Introduction to Cryptography 02/22/2018

Lecture 11: Key Agreement

Instructor: Vipul Goyal *Scribe: Francisco Maturana*

1 Hardness Assumptions

In order to prove the security of cryptographic primitives, we need to be able to show that certain problems cannot be solved efficiently. This, however, has proven to be a very difficult task, so we often have to rely on hardness assumptions for our proofs of security. These assumptions are based on longstanding problems for which no efficient algorithms are known, and are therefore widely believed to be hard.

Assumption 1 (Discrete Log Assumption (DL)) *Let G_q be an order q multiplicative group, where q is a prime, and let $g \in G_q$ be a generator. Then, for all PPT adversary A:*

$$\Pr[x \xleftarrow{\$} \mathbb{Z}_q : A(g^x) = x] \leq \text{negl}(|q|)$$

We can construct a one-to-one OWF $f_g : \mathbb{Z}_q \to G_q$ based on the DL assumption:

$$f_g(x) = g^x$$

Lemma 1 *The function f_g is a one-to-one OWF.*

Proof. First we show that f_g is one-to-one. Since g is a generator, and G_q is of order q it must be the case that for all $x, y \in \mathbb{Z}_q$ such that $x \neq y$, $g^x \neq g^y$, or otherwise g would not be able to generate all the elements of G_q. Then, if $x, y \in \mathbb{Z}_q$ are such that $f_g(x) = f_g(y)$, we must have that $x = y$.

The function f_g can be efficiently computed by using *exponentiation by squaring*. This will take $O(\log |\mathbb{Z}_q|) = O(|q|)$ group operations in the worst case.

Finally, f_g is hard to invert as a direct consequence of the DL assumption. ■

Given that we typically work with bit-strings instead of prime order groups, it would be useful to adapt f_g to work in the more familiar setting. We can construct a one-to-one OWF that takes bit strings as input by making use of the following lemma.

Lemma 2 *Let $f : \mathbb{Z}_q \to G_q$ be a function, let n be such that $2^n \leq q < 2^{n+1}$, let m be such that $2^m \geq |G_q|$, and let $f' : \{0,1\}^n \to \{0,1\}^m$ be such that:*

$$f'(x) = f(x) \quad \text{for all } x \in \{0,1\}^n$$

where $x \in \{0,1\}^n$ and $f(x) \in G_q$ are respectively mapped to elements of $\mathbb{Z}_q$ and $\{0,1\}^m$ in the trivial way. If f is a OWF, then f' is a OWF.

Proof. Suppose, towards contradiction, that there is an adversary A that can invert f' with noticeable probability p. Consider the adversary A' that attempts to invert f as follows: on input $Y \in G_q$, A' converts Y to an element $y \in \{0,1\}^m$ and outputs $A(y)$.

Given our choice of n, all the elements in $\{0,1\}^n$ have a corresponding element in $\mathbb{Z}_q$, and at least one half of the elements in $\mathbb{Z}_q$ have a corresponding element in $\{0,1\}^n$. This means that an

element $X \in \mathbb{Z}_q$ chosen at random will have a corresponding element $x \in \{0,1\}^n$ with probability at least $1/2$. The probability that A' succeeds given that such an x exists, is exactly p. Therefore, the overall probability that A' succeeds is at least $p/2$, which is noticeable.

It is also worth noting that if f is one-to-one, then f' is also one-to-one, since the mappings $\{0,1\}^n \to \mathbb{Z}_q$ and $G_q \to \{0,1\}^m$ are one-to-one. ■

Assumption 2 (Computational Diffie-Hellman Assumption (CDH)) *Let G_q be a prime-order multiplicative group and let $g \in G_q$ be a generator. Choose x, y uniformly at random from $\mathbb{Z}_q$. Let $X = g^x$ and $Y = g^y$. Then, for all PPT adversary A:*

$$\Pr[A(g, X, Y) = g^{xy}] \leq \text{negl}(|q|)$$

Notice that this is a potentially stronger assumption than DL, since if we are able to solve DL efficiently, we are also able to solve CDH efficiently.

Assumption 3 (Desicional Diffie-Hellman Assumption (DDH)) *Let G_q, g, X, Y be defined as before. Let $Z = g^{xy}$ and $R \xleftarrow{\$} G_q$. Then, for all PPT adversary A:*

$$|\Pr[A(g, X, Y, Z) = 0] - \Pr[A(g, X, Y, R) = 0]| \leq \text{negl}(|q|)$$

Notice that this assumption is potentially stronger than CDH. Clearly, if we are able to compute g^{xy} efficiently, then we can just compute it and check if $Z = g^{xy}$.

The tuples (g^x, g^y, g^{xy}) are sometimes called DDH triplets or DDH tuples in the literature.

1.1 RSA assumption/function

Let $N = pq$ where p and q are distinct primes. Then, as we previously mentioned:

$$|\mathbb{Z}_N^*| = \Phi(N) = (p-1)(q-1)$$

Choose an $e \in \{1, 2, \ldots, \Phi(N) - 1\}$ such that e is relatively prime to $\Phi(N)$. Compute d such that $ed = 1 \mod \Phi(N)$ (d must exist since e is relatively prime to $\Phi(N)$). We call the following function the **RSA function**:

$$f_{N,e}(x) = x^e \mod N$$

The RSA assumption states that given e and N, it is hard to compute e-th roots. More formally:

Assumption 4 (RSA assumption) *Let Π_n be the set of primes of length n. Then:*

$$\Pr\left[\begin{array}{c} p, q \leftarrow \Pi_n,\ p \neq q,\ N = pq, \\ e \leftarrow \mathbb{Z}^*_{\Phi(N)},\ x \leftarrow \mathbb{Z}^*_N \end{array} : A(x^e \mod N) = x \right] \leq \text{negl}(n)$$

One key property of the RSA assumption is that when we know d it becomes very easy to invert.

Inverting RSA Given $x^e \mod N$ we can compute:

$$x^{ed} \mod N = x^{ed \mod \Phi(N)} \mod N = x^{1 \mod \Phi(N)} \mod N = x \mod N = x$$

Given $\Phi(N)$, it is easy to compute d from e by using the extended euclidean algorithm. Therefore, RSA becomes easy to invert on we know $\Phi(N)$. Computing $\Phi(N)$ in this case, however, is believed to be hard.

One way in which we may compute $\Phi(N)$ is by factorizing N into p and q, and then computing $(p-1)(q-1)$. This means that factoring being hard is a necessary condition for the RSA assumption to hold. It is not known if this is a sufficient condition, however, since there might be other ways of computing $\Phi(N)$ that do not involve factoring.

The RSA assumption, however, implies that the RSA function is hard to invert when we have no additional information.

Lemma 3 *The RSA function is a OWP under the RSA assumption.*

Proof. First we show that it is a permutation. Suppose there are two preimages $x, y \in \mathbb{Z}_N^*$ mapping to the same image. Then:

$$\begin{aligned} x^e &= y^e \mod N \\ (x^e)^d &= (y^e)^d \mod N \\ x^{1 \mod \Phi(N)} &= y^{1 \mod \Phi(N)} \mod N \\ x &= y \mod N \end{aligned}$$

Since the domain and codomain of the function are the same, we deduce that it is a permutation.

We can efficiently compute the RSA function by using *exponentiation by squaring*. This will take $O(\log |\mathbb{Z}_{\Phi(N)}^*|)$ group operations.

Finally, the fact that the RSA function is hard to invert follows directly from the RSA assumption. ■

RSA is often presented as an asymmetric encryption scheme, where the RSA function with the *public key* e is used to encrypt messages and the RSA function with the *private key* d is used to decrypt them. It is worth noting, however, that RSA is **not** a secure encryption scheme, since it is deterministic. As shown in a previous lecture, no deterministic encryption scheme is secure for multiple messages.

1.2 Learning with Errors (LWE)

Consider the following problem. Let $x = (x_1, x_2, \ldots, x_n)$ where $x_i \in \mathbb{Z}_q$ for all $i \in [n]$. Given several equations:

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n &= b_1 \mod q \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n &= b_2 \mod q \\ &\vdots \end{aligned}$$

The goal is to find an x satisfying the system of equations. This is a classical problem easily solved using Gaussian elimination.

LWE introduces the following variation. We are given noisy version $\tilde{b}_1, \tilde{b}_2, \ldots, \tilde{b}_n$ of $b_1, b_2, \ldots, b_n$ such that:

$$\tilde{b}_i = b_i + e_i \quad \text{for all } i \in [n]$$

where e_i is noise sampled from a given distribution.

Assumption 5 (Learning with Errors (LWE)) *For all PPT adversary A:*

$$\Pr[A(a_{11}, a_{12}, \ldots, \tilde{b}_1, \tilde{b}_2, \ldots) = x] \leq \text{negl}(n)$$

LWE is believed to be hard when noise is sampled from a normal distribution with certain standard deviations.

This assumption is often used in lattice-based cryptography.

2 Diffie-Hellman Key Exchange

A key exchange protocol (KEP) allows two PPT parties—say Alice and Bob having randomness r_A, r_B respectively—to interact with each other. Let τ be the public transcript at the end of the protocol. The view of Alice is $V_A = (r_A, \tau)$ and the view of Bob is $V_B = (r_B, \tau)$. Given V_A and V_B, Alice and Bob can compute k_A and k_B respectively. If $k_A = k_B$, denote them by k.

- **Correctness**: the protocol is correct if:

$$\Pr_{r_A, r_B}[k_A = k_B] = 1$$

- **Security**: Consider PPT Eve (denoted by E) eavesdropping the communication channel. The view of Eve is τ. Let k be the key and let $U_n \xleftarrow{\$} \{0,1\}^n$. The protocol is secure if:

$$|\Pr[E(\tau, k) = 0] - \Pr[E(\tau, U_n) = 0]| \leq \text{negl}(n)$$

  where the probability is taken with respect to the coins of the entire experiment (U_n and the randomness of Alice, Bob, and Eve).

2.1 DH key exchange protocol

Let G_q be a multiplicative group of prime order q, and let $g \in G_q$ be a generator. The protocol is the following:

1. A picks $x \xleftarrow{\$} \mathbb{Z}_q$, computes $X = g^x$, and sends X to B;
2. B picks $y \xleftarrow{\$} \mathbb{Z}_q$, computes $Y = g^y$, and sends Y to A;
3. A computes $k_A = Y^x = (g^y)^x = g^{xy}$;
4. B computes $k_B = X^y = (g^x)^y = g^{xy}$.

This protocol satisfies the definition of a KEP.

- **Correctness**: clear from the protocol.
- **Security**: Eve is given $X = g^x$, $Y = g^y$, and has to distinguish $k = g^{xy}$ from a uniformly random U_n. Suppose Eve can distinguish them: then we can construct an algorithm B that can solve the DDH problem.

2.2 Active adversaries

Notice that throughout this section, we assumed that Eve can only observe messages, but not modify or interfere with them in any way. Suppose that we allow Eve to modify messages. Since Alice and Bob have no way of telling who they are interacting with, Eve can perform the KEP with Alice and Bob separately and then mediate the conversation between them. Alice and Bob will think that they are interacting with each other, but Eve can read (and modify) all of their messages. This is what is typically called a *man-in-the-middle attack* (MITM).