Lecture 8

1 Some More Security Definitions for Encryption Schemes

1.1 Real-or-random (rr) security

Real-or-random security, the definition of security that we used in previous lectures, measures the indistinguishability of the encryption of a plaintext with the encryption of a randomized plaintext. $E_K \circ \$$ represents an oracle that takes a plaintext query, creates a new random plaintext of the same length, and returns the encryption of that randomized plaintext, so the advantage of an adversary in the real-or-random sense is

$$Adv^{rr} A = \left| Pr[A^{E_K} = 1] - Pr[A^{E_K \circ \$} = 1] \right|.$$

1.2 Find-then-guess (fg) security

The adversary A in find-then-guess security is a two-phase algorithm.

- Phase 1: "find". In this phase A may make queries to its oracle, but must eventually return a triple (m_0, m_1, s), where m_0 and m_1 are two messages of the same length (and s is a notational convention indicating that A may maintain some state between the two phases.) The messages m_0 and m_1 are essentially the challenge that A is taking on; A is saying, "I bet I can distinguish between the encryptions of these two messages."
- Phase 2: "guess". In transitioning to this phase, one of $\{m_0, m_1\}$ is picked at random, encrypted with E_K, and the result is given to A. A may then make more oracle queries, but must output a guess as to which message's encryption it was given.

The advantage of A can be written in the following way:

$$Adv^{fg} A = 2 \left| Pr[(m_0, m_1, s) \leftarrow A^{E_K}(\text{“}find\text{”}); b \xleftarrow{\$} \{0,1\}; A^{E_K}(\text{“}guess\text{”}, s, E_K(m_b)) = b] - \frac{1}{2} \right|.$$

Since an adversary could achieve a probability $\frac{1}{2}$ of guessing right by chance alone, we want to measure how much better than that A is doing. The factor of 2 in front is so that an absolute difference of $\frac{1}{2}$ from $Pr[\cdot] = \frac{1}{2}$, i.e. 0 or 1, means an advantage of 1. An alternative but identical formulation of the advantage expression, more similar to that given for real-or-random security, is

$$\begin{aligned} Adv^{fg} A = \big| & Pr[(m_0, m_1, s) \leftarrow A^{E_K}(\text{“}find\text{”}); A^{E_K}(\text{“}guess\text{”}, s, E_K(m_0)) = 1] - \\ & Pr[(m_0, m_1, s) \leftarrow A^{E_K}(\text{“}find\text{”}); A^{E_K}(\text{“}guess\text{”}, s, E_K(m_1)) = 1] \big|. \end{aligned}$$

There are a few points to note about find-then-guess security. If E_K is deterministic, then it is necessarily find-and-guess insecure. A could simply pick two messages, query its oracle to find the encryptions of those two messages, and then when we transition to the "guess" phase and give the encryption of one of the two messages to A, it will know for sure which one it is. E_K being deterministic means that the encryption it produces for m_b in the challenge portion is the same as the one it gave in answer to one of A's oracle queries.

Also, note that being find-then-guess secure does not mean that "any pair of messages is indistinguishable when encrypted"; it only means that "any pair of messages that an adversary A may come up with is indistinguishable when encrypted."

Finally, note that a one-time pad is secure under this definition (as it is under all of these), because each oracle query m in either phase, from the point of view of the adversary, returns $m \oplus$ a never-before-seen uniform random value (a new pad) = a random value, from which no information can be gleaned.

1.3 Left-or-right (*lr*) security

Left-or-right security is a generalization of find-then-guess security that allows an adversary to try any number of this-or-that challenges. First, a little bit of notation: the selector functions $\S_0$ and $\S_1$ each take two arguments; $\S_0$ returns its first, and $\S_1$ returns its second. That is,

$$\S_0(m, m') = m \text{ and } \S_1(m, m') = m'.$$

Thus the oracles $E_K \circ \S_0$ and $E_K \circ \S_1$ each take two input plaintexts; the first returns the encryption of the first input, and the second returns the encryption of the second input. The adversary A is given one of these two oracles, but it does not know which. In other words, it's either in a "left" world, where its oracle queries always result in the encryption of the left argument, or it's in a "right" world, where its oracle queries always result in the encryption of the right argument.

A may make some number of oracle queries, and the result of each may give it some information about whether it's in the "left" world or the "right" one. Note that each query (m, m') must consist of two messages of the same length, and also that if A wants to make a "traditional" oracle query and get back the encryption of one message m for sure, it can just make the query (m, m). Eventually A must make a guess about which world it's in; like for find-then-guess security, we can express its advantage equivalently as either

$$Adv^{lr}A = \left| Pr[A^{E_K \circ \S_0} = 1] - Pr[A^{E_K \circ \S_1} = 1] \right|$$

or

$$Adv^{lr}A = 2 \left| Pr[b \xleftarrow{\$} \{0,1\}; A^{E_K \circ \S_b} = b] - \frac{1}{2} \right|.$$

1.4 Randomness of ciphertexts (*rc*) security

The definition of advantage under randomness of ciphertexts looks quite similar to that for real-or-random:

$$Adv^{rc}A = \left| Pr[A^{E_K} = 1] - Pr[A^{\$\circ E_K} = 1] \right|$$

but the order of function composition for the "ideal world" oracle is reversed. In real-or-random security, the "ideal world" oracle randomizes the plaintext before encrypting it, whereas in this definition the ciphertext is randomized. Essentially, then, *rc*-security means that a ciphertext produced by E_K is indistinguishable from a random string of the same length.

To clarify the difference, consider counter mode, which we found earlier to be secure in the real-or-random sense. However, it is trivially insecure in the randomness of ciphertexts sense, because a sequence of ciphertexts from counter mode would begin with a successively incrementing counter, a pattern which would have a very low probability of occuring in randomized ciphertexts.

1.5 Semantic security

In semantic security, a quite powerful adversary is permitted. The adversary can send not merely a single message to its oracle, but a distribution over messages, and the oracle will sample from it and encrypt the result. (Chosen-plaintext queries can still be made by choosing a distribution concentrated in one message.) Furthermore, the adversary is not required to distinguish between two ciphertexts as in *fg*- and *lr*-security; instead, it picks a function encapsulating some notion of partial information that it claims it can figure out from some sequence of messages, and then must only show that it can in fact get that partial information.

Of course, an adversary could just pick a function whose value is a known constant for any sequence of messages, and then say "look, I get it right every time!" So to determine the advantage of an adversary, we find the difference between the probability that it's correct, and the probability that it would be correct when given no useful information; in other words, the advantage measures the ability of the adversary to make use of the information given to it by the oracle.

To implement a real/ideal world split, we use a selector function akin to the one used in left-or-right security, and combine it with the sampling operation. So there are two sampling/selecting functions, $S_0(\mathcal{M})$ and $S_1(\mathcal{M})$, that take a distribution over messages $\mathcal{M}$. The pseudocode of $S_0(\mathcal{M})$ and $S_1(\mathcal{M})$ is as follows:

```
1: $i \leftarrow i + 1$
2: $M_i \xleftarrow{\$} \mathcal{M}()$
3: $M'_i \xleftarrow{\$} \mathcal{M}()$
4: **return** $\S_b(M_i, M'_i)$
```

Algorithm 1: $S_b(\mathcal{M})$

where $\S_b$ is the selector function defined in Section 1.3. A sequence of samples $(M_1, \ldots, M_q)$ from $\mathcal{M}$ is produced and used as input to A's partial information function; in the case of S_0 (the "real world"), that sequence is also encrypted and given to A, and in the case of S_1 (the "ideal world"), another sequence of the same length is independently sampled, encrypted, and given to A. Now a brief detour to give the definition of advantage in this sense:

$$AdvA = \left|Pr[(f,Y) \leftarrow A^{E_K \circ S_0}; f(M_1,\ldots,M_q) = Y] - Pr[(f,Y) \leftarrow A^{E_K \circ S_1}; f(M_1,\ldots,M_q) = Y]\right|$$

The output of the adversary A is a pair (f, Y), which are respectively, a function over a sequence of messages, and a claim as to the result of that function. It is important to note that in the $f(M_1, \ldots, M_q) = Y$ expression where we check to see if A's claim was correct, the sequence $M_1, \ldots, M_q$ is the "left" of the two sequences produced by q invocations of the oracle. In the "real world", using oracle $E_K \circ S_0$, that same sequence, encrypted, was given to A as the result of its oracle queries. But in the "ideal world", using oracle $E_K \circ S_1$, A was given the encryption of a *different* sequence from the one put into f.

In summary, semantic security asks the question of an adversary, "Can you guess some function of a bunch of messages better than you could guess without actually knowing the messages used to compute the function?"

1.6 An Insecure Example: CBC Initialized With a Counter

For an example of an encryption scheme which is not secure under any of the above definitions, consider CBCC, that is, CBC with a counter used as the initialization vector.

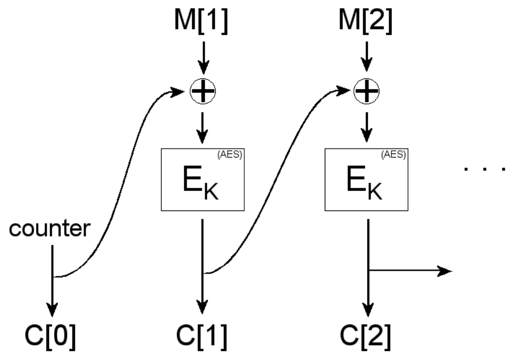

Figure 1: CBC w/counter

Each successive (possibly multi-block) message has a $C[0]$ one greater than the previous message.

1.6.1 Not real-or-random secure

Consider the one-block message M consisting of 128 0s, and M' consisting of 127 0s and a 1, i.e. $0000\cdots01$. An adversary A has a pretty good way of determining whether it's in the real (non-randomizing oracle) or ideal (oracle that randomizes plaintexts before encrypting) world.

First, query with M, resulting (in the real world) in $(C[0], C[1]) = (\mathit{counter}, AES(0 \oplus \mathit{counter})) = (0, AES(0))$. Then, query with M', resulting in $(C[0], C[1]) = (\mathit{counter}, AES(1 \oplus \mathit{counter})) = (1, AES(0))$. Thus $C[1]$ in both cases will be the same, $AES(0)$. Since for this to be true, $M[1] \oplus 0 = M'[1] \oplus 1$, which is very unlikely to occur for randomized plaintexts M and M', checking whether it is the case serves as a very good test of wether we're in the real world or the ideal world.

1.6.2 Not find-then-guess secure

The following algorithm has a very high advantage against CBCC in the find-then-guess sense.

```
1: $E \leftarrow$ the oracle's response to $M$
2: return $(m_0, m_1, s) = (M', \$, E)$
```

Algorithm 2: "Find" phase

```
1: if $E_K(m_b) = E$ then
2:     return "b = 0"
3: else
4:     return "b = 1"
5: end if
```

Algorithm 3: "Guess" phase

As seen above, E will be $(0, AES(0))$. $E_K(m_0) = E$, so if the test in line 1 of the guess phase is false, b is definitely 1. $E_K(m_1) = AES(\$)$, i.e. AES of a uniform random number, which is very unlikely to be equal to E, so if the test in line 1 is true, b is 0 with very high probability.

1.6.3 Not left-or-right secure

This can be shown in an analagous way as for real-or-random security. The adversary A that has a very high advantage in the left-or-right sense does the following. First send the query $(M, \$)$, and then $(M', \$)$; if the second blocks of the results are the same, we can assume with high probability that we are in the left world, and if not, we are definitely in the right world.

1.6.4 Not randomness-of-ciphertexts secure

A sequence of outputs from CBCC is trivially distinguishable from random messages of the same length, because the first block of successive outputs from CBCC will simply increment starting at 0.

2 Implications Between Security Definitions

One could go on indefinitely creating different definitions of cryptographic security, and all of the above seem plausible in their own ways, so a natural question to ask is whether any imply others, whether we can use the fact that a scheme is secure under one definition to instantly know that it's secure under another. Also, the explanation of CBCC's insecurity in the left-or-right sense should seem suspiciously similar to that of its insecurity in the real-or-random sense, leading one to wonder whether there's a correspondence that can be automated.

In fact, there are several implications that can be proven between these different security notions, forming an equivalence class and some hangers-on. Let's first give an overview of the relationships and then sketch proofs of some.

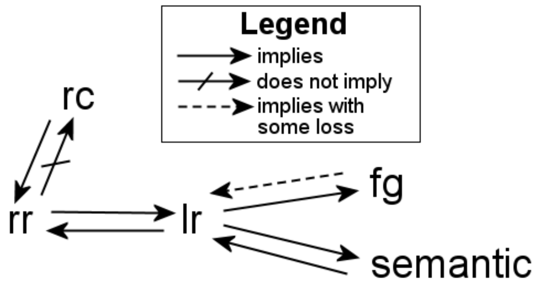

Figure 2: Implications between security definitions

2.1 Left-to-right ⇒ real-or-random

This implication can be shown by generalizing the way in which the real-or-random insecurity "proof" of CBCC was transformed to show its left-or-right insecurity. The statement $lr \Rightarrow rr$ means that left-or-right security implies real-or-random security, or, in the contrapositive, real-or-random *insecurity* implies left-or-right *insecurity.* This leads to a common way to show implications of this type: showing how any successful real-or-random attack can be transformed into a left-or-right attack.

An algorithm A_{rr} consists of a series of oracle queries $M_1, \ldots, M_q$ followed by a guess, "real or ideal world?" The equivalent left-or-right algorithm A_{lr} runs, for each query M_i, the query $(M_i, \$(M_i))$, where $\$(M_i)$ means a random message of the same length at M_i (remember that oracle queries in the left-or-right system are pairs.) Having made these queries, A_{lr} being in the "left" world is equivalent to A_{rr} being in the "real" world (plaintext queries not being randomized) and A_{lr} being in the "right" world is equivalent to A_{rr} being in the "ideal" world (plaintext queries being randomized.) Consequently it can be proven that an encryption scheme that is (t, q, ϵ)-secure in the left-or-right sense is $(t - \text{a little bit}, q, \epsilon)$-secure in the real-or-random sense.

2.2 Real-or-random ⇒ left-to-right

For brevity, let's use the notation $D_1 \sim D_2$ to indicate that two oracles are computationally indistinguishable from each other. If an encryption algorithm E_K is real-or-random secure, then $E_K \sim E_K \circ \$$. By the data processing lemma this also holds when E_K and $E_K \circ \$$ are composed with another function, so

$$\begin{aligned} E_K \circ \S_0 &\sim E_K \circ \$ \circ \S_0 \text{ and likewise} \\ E_K \circ \S_1 &\sim E_K \circ \$ \circ \S_1. \end{aligned}$$

Since a restriction on a left-or-right adversary is that any query must consist of a pair of messages of the same length, $\$ \circ \S_0$ and $\$ \circ \S_1$ both simply return random messages of that length, and thus are indistinguishable. Combining all these relations into a hybrid argument,

$$E_K \circ \S_0 \sim E_K \circ \$ \circ \S_0 \sim E_K \circ \$ \circ \S_1 \sim E_K \circ \S_1$$

and therefore

$$E_K \circ \S_0 \sim E_K \circ \S_1,$$

which means that Adv^{lr} of any adversary A is very low. Specifically, if E_K is (t, q, ϵ)-secure in the real-or-random sense, then it's $(t - \text{a little bit}, q, 2\epsilon)$-secure in the left-or-right sense (the factor of 2 coming from the two uses of the $E_K \sim E_K \circ \$$ relation.)

2.3 Real-or-random $\not\Rightarrow$ randomness of ciphertexts

The lack of an implication between two security definitions can be proven with just a counterexample. In this case, an easy counterexample is counter mode, which is real-or-random secure as we discussed in previous lectures but is not randomness-of-ciphertexts secure as shown in Section 1.4.

N.B.: It is not known for certain whether any practical real-or-random secure schemes in fact exist. It may be the case that $P = NP$, and none do exist, so this non-implication should be phrased carefully as "If a real-or-random secure scheme exists, then there exists a scheme which is real-or-random secure but not randomness-of-ciphertexts secure."

2.4 Left-or-right $\Rightarrow$ find-then-guess

To show this implication, let's use the same method as in Section 2.1, providing a mechanical way to transform a successful find-then-guess adversary A_{fg} into a left-or-right adversary A_{lr}. The run of A_{fg} consists of some number of single oracle queries and one "give me the encryption of one of these two messages, and I'll guess which one." Each single oracle query M can be converted into an oracle query (M, M) made by A_{lr}, and the pair (m_0, m_1) that A_{fg} returns from its "find" phase can be just used as is as an oracle query by A_{lr}. A_{lr} must then guess which message's encryption was returned using the same process as A_{fg}, making queries (M, M) where A_{fg} makes queries M.

2.5 Find-then-guess $\Rightarrow$ left-or-right

For this implication we will start out with not the contrapositive (left-or-right insecurity implies find-then-guess insecurity), but the original implication, using the assumption of find-then-guess security to show left-or-right security. An encryption scheme being left-or-right secure means that the oracles $E_K \circ \S_0$ and $E_K \circ \S_1$ are indistinguishable. We will show this to be the case using a hybrid argument.

Consider a chain of hybrid oracles $\mathcal{H}_0, \ldots, \mathcal{H}_i, \ldots, \mathcal{H}_q$, defined as follows:

$$\begin{array}{rl}
\mathcal{H}_0: & \text{answers all queries } (m_n, m'_n) \text{ with } E_K(m_n) \\
\mathcal{H}_1: & \text{answers first query } (m_1, m'_1) \text{ with } E_K(m'_1)\text{, rest with } E_K(m_n) \\
 & \vdots \\
\mathcal{H}_i: & \text{answers first } i \text{ queries } (m_n, m'_n) \text{ with } E_K(m'_n)\text{, rest with } E_K(m_n) \\
 & \vdots \\
\mathcal{H}_q: & \text{answers all queries } (m_n, m'_n) \text{ with } E_K(m'_n)
\end{array}$$

Note that $\mathcal{H}_0$ is, by definition, $E_K \circ \S_0$, and $\mathcal{H}_q$ is, by definition, $E_K \circ \S_1$. Also, any two adjacent oracles $\mathcal{H}_i$ and $\mathcal{H}_{i+1}$ are indistinguishable: consider a find-then-guess adversary A that makes i queries $m'_1, \ldots, m'_i$ in the "find" phase, returns the pair (m_{i+1}, m'_{i+1}), and then makes $q - i - 1$ queries $m_{i+2}, \ldots, m_q$ in the "guess" phase. If $\mathcal{H}_i$ and $\mathcal{H}_{i+1}$ were distinguishable, then A could distinguish between $E_K(m_{i+1})$ and $E_K(m'_{i+1})$ (the only way $\mathcal{H}_i$ and $\mathcal{H}_{i+1}$ differ), yielding a high $Adv^{fg}A$ and violating our assumption of find-then-guess security.

Thus we have the similarity chain

$$E_K \circ \S_0 \equiv \mathcal{H}_0 \sim \cdots \sim \mathcal{H}_i \sim \cdots \sim \mathcal{H}_q \equiv E_K \circ \S_1$$

showing that, assuming that E_K is find-then-guess secure, it is also left-or-right secure. However, note that this implication is weaker that the others that we have discussed. Each step in the above similarity chain is true because E_K is (t, q, ϵ)-secure in the find-then-guess sense, and thus each step has a separation of ϵ. $E_K \circ \S_0 \sim E_K \circ \S_1$ via that chain, yielding a total separation of $q\epsilon$, so E_K is only $(t, q, q\epsilon)$-secure in the left-or-right sense.

2.6 Summary of Relationships

Putting aside randomness of ciphertexts, which is somewhat separate from the other notions of security we've discussed because it is not implied by the others, we have an equivalence class of sorts because *rr*, *lr*, *fg* (ignoring the factor of q loss), and semantic security all imply each other. This is an important discovery, because each of these security definitions was created separately, and the fact that there is nevertheless some equivalence suggests that there is a certain idea of security that they all refer to. The *rr*, *lr*, and *fq* definitions are in fact all grouped together into what's known as **IND-CPA**, or *message INDistinguishability under Chosen-Plaintext Attack.*

3 Chosen-Plaintext Attacks vs. Chosen-Ciphertext Attacks

Though chosen-plaintext attacks are an important kind of attack to model with respect to cryptographic systems, they are far from the only kind of attack. In particular, chosen-plaintext security is only a property of the encryptor, having nothing to do with the decryption algorithm. It does not allow an adversary to actively modify ciphertext, an ability which it is far from unreasonable to allow to an adversary.

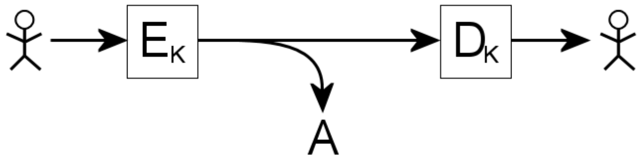

Figure 3: Chosen-Plaintext Attack model

All of the definitions of security we have discussed here are chosen-plaintext security, modelling the communication channel as shown in Figure 3. The adversary A has a way of causing specific plaintexts to be sent, and then can see the resulting ciphertext by eavesdropping on the channel. Note that it is merely a passive observer with regards to the channel.

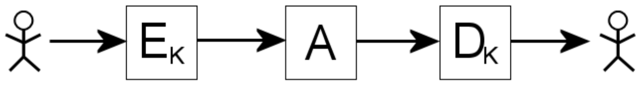

Figure 4: Chosen-Ciphertext Attack model

On the other hand, Figure 4 shows a model that allows active meddling on the part of the adversary. Not only can A choose plaintexts and observe the resulting ciphertexts, but it can modify ciphertexts and look at what the decryptor does with the modified ciphertext. In this model, security requires that learning the decryptions of some ciphertexts doesn't help you decipher other ciphertexts. One formalization of this concept is:

$$Adv^{CCA2}A = \left| Pr[A^{E_K \circ \S_0, D_K} = 1] - Pr[A^{E_K \circ \S_1, D_K} = 1] \right|,$$

extending the idea of left-or-right security to give the adversary the ability to query a decryption oracle, asking "what's the decription $D_K(c) = m$ of this ciphertext $c = E_K(m)$?" Of course, as is, this makes the challenge "what message, m_0 or m_1, yielded this ciphertext?" trivial, because the adversary could just query its decryption oracle to find the answer. So we add a restriction on the operation of A: it cannot query D_K with any c that is returned to it by a query on the $E_K \circ \S_b$ oracle.

We will see later that CCA security can be built out of message authentication techniques.