

1 Topics Covered

- Public Key Encryption
- A Public Key Encryption from the DDH Assumption
- El Gamal Encryption
- CRHF from Discrete Log
- PRG from DDH

2 Recall

Recall the three number theoretic assumptions we saw last time. We will build Cryptographic schemes or protocols based on the hardness of these problems.

DEFINITION 1 $(G, g, q) \leftarrow \text{Groupgen}(1^n)$ ◇

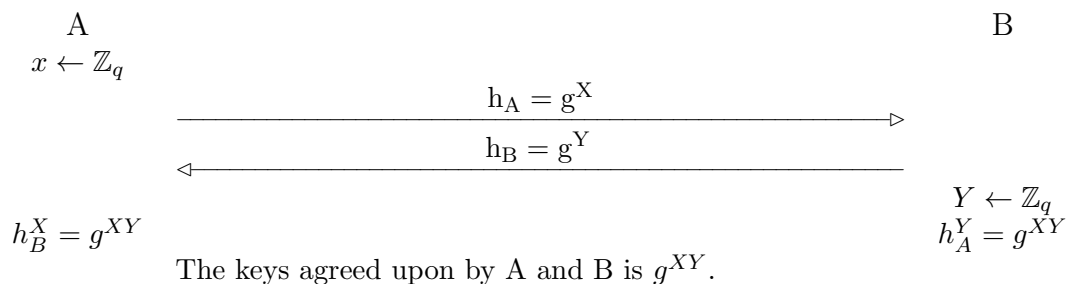
Assumption 1 DL Given g, g^X , it is hard to find X .

Assumption 2 Computational Diffie Hellman Given g, g^X, g^Y , it is hard to find g^{XY} .

Assumption 3 Decisional Diffie Hellman Given g, g^X, g^Y , it is hard to distinguish between g^{XY} and g^Z , where Z is chosen at random.

$$(g, g^X, g^Y, g^{XY}) \approx (g, g^X, g^Y, g^Z)$$

3 Key Agreement from the Diffie Helman scheme



It is interesting to note that in this scheme, A and B were able to agree upon a key without communicating about it. Each party generates a puzzle uniformly at random: A generates $h_A = g^X$, and B generates $h_B = g^Y$. Then, they send their puzzles to each other, and establish the key to be g^{XY} . Proving this scheme is secure is equivalent to showing that the DDH assumption holds.

4 Public Key Encryption

The general syntax of Public Key Encryption is the following. There will be two keys: one public key p_k and a private or secret key s_k . Any sender encrypts the message using the public key of the receiver. The receiver decrypts the message using her own secret key. The private key p_k defines a message space \mathcal{M}_{p_k} .

$$\begin{aligned}(p_k, s_k) &\leftarrow \text{Gen}(1^n) \\ c &\leftarrow \text{Enc}(p_k, m) \\ m &\leftarrow \text{Dec}(s_k, c)\end{aligned}$$

Correctness: For correctness, we must satisfy the condition as follows, that decoding of a valid encryption is always correct:

$$\forall (p_k, s_k) \in \text{Gen}(1^n), \forall m \in \mathcal{M}_{p_k},$$

$$\Pr [\text{Dec}(s_k, \text{Enc}(p_k, m)) = m] = 1$$

Security: To show the security of Public Key Encryption, we define the following experiment.

$$\text{Exp}_A^b(1^n) :$$

$$\begin{aligned}(p_k, s_k) &\leftarrow \text{Gen}(1^n) \\ (M_0, M_1) &\leftarrow A(1^n, p_k), \text{ where } M_0, M_1 \in \mathcal{M}_{p_k} \\ c &\leftarrow \text{Enc}(p_k, M_b) \\ b' &\leftarrow A(c)\end{aligned}$$

The adversary can read two(2) messages M_0, M_1 , and is trying to determine which experiment is current, that is, tries to distinguish between the encryption of them. That is, given M_b , it attempts to find out whether $b \stackrel{?}{=} 0, 1$. It outputs b' and wins the game if and only if $b = b'$.

We shall prove the security of this game by showing that the experiments Exp^0 and Exp^1 are computationally indistinguishable. Given a vector of messages, the argument goes via a hybrid argument. That is,

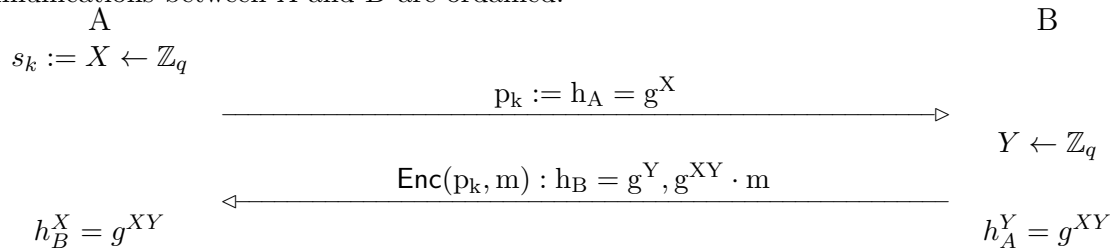
$$\text{Exp}^0 \approx \text{Exp}^1 \Rightarrow \forall \text{PPTA},$$

$$|\Pr [\text{Exp}_A^0(1^n) = 1] - \Pr [\text{Exp}_A^1(1^n) = 1]| = \text{negl}(n)$$

Remark 1 *If the Encoder Enc is deterministic, it is easy for the adversary to distinguish between $\text{Enc}(M_0), \text{Enc}(M_1)$. Since the encoder is public, the adversary does not need a random oracle to encode the messages. The adversary can invoke the encoder and encode the messages and compare with M_b . Therefore, we see that the Enc must be randomized.*

5 Public Key Encryption from DDH

We can use the DDH assumption to build a public key encryption as follows, by a minor modification of the key exchange protocol we saw before. The following protocol of communications between A and B are ordained:



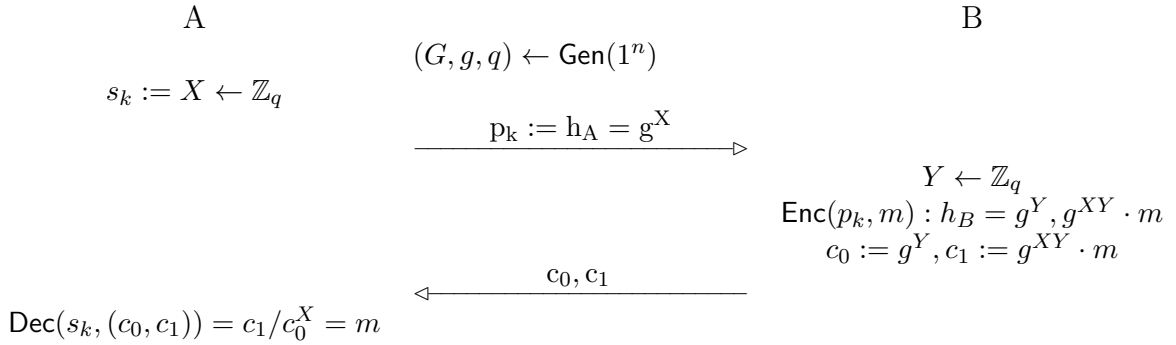
Thus, A the recipient first selects its secret key s_k by a random sampling, and builds the public key p_k , which it communicates to the sender B. Then, the sender B generates a random sample Y , using which and the public key p_k , it encrypts the message m and sends over to A. Note that the recovery of s_k from p_k is subject to the DL hardness assumption.

6 El Gamal Encryption

From the DDH based scheme we get the El Gamal public key cryptography scheme.

$$\begin{aligned}
 (G, g, q) &\leftarrow \text{Gen}(1^m) \\
 X &\leftarrow \mathbb{Z}_q \\
 s_k &:= X \\
 p_k &:= g^X = h_A \\
 \text{Enc}(p_k, m) &: Y \leftarrow \mathbb{Z}_q \text{ and } (g^Y, h_A^Y \cdot m) \\
 c_0 &:= g^Y, c_1 := h_A^Y \cdot m \\
 s_k &:= X \\
 \text{Dec}(s_k, (c_0, c_1)) &= c_1/c_0^X = g^{XY} \cdot m/g^{XY} = m
 \end{aligned}$$

This is essentially the same as the key exchange scheme as modified before. We can rewrite this in the same framework of the Diffie Helman Key exchange scheme as before.



As before, A selects a secret / private key s_k and sends across the public key p_k . We prove the security of the scheme by the following hybrid argument.

$$\begin{aligned} \text{Exp}^0 : g, p_k = g^X, c = (g^Y, g^{XY} m_0) \\ H : g, p_k = g^X, c = (g^Y, g^Z \cdot m_0) \\ \text{Exp}^1 : g, p_k = g^X, c = (g^Y, g^{XY} \cdot m) \end{aligned}$$

Here, $\text{Exp}^0 \approx H \approx \text{Exp}^1$

This hybrid argument is also a form of reduction. We use the fact that: $g^Z \cdot m_0 \approx g^Z$, which is essentially the fact that a totally random quantity multiplied by anything arbitrary will give something that is still totally random.

7 CRHF from DL

We will build Collision Resistant Hash Function from the Discrete Log hardness. We use a cyclic group G of prime order q . **SeedGen** is an oracle that generates a purely random seed. That is, the hash family contains hash functions indexed by the seed s generated by **SeedGen**. Such a Hash Function H_s maps the domain D_s to the range R_s

$$\begin{aligned} s &\leftarrow \text{SeedGen}(1^n) \\ H_s &: D_s \rightarrow R_s \end{aligned}$$

Security: The guarantee that collision is highly unlikely is given by the following statement which is akin to the security statement of the public key encryption schemes.

$\forall \text{PPTA}$:

$$\Pr[x \neq x' \in D_s : s \leftarrow \text{SeedGen}(1^n), x, x' \leftarrow A(1^n, s)] = \text{negl}(n)$$

7.1 Construction

The construction is described below.

$$\begin{aligned}
s &= (g, h = g^X) \\
x &\leftarrow \mathbb{Z}_q \\
H_s &: \mathbb{Z}_q^2 \rightarrow G \\
H_s(a, b) &= g^a \cdot h^b
\end{aligned}$$

Suppose the adversary gives you $a, \neq b$, with the same hash.

Then, $x = (a, b) \neq x' = (a', b')$

$$g^a h^b = g^{a'} h^{b'}$$

$$g^{(a-a')/(b'-b) \bmod q} = h$$

$$g^z = h, z = (a - a')/(b' - b)$$

Security comes directly from the definition of DL security assumption.

8 Pseudo-random Generators from DDH

We can also build Pseudo-random Generators from the Decisional Diffie Helman assumption.

PRG from DDH:

$$\begin{aligned}
(G, g, q) &\leftarrow \text{Gen}(1^n) \\
x &\leftarrow \mathbb{Z}_q \\
y &\leftarrow \mathbb{Z}_q \\
\text{PRG}_g(x, y) &= [g^x, g^y, g^{xy}] \\
\text{PRG} &: \mathbb{Z}_q^2 \rightarrow G^3
\end{aligned}$$

Here, x, y are randomly sampled from \mathbb{Z}_q , where q is a prime. From 2 such uniformly picked random values, PRG_g produces an extra bit g^{xy} , that is computationally indistinguishable from a random element of the group G . It follows directly from the DDH assumption that this is a good PRG.

Also, we can extend the PRG with stretch of l as follows, for any given l :

$$\begin{aligned}
\text{PRG}_g(X, Y_1, \dots, Y_l) &= [g^X, g^{Y_1}, g^{XY_1}, g^{Y_2}, g^{XY_2} \dots g^{Y_l}, g^{XY_l}] \\
\mathbb{Z}^{l+1} &\rightarrow G^{2l+1}
\end{aligned}$$

8.1 Security

We prove the security of this construction by a hybrid argument as follows.

$$\begin{aligned}
H^0 &= g, g^X, g^{Y_1}, g^{XY_1}, g^{Y_2}, g^{XY_2} \dots \\
H^1 &= g, g^X, g^{Y_1}, g^Z, g^{Y_2}, g^{XY_2} \dots \\
H_0 &= f(g, g^X, g^Y, g^{XY}) = [g, g^X, g^{Y_1}, g^Z, g^{Y_2}, g^{XY_2} \dots] \\
H_1 &= f(g, g^X, g^Y, g^Z)
\end{aligned}$$

Here, $H^0 \approx H^1$, from the DDH assumption. This is because for any Z picked at random, we have

$$(g, g^X, g^Y, g^{XY}) \approx (g, g^X, g^Y, g^Z)$$

Now, we have $H^1 \approx H_0$ via the fact that, if we consider our focus on any triplet, say g, g^{y_2}, g^{xy_2} , we have that $Y_2 \dots$ can be picked uniformly at random, and will remain indistinguishable.

Finally, $H_0 \approx H_1$. This follows because we can replace g^{XY} by g^Z .