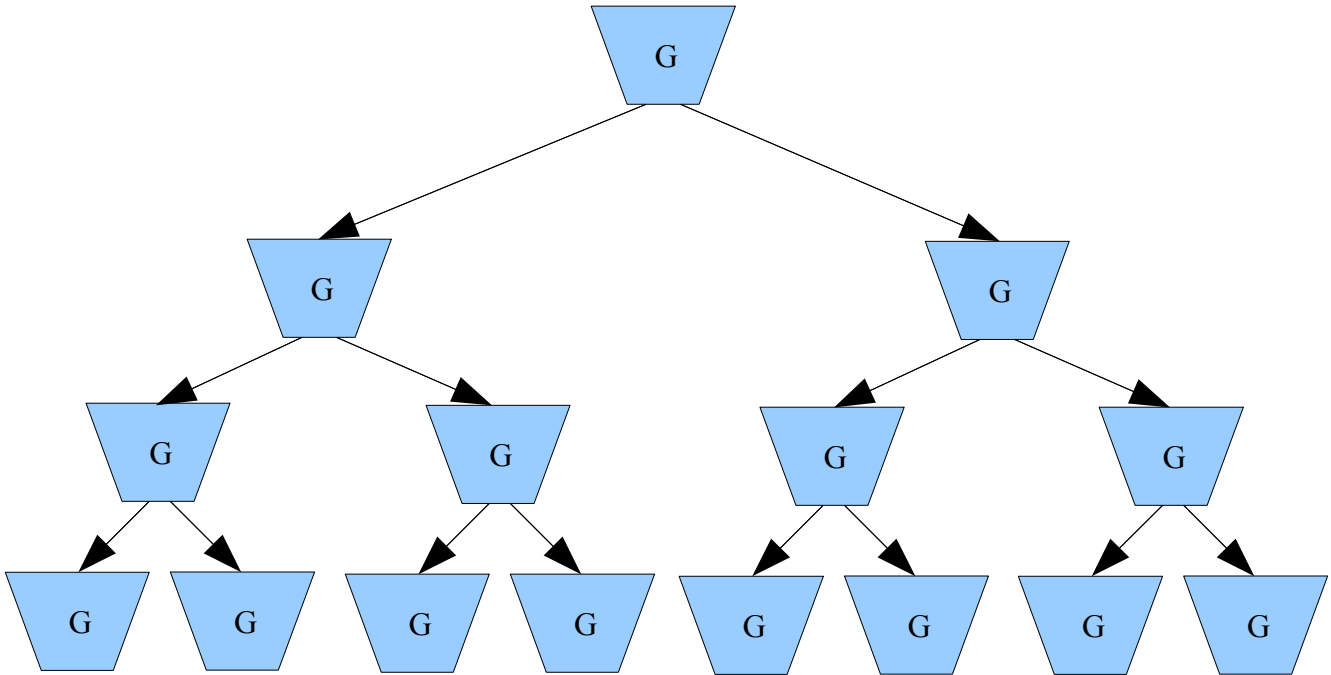


Review:

If $G: \{0,1\}^l \rightarrow \{0,1\}^L$ is a (t, ϵ) secure PRG then

$$U_L \parallel G_0 U_1 \stackrel{t}{\sim} U_L \parallel U_L$$

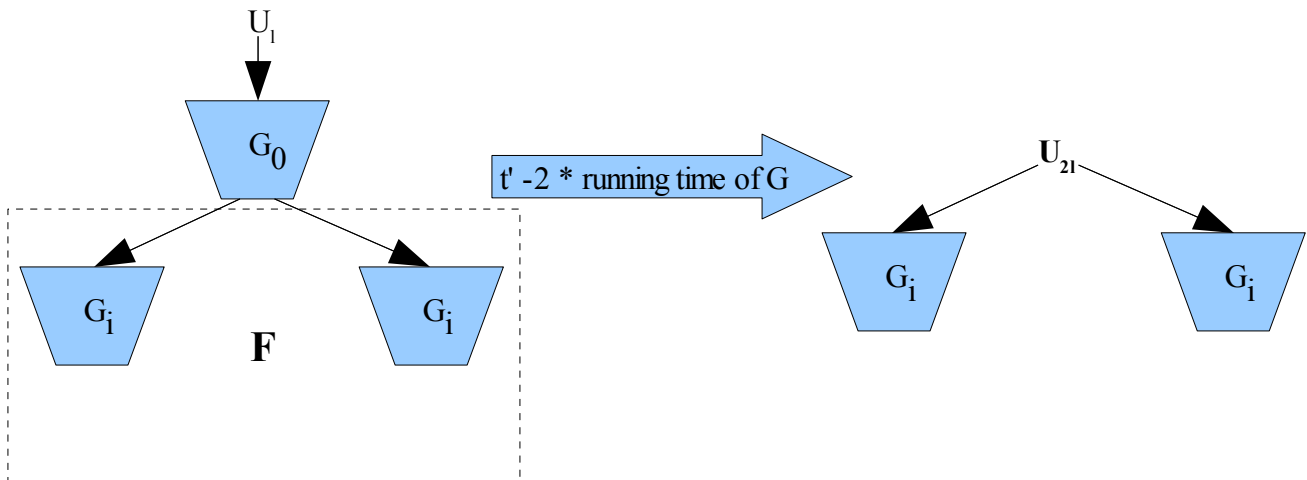
GGM (Goldwasser, Goldreich, Micali) Construction:



Suppose $G_0: \{0,1\}^l \rightarrow \{0,1\}^{2l}$ is a (t, ϵ) secure PRG then define:

$$G_i(x) = G_{i-1}(\text{left}(G_0(x))) \parallel G_{i-1}(\text{right}(G_0(x)))$$

Suppose G_i is (t', ϵ') secure



We prove the security of G_i using induction.

$$G_{i+1} \circ U_1 = F(G_0 \circ U_1) \sim F(U_{2i}) = G_i \circ U_1 \parallel G_i \circ U_1 \sim U_2^{(i+2)} \quad \dots \quad (I)$$

Proof:

Base Case:

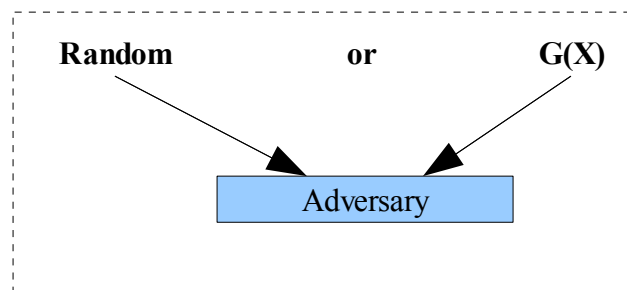
$$G_0 \circ U_1 \sim U_{2i} \quad (\text{by definition of } G_0)$$

Inductive Step:

$$\text{Suppose } G_i \circ U_1 \sim U_2^{(i+1)}$$

$$\text{Then by (I) } G_{i+1} \circ U_1 \sim U_2^{(i+2)}$$

Hence Proved



The underlying assumption behind the security definition of a PRG is that the Adversary gets one output and he has to distinguish it from a random number. When we talk about an encryption scheme, the Adversary gets many outputs to analyze. In this case, our security definition for PRG proves to be insufficient. We will define a new security definition for an encryption scheme, in which we also introduce another parameter in the form of the number of efforts allowed to the Adversary, which translates to the number of outputs he gets to analyze in order to distinguish the scheme from a random function.

Notation:

$\text{Funcs}(X, Y) = \{f: X \rightarrow Y\}$ Set of all functions from X to Y
 $R \leftarrow \text{Funcs}(X, Y)$ R is a function chosen from the set

for example:

$$R \leftarrow \text{Funcs}(\{0,1\}^{128}, \{0,1\}^{1024})$$

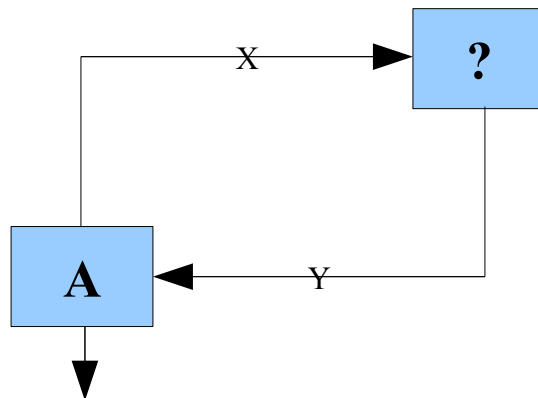
Def: $F: \text{Keys} \times X \rightarrow Y$ is a (t, q, ϵ) secure PRF if for all adversaries A running in time $\leq t$ and making at most q oracle queries:

$$\text{Adv } A = |\Pr[A^{FK} = 1 \mid K \leftarrow U_{\text{keys}}] - \Pr[A^R = 1 \mid R \leftarrow \text{Funcs}(X, Y)]| \leq \epsilon$$

Such a function can be designed in the form of a table and each time it encounters a new argument, it computes the output and stores it in the table. On all subsequent queries for the same argument, the value kept in the table is returned.

1234	Value1
1456	Value1
...	...
...	...
...	...

Attack outline:



“This is F”

OR

“This is a random number generator”