7.3.2 - Discrete Logarithm and Diffie-Hellmann Assumptions

- We now introduce a number of computational problems that can be defined for any class of cyclic groups

- These problems are fundamental for providing security of a public key encryption scheme.

Recall If G is a cyclic group of order q, then there exists a generator g s.t. $\{g^0, g^1, \ldots, g^{q-1}\} = G$. Then for $\forall h \in G$, $\exists x \in \mathbb{Z}_q$ s.t. $g^x = h$.

Def. The discrete logarithm of h with respect to g is $x = \log_g h$.

Note. Discrete logarithms obey the same properties as standard logarithm

$$\log_g(h_1 h_2) = \left(\log_g h_1 + \log_g h_2\right) \bmod q$$

$$\log_g(h_1 h_2) = x \leftrightarrow h_1 h_2 = g^x = g^{x_1 x_2} = g^{x_1} x_2$$

Intuition

The discrete logarithm problem in a cyclic group G with a given generator g is to compute $\log_g h$ given a random element $h \in G$ as input.

or $\boxed{\text{Solve } h = g^x \text{ given } h.}$

Try Example

Formally

Let $\mathcal{G}$ be a polynomial time algorithm.

Input: 1^n

Output: Cyclic group G order q where $\|q\| = n$ and contains generator g.

The discrete logarithm experiment $DLog_{A,\mathcal{G}}(n)$

1. Run $\mathcal{G}(1^n)$, obtain (G, q, g)
2. Choose $h \leftarrow G$ by choosing $x' \leftarrow \mathbb{Z}_q$ and set $h := g^{x'}$
3. A is given G, q, g, h and outputs $x \in \mathbb{Z}_q$.
4. Output 1 if $g^x = h$ Otherwise output 1

Def. The discrete logarithm problem is hard relative to G if for all PPT algorithms A, there exists a negligible function negl such that

$$\Pr\left[\mathrm{DLog}_{A,G}(n) = 1\right] \le \mathrm{negl}(n)$$

Related to the discrete logarithm problem is the computational Diffie-Hellman problem and the decisional Diffie-Hellman Problem.

We will not use computational Diffie-Hellman Problem but it is helpful to introduce first.

Note. The discrete logarithm problem relative to G is assumed to be hard

Computational Diffie-Hellman Problem

> Fix a cyclic group G and a generator $g \in G$.
> Given h_1, h_2, define $DH_g(h_1, h_2) = g^{\log_g h_1 \log_g h_2}$.

If $h_1 = g^{x_1}$ and $h_2 = g^{x_2}$ then $DH(h_1, h_2) = g^{x_1 x_2} = h_1^{x_2} = h_2^{x_1}$.

> Given randomly chosen h_1, h_2 compute $DH(h_1, h_2)$

Formally (Not in book

Let $\mathcal{G}$ be a group generating algorithm.

Input: 1^n

Output: $(G, q, \|q\|=n, g)$

order generator

The computational Diffie-Hellman problem is hard relative to $\mathcal{G}$ if

$\forall$ ppt A, $\exists$ $negl(n)$ s.t.

$$\Pr[A(G, q, g, h_1, h_2) = 1] \leq negl(n)$$

Where $A(G, q, g, h_1, h_2) = 1$ if attacker computes $DH_G(h_1, h_2)$.

Note

1. If discrete logarithm is easy in G, then CDH is easy.

Given h_1, h_2 first compute $x_1 = \log_g h_1$ then output the answer $h_2^{x_1}$

1.5 This is only known way of solving CDHP

2. If CDH is hard then discrete log is hard.

3. DLP is weaker assumption than CDHP? Open question - One way is known

~~[illegible]~~ Decisional Diffie-Helman Problem

Intuition

Given randomly-chosen h_1, h_2 and a candidate solution y decide wheter $y = DH_g(h_1, h_2)$ or whether y was chosen at random from G.

Formally

Let $\mathcal{G}$ be the group generating algorithm. Then DDH is hard relative to $\mathcal{G}$ if for all PPT Algorithms A there exists a negligible function negl such that

$$\left| \Pr[A(G, q, g, g^x, g^y, g^z) = 1] - \Pr(A(G, q, g, g^x, g^y, g^{xy}) = 1) \right| \leq$$

negl(n)

Where $x, y, z \in \mathbb{Z}_q$ are chosen uniformly.

<u>Note.</u>

- If CDH problem is easy for some group G, then DDH Problem is easy.
- Converse is not true. DDH being easy $\not\Rightarrow$ CDH is easy
- Since if CDH problem is easy then we can compute g^{ab} given (g, g^{ab}, g^a). Therefore we could determine if g^z is uniform or not!
- The assumption that DDH is hard is strongest assumption
- $\boxed{\text{DDH = Hard}} \rightarrow$ CDH is Hard $\rightarrow$ DLP is hard
- DLP = Hard $\xrightarrow{?}$ CDH is Hard $\xrightarrow{\text{Not true}}$ DLP Hard

9.1 - Limitations of Private-Key Cryptography

9.1.1 - Key-Management Problem

1. Private Key encryption allows us to communicate over an insecure channel. To do this users need keys!
2. But how do we share keys? We can't use a insecure channel!
3. Easy to see we have reached an impasse

Key distribution and Setup

1. Initial sharing of a private key, can be done with a Courier Service.

2. Another method would be for two parties to physically meet and generate a copy of the key.

3. However both of these solutions are restrictive: Either they are expensive or they do not scale to beyond 2 people

4. A partial solution

(4) Example.

Imagine a company with n employees, Each need a private key to encrypt messages with each other pairwise. $\frac{n!}{2!(n-2)!}$ $\frac{n!}{(n-1)!(n-n-1)!}$

Each employee would need to generate n-1 keys. Therefore there is $\binom{n}{2}$ number of keys. If a workplace had 100 employees each employee would need to keep track of 99 keys.

Limitations of private key encryption

1. Key management is difficult to deploy and maintain
2. Not possible to use private key encryption in all settings.

The public key Revolution

- Diffie-Hellman revolutionized cryptography.
- They introduced the idea of public key encryption by observing assymmetric problems: there are certain operations that can be carried out but not inverted.
- Easy to multiply primes: difficult to factor their product.
- They introduced encryption schemes were security is preserved even against an adversary that knows the key!
- These encryption schemes are called public key encryption schemes. (Asymmetric schemes.)
- In public key encryption schemes. the encryption key is called the public key while the decryption key is called the private key. encryption

Key distribution Solution

1. With public key encryption, Public keys may be posted Publically
2. Each user would then only need to keep track of their Private key.

three Public key primitives

1. Notion of public key encryption
2. Digital Signatures - Used to prevent any undetected tampering of signed message.
   - Verification is done by anyone who knows public key
   - Only ~~sender~~ Owner of private key can generate a digital signature.
   - Ex. Sign a document and send it to third party. Digital signature is proof Alice sent the document.

(5.) Interactive key exchange

- A method whereby parties who have do not have secret information can generate a private key by communicating over a open channel.

- "As if you and a friend stand on opposite sides of a room, you can shout messages to each other in a way that will allow you to generate a shared secret that no one else understands"

- Different than encryption, since it requires both parties to be online.

The Diffie-Hellman Key exchange

1. Based on difficulty of discrete Log Problem
2. Can be formally proven secure under decisional Diffie-Hellman problem

Intuition

Let $h_2 = g^y$ and thus $h_2^x = g^{yx} = g^{xy}$.

Likewise $h_1 = g^x$ and thus $h_1^y = g^{xy}$

Therefore $h_2^x = h_1^y$

- An adversary would not know x or y since this implies solving the discrete log problem.

- However what if attacker uses h_1, h_2 and computes $K = g^{xy}$? This is the Computational Diffie Hellman Problem.

- Does not guarantee Security. Since it may be possible to Distinguish g^{xy} from a random element.

- Therefore a strong enough assumption is that the output key g^{xy} is not distinguished from a random element

(Q.3) Diffie-Hellman Key exchange

- We will present the Diffie Hellman Key exchange and prove Security in presence of eavesdropping adversaries.

Definitions

(1) Consider two parties: Bob and Alice that run a protocol in order to exchange Keys.

(2) Denote protocol as π

(3) As input Bob and Alice use the security parameter 1^n and random coins for computations.

(4) Random coin. Input (1) output {1 or 0}. Denote r_A to alice random coin and r_B to be Bob's.

random strings

(5.) $\text{output}_{A,\pi}(1^n, r_A, r_B)$ and $\text{output}_{B,\pi}(1^n, r_A, r_B)$ denote the respective outputs of BoB & alice

$\text{output}_{A,B}(1^n, r_A, r_B)$ is output of running Π upon input

5. $\text{transcript}_\Pi(1^n, r_A, r_B)$ denotes transcript of all messages sent by Bob and Alice in an execution of Π. Output is an [illegible] key shared between Bob and Alice.

<u>Def.</u> A protocol Π for a key exchange is correct if $\exists$ a negligible function negl s.t. for every n,

$$\Pr[\text{output}_{A,\Pi}(1^n, r_A, r_B)] \neq \Pr[\text{output}_{B,\Pi}(1^n, r_A, r_B)] \leq \text{negl}(n)$$

<u>Def.</u> A key exchange protocol is secure in the presence of eavesdropping adversaries if for every PPT adversary Eve there exists a negligible function negl s.t.

$$\Pr[KE^{\text{eav}}_{Eve,\Pi}(n) = 1] \leq 1/2 + \text{negl}(n)$$

Where $KE^{cov}_{Eve,\Pi}(n)$ is:

1. Random strings r_A, r_B are chosen of an appropriate length
2. $b \leftarrow \{0,1\}$ is chosen; if $b=0$ set $K = \text{output}_{A,\Pi}(1^n, r_A, r_B)$

   if $b=1$ set $K \leftarrow \cancel{\text{output}_{B,\Pi}(1^n, r_A, r_B)}$ $\{0,1\}^n$

3. Adversary Eve is given 1^n, $\text{transcript}_\Pi(1^n, r_A, r_B)$ and key k and outputs b'.

4. Output 1 if $b=b'$, OW output 0.

<u>Intuition</u>. Can A distinguish between output key and completely random key?

Construction 9.3 Diffie-Hellman Key exchange

Input: Security parameter n

Protocol

1. Alice generates a Group G and generator g using $\mathcal{G}$ with input 1^n and sends the result to Bob. Let m be the order of g.

2. Alice chooses a index $x \in \{1, \ldots, m-1\}$ and computes $h_1 = g^x$. Alice sends h_1 to Bob.

3. Bob chooses a random $y \in \{1, \ldots, m-1\}$ and computes $h_2 = g^y$. Bob sends h_2 to Alice

4. Alice outputs computes $K = h_2^x$

5. Bob outputs $K = h_1^y$

Thm. Assuming the decisional Diffie-Hellman problem is hard relative to group Generation $\mathcal{G}$, the Diffie Hellman Key exchange is correct and secure in the presence of eavesdropping adversaries