CSE 594 : Modern Cryptography 04/04/2017

Lecture 17: More Constructions

Instructor: Omkant Pandey *Scribe: Aravind Warrier, Bharathkrishna Guruvayoor Murali*

1 Discrete Log Based Collection of OWFs

Consider the following Discrete Log based collections of one way functions, *DL:* $\{f_i\colon D_i \rightarrow R_i\}$ defined as follows:

- I: {(q, g) q $\in \Pi_n$ g $\in GEN_{G_g}$ }
- D_i: {x ∥ x ∈ Q}
- R_i: G_q
- $f_{g,q}$(d):

$g^d \in G_q$. g^d means applying the group operator, d times. When you do this you are circling over all the elements in the group. So you can identify the elements with just d.

Each of the functions above are easy to compute. From the discrete logarithm assumption, these functions $f_{q,g}$ are hard to invert at the same time. The only issue with this is sampling the index *(q, g)* where g is the generator. In general, it is not known however, in special cases such as G_q being a subgroup of $\mathbb{Z}_p^*$ for a safe prime p, it is easy. So we have to assume that G_q comes with an algorithm to sample from *I*.

Collision Resistant Hash Functions

The Discrete Log problem is defined as "Given g, a generator and p, a prime number and $y = g^x$ *mod* p, find x". Here we are slightly modifying it. Instead of working with $\mathbb{Z}_p^*$ we are working with prime order sub group of $\mathbb{Z}_p^*$. In general, we can work with any prime order sub group. The hash functions are indexed by the index i and are identified by *(p, g, y)*. y is sampled randomly. Because in CRHF, you don't have to keep any secret. Taking discrete log of y w.r.t g should be hard. This function is for compressing only 1 bit. If we can compress one bit, we can compress polynomially many bits. Collision resistant hash functions is a set of such hash functions defined as H = $\{h_i\}_i$, where i is the index defined as above.

The input for one-bit compression will be x concatenated by a bit b, where x $\in \mathbb{Z}_q$

$$h_i(x||b) = g^x \cdot y^b \tag{1}$$

If we can find devise and Adversary $\mathbb{A}$ that can find collision for random i, then we can use the same Adversary to compute the Discrete Logarithm efficiently. So if the Discrete Logarithm problem is hard. This shouldn't be possible.

Proof.

We know that:
x || b ≠ x' || b'
but, *h_i(x || b) = h_i(x' || b')*

$$h_i(x||b) = h_{p,g,y}(x,b) = h_i(x'||b') \tag{2}$$
$$\tag{3}$$

if *b = b'*, then since hash will be a permutation, the discrete logarithm should also be unique.
if *b = b'*, then *x = x'*. So for the input to be distinct, then b $\neq$ b'.
With out loss of generality, lets assume *b = 0* and *b' = 1* .

$$g^x \cdot y^b = g^{x'} \cdot y^{b'} \tag{4}$$
$$\Rightarrow g^x \ mod\ p = g^{x'} \cdot y \ mod\ p \tag{5}$$
$$\Rightarrow y = g^{x-x'} \ mod\ p \tag{6}$$
$$\Rightarrow x \text{ - } x' \text{ is the DL of y} \tag{7}$$

We already know *x* and *x'*. The adversary already gave it.

Extending to compress many bits

The proof above shows how to compress 1-bit extension, because we were working with $\mathbb{Z}_p^*$. But if you work with any prime order sub group, then it works for two elements as well. We can two group elements and do the exact same computation. This is an efficient construction. Lets assume that G_q is a *q*-order sub group of Z_p^*. This time the input won't be *x* concatenated with a bit *b*, instead, the input contains two elements: $(x_1,\ x_2)$, where $x_1,\ x_2 \in Z_q$. The hash function h_i is defined as:

$$h_i(x_1||x_2) = h_{p,g,y}(x_1||x_2) = g^{x_1} \cdot y^{x_2} \ mod\ p \tag{8}$$

If there exists and adversary *A*, which finds a collision $x_1 \ || \ x_2 \neq x_1' \ || \ x_2'$, such that $h(x_1 \ || \ x_2) = h(x_1' \ || \ x_2')$

$$g^{x_1} \cdot y^{x_2} \ mod\ p = g^{x_1'} \cdot y^{x_2'} \ mod\ p \tag{9}$$
$$\Rightarrow y^{x_2 - x_2'} = g^{x_1 - x_1'} mod p \tag{10}$$

Since *g* generates an order *q* subgroup, the *DL* of *y* w.r.t. *g* is:

$$(x_1 - x_1') \times (x_2 - x_2')^{-1} mod\ p \tag{11}$$

Since *g* is a generator and it is an order *q* sub group, from Euler's theorem, we can take the inverses in the exponent w.r.t *mod q*.

If you are working with a *q* order sub group of G, then

$$g^x = g^{x\ mod\ |G|} \tag{12}$$
$$\text{Ferma's was a special case when working with } Z_p^* : g^{p-1} = 1\ mod\ p. \tag{13}$$
$$\text{Euler's was a special case:} g^{\phi(n)} = 1\ mod(n) \tag{14}$$
$$\Rightarrow \forall x\ g^x\ mod\ n = g^{x\ mod\ (\phi(n))}\ mod\ N \tag{15}$$

Here, our group if G_q, which is defined by safe prime p and its corresponding prime q.

$$\Rightarrow g^x \ mod \ p = g^{x \ mod \ q} mod \ p \tag{16}$$
$$\tag{17}$$

Lets look at the equation:

$$y^{x_2 - x'_2} = g^{x_1 - x'_1} mod \ p \tag{18}$$
$$y^{(x_2 - x'_2) \ mod \ q} mod \ p = g^{(x_1 = x'_1) mod \ q} mod \ p \tag{19}$$

Since, we are working with Z_p*, there exists inverse. $\therefore$ (x_2 - x'_2) has an inverse in Z_p^* and lets call it $(x_2 - x'_2)^{-1}$. Multiplying both sides of Eq 19, by $(x_2 - x'_2)^{-1}$

$$y^{(x_2 - x'_2) \cdot (x_2 - x'_2)^{-1} mod \ q} = g^{(x_1 - x'_1) \cdot (x_2 - x'_2) \ mod \ q} mod \ p \tag{20}$$
$$\Rightarrow y = g^{(x_1 - x'_1) \cdot (x_2 - x'_2) \ mod \ q} mod \ p \tag{21}$$

What if x_2 = x_2', then as we did for the bit case, we can safely conclude that $x_1 = x'_1$

2 Key Exchange

There are two parties, and they talk in public where anyone can listen. By taking in public, they both create public keys using their local randomness. They do not know each others randomness. They engage in a protocol and create a transcript τ. The key exchange takes place as shown:

- Alice picks a local randomness r_A
- Bob picks a local randomness r_B
- Alice and Bob engage in a protocol and generate the transcript τ.
- Alices view $V_A = (r_A, \tau)$ and Bob's view $V_B = (r_B, \tau)$.
- Eavesdroppers view $V_E = \tau$
- Alice outputs K_A as a function of V_A and Bob outputs K_B as a function of V_B
- Correctness: $Pr_{r_A, r_B}[K_A = k_B] \approx 1$
- Security: $(k_A, \tau) \equiv (k_B, \tau) \approx (r, \tau)$

For correctness, some noticeable probability is enough, the protocol can be amplified and they can always exchange key with correctness roughly of probability 1.

3 Diffie-Hellman Key Exchange

The Diffie-Hellman key exchange is considered the first primitive, which made the move from symmetric cryptography to asymmetric cryptography. We can create key-exchange from public key encryption :

Alice generates (public key, secret key) pair using a generation function Gen, and randomness r. Alice sends the public key P_{k_A} to Bob. Bob encrypts a key k using the public key P_{k_A} received from Alice. Let this be c. Hence, Bob's key $K_B = k$.
Bob sends back c to Alice, who then decrypts it with the Secret key S_{k_A}. Let this be equal to K_A. The flow of algorithm is shown below:

Alice		**Bob**
$(P_{k_A}, Sk_A) \leftarrow Gen(r)$	$\xrightarrow{p_{k_A}}$	
		c = $Enc_{P_{k_A}}$(k)
	$\xleftarrow{c}$	
K_A = Dec(c,S_{k_A})		

We can see here that the key generated by Alice K_A is equal to the key k used by Bob. The correctness of the public key encryption scheme ensures that $K_A = k_B = k$.
The transcript here is $\tau = (P_{K_A}, c)$. By security of encryption, this ciphertext is indistinguishable from the distribution for a random string.
Hence, with this kind of exchange using public key encryption, Alice and Bob ends up having a key k. So we prove that public key encryption can be used for key exchange.

The Diffie-Hellman key exchange is based on discrete-logarithms. It works as follows:

- Let p be a large safe prime, ie., p = 2q+1 for prime q
- Let g be a generator of order q subgroup of G_q of Z_p^*
- Alice picks $x \leftarrow Z_p^*$ and sends $X = g^x$ mod p to Bob
- Bob picks $y \leftarrow Z_p^*$ and sends Y = g^y mod p to Alice

Alice and Bob both can compute K = g^{xy} mod p as follows:

Alice	**Bob**
K = Y^x mod p	K = X^y mod p
= $(g^y \text{ mod p})^x$ mod p	= $(g^x \text{ mod p})^y$ mod p
= g^{xy} mod p	= g^{xy} mod p

3.1 Why is this secure?

The transcript here is (X,Y,G,p). Diffie and Hellman, who at that time were not clear what security really is, treated this as a one-way function. They claimed that this cannot be broken unless discrete log is broken. Since we are working with slightly better definitions now, we talk about indistinguishability. We can realize here that (X,Y,G,p) roughly represents the encryption of G^y under the key G^x. This is really the Elgamal encryption.Hence, if we directly assume that DDH is secure, we can directly prove that Diffie-Hellman key exchange is also secure.

4 Two round key exchange implies PKE

We have a 2 round key-exchange protocol after which both parties have a common key. This is enough to build a public key encryption. We can think of the first round as sending the public key, and the second round as sending the cipher text. The idea here is to use the key as a computational one-time pad. Because the key is hidden, it is indistinguishable from random by definition.
Hence, we can use it to mask any message. So it is possible to build public-key encryption from 2 round key exchange protocols.