University of Illinois, Urbana Champaign
CS/ECE 498AC3/4 Applied Cryptography

Instructor: Dakshita Khurana
Scribe: Kyo Kim, Tzu-Bin Yan
Date: September 24, 2020

LECTURE
10

Public-Key Encryption 2

In this lecture, we will move on to learn about how hard problems, like the discrete logarithm introduced in the last lecture, can in fact be employed to construct what is called public-key encryption systems, which, contrary to what we have seen so far, does not require a shared private key among communicating parties. In the following text, we will first look into the definition and security of the public-key encryption scheme. We will then move on to see two examples for such a scheme, with each relying on hard problems stated as assumptions for their respective security.

Due to the relief of requiring shared secret key among communicating parties, public-key encryption has been widely employed in various real-world scenarios such as in both TLS, SSH, and the Pretty Good Privacy (PGP) protocol, which thus renders it essential for students to understand what public-key encryption is and more importantly, why they are secure from a formal perspective, to work our way toward provable security via cryptography.

10.1 Public-key encryption

As we left off with a high-level introduction to public-key encryption last lecture, let's start this lecture with a brief review of what public-key encryption is. Public-key encryption intuitively consists of three components: a key generation algorithm G to generate a public and private key pair (pk, sk) based on randomness, an encryption algorithm E taking the public-key pk, a plaintext message m and some randomness to generate a ciphertext c, and a decryption algorithm D taking the private key sk and a ciphertext c to recover the plaintext message m. We now introduce the formal definition of public-key encryption:

Definition 10.1. A **public-key encryption system** $\varepsilon = (G, E, D)$ is a set of three efficient algorithms:

- $G = Gen$ is a randomized algorithm generating the public and private key pair (pk, sk), i.e. $(pk, sk) \xleftarrow{R} G(.)$
- E is a randomized algorithm taking the public-key pk and message $m \in M$ to generate ciphertext $c \in C$, i.e. $c \xleftarrow{R} E(pk, m)$

- D is a deterministic algorithm taking the private key sk and ciphertext $c \in C$ to recover the plaintext $m \in M$ or $\perp$ (indicating failure of decryption for c), i.e. $m \leftarrow D(sk, c)$ or $\perp \leftarrow D(sk, c)$

that satisfies the correctness property, where $\forall(pk, sk)$ generated by G

$$(\forall m \in M)\ D(sk, E(pk, m)) = m$$

where a randomized algorithm indicates that the algorithm employs some randomness during operation, and M, C denotes the message (plaintext) space and ciphertext space for ε, respectively.

Now given that we have defined the public-key encryption scheme, let's look back to the key exchange problem last lecture, which we partially *solved* by introducing the Diffie-Hellman protocol. Observe that public-key encryption can also be utilized to solve the same problem by the following process: suppose there are two parties: Alice and Bob, who would like to establish a shared secret. Alice has already generated a public-private key pair (pk, sk) with pk being publicly known. For Alice and Bob to come up with a shared secret, it would be sufficient for Bob to pick some random secret x and then encrypt the random secret with Alice's public-key pk, getting $E(pk, x)$, before sending the encrypted secret onto the public communication channel to Alice. Alice will then be able to recover the secret x via decrypting with her secret key sk, thus allowing the two parties to establish the shared secret x. Observe that in this case, assuming semantic security (we will see what this means in a moment) of the public-key encryption system employed, passive attackers (i.e. attackers that cannot actively modify traffic in the public channel) will be shielded away from the chosen secret, thus allowing a secure key exchange process.

Similar to private key encryption, it is then essential for us to discuss the semantic security properties of public-key encryption. As one would guess, it is a bit similar to what we have seen in private key encryption, with a little bit of a twist due to the availability of the public-key to the adversary, as we will see in the next section.

10.2 Semantic security in public-key encryption

Public encryption semantic security can be defined in terms of the following experiment.

Definition 10.2. **Public-key encryption one-time attacker game**. Experiment $b \in \{0, 1\}$, denoted as $EXP_{one}(b)$, is defined using a challenger C and a efficient (i.e. PPT) adversary $\mathcal{A}$ where:

1. C has $b \in \{0, 1\}$ unknown to $\mathcal{A}$
2. C sends public-key, pk, to $\mathcal{A}$
3. $\mathcal{A}$ sends $m_0, m_1 \in M$ where M is the set of all possible messages and $|m_0| = |m_1|$
4. C chooses m_b and sends $E(pk, m_b)$ to $\mathcal{A}$
5. $\mathcal{A}$ predicts which message was encrypted as $b' \in \{0, 1\}$

At a high level, we see that there is a single round of interaction between the challenger and the adversary and that the experiment is used to measure how *good* the encryption is by measuring how well the adversary predicts.

As usual, we want the adversary to have no advantage in one experiment over the other as in the semantic security game of private key encryption. Therefore, we can define one-time semantic security as follows.

DEFINITION 10.3. Public-key encryption is **one-time semantic secure** if no PPT adversary can distinguish $EXP_{one}(0)$ and $EXP_{one}(1)$, with non-negligible probability, i.e. $\forall$ PPT adversary,

$$|Pr[b' = 1|EXP_{one}(0)] - Pr[b' = 1|EXP_{one}(1)]| < \epsilon_{one}$$

Now consider we modify the experiment such that multiple rounds of interaction occurs between the challenger and adversary.

DEFINITION 10.4. **Public-key encryption many-time attacker game**. Experiment $b \in \{0, 1\}$, denoted $EXP_{many}(b)$, is defined using a challenger C and a PPT adversary $\mathcal{A}$ where $i \in \mathbb{Z}^+$ of interactions occur:

1. C has $b \in \{0, 1\}$ unknown to $\mathcal{A}$
2. C sends public-key, pk, to $\mathcal{A}$
3. For $j = 0; j < i; j++$
   (a) $\mathcal{A}$ sends $m_0^j, m_1^j \in M$ where M is the set of all possible messages and $|m_0^j| = |m_1^j|$
   (b) C chooses m_b^j and sends $E(pk, m_b^j)$ to $\mathcal{A}$
4. $\mathcal{A}$ predicts which message was encrypted as $b' \in \{0, 1\}$

Similar to the one-time semantic security definition, we now define many-time semantic security using the multi-round game proposed:

DEFINITION 10.5. Public-key encryption is **many-time semantic secure** if no PPT adversary can distinguish $EXP_{many}(0)$ and $EXP_{many}(1)$, with non-negligible probability, i.e. $\forall$ PPT adversary,

$$|Pr[b' = 1|EXP_{many}(0)] - Pr[b' = 1|EXP_{many}(1)]| < \epsilon_{many}$$

The high-level idea is that no adversary should be able to effectively leverage any patterns in the multiple interactions with the challenger.

Interestingly, for public-key encryption, one-time semantic security and many-time semantic security are equivalent:

THEOREM 10.6. *Public-key encryption scheme is many-time semantic secure if and only if it is one-time semantic secure.*

To prove the theorem, we must prove the implications in both directions.

LEMMA 10.7. *Many-time semantic security implies one-time semantic security.*

Proof. Proof for this direction is straightforward as many-time semantic security is a generalization of one-time semantic security. □

LEMMA 10.8. *One-time semantic security implies many-time semantic security.*

The one-time to many-time part is harder to prove. Before we dive into the details, let's design a new game that will help us along the process.

DEFINITION 10.9. **m-modified public-key encryption many-time attacker game**. Experiment $b \in \{0,1\}$, denoted $EXP_{modm}(b)$, is defined using a challenger C_{modm} and an adversary $\mathcal{A}$, where $i \in \mathbb{Z}^+$ of interactions occur and $m \in \mathbb{Z}^+$, $m \leq i$:

1. C_{modm} has $b \in \{0,1\}$ unknown to $\mathcal{A}$
2. C_{modm} sends public-key, pk, to $\mathcal{A}$
3. For $j = 0; j < m; j++$
   (a) $\mathcal{A}$ sends $m_0^j, m_1^j \in M$ where M is the message space and $|m_0^j| = |m_1^j|$
   (b) C_{modm} sends $E(pk, m_0^j)$ to $\mathcal{A}$
4. For $j = m; j < i; j++$
   (a) $\mathcal{A}$ sends $m_0^j, m_1^j \in M$ and $|m_0^j| = |m_1^j|$
   (b) C chooses m_b^j and sends $E(pk, m_b^j)$ to $\mathcal{A}$
5. $\mathcal{A}$ predicts which message was encrypted as $b' \in \{0,1\}$

Notice that the difference between EXP_{modm} and EXP_{many} is that in the first m round of interaction, the challenger always sends the encrypted m_0^j for round j. Now to the real proof.

Proof. Let's use proof by contradiction to prove this direction. Suppose there is a public encryption system $\varepsilon = (G, E, D)$, that is one-time semantic secure but not many-time semantic secure. Therefore, there exist a PPT adversary $\mathcal{A}$ that can distinguish $EXP_{many}(\cdot)$. Consider the modified game 10.9 with m being 1 (i.e. $EXP_{mod1}(.)$). The argument that we want to construct here is that the assumed $\mathcal{A}$ that can break many-time semantic security of ε can also have a non-negligible advantage in distinguishing $EXP_{mod1}(\cdot)$:

CLAIM 10.10. *$\mathcal{A}$ that can break game 10.4 can defeat $EXP_{mod1}(.)$ with the non-negligible advantage $\epsilon_{many} - \epsilon_{one}$, i.e.*

$$|Pr[b' = 1|EXP_{mod1}(0)] - Pr[b' = 1|EXP_{mod1}(1)]| > \epsilon_{many} - \epsilon_{one}$$

The claim can be proved by contradiction, where a new adversary $\mathcal{B}$ capable of defeating the one-time semantic security game 10.2 (by having an advantage of $> \epsilon_{one}$) can be created by *simulating* EXP_{mod1} for $\mathcal{A}$, which cannot defeat EXP_{mod1}. The exact proof for this part is left for exercise.

Now consider the modified game 10.9 with m being 2 (i.e. $EXP_{mod2}(.)$). The argument that we want to construct here is that the assumed $\mathcal{A}$ that can break many-time semantic security of ε can defeat $EXP_{mod2}(.)$ as well:

Claim 10.11. *$\mathcal{A}$ that can break game 10.4 can defeat $EXP_{mod2}(.)$ with the non-negligible advantage $\epsilon_{many} - 2\epsilon_{one}$, i.e.*

$$|Pr[b' = 1|EXP_{mod2}(0)] - Pr[b' = 1|EXP_{mod2}(1)]| > \epsilon_{many} - 2\epsilon_{one}$$

The claim can also be proved by contradiction, which is left as an exercise. Note that we can in fact iteratively prove similar claims for m up to i, the maximum number of interaction rounds, which leads us to the final claim:

Claim 10.12. *$\mathcal{A}$ that can break game 10.4 can defeat $EXP_{modi}(.)$ with the non-negligible advantage $\epsilon_{many} - i\epsilon_{one}$, i.e.*

$$|Pr[b' = 1|EXP_{modi}(0)] - Pr[b' = 1|EXP_{modi}(1)]| > \epsilon_{many} - i\epsilon_{one}$$

However, observe that in game 10.9 with m being i, no information regarding b is actually present to $\mathcal{A}$, indicating that it is impossible for $\mathcal{A}$ to have an advantage in guessing b, i.e.

$$|Pr[b' = 1|EXP_{modi}(0)] - Pr[b' = 1|EXP_{modi}(1)]| = 0$$

With $\epsilon_{many} - i\epsilon_{one} >= 0$ by definition, we have a contradiction.

□

We thus have that the theorem 10.6 holds by lemma 10.7 and lemma 10.8.

10.3 El-Gamal encryption

After having formally defined public-key encryption, let's take a look into an example: the El-Gamal encryption. The El-Gamal encryption in some sense combines the Diffie-Hellman protocol with message encryption. Suppose there is a finite cyclic group G with a generator $g \in G$ (G and g publicly known), recall that in the Diffie-Hellman protocol, the two parties involved in the key exchange process would each pick some random secret $\in G$ (say a and b) and calculate the discrete exponentiation of the secret, i.e. g^a and g^b. The exponentiation results are then passed to the other party, where both parties utilize the obtained result and their respective chosen secret to, again, calculate the discrete exponentiation, i.e. $(g^a)^b = g^{ab} = (g^b)^a$. The g^{ab} is then the shared secret among the two parties.

In the El-Gamal encryption, we modify the Diffie-Hellman key exchange process by combining the establishment of the shared secret with the encryption of some plaintext. More concretely, consider the previous case of Alice and Bob in the Diffie-Hellman protocol example on G. In El-Gamal encryption, one of the party (say Alice), generate the public-private key pair (g^a, a) in advance, where g is a generator for G and $a \in G$ is Alice's secret key. For Bob to encrypt some message m for Alice, he will first generate some random secret $b \in G$ and calculate the "shared" secret $s = g^{ab}$ the same way via discrete exponentiation as in the Diffie-Hellman protocol. The secret is further hashed with some public hash algorithm H, and the resulting hash $H(s)$ is the actual key that is used afterwards for message encryption, which Bob employs AES with some appropriate mode of operation in this case. The resulting ciphertext Bob sends onto the network is the pair $(g^b, AES.Enc_k(m))$ where $k = H(s)$. For Alice to decrypt the ciphertext, Alice first derives the "shared" secret by performing discrete exponentiation against the first component g^b of the ciphertext, getting the secret $s = (g^b)^a = g^{ab}$. Alice then hashes the resulting s to get the actual key $k = H(s)$,

which can then be employed to decrypt $AES.Enc_k(m)$, and in turn obtain Bob's original message m.

In summary, the definition of the El-Gamal encryption is as follows:

DEFINITION 10.13. The El-Gamal encryption system (Gen, Enc, Dec) is a set of three efficient algorithms based on the following components:

- a cyclic group G of order n with a generator $g \in G$,
- the AES symmetric cipher $(AES.Enc, AES.Dec)$ defined over (K, M, C).
- a public hash function $H : G \to K$

where the message space for the system is M and the ciphertext space is $G \times C$. We now provide definition for (Gen, Enc, Dec)

- Gen generates the public and private key pair (pk, sk) via the following operations: $sk \xleftarrow{R} G$, $pk \leftarrow g^{sk}$.
- Enc takes pk and message $m \in M$ to generate the ciphertext (v, c) via the following operations: $b \xleftarrow{R} G$, $v \leftarrow g^b$, $s \leftarrow pk^b$, $k \leftarrow H(s)$ and $c \leftarrow AES.Enc_k(m)$.
- Dec takes sk and ciphertext (v, c) to recover the plaintext m via the following operations: $s \leftarrow v^{sk}$, $k \leftarrow H(s)$ and $m \leftarrow AES.Dec_k(m)$.

Note that the encryption algorithm we presented here is actually a modified version of the El-Gamal encryption algorithm. In the original El-Gamal encryption algorithm, no hashing of the "shared" secret was done, and AES was not present in the algorithm. Instead, Bob encrypts the message by simply XORing his message with the secret s, effectively using s as a one-time pad. The introduction of AES to increase efficiency is straightforward. On the other hand, the hashing of the secret is done to introduce randomness close to some uniform distribution to the key, as originally the secret only exhibits the unpredictable property and may follow some non-uniform distribution, which is not ideal.

After we have looked into how the El-Gamal encryption protocol works, a natural question follows: what kind of security does it provide? We thus look into the semantic security property of the El-Gamal encryption protocol, which we defined earlier for public-key encryption protocols. Here, we focus on the one-time semantic security property of the El-Gamal encryption protocol. By the claim 10.6, many-time semantic security is directly implied if one-time semantic security holds. Before we jump into the proof, let's first take a look at two assumptions that are central to the proof of one-time semantic security of the El-Gamal encryption protocol, i.e., the computational Diffie-Hellman assumption (CDH) and the hash Diffie-Hellman assumption (HDH).

DEFINITION 10.14. Given G being a finite cyclic group of order n, the computational Diffie-Hellman (CDH) assumption holds in G if $\forall$ efficient algorithms A, $Pr[A(g, g^a, g^a b) = g^{ab}] < \epsilon$, where $a, b \leftarrow Z_n$, g is a generator of the group G, and ϵ is negligible.

Observe that the CDH assumption captures the idea of it being hard for a passive attacker to recover the shared key in the Diffie-Hellman protocol. On the other hand, for us to prove the semantic security of the El-Gamal encryption, a slightly modified assumption is preferable:

DEFINITION 10.15. Given G being a finite cyclic group of order n, and a hash function $H : G \to K$ on some space K, the hash Diffie-Hellman (HDH) assumption holds for (G, H) if the pair $(g, g^a, g^b, H(g^{ab}))$ is (computationally) indistinguishable from (g, g^a, g^b, R), where $a, b \in Z_n$, g is a generator of Z_n and R is randomly sampled from K.

Not surprisingly, the two assumption is related:

THEOREM 10.16. *The HDH assumption holds implies the CDH assumption holds.*

Proof. Proving the above statement is the same as proving that the CDH assumption does not hold implies the HDH assumption does not hold. This new statement is straightforward as the CDH assumption does not hold indicates there exist some adversary that has a non-negligible probability of finding g^{ab}. We can thus use this adversary to construct a new adversary with a non-negligible advantage against the HDH games by simply applying hash to the output of the CDH adversary, and check the result against the hash given by the HDH game challenger. □

The idea behind the HDH assumption is that we want the case that even if the adversary has access to the public-keys g^a, g^b and the generator g, the adversary cannot differentiate $H(g^{ab})$ from the uniform random string, which is required for the semantic security of AES encryption used in the El-Gamal encryption protocol. Note that the hash function in the HDH assumption having the property to take in any input of G of arbitrary distribution and give outputs of K in a distribution that is indistinguishable to uniform distribution is called an extractor.

With the assumptions defined, we can then briefly show that the El-Gamal encryption is one-time semantic secure based on the HDH assumption.

THEOREM 10.17. *The El-Gamal Encryption Protocol on a finite cyclic group G and some hash function H is one-time semantic secure if the Hash Diffie-Hellman (HDH) assumption holds for (G, H).*

Proof. It is clear that proving one-time semantic security of the El-Gamal encryption protocol is the same as proving that the pair $(g^a, g^b, AES.Enc_k(m0))$ is indistinguishable from $(g^a, g^b, AES.Enc_k(m1))$ for any $m0, m1 \in M$, where M is the message space and $k = H(g^{ab})$. Suppose we have that the HDH assumption is true against (G, H) of the El-Gamal Encryption Protocol, we thus have that $(g^a, g^b, AES.Enc_k(m0))$ is indistinguishable from the pair $(g^a, g^b, AES.Enc_R(m0))$ and $(g^a, g^b, AES.Enc_k(m1))$ is indistinguishable from $(g^a, g^b, AES.Enc_R(m1))$, where R is some random string in key space K. It thus remain for us two show that $(g^a, g^b, AES.Enc_R(m0))$ is indistinguishable from $(g^a, g^b, AES.Enc_R(m1))$, which directly follows from the property of AES. □

Having seen the El-Gamal encryption example, let's move on to another well-known example of public-key encryption, namely the RSA system.

10.4 RSA

Recall the definition of inverse in a set.

DEFINITION 10.18. Inverse of $x \in \mathbb{Z}_N$ is $y \in \mathbb{Z}_N$ such that $(x \cdot y)(mod N) = 1$

As a result, an element is **invertible** if the inverse exists in the set. The following was proven in lecture 8.

FACT 10.19. *Let Z_N be a set $\{0, 1, \cdots, N-1\}$ and N be a product of two primes p, q. Then $x \in Z_n$ is invertible $\iff$ $gcd(x, N) = 1$ and the number of invertible elements in Z_N, $(Z_N)^*$, is $\phi(N) = (p-1)(q-1) = N - p - q + 1$*

Therefore, $\phi(N)$ counts the number of relative primes w.r.t. N in the set. Recall Euler's theorem:

THEOREM 10.20. $\forall x \in (Z_N)^*, x^{\phi(N)} = 1(modN)$

Using the above information, we can look at textbook RSA system defined as following:

DEFINITION 10.21. Textbook RSA is composed of three functions:

- $Gen(\cdot)$: choose random primes p, q and construct $N = p \cdot q$. Choose integers e, d such that $e \cdot d = 1(mod\phi(N))$. Create public-key, $pk = (N, e)$ and private key $sk = (N, d)$.
- $RSA\_Enc(pk, x) = x^e(mod\mathbb{Z}_N)$
- $RSA\_Dec(pk, y) = y^d(mod\mathbb{Z}_N)$

Why did we pick such e, d? Suppose we have plain-text x and create encrypted message $y = RSA\_Enc(pk, x) = x^e$. Lets decrypt y using the scheme above.

$$RSA_Dec(pk, x^e) = x^{ed} = x^{k\cdot\phi(N)+1}, k \in \mathbb{Z} \tag{10.1}$$

Notice that last equality in equation 10.1 is true because how e and d were picked.

$$x^{k\cdot\phi(N)+1} = (x^{\phi(N)})^k \cdot x = (1)^k \cdot x = 1, k \in \mathbb{Z} \tag{10.2}$$

Equation 10.2 uses Euler's theorem (theorem 10.20) for the second equality. As a result, we get back the x. The encryption and decryption process is deterministic. Do keep in mind that the Textbook RSA is insecure.

The generalization of this is called **trapdoor permutation**.

DEFINITION 10.22. Trapdoor permutation consists of three algorithms (G, F, F^{-1}).

- G: generates pk and sk
- $F(pk, x)$: easy to compute function that evaluates on x
- $F^{-1}(sk, y)$: Inverts the input y using sk

A secure trapdoor permutation is when inversing the $F(pk, \cdot)$ is hard without the "trapdoor" sk when pk and x are sampled at random.

DEFINITION 10.23. RSA Assumption: for all efficient adversary $\mathcal{A}$, $Pr[A(N, e, y) = y^{\frac{1}{e}}] < \epsilon$ where ϵ is negligible, p, q are randomly sampled n-bit primes, $N = p \cdot q$, $y \in \mathbb{Z}_N^*$ is the encrypted message.

Notice that $\mathcal{A}$ does not know p, q, only N. As a result, the hardness of the RSA assumption stems from the hardness of finding the prime factors of N. If the factoring was easy, then RSA can be broken. Factoring can be made easy using quantum computers or having a poor sampling method of picking p, q.

10.5 Summary

In this lecture, we looked into public-key encryption and the semantic security of public-key encryption. We have also discussed two examples of public-key encryption, *El-Gamal* and *RSA*, and assumptions such as CDH and HDH. The main takeaway is that the one-time semantic security holds for the public-key if and only if many-time semantic security holds.

Acknowledgement

These scribe notes were prepared by editing a light modification of the template designed by Alexander Sherstov.

References

[1] D. Khurana. Public key encryption 2. CS498 AC3/AC4 Lecture Slides, 2020.