1 Review of Lectures 29 and 30

In the last two lectures, we introduced two popular number-theoretic hardness assumptions, that are widely used for public key cryptography. The first one is the Decisional Diffie-Hellman (DDH) assumption, which states that no polynomial time adversary can distinguish between the following two distributions:

$$\mathcal{D}_0 = \left\{(g, g^a, g^b, g^{ab}) \ : \ g \leftarrow \mathbb{G}, a, b \leftarrow \mathbb{Z}_q\right\} \qquad \mathcal{D}_1 = \left\{(g, g^a, g^b, g^c) \ : \ g \leftarrow \mathbb{G}, a, b, c \leftarrow \mathbb{Z}_q\right\}$$

Here, q is a large prime, and $\mathbb{G}$ is a group of size q. Elgamal showed a **semantically secure encryption scheme** whose security relies on the DDH assumption.

Next, we saw the RSA assumption. This states that no p.p.t. adversary, given a random modulus $N = p \cdot q$ (p and q are random n bit primes), a random exponent e that is co-prime to $\phi(N)$, and a random element $y \leftarrow \mathbb{Z}_N^*$, can find the unique $x \in \mathbb{Z}_N^*$ such that $x^e \bmod N = y$. We saw that the textbook RSA encryption scheme does not offer semantic security. Next, we saw a few variants which are not immediately insecure. In all these variants, the key generation algorithm is identical. The public key consists of (N, e), while the secret key consists of (N, d) (where $e \cdot d \mod \phi(N) = 1$).

Construction 1: An attempt closely related to the textbook RSA encryption scheme

Message space is $\mathbb{Z}_N^*$.

- $\mathsf{Enc}(m, (N, e))$: Choose $x \leftarrow \mathbb{Z}_N^*$, output $\mathsf{ct}_1 = x^e \bmod N$, $\mathsf{ct}_2 = (m + x)^e \bmod N$.
- $\mathsf{Dec}((\mathsf{ct}_1, \mathsf{ct}_2), (N, d))$: Compute $x = \mathsf{ct}_1^d \bmod N$, $y = \mathsf{ct}_2^d \bmod N$, output $y - x \bmod N$.

First, note that $(m + x)$ need not be in $\mathbb{Z}_N^*$. However, for all $\alpha \in \mathbb{Z}_N$, $(\alpha^e)^d \bmod N = \alpha$.[a] Therefore, decryption correctness holds.

It is not clear if this scheme is secure/insecure; we couldn't find any attacks on the same. See Question (Pizza Problem) below.

[a] The proof of this uses the Chinese Remainder Theorem (CRT). If you know CRT, you are encouraged to prove this.

Construction 2: A provably secure practically inefficient scheme based on RSA

Message space is $\{0, 1\}$. Here, we assume that the primes p and q are n-bit primes, and therefore the modulus N is an $\ell = 2n$-bit number.

- $\mathsf{Enc}(m, (N, e))$: Choose $\widetilde{x} \leftarrow \{0, 1\}^{\ell - 2}$, set $x = \widetilde{x} \ || \ m$, interpret x as a number in $\mathbb{Z}_N$. Output $\mathsf{ct}_1 = x^e \bmod N$.
- $\mathsf{Dec}(\mathsf{ct}_1, (N, d))$: Compute $x = \mathsf{ct}_1^d \bmod N$, express it as an $\ell - 1$ bit string. Output the least significant bit of x.

Correctness follows immediately (again, using the fact that for all $\alpha \in \mathbb{Z}_N$, $(\alpha^e)^d \bmod N = \alpha$).

The security of this scheme relies on a variant of the RSA assumption. This variant is equivalent to the RSA assumption (however, we will not discuss the equivalence proof in this course. Interested students should refer to [ACGS84]).

Assumption 31.01 (RSA: Hardcore bit). For any p.p.t. adversary $\mathcal{A}$, there exists a negligible function $\mu(\cdot)$ such that for all n, $|\Pr\left[\mathcal{A}(N, e, x^e, \mathsf{lsb}(x)) = 1\right] - \Pr\left[\mathcal{A}(N, e, x^e, \overline{\mathsf{lsb}(x)}) = 1\right]| \leq \mu(n)$. Here, $\mathsf{lsb}(x)$ denotes the least-significant-bit of x, and the probabilities are over the choice of two random n-bit primes p, q, setting $N = p \cdot q$, the choice of random exponent e such that $\gcd(e, \phi(N)) = 1$, the choice of $x \leftarrow \mathbb{Z}_N$ and the randomness of $\mathcal{A}$.

Construction 3: A practically efficient, heuristic scheme based on RSA

Message space is $\{0,1\}^*$. This construction uses a hash function, say SHA. We assume that SHA takes arbitrary length inputs, and outputs an n bit string. The scheme also uses a symmetric key encryption scheme $\mathcal{E}_{\text{sym}} = (\mathsf{Enc}_{\text{sym}}, \mathsf{Dec}_{\text{sym}})$. For the symmetric key encryption scheme, one can use the no-query semantically secure PRG based encryption scheme.

- $\mathsf{Enc}(m, (N, e))$: Choose $x \leftarrow Z_N$, compute $\mathsf{ct}_1 = x^e \bmod N$, $k = \mathsf{SHA}(x)$, $\mathsf{ct}_2 = \mathsf{Enc}_{\text{sym}}(m, k)$.
- $\mathsf{Dec}(\mathsf{ct}_1, (N, d))$: Compute $x = \mathsf{ct}_1^d \bmod N$, $k = \mathsf{SHA}(x)$. Output $\mathsf{Dec}_{\text{sym}}(\mathsf{ct}_2, k)$.

Correctness follows immediately using the correctness of $\mathcal{E}_{\text{sym}}$.

However, what property of SHA is needed to prove that this scheme is semantically secure? Check that properties like one-wayness or collision-resistance do not suffice for proving semantic security of this scheme. In fact, there is no security proof for this construction.

Heuristic constructions like the one above are widely used in practice, and therefore we would like to gain some confidence in these schemes. While a proof of security in the standard model would be great, we will need to rely on proofs in *idealized models*. This is the focus of this lecture.

1.1 Decisional vs Computational Assumptions

Before we move on to security proofs in idealized models, let us quickly discuss one last point related to RSA and DDH. Using DDH, we saw a fairly simple construction (the Elgamal scheme). However, finding a similar RSA-based construction seems challenging. Similarly, when we discussed key exchange, there was an immediate construction based on DDH, but the RSA based key-exchange protocol had a weaker security guarantee (the adversary cannot learn the entire shared key, but it could learn some partial information about the key).

There is a fundamental difference between these two assumptions. In RSA, the adversary needs to compute the e^{th} root, while DDH is a decision problem - the adversary needs to distinguish between two distributions. Similar to DDH, we can have a weaker assumption where the adversary, given g, g^a, g^b, must compute g^{ab}. This is known as the *computational Diffie-Hellman* (CDH) assumption. CDH is a weaker (and therefore, better) assumption than DDH. However, if we try to construct an encryption scheme whose security is based on CDH, we will run into similar issues. We will either need to rely on heuristics, or we will have a less practical scheme (based on the hardcore bit property). See Appendix A for more details.

2 The Random Oracle Heuristic

So far in this course, whenever we have proposed a construction, we have also provided a security proof by assuming certain properties about the underlying building blocks. However, for the SHA-based construction defined above, it is not clear what we should assume about SHA in order to have a formal proof of security. A lot of practical cryptosystems are heuristic constructions, and we would like to have some sort of formal guarantees about these schemes.

The 'random oracle model' is a security model that gives us some confidence in these heuristic constructions. In such constructions, we have one or more publicly known functions (such as SHA). In the security proof, we assume that the adversary does not know the function(s), but must instead query a random oracle to learn the evaluation at any input of its choice. More formally, let us consider the security of a public key encryption scheme in the random oracle model. Every cryptographic primitive must have a separate definition of security in the random oracle model.

Semantic security in the random oracle model

The construction uses a hash function $H : \{0,1\}^* \to \{0,1\}^n$. Suppose the encryption and decryption algorithms use the hash function H.

- (Setup Phase) Challenger chooses $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}$ and sends pk to the adversary. The challenger also chooses a uniformly random function $f : \{0,1\}^* \to \{0,1\}^n$.
- (Random Oracle Queries) The adversary is allowed polynomially many pre-challenge random oracle queries. It sends an input x, and receives $f(x)$.
- (Challenge Phase) The adversary sends two messages (m_0^*, m_1^*). The challenger chooses $b \leftarrow \{0,1\}$, computes $\mathsf{ct} \leftarrow \mathsf{Enc}^f(m_b^*, \mathsf{pk})$ and sends it to the adversary. Note that since encryption uses the hash function, the encryption algorithm is allowed to make calls to the function f.
- (Random Oracle Queries) The adversary is allowed polynomially many post-challenge random oracle queries. It sends an input x, and receives $f(x)$.
- (Guess) The adversary sends its guess b' and wins if $b = b'$.

Figure 1: Security game for defining security of an encryption scheme in the random oracle model

We say that an encryption scheme is secure in the random oracle model if no p.p.t. adversary can win the game defined in Figure 1 with non-negligible advantage. Here the probability is over the randomness used by the challenger and adversary, **and also the choice of** f. Note that the random oracle must be a consistent oracle (if same input is queried twice, the output must be the same string).

What we can conclude, given security in the random oracle model Security in the random oracle model tell us that there are no 'generic' attacks on the scheme. An adversary will need to exploit the internal structure of the hash function in order to launch an attack. However, it does not offer any formal guarantees in the real world. It is possible that a cryptosystem is secure in the random oracle model, but when instantiated with a hash function in the real-world, it is completely insecure.

3 Proving security in the random oracle model

Let us first consider a few simple cryptographic primitives. We will show constructions (using a hash function H) and prove security in the random oracle model.

3.1 CRHF in the random oracle model

Let $H : \{0,1\}^* \to \{0,1\}^n$ be a keyless deterministic function. We wish to construct a keyed hash function family $\mathcal{F} = \{F_k : \{0,1\}^* \to \{0,1\}^n\}_{k \in \mathcal{K}}$, and prove collision resistance in the random oracle model. First, let us define collision resistance in the random oracle model. The definition is analogous to the definition in standard model. The adversary receives a CRHF key k. It is then allowed queries to the random oracle. After polynomially many queries, it outputs two strings x_1, x_2, and wins if $F_k(x_1) = F_k(x_2)$ (note that the winning condition depends on the random oracle, since the definition of F_k depends on H which is modeled as a random oracle in the proof).

Let us take the most natural idea: the key space is $\{0,1\}^n$, and $F_k(x) = H(k \mid\mid x)$. In the random oracle model, the adversary receives a fresh random string for each new query. Using the birthday bound, it follows that an adversary making at most q queries can win with probability at most $q^2/2^n$, which is negligible if q is $\mathsf{poly}(n)$.

Discussion: Recall, when we were discussing CRHFs, we talked about keyless hash functions. How do we make sense of the security guarantees offered by practical hash functions such as SHA? One way of looking at this is by assuming that the hash key was chosen when SHA was designed, and this key is embedded in the design of SHA. Another way to look at this is via the random oracle model. Even if SHA is a truly

keyless function, because of its complex design, it 'behaves' like a random oracle. And we have shown that it is a secure CRHF in the random oracle model.

A (big?) leap of faith: When we say that SHA 'behaves' like a random oracle, we are taking a leap of faith here. This is very different from saying that AES, with a random key, 'behaves' like a random permutation. Please make sure you understand this distinction. Consider, for instance, a deterministic function H that maps all bit strings to 0^n. Clearly, if you use H as discussed above, then the resulting function family $\{F_k\}$ is not a secure CRHF. However, if we model H as a random oracle, then our proof above shows that $\mathcal{F}$ is a secure CRHF. This illustrates that security in the ROM does not imply security in the standard model. However,

3.2 MAC in the random oracle model

Let H be a deterministic function (with unbounded length inputs, and outputting n bit strings). Using H, we want to build a MAC scheme that can handle unbounded length messages. Below, we describe the most natural idea. The key space is $\mathcal{K} = \{0,1\}^{n_1}$.

A message authentication code using hash functions

$H : \{0,1\}^* \to \{0,1\}^{n_2}$ is a function (that will be modeled as a random oracle in the security proof). We wish to construct a MAC scheme using H.

- $\mathsf{Sign}(m,k)$: The signing algorithm outputs $\sigma = H(k \mid\mid m)$.
- $\mathsf{Verify}(m,\sigma,k)$: The verification algorithm checks if $\sigma = H(k \mid\mid m)$.

Claim 31.01. The above construction is a secure MAC in the random oracle model.

Proof. First, let us recall the MAC security game in the random oracle model. The challenger chooses a key $k^* \leftarrow \{0,1\}^{n_1}$. We will choose the uniformly random function f 'on-the-fly' by maintaining a table $\mathcal{T}$, which is initially empty.

The adversary is allowed two different kinds of queries: random oracle queries and signature queries.

- Random oracle queries: the adversary sends inputs of the form *key* || *string*. The challenger uses the table $\mathcal{T}$ for responding to these queries. If the query is already present in the table, it sends back the same response. Else, if $(k' \mid\mid x)$ is a fresh query, it chooses a uniformly random string $y \leftarrow \{0,1\}^n$, sends y to the adversary, and adds $(k' \mid\mid x \ \ , \ \ y)$ to $\mathcal{T}$.
- MAC queries: the adversary queries for signatures on any message of its choice. On query m, the challenger checks if $\mathcal{T}$ contains $k^* \mid\mid m$. If it does, then it sends the corresponding table entry as the signature. Else, it picks a uniformly random string y, adds $(k^* \mid\mid m \ \ , \ \ y)$ to $\mathcal{T}$ and sends y.

At the end, the adversary sends a forgery (m^*, σ^*) and wins if $H(k^* \mid\mid m^*) = \sigma^*$ and $m^* \notin \{m_i\}$. Note that this is a scheme with a unique signature for each message, hence weak unforgeability and strong unforgeability are equivalent. There are two cases:

- none of the random oracle queries are equal to $k^* \mid\mid m^*$. In this case, the adversary's signature is a valid forgery with probability $1/2^{n_2}$.
- for at least one of the random oracle queries, the first n_1 bits are equal to k^*. Suppose the adversary makes q queries (either signing or RO queries). Then the probability of this event is at most $q/2^{n_1}$. (Why?)

Therefore, the success probability of the adversary in the random oracle model is at most $q/2^{n_1} + 1/2^{n_2}$, which is negligible if q is polynomial in the security parameter, and n_1, n_2 are large enough (Θ(security parameter)). □

Key ROM feature used in this proof: The main idea here is the following: even after seeing the hash function evaluations on a number of inputs, the output on a new input is uniformly random.

3.3 MAC in the ROM, using bounded-input hash function

Let us now consider the following scenario: our hash function does not allow arbitrary length inputs. Instead, the input space is $\{0,1\}^{2n}$, while the output space is $\{0,1\}^n$. Again, we would like to construct a MAC scheme for unbounded length inputs. Recall, the Merkle-Damgard (MD) transform allows us to expand the input space of a hash function. Can we combine the MD transform with the previous construction, to obtain a secure MAC in the random oracle model? The construction is described below. For simplicity, we assume the message length is a multiple of n.

The key space is $\{0,1\}^n$. Let $m = (m_1, m_2, \ldots, m_\ell)$ be the message to be signed, where each $m_i \in \{0,1\}^n$. For simplicity, we are assuming that the message length is a multiple of n. If that's not the case, we will need to use some appropriate padding.

- $\mathsf{Sign}(m = (m_1, \ldots, m_\ell), k)$: Set $y_1 = H(k \mid\mid m_1)$. For all $i \in [2, \ell]$, compute the following:
  - $y_i = H(y_{i-1} \mid\mid m_i)$.

  Output y_ℓ as the final signature.

The verification algorithm can be defined similarly.

This construction is not secure even in the random oracle model! The adversary can query for a signature on $m_1 \in \{0,1\}^n$, receives σ_1. Next, it picks a random message $m_2 \in \{0,1\}^n$, queries the random oracle on $(\sigma_1 \mid\mid m_2)$ and receives an n bit-string y. The adversary is now ready to send its forgery: it sends (m_1, m_2) as the message, and y as the corresponding signature. Check that this adversary indeed wins the MAC forgery game in the random oracle model.

Suppose we modify the above scheme, and append the key at the end. That is, we use Sign' defined below.

- $\mathsf{Sign}'(m = (m_1, \ldots, m_\ell), k)$: Set $y_1 = H(m_1 \mid\mid m_2)$. For all $i \in [2, \ell - 1]$, compute the following:
  - $y_i = H(y_{i-1} \mid\mid m_{i+1})$.

  Set $y_\ell = H(y_{\ell-1} \mid\mid k)$ and output y_ℓ as the final signature.

This scheme is also not a secure MAC, even in the random oracle model. The adversary can query the random oracle on $m_1 \mid\mid m_2$, and receives y_1 in response. Next, it queries for a signature on y_1, and receives σ. Finally, the adversary sends a forgery: it sends (m_1, m_2) as the message, and σ as the signature. Note that this is a valid forgery.

Check why the proof in Section 3.2 does not work for the constructions defined in this section.

4 Lecture summary, plan for next lecture, additional resources

Summary: The random oracle model is a security model using which we can prove some formal guarantees about commonly used heuristic constructions. Suppose we have a cryptosystem in which the various algorithms use some complicated deterministic function $H : \{0,1\}^* \to \{0,1\}^n$. When this cryptosystem is used, all the parties running the various algorithms know how to evaluate the function H. For instance, if we have an encryption scheme where both Enc and Dec use this function H, then we assume that the encryptor and decryptor both know how to evaluate H. However, to prove security, we assume that the adversary does not know H. Instead, it must query the challenger to get evaluations on inputs of its choice. This is the random oracle security model.

Many cryptographic constructions rely on heuristics, and the random oracle model is one of the popular models for gaining some confidence in these systems. Note that a ROM proof does not give any formal guarantees in the standard model. We can have schemes that are secure in the ROM, but completely insecure in the standard model (we discussed one such example in Section 3.1).

Next, we saw a secure MAC scheme in the random oracle model, and two insecure MAC schemes in the ROM. Consider the following variant of the constructions in Section 3.3:

- $\mathsf{Sign}_1(m = (m_1, \ldots, m_\ell), k)$: Set $y_1 = H(k \mid\mid m_1)$. Next, for each i, set $y_i = H(y_{i-1} \mid\mid m_i)$. Finally, output $H(y_\ell \mid\mid k)$ as the signature.

This construction should be provably secure in the random oracle model. As a result, this construction is 'better' than the two constructions in Section 3.3, even though none of them have a security proof in the standard model.

Next lecture: We will go back to our RSA-based encryption schemes, and prove security for Construction 3 (Section 1) in the random oracle model.

Additional resources There is a nice discussion about random oracles in Katz-Lindell's textbook. I've uploaded the relevant chapter on Moodle.

5 Questions

Question 31.01. Let $H : \{0,1\}^* \to \{0,1\}^n$ be a hash function. Using H, construct a keyed function $F : \{0,1\}^* \times \{0,1\}^n \to \{0,1\}^n$ that is a secure pseudorandom function in the random oracle model.

Question (Pizza Problem). [a]
This question is with regard to Construction 1 (see Section 1). Currently, we neither have a proof, nor an attack for the same. Perhaps it is possible to prove security in some *idealized models*. Similar to the random oracle model, there are other idealized models which are specific to the factoring or group-theoretic assumptions. Find an appropriate idealized model for RSA/factoring based schemes, and prove security of Construction 1 in this idealized model.

[a] This is not a typo, I indeed meant pizza here, not Piazza

References

[ACGS84] Werner Alexi, Benny Chor, Oded Goldreich, and Claus-Peter Schnorr. Rsa/rabin bits are 1/2 + 1/poly(log N) secure. In *25th Annual Symposium on Foundations of Computer Science, West Palm Beach, Florida, USA, 24-26 October 1984*, pages 449–457. IEEE Computer Society, 1984.

A Post-lecture discussions

Below, I have summarized a few interesting points that came up after today's class. These are not part of the syllabus.

A.1 CDH based encryption schemes

The CDH assumption states that given g, g^a, g^b, it is hard to compute g^{ab}. Similar to the RSA based constructions, we can construct either provably secure, or heuristic encryption schemes from CDH.

CDH based bit encryption scheme

As in Construction 2, this construction will also rely on a variant of CDH (which is equivalent to CDH). However, the picture is not as clean as it was for RSA. This is because the quantity to be computed here (g^{ab}) is a group element, and therefore, the hardcore bit depends on the choice of group $\mathbb{G}$. Let us assume, for simplicity of our discussion, that the least significant bit of g^{ab} is hard to predict, given g, g^a, g^b. Then, we can have an analogous bit encryption scheme.

- $\mathsf{Enc}(m, \mathsf{pk} = (g, g^a))$: Choose $b \leftarrow \mathbb{Z}_q$, output $\mathsf{ct}_1 = g^b$, $\mathsf{ct}_2 = m \oplus \mathsf{lsb}(g^{ab})$.
- $\mathsf{Dec}(\mathsf{ct} = (\mathsf{ct}_1, \mathsf{ct}_2), \mathsf{sk} = a)$: Output $\mathsf{ct}_2 \oplus \mathsf{lsb}((\mathsf{ct}_1)^a)$.

A general note about hardcore bits: This is no coincidence that RSA and CDH have hardcore bits. You can read more about hardcore bits here.

A.2 CDH based heuristic encryption schemes

Let $H : \{0,1\}^* \to \{0,1\}^n$ be a hash function (which will be modeled as a random oracle in the proof). Let $\mathcal{E}_{\text{sym}} = (\mathsf{Enc}_{\text{sym}}, \mathsf{Dec}_{\text{sym}})$ be a symmetric key encryption scheme with message space $\{0,1\}^*$.

CDH based heuristic encryption scheme

This will be similar to Construction 3 from Section 1.

- $\mathsf{Enc}(m, \mathsf{pk} = (g, g^a))$: Choose $b \leftarrow \mathbb{Z}_q$, compute $\mathsf{ct}_1 = g^b$, $k = H(g^{ab}))$, $\mathsf{ct}_2 = \mathsf{Enc}_{\text{sym}}(m, k)$. Output $(\mathsf{ct}_1, \mathsf{ct}_2)$.
- $\mathsf{Dec}(\mathsf{ct} = (\mathsf{ct}_1, \mathsf{ct}_2), \mathsf{sk} = a)$: Compute $k = H(\mathsf{ct}_1^a)$, output $\mathsf{Dec}_{\text{sym}}(\mathsf{ct}_2, k)$.

The random-oracle based proof of security is a bit more complicated for this construction. I might include this as a practice/assignment question (with sufficient hints).

A natural question to ask is the role of the hash function in the above construction. The short answer is that it makes the security proof work. Suppose, instead of using H, you simply set $\mathsf{ct}_2 = m \cdot g^{ab}$ and output $(\mathsf{ct}_1, \mathsf{ct}_2)$ as the ciphertext. When proving security, the reduction algorithm receives (g, g^a, g^b) from the CDH challenger. It is not clear how this reduction algorithm can use an adversary that breaks semantic security of the above construction.

Another interesting question is whether we can use Discrete Log (instead of CDH). For this, I don't know even a heuristic construction, or a good reason to believe why this is implausible. If someone finds anything with regard to this, please let me know.