 0 or 1 with even probability.

(b) What is the maximum advantage possible in distinguishing these two experiments? Explain why.

Solution:

(a):

(i): $SSadv[\mathcal{A}, \mathcal{E}] = |Pr[W_0] - Pr[W_1]| = |\frac{1}{2} - 1| = \frac{1}{2}$

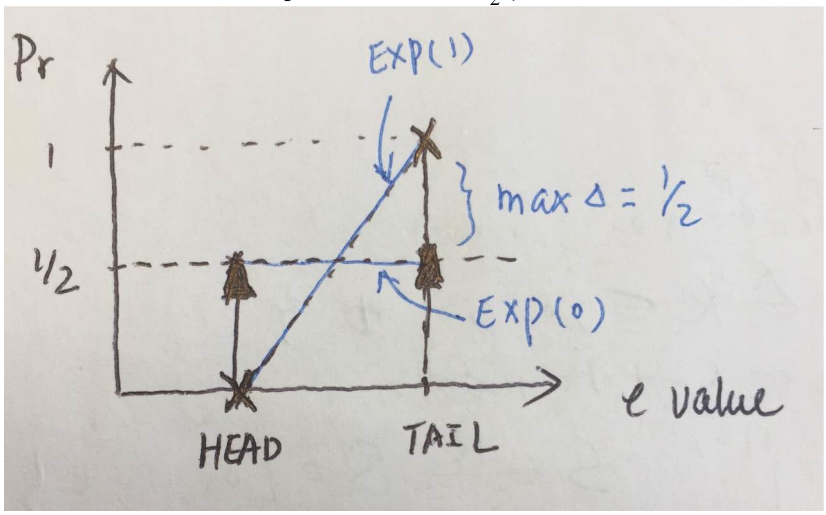
(ii): $SSadv[\mathcal{A}, \mathcal{E}] = |Pr[W_0] - Pr[W_1]| = |\frac{1}{4} - \frac{1}{2}| = \frac{1}{4}$

(iii): $SSadv[\mathcal{A}, \mathcal{E}] = |Pr[W_0] - Pr[W_1]| = |\frac{1}{2} - 0| = \frac{1}{2}$

(iv): $SSadv[\mathcal{A}, \mathcal{E}] = |Pr[W_0] - Pr[W_1]| = |\frac{1}{2} - 1| = \frac{1}{2}$

(v): $SSadv[\mathcal{A}, \mathcal{E}] = |Pr[W_0] - Pr[W_1]| = |(\frac{1}{2} + \frac{1}{2} \times \frac{1}{2}) - \frac{1}{2}| = \frac{1}{4}$

(b): The maximum advantage is $\frac{1}{2}$, my intuitively proof is shown as follow (i.e. the maximum discrepancy between the probability distribution over the two possible values is $\frac{1}{2}$.) [7]:



2.14 (permutation cipher)

Question:

2.14 (Permutation cipher). Consider the following cipher (E, D) defined over $(\mathcal{K}, \mathcal{M}, \mathcal{C})$ where $\mathcal{C} = \mathcal{M} = \{0,1\}^\ell$ and $\mathcal{K}$ is the set of all $\ell!$ permutations of the set $\{0, \ldots, \ell - 1\}$. For a key $k \in \mathcal{K}$ and message $m \in \mathcal{M}$ define $E(k, m)$ to be result of permuting the bits of m using the permutation k, namely $E(k, m) = m[k(0)]...m[k(\ell - 1)]$. Show that this cipher is not semantically secure by showing an adversary that achieves advantage 1.

Solution:

$$m_0[0] = m_0[1] \rightarrow c[0] = c[1]$$
$$m_1[0] \neq m_1[1] \rightarrow c[0] \neq c[1]$$
$$\Rightarrow SSadv[\mathcal{A}, \mathcal{E}] = 1$$

2.15 (nested encryption)

Question:

2.15 (Nested encryption). For a cipher $\mathcal{E} = (E, D)$ define the nested cipher $\mathcal{E}' = (E', D')$ as

$$E'\big((k_0, k_1), m\big) = E\big(k_1, E(k_0, m)\big) \quad \text{and} \quad D'\big((k_0, k_1), c\big) = D(k_0, D(k_1, c)) \ .$$

Our goal is to show that if $\mathcal{E}$ is semantically secure then $\mathcal{E}'$ is semantically secure even if the adversary is given one of the keys k_0 or k_1.

(a) Consider the following semantic security experiments, Experiments 0 and 1: in Experiment b, for $b = 0, 1$, the adversary generates two messages m_0 and m_1 and gets back k_1 and $E'\big((k_0, k_1),\ m_b\big)$. The adversary outputs $\hat{b}$ in $\{0, 1\}$ and we define its advantage, $\mathrm{NE}\mathsf{adv}[\mathcal{A}, \mathcal{E}]$ as in the usual the definition of semantic security. Show that for every nested encryption adversary $\mathcal{A}$ attacking $\mathcal{E}'$, there exists a semantic security adversary $\mathcal{B}$ attacking $\mathcal{E}$, where $\mathcal{B}$ is an elementary wrapper around $\mathcal{A}$, such that

$$\mathrm{NE}\mathsf{adv}[\mathcal{A}, \mathcal{E}] = \mathrm{SS}\mathsf{adv}[\mathcal{B}, \mathcal{E}] \ .$$

Draw a diagram with $\mathcal{A}$ on the right, $\mathcal{B}$ in the middle, and $\mathcal{B}$'s challenger on the left. Show the message flow between these three parties that takes place in your proof of security.

(b) Repeat part (a), but now when the adversary gets back k_0 (instead of k_1) and $E'\big((k_0, k_1), m_b\big)$ in Experiments 0 and 1. Draw a diagram describing the message flow in your proof of security as you did in part (a).

This problem comes up in the context of anonymous routing on the Internet as discussed in Section 2.5.

Solution:

(a): Upon receiving m_0, m_1 from $\mathcal{A}$, the elementary wrapper $\mathcal{B}$ works as follow:

- it forwards m_0, m_1 to $\mathcal{B}$'s SS challenger, who would randomly choose a key $k_0 \xleftarrow{R} \mathcal{K}$ and send back $c_0 \leftarrow E(k_0, m_b)$ to $\mathcal{B}$
- $\mathcal{B}$ further selects $k_1 \xleftarrow{R} \mathcal{K}$ and send the nestedly encrypted $c_1 \leftarrow E(k_1, c_0)$ together with k_1 to $\mathcal{A}$
- finally $\mathcal{B}$ will output whatever $\mathcal{A}$ outputs

Let p_b be the probability $\mathcal{A}$ outputs 1 in its nested encryption game, $Pr[W_b]$ be the probability of $\mathcal{B}$ outputs 1 in its SS game. We could observe that $Pr[W_0] = p_0, Pr[W_1] = p_1$, thus: [8]

$$SSadv[\mathcal{B}, \mathcal{E}] = |p_0 - p_1| = NEadv[\mathcal{A}, \mathcal{E}] \qquad (4)$$

(b): [9] Now in Exp(0), c' and k_0 are not related, thus $p_0 = \frac{1}{2}$; while in Exp(1), let p be the probability of $\mathcal{A}$ outputs a $\hat{b} : b = \hat{b}$, clearly $p_1 = p$.

By definition, $SSadv[\mathcal{B}, \mathcal{E}] = |p_0 - p_1|$ and $NEadv[\mathcal{A}, \mathcal{E}] = |p - \frac{1}{2}|$. From there, we could see how Eq.4 still hold. [10]

2.16 (self-referential encryption)

Question:

2.16 (Self referential encryption). Let us show that encrypting a key under itself can be dangerous. Let $\mathcal{E}$ be a semantically secure cipher defined over $(\mathcal{K}, \mathcal{M}, \mathcal{C})$, where $\mathcal{K} \subseteq \mathcal{M}$, and let $k \xleftarrow{R} \mathcal{K}$. A ciphertext $c_* := E(k, k)$, namely encrypting k using k, is called a **self referential encryption**.

(a) Construct a cipher $\tilde{\mathcal{E}} = (\tilde{E}, \tilde{D})$ derived from $\mathcal{E}$ such that $\tilde{\mathcal{E}}$ is semantically secure, but becomes insecure if the adversary is given $\tilde{E}(k, k)$. You have just shown that semantic security does not imply security when one encrypts one's key.

(b) Construct a cipher $\hat{\mathcal{E}} = (\hat{E}, \hat{D})$ derived from $\mathcal{E}$ such that $\hat{\mathcal{E}}$ is semantically and remains semantically secure (provably) even if the adversary is given $\hat{E}(k, k)$. To prove that $\hat{\mathcal{E}}$ is semantically secure, you should show the following: for every adversary $\mathcal{A}$ that attacks $\hat{\mathcal{E}}$, there exists and adversary $\mathcal{B}$ that attacks $\mathcal{E}$ such that (i) the running time $\mathcal{B}$ is about the same as that of $\mathcal{A}$, and (ii) $\mathsf{SSadv}[\mathcal{A}, \hat{\mathcal{E}}] \leq \mathsf{SSadv}[\mathcal{B}, \mathcal{E}]$.

Solution:

(a): $\tilde{E} = E(k, m) \oplus E(k, m \oplus k)$

To break $\tilde{E}$, we set $m_0 = \{0\}^L, m_1 \xleftarrow{R} |\mathcal{M}|$. Observing that $\tilde{E}(k, 0) = \tilde{E}(k, k)$, thus given the latter, the adversary could tell which experiment he/she's in with absolute confidence, or probability of 1.

(b): $\hat{E} = E(k, m \oplus k)$ or $E(k, m) \oplus k$ etc.

2.17 (compression encryption)

Question:

2.17 (Compression and encryption). Two standards committees propose to save bandwidth by combining compression (such as the Lempel-Ziv algorithm used in the zip and gzip programs) with encryption. Both committees plan on using the variable length one time pad for encryption.

- One committee proposes to compress messages before encrypting them. Explain why this is a bad idea.

  Hint: Recall that compression can significantly shrink the size of some messages while having little impact on the length of other messages.

- The other committee proposes to compress ciphertexts after encryption. Explain why this is a bad idea.

Over the years many problems have surfaced when combining encryption and compression. The CRIME [108] and BREACH [104] attacks are good representative examples.

Solution:

"compress-then-encrypt": information leakage from the length of the cipher text, as during the compression, only some portion of the messages are significantly shrinked. While the variable one-time pad could be perfectly semantic secure, yet in reality, length of the compressed file (encrypted or unencrypted) could reveal information about the original file.

"encrypt-then-compress": would require a really big key, at least as large as the file, which could sometimes be gigabytes in size.

2.18 (voting protocol)

Question:

2.18 (Voting protocols). This exercise develops a simple voting protocol based on the additive one-time pad (Example 2.4). Suppose we have t voters and a counting center. Each voter is going to vote 0 or 1, and the counting center is going to tally the votes and broadcast the total sum S. However, they will use a protocol that guarantees that no party (voter or counting center) learns anything other than S (but we shall assume that each party faithfully follows the protocol).

The protocol works as follows. Let $n > t$ be an integer. The counting center generates an encryption of 0: $c_0 \xleftarrow{R} \{0, \ldots, n-1\}$, and passes c_0 to voter 1. Voter 1 adds his vote v_1 to c_0, computing $c_1 \leftarrow c_0 + v_1 \bmod n$, and passes c_1 to voter 2. This continues, with each voter i adding v_i to c_{i-1}, computing $c_i \leftarrow c_{i-1} + v_i \bmod n$, and passing c_i to voter $i+1$, except that voter t passes c_t to the counting center. The counting center computes the total sum as $S \leftarrow c_t - c_0 \bmod n$, and broadcasts S to all the voters.

(a) Show that the protocol correctly computes the total sum.

(b) Show that the protocol is perfectly secure in the following sense. For voter $i = 1, \ldots, t$, define $View_i := (S, c_{i-1})$, which represents the "view" of voter i. We also define $View_0 := (c_0, c_t)$, which represents the "view" of the counting center. Show that for each $i = 0, \ldots, t$ and $S = 0, \ldots, t$, the following holds:

> as the choice of votes $v_1, \ldots, v_t$ varies, subject to the restrictions that each $v_j \in \{0, 1\}$ and $\sum_{j=1}^{t} v_j = S$, the distribution of $View_i$ remains the same.

(c) Show that if two voters i, j collude, they can determine the vote of a third voter k. You are free to choose the indices i, j, k.

Solution:

(a)

$$\begin{aligned} \text{From protocol:} \quad & C_t = (C_0 + \Sigma V_i) \bmod n \\ \rightarrow \quad & \sum v_i = (C_t - C_0) \bmod n \\ \rightarrow \quad & \textit{correctness} \end{aligned}$$

(b):

(c):

2.19 (two-way split keys)

Question:

2.19 (Two-way split keys). Let $\mathcal{E} = (E, D)$ be a semantically secure cipher defined over $(\mathcal{K}, \mathcal{M}, \mathcal{C})$ where $\mathcal{K} = \{0,1\}^d$. Suppose we wish to split the ability to decrypt ciphertexts across two parties, Alice and Bob, so that both parties are needed to decrypt ciphertexts. For a random key k in $\mathcal{K}$ choose a random r in $\mathcal{K}$ and define $k_a := r$ and $k_b := k \oplus r$. Now if Alice and Bob get together they can decrypt a ciphertext c by first reconstructing the key k as $k = k_a \oplus k_b$ and then computing $D(k, c)$. Our goal is to show that neither Alice nor Bob can decrypt ciphertexts on their own.

(a) Formulate a security notion that captures the advantage that an adversary has in breaking semantic security given Bob's key k_b. Denote this 2-way key splitting advantage by $\mathsf{2KSadv}[\mathcal{A}, \mathcal{E}]$.

(b) Show that for every 2-way key splitting adversary $\mathcal{A}$ there is a semantic security adversary $\mathcal{B}$ such that $\mathsf{2KSadv}[\mathcal{A}, \mathcal{E}] = \mathsf{SSadv}[\mathcal{B}, \mathcal{E}]$.

Solution:

2.20 (simple secret sharing)

Question:

2.20 (Simple secret sharing). Let $\mathcal{E} = (E, D)$ be a semantically secure cipher with key space $\mathcal{K} = \{0,1\}^L$. A bank wishes to split a decryption key $k \in \{0,1\}^L$ into three shares p_0, p_1, and p_2 so that two of the three shares are needed for decryption. Each share can be given to a different bank executive, and two of the three must contribute their shares for decryption to proceed. This way, decryption can proceed even if one of the executives is out sick, but at least two executives are needed for decryption.

(a) To do so the bank generates two random pairs (k_0, k_0') and (k_1, k_1') so that $k_0 \oplus k_0' = k_1 \oplus k_1' = k$. How should the bank assign shares so that any two shares enable decryption using k, but no single share can decrypt?

Hint: The first executive will be given the share $p_0 := (k_0, k_1)$.

(b) Generalize the scheme from part (a) so that 3-out-of-5 shares are needed for decryption. Reconstituting the key only uses XOR of key shares. Two shares should reveal nothing about the key k.

(c) More generally, we can design a t-out-of-w system this way for any $t < w$. How does the size of each share scale with t? We will see a much better way to do this in Section 11.6.

Solution:

2.21 (simple threshold encryption)

Question:

2.21 (Simple threshold decryption). Let $\mathcal{E} = (E, D)$ be a semantically secure cipher with key space $\mathcal{K}$. In this exercise we design a system that lets a bank split a key k into three shares p_0, p_1, and p_2 so that two of the three shares are needed for decryption, as in Exercise 2.20. However, decryption is done without ever reconstituting the complete key at a single location.

We use nested encryption from Exercise 2.15. Choose a random key $k := (k_0, k_1, k_2, k_3)$ in $\mathcal{K}^4$ and encrypt a message m as:

$$c \xleftarrow{R} \Big(E\big(k_1, E(k_0, m)\big), \quad E\big(k_3, E(k_2, m)\big) \Big).$$

(a) Construct the shares p_0, p_1, p_2 so that any two shares enable decryption, but no single share can decrypt. Hint: the first share is $p_0 := (k_0, k_3)$.

Discussion: Suppose the entities holding shares p_0 and p_2 are available to decrypt. To decrypt a ciphertext c, first send c to the entity holding p_2 to partially decrypt c. Then forward the result to the entity holding p_0 to complete the decryption. This way, decryption is done without reconstituting the complete key k at a single location.

(b) Generalize the scheme from part (a) so that 3-out-of-5 shares are needed for decryption. Explain how decryption can be done without reconstituting the key in a single location.

An encryption scheme where the key can be split into shares so that t-out-of-w shares are needed for decryption, and decryption does not reconstitute the key at a single location, is said to provide **threshold decryption**. We will see a much better way to do this in Section 11.6.

Solution:

2.22 (bias correction)

Question:

2.22 (Bias correction). Consider again the bit-guessing version of the semantic security attack game (i.e., Attack Game 2.4). Suppose an efficient adversary $\mathcal{A}$ wins the game (i.e., guesses the hidden bit b) with probability $1/2 + \epsilon$, where ϵ is non-negligible. Note that ϵ could be positive or negative (the definition of negligible works on absolute values). Our goal is to show that there is another efficient adversary $\mathcal{B}$ that wins the game with probability $1/2+\epsilon'$, where ϵ' is non-negligible and positive.

(a) Consider the following adversary $\mathcal{B}$ that uses $\mathcal{A}$ as a subroutine in Attack Game 2.4 in the following two-stage attack. In the first stage, $\mathcal{B}$ plays challenger to $\mathcal{A}$, but $\mathcal{B}$ generates its own hidden bit b_0, its own key k_0, and eventually $\mathcal{A}$ outputs its guess-bit $\hat{b}_0$. Note that in this stage, $\mathcal{B}$'s challenger in Attack Game 2.4 is not involved at all. In the second stage, $\mathcal{B}$ restarts $\mathcal{A}$, and lets $\mathcal{A}$ interact with the "real" challenger in Attack Game 2.4, and eventually

$\mathcal{A}$ outputs a guess-bit $\hat{b}$. When this happens, $\mathcal{B}$ outputs $\hat{b} \oplus \hat{b}_0 \oplus b_0$. Note that this run of $\mathcal{A}$ is completely independent of the first — the coins of $\mathcal{A}$ and also the system parameters are generated independently in these two runs.

Show that $\mathcal{B}$ wins Attack Game 2.4 with probability $1/2 + 2\epsilon^2$.

(b) One might be tempted to argue as follows. Just construct an adversary $\mathcal{B}$ that runs $\mathcal{A}$, and when $\mathcal{A}$ outputs $\hat{b}$, adversary $\mathcal{B}$ outputs $\hat{b} \oplus 1$. Now, we do not know if ϵ is positive or negative. If it is positive, then $\mathcal{A}$ satisfies are requirements. If it is negative, then $\mathcal{B}$ satisfies our requirements. Although we do not know which one of these two adversaries satisfies our requirements, we know that one of them definitely does, and so existence is proved.

What is wrong with this argument? The explanation requires an understanding of the mathematical details regarding security parameters (see Section 2.4).

(c) Can you come up with another efficient adversary $\mathcal{B}'$ that wins the bit-guessing game with probability at least $1 + |\epsilon|/2$? Your adversary $\mathcal{B}'$ will be less efficient than $\mathcal{B}$.

Solution:

Shame-free Cheat Sheet

Theorem 2.1:

Theorem 2.1. *Let $\mathcal{E} = (E, D)$ be a Shannon cipher defined over $(\mathcal{K}, \mathcal{M}, \mathcal{C})$. The following are equivalent:*

(i) $\mathcal{E}$ is perfectly secure.

(ii) For every $c \in \mathcal{C}$, there exists N_c (possibly depending on c) such that for all $m \in \mathcal{M}$, we have

$$|\{k \in \mathcal{K} : E(k, m) = c\}| = N_c.$$

(iii) If the random variable k is uniformly distributed over $\mathcal{K}$, then each of the random variables $E(\mathsf{k}, m)$, for $m \in \mathcal{M}$, has the same distribution.

Theorem 2.5:

Theorem 2.5 (Shannon's theorem). *Let $\mathcal{E} = (E, D)$ be a Shannon cipher defined over $(\mathcal{K}, \mathcal{M}, \mathcal{C})$. If $\mathcal{E}$ is perfectly secure, then $|\mathcal{K}| \geq |\mathcal{M}|$.*

Theorem 2.3:

Theorem 2.3. *Let $\mathcal{E} = (E, D)$ be a Shannon cipher defined over $(\mathcal{K}, \mathcal{M}, \mathcal{C})$. Consider a probabilistic experiment in which $\mathbf{k}$ is a random variable uniformly distributed over $\mathcal{K}$. Then $\mathcal{E}$ is perfectly secure if and only if for every predicate ϕ on $\mathcal{C}$, for all $m_0, m_1 \in \mathcal{M}$, we have*

$$\Pr[\phi(E(\mathbf{k}, m_0))] = \Pr[\phi(E(\mathbf{k}, m_1))].$$

Bit-guessing:

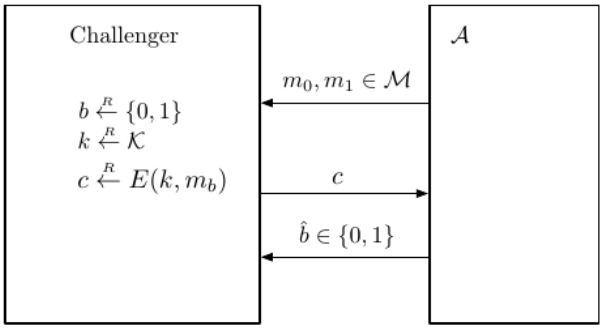
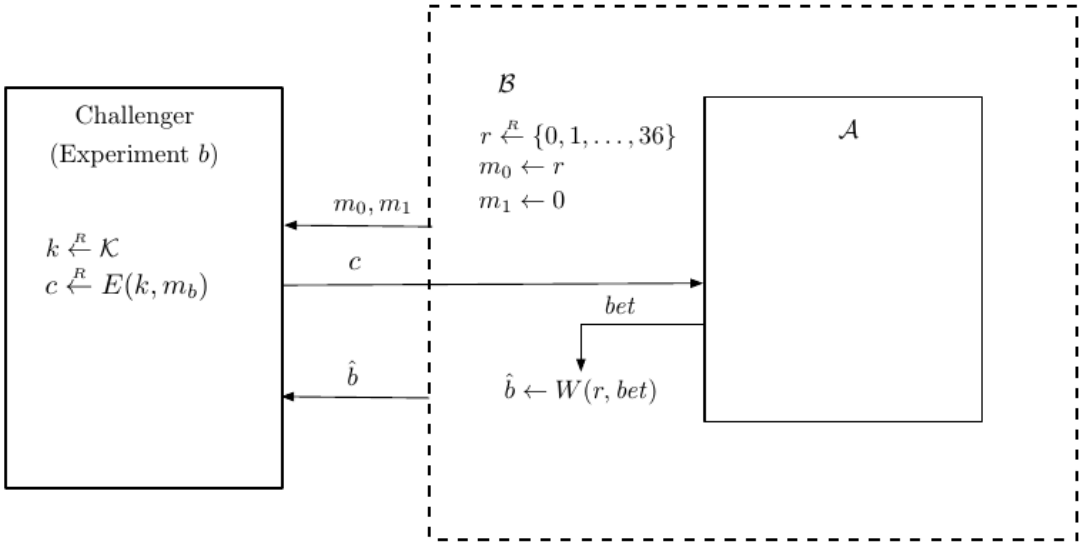


Figure 2.5: Attack Game 2.4

Theorem 2.10. *For every cipher $\mathcal{E}$ and every adversary $\mathcal{A}$, we have*

$$\mathsf{SSadv}[\mathcal{A}, \mathcal{E}] = 2 \cdot \mathsf{SSadv}^*[\mathcal{A}, \mathcal{E}]. \tag{2.10}$$

Section 2.3.4:


Challenger
(Experiment b)
$k \xleftarrow{R} \mathcal{K}$
$c \xleftarrow{R} E(k, m_b)$
$\mathcal{B}$
$r \xleftarrow{R} \{0, 1, \ldots, 36\}$
$m_0 \leftarrow r$
$m_1 \leftarrow 0$
$\mathcal{A}$
m_0, m_1
c
bet
$\hat{b}$
$\hat{b} \leftarrow W(r, bet)$