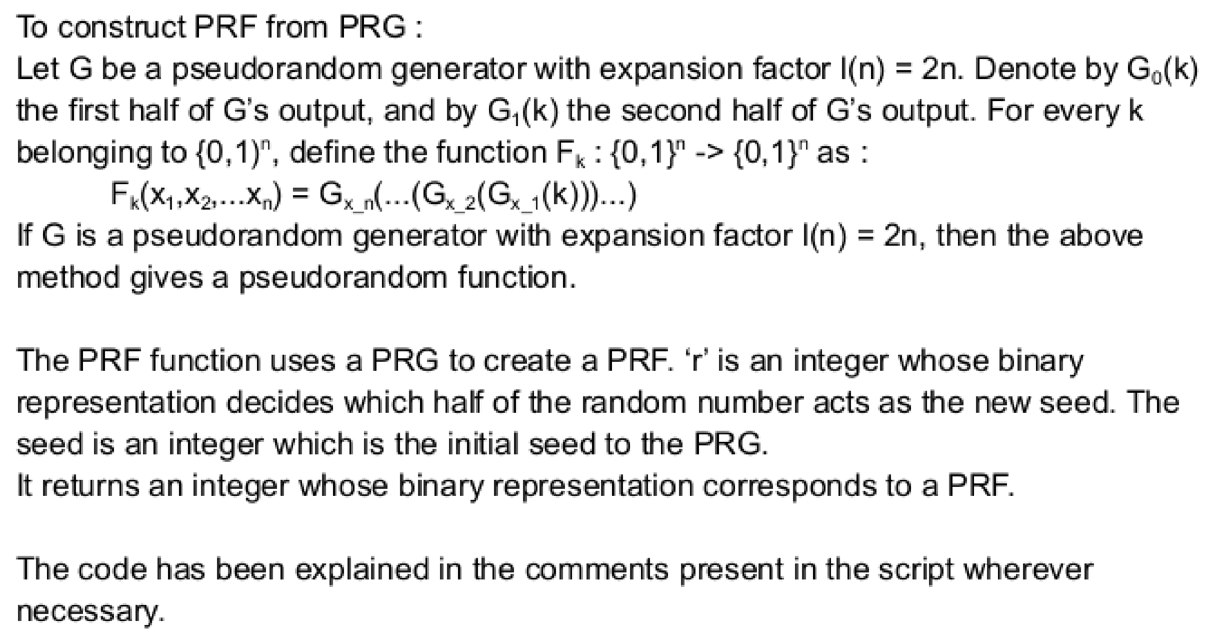
2022201041

Build a provably secure PRF from PRG



To construct PRF from PRG :
Let G be a pseudorandom generator with expansion factor $l(n) = 2n$. Denote by $G_0(k)$ the first half of G's output, and by $G_1(k)$ the second half of G's output. For every k belonging to $\{0,1)^n$, define the function $F_k : \{0,1\}^n \rightarrow \{0,1\}^n$ as :

$$F_k(x_1,x_2,\ldots x_n) = G_{x\_n}(\ldots(G_{x\_2}(G_{x\_1}(k)))\ldots)$$

If G is a pseudorandom generator with expansion factor $l(n) = 2n$, then the above method gives a pseudorandom function.

The PRF function uses a PRG to create a PRF. 'r' is an integer whose binary representation decides which half of the random number acts as the new seed. The seed is an integer which is the initial seed to the PRG.
It returns an integer whose binary representation corresponds to a PRF.

The code has been explained in the comments present in the script wherever necessary.

A Pseudo-Random Function (PRF) is a deterministic function that emulates the behavior of a random function. PRFs are commonly used in cryptography to provide security in various protocols such as encryption, message authentication, and key derivation.

A PRF is considered secure if it satisfies the following properties:

Pseudorandomness: The output of the PRF function must appear indistinguishable from random data. In other words, given a PRF output, it should be infeasible to determine whether the output was generated by a PRF or by a truly random function.

Keyed: The PRF must take an input key that is kept secret and used to generate the output. Without knowledge of the key, it should be infeasible to compute the PRF output.

Efficient: The PRF should be computationally efficient, meaning that it can be computed in a reasonable amount of time for a given key length and input size.

If a PRF satisfies these properties, then it can be considered secure for use in cryptographic applications

Let G be a pseudorandom generator with expansion factor `(n) = 2n,

and define G0 , G1 as in the text. For k ∈ {0, 1}n , define the function

Fk : {0, 1}n → {0, 1}n as:

Fk (x1 x2 · · · xn) = Gxn (· · · (Gx2 (Gx1 (k))) · · ·) .

It is useful to view this construction as defining, for each key $k \in \{0,1\}^n$, a complete binary tree of depth n in which each node contains an n-bit value. (See Figure 8.2, where $n = 3$.) The root has value k, and every non-leaf node with value v has left child with value $G_0(v)$ and right child with value $G_1(v)$. The result $F_k(x)$ for $x = x_1 \cdots x_n$ is defined to be the value on the leaf node reached by traversing the tree according to the bits of x, where $x_i = 0$ means "go left" and $x_i = 1$ means "go right." (The function is only defined for inputs of length n, and thus only values at the leaves are ever output.) The size of the tree is exponential in n. Nevertheless, to compute $F_k(x)$ the entire tree need not be constructed or stored; only n evaluations of G are needed.

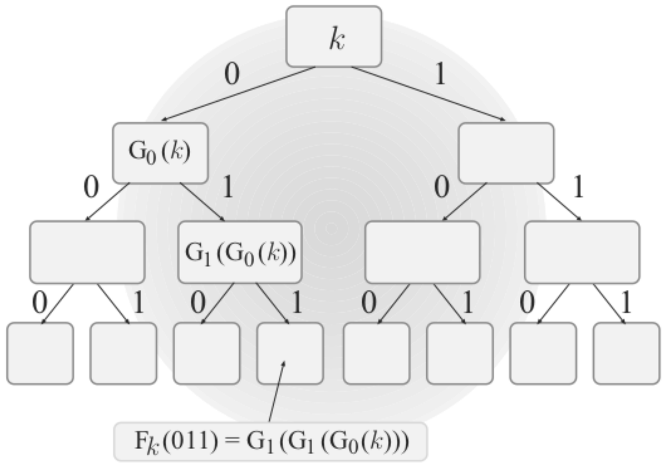


FIGURE 8.2: Constructing a pseudorandom function.