1 Pseudorandom Function

A **Psudorandom Function (PRF)** is a function is motivated under the following idea. For a PRG that can produce an exponential number of bits, a PRF should be able to take in an index as input and return the value of the PRF at that particular index. A PRF generated by seed s denoted as $PRF_s(i)$ takes in input i and generates the pseudorandom value of PRG(s) at index i.

We define the security of a PRF as follows:

1. A challenger C exists in world 1, where it can interact with an adversary A that generates strings using the PRF or world 2 where A generates strings completely from random.

2. For polynomially many rounds, C can ask A to give it the value of either the PRF or completely random string at index i. In the case of world 1, A returns $PRF_s(i)$ in the case of world 2 it returns a completely random string (In the case of world 2, the Adversary saves the i values so it can always return the same random string if queried for index i again).

3. The challenger C wins if it can differentiate the two worlds with non-negligible advantage.

We say that a PRF_s is secure if the probability of all PPT adversaries being able to distinguish the two worlds with probability $\frac{1}{2} + \epsilon$ is negligible.

1.1 Applications

This idea is used in Airforce friend or foe systems where two vehicles can determine if the other is an ally by testing whether or not they can determing $PRF_s(i)$ for some i which should only be possible if they know the secret seed s.

1.2 Construction [Goldreich, Goldwasser, Micali]

Given a $PRG : \{0,1\}^n \to \{0,1\}^{2n}$ we can construct a PRF in the following manner. Starting with seed s PRF_s:

1. Given index $i = i_0 i_1 ... i_k$, we perform the following n operations with PRG.

2. If $i_j = 0$, set $s = PRG(s)_{0...n}$, if $i_j = 1$, $s = PRG(s)_{n+1...2n}$. In other words, from $j = 1...k$, if i_j is a 0, we take the seed s and reassign it to the left half bits of $PRG(s)$, else we set it to the right half bits.

3. The algorithm can either return the final n bits of the final s value, or just the first bit indicating the bit value at i.

1.3 Proof of Security

(I cannot produce images so I would recommend looking at the class notes pg 62) We argue that the above construction is secure. To prove this, consider if it was not, then there exists some adversary A that can differentiate between the outputs of a random n length strings and the output of $PRF_s(i)$. We consider the length of the index to be fixed at length h. Now, we define trees T_i as generating the first i levels of the tree using completely random numbers, and the $h - i$ last levels of the tree by using PRG_s outputs. One can think of this as only using the last $h - i$ bits of the input index to determine the output of the PRG.

First, if this PRF construction is insecure, then there exists some level i where the output of A with trees T_i and T_{i+1} differ by at least ϵ/n. This follows by hybrid argument. Therefore, we can construct an adversary A that can determine if a polynomially many samples are generated by a PRG or truly randomly given access to an adversary A' that can determine if a tree is T_i or T_{i+1} as follows.

1. A receives polynomially many samples generated by either $PRG(s)$ or $r \leftarrow \$$ refered to as $\{Z\}$.

2. A' asks for polynomially many samples which A generates by using $\{Z\}$ as the $i + 1$ level of the tree.

3. A outputs whatever A' outputs.

This works since, in the case that A receives truly random samples for $\{Z\}$, it is exactly T_{i+1} and if it receives values generated by a PRG it is exactly T_i. Note that A must have enough samples from $\{Z\}$ to answer all queries A' has.

2 Introduction to Signatures

2.1 Merkle Trees

A hash function is **collision resistant** if no poly time adversary can find $x_1, x_2 | x_1 \neq x_2$ s.t. $H(x_1) = H(x_2)$.

Merkle Hash Tree
Merkle hash trees use hash function $f\{0,1\}^{2n} \rightarrow \{0,1\}^n$ and takes a document of length kn and hashes it to length n by hashing adjacent blocks of length n over and over.

Definition of Digital Signature Scheme
A Digital signature scheme is a triple of algorithms $\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify}$ that do the following:

- $\mathsf{KeyGen}(1^k, R) \rightarrow (pk, sk)$: Takes in some random seed R and security parameter 1^k to generate public key pk and secret key sk.
- $\mathsf{Sign}(m, r, sk, pk) \rightarrow (m, \mathsf{Sign}(m))$: Generates the signature of message m using public and secret keys pk, sk and randomness r.
- $\mathsf{Verify}(m, \mathsf{Sign(m)}, pk) \rightarrow \{\mathsf{Yes}, \mathsf{No}\}$ Which takes in signature along with the message and outputs either yes or no.

The following definition of security for Digital Signature Schemes was defined by Goldwasser, Micali, Rivest. We define it as a game between challeneger C and adversary A

1. Challenger C generates $(pk, sk) \leftarrow \mathsf{KeyGen}(1^n, R)$ and sends A pk.
2. A can send polynomially many m to which C generates $\mathsf{Sign}(pk, sk, R, m)$ and sends the signature to A.
3. A tries to generate a signature s for message m' which they have not sent before such that $\mathsf{Verify}(m, s) = \mathsf{Yes}$

We say a scheme is secure if the probability of any PPT adversary generating a valid signature s is