

# Topic 4. Pseudo-random Sequence/Number Generators

- 1 One-time-pad and Design Principles of Stream Ciphers
- 2 Filter Function Generators
- 3 Combinatorial Function Generators
- 4 Clock-control Generators and Shrinking Generators
- 5 Blum-Blum Generators
- 6 Known Attacks

# 1. One-time-pad and Design Principles of Stream Ciphers

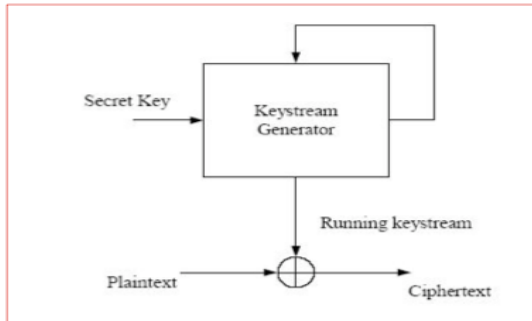


Figure: **A General Model of Stream Cipher**

# One-time-pad

- One-time-pad means that different messages are encrypted by different key streams.
- Shannon's Result (1948): One-time-pad is unbreakable. Request for large period.
- Massey's Discovery (1969): If a binary sequence has linear span  $n$  then the entire sequence can be reconstructed from  $2n$  consecutive known bits by the Berlekamp-Massey algorithm. Request for large linear span.

# Randomness Measurements for PSG

- **Long** Period
- Balance Property
- Run Property
- $n$ -tuple Distribution
- Two-level Auto Correlation and Low Cross Correlation
- Large Linear Span
- **Indistinguishability:** Indistinguishability of a pseudo-random sequence: a sequence, schematically generated by an algorithm or a device, cannot be distinguished from a truly random sequence in terms of any polynomial algorithm with negligible probability.

# Nonlinear Generators

- **Linear feedback shift register (LFSR)** sequences are widely used as basic functional blocks in key stream generators in stream cipher models due to their fast implementation in hardware as well as in software in some cases.
- In LFSR based stream ciphers, there are mainly two types of operations which operate on the LFSRs:
  - ▶ a) outputs of one LFSR or multiple LFSRs are transformed by **nonlinear functions** with or without memory states, including those by using mixed finite field operations and integer modular operations (filtering or combinatorial generators);
  - ▶ b) **change the clock** of the LFSRs to an irregular clock or by deleting some output bits of the LFSRs (clock control or shrinking generators).

## Nonlinear Generators (cont.)

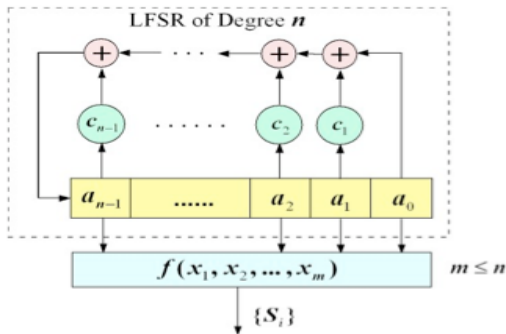
- In practice, such key stream generators include E0 in Bluetooth, and the most of the submissions in EStream Project.
- In this type of key stream generators, the **initial states** the LFSRs are served as a cryptographic key in each communication session.
- The goal of an attack is to recover the key, i.e., the initial states of the LFSRs from some **known bits** of the key stream, then generates the rest of bits of the key stream used in that particular session.
- There are many attacks proposed in the literature for the LFSR based key stream generators.
- In this chapter, we consider attacks related to solve a system of linear equations, a class of algebraic attacks, in depth.

## 2. Filtering Sequence Generators

- **Construction:** Let  $\mathbf{w} = \{w_t\}$  be an  $m$ -sequence of period  $2^n - 1$ , and  $0 \leq d_0 < d_1 < \dots < d_{m-1} < n$ . A sequence  $\mathbf{s} = \{s_t\}$  is referred to as a **filtering sequence** if

$$s_t = f(w_{d_0+t}, w_{d_1+t}, \dots, w_{d_{m-1}+t}), t = 0, 1, \dots \quad (1)$$

where  $f(x_0, x_1, \dots, x_{m-1})$  is a boolean function in  $m$  variables. The boolean function  $f$  is referred to as a **filtering function**.



# Profile of Filtering Sequences

- The period of  $\mathbf{s}$  is given by  $2^n - 1$ .
- The linear span of  $\mathbf{s}$ ,  $LS(\mathbf{s})$ , is upper bounded by

$$LS(\mathbf{s}) \leq \sum_{k=1}^m \binom{n}{k}.$$

In particular, if  $m = 2$ , then  $LS = n(n + 1)/2$ .

- If  $d_i$ 's are equally spaced, then the linear span will be lower bounded by

$$LS(\mathbf{s}) \geq \binom{n}{m}.$$

- It is not clear for other randomness properties.

### 3. Combinatorial Sequence Generators

- **Construction:** Let

$$s_t = f(w_{0,t}, w_{1,t}, \dots, w_{m-1,t}), t = 0, 1, \dots \quad (2)$$

where  $\mathbf{w}_i = \{w_{i,t}\}_{t \geq 0}$  are  $m$ -sequences with period  $2^{n_i} - 1$ , respectively. Then  $\mathbf{s} = \{s_t\}$  is called a **combinatorial sequence**.

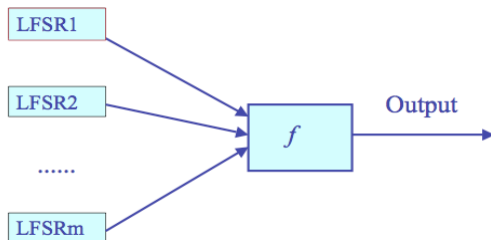


Figure: **A Diagram of the Combinatorial Sequence Generator**

# Remarks

- **In other words**, if we apply a boolean function in  $m$  variables to  $m$  LFSRs which generates  $m$ -sequences, then the output sequence of the boolean function is a combinatorial sequence.
- When the  $m$  LFSRs have the **same** characteristic primitive polynomials, then a combinatorial sequence becomes a filtering sequence. Thus a filtering sequence generator is a **special case** of combinatorial sequence generators.

# Profile of Combinatorial Sequences

## Period

- The period of  $\mathbf{s}$ ,  $per(\mathbf{s})$ , is a factor of the least common multiple of  $2^{n_i} - 1$ ,  $i = 1, \dots, m$ , i.e.,

$$per(\mathbf{s}) \mid \text{lcm}\{2^{n_i} - 1 \mid i = 1, \dots, m\}.$$

- If  $\gcd(n_i, n_j) = 1$  for any  $i \neq j$ , let  $N = \prod_{i=0}^{m-1} (2^{n_i} - 1)$ , then  $N$  is a period of  $\mathbf{s}$ . But it may not be the least period of  $\mathbf{s}$ . In the following, we only consider this special case.
- Furthermore, if the periods of any pair of  $m$ -sequences are **coprime**, then  $N$  is the least period of  $\mathbf{s}$  when all the LFSRs generates nonzero sequences.

# Profile of Combinatorial Sequences (cont.)

## Linear Span

- Let the **algebraic normal form** of  $f$  be given as follows

$$f(x_0, \dots, x_{m-1}) = \sum_{(i_1, \dots, i_r)} x_{i_1} \cdots x_{i_r} \quad (3)$$

summed over some vectors  $(i_1, \dots, i_r)$  with

$$\{i_1, \dots, i_r\} \subset \mathbb{Z}_m = \{0, 1, \dots, m-1\}, r \leq m.$$

- Let  $D$  be the set consisting of all the vectors  $(i_1, \dots, i_r)$  for which  $x_{i_1} \cdots x_{i_r}$  is a **monomial term** of  $f$ .
- If  $\gcd(n_i, n_j) = 1$  for any  $i \neq j$ , then the **linear span** of  $\mathbf{s}$  is equal to

$$LS(\mathbf{s}) = \sum_{(i_1, \dots, i_r) \in D} n_{i_1} \cdots n_{i_r}.$$

# How to select $f$ ?

- $f$  should have **large degree**, in order to have large linear span.
- In order to resistance to **correlation attack** ( Siegenthaler, 1984),  $f$  should be balanced, has low correlation from all affine function (so-called nonlinearity), and large correlation immunity. (Many tricks for selection of  $f$ !)
- In other words,  $f$  cannot be approximated by any affine functions with high probability. This is resistant to **linear cryptanalysis**.
- There is a **trade-off** between degree and correlation immunity.
- In order to be resistant to **differential cryptanalysis**, it requests that difference between  $f(\mathbf{x} + \mathbf{a})$  and  $f(\mathbf{x})$  should be large, i.e., the correlation between  $f(\mathbf{x} + \mathbf{a})$  and  $f(\mathbf{x})$  should be small.
- It should also have large **algebraic immunity**.

## 4. Clock-control Generators and Shrinking Generators

- **Clock Controlled Generator:** The basic idea is to change the clock pulse in LFSRs. For example, the clock pulse of LFSR 1 is controlled by LFSR 2.

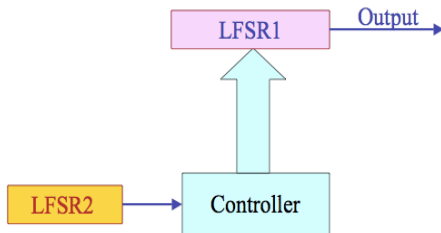


Figure: **Model of the Clock Controlled Generator and Shrinking Generator**

## A simple model of clock-control generator: stop-and-go generator

- Let two sequences  $\mathbf{a} = \{a(t)\}$  and  $\mathbf{b} = \{b_i\}$  be generated by LFSR1 and LFSR 2 respectively.
- The output sequence is denoted as  $\mathbf{u} = \{u(t)\}$ . At time instance  $t$ , assume that the previous bit is  $u(t-1) = a(i_t - 1)$ , then

$$u(t) = \begin{cases} a(i_t) & \text{if } b_t = 1 \\ a(i_t - 1) & \text{if } b_t = 0. \end{cases}$$

- In other words, if  $b_t = 1$ , then the generator outputs the current bit  $a(i_t)$  at LFSR1. Otherwise, the generator repeats the previous output bit  $a(i_t - 1)$  (equivalent to an inserting operation) of LFSR1.
- Furthermore, in formula, we have

$$u(t) = a\left(\sum_{i=0}^t b_i\right), t = 0, 1, \dots$$

## Example

- Let two sequences **a** and **b** be given by

**a** = 1 0 0 0 0 1 0 1 0 1 1 1 0 1 1 0 0 0 1 1 ...

**b** = 1 1 1 1 1 0 1 1 1 0 0 0 1 0 1 0 1 1 0 0 ...

- The **inserting process** is shown below.

1 0 0 0 0 1 0 1 0 1 1 1 0 1 1 0 0 0 1 1

Figure: **Inserting Operation in the Clock-control Generator**

- The first 20 output bits are:

1 0 0 0 0 0 1 0 1 1 1 1 0 0 1 1 1 1 1 0

The underlined bits are the ones who have been inserted.

# Shrinking Generators

- Two **input sequences** **a** and **b** are the same as those in the stop-and-go generator.
- **Output** is  $\mathbf{u} = \{u(t)\}$  whose elements are given as follows.
  - (a) Let the **previous output** be  $u(i - 1)$  where  $i = \sum_{j=0}^{t-1} b_j$  with the initial state  $u(0) = a(s)$  where  $b_0 = b_1 = \dots = b_{s-1} = 0$  and  $b_s = 1$ .
  - (b) At time instance  $t$ , if  $b_t = 1$ , then the generator **outputs the current bit**  $a(t)$  of LFSR1, i.e.,  $u(i) = a(t)$ ,  $i > 0$ . Otherwise, the generator **discards**  $a(t)$ .
- In other words, for  $i = \sum_{j=0}^t b_j$ ,  $i > 0$ , we have

$$u(i) = \begin{cases} a(t) & \text{if } b_t = 1 \\ \text{discards } a(t) & \text{if } b_t = 0. \end{cases} \quad (4)$$

# Example

- Let **a** and **b** be given by

$$\begin{array}{rcccccccccccccccccccc} \mathbf{a} & = & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & \dots \\ \mathbf{b} & = & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & \dots \end{array}$$

## Deleting Operation in the Shrinking Generator

- The first 10 output bits from the shrinking generator are:

1 0 0 1 0 0 1 0 0 1.

## 5. Blum-Blum-Shub (BBS) Generators

### $x^2 \bmod N$ Generator

- Let  $N = pq$  where  $p$  and  $q$  are distinct primes  $\equiv 3 \pmod{4}$
- Inputs are  $(N, x_0)$  where  $x_0$  is referred to as a **seed** of the generator.
- The outputs of the generator is a pseudo-random sequence  $\mathbf{s} = \{s_i\}$  whose elements are given by

**Computing the integer:**  $x_{i+1} = x_i^2 \pmod{N}, i = 0, 1, \dots$

**Extracting the bit:**  $s_i = \text{parity}(x_i)$

where the parity function is defined by

$$\text{parity}(x) = \begin{cases} 1 & \text{if } x \text{ is odd} \\ 0 & \text{if } x \text{ is even} \end{cases}$$

- The BBS generator can be considered as a **filtering generator** where the LFSR is an NLFSR of one stage with the feedback function  $x^2 \pmod{N}$ , and the filtering function is the parity check function which maps  $\log N$  bits to one bit.

# BBS as a Filtering Model

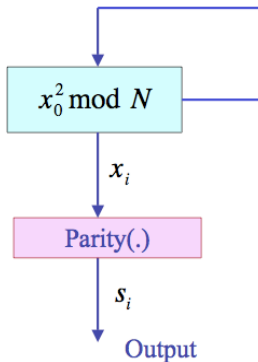


Figure: **A Diagram of BBS as a Filtering Generator**

## Example

- Let  $N = 7 \times 19 = 133$  and  $x_0 = 4$ .
- Then the sequence  $x_0, x_1 = x_0^2 \pmod{133}, \dots$  has period 6:

$$\begin{aligned}\{x_i\} &= 4, 16, 123, 100, 25, 93, \dots \\ \{s_i\} &= 0 \quad 0 \quad 1 \quad 0 \quad 1 \quad 1 \quad \dots\end{aligned}$$

- Note that  $\lambda(133) = 18$  and  $\lambda(\lambda(133)) = 6$  where  $\lambda(n)$  is defined below.

# Property of BBS Generators

- The BBS generator is **unpredictable**, in the sense that given  $x_0$  and  $N$ , but not the factors of  $N$ , a cryptanalyst cannot effectively make any prediction about the values of  $x_{-1}, x_{-2}, \dots$ . Or equivalently, given a portion of the key stream  $\{s_i\}$ , it is computational infeasible to recover the seed  $x_0$  by knowing  $N$  without knowing the factors of  $N$ .
- For positive integers

$$n = 2^e p_1^{e_1} \cdots p_k^{e_k}, \text{ where } p_1 < p_2 < \cdots < p_k \text{ are odd primes,}$$

we define the lambda function  $\lambda(n)$  by

$$\begin{aligned} \lambda(2) &= 1, \lambda(4) = 2, \lambda(2^e) = 2^{e-2}, e > 2 \\ \lambda(p^r) &= \phi(p^r) = p^{r-1}(p-1) \text{ for } p \text{ odd} \\ \lambda(n) &= \text{lcm}\{\lambda(2^e), \lambda(p_1^{e_1}), \dots, \lambda(p_k^{e_k})\}. \end{aligned}$$

## Property of BBS Generators (cont.)

- If  $x_0$  is a **quadratic residue** modulo  $N$ , then the period of the BBS sequence  $\mathbf{s}$  satisfies

$$\text{per}(\mathbf{s}) \mid \lambda(\lambda(N)).$$

- The **imbalanced range** of  $\mathbf{s}$  is defined by

$$I(\mathbf{s}) = | \text{number of 0's in } \mathbf{s} - \text{number of 1's in } \mathbf{s} |.$$

Then the **average of imbalance** is with order no larger than  $N^{1/4} \log N$ .

- Note that the **average** of imbalance for a random sequence with period  $\lambda(\lambda(N))$  is with order  $\sqrt{N}$ , which is much bigger than  $N^{1/4} \log N$ .
- In this sense, the BBS generator is better than the random sequences of the **same period**  $\lambda(\lambda(N))$ .

## 6. Known Attacks

- **Correlation** attacks
- Linear cryptanalysis
- **Differential** cryptanalysis
- Time and memory trade-off attacks
- **Algebraic** attacks

# Algebraic Attacks

**Goal:** Find the seed in the generator, or the initial state in the LFSRs involved in the generator.

- **Linearizations**
- **Algebraic** Attacks
- Fast Algebraic Attacks
- **Selective** Discrete Fourier Transform (DFT) Attacks

# Algebraic Attacks (cont.)

- **Algebraic** attacks and fast algebraic attacks have been shown as an important cryptanalysis method for symmetric-key cryptographical systems in recent work by many researchers.
- Especially, it significantly improved efficiency of attacks on **stream cipher** systems in which key streams are generated by linear feedback shift register based systems.
- Those attacks usually contain **three steps**: (a) **pre-computation**, (b) **substitution** for establishing a system of linear equations over  $\mathbb{F}_2$  or  $F_{2^n}$  from known key stream bits, and (c) **solving** the system.

## A. Equations with Unknown Keys

- In a stream cipher model, a ciphertext is a bit stream  $c_0, c_1, \dots$ , obtained by exclusive-or a message bit stream  $m_0, m_1, \dots$ , with the key stream  $s_0, s_1, \dots$ , i.e.,

$$c_i = m_i + s_i, i = 0, 1, \dots, \text{ in } \mathbb{F}_2.$$

- One of strong attacks comes from **known plaintext attacks**, i.e., if a certain plaintext is known, then some bits of  $\{s_t\}$  can be recovered.
- If the key can be recovered from **those known bits** of  $\{s_t\}$ , then the rest of bits of the key stream, i.e., all bits of  $\{s_t\}$ , can be reconstructed.

# Known Plaintext Attack

- An **initial state** of the LFSR is a key when  $\{s_t\}$  is served as a key stream generator, denoted by  $K = (k_0, \dots, k_{n-1})$ ,  $k_i \in \mathbb{F}_2$ .
- Then

$$s_t = f(L^t(k_0, k_1, \dots, k_{n-1})) = f_t(k_0, \dots, k_{n-1}), t = 0, 1, \dots \quad (5)$$

where

$$f_t(x_0, \dots, x_{n-1}) = f(L^t(x_0, \dots, x_{n-1})).$$

- **Known plaintext attack**: if a certain plaintext is known, then some bits of  $\{s_t\}$  can be recovered. If the key can be recovered from those known bits of  $\{s_t\}$ , then the rest of bits of the key stream can be reconstructed.

## B. Linearization

**Question:** How can one efficiently solve (5) with minimized known bits of the key stream?

- The system of the equations (5) can be **linearized** when each monomial in  $k_{i_1} \cdots k_{i_s}$  is treated as a variable.
- **The number of unknowns** in (5) is varied, but it is dominated by the degree of  $f$ .
- For **filtering function generators**, i.e., apply  $f$  on  $m$  tap positions of an LFSR of degree  $n$ , the number of unknown in (5) is upper bounded by  $T_{deg(f)}$  where  $deg(f)$  is the degree of  $f$  and  $T_j$  is defined as

$$T_j = \sum_{i=0}^j \binom{n}{i}. \quad (6)$$

## B. Algebraic Attacks

- The algebraic attack is to **multiply**  $f$  by a function  $g$  with a degree lower than  $f$  such that the product  $fg$  is zero. In other words, using  $g$  with  $\deg(g) < \deg(f)$  such that  $fg = 0$ , we have the system of the linear equations as follows

$$s_t g_t(K) = 0, t = 0, 1, \dots \quad (7)$$

- In this case, the number of the unknowns of (7) is now **dominated by the degree of  $g$**  instead of  $f$ .
- For example, in the filtering generators, this is equal to  $T_{\deg(g)}$ . However

$$T_{\deg(g)} < T_{\deg(f)}.$$

# Comparisons

- Compared with the system of the linear equations directly from linearization, the algebraic attack can reduce the complexity for solving a system of linear equations by **decreasing the number of unknowns** in the system of linear equations as well as reducing the number of the required known bits of the key stream.
- From this result, study for algebraic immunity of boolean functions is in fashion for **resistance** against this attack.

# Algebraic Immunity

- Let  $\mathcal{B}_n$  be the set consisting of all boolean functions in  $m$  variables. The **algebraic immunity** of  $f$  is defined as the smallest degree  $\deg(g)$  such that  $fg = 0$  or  $(1 + f)g = 0$ , denoted by  $AI(f)$ , i.e.,

$$AI(f) = \min_{g \in \text{Ann}(f)} \deg(g), \quad (8)$$

where

$$\text{Ann}(f) = \{g \in \mathcal{B}_n \mid fg = 0 \text{ or } g(f + 1) = 0\}.$$

## C. Fast Algebraic Attacks

- The fast algebraic attack (FAA) (2003) on stream ciphers is to **accelerate the algebraic attack** by introducing linear relations among the key stream bits.
- The **idea** is that if we can find some  $g$  such that  $h = fg$  where  $\deg(g) < AI(f)$ . In this way, one could further reduce the number of the unknowns in the linear equations.
- **Note** that

$$h = fg \implies f(g + h) = 0 \implies g + h \in \text{Ann}(f).$$

But  $\deg(g + h)$  may be greater than  $AI(f)$ .

## C. Fast Algebraic Attacks (cont.)

- FAA consists of **two steps**:
  - (a) **find a boolean function**  $g$  with  $d = \deg(g) < \deg(f)$  such that the product  $h = fg \neq 0$  with  $e = \deg(h) > 0$  where  $d < e$ ;
  - (b) **compute**  $q(x)$  which is a characteristic polynomial of the output sequence or a factor of it, and **apply**  $q(x) = \sum_i c_i x^i$  to  $s_t g_t(K) = h_t(K)$  which results in

$$\sum_{i=0}^r c_i s_{i+t} g_{i+t}(K) = \sum_{i=0}^r c_i h_{i+t}(K). \quad (9)$$

- If  $v_t = \sum_{i=0}^r c_i h_{i+t}(K)$ ,  $t = 0, 1, \dots$  is equal to zero, then (9) is the system of linear equations in at most  $T_d$  variables which can be solved by known  $T_d + T_e$  **consecutive bits**.
- If we choose  $h(x)$  such that  $\{v_t\}$  is a **nonzero** sequence, then (9) by known  $T_e$  **consecutive bits**, which is less than the case that  $\{v_t\}$  is a zero sequence.

## C. Fast Algebraic Attacks (cont.)

- According to the above analysis, **the number of unknowns** in (9) is small than the number of unknowns in the algebraic attack.
- But in FAA, the known bits of the key stream should be **consecutive**. However this condition is **not needed in the algebraic attack**.

## D. Selective DFT Attacks

- More recently, it is proposed a new method to recover an initial state in a filtering sequence generator by reducing the number of unknowns in the system of linear equations to the **minimum**, which is the degree of the LFSR by an **increased complexity** in the both pre-computation and substitution, especially, in the substitution step.
- In those methods, the **pre-computation and substitution** are done by forming a system of linear equations over  $GF(2^n)$  instead of linear equations over  $\mathbb{F}_2$ .
- **The number of the required consecutive bits**, say  $j$ , is reduced to the linear span of the filtering sequence, denoted as  $LS(\mathbf{s})$ .
- **If  $j \geq LS(\mathbf{s})$** , then the number of the unknowns is reduced to  $n$  which is the degree of the LFSR.
- **If  $j < LS(\mathbf{s})$** , the FAA fails. However, the system of the equations is solvable by the selective DFT method.

## D. Selective DFT Attacks (cont.)

- In other words, the select DFT attack results in either a **more efficient** attack than FAA or it can work for the case that the number of the known consecutive bits of the key stream is too small to apply FAA.
- However, this method needs to know the **exact linear span** of the key stream sequence  $\{s_t\}$  and their respective discrete Fourier transform (DFT) spectra of the multiplier sequence and the product sequence, which are not needed in FAA.
- All the **solving equations related attacks** can be considered as a certain special case of the selective DFT attacks.

# A General Model

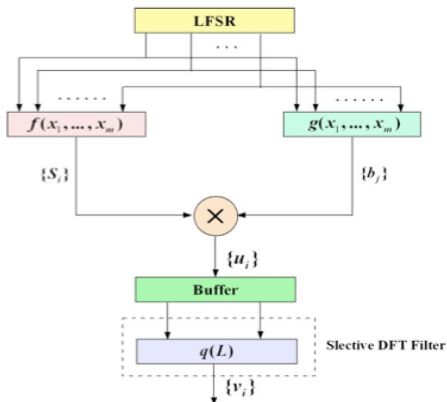


Figure: **Model of Solving Equations Related Attacks on the Filtering Generator**