1 Computationally bounded security

The key insight that enables cryptography is that although an observer can in principle distinguish between the distributions $Enc(K, M)$ and $Enc(K, M')$ for two messages M and M', in practice doing so may take an inordinate amount of time. It seems reasonable to assume that the distinguisher has only limited computational resources at his disposal.

It will be convenient to measure the computational resources of the distinguisher in terms of circuit complexity. A *circuit* with n inputs is a directed acyclic graph with n sources and one sink, where all vertices (called *gates*) except for the sources are labeled by AND, OR, or NOT (NOT gates have fan-in one). A computation proceeds along the circuit in a natural way. The *size* of a circuit is the number of AND and OR gates in it.[1]

You are probably more familiar with computer programs than with circuits, and the two are in fact related. A program that runs in time t and uses s bits of memory (for the program itself plus the data) can be implemented by a circuit of size $O(ts)$. Conversely, a circuit of size s can be evaluated by a program that runs in time $O(s)$ and uses $O(s)$ bits of memory.

It will be convenient to give our circuits also access to randomness. You can think of a randomized circuit as having access to a special set of source gates that produces unbiased and independent random bits.

We can now define the computational variant of message indistinguishability.

Definition 1. A private-key encryption scheme (Enc, Dec) is $(s(k), \varepsilon(k))$-*message indistinguishable* if for every integer k and every pair of messages M, M' of length $m(k)$, and every circuit A of size at most $s(k)$,

$$|\Pr_{K\sim\{0,1\}^k}[A(Enc(K, M)) = 1] - \Pr_{K\sim\{0,1\}^k}[A(Enc(K, M')) = 1]| \leq \varepsilon(k).$$

What are reasonable choices for the parameters $s(k)$ and $\varepsilon(k)$? Let's think about $s(k)$ first. As we discussed, it seems reasonable to assume that the adversary is willing to put at least as much work into breaking the scheme as Alice and Bob are investing in running it. But how much more work? Clearly this depends on the application in question. However, at least as a first approximation, we can adopt the convention in complexity theory that "reasonable resources" means polynomial time (in the security parameter k).[2] So we will work with the following convention:

- The honest parties in a cryptographic protocol are required to run in time polynomial in the security parameter.
- The protocol should remain secure against any adversary whose circuit size is $p(k)$, for *every* polynomial p.

[1] For technical reasons we won't count the NOT gates, although it doesn't make much of a difference.
[2] You can find some justifications for this conventions in these lecture notes.

In the case of encryption, this means that we want our scheme to run in some *fixed* polynomial time – say $O(k)$ or $O(k^2)$ – but it ought to be secure against adversaries that run in *arbitrary* polynomial time $O(k^2)$, $O(k^5)$, and even $O(k^{100})$.

We will assume that all parties have access to randomness.

Let us now turn to the indistinguishability parameter ε. While the circuit size measures the effort (basically, time) we need to put in to break the encryptions, the parameter ε essentially measures the luck of breaking it if we try to do so at random. After all, even the one-time pad can be broken with probability 2^{-m}, if we are lucky enough to guess the message that was encrypted. A good rule of thumb is that luck should be inversely proportional to time: If we can break an encryption with probability ε, then by repeating the attack independently k times the chances of breaking it becomes $1-(1-\varepsilon)^k \approx \varepsilon k$ (when k is much smaller than $1/\varepsilon$). So by doing k times the amount of work, we increase our probability of success by a factor of k.

Since we are aiming to beat an adversary whose complexity is larger than any polynomial, it makes sense to require that the chance of distinguishing the two encryptions is less than any inverse polynomial.[3]

Definition 2. An *efficient, asymptotically message-indistinguishable encryption scheme* is a pair of randomized polynomial-time algorithms (Enc, Dec) and a function $m(k)$ (the message length) so that:

- **Functionality:** For every integer k, every key K of length k and every message m of length $m(k)$, $Dec(K, Enc(K, M)) = M$.
- **Security:** For every polynomial p, every sufficiently large integer k, every (possibly randomized) circuit A of size $p(k)$, and every pair of messages M and M' of length $m(k)$,

$$|\Pr_{K\sim\{0,1\}^k}[A(Enc(K,M))=1] - \Pr_{K\sim\{0,1\}^k}[A(Enc(K,M'))=1]| \leq 1/p(k).$$

2 Semantic security

We now have a working notion of security, but how do we know it is the correct one? One way to convince ourselves is to think of alternative definition of security and see how it relates to the one we just proposed.

Let's put ourselves in Eve's position and see what kinds of things we can hope to learn by observing the communication between Alice and Bob. In general, Eve may have some prior knowledge about the message; this knowledge can be represented as a (prior) probability distribution on the messages between Alice and Bob. Ideally Eve's objective is to learn what the actual encrypted message is. But even short of this, obtaining any kind of information about the message – say, how many times the word `eve` appears in the message – may be meaningful for her.

In general, we have no control over the distribution on messages, so we might as well let Eve *choose* the distribution. So we have the following model of adversary: After the key K is distributed to Alice and Bob, Eve chooses a distribution $\mathcal{D}$ on messages and some function f on messages (the property of the message she is interested in). Then Alice chooses a message M from $\mathcal{D}$ and sends $Enc(K, M)$ to Bob. The objective of Alice is to "learn" $f(M)$ only by observing $Enc(K, M)$.

[3] A function $\varepsilon(k)$ such that for every polynomial p and every sufficiently large k, $\varepsilon(k) < 1/p(k)$ is called *negligible*.

But what does it mean for Alice to learn $f(M)$? Actually, let's ask the opposite question: What does it mean for Alice to remain ignorant of $f(M)$? It means that observing the interaction between Alice and Bob did not help Eve at all in finding out the value $f(M)$; in other words, whatever Eve could have learned about M by observing $Enc(K, M)$, she could have also learned by not observing anything at all. Let's try to formalize this; we want to say that

> When given $Enc(K, M)$ as an input, Eve is just as likely to output $f(M)$ as when she is given no input at all.

Let A be the algorithm run by Eve when she is observing the interaction between Alice and Bob, and A' be Eve's guess for the value $f(M)$ when she is given no input. What we want to say is that the probability that $A(Enc(K, M)) = f(M)$ is very close to the probability that $A' = f(M)$, where A' is some (randomized) circuit that takes no input.

Definition 3. An encryption scheme (Enc, Dec) is (s, s', ε) *semantically secure* if for every distribution $\mathcal{D}$ over messages, every function f, and every circuit A of size s, there is a circuit A' of size s' such that

$$|\Pr_{M\sim\mathcal{D},K,A}[A(Enc(K, M)) = f(M)] - \Pr_{A'}[A' = f(M)]| \leq \varepsilon.$$

We can now prove that message indistinguishability is essentially equivalent to semantic security:

Claim 4. *If (Enc, Dec) is (s, ∞, ε) semantically secure, then it is $(s, 2\varepsilon)$ message indistinguishable.*

Proof. The proof is by contrapositive: Suppose (Enc, Dec) is not $(s, 2\varepsilon)$ message indistinguishable, so there exists a pair of messages M_0 and M_1 and a circuit A of size s so that

$$\Pr[A(Enc(K, M_1)) = 1] - \Pr[A(Enc(K, M_0)) = 1] > 2\varepsilon.$$

(Without loss of generality, we can remove the absolute value by adding a NOT gate to the output of A if necessary.) We can rewrite this as

$$\frac{1}{2}(\Pr[A(Enc(K, M_1)) = 1] + \Pr[A(Enc(K, M_0)) = 0]) > \frac{1}{2} + \varepsilon.$$

Let $\mathcal{D}$ be the uniform distribution over messages M_0 and M_1 and f be the function $f(M_0) = 0$, $f(M_1) = 1$. Then

$$\Pr_{M\sim\mathcal{D}}[A(Enc(K, M)) = f(M)] = \Pr_{b\sim\{0,1\}}[A(Enc(K, M_b)) = b] > \frac{1}{2} + \varepsilon.$$

On the other hand, since $f(M)$ is a random bit when M is chosen from $\mathcal{D}$, $\Pr[A' = f(M)]$ is at most $1/2$ for every A'. □

Claim 5. *If (Enc, Dec) is (s, ε) message indistinguishable and Enc has circuit complexity t, then it is $(s, s' + t, \varepsilon)$ semantically secure.*

Proof. Assume (Enc, Dec) is (s, ε) message indistinguishable. Let M and M' be any pair of messages. Then by indistinguishability for every A of size at most $s - 2$

$$|\Pr_{A,K}[A(Enc(K, M)) = f(M)] - \Pr_{A,K}[A(Enc(K, M')) = f(M)]| \leq \varepsilon$$

(If this is not the case, consider the circuit $B(z)$ that outputs 1 if $z = f(M)$ and 0 if not. Then B can be implemented in size s and B distinguishes $Enc(K, M)$ from $Enc(K, M')$.) Fix M' to be an arbitrary message. By averaging, for every distribution $\mathcal{D}$ over messages M we have

$$|\Pr_{M\sim\mathcal{D},A,K}[A(Enc(K,M)) = f(M)] - \Pr_{M\sim\mathcal{D},A,K}[A(Enc(K,M')) = f(M)]| \leq \varepsilon.$$

Let A' be the following randomized circuit: A' takes no input, it chooses a random string K of length k and outputs $A(Enc(K, M'))$. Then

$$|\Pr_{M\sim\mathcal{D},A,K}[A(Enc(K,M)) = f(M)] - \Pr_{A'}[A' = f(M)]| \leq \varepsilon. \qquad \square$$

3 Pseudorandom generators

How do we go about achieving computational security (with short keys and long messages)? To get some idea, let's go back to perfect security. This we could achieve easily with the one-time pad; we simply use the key as a mask that allows Alice and Bob to communicate the message, but hides the message completely from Eve.

The mask managed to hide the message because from Eve's perspective, it looked completely random. Now we want a shorter key, so we cannot use the whole key as a mask. But can we maybe extend the key in some way – compute some function of it – so that it still looks random to Eve? From our discussion last time we know perfect randomness is unattainable, but we can aim to make the message appear random to every adversary with reasonably limited computational resources.

Here is the scheme we have in mind. Alice wants to send Bob an m-bit message M, but they only have a secret key K of length $k < m$. Suppose that Alice and Bob (and also Eve) have access to some public function $G\colon \{0,1\}^k \to \{0,1\}^m$ so that when G is evaluated at a random input, its output "looks" random. Then, imitating the one-time pad, Alice encrypts M using the algorithm $Enc(K, M) = M + G(K)$, and Bob decrypts using $Dec(K, C) = C + G(K)$. Clearly this is a functional private-key encryption scheme; but how secure is it?

The security of the scheme depends on the choice of the function G. Here is the "random-looking" property that we want G to have:

Definition 6. A function $G\colon \{0,1\}^k \to \{0,1\}^m$ (where $m > k$) is a *pseudorandom generator* against size s and bias ε if for every circuit D (the distinguisher) of size at most s,

$$|\Pr_{x\sim\{0,1\}^k}[D(G(x)) = 1] - \Pr_{y\sim\{0,1\}^m}[D(y) = 1]| \leq \varepsilon.$$

The input x is called the *seed* of the pseudorandom generator. It will also be useful to have an asymptotic version of this definition: A family of functions $G_k\colon \{0,1\}^k \to \{0,1\}^{m(k)}$ (where $m(k) > k$ for all k) is called a (asymptotically secure) *pseudorandom generator* if

1. On input x of length k, $G_k(x)$ is computable in time polynomial in k,[4] and
2. For every polynomial p and every sufficiently large k, G_k is pseudorandom against size $p(k)$ and bias $1/p(k)$.

[4] Certain definitions require the time to be polynomial in $m(k)$, but here we will only consider settings where $m(k)$ is itself polynomial in k, so it won't make a difference.

Is the circuit D deterministic or randomized? It turns out that it doesn't make a difference for the definition. A randomized D is certainly at least as powerful as a deterministic one, but in fact the randomness can be removed by fixing it to the value for which the difference in distinguishing probabilities is maximized.

Which functions are pseudorandom generators? The answer to this question is not known, and we cannot even say with certainty that pseudorandom generators exist. However, pseudorandom generators are *necessary* to do most tasks in cryptography, including the construction of (asymptotically) message-indistinguishable encryption schemes, so it is widely believed that they do exist.

However, the existence of pseudorandom generators is thought very difficult to prove because it *implies* that P does not equal NP. The conclusion sounds reasonable, so what is the problem? Well, it means that proving pseudorandom generators exist in particular *requires* us to prove that P does not equal NP first, which is one of the greatest open problems in computer science and mathematics.[5] If you are familiar with P and NP (and a bit of circuit complexity), you shouldn't have much trouble proving the following claim:

Claim 7. *If* P = NP*, then no asymptotically pseudorandom generator exists.*

So let us ask the question in a different way: Which functions are *believed* to be pseudorandom generators? In the answer to this question, theory and practice diverge significantly. Practitioners usually value constructions that have very fast implementations and have not been broken (although sometimes these constructions *do* get broken after a decade or so). Theoreticians try to construct pseudorandom generators whose security follows from some well studied assumption.

Some constructions of pseudorandom generators For a pseudorandom generator used in practice, you can take a look at this description of RC4. This is a very efficient scheme that takes a seed of 4096 bits. Its output length is unspecified; the implementation allows us to construct more and more pseudorandom bits the longer we run it. However, the security of this construction is not understood – that is, we do not really know for which values of s and ε it is pseudorandom. The original construction of RC4 has been broken; for example it is known that a distinguisher that simply outputs the second bit of the output succeeds with probability $\varepsilon = 1/256$. There are variants of it that are used today.

Here is a theory-inspired construction with $m = 2k$ (other values of m are also possible). The function G is described by m sequences of five numbers each $(a_{i1}, a_{i2}, a_{i3}, a_{i4}, a_{i5})$, $1 \le i \le m$, where each number a_{ij} is between 1 and k. We compute the output $G(x)$ bit by bit: The ith bit of $G(x)$ is defined as

$$(i\text{th bit of } G(x)) = x[a_{i1}] + x[a_{i2}] + x[a_{i3}] + x[a_{i4}]x[a_{i5}]$$

where all the operations are modulo 2, and $x[j]$ is the jth bit of x. To specify this function, we have to say what the numbers a_{ij} are. It is conjectured that if they are chosen at random (publicly, in a way known to Alice, Bob, and Eve), for most choices the function $G(x)$ is a pseudorandom generator against size $p(k)$ and bias $1/p(k)$ for every polynomial p. There is some theoretical support, for this conjecture, but not much is known for specific values of k, e.g. $k = 4096$.

[5] If we somehow manage to prove P ≠ NP, will it immediately follow that pseudorandom generators exist? Even this is not known – although it is believed to be a much easier question than P ≠ NP.

4 Message indistinguishable encryption

Let us now justify the security of the proposed encryption scheme. Let $G\colon \{0,1\}^k \to \{0,1\}^m$ be a pseudorandom generator and consider the following private-key encryption scheme:

$$Enc(K, M) = G(K) + M \qquad Dec(K, C) = G(K) + C$$

where $M \in \{0,1\}^m$ and $K \in \{0,1\}^k$.

Claim 8. *If G is a pseudorandom generator against size s and bias ε, then (Enc, Dec) is $(s, 2\varepsilon)$-message indistinguishable.*

Proof. For contradiction, assume not. Then there exists a circuit of size s and messages M and M' such that

$$|\Pr_{K,A}[A(G(K) + M) = 1] - \Pr_{K,A}[A(G(K) + M') = 1]| > 2\varepsilon$$

Let Y denote a random string of length m. By the pseudorandomness of G, we have that

$$|\Pr_{K,A}[A(G(K) + M) = 1] - \Pr_{Y,A}[A(Y + M') = 1]| \leq \varepsilon.$$

(If this was not the case, the circuit $D(x) = A(x + M)$ would be a distinguisher for A of size s.) By the same argument, we have

$$|\Pr_{K,A}[A(G(K) + M') = 1] - \Pr_{Y,A}[A(Y + M') = 1]| \leq \varepsilon.$$

By the triangle inequality, it must be the case that

$$|\Pr_{Y,A}[A(Y + M') = 1] - \Pr_{Y,A}[A(Y + M) = 1]| > 0.$$

This is impossible, because the random variables $Y + M$ and $Y + M'$ are identically distributed. □

Notice what happened in this proof: We *reduced* the task of arguing the computational security of (Enc, Dec) to the task of arguing its perfect security when $G(K)$ is replaced by a completely random string Y. This is characteristic of many proofs in cryptography. First we come up with an ideal notion (perfect security), which for whatever reason we cannot achieve. Then we relax it to a feasible notion (computationally bounded security), and construct a scheme by analogy (i.e., each instance of the one-time pad is replaced by a pseudorandom output). To argue its security, we essentially prove that the cost of this replacement is small (in this case, we pay a price of 2ε, where ε is the bias of the pseudorandom generator).

So now we know that we can get secure encryption from any pseudorandom generator. However, the message length of the encryption is dictated by the output length of the pseudorandom generator. If the output length is $m = 2k$, it means we can only encrypt messages whose length is double the key length. What about longer ones? We will next see a construction that allows us to extend the length of the pseudorandom generator essentially as much as we want, so that we can use this scheme to encrypt arbitrarily long messages.

Before we go on, there is one last issue to clear out. We just saw how to construct a message indistinguishable encryption scheme from any pseudorandom generator. But we are not really 100% sure that pseudorandom generators exist at all. So does message indistinguishable encryption *require* pseudorandom generators? The answer is yes, although we won't be able to prove it in this class.

5 Extending the length of pseudorandom generators

We now show how, given a pseudorandom generator G that takes k bits of seed to $k+1$ bits of output, we can get another one G' that takes k bits of seed to $k+d$ bits of output. The complexity of G' will grow with t, and the security of G' will be somewhat worse than that of G, but not by much.

We obtain G' by applying G iteratively, where at each iteration we get one extra bit of output. Suppose we start with a random seed $X_0 \in \{0,1\}^k$. After d iterations, we will have a pseudorandom output $X_d \in \{0,1\}^{k+d}$. To go from iteration d to iteration $d+1$, we define

$$X_{d+1} = G_d(X_0) = (G(\text{first } k \text{ bits of } X_d), \text{last } d \text{ bits of } X_d).$$

Clearly if G can be computed by a circuit of size t, then G_d can be computed by a circuit of size dt. But how secure is it?

Theorem 9. *If G can be computed by a circuit of size t and G is pseudorandom against size s and bias ε, then G_d is pseudorandom against size $s-(d-1)t$ and bias εd.*

Since the construction of G_d is recursive, it makes sense to try a proof by induction. Certainly when $d=1$ the theorem is true. Now suppose that we have proved G_d is pseudorandom against size $s-(d-1)t$ and bias $d\varepsilon$. We want to argue that G_{d+1} is then pseudorandom against size $s-dt$ and bias $(d+1)\varepsilon$.

Proof. For contradiction, let us suppose that G_{d+1} is not pseudorandom. So there exists a circuit D of size $s-dt$ such that

$$|\Pr[D(G_{d+1}(X_0)) = 1] - \Pr[D(Y) = 1]| > (d+1)\varepsilon.$$

Here, Y is a truly random string of length $k+d+1$. Now recall that G_{d+1} was obtained by running G on the first k bits of X_d (the output of G_d) and copying the last k bits. Let $X_d = X_L X_R$, where X_L are the first k bits and X_R are the last d. Also let $Y = Y_L Y_R$ where Y_L are the first $k+1$ bits and Y_R are the last d bits. Then

$$|\Pr[D(G(X_L), X_R) = 1] - \Pr[D(Y_L, Y_R) = 1]| > (d+1)\varepsilon.$$

Now let Z be a truly random string of length k, independent of everything else. It follows that at least one of these two inequalities must hold:

$$|\Pr[D(G(X_L), X_R) = 1] - \Pr[D(G(Z), Y_R) = 1]| > d\varepsilon \qquad \text{or}$$
$$|\Pr[D(G(Z), Y_R) = 1] - \Pr[D(Y_L, Y_R) = 1]| > \varepsilon.$$

Suppose the first inequality holds. Then we can distinguish X from a truly random string as follows:

> Circuit D': On input u, write $u = u_L u_R$ (the first k and last d bits) and output $D(G(u_L), u_R)$.

This circuit D' has size $s-(d-1)t$ and by the first inequality

$$|\Pr[D(G(X_L), X_R) = 1] - \Pr[D(G(Z), Y_R) = 1]| > d\varepsilon$$

contradicting our inductive assumption. So the second inequality must hold. But then we can distinguish the output of G from random by the following circuit: On input u, choose a random string Y_R of length d and output $D(u, Y)$. By the second inequality, this is a circuit of size $s - dt + d \leq s$ that distinguishes the output of G from random with advantage ε, contradicting the pseudorandomness of G. □

If you are a bit confused about the induction, there is nothing special about it. You can convert it into an iterative argument, which will tell you that when you apply each iteration, you have to pay an ε in distinguishing probability and a t in distinguishing circuit size.

But within the inductive argument, there is one interesting thing that happens. We wanted to argue that two distributions – the output of G_{d+1} and the uniform distribution on $k+d+1$ bits – are indistinguishable. To do so directly is difficult. So what we did was we introduced an intermediate distribution, which describes what happens when the $(d+1)$st iteration is applied not to the output of G_d but to a truly random string. This made sense because we already knew by the inductive hypothesis that the output of G_d looks pseudorandom. So by comparing the real scenario (G applied to the left part of the output of G_d) to an ideal scenario (G applied to the left part of a truly random string) we managed to derive our conclusion.

In the course of this we used the following fact: If two distributions X and Z are distinguishable, then for any distribution Y either X and Y are distinguishable or Y and Z are distinguishable. This is a very common type of argument in cryptography called *the hybrid argument*, because the distribution Y is usually some sort of hybrid of X and Z. Sometimes it helps to introduce several intermediate distributions, which allows us to move from X to Z "smoothly". We will see an example of this in the next lecture.