CS6115: Structure versus Hardness in Cryptography Oct 06, 2021

Lecture 18 : Pairings and its Applications

Lecturer: *Shweta Agrawal* Scribe: *Anshu Yadav*

In this lecture, and following few lectures, we will study bi-linear maps (pairings) and its applications. Before defining bi-linear maps we will define some preliminary hard problems in groups. Let us begin with a motivating example.

Suppose two parties, Alice and Bob want to agree on a common secret key (which they can further use for exchanging secret messages using symmetric key encryption). But they are constrained to send any message over a public channel, where anyone can tap on the channel and get access to the messages sent over it. However, for simplicity, we assume that no one else, other than the legitimate parties can insert or modify the messages on this channel. Thus, we want the following: Alice sends some message, say m_A to Bob and similarly Bob sends some message m_B to Alice over a public channel; and then they both compute a common key K which, given m_A and m_B, appears random to anyone other than Alice and Bob. This problem is known as *Key Agreement Problem*. Before looking at a solution to this problem let us first define some preliminary hard problems in groups.

1 Preliminaries

Definition 1.1 (**Discrete Log Problem).** In a cyclic group $G = \langle g \rangle$ (g is a generator of G) of order $q > 2^\lambda$, the Discrete Logarithm problem is defined as follows. Given $g \in G$ and $h \leftarrow G$, find $x \in \mathbb{Z}_q$ such that $g^x = h$. We say that the Discrete Logarithm assumption holds in G if, for any probabilistic polynomial-time (PPT) algorithm $\mathcal{A}$, we have $\mathsf{Adv}^{\mathsf{DL}}_{\mathcal{A}}(\lambda) := \Pr[h = g^x \mid x \leftarrow \mathcal{A}(G, g, h), h \leftarrow G] \in \mathsf{negl}(\lambda)$, where the probability is taken over all coin tosses and $\mathsf{negl}(\lambda)$ denotes the set of negligible functions of $\lambda \in \mathbb{N}$.

Next we describe two related problems, called *Computational Diffie Hellman problem* and *Decisional Diffie Hellman problem*.

Definition 1.2 (Computational Diffie Hellman Problem)**.** The Computational Diffie-Hellman (CDH) problem in G is to compute g^{ab} given (g, g^a, g^b), with $a, b \leftarrow \mathbb{Z}_q$. We say that the CDH assumption holds in G if, for any PPT algorithm $\mathcal{A}$, it holds that $\mathsf{Adv}^{\mathsf{CDH}}_{\mathcal{A}}(\lambda) := \Pr[h = g^{ab} \mid h \leftarrow \mathcal{A}(G, g, g^a, g^b); a, b \leftarrow \mathbb{Z}_q] \in \mathsf{negl}(\lambda)$, where the probability is taken over all coin tosses.

Definition 1.3 (Decisional Diffie Hellman Problem)**.** The Decisional Diffie-Hellman problem in group G is to distinguish between two distributions $D_0 = \{(g, g^a, g^b, g^{ab}) \mid a, b \leftarrow \mathbb{Z}_q\}$ and $D_1 = \{(g, g^a, g^b, g^c) \mid a, b, c \leftarrow \mathbb{Z}_q\}$. We say that the DDH assumption holds in G if the distributions D_0 and D_1 are computationally indistinguishable, i.e. for all PPT $\mathcal{A}$,

$$\mathsf{Adv}^{\mathsf{DDH}}_{\mathcal{A}}(\lambda) := |\Pr_{x \leftarrow D_0}[\mathcal{A}(x) = 1] - \Pr_{x \leftarrow D_1}[\mathcal{A}(x) = 1]| \in \mathsf{negl}(\lambda), \text{ i.e.}$$

$$|\Pr[\mathcal{A}(g, g^a, g^b, g^{ab}) = 1 \mid a, b \leftarrow \mathbb{Z}_q] - \Pr[\mathcal{A}(g, g^a, g^b, g^c) = 1 \mid a, b, c \leftarrow \mathbb{Z}_q]| \in \mathsf{negl}(\lambda).$$

1.1 Applications of DL, DDH and CDH Assumptions

Key Agreement Protocol We return to the problem of key agreement that we defined in the beginning. The DDH assumption has its direct application in Diffie-Hellman key agreement protocol given by Diffie and Hellman in 1976 [DH76]. Recall that in key agreement problem, we want two remote parties Alice and Bob to setup a common random key K between them using a public channel. For correctness, we want that both Alice and Bob compute the same key and for security we want that K must be pseudorandom. The protocol is described as follows:

Alice $\rightarrow$ Bob : $m_A = g^a, a \leftarrow \mathbb{Z}_q$

Bob $\rightarrow$ Alice : $m_B = g^b, b \leftarrow \mathbb{Z}_q$

Alice computes key as $K = m_B^a = (g^b)^a = g^{ab}$;
Bob computes key as $K = m_A^b = (g^a)^b = g^{ab}$.

Correctness and Security: Both Alice and Bob compute the same key $K = g^{ab}$ which proves the correctness. Security also follows directly from the DDH assumption.

An important point to note here is that the protocol is secure only against passive adversary who can only view the messages sent over the public channel but cannot modify or inject its own messages.

ElGamal Encryption Another application of DDH assumption is the construction of a public key encryption scheme given by ElGamal [ElG85]. The scheme is defined as follows:

KeyGen(1^λ): On input 1^λ, the keygen algorithm does the following:

1. Chooses a group G of prime order $q > 2^\lambda$, a group generator g for G and $x \leftarrow \mathbb{Z}_q$.
2. Computes $h = g^x \mod q$ and outputs $\mathsf{pk} = (G, g, q, h)$ and $\mathsf{sk} = x$.

Enc(pk, M): To encrypt a message $M \in G$, the encryption algorithm does the following.

1. Samples $r \leftarrow \mathbb{Z}_q$ and outputs

$$\mathsf{ct} = (c_1, c_2), \text{ where } c_1 = g^r \text{ and } c_2 = M \cdot h^r.$$

Dec$(\mathsf{sk}, \mathsf{ct})$: To decrypt a ciphertext $\mathsf{ct} = (c_1, c_2)$ using secret key $\mathsf{sk} = x$, the decryption algorithm computes and outputs the following.

$$M = \frac{c_2}{c_1^x}$$

Correctness and Security: Correctness is straightforward:

$$\frac{c_2}{c_1^x} = \frac{M \cdot h^r}{(g^r)^x} = \frac{M \cdot g^{xr}}{g^{rx}} = M$$

For security we prove the following theorem:

Theorem 1. Assume that the DDH assumption holds in G. Then the ElGamal encryption scheme presented above satisfies IND-CPA security.

Proof. We can prove the above theorem using following hybrids:

Hybrid_0 : In this hybrid, the challenger encrypts message M_0. That is, the challenger sends $\mathsf{pk} = (G, g, q, g^x)$ to $\mathcal{A}$. $\mathcal{A}$ outputs two messages M_0 and M_1. The challenger sends challenge ciphertext as $\mathsf{ct} = (g^r, M_0 \cdot g^{xr})$ to $\mathcal{A}$.

Hybrid_1 : This hybrid is same as the previous one, except that in this hybrid, the challenger computes ct as $\mathsf{ct} = (g^r, M_0 \cdot g^c)$, where $c \leftarrow \mathbb{Z}_q$.

Hybrid_2 : This hybrid is same as the previous one, except that in this hybrid, M_0 is replaced by M_1, i.e. ct is computed as $\mathsf{ct} = (g^r, M_1 \cdot g^c)$, where $c \leftarrow \mathbb{Z}_q$.

Hybrid_3 : This hybrid is same as the previous one, except that in this hybrid, ct is a valid encryption of M_1, i.e. $\mathsf{ct} = (g^r, M_1 \cdot g^{xr})$.

Observe that our goal is to show that Hybrid_0 and Hybrid_3 are computationally indistinguishable. For this, it suffices to show that the consecutive hybrids are indistinguishable.

Claim: If any PPT adversary $\mathcal{A}$ can distinguish between Hybrid_0 and Hybrid_1 with advantage ϵ, then there exists an adversary $\mathcal{B}$ against DDH problem with advantage at least ϵ.

Proof. $\mathcal{B}$ is defined as follows:
Upon receiving a challenge (G, g, q, g^a, g^b, t) from the DDH challenger, $\mathcal{B}$ does the following:

1. Sets $\mathsf{pk} = (G, g, q, g^a)$ and invokes $\mathcal{A}$ with pk. $\mathcal{B}$ implicitly sets $\mathsf{sk} = a$.
2. Upon receiving challenge messages M_0 and M_1 from $\mathcal{A}$, $\mathcal{B}$ computes $\mathsf{ct} = (g^b, M_0 \cdot t)$ and sends ct to $\mathcal{A}$.
3. In the end, $\mathcal{A}$ outputs a bit d'. $\mathcal{B}$ returns the same bit to the DDH challenger.

We can observe that if $t = g^{ab}$, then $\mathcal{B}$ simulated Hybrid_0 and if $t = g^c$ for $c \leftarrow \mathbb{Z}_q$, then $\mathcal{B}$ simulated Hybrid_1 with $\mathcal{A}$. Therefore, advantage of $\mathcal{B}$ is equal to

$$\begin{aligned} &|\Pr[\mathcal{B} \text{ outputs } 1 \mid t = g^{ab}] - \Pr[\mathcal{B} \text{ outputs } 1 \mid t = g^c]| \\ &\qquad = |\Pr[\mathcal{A} \text{ outputs 1 in } \mathsf{Hybrid}_0] - \Pr[\mathcal{A} \text{ outputs 1 in } \mathsf{Hybrid}_1]| = \epsilon. \end{aligned}$$

□

- Hybrid_1 and Hybrid_2 are statistically indistinguishable because of the following arguments. The only difference between the two hybrids is that in Hybrid_1, $c_2 = M_0 \cdot g^c$, while in Hybrid_2, $c_2 = M_1 \cdot g^c$. For $d \in \{0, 1\}$, let $M_d = g^{m_d}$ for some $m_d \in \mathbb{Z}_q$. Then $M_d \cdot g^c = g^{m_d + c}$. Since, c is uniformly random, $m_d + c$ is also uniformly random in $\mathbb{Z}_q$ implying that $M_d \cdot g^c$ is uniformly random in G, regardless of whether $d = 0$ or $d = 1$. Hence, the two hybrids are statistically indistinguishable.
- Indistinguishability of Hybrid_2 and Hybrid_3 can be argued in the same way as that for indistinguishability between Hybrid_0 and Hybrid_1.

□

2 Pairings and its Applications

We start with some definitions.

Definition 2.1 (Pairings)**.** A pairing is a bilinear map $e : G_1 \times G_2 \to G_T$ for cyclic abelian groups G_1, G_2, G_T of order p (usually prime) such that:

1. $e(g_1^a, g_2^b) = e(g_1^b, g_2^a) = e(g_1, g_2)^{ab}$, $\forall\, g_1 \in G_1, \forall\, g_2 \in G_2, \forall\, (a, b) \in \mathbb{Z}_q$;
2. If p is prime, then $e(g_1, g_2) = 1_{G_T} \iff g_1 = 1_{G_1}$ and $g_2 = 1_{G_2}$.

We need to assume that the pairing is efficiently computable.

Remarks:

1. If $G_1 = G_2$, then e is called symmetric map.
2. If $G_1 = G_2 = G$, then DDH problem is easy in G because of the following observation: Given (G, g, p, g^a, g^b, g^c), where $a, b \leftarrow \mathbb{Z}_q$ and $c \leftarrow \mathbb{Z}_q$ or $c = ab$,
$$c = ab \iff e(g^a, g^b) = e(g, g^c).$$
3. Discrete Logarithm problem is not any harder in the groups G_1 or G_2 (regardless of whether $G_1 = G_2$ or not) than in G_T. For example, if $g, g_1 \in G_1$, then $\log_g(g_1) = log_{e(g,h)} e(g_1, h)$ for any $h \in G_2$, so that a Discrete Logarithm instance in G_1 is easily turned into a Discrete Logarithm instance in G_T . The same holds for a discrete log instance in G_2.

Now we define a computational assumption on pairings. For simplicity we consider $G_1 = G_2 = G$.

Definition 2.2 (Decisional Bilinear Diffie-Hellman (DBDH) Assumption)**.** The Decisional Bilinear Diffie-Hellman (DBDH) assumption holds in (G, G_T) if the distributions

$$D_0 = \{(g, g^a, g^b, g^c, e(g, g)^{abc} \mid a, b, c \leftarrow \mathbb{Z}_p\}, \text{ and}$$
$$D_1 = \{(g, g^a, g^b, g^c, e(g, g)^d \mid a, b, c, d \leftarrow \mathbb{Z}_p\}$$

are computationally indistinguishable. The advantage of a distinguisher can be defined as the distance between two probabilities, analogously to the DDH case as follows:

$$\begin{aligned}\mathsf{Adv}_{\mathcal{A}}^{\mathsf{DBDH}}(\lambda) = |\Pr[\mathcal{A}(g, g^a, g^b, g^c, e(g, g)^{abc}) = 1 \mid a, b, c \leftarrow \mathbb{Z}_p] \\ - \Pr[\mathcal{A}(g, g^a, g^b, g^c, e(g, g)^d) = 1 \mid a, b, c, d \leftarrow \mathbb{Z}_p]|\end{aligned}$$

The DBDH assumption holds in (G, G_T) if for all PPT $\mathcal{A}$, $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{DBDH}} \leq \mathsf{negl}(\lambda)$.

Observation:
We can observe that the DBDH assumption in groups (G, G_T) is a natural extension of the DDH assumption in a group G without pairings. DDH assumption says that given (g, g^a, g^b), g^{ab} is computationally indistinguishable from a random element in G. That is, it says that it is hard to compute even a single multiplication in the exponent. Analogously, pairings let us do one multiplication in the exponent (in other words, allows degree 2 term in the exponent) but again,

it is hard to go beyond one multiplication. That is, given $e : G \times G \to G_T$ and g^a, g^b, g^c for randomly chosen a, b, c, it is easy to compute $e(g,g)^{ab}$ or $e(g,g)^{bc}$ or $e(g,g)^{ac}$, but $e(g,g)^{abc}$ is computationally indistinguishable from a random element in G_T, i.e. by doing even one more multiplication in the exponent, we get a pseudorandom value.
The definition of bi-linear maps can in fact, be generalized to higher orders which allows more than one multiplication in the exponent and is called multi-linear maps. For bi-linear maps, there exist groups on elliptic curves for which we know of instantiations for pairings that are efficiently computable. However, for higher degrees, i.e. multilinear maps, even though there exist such maps, we don't know of any map that is efficiently computable and this is a big open problem in cryptography.

3-way Key Agreement Protocol using Pairings The Diffie-Hellman key agreement protocol naturally extends to a single round 3-way key agreement protocol using pairings and the DBDH assumption. In 3-way key agreement, three parties, A, B and C agree upon a common key by communicating through a public channel.
The protocol is defined as follows: Let $e : G \times G \to G_T$ be a symmetric bilinear map, with a generator $g \in G$. The three parties proceed as follows.

1. A chooses $a \leftarrow \mathbb{Z}_p$ and sends g^a to B and C.
2. B chooses $b \leftarrow \mathbb{Z}_p$ and sends g^b to A and C.
3. C chooses $c \leftarrow \mathbb{Z}_p$ and sends g^c to A and B.

When the protocol ends, A, B and C compute the shared key

$$K = e(g,g)^{abc} = e(g^b, g^c)^a = e(g^a, g^c)^b = e(g^a, g^b)^c,$$

which is pseudo-random to everyone except A, B or C under the DBDH assumption which says that $K = e(g,g)^{abc}$ is computationally indistinguishable from a random element of G_T given (g, g^a, g^b, g^c).

2.1 Identity Based Encryption

The concept of identity based encryption was first given by Shamir in 1984 [SHA84]. Identity based encryption is an extension of public key encryption (PKE), in which the public key can be any string of one's own choice. For example, one can choose her own identity as her public key and then get a corresponding secret key from some "key generating authority". This is different from public key encryption in which both the public key and the secret key are generated by the setup algorithm. The public key thus generated is long and does not generally have any structure and hence, hard to remember. Moreover, in case of PKE, we need some certification authority to certify the mapping between one's identity and his public key. In addition, we need a public repository which maps different identities with corresponding public keys so that anyone who wants to send some secret message to someone can get his public key from the repository. In case of IBE, since the public key of any user is her identity itself, we do not need any certificate or mapping. Thus motivation behind IBE is to simplify key management and minimize the use of certification authority.

Definition 2.3 (Identity Based Encryption)**.** An Identity Based Encryption (IBE) scheme is a tuple of algorithms (Setup, KeyGen, Enc, Dec) such that:

$\mathsf{Setup}(1^\lambda) \to (\mathsf{pp}, \mathsf{msk})$: Given a security parameter 1^λ, the setup algorithm outputs a pair $(\mathsf{pp}, \mathsf{msk})$. (This algorithm is run by an authority called Private-Key Generator (PKG)).

$\mathsf{KeyGen}(\mathsf{msk}, \mathsf{ID}) \to \mathsf{sk}_{\mathsf{ID}}$: Given msk and user's identity ID, the keygen algorithm outputs a secret key $\mathsf{sk}_{\mathsf{ID}}$ corresponding to ID. (this algorithm is also run by the PKG).

$\mathsf{Enc}(\mathsf{pp}, \mathsf{ID}, M) \to \mathsf{ct}$: Given pp, the receiver's identity ID and a plaintext M, the encryption algorithm outputs a ciphertext ct.

$\mathsf{Dec}(\mathsf{pp}, \mathsf{sk}_{\mathsf{ID}}, \mathsf{ct}) \to M'$ or $\perp$: Given pp, a private key $\mathsf{sk}_{\mathsf{ID}}$ and a ciphertext ct, the decryption algorithm outputs M' or an error symbol $\perp$ indicating a decryption failure.

Correctness:

$$\Pr\left[M \leftarrow \mathsf{Dec}(\mathsf{pp}, \mathsf{sk}_{\mathsf{ID}}, \mathsf{ct}) : \begin{array}{l} (\mathsf{pp}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda), \\ \mathsf{sk}_{\mathsf{ID}} \leftarrow \mathsf{KeyGen}(\mathsf{pp}, \mathsf{msk}, \mathsf{ID}), \\ \mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{pp}, \mathsf{ID}, M) \end{array} \right] \geq 1 - \mathsf{negl}(\lambda).$$

Security:
An IBE scheme satisfies IND-ID-CPA security if for all PPT adversary $\mathcal{A}$, for all $\lambda \in \mathbb{N}$, advantage of $\mathcal{A}$ is negligible in the following game.

Experiment $\mathsf{Expt}_{\mathcal{A}}^{\mathsf{IBE}}(\lambda)$:

1. **Setup phase:** The challenger generates $(\mathsf{pp}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda)$, gives pp to $\mathcal{A}$ and initializes a set Q.
2. **Key queries:** $\mathcal{A}$ adaptively corrupts polynomially many identities of its choice by repeating the following kind of queries:
   - $\mathcal{A}$ chooses an identity ID and obtains $\mathsf{sk}_{\mathsf{ID}} \leftarrow \mathsf{KeyGen}(\mathsf{msk}, \mathsf{ID})$ from the challenger.
   - The challenger updates $Q := Q \cup \{\mathsf{ID}\}$.

   Note that each query may depend on the answer to previous queries.
3. **Challenge Query:** $\mathcal{A}$ outputs two messages M_0, M_1 and an $\mathsf{ID}^* \notin Q$. The challenger chooses a bit $b \leftarrow \{0, 1\}$ and returns $\mathsf{ct}^* = \mathsf{Enc}(\mathsf{pp}, \mathsf{ID}^*, M_b)$ to $\mathcal{A}$.
4. $\mathcal{A}$ corrupts more identities under the restriction that ID^* can never be corrupted. Hence, it must hold that $\mathsf{ID}^* \notin Q$ at the end of the game.
5. $\mathcal{A}$ outputs its guess bit b' and wins if an only if $b' = b$.

Advantage of $\mathcal{A}$ is defined as $\Pr[b' = b] - 1/2$.

Construction First construction of IBE was given by Boneh and Franklin in 2001 [BF01], for which they won Gödel prize in 2013 (along with Antoine Joux) [ACM]].
Boneh and Franklin's construction uses pairings and is secure in random oracle model. Let us first understand the random oracle model. Then we will study the construction in next lecture.

Random Oracle Model Consider a family of functions $\mathcal{F} = \{f : \{0,1\}^n \rightarrow \{0,1\}^m\}$. For simple case of $m = 1$, each function has truth table of length 2^n bits. Thus, $|\mathcal{F}| = 2^{2^n}$. In general m is large so that for a randomly chosen function R_f from $\mathcal{F}$, the function value is unique (i.e. different at different points) with high probability at p many points, where $p = p(\lambda)$ is a polynomial in the security parameter λ. Observe that the truth table of any function in $\mathcal{F}$ is exponentially large and therefore, in general, it may not always be possible to efficiently represent a random function.
Random oracle model is a theoretical model in which we assume that everyone in the system has ”query access” to a random function R_f sampled from $\mathcal{F}$. By query access, we mean that anyone in the system can query the oracle for R_f on any input x of its choice and get y in response. Since, R_f is a random function, y is uniformly random in $\{0,1\}^m$. However, by the definition of a function, if different entities in the system query the random oracle for R_f on same input x, then they all receive the same value y.
In practice, the use of random oracles is justified by the existence of certain efficiently representable functions whose outputs are sufficiently randomized to look random. These functions generate pseudorandom outputs. For e.g. SHA family of hash functions is a good candidate (and mostly used in practice) for random functions.

References

[LEC01] Lecture notes by Benoît Libert, Scribed by Benjamin Hadjibeyli, CR01: Advanced Cryptographic Primitives, Lecture Notes, 2014.
`http://perso.ens-lyon.fr/damien.stehle/downloads/Course1.pdf`

[DH76] Diffie, Whitfield, and Martin Hellman. ”New directions in cryptography.” IEEE transactions on Information Theory, 1976.

[ElG85] ElGamal, Taher. ”A public key cryptosystem and a signature scheme based on discrete logarithms.” IEEE transactions on information theory, 1985.

[SHA84] Shamir, Adi. ”Identity-based cryptosystems and signature schemes.” Workshop on the theory and application of cryptographic techniques. Springer, Berlin, Heidelberg, 1984.

[BF01] Boneh, Dan, and Matt Franklin. ”Identity-based encryption from the Weil pairing.” CRYPTO, 2001.

[ACM] `https://www.acm.org/media-center/2013/may/acm-group-presents-godel-prize-for-advances-in-cryptography`