Today’s Agenda

- Motivation
- Definition
- GGM construction

1 Motivation and Definition

We have seen how a PRG can be used to create a pseudorandom string of length polynomial in the length of the seed. But what if given a huge pseudorandom string, you want to have random access to it? We want that given seed $s$ and an index $i$, we are able to efficiently compute or retrieve the $i$-th bit or chunk of $G(s)$. What if this index $i$ is not bounded by a polynomial but can be any $k$-bit integer.

Consider the following situation where this could be useful: Suppose you have to conduct a huge scientific experiment in which you simulate a lot of runs for it. Not all the results of this simulation will be interesting, only some of the simulations which you would like to store. So you need an efficient way of accessing the randomness for those particular simulations. This is achieved via a pseudorandom function (PRF).

Intuition: Any ppt $\mathcal{A}$ cannot distinguish between the following two experiments:

Exp 1

1. Choose $s \leftarrow \{0,1\}^k$
   $\mathcal{A}(1^k)$ adaptively queries $F(s, \cdot)$
   on inputs $x_1, \ldots, x_q$ of his choice
   and obtains values $F(s, x_i)$ for $1 \leq i \leq q$

2. $\mathcal{A}$ outputs a guess $b'$

Exp 2

1. Pick a random function $R()$ from $\{0,1\}^k \mapsto \{0,1\}$
   $\mathcal{A}(1^k)$ adaptively queries $R(\cdot)$
   on inputs $x_1, \ldots, x_q$ of his choice
   and obtains values $R(x_i)$ for $1 \leq i \leq q$

2. $\mathcal{A}$ outputs a guess $b'$
**Definition 1** (Pseudorandom function). \( F : \{0,1\}^k \times \{0,1\}^k \rightarrow \{0,1\} \) is a pseudorandom function (PRF) if for all PPT \( A \), there exists a negligible \( \nu() \) such that \( |p_{A,F}(k) - p_{A,R}(k)| = \nu(k) \) where

\[
p_{A,F}(k) = \Pr[s \leftarrow \{0,1\}^k ; b' \leftarrow A^{F(s,\cdot)} : b' = 0]
p_{A,R}(k) = \Pr[R \leftarrow \text{Random functions} ; b' \leftarrow A^{R(\cdot)} : b' = 0]
\]

## 2 GGM Construction

Start with a PRG \( G : \{0,1\}^k \rightarrow \{0,1\}^k \times \{0,1\}^k \). So you can think of \( G(s) \) as producing a string of length \( 2k \), we will call first half as \( G_0(s) \) and second half as \( G_1(s) \). Let input be \( x = x_1x_2\ldots x_n \) where each \( x_i \in \{0,1\} \), then the PRF, with key \( K = s \) is defined as follows:

\[
F(s,x_1x_2\ldots x_n) = G_{x_n}(\ldots G_{x_2}(G_{x_1}(s))\ldots)
\]

Basically, based on the value of input \( x \) you are traversing from the root of the GGM tree (as shown in figure) to reach a particular leaf node which is the PRF value for that \( x \). We can prove the security of this PRF by a two-dimensional hybrid argument: One dimension is the number of queries \( (l) \) that \( A \) asks and the other dimension is the height of the GGM \( (i) \) tree. We start with all queries being answered by the \( F(s,\cdot) \) oracle and then for each query, at each height of the tree we replace pseudorandom strings by truly random. We define hybrids as follows:

Hybrid 0: \( F(s,\cdot) \)

Hybrid \( k \): Random function \( R(\cdot) \)

In hybrid \( i \), on input \( x_j \), look at first \( i \) bits of \( x_j \), \( \alpha^j = \alpha^j_1\alpha^j_2\ldots\alpha^j_i \). Let \( s_{\alpha^j} \) denote the value obtained by traversing the tree with bits \( \alpha^j_1\alpha^j_2\ldots\alpha^j_i \) that is, \( s_{\alpha^j} = G_{\alpha^j_1}(\ldots G_{\alpha^j_i}(G_{\alpha^j_1}(s))\ldots) \). If \( s_{\alpha^j} \) already defined, compute \( s_{x_j} \) deterministically from it. Else, pick \( s_{\alpha^j} \leftarrow \{0,1\}^k \) and output \( s_{x_j} \) from it.

We want to prove that for all PPT \( A \), there exists a negligible \( \nu() \) such that for all \( 0 \leq i < k \)

\[
p_{A}(k,i) = \Pr[\text{Oracle} \leftarrow \text{Hybrid } i \text{ oracle for } k ; b' \leftarrow A^{\text{Oracle}_{\cdot}} : b' = 0]
\]

Now we have to prove that adjacent hybrids are indistinguishable. These hybrids correspond to two adjacent heights on the tree.
2.1 Attempt 1 of defining hybrids

Hybrid\((i, l)\):

- On input \(x_j\), look at first \(i\) bits of \(x_j\), call them \(\alpha^j\).
- If \(\alpha^j < l\), pick \(s_{\alpha^j0}\) and \(s_{\alpha^j1}\) at random, compute \(s_{x_j}\) deterministically from them.
- Else pick \(s_{\alpha^j}\) at random and compute \(s_{x_j}\).

If we have \(l\) such that \(\alpha < l\), then \((i + 1)\) st level is random. Otherwise, if \(\alpha \geq l\) then \(i\) th level is random. Is it sufficient to show that \(\text{Hybrid}(i, l) \approx \text{Hybrid}(i, l + 1)\)? Note that with this approach, we will get exponential number of hybrids in this case and security is no longer guaranteed. So we will define our second dimension of hybrids as follows:

2.2 Attempt 2 of defining hybrids

Hybrid\((k, i, l)\):

- On input \(x_j\) if \(l < j\), look at first \(i\) bits of \(x_j\), \(\alpha^j = \alpha^j_1\alpha^j_2\ldots\alpha^j_i\).
  - If \(s_{\alpha^j}\) already defined, compute \(s_{x_j}\) deterministically from it.
  - Else, pick \(s_{\alpha^j} \leftarrow \{0, 1\}^k\) and output \(s_{x_j}\) from it.
- If \(l \geq j\), look at first \((i + 1)\) bits of \(x_j\), \(\alpha^j = \alpha^j_1\alpha^j_2\ldots\alpha^j_i\alpha^j_{i+1}\).
  - If \(s_{\alpha^j}\) already defined, compute \(s_{x_j}\) deterministically from it.
  - Else, pick \(s_{\alpha^j} \leftarrow \{0, 1\}^k\) and output \(s_{x_j}\) from it.

Hence we get that \(\text{Hybrid}(k, i, 0) = \text{Hybrid}(k, i)\). If \(l = q(k)\) for some polynomial \(q()\) then \(l > j\) and hence \(\text{Hybrid}(k, i, q(k)) = \text{Hybrid}(k, i + 1)\).

It is now sufficient to show that for all PPT \(\mathcal{A}\), there exists a negligible \(\nu()\) such that for all \(0 \leq i < k\) and \(0 \leq l < q^2(k)\) \(|p_{\mathcal{A}}(k, i, l) - p_{\mathcal{A}}(k, i, l + 1)| = \nu(k)\)

Suppose there exists \(\mathcal{A}\) s.t. \(\exists i, l\) s.t. \(|p_{\mathcal{A}}(k, i, l) - p_{\mathcal{A}}(k, i, l + 1)|\) is non-negl. \(\epsilon(k)\). Suppose \(\mathcal{B}\) is given \(\mathcal{A}\) and also knows \(i, l\). Let \(y\) denote the first half of the output of \(G\) and let \(z\) denote the second half. \(\mathcal{B}\) will behave as follows:

\(\mathcal{B}(y, z)\):

- For first \(l\) queries, answer based on \(\alpha^j\) of length \((l + 1)\)
- For \((l + 1)\) st query, look at \(\alpha = x_1\ldots x_{i+1}\), set
  
  \[s_\alpha = \begin{cases} y & \text{if } x_{i+1} = 0 \\ z & \text{otherwise} \end{cases}\]

This also defines the sibling of \(s_\alpha\)

- For root of them, answer based on \(\alpha^j\) of length \(l\)

Difference between \(\text{Hybrid}(k, i, l)\) and \(\text{Hybrid}(k, i, l + 1)\) is how to answer the \((l + 1)\) st query. Hence eventually you will reach the final hybrid in which \(\mathcal{A}\) interacts with a random function \(R()\).