Lecture 11: Key-Agreement

Notes by Yael Kalai

MIT - 6.5620
Lecture 11 (October 8, 2025)

> ***Warning:*** This document is a rough draft, so it may contain bugs. Please feel free to email me with corrections.

Recap

During the last two lectures we constructed signature schemes from collision-resistant hash functions. These lectures transitioned us from the world of symmetric cryptography to the world of public-key cryptography.

- We first constructed a one-time secure signature scheme from one-way function (Lamport's scheme). In this scheme the keys are longer than the messages!
- We then introduced the hash-then-sign paradigm, and used it to construct a one-time signature scheme for messages in $\{0,1\}^*$.
- Then we showed how to use a one-time secure scheme with long messages (messages of length two keys), togegher with a PRF, to construct a many-time secure signature scheme by generating a tree of keys (on the fly) and using a different key to sign each message.

Today

Today, we are heading in a tunnel that will take us to public-key encryption. In this tunnel we will learn how to agree on a secret-key while communicating over a public channel.

Key-Agreement

Our goal is for two parties, Alice and Bob, to agree on a secret key k without ever meeting, and while exchanging messages over a public channel. This may seem paradoxical at first! When Bob communicates with Alice, what distinguishes an eavesdropper Eve, from

Alice?? We will see how this can be done using some basic number theory.

We note that so far in class, we managed to do everything from only one-way functions: PRG, PRF, CPA-secure encryption, MAC, CCA-secure encryption, and even digital signature schemes!

Remark. Recall that our signature scheme construction relied on the existence of collision resistant hash functions, but actually this same construction can be based on one-way functions alone! The observation, due to Rompel [8], is that in the hash-then-sign, we could have used a "target collision resistant hash" (which is a weaker primitive than collision resistant hash), and can in turn be constructed from any one-way function.

While we do not have a proof that one-way functions exist (indeed we do not even have a proof that $P \neq NP$), we strongly believe they do. In Russell Impagliazzo's famous paper [4], introducing "Russell's five worlds," he introduces five possible worlds we may live in: Algorithmica, Heuristica, Pessiland, Minicrypt and Cryptomania, where one-way functions (and hence cryptography) only exist in Minicrypt and Cryptomania. Minicrypt contains only one-way functions, but no advanced cryptographic primitives (such as public-key encryption). Today we enter the world of Cryptomania, where public-key cryptography exists!

We will construct a key-agreement protocol under specific number theoretic assumptions. We do not know how to construct key-agreement from OWF or from collision resistant hash functions.

Group Theory Background

Our schemes use a group G of size roughly 2^λ, with the following properties:

1. Computing the group operation is easy (i.e., it is computable in time $\text{poly}(\lambda)$). Namely, given $g, h \in G$ it is easy to compute $g \cdot h$, where $\cdot$ denotes the group operation.

   Note that this implies that given $g \in G$ and given $x \in [|G|]$, it is easy to compute g^x via repeated squaring. Namely, compute

   $$g, g^2, g^4, g^8, \ldots, g^{2^\lambda}$$

   and then compute

   $$g^x = \prod_{i \in \{0,1,\ldots,\lambda\}:\ x_i = 1} g^{2^i}.$$

   where $x = \sum_{i=0}^{\lambda} x_i \cdot 2^i$

2. Given $g \in G$ it is easy to compute its inverse g^{-1}.
3. G is cyclic; namely, there exists a *generator* $g \in G$ such that $G = \{g^i\}_{i \in [|G|]}$.
4. The following problem, referred to as the *Discrete Log problem*, is hard: Given a pair (g, g^x) where $g \in G$ is a random generator and x is random in $[|G|]$, compute x. In other words the property is that for every poly-size $\mathcal{A}$ there exists a negligible μ such that for every $\lambda \in \mathbb{N}$,
$$\Pr[\mathcal{A}(g, g^x) = x] \leq \mu(\lambda)$$
where the probability is over a random generator $g \in G$ and a random $x \leftarrow [|G|]$.

Remark. Item 4 implies that the following is a one-way function for any generator $g \in G$:
$$f_{G,g}(x) = g^x,$$
where $f_{G,g} : [|G|] \to G$. In other words, the hardness of inverting $f_{G,g}$ does not come from the choice of the generator g.

The reason is that if there exists a generator $h \in G$ such that $f_{G,h}$ is easy to invert; i.e., there exists a poly-size $\mathcal{A}$ and a non-negligible $\epsilon : \mathbb{N} \to [0,1]$ such that for every $\lambda \in \mathbb{N}$,
$$\Pr_{x \leftarrow [|G|]}[\mathcal{A}_{G,h}(h^x) = x] \geq \epsilon(\lambda),$$
then there exists a poly-size $\mathcal{B}$ such that for every $\lambda \in \mathbb{N}$,
$$\Pr[\mathcal{B}(g, g^x) = x] \geq \epsilon^2(\lambda)$$
where the probability is over a random generator $g \in G$ and a random element $x \in [|G|]$.

$\mathcal{B}(g, g^x)$ does the following:

1. Compute $\alpha = f_{G,h}(g)$.
2. Compute $\beta = f_{G,h}(g^x)$.
3. Output $x = \beta \cdot \alpha^{-1} \bmod |G|$.

Note that
$$\Pr[h^a = g \ \wedge \ h^b = g^x] \geq \epsilon^2(\lambda)$$
which implies that
$$\Pr[x = \beta \cdot \alpha^{-1} \bmod |G|] \geq \epsilon^2(\lambda).$$

Examples An example of a group that we believe satisfies the above properties is $\mathbb{Z}_p^*$ for a random prime p. The inverse of an element $x \in \mathbb{Z}_p^*$ can be computed efficiently via the extended GCD algorithm. Another example is elliptic curves, which we will not discuss in this class.

In practice, elliptic curves are used more often since there are no non-trivial algorithms for solving the Discrete Log problem over elliptic curves, while there are non-trivial algorithms over $\mathbb{Z}_p^*$. For example, the Number Field Sieve algorithm for Discret Log, inverts f_G in time roughly $2^{\lambda^{1/3} \cdot (\log \lambda)^{2/3}}$. As a result when we use $\mathbb{Z}_p^*$ we need to use much larger security parameter, which results with schemes that are less efficient.

When working over $\mathbb{Z}_p^*$, we cannot choose p arbitrarily. For example, the Pohlig-Hellman algorithm solves the Discrete-Log problem in time $O(\sqrt{q})$ time where q is the largest prime factor of the order of group (which is $p-1$ in the case of $\mathbb{Z}_p^*$). That is, there are primes for which the Discrete-Log problem is easy. We believe that for a random prime p the Discrete-Log problem is hard.

Prime order groups In many applications, such as the ones we will see today, we need the group to be of prime order.[1] Note that the group $\mathbb{Z}_p^*$ is of order $p-1$,and hence is not a prime order group.

[1] The order of a group is the number of elements in the group.

Subgroup of $\mathbb{Z}_p^*$: Instead, we can take our group to be a sub-group of $\mathbb{Z}_p^*$ of prime order. Specifically, if $p = 2q+1$ where q is prime (such p is called a safe prime and such q is called Sophie-Germain prime), then we can take

$$G = \{x^2 : \ x \in \mathbb{Z}_p^*\}$$

be the set of all quadratic residues modulo p, and where the operation is multiplication modulo p. Note that $|G| = q$ is a prime order group. Any element $g \in G$ s.t. $g \neq 1$ is a generator, since the order of an element divides the order of the group.

The question is how do we find such a safe prime p? Well, we can choose a random number $p \leftarrow \{0,1\}^\lambda$ and check if it is a safe prime. If it is a safe prime then great! Otherwise, we try again.

While we do not even know if there are infinitely many safe primes, we do believe that they are "dense"; i.e., we believe that there are roughly $2^\lambda / \operatorname{poly}(\lambda)$ safe primes of size at most 2^λ. Therefore, we expect to find a safe prime after $\operatorname{poly}(\lambda)$ tries.

Another good news: We can efficiently test if a number is a safe prime, thanks to known primality tests [5, 6, 1] that check if a λ-bit number if prime in time $\operatorname{poly}(\lambda)$.

As mentioned above, in practice we often use elliptic curve groups of prime order.

The Diffie-Hellman Key-Exchange Protocol

Diffie and Hellman, in their very influential paper "New directions in Cryptography" [2], proposed the idea of public-key encryption and digital signature schemes. They did not construct such schemes but they did construct the following key-agreement protocol.

- Fix a multiplicative group G of prime order q (where q is of size roughly 2^λ), and let $g \in G$ be a generator of the group G.
- Alice samples a random element $a \leftarrow \{1, \ldots, q\}$ and sends $A = g^a$ to Bob.
- Bob samples a random element $b \leftarrow \{1, \ldots, q\}$ and sends $B = g^b$ to Alice.
- They agree on the secret key $k = g^{a \cdot b}$, where Alice computes this key by computing B^a and Bob computes this key by computing A^b.

Is this scheme secure? Yes, if you assume it is :-)

*Decisional Diffie-Hellman (*DDH*) Assumption*

Definition 1. The DDH assumption w.r.t. a group G prime order q with generator $g \in G$, states that

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

where $a, b, c \leftarrow \mathbb{Z}_q$.

Under this assumption, the Diffie-Hellman key-agreement, presented above, is secure against a poly-size eavesdropper (i.e., passive adversary).

The point is that now Alice and Bob can encrypt messages to each other using their secret key $k = g^{a \cdot b}$.

Remark. Another assumption commonly used in cryptography is the Computational Diffie-Hellman (CDH), w.r.t. a group G of a prime order group and a generator g, which asserts that for every poly-size $\mathcal{A}$ there exists a negligible function μ such that for every $\lambda \in \mathbb{N}$,

$$\Pr[\mathcal{A}(g^a, g^b) = g^{a \cdot b}] = \mu(\lambda).$$

Note that this assumption is not strong enough to make the key-agreement secure!

The DDH assumption is at least as strong as the CDH assumption, which is at least as strong as the Discrete-Log assumption.

As we will see, the Diffie-Hellman key agreement protocol, presented above, almost immediately gives us a public-key encryption scheme! However, this was observed only several years later by ElGamal [3] who used this key agreement protocol to construct a public-key encryption scheme. The first public-key encryption scheme was invented here at MIT by Rivest, Shamir and Adleman [7], two years after Diffie and Hellman's paper was published.

Public-Key Encryption Scheme

Definition 2. A public-key encryption scheme, w.r.t. message space $\mathcal{M} = \{\mathcal{M}_\lambda\}_{\lambda\in\mathbb{N}}$, consists of three PPT algorithms $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ with the following syntax:

- Gen is a PPT algorithm that takes as input the security parameter 1^λ (in unary) and outputs a pair of public and secret keys $(\mathsf{pk}, \mathsf{sk})$.
- Enc is a PPT algorithm that takes as input a public-key pk and a message $m \in \mathcal{M}_\lambda$ and outputs a ciphertext ct.
- Dec is a poly-time algorithm that takes as input a secret-key sk and a ciphertext ct and outputs a message m.

The encryption scheme is required to satisfy the following completeness guarantee:

Completeness: For every $\lambda \in \mathbb{N}$, and every $m \in \mathcal{M}_\lambda$,

$$\Pr[\mathsf{Dec}(\mathsf{sk}, \mathsf{Enc}(\mathsf{pk}, m)) = m] = 1$$

where the probability is over $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda)$ and over the randomness of Enc.[2]

[2] Sometimes this perfect correctness requirement is relaxed to be correct with probability $1 - \mathrm{negl}(\lambda)$.

Definition 3. A public-key encryption scheme $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ with message space $\mathcal{M} = \{\mathcal{M}_\lambda\}_{\lambda\in\mathbb{N}}$ is said to be semantically secure if for every poly-size $\mathcal{A}$ there exists a negligible μ such that for every $\lambda \in \mathbb{N}$ and every $m_0, m_1 \in \mathcal{M}_\lambda$,

$$\Pr[\mathcal{A}(\mathsf{pk}, \mathsf{Enc}(\mathsf{pk}, m_b)) = b] = \frac{1}{2} + \mu(\lambda).$$

A stronger definition is one where the adversary $\mathcal{A}$ can choose the two messages m_0 and m_1 as a function of pk.

Definition 4. A public-key encryption scheme $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ with message space $\mathcal{M} = \{\mathcal{M}_\lambda\}_{\lambda\in\mathbb{N}}$ is said to be CPA-secure if for every poly-size $\mathcal{A}$ there exists a negligible μ such that for every $\lambda \in \mathbb{N}$ the adversary $\mathcal{A}$ wins the game below with probability at most $\frac{1}{2} + \mu(\lambda)$:

1. The challenger samples $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda)$ and sends pk to $\mathcal{A}$.

2. $\mathcal{A}$ chooses $m_0, m_1 \in \mathcal{M}_\lambda$ and send these messages to the challenger.
3. The challenger samples a random bit $b \leftarrow \{0,1\}$ and computes $\mathsf{ct}_b \leftarrow \mathsf{Enc}(\mathsf{pk}, m_b)$. and sends ct_b to $\mathcal{A}$.
4. $\mathcal{A}$ outputs a bit b' (as a guess for b).

$\mathcal{A}$ wins if and only if $b' = b$.

Remark. Note that we do not need to give $\mathcal{A}$ oracle access to $\mathsf{Enc}(\mathsf{pk}, \cdot)$ since he can compute $\mathsf{Enc}(\mathsf{pk}, \cdot)$ on its own.

The El-Gamal Encryption Scheme

- Let $G = G_\lambda$ be a group of prime order q of size roughly 2^λ, and let $g \in G$ be a generator. The message space $\mathcal{M}_\lambda$ is G.
- $\mathsf{Gen}(1^\lambda)$:
  1. Choose at random $s \leftarrow \mathbb{Z}_q$.
  2. Set $\mathsf{pk} = g^s$ and $\mathsf{sk} = s$.
  3. Output $(\mathsf{pk}, \mathsf{sk})$.
- $\mathsf{Enc}(\mathsf{pk}, m)$:
  1. Choose at random $r \leftarrow \mathbb{Z}_q$.
  2. Output $\mathsf{ct} = (g^r, \mathsf{pk}^r \cdot m)$.
- $\mathsf{Dec}(\mathsf{sk}, \mathsf{ct})$:
  1. Parse $\mathsf{ct} = (R, \mathsf{ct}')$.
  2. Output $m = \mathsf{ct}' \cdot (R^{-\mathsf{sk}})$.

Theorem 5. *This scheme is* CPA *secure under the* DDH *assumption.*

References

[1] Manindra Agrawal, Neeraj Kayal, and Nitin Saxena. Primes is in p. Technical Report TR02-004, Electronic Colloquium on Computational Complexity (ECCC), 2002. Preliminary version; journal version in Ann. of Math. 160 (2004), 781–793.

[2] Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6), 1976.

[3] Taher ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In G. R. Blakley and David Chaum, editors, *Advances in Cryptology — CRYPTO '84*, volume 196 of *Lecture Notes in Computer Science*, pages 10–18, Santa Barbara, CA, USA, 1984. Springer.

[4] Russell Impagliazzo. A personal view of average-case complexity. In *Proceedings of the Tenth Annual Structure in Complexity Theory Conference, Minneapolis, Minnesota, USA, June 19–22, 1995*, pages 134–147. IEEE Computer Society, 1995.

[5] Gary L. Miller. Riemann's hypothesis and tests for primality. *Journal of Computer and System Sciences*, 13(3):300–317, 1976.

[6] Michael O. Rabin. Probabilistic algorithm for testing primality. *Journal of Number Theory*, 12(1):128–138, 1980.

[7] Ronald L. Rivest, Adi Shamir, and Leonard Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, 1978.

[8] J. Rompel. One-way functions are necessary and sufficient for secure signatures. In *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing*, STOC '90, pages 387–394, New York, NY, USA, May 1990. Association for Computing Machinery.