7.1 Analyzing the Security of Encryption

Are any of the constructions from the last section secure? If so, can we say that some are more secure than others? The old approach to answering these sorts of questions, dating back to the birth of cryptography, is to think about how the constructions might be used and to look for attacks. In practice this approach is effective for some types of problems, and is still important today; In future sets of notes we’ll look at encryption security from that point of view.

Since about the 1980’s, another approach has been developed that extends Shannon’s idea of giving mathematical definitions of security. For better or worse, this approach is called “provable security.” We have seen a bit of this already in our analysis of stream ciphers, when we used a definition of pseudorandom generator distinguishing advantage.

At a high level, the provable security approach works in three distinct steps: (1) Defining security goals, (2) Defining computational assumptions, and (3) Proving security via reductions. Each of these steps involves idiosyncrasies that will be discussed in turn.

In these notes we begin this process, identifying an initial definition and exploring its implications.

7.1.1 Defining Security Goals

Find a definition of “security” that enumerates what an adversary is allowed to do and how much computational effort it my expend. In this step we seek to formalize something like

“The adversary may mount a chosen-plaintext attack, and employ any algorithmic strategy in \( 2^{100} \) time to analyze ciphertexts to learn about plaintexts it does not know.”

The salient features of this informal definition include the adversary capabilities (chosen-plaintext attack), a time bound \( 2^{100} \), and a goal (learning about plaintexts that it does not know). Also central to this approach is that we want to allow any algorithmic strategy, including ones we don’t know about ourselves. This is remarkable because the current state of theoretical computer science is not very good at reasoning about the limits of general time-bounded algorithms. This situation is very similar to that of pseudorandom generators, where we wanted to think only of time-bounded distinguishers failing, but we have no real tools for proving anything conclusive about them.
7.2 One-Time Chosen-Plaintext Security of Encryption

With that said, let’s just run through a definition and interpret it later. In this definition, the symbol $A$ refers to an “algorithm”, which we think of informally; It’s an entity that can be “run”, with or without input. It has some internal persistent memory, and the next time it is run it will “remember” everything it has seen.

The definition uses the notation $x \overset{\$}{\leftarrow} X$ to mean “select a random sample from the set $X$, and call it $x$.”

**Definition 7.1.** Let $\Pi = (\text{Enc}, \text{Dec})$ be a deterministic encryption scheme with key-space $K$, message-space $M$, and ciphertext-space $C$. We assume that the message-space $M$ is a set of bit strings, i.e. $M \subseteq \{0,1\}^*$. Let $A$ be an algorithm. Define algorithm $\text{Expt}_{\Pi}^{\text{1-cca}}(A)$ as

<table>
<thead>
<tr>
<th>Alg $\text{Expt}_{\Pi}^{\text{1-cca}}(A)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>01 Run $A$, which produces $(m_0, m_1) \in M$</td>
</tr>
<tr>
<td>02 If $m_0$ and $m_1$ are different lengths: Output 0</td>
</tr>
<tr>
<td>03 Pick $k \overset{$}{\leftarrow} K, b \overset{$}{\leftarrow} {0,1}$</td>
</tr>
<tr>
<td>04 Compute $c = \text{Enc}(k, m_b)$ and run $A(c)$, which produces $\hat{b} \in {0,1}$</td>
</tr>
<tr>
<td>05 If $\hat{b} = b$: Output 1</td>
</tr>
<tr>
<td>06 Else: Output 0</td>
</tr>
</tbody>
</table>

Define the one-time CPA advantage of $A$ against $\Pi$ as

$$\text{Adv}_{\Pi}^{\text{1-cca}}(A) = \left| \Pr[\text{Expt}_{\Pi}^{\text{1-cca}}(A) = 1] - \frac{1}{2} \right|$$

The algorithm $\text{Expt}_{\Pi}^{\text{1-cca}}(A)$ should be thought of as a “test-harness” for $A$. This algorithm is not something we would ever use in practice, as it does not do anything useful other than evaluate a potential adversary $A$. As with pseudorandom generators, the advantage $\text{Adv}_{\Pi}^{\text{1-cca}}(A)$ is a “score”, this time between 0 and 1/2: The higher the advantage, the better the adversary.

Most of the complexity is in this definition is in $\text{Expt}_{\Pi}^{\text{1-cca}}(A)$. Intuitively, the experiment has $A$ declare two messages that it would like to be challenged on. If $A$ chooses messages of different lengths, then it automatically loses (because the experiment outputs 0, which means the adversary has lost). Otherwise, the experiment picks a random key and a bit $b$, and encrypts $m_b$ under the chosen key and hands the ciphertext back to the adversary. The adversary then outputs its guess $\hat{b}$: Intuitively it is trying to determine which message was encrypted, and $\hat{b}$ represents which message it thinks was more likely to be encrypted. If correct, the experiment indicates a win by outputting 1, and otherwise it indicates a loss with 0.

Note that even a very simple $A$ can win with probability 1/2, say by always outputting 0. The goal of the adversary is to do better than that. In practice, an adversary running in time (say) $2^{100}$ and achieving $\text{Adv}_{\Pi}^{\text{1-cca}}(A) = 1/2^{100}$ may be considered a “break,” depending on the exact parameters like key-length.

**Example 7.1.** Let $\Pi = (\text{Enc}, \text{Dec})$ be ECB encryption with AES. We give an efficient $A$ such that $\text{Adv}_{\Pi}^{\text{1-cca}}(A) = 1/2$. It works as follows:
Adversary $A$

01 Output $m_0 = 0^{128} || 0^{128}, m_1 = 0^{128} || 1^{128}$.
02 On input $c$, parse $c[1] || c[2] \leftarrow c$, where $c[1], c[2] \in \{0, 1\}^{128}$.
03 If $c[1] = c[2]$: Output $\hat{b} = 0$
04 Else: Output $\hat{b} = 1$.

Let’s check that $\Pr[\text{Expt}_{\Pi}^{1\text{-cpa}}(A) = 1] = 1$. If $b = 0$, then the message $m_0 = 0^{128} || 0^{128}$ will be encrypted, and we will have $c[1] = c[2]$ and $A$ will output $\hat{b} = 0$. If $b = 1$ then we’ll get that $A$ outputs $\hat{b} = 1$ instead, because $c[1] = \text{AES}(k, 0^{128}) \neq \text{AES}(k, 1^{128}) = c[2]$. Thus $\hat{b} = b$ with probability 1.

The next example shows that good one-time CPA security implies it is hard to compute the first bit of a message from a ciphertext.

Example 7.2. Let $\Pi = (\text{Enc}, \text{Dec})$ be a some deterministic encryption scheme with $\mathcal{M} = \{0, 1\}^{128}$. Suppose $A$ has the property that for all $k \in \mathcal{K}, m \in \mathcal{M}$, $A(\text{Enc}(k, m))$ outputs the first bit of $m$.

Consider the following adversary $B$:

Adversary $B$

01 Output $m_0 = 0^{128}, m_1 = 1^{128}$.
02 On input $c$, run $A(c)$, which outputs a bit $d$.
04 Output $\hat{b} = d$.

Once again, let’s check that $\Pr[\text{Expt}_{\Pi}^{1\text{-cpa}}(B) = 1] = 1$. If $b = 0$, then the message $m_0 = 0^{128}$ will be encrypted, and $A$ (and hence $B$) will output 0. If $m_1 = 1^{128}$ then the message $m_0 = 1^{128}$ will be encrypted, and $B$ will end up outputting 1 for the same reason. Putting these together, $\hat{b} = b$ with probability 1.

7.2.1 Relation to Perfect Secrecy

The following theorem formalizes the relationship between one-time CPA security and perfect secrecy.

Theorem 1. Let $\Pi = (\text{Enc}, \text{Dec})$ be a deterministic encryption scheme with $\mathcal{M} = \{0, 1\}^\ell$ for some integer $\ell$. Then $\Pi$ is perfectly secret if and only if for all adversaries $A$,

$$\text{Adv}_{\Pi}^{1\text{-cpa}}(A) = 0.$$ 

We will not prove it formally, but it is important to understand this intuitively: If all pairs of messages generate ciphertexts with the same probability, then there is no way to win the experiment with probability better than random guessing.

Corollary 1. Let $\Pi_{\text{otp}} = (\text{Enc}_{\text{otp}}, \text{Dec}_{\text{otp}})$ be the $\ell$-bit one-time pad encryption scheme. Then for all adversaries $A$,

$$\text{Adv}_{\Pi_{\text{otp}}}^{1\text{-cpa}}(A) = 0,$$

and in other words

$$\Pr[\text{Expt}_{\Pi_{\text{otp}}}^{1\text{-cpa}}(A) = 1] = \frac{1}{2}.$$