CSCI 5440: Cryptography **Lecture 2**
The Chinese University of Hong Kong, Fall 2020 14 and 15 September 2020

In Lecture 1 we saw that statistically secure encryption cannot exist unless the key is as long as the message. However, even though the adversary Eve can in principle distinguish between the distributions $Enc(K, M)$ and $Enc(K, M')$ for two messages M and M', we did not say how she would carry out her attack with feasible computing power within a reasonable amount of time.

The main premise of cryptography is that a realistic adversary has large but *bounded* computational resources at his disposal. This turns out to make a big difference.

1 Computationally bounded security

Two important measures of computational resources are program size and running time. These measures are not so easy to define and calculate precisely as they depend on the specifics of programming languages and computer architectures. In the theory of cryptography the distinguisher is often modelled by an imaginary device called a circuit.

A *Boolean circuit* is a directed acyclic graph whose sources are labeled by input variables $x_1, \dots, x_n$ or the constants $0, 1$, whose sinks are labeled by output variables $y_1, \dots, y_m$, and whose internal nodes are labeled by AND, OR, or NOT gates. The circuit computes the function $f(x_1, \dots, x_n) = (y_1, \dots, y_m)$ by plugging in the inputs at the sources, calculating the intermediate values at every gate in order, and reporting the values at the sinks. The *size* of a circuit is the number of AND and OR gates in it. This is our complexity measure.

Circuits can also perform randomized computations. The non-constant source nodes of a randomized circuit are divided into two types: The input bits $x_1, \dots, x_n$ and the random seed $r_1, \dots, r_k$. The latter are instantiated with uniform random bits. The output then becomes a probabilistic function of the input x. A randomized circuit that takes no input is called a *sampler.* A circuit that takes no random bits is called *deterministic.*

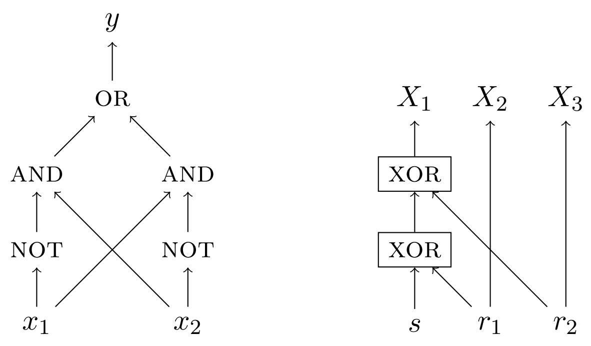

Figure 1: Examples of circuits. The circuit on the left computes the XOR function $y = x_1 + x_2$ and has size 3. The circuit on the right implements the share sampling procedure from Section 1 of Lecture 1: When r_1 and r_2 are random bits, the output $X_1X_2X_3$ is a uniform 3-bit string conditioned on $X_1 + X_2 + X_3 = s$.

Computer programs and circuits are closely related: A program that runs in time t and uses s bits of memory 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. In this sense, the size of a circuit simultaneously captures a program's size and its running time.

Now that we have a precise notion of computation, we can define the desired relaxation of statistical closeness that enables most of modern cryptography.

Definition 1. Random variables X and X' taking values in $\{0,1\}^k$ are (s,ε)-*computationally indistinguishable* (in short, (s,ε)-indistinguishable) if for every circuit $D\colon \{0,1\}^k \to \{0,1\}$ of size at most s, $|\Pr[D(X)=1] - \Pr[D(X')=1]| \le \varepsilon$.[1]

Notice that ε-statistical closeness is the same as (∞,ε)-indistinguishability, that is computational indistinguishablity for circuits of unrestricted size. In fact, the same is true for $(2^k,\varepsilon)$-indistinguishability because circuits of size 2^k can compute all possible functions with k inputs and one output. However, as soon as s is $o(2^k/k)$ there exist functions that cannot be computed by any circuit of size s, so computational indistinguishability and statistical closeness are no longer the same.

Definition 2. A private-key encryption scheme (Enc, Dec) is (s,ε)-*message indistinguishable* if for every pair of messages M, M' of length m, the random variables $Enc(K,M)$ and $Enc(K,M')$ are (s,ε)-indistinguishable, where K is uniformly random.

What are reasonable choices for the parameters s and ε? Let's think about s first. A basic principle of cryptographic design is that breaking a system should take a lot more effort than operating it, so the complexity s of Eve should be much larger than the ones of Alice and Bob. In theoretical cryptography, it is common to think of k as a tuneable *security parameter*, to require for Alice and Bob to operate in time polynomial in k (which is the first cut for efficiency in algorithm design), and to think of Eve's complexity s as growing exponentially in k (e.g., $s = 2^{k/3}$ or $s = 2^{\sqrt{k}}$).

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 $2^{-m} = 2^{-k}$ if Eve is lucky enough to guess the message that was encrypted. A good rule of thumb is that luck should be inversely proportional to time or size, i.e, $\varepsilon \approx 1/s$: If Eve can break an encryption with probability ε, then by repeating the attack independently t times the chances of breaking it becomes $1-(1-\varepsilon)^t$, which is about εt when t is smaller than $1/\varepsilon$.

To get some practice with this notion, let us also give the simulation-based definition of security and prove that the two are equivalent.

Definition 3. A private-key encryption scheme (Enc, Dec) is (s,ε)-*message simulatable* by size t if there exists a sampler Sim of size t for which the output of Sim is (s,ε)-indistinguishable from the output of $Enc(K,M)$ for every M and uniformly random K.

Claim 4. *If (Enc, Dec) is (s,ε)-message simulatable by any size then it is $(s,2\varepsilon)$-message indistinguishable.*

Proof. We argue by contrapositive. Suppose $Enc(K,M)$ and $Enc(K,M')$ are $(s,2\varepsilon)$-distinguishable, namely

$$|\Pr[D(Enc(K,M))=1] - \Pr[D(Enc(K,M'))=1]| > 2\varepsilon.$$

for some circuit D of size s. By the triangle inequality, for every distribution Sim

$$|\Pr[D(Enc(K,M))=1] - \Pr[D(Sim)=1]| > \varepsilon \text{ or } |\Pr[D(Enc(K,M'))=1] - \Pr[D(Sim)=1]| > \varepsilon.$$

In words, D distinguishes the output of any potential simulator from the encryption of one of the two messages. So (Enc, Dec) is not (s,ε)-message simulatable. $\square$

[1] Without loss of generality, the circuit D can be assumed deterministic: If a randomized distinguisher exists, so does a deterministic one of the same size.

Claim 5. *If (Enc, Dec) is (s, ε)-message indistinguishable then it is (s, ε)-message simulatable by size t, where t is the circuit size of Enc.*

Proof. Let Sim be sampler which on randomness K outputs $Enc(K, M_0)$ for any fixed message M_0, e.g. $M_0 = 0^m$. The circuit size of Sim is the same as the circuit size of Enc since it was derived from the latter by fixing some inputs. The distribution Sim, which is identical to $Enc(K, M_0)$, is then (s, ε)-indistinguishable from $Enc(K, M)$ for every possible message M. □

2 Pseudorandom generators

The one-time pad masks the message M by a key Y which is as long as M itself:

$$Enc(Y, M) = M + Y \qquad Dec(Y, C) = C + Y.$$

One possible strategy for shortening the key is to replace the uniformly random Y by some public, deterministic function G of a much shorter random key K. The ciphertexts $Enc(Y, M) = M + G(K)$ would no longer be uniformly random, but they could plausibly still be computationally indistinguishable from a uniform random variable. A function G that has this property is called a pseudorandom generator.

Definition 6. Let $m > k$. A function $G\colon \{0, 1\}^k \to \{0, 1\}^m$ is an (s, ε)-*pseudorandom generator* if the random variable $G(K)$, where $K \sim \{0, 1\}^k$ is uniformly random is (s, ε)-indistinguishable from a uniformly random m-bit string Y.

It is not known if pseudorandom generators exist. The existence of pseudorandom generators is closely related to the famous "P versus NP" question. If P were to equal NP then pseudorandom generators would not exist. It is also known that if there are no pseudorandom generators, then most of modern cryptography, including encryption with short keys, is impossible. Shouldn't this state of affairs keep cryptographers awake at night?

Cryptographers are indeed rumored to seldom sleep well,[2] but usually not for this reason. While there are no provably secure pseudorandom generators out there, several different types of proposals have been studied extensively and are believed to be quite secure. In applied cryptography, proposals are closely scrutinized by cryptanalysts who compete in coming up with faster and better distinguishers. If nobody can come up with an attack that significantly outperforms brute-force search for the key or random guessing, more and more people become confident that the proposal is secure. Practical constructions must be highly efficient (sometimes on a specific computer architecture) in addition to being extremely secure.

A famous theorem in cryptography says that pseudorandom generators can be obtained from a more believable primitive called a *one-way function*. One-way functions are functions that are easy to compute but hard to invert. One very specific strategy for breaking a candidate pseudorandom function is by trying to invert it, namely designing a procedure that recovers the key K from the value $G(K)$ with reasonable probability. If there is an efficient way to do this then $G(K)$ cannot be pseudorandom: There are at most 2^k values $y \in \{0, 1\}^m$ that are in the image of G, so the inverter can succeed with probability at most $2^k/2^m \leq 1/2$ when its input is a uniformly random Y. Any inversion algorithm can therefore be used to distinguish samples of the form $G(K)$ (in which case the algorithm usually finds an inverse) from uniformly random samples Y (in which case the inverse is unlikely to even exist). The theorem says that inversion attacks are the only attacks that we need to worry about (assuming that the pseudorandom generator is designed properly): If we can

[2]Quote attributed to Silvio Micali by Joe Kilian, *Founding cryptography on oblivious transfer*, STOC 1988

come up with a function f that is easy to compute but hard to invert, then there is some other function G that is a pseudorandom generator.

In a couple of weeks we will give some specific examples of pseudorandom generators whose use extends well beyond secure encryption.

Let us now prove that pseudorandom generators in fact give secure encryption.

Claim 7. *If $G\colon \{0,1\}^k \to \{0,1\}^m$ is an (s,ε)-pseudorandom generator then the following encryption scheme is (s,ε)-message simulatable by size zero:*

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

Proof. The simulator Sim outputs a uniformly random m-bit string Y. We prove the contrapositive. Suppose that there exists a distinguisher D of size s such that

$$|\Pr[D(M+G(K)) = 1] - \Pr[D(Y) = 1]| > \varepsilon$$

for some message M. Let D' be the distinguisher that on input x outputs $D(x+M)$. D' has the same size as D because it only modifies D by applying NOT gates to some of its inputs. Then $D(M+G(K)) = D'(G(K))$ and $D(Y) = D'(Y+M)$. We can therefore rewrite the above inequality in the form

$$|\Pr[D'(G(K)) = 1] - \Pr[D'(Y+M) = 1]| > \varepsilon.$$

For any fixed M, the random variable $Y+M$ is also uniformly distributed, so D' distinguishes $G(K)$ from a uniformly random string with advantage more than ε. Therefore G is not an (s,ε)-pseudorandom generator. □

3 Extending the length of pseudorandom generators

Claim 7 shows that we can get secure encryption from any pseudorandom generator. The message length of the encryption is determined by the output length of the pseudorandom generator. If, say, $k = 500$ and $m = 2000$ this only allows us to encrypt 2000-bit messages with a 500-bit long secret key. What about longer messages? We now show how to take any pseudorandom generator, even one that stretches the key by one bit, and use it to build one with essentially unbounded output.

The *stretch* of $G\colon \{0,1\}^k \to \{0,1\}^m$ is $m-k$. To improve stretch we bootstrap. Let

$$G'(K) = (\text{first } m-k \text{ bits of } G(K), G(\text{last } k \text{ bits of } G(K))) = (X_L, G(X_R))$$

See figure 2 (a).

Theorem 8. *If G is an (s,ε)-pseudorandom generator of stretch $m-k$ then G' is an $(s-t, 2\varepsilon)$-pseudorandom generator of stretch $2(m-k)$, where t is the circuit size of G.*

Proof. We prove the contrapositive. Let us suppose that G' is not (s',ε')-pseudorandom for values of s' and ε' that will turn out to match those in the statement of the theorem. So there exists a circuit D of size s' such that

$$|\Pr[D(G'(K)) = 1] - \Pr[D(Y) = 1]| > \varepsilon',$$

where Y is a uniformly random string. We split Y into its first $m-k$ bits Y_L and its last k bits Y_R as in figure 2 (c). In this notation, we can rewrite the inequality as

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

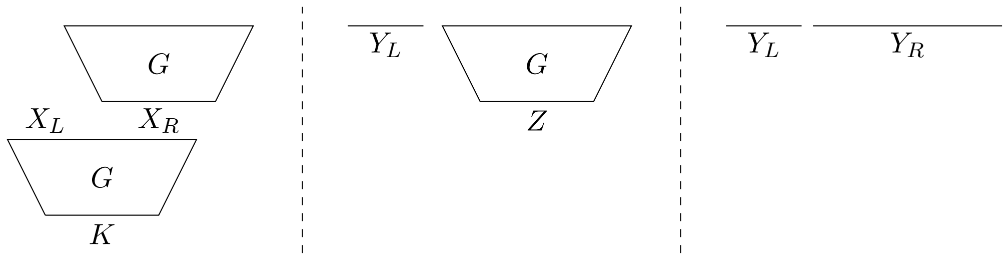

Figure 2: Length extension. (a) the construction; (b) the hybrid distribution; (c) the uniform distribution

We now consider the *hybrid* random variable $(Y_L, G(Z))$ where Z is uniformly random and independent of everything (see figure 2 (b)). Then one of these two inequalities must hold:

$$|\Pr[D(X_L, G(X_R)) = 1] - \Pr[D(Y_L, G(Z)) = 1]| > \varepsilon'/2 \qquad \text{or}$$
$$|\Pr[D(Y_L, G(Z)) = 1] - \Pr[D(Y_L, Y_R) = 1]| > \varepsilon'/2.$$

Let's deal with the second case first. If we think of D as a randomized circuit that takes Y_L as its randomness, then D distinguishes $G(Z)$ from Y_R, so G cannot be $(s', \varepsilon'/2)$-pseudorandom.[3]

Now for the first case. Remember that (X_L, X_R) is the output of $G(K)$ and (Y_L, Z) is a uniformly random string. So D effectively distinguishes $G(K)$ from a uniformly random string. More precisely,

$$D(X_L, G(X_R)) = D'(G(K)) \quad \text{and} \quad D(Y_L, Y_R) = D'(Y_L, Z)$$

where D' is the circuit

$$D'(u) = D(\text{first } m - k \text{ bits of } u, G(\text{last } k \text{ bits of } u)).$$

This circuit D' has size $s' + t$ and distinguishes the output of G from a random string with advantage $\varepsilon'/2$, so G is not $(s' + t, \varepsilon')$-pseudorandom.

In conclusion, we showed that if G' is not (s', ε') pseudorandom, then G is not $(s', \varepsilon'/2)$-pseudorandom or it is not $(s' + t, \varepsilon'/2)$-pseudorandom. The second alternative subsumes the first. If we set $s = s' + t$ and $\varepsilon = \varepsilon'/2$ we obtain the theorem. □

In this proof, it would have been difficult to argue the indistinguishability of $G'(K)$ and Y directly. With the intermediate random variable $(Y_L, G(Z))$ in hand we broke up the problem into two manageable parts. This choice shouldn't be mysterious. After all, the reason we believe $G'(K)$ is pseudorandom is because the first application of G produces an output that looks random, so the second application is also secure. The proof makes this reasoning mathematically precise.

Notice also that the security parameters of G' are worse than those of G. What does this really mean? Regarding the advantage ε, our working assumption is that it was "exponentially" small to begin with, so doubling it does not have such significant effect. As for the distinguisher size s, it was "exponentially" large to begin with, while t (the size of the generator) was "polynomial". So the value of $s - t$ is extremely close to the value of s in intended application.

Another question that you may ask is why does G' end up being "less secure" than G? Since G' is obtained by applying G multiple times, maybe it should be more secure. This may indeed end up being the case for specific choices of G. In the theorem, however, G can be instantiated by any (s, ε)-pseudorandom generators, and in fact there are choices of G for which the distinguishing

[3]If you want to turn D into a deterministic distinguisher, observe that there must always be *some* value $Y_L = y_L$ for which $|\Pr[D(y_L, G(Z)) = 1] - \Pr[D(y_L, Y_R) = 1]| > \varepsilon'/2$. By fixing the first s inputs of D to this value we obtain a deterministic distinguisher of the same size.

advantage parameter of G' is about 2ε.[4] In conclusion, Theorem 8 provides *worst-case* guarantees on security, which might or might not be attained in specific instantiations.

Theorem 8 can be applied recursively, leading to pseudorandom generators of arbitrary stretch with security parameters as in this Corollary.

Corollary 9. *Let G' be the generator obtained by bootstrapping G on its last k output bits ℓ times. If G is (s, ε)-pseudorandom then G' is $(s - (\ell - 1)t, \ell\varepsilon)$-pseudorandom.*

One nice feature of this construction is that the output of G' is generated in a streaming fashion. Even if Alice does not the length of her message ahead of time she can encrypt using G', stretching more bits on-the-fly.

4 Multiple encryptions

We now have a scheme by which Alice can securely encrypt any message. What if she wants to encrypt two messages? One possibility is to treat both messages as parts of one very long message. This requires Alice to keep state (i.e. know how much of the pseudorandom output has been used) which may not always be practical or realistic. Maybe there are multiple Alices that share the same secret key and want to encrypt messages without coordination.

Using the encyption scheme from Claim 7 twice is a bad idea. If the two messages M_1 and M_2 happen to be identical then their encryptions $Enc(K, M_1)$ and $Enc(K, M_2)$ will look exactly the same and Eve will be able to deduce this. This is a problem for any deterministic encryption scheme. In order to be secure, *encryptions must be randomized.* This basic insight earned Shafi Goldwasser and Silvio Micali the 2012 A. M. Turing award.[5]

The security notion that we are after should say that Eve does not find out any information even if she sees encryptions of multiple messages. Here is the simulation-based definition.

Definition 10. (Enc, Dec) is (s, ε)-message simulatable for two messages (by size t) if there exists a sampler Sim (of size t) such that for any two messages $M_1, M_2 \in \{0,1\}^m$, $(Enc(K, M_1), Enc(K, M_2))$ and the output of Sim are (s, ε)-indistinguishable.

Here is one randomized proposal that will get us started, although it will turn out to be unsatisfactory. The main component is again a pseudorandom generator $G\colon \{0,1\}^k \to \{0,1\}^n$, but its output length n is now much longer than the message length m. To encrypt a message $M \in \{0,1\}^m$, Alice chooses a random number R between 0 and $n - m$ and encrypts M by the string

$$Enc(K, M) = (R, M_1 + G(K)_{R+1}, M_2 + G(K)_{R+2}, \dots, M_m + G(K)_{R+m}),$$

where $G(K)_i$ is the i-th output bit of G. To decrypt, Bob first reads R from the ciphertext and then XORs the remaining part with the relevant portion of $G(K)$.

What happens when Alice encrypts two messages M_1 and M_2? The indices R_1 and R_2 generated in the two runs of Enc are independent. As long as the intervals $[R_1 + 1, R_1 + m]$ and $[R_2 + 1, R_2 + m]$ do not overlap, $Enc(K, M_1)$ and $Enc(K, M_2)$ will use different portions of G's output and they will look indistinguishable from two *independent* random strings, revealing no information to Eve. The probability that the above two intervals do intersect is about m/n. Formalizing this argument along the lines of the proof of Claim 7 would give that (Enc, Dec) is $(s, \varepsilon + m/n)$-message simulatable for two messages, assuming G is (s, ε)-pseudorandom. Is this satisfactory?

[4]In contrast, it is not known whether the size of the best distinguisher must also drop from s to $s - t$.

[5]It was really awarded for their many influential contributions to cryptography, some of which we will see later.

The answer depends on our choice of n: The larger n, the more secure the scheme. However, the larger n is, the more work Alice and Bob have to put into calculating $G(K)$. Specifically, if they use the length-extension procedure from Corollary 9 the amount of work to compute $G(K)$ becomes linear in n. Since Alice and Bob have only "polynomial" resources, the ratio m/n will always be inverse-polynomial in the security parameter k, which is far short of our exponential security standard.

Perhaps our analysis was too pessimistic and the actual distinguishing advantage of Eve is never as large as m/n. Unfortunately this is not the case. As an exercise, you can show that encryptions for two messages can be distinguished in size $O(m)$ with advantage $\Omega(1/n)$.

To make this idea feasible, it looks like we need a "pseudorandom generator" with exponentially long output. Such objects are called pseudorandom functions and will be the topic of the next lecture.