Homework 3

1 Problem 1 (20 points)

A bilinear map gives a "gap Diffie-Hellman" group, where decisional Diffie-Hellman is easy, but the computational version remains hard.

(a) Explain why the Diffie-Hellman protocol, when using a gap Diffie-Hellman group, no longer yields a pseudorandom key.

(b) Explain how to tweak the protocol to yield a pseudorandom key. Your scheme should:

- Support keys of length at least λ.
- Remain non-interactive: there is a single message from Alice to Bob and from Bob to Alice, and both messages are sent simultaneously.
- Before the protocol begins, the only information that Alice and Bob share is the group $\mathbb{G}$ and a generator g.

Remember to prove the security of your protocol.

Hint: for part (b), it will be useful to have the following strengthening of the Goldreich-Levin theorem. Let S be a PPT algorithm that takes as input the security parameter, and samples pairs (s, aux). We say that S is computationally unpredictable if, for all PPT A,

$$\Pr[A(\mathsf{aux}) = s : (\mathsf{aux}, s) \leftarrow S(1^\lambda)] < \mathsf{negl}(\lambda)$$

Let $n(\lambda)$ be the length of s outputted by S. Then for any computationally unpredictable S, the following two distributions are computationally indistinguishable:

$$(r, \mathsf{aux}, < r, s >) : (\mathsf{aux}, s) \leftarrow S(1^\lambda), r \leftarrow \{0,1\}^{n(\lambda)} \text{ and}$$
$$(r, \mathsf{aux}, b) : (\mathsf{aux}, s) \leftarrow S(1^\lambda), r \leftarrow \{0,1\}^{n(\lambda)}, b \leftarrow \{0,1\}$$

2 Problem 2 (20 points)

Suppose you are given a group $\mathbb{G}$ of *unknown order* p, along with a generator g. The only thing you know about $\mathbb{G}$ is that it's order is between 2^λ and $2^{\lambda+1}$.

(a) Assume decisional Diffie-Hellman is still hard on $\mathbb{G}$. Show how to build a multiparty non-interactive key agreement protocol using $\mathbb{G}$. This will require tweaking the usual Diffie-Hellman protocol.

(b) Consider the Weil pairing $e : \mathbb{G} \times \mathbb{G}' \to \mathbb{G}_2$, but suppose that $\mathbb{G}$ is a group of unknown order. Explain why it is impossible to compute the Weil pairing in this case, as we computed in class.

3 Problem 3 (30 points)

In class, we saw how to construct an identity-based encryption scheme. Here, we will see that any identity-based encryption scheme gives rise to a signature scheme.

Let $(\mathtt{Gen}, \mathsf{Extract}, \mathtt{Enc}, \mathtt{Dec})$ be an IBE scheme. Define the following signature scheme.

- $\mathtt{Gen}$ is the same as $\mathtt{Gen}$. The signing key sk for the signature scheme is the master secret key msk for the IBE scheme, and the public key pk is the master public key mpk.
- $\mathtt{Sign}(\mathsf{sk}, m)$. Interpret m as an identity id for the IBE scheme. Then run $\mathsf{sk}_{\mathsf{id}} \leftarrow \mathsf{Extract}(\mathsf{msk}, \mathsf{id})$. Output the signature $\sigma = \mathsf{sk}_{\mathsf{id}}$.

(a) Explain how to verify a signature using this scheme.

Consider the IBE security game, between an adversary A and challenger Ch.

- The challenger is given an input bit b and security parameter λ. It runs $\mathtt{Gen}(1^\lambda)$ to get $(\mathsf{msk}, \mathsf{mpk})$. It then gives mpk to A.
- A is allowed to make arbitrary extract queries on identities id of it's choice. In response, Ch returns $\mathsf{sk}_{\mathsf{id}} \leftarrow \mathsf{Extract}(\mathsf{msk}, \mathsf{id})$.
- At some point, A makes a challenge query on tuple $(\mathsf{id}^*, m_0, m_1)$. The requirement is that id^* must not have been queries in an extract query. In response, Ch returns $c^* \leftarrow \mathtt{Enc}(\mathsf{mpk}, \mathsf{id}^*, m_b)$.
- A is allowed to make more extract queries on identities id, conditioned on $\mathsf{id} \neq \mathsf{id}^*$.

- Finally, A outputs a guess b' for b.

IBE security is defined in the usual way given the game above.

(b) Prove that the signature scheme given above is secure (that is, weakly EUF-CMA secure), provided the underlying IBE scheme is secure and the message space for the IBE scheme has exponentially-many messages.

(c) Recall the IBE scheme that we saw in class, and apply the above transformation to it. Show that the resulting verification algorithm can be simplified. Does the resulting signature scheme look familiar?

4 Problem 4 (30 points)

In this problem, you will show how to sample from the discrete Gaussian distribution $D_{\sigma,c}$. You are given the following fact:

Theorem 1 *Suppose $\sigma \geq 1$. Let t be a function that grows faster than $\sqrt{\log \lambda}$. Then there is a negligible function* negl *such that*

$$\Pr[|x - c| > \sigma t(\lambda) : x \leftarrow D_{\sigma,c}] < \mathsf{negl}(\lambda)$$

Also, we will assume access to a procedure that samples uniformly random real numbers between 0 and 1. We will not worry about the precision of real numbers; assume we can compute and store infinitely precise numbers.

Let $\rho_\sigma(y) = e^{-\pi y^2/\sigma^2}$. Notice that $0 \leq \rho_{\sigma,c} \leq 1$. Let $p_{\sigma,c}(x) = \rho_\sigma(x-c)/\sum_{z=-\infty}^{\infty} \rho_\sigma(z-c)$. Then $p_{\sigma,c}(x) = \Pr[x : x \leftarrow D_{\sigma,c}]$.

We will use rejection sampling. One approach is to choose a random integer x, and then with probability $p_{\sigma,c}(x)$, accept and output x. Otherwise, throw away x and repeat from the beginning. Notice that in each iteration, any x is outputted with probability proportional to $p_{\sigma,c}(x)$. Therefore, once the algorithm terminates, the distribution of outputs is exactly $D_{\sigma,c}$.

Unfortunately, the above does not quite work for two reasons:

- It is not possible to sample a uniformly random x over all integers (since the expected length of a random integer is infinite)
- Suppose σ is quite large (say, exponential). Then $p_{\sigma,c}(x)$ is exponentially small for all x. In this case, the procedure above will take an exponential number of iterations to terminate, and is therefore inefficient.

Show how to fix the above problems and give a protocol that terminates in an expected polynomial number of iterations, and outputs a sample x from the discrete Gaussian. The algorithm may output a distribution that is slightly different from the discrete Gaussian, but it should be negligibly close (in the parameter λ). You may assume that $\sigma \geq 1$. Prove that the number of iterations is bounded by a polynomial in λ.