2.6 A General Prediction Method

The method of BOYAR (née PLUMSTEAD) admits a broad generalization by the **BK algorithm** (named after BOYAR and KRAWCZYK): It applies to recursive formulas that have an expression in terms of (unknown) linear combinations of known functions. A suitable language for its description is commutative algebra, that is, rings and modules.

So let R be a commutative ring (with $1 \neq 0$), and X, Z be R-modules. Let

$$\Phi^{(i)} : X^i \longrightarrow Z \quad \text{for } i \geq h$$

be a family of maps that we consider as known, and

$$\alpha : Z \longrightarrow X$$

be a linear map considered as secret. From these data we generate a sequence $(x_n)_{n\in\mathbb{N}}$ in X by the following algorithm, see Figure 2.2:

- Set $x_0, \ldots, x_{h-1} \in X$ as initial values.
- After generating $x_0, \ldots, x_{n-1}$ for some $n \geq h$ let

$$\begin{aligned} z_n &:= \Phi^{(n)}(x_0, \ldots, x_{n-1}) \in Z, \\ x_n &:= \alpha(z_n) \in X. \end{aligned}$$

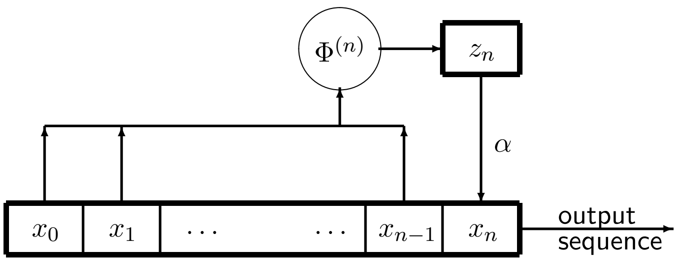

Figure 2.2: A very general generator

Here, in greater generality as before, we allow that each element of the sequence depends on *all* of its predecessors, that is, on the complete "past". A reasonable use for pseudorandom generation of course supposes that the $\Phi^{(i)}$ are efficiently computable. In the sample case $R = \mathbb{Z}/m\mathbb{Z}$, $X = R^k$, the cost should grow at most polynomially with $\log(m)$, h, and k.

Examples

1. The linear congruential generator: $R = \mathbb{Z}/m\mathbb{Z} = X$, $Z = R^2$, $h = 1$, $x_n = ax_{n-1} + b$,

$$\Phi^{(i)}(x_0, \ldots, x_{i-1}) = \begin{pmatrix} x_{i-1} \\ 1 \end{pmatrix},$$

$$\alpha\begin{pmatrix} s \\ t \end{pmatrix} = as + bt.$$

2. The linear-inversive congruential generator: R, X, Z, h, α as above, $x_n = ax_{n-1}^{-1} + b$,
$$\Phi^{(i)}(x_0, \ldots, x_{i-1}) = \begin{pmatrix} x_{i-1}^{-1} \bmod m \\ 1 \end{pmatrix}.$$
(Set the first component to 0 if x_{i-1} is not invertible $\bmod m$.)

3. Congruential generators of higher degree: $R = \mathbb{Z}/m\mathbb{Z} = X$, $Z = R^{d+1}$, $h = 1$, $x_n = a_d x_{n-1}^d + \cdots + a_0$,
$$\Phi^{(i)}(x_0, \ldots, x_{i-1}) = \begin{pmatrix} x_{i-1}^d \\ \vdots \\ x_{i-1} \\ 1 \end{pmatrix},$$
$$\alpha\begin{pmatrix} t_0 \\ \vdots \\ t_d \end{pmatrix} = a_d t_0 + \cdots + a_0 t_d.$$

4. Arbitrary congruential generators: $R = \mathbb{Z}/m\mathbb{Z}$, $x_n = s(x_{n-1})$, $h = 1$. If m is prime, then each function $s\colon R \longrightarrow R$ has an expression as a polynomial of degree $< m$, as in Example 3. For a more general module m we may use the basis $\{e_0, \ldots, e_{m-1}\}$ with $e_i(j) = \delta_{ij}$ of R^R. The basis representation is $s = \sum_{i=0}^{m-1} s(i)e_i$. Thus we set $X = R$, $Z = R^m$, and
$$\Phi^{(i)}(x_0, \ldots, x_{i-1}) = \begin{pmatrix} e_0(x_{i-1}) \\ \vdots \\ e_{m-1}(x_{i-1}) \end{pmatrix},$$
$$\alpha\begin{pmatrix} t_0 \\ \vdots \\ t_{m-1} \end{pmatrix} = s(0)t_0 + \cdots + s(m-1)t_{m-1}.$$

5. For multistep congruential generators set h equal the recursion depth.

6. For nonlinear feedback shift registers see the next section 2.7.

For cryptanalysis we assume that the $\Phi^{(i)}$ are known, but α is unknown. (Later on, in the case $R = \mathbb{Z}/m\mathbb{Z}$, we'll also treat m as unknown.) The question is: Given an initial segment $x_0, \ldots, x_{n-1}$ ($n \geq h$) of the output sequence, is there a method to predict the next element x_n?

To this end we consider the ascending chain $Z_h \subseteq Z_{h+1} \subseteq \ldots \subseteq Z$ of submodules with

$$Z_n = Rz_h + \cdots + Rz_n.$$

If $Z_n = Z_{n-1}$, then $z_n = t_h z_h + \cdots + t_{n-1}z_{n-1}$ with $t_h, \ldots, t_{n-1} \in R$, and applying α we get the formula

$$x_n = t_h x_h + \cdots + t_{n-1}x_{n-1}$$

that predicts x_n from $x_0, \ldots, x_{n-1}$ without using knowledge of α.

If Z is a Noetherian R-module, then we encounter a stationary situation after finitely many steps: $Z_n = Z_l$ for $n \geq l$. Beginning with this index the complete sequence x_n is predictable by the following “algorithm”:

1. Calculate $z_n = \Phi^{(n)}(x_0, \ldots, x_{n-1})$.
2. Find a linear combination $z_n = t_h z_h + \cdots + t_{n-1}z_{n-1}$.
3. Set $x_n = t_h x_h + \cdots + t_{n-1}x_{n-1}$.

The Noetherian principle allows the prediction by a linear relation (that however might change from step to step).

To transform the “algorithm” into a true algorithm we need a procedure that explicitly finds a linear combination in step 2, solving a system of linear equations in Z.

For our standard example of a congruential generator with module $m = 8397$ (here assumed to be known), $x_0 = 2134$, $x_1 = 2160$, $x_2 = 6905$, we calculate

$$z_1 = \begin{pmatrix} 2134 \\ 1 \end{pmatrix}, \quad z_2 = \begin{pmatrix} 2160 \\ 1 \end{pmatrix}, \quad z_3 = \begin{pmatrix} 6905 \\ 1 \end{pmatrix}.$$

Trying to write z_3 as a linear combination $t_1z_1 + t_2z_2$ we get the system

$$\begin{aligned} 2134t_1 + 2160t_2 &= 6905 \\ t_1 + t_2 &= 1 \end{aligned} \tag{1}$$

of linear equations in $R = \mathbb{Z}/m\mathbb{Z}$. By elimination we find

$$26t_1 = -4745 = 3652.$$

The inverse of 26 mod 8397 is 323, and thus we get $t_1 = 4016$, $t_2 = 4382$. This result correctly predicts $x_3 = 3778$.

Proceeding in this way we correctly predict the complete output sequence. The reason is that $Z_2 = Z$:

$$z_2 - z_1 = \begin{pmatrix} 26 \\ 0 \end{pmatrix}, \quad e_1 = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \in Z_2, \quad e_2 = \begin{pmatrix} 0 \\ 1 \end{pmatrix} = z_1 - 2134 \cdot e_1 \in Z_2.$$

This example contains a partial answer to the question of when the chain of submodules Z_n becomes stationary: At least when $Z_l = Z$. But in a more general case this might never happen. Note also that from $Z_l = Z_{l+1}$ we can't conclude that the chain is stationary at Z_l—later on it could ascend again. For a bound on the number of proper increments see Proposition 5.

In each single loop of the prediction algorithm there are two possible alternative events:

- $z_n \notin Z_{n-1}$. Then predicting x_n is impossible, and Z_{n-1} properly extends to $Z_n = Z_{n-1} + Rz_n$.
- $z_n \in Z_{n-1}$. Then the algorithm correctly predicts x_n.

By Proposition 5 the first of these two events may happen at most $\log_2(\#Z)$ times (or $\operatorname{Dim} Z$ times if R is a field). For each of these events we need access to the next element x_n of the output sequence to get ahead. On first sight this looks disappointing, but some thought brings to mind that it is a realistic situtation for cryptanalysis: In the process of breaking a cipher the cryptanalyst works with a supposed key until she gets nonsense "plaintext". Then she tries to guess the following plaintext characters by context knowledge, corrects the supposed key and goes on with deciphering. Remember that we already encountered this effect in the last section. And note that the present algorithm is fairly simple but contents itself with predicting elements instead of determining the unknown parameters of the random generator.