CS276 Lecture 18 Scribe Notes

Scribe: Gautam Altekar

1 The Discrete Log Problem

Definition (discrete log). Let G be a cyclic group of order n, and let g be a generator for G. In other words, $\langle g \rangle = \{g^i : i \in Z/nZ\} = G$. Then we know that for every $y \in G$, there is a unique element $x \in Z/nZ$ such that $g^x = y$. We call x the discrete log of y to base g.

The exponentiation function $f : Z/nZ \to G$ given by $f(x) = g^x$ is not a trap-door one-way permutation (TOWP) because the domain and range are not the same. In particular, it maps from Z/nZ to G. However, it is one-to-one and onto (i.e., bijective).

Definition (discrete log problem). If we let the group G be cyclic and of prime order, then it is conjectured that the map $f : x \mapsto g^x$ is one-way, meaning that it is easy to compute, but hard to invert. The problem of finding x given g^x in a reasonable amount of time is called the discrete log problem.

An example of a cyclic group for which the discrete log problem is though to be hard is $G = (Z/pZ)^*$, where p is prime. Unlike the RSA function, the discrete log function doesn't have a trapdoor. And just like factoring, we have no proof that computing the discrete log is hard. However, it is interesting that the runtime of the best-known factoring algorithm is exactly the same as the best-known discrete log algorithm. Furthermore, new tricks in factoring typically apply to the discrete log problem. There seems to be a connection between the two problems, but no one knows what that connection is.

2 The Diffie-Hellman Cryptosystem

Diffie and Hellman are credited for discovering the first public-key cryptosystem.

Diffie-Hellman Encryption Scheme (DH). Let g be a generator for group G, let g^x be Alice's public key, and let g^y be Bob's public key. Diffie-Hellman encryption proceeds as follows. First, Alice and Bob exchange public keys. Once Alice has obtained Bob's public key g^y, she creates the encryption key by raising Bob's public key to the x-th power, yielding key $k = (g^y)^x = g^{xy}$. Similarly, once Bob has obtained Alice's public g^x, he creates the encryption key by raising Alice's public key to the x-th power, yielding key $k = (g^x)^y = g^{xy}$. After this key-exchange procedure, both parties can encrypt and decrypt using k since both of them know it. More precisely, the encryption function $E(m) = (g^y)^x \oplus m$ and decryption function $D(\alpha) = \alpha \oplus (g^x)^y$, where α is the cipher-text. The values g^x, g^y are used only once and new values must be chosen each time you want to send another message.

Understanding Diffie-Hellman. Since the channel between Alice and Bob is assumed to be public, an eavesdropper can easily obtain the values of g^x and g^y during the initial public-key exchange phase. It would be easy for the adversary to compute $g^x \cdot g^y = g^{x+y}$. But that would do her little good since the key is g^{xy}. One way in which the adversary can obtain xy is to solve the discrete log problem for g^x and g^y, then multiply the results together. However, our assumption here is that finding the discrete log is hard and therefore the adversary shouldn't be able to find xy in a practical amount of time.

2.1 Hardness Assumptions

We assume that the discrete log is hard to compute. But is that assumption sufficient to ensure that Diffie-Hellman will be secure? The answer is no. There are additional hardness assumptions that we need to make in order to convince ourselves of the security of DH.

1. **Discrete Log (DL).** The assumption here is that given g and g^x where $x \xleftarrow{\$} Z/qZ$, it is hard for an adversary to find x. More formally, there exists a negligible ϵ such that

$$\text{Adv A} = \Pr[x \xleftarrow{\$} Z/qZ; A(g^x) = x] \leq \epsilon$$

for all adversaries A using practical amount of resources.

2. **Computational Diffie-Hellman (CDH).** The assumption here is that given g, g^x, and g^y where $(x, y \xleftarrow{\$} Z/qZ)$, it is hard for an adversary to find g^{xy}. More formally, there exists a negligible ϵ such that

$$\text{Adv A} = \Pr[x \xleftarrow{\$} Z/qZ; y \xleftarrow{\$} Z/qZ; A(g^x, g^y) = g^{xy}] \leq \epsilon$$

for all adversaries A using a practical amount of resources.

3. **Decisional Diffie-Hellman (DDH).** The assumption here is that given g, g^x, g^y, g^z where $(x, y \xleftarrow{\$} Z/qZ)$ and where $z = xy$ or $z \xleftarrow{\$} Z/qZ$, it is hard for an adversary to decide if she is looking at $z = xy$ or $z \xleftarrow{\$} Z/qZ$. More formally, there exists a negligible ϵ such that

$$\text{Adv A} = |\Pr[A(g^x, g^y, g^{xy}) = 1] - \Pr[A(g^x, g^y, g^z) = 1]| \leq \epsilon$$

for all adversaries A using a practical amount of resources.

2.2 Relationships Between Assumptions

How do these hardness assumptions relate? If assumption (1) is broken, then assumption (2) is broken. This is because it would be easy for the adversary to compute (DLog(g^x) $\times$ DLog(g^y)) if (1) is broken. The converse does not apply. Specifically, if (2) is broken, then (1) is not necessarily broken since (2) is a weaker assumption than (1).

If assumption (2) is broken, then assumption (3) is also broken. An adversary could compute g^{xy} and return 1 when $g^z = g^{xy}$, giving it a high advantage. The converse does not apply: if (3) is broken, then (2) is not necessarily broken since (3) is a weaker assumption than (2).

DDH is easy to solve in some groups. We will show that solving DDH in the group $(Z/pZ)^*$ is easy by constructing an adversary A such that

$$\text{Adv A} = |\Pr[A(g^x, g^y, g^{xy}) = 1] - \Pr[z \xleftarrow{\$} Z/qZ; A(g^x, g^y, g^z) = 1]| = 1/2.$$

Given input (α, β, γ), A does the following:

1. if $\text{lsb}(\text{DLog}(\alpha) \times \text{lsb}(\text{DLog}(\beta)) = \text{lsb}(\text{DLog}(\gamma)) \pmod 2$ return 1

2. else return 0

Observe that if $\gamma = g^z$, then A returns 1 with probability 1/2. But if $\gamma = g^{xy}$, then A will always return 1. Therefore, A's advantage is $|1 - 1/2| = 1/2$.

Now, the question is, how does A compute the lsb of the discrete log? After all, our assumption is that the discrete log problem is one-way and therefore it shouldn't be possible to easily guess a hard-core predicate. It turns out there is a trick that we can use in this situation. The trick involves the use of the Legendre symbol, which we define next.

Definition (Legendre symbol). Let p be a prime. The Legendre symbol is defined as

$$\left(\frac{x}{p}\right) = \begin{cases} 1 & \text{if } x \text{ is a square (i.e., quadratic residue),} \\ 0 & \text{if } x = 0 \pmod p, \\ -1 & \text{if } x \text{ is a non-square.} \end{cases}$$

It turns out that we can compute $\left(\frac{x}{p}\right)$ efficiently in $(Z/pZ)^*$ as

$$\left(\frac{x}{p}\right) = x^{(p-1)/2} \pmod p,$$

assuming $x \neq 0$. Thus, if x is a square, then $x = w^2 \implies (w^2)^{(p-1)/2} = w^{p-1} = 1$. If x is not a square, then $x^{(p-1)/2} = -1$. Now observe that

$$\left(\frac{g^x}{p}\right) = \begin{cases} 1 & \text{if } x \text{ is even,} \\ -1 & \text{if } x \text{ is odd.} \end{cases}$$

In other words, if g^x is a square, then x must be divisible by 2 and therefore even. Thus, even if we don't know what x is, we can recover its lsb, implying that lsb(x) is not a hardcore bit.

Making DDH hard to solve. How can we fix DDH so that it is hard to solve? As a start, we can make it hard for an adversary to employ the Legendre trick. There are several ways to do this, but we outline only two of them here. Both of them involve looking at subgroups of $\langle g \rangle$.

1. Let p and q be primes such that $p = 2q + 1$. Pick subgroup $\mathrm{QR}_p = \{x \in (Z/pZ)^* : \left(\frac{x}{p}\right) = 1\}$ (i.e., the subgroup of quadratic residues). QR_p has $(p-1)/2 = q$ elements.

2. Let p and q be primes such that q divides $p - 1$. Pick $G = \{x^{(p-1)/q} : x \in (Z/pZ)^*\}$. G has q elements.

In general, a cyclic group of prime order precludes the Legendre tricks. It is commonly conjectured that DDH is secure when G has q elements where q is a prime.

Security of Diffie-Hellman. We need the DDH assumption to hold to hold in order for Diffie-Hellman to be secure. For example, knowing that the DL assumption holds is not sufficient since an adversary might be able to break the CDH assumption through some means other than computing the discrete log. Similarly, knowing that the CDH assumption holds is not sufficient since an adversary might be able to break DDH through a means that doesn't require an effective adversary for CDH.

3 The El Gamal Cryptosystem

El Gamal Encryption Scheme (EG). Let G be a cyclic group and let g be a generator of G of prime order q. El Gamal is very similar to Diffie-Hellman in the sense that both parties exchange their public keys g^x and g^y. More precisely, to encrypt a message m to someone whose public key is g^x, we compute $E(m)$ as follows:

1. $r \xleftarrow{\$} Z/qZ$.

2. return $(g^r, (g^x)^r \cdot m)$

Let $\alpha = g^r$ and $\beta = (g^x)^r \cdot m$. The El Gamal decryption function $D(\alpha, \beta)$ works as follows:

1. return β/α^x.

El Gamal is like Diffie-Hellman, in that $\alpha^x = (g^r)^x$ is the shared key for this message. However, one difference is that in El Gamal, the recipient's public key g^x stays constant, while the sender's value g^r changes for every message. Another difference is that El Gamal uses multiplication rather than xor to mask the message m.

Properties of El Gamal.

1. If m, m' are any two messages, then

$$\begin{aligned} E(m) \cdot E(m') &= (g^r \cdot g^{r'}, (g^y)^r m \cdot (g^y)^{r'} m') \\ &= (g^{r+r'}, (g^y)^{r+r'} \cdot (m \cdot m'))) \\ &= E(m \cdot m') \end{aligned}$$

   This can be done even without knowing m and m'. The receiver could, for example, select a subset of the incoming ciphertexts, multiply them together, and send them back. The sender can decrypt to obtain the product of the corresponding plaintexts.

2. For any message m, we have

$$E(m) \cdot E(1) = E(m)$$

   This says that if you receive an encryption of m, then without knowing m, you can re-randomize the encryption to get another valid and independent encryption of m.