1 Symmetric Encryption

1.1 Recap

Recall that in the last lecture we discussed cryptographic security in the symmetric setting. The typical arrangement is that Alice and Bob share a common key that they use for encryption and decryption. This shared secret creates an distinction between the communicating parties (Alice and Bob) and the adversarial eavesdropper (Eve). The goal in this setting is to ensure that without the shared secret key, Eve will not be able to learn anything about the messages being sent between Alice and Bob (except, of course, the fact that Alice and Bob are communicating via messages from a known space).

Perfect secrecy (equivalently, Shannon secrecy) is typically impractical in the symmetric setting because it requires that the shared key be at least as large as any message sent, and each key may only be used once. For practicality, we relaxed our notion of perfect secrecy to allow for a computationally bounded adversary, and we arrived at the notion of *indistinguishability under (adaptive) chosen plaintext attack* (IND-CPA). In this security notion, we allow a computationally bounded adversary access to an encryption oracle $\mathsf{Enc}_k(\cdot)$ and a challenge oracle $\mathsf{C}_k^b(\cdot,\cdot)$. The adversary may make (a polynomially bounded number of) calls to the encryption oracle, adapting its queries to the answers it has already received. The adversary may make a single call to the challenge oracle, passing two messages of his choice. The challenge oracle encrypts one of the two messages (determined by the bit b) and returns the ciphertext to the adversary. The adversary examines the returned ciphertext and attempts to determine which of the two messages was encrypted. If there does not exist *any* efficient adversary that can tell which message was encrypted (with better than negligible advantage), then the scheme is considered IND-CPA secure. Formally, the pairs of oracle

$$\langle \mathsf{Enc}_k(\cdot), \mathsf{C}_k^b(\cdot,\cdot) := \mathsf{Enc}_k(m_b) \rangle$$

are indistinguishable for $b = 0, 1$ (over the random choice of $k \leftarrow \mathsf{Gen}$ and any randomness of the oracles).

Last time, we constructed the following symmetric cryptosystem SKC with $\mathcal{M} = \mathcal{K} = \{0,1\}^n$, based on a PRF family $\{f_k : \{0,1\}^n \to \{0,1\}^n\}$.

- Gen: output $k \leftarrow \{0,1\}^n$.
- $\mathsf{Enc}_k(m)$: choose $r \leftarrow \{0,1\}^n$ and output $c = (r, f_k(r) \oplus m)$
- $\mathsf{Dec}_k(r, c = (r, c'))$: output $f_k(r) \oplus c$

Completeness is evident by inspection.

1.2 Security

Theorem 1.1. *The SKC scheme described above is IND-CPA-secure (assuming $\{f_k\}$ is a PRF family).*

Proof. Note that it suffices to show that for either $b \in \{0,1\}$,

$$\langle \mathsf{Enc}_k(\cdot), \mathsf{C}_k^b(\cdot,\cdot) \rangle \overset{c}{\approx} \langle \mathsf{U}(\cdot), \mathsf{U}(\cdot,\cdot) \rangle,$$

where the oracles U answer each query with a fresh uniformly random $(r, c') \leftarrow U_{2n}$. This is because the hybrid lemma allows us to transition from $b = 0$ to $b = 1$ in the IND-CPA definition via $\langle \mathsf{U}(\cdot), \mathsf{U}(\cdot,\cdot) \rangle$.

We construct three hybrid experiments, H_0, H_1, and H_2, which each define the two oracles with which a adversary (attacking the cryptosystem) expects to interact. Experiment H_0 will correspond to the IND-CPA

attack on our actual cryptosystem (for arbitrary $b \in \{0,1\}$); experiment H_1 will correspond to the IND-CPA attack on our cryptosystem *as if it were implemented with a truly random function*; and experiment H_2 will correspond to $\langle \mathsf{U}(\cdot), \mathsf{U}(\cdot,\cdot) \rangle$. Then we will show that $H_0 \overset{c}{\approx} H_1 \overset{s}{\approx} H_2$, proving the theorem.

- H_0: choose $k \leftarrow \{0,1\}^n$.

  Answer each query m to the first (encryption) oracle by choosing $r \leftarrow \{0,1\}^n$ and answering $(r, f_k(r) \oplus m)$.

  Answer the query (m_0, m_1) to the second (challenge) oracle by choosing $r \leftarrow \{0,1\}^n$ and answering $(r, f_k(r) \oplus m_b)$.

- H_1: this experiment "lazily" constructs a truly random function $F : \{0,1\}^n \to \{0,1\}^n$, defining its output values as needed. That is, whenever the value of $F(r)$ is desired, it checks whether $F(r)$ has already been defined, and if so, uses that value. Otherwise, it defines $F(r)$ to be a fresh uniformly random value in $\{0,1\}^n$.

  Answer each query m to the first (encryption) oracle by choosing $r \leftarrow \{0,1\}^n$, and answering $(r, F(r) \oplus m)$.

  Answer the query (m_0, m_1) to the second (challenge) oracle by choosing $r \leftarrow \{0,1\}^n$ and answering $(r, F(r) \oplus m_b)$.

- H_2: answer each query (to either oracle) by a fresh uniformly random value in $\{0,1\}^{2n}$.

The difference between H_0 and H_1 is simply that we have replaced f_k with a truly uniform F. Using the fact that $\{f_k\}$ is a PRF family, we would like to show that $H_0 \overset{c}{\approx} H_1$. This is done by constructing an (nuppt) simulator $\mathcal{S}^g$ that emulates either H_0 or H_1, depending on whether g is drawn from $\{f_k\}$ or is a uniformly random function, respectively. $\mathcal{S}^g$ works in the obvious way: it answers encryption queries m by choosing $r \leftarrow \{0,1\}^n$ and returning $(r, g(r) \oplus m)$, and answers the challenge query (m_0, m_1) by choosing $r \leftarrow \{0,1\}^n$ and returning $(r, g(r) \oplus m_b)$.

The difference from H_1 to H_2 is that we answer every query by a uniformly random reply. A cursory inspection of H_1 and H_2 may lead one to believe that H_1 and H_2 are *identical*, but this is not quite the case: we must consider the case where H_1 happens to choose the same r when answering two queries to its oracles. In H_1, the value $F(r)$ used in the second component of the ciphertext remains the same, whereas in H_2, the second component of the ciphertext is still chosen uniformly at random.

Fortunately, it is the case that H_1 and H_2 are identical, *conditioned on the event that distinct queries choose distinct values of* r. This is because in this event, both H_1 and H_2 produce a uniformly random and independent second ciphertext component for each query. Therefore, we just need to bound the probability of the "bad" event, that some two queries choose the same r. Because the adversary makes a total of at most $q = \mathrm{poly}(n)$ queries to its oracles, this probability is at most $q^2/2^n = \mathrm{negl}(n)$, by taking a union bound over all pairs of queries. This means that there is a negligible statistical distance between H_1 and H_2, and $H_1 \overset{s}{\approx} H_2$ as desired.

Notice that the above argument is not just a "technicality" of the proof — it is crucial for the actual security of the scheme! Suppose that in a real attack, the challenge oracle happens to choose the same r as in one of the encryption queries. The adversary can easily compute the value of $f_k(r)$ from that query, and therefore can trivially determine which of the two messages was encrypted by the challenge oracle! $\square$

1.3 Summary of Symmetric Cryptography

In the lectures to date, we have seen the following implications (among others):

$$\text{weak-OWF} \Rightarrow \text{OWF} \Rightarrow \text{PRG} \Rightarrow \text{PRF} \Rightarrow \text{SKC}.$$

Thus, a "world" in which one-way functions exists is very rich, allowing us to do a lot of cryptography (and we will see even more applications later on). This world is often called "minicrypt," because it follows from the minimal cryptographic assumption of a (weak) OWF. Minicrypt is fairly well-understood, though new applications are still being found today.

Next, we will move beyond minicrypt into the brave new world of

CRYPTOMANIA,

in which we shall observe even more mind-boggling, seemingly paradoxical notions. However, as we will see, it is less clear what is the "minimal" assumption for Cryptomania to exist.

2 Asymmetric Encryption

In the 1970s, people started asking a dangerous question: must Alice and Bob share the *same* key to perform secure encryption? We present here a cryptographic model called asymmetric (or "public-key") encryption, in which Alice and Bob may communicate securely without a shared secret. Specifically, Bob (and the rest of the world) will encrypt messages to Alice using her *public* encryption key, and Alice will decrypt those messages using a related *secret* decryption key that only she knows. (If Bob wants to receive encrypted messages, he will generate his own pair of asymmetric keys.)

Our formal model for a public-key cryptosystem PKC with message space $\mathcal{M}$ is as follows:

- $\mathsf{Gen}(1^n)$ outputs a public key pk and secret key sk.
- $\mathsf{Enc}_{pk}(m) := \mathsf{Enc}(pk, m)$ for a message $m \in \mathcal{M}$ outputs a ciphertext c.
- $\mathsf{Dec}_{sk} := \mathsf{Dec}(sk, c)$ outputs a message $m \in \mathcal{M}$.

The notion of completeness is as expected: for $(pk, sk) \leftarrow \mathsf{Gen}$, we ask that $\mathsf{Dec}_{sk}(\mathsf{Enc}_{pk}(m)) = m$ for all $m \in \mathcal{M}$.

What about security? If we just define IND-CPA security exactly as in the symmetric setting (giving the adversary oracle access to $\mathsf{Enc}_{pk}(\cdot)$ and C^b_{pk}), we miss a crucial point: the adversary knows the public key explicitly! (After all, it is public information.) Notice, then, that there is no point in giving *oracle* access to the encryption algorithm Enc_{pk}, because the adversary can run the algorithm itself (possibly even on "bad" randomness or messages, if it likes). We are therefore left with providing pk and the challenge oracle, C^b_{pk}, and a notion of indistinguishability between $b = 0$ and $b = 1$ that is the same as in the symmetric-key setting.

Definition 2.1 (Indistinguishability under chosen-plaintext attack for asymmetric encryption)**.** We say that PKC is IND-CPA secure if the following are computationally indistinguishable for $b = 0, 1$:

$$\langle pk, \mathsf{C}^b_{pk}(m_0, m_1) := \mathsf{Enc}_{pk}(m_b) \rangle.$$

To simplify the definition even further, we can restrict the message space to $\mathcal{M} = \{0,1\}$, yielding the following "oracle-free" definition:

$$\langle pk, \mathsf{Enc}_{pk}(0)\rangle \overset{c}{\approx} \langle pk, \mathsf{Enc}_{pk}(1)\rangle.$$

As an exercise, show that a secure scheme for single-bit messages implies an IND-CPA-secure scheme for any $\mathcal{M} = \{0,1\}^{\mathrm{poly}(n)}$.

As in the symmetric setting, note that Enc cannot be deterministic if it is to be IND-CPA-secure: an adversary could simply compare the challenge ciphertext with its own encryption of a 0 or 1. Note also that it doesn't make a lot of sense to talk about a *stateful* deterministic encryption algorithm in the public-key setting, because many different parties can encrypt with the same public key, and may not be able to keep joint state.

2.1 Constructions

Is public-key cryptography possible? Attempts to construct public-key cryptosystems from one-way-functions generally fail, and there is some theoretical explanation why (which is beyond the scope of this course). But we *can* construct public-key cryptosystems from seemingly stronger assumptions.

2.1.1 From Trapdoor OWPs

Informally, a *trapdoor* OWP family is a family of one-way permutations where there is additionally some "trapdoor" that allows for efficient inversion. Formally, $\{f_s : D_s \to D_s\}$ is a family of trapdoor OWPs if:

- there exists a PPT function sampler $\mathsf{S}(1^n)$ that outputs a function index s and trapdoor t, and a poly-time inversion algorithm F^{-1} such that $\mathsf{F}^{-1}(t,y) = f_s^{-1}(y)$ for such s,t and every $y \in D_s$.
- it is a OWP family with respect to S, where as usual the inverter is given 1^n, s, and $f_s(x)$ (but t is withheld).

Note that the family still has a hard-core predicate, because it is one-way; the trapdoor is just an additional *functional* property that is not exposed to the adversary.

Using a trapdoor OWP family with hard-core predicate h, we can construct a public-key cryptosystem PKC for $\mathcal{M} = \{0,1\}$ as follows:

- Gen: $(s,t) \leftarrow \mathsf{S}$. Output $pk = s$ and $sk = t$.
- $\mathsf{Enc}_s(m)$: choose $r \leftarrow D_s$ and output $c = (f_s(r), h(r) \oplus m)$.
- $\mathsf{Dec}_t(c = (y,c'))$: output $m' = h(f_s^{-1}(y)) \oplus c'$

Completeness of the scheme is evident by inspection (note that Dec_t can compute f_s^{-1} because it knows the trapdoor t). It is important that we are using a *hard-core* predicate $h(r)$ to conceal the message bit m, because in general f_s might reveal certain bits of r while still being one-way.

Theorem 2.2. *The PKC scheme described above is IND-CPA-secure (assuming $\{f_s\}$ is a trapdoor OWP family with hard-core predicate h).*

Proof. We need to show that

$$\langle s, f_s(r), h(r) \oplus 0 \rangle \overset{c}{\approx} \langle s, f_s(r), h(r) \oplus 1 \rangle.$$

By the hypothesis that h is hard-core, we have

$$\langle s, f_s(r), h(r) \oplus 0 \rangle \overset{c}{\approx} \langle s, f_s(r), \mathsf{U}_1 \rangle \equiv \langle s, f_s(r), \mathsf{U}_1 \oplus 1 \rangle \overset{c}{\approx} \langle s, f_s(r), h(r) \oplus 1 \rangle. \qquad \square$$

Do trapdoor OWP families exist? Here are some plausible candidates:

1. Rabin's function: $f_N \colon \mathbb{QR}_N^* \to \mathbb{QR}_N^*$, defined as $f_N(x) = x^2 \bmod N$, for N a product of two primes.
2. RSA: $f_{N,e} \colon \mathbb{Z}_N^* \to \mathbb{Z}_N^*$, defined as $f_{N,e}(x) = x^e \bmod N$, for $N = pq$ a product of two primes and e coprime with $\varphi(N) = (p-1)(q-1)$. The trapdoor is $d = e^{-1} \bmod \varphi(N)$, which can be used to calculate $f_{N,e}^{-1}(y) = y^d \bmod N$. This works because $x^{ed} = x^{ed \bmod \varphi(N)} = x \in \mathbb{Z}_N^*$, by Euler's theorem and the fact that $\mathbb{Z}_N^*$ has order $\varphi(N)$.
3. What about modular exponentiation $f_{p,g}(x) = g^x \bmod p$? It is unknown if there is any trapdoor for the discrete log problem.
4. Assorted others: Paillier's function (works mod N^2), "lossy" trapdoor functions (which are not permutations, but are injective, which suffices for encryption), and injective TDFs based on lattices.

2.1.2 From Diffie-Hellman

Even though we lack a *trapdoor* for the discrete logarithm problem, it turns out that we can still construct a public-key cryptosystem that uses modular exponentiation. The starting point is the Diffie-Hellman key-exchange protocol, which can be used by Alice and Bob to agree on a shared secret value over a public channel.

Let $G = \langle g \rangle$ be a (multiplicative) cyclic group of known order q, with public generator g. Alice chooses some $a \leftarrow \mathbb{Z}_q$ and Bob chooses some $b \leftarrow \mathbb{Z}_q$. Alice sends $\alpha = g^a$ and Bob sends $\beta = g^b$. They then can agree on a shared value g^{ab}: Alice by computing β^a, and Bob by computing α^b. It is conjectured that in certain groups, it is computationally infeasible for any algorithm to compute g^{ab} given only g, $\alpha = g^a$, and $\beta = g^b$. Moreover, it is believed that the shared value g^{ab} "looks random," even given those other values.

Conjecture 2.3 (DDH Assumption)**.** The Decision Diffie-Hellman assumption for a group $G = \langle g \rangle$ of order q says that

$$(g, g^a, g^b, g^{ab}) \overset{c}{\approx} (g, g^a, g^b, g^c),$$

where $a, b, c \leftarrow \mathbb{Z}_q$ are uniformly random and independent.[1]

For a prime p, is DDH true or false for $\mathbb{Z}_p^*$? It is actually *false*! Think about the probability that $a \cdot b$ is even, versus the probability that c is even, and recall that in $\mathbb{Z}_p^*$ we can test whether z is even, given g^z. We can avoid this annoying property by working in the group of *quadratic residues* $G = \mathbb{QR}_p^*$, where $p = 2q + 1$ and q itself is prime. This group has order $(p-1)/2 = q$, and DDH is believed to hold in it (and some others, such as those defined over elliptic curves).

By simply "packaging" the Diffie-Hellman protocol in the appropriate way, we obtain the *ElGamal* public-key cryptosystem over the group G, with message space $\mathcal{M} = G$.

[1] To be completely formal, to give a meaningful asymptotic definition using computational indistinguishability, we should define the DDH assumption over an *infinite family* of groups G.

- Gen: choose secret key $sk = a \leftarrow \mathbb{Z}_q$, and let $pk = \alpha = g^a$.
- $\mathsf{Enc}_\alpha(m)$: choose $b \leftarrow \mathbb{Z}_q$, and output $c = (g^b, \alpha^b \cdot m)$.
- $\mathsf{Dec}_a(c = (\beta, c'))$: output c'/β^a.

Correctness is by inspection. Security follows directly from the DDH assumption: it is a simple exercise to show that the view of the adversary is computationally indistinguishable from (g, g^a, g^b, g^c) (where $a, b, c \leftarrow \mathbb{Z}_q$), no matter which message is encrypted by the challenger.