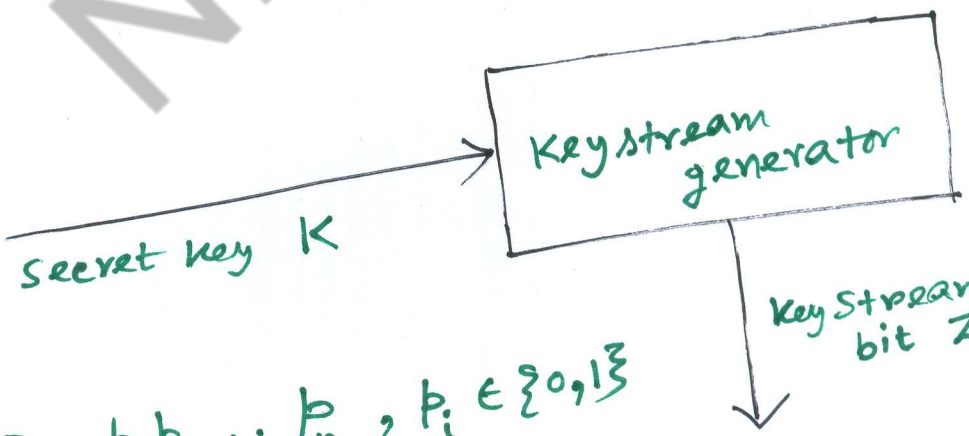
Week 11

Topics: Cryptanalysis on Stream Cipher
Modern Stream Ciphers
Shamir's secret sharing and BE
Identity based Encryption (IBE)
Attribute based Encryption (ABE)

Cryptanalysis on Stream Cipher :

Stream cipher →



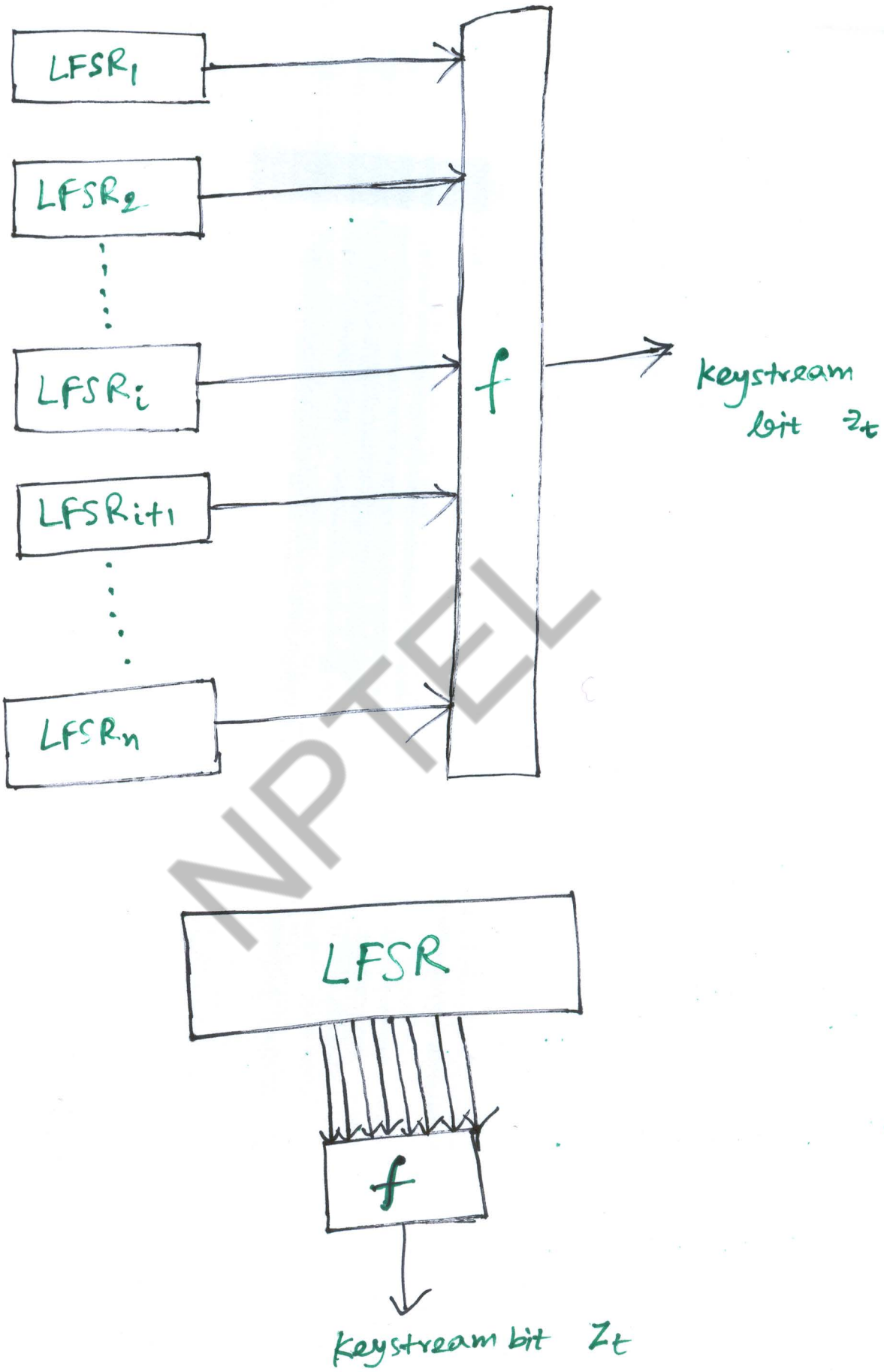
plaintext $P = p_0 p_1 \cdots p_n$, $p_i \in \{0,1\}$

Ciphertext $C = c_0 c_1 \cdots c_n$

Encryption: $c_i = p_i \oplus z_i$

Decryption: $p_i = c_i \oplus z_i$

LFSR based stream cipher :



- So the expression of the keystream bit at t-th clocking will be $Z_t = f(S^t)$

Consider one LFSR based stream cipher.

- The linear feedback function of the LFSR is l.
- Firstly, we initialize the state of the LFSR by one secret key K, i.e., $s_i = k_i$ for $i = 0, \cdots, n-1$, where, the secret key bits are denoted by k_i, $i = 0, \cdots, n-1$ and the state bits are denoted by s_i, $i = 0, \cdots, n-1$.
- State update function,

$$S^t = \begin{cases} s_{i-1}^t = s_i^{t-1} & \forall i = 1, \cdots, n-1 \\ s_{n-1}^t = l(s_0, \cdots, s_{n-1}). \end{cases}$$

- key stream bit $\Rightarrow f(S^t) = z_t$

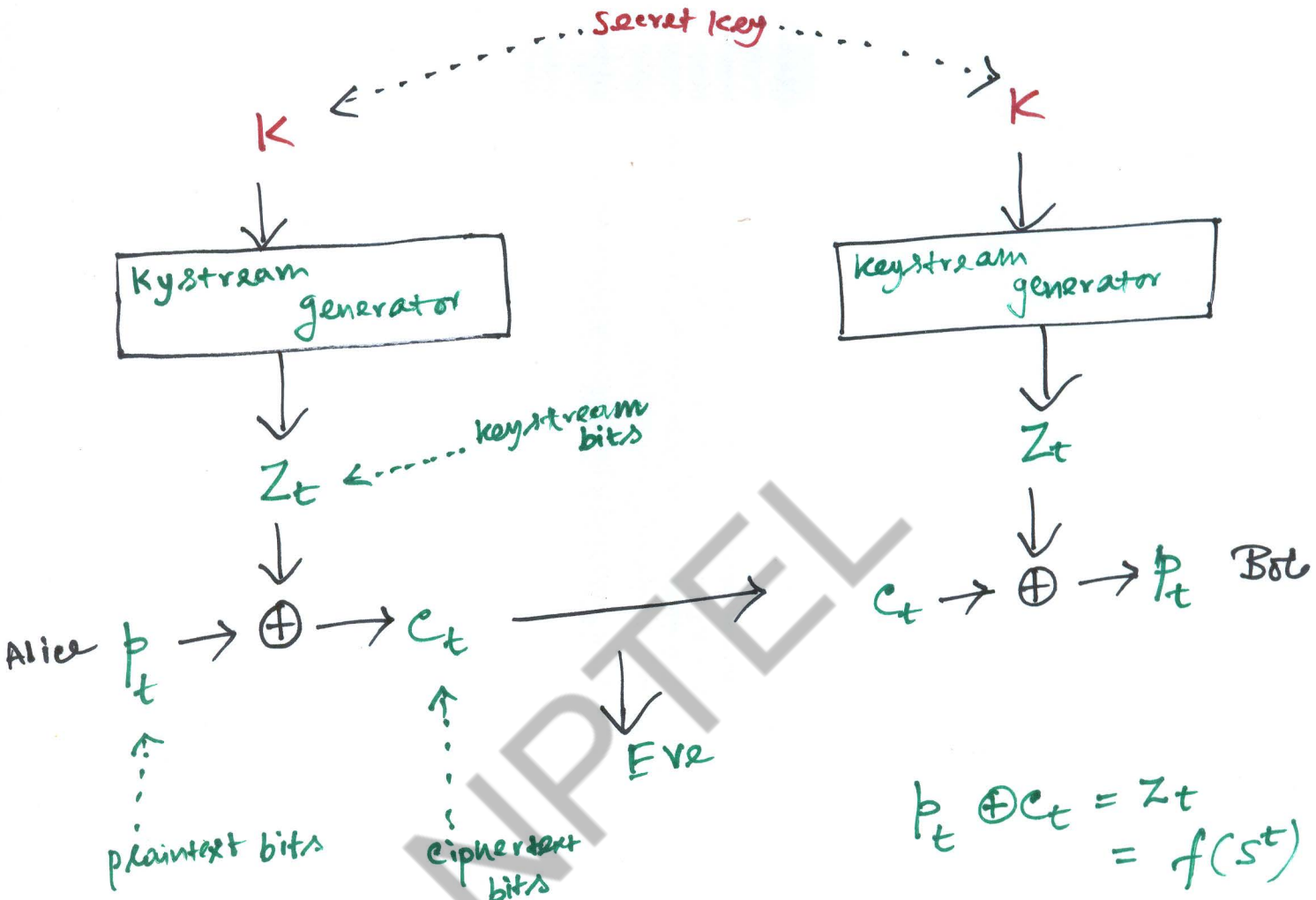
Algebraic attack :

Expressing the whole cryptosystem as a large system of multivariate algebraic equations which can be solved to recover the secret key.

Algebraic attack is based on two basic strategies –

- Construct a system of algebraic equations involving the secret key using some known plaintext bits and corresponding ciphertext bits.
- Solve the system to find the secret key.

Generating the multivariate equations



$$p_t \oplus c_t = Z_t = f(S^t)$$

So, if attacker knows n plaintext bits and corresponding ciphertext bits then he can construct the following system of equations ?

$$f(L^{K_1}(K^0)) = Z_{K_1}$$

$$f(L^{K_2}(K^0)) = Z_{K_2}$$

$$\vdots$$

$$f(L^{K_n}(K^0)) = Z_{K_n}$$

where

- $K^0 \Rightarrow$ secret key.
- $L \Rightarrow$ state update function of the LFSR.
- $f \Rightarrow$ Nonlinear filter function used in the cipher.
- $Z_t \Rightarrow$ keystream bit at t-th clocking.

- Question: How can we solve the system?
- Ans: There are some algorithms to find the solution of this type of system, e.g., Linearization, Re-linearization, XL algorithm.

Linearization:

- The basic idea of this algorithm is to linearize the total system by introducing new variables in the system of equations and finally solve by Gauss elimination method.

- The maximum number of monomials can be present in a system of equations with n variables of degree maximum d is $\leq \sum_{i=1}^{d} \binom{n}{i}$.

- So, to linearize one system of equations of n variables of degree maximum d we need maximum $\sum_{i=1}^{d} \binom{n}{i}$ number of new variables to linearize the final system.

- Final system will have unique solution if the number of independent equation is same as the number of variables present in the final system.

- But in many cases it may not happen.

Then we need to think for generating some extra linearly independent equations.

Re-linearization :

- We know that multiplication of the variables is a commutative operation, in this re-linearization method we use this commutativity property to generate more equations.

- Consider one example, suppose we have this monomial $x_1x_2x_3x_4$. Now we can rewrite this monomial in different ways

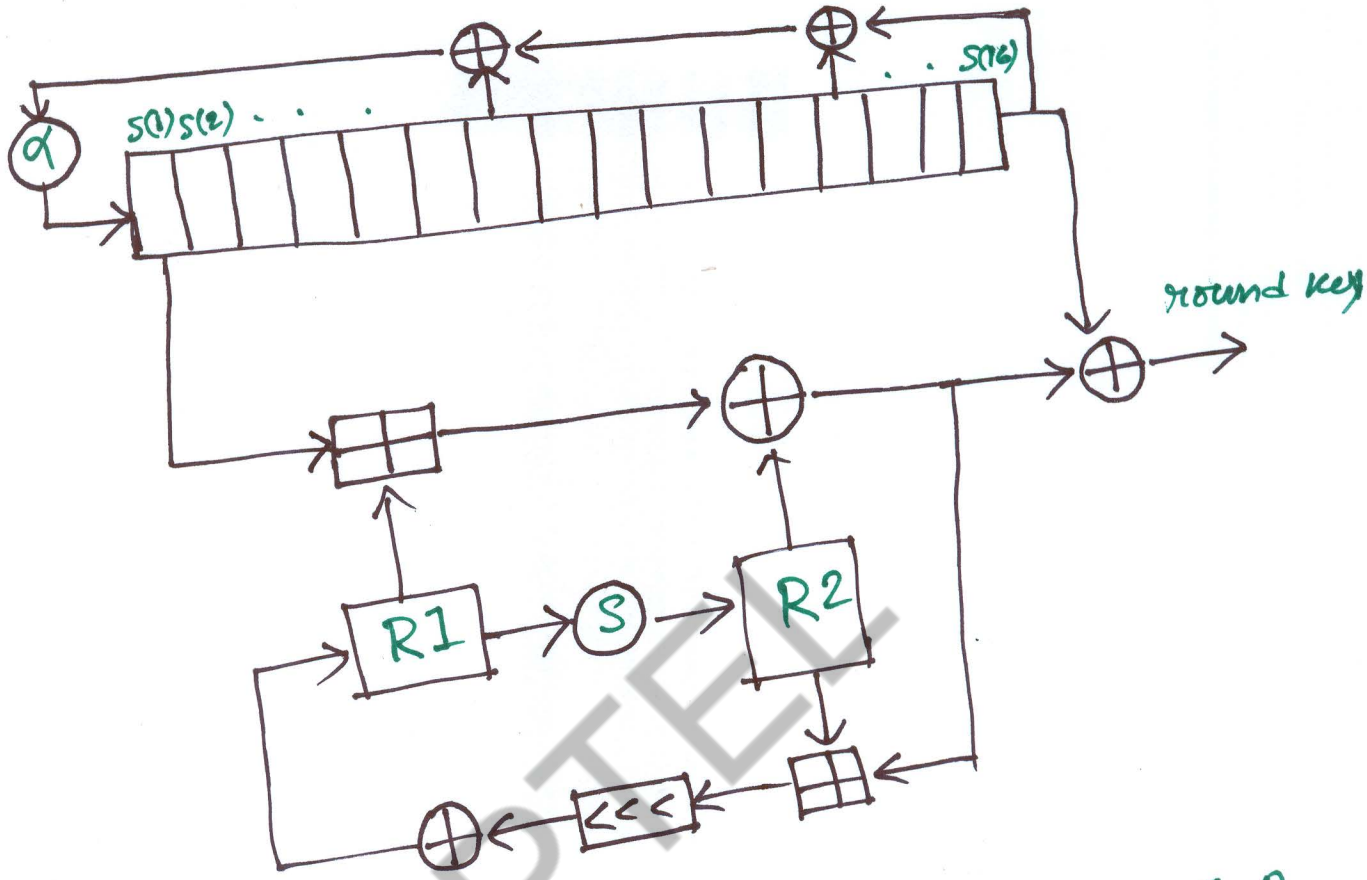
$$(x_1x_2)(x_3x_4) = (x_1x_3)(x_2x_4) = (x_1x_4)(x_2x_3)$$

- We rename each form as different variables say, $y_{ij} = x_ix_j$.

- After substituting these new variables we will get two new linearly independent equations $y_{12}y_{34} = y_{13}y_{24}$ and $y_{12}y_{34} = y_{14}y_{23}$.

- It was observed that many equations generated by re-linearization algorithm are linearly independent.

SNOW 1.0 :



Design Specification of SNOW 1.0

- One 16 bit LFSR over $\mathbb{F}_{2^{32}}$, feeding a finite state machine.
- The FSM consists of two 32 bit registers called R_1 and R_2, as well as some operations to calculate the output and the next state (the next value of R_1 and R_2).

- First, key initialization is done. This procedure provides initial values for the LFSR as well as for the R_1, R_2 registers in the finite state machine.

- Next, the first 32-bits of the keystream is calculated by bitwise adding the output of the FSM and the last entry of the LFSR.

- After that the whole cipher is clocked once, and the next 32 bits of the keystream is calculated by again bitwise adding the output of the finite state machine and the last entry of the LFSR.

- We clock again and continue in this fashion.

- The primitive polynomial (over $\mathbb{F}_{2^{32}}$) corresponding to the feedback function of the LFSR is,

$$p(x) = x^{16} + x^{13} + x^7 + \alpha^{-1}$$

where $\mathbb{F}_{2^{32}}$ is generated by the irreducible polynomial

$$\pi(x) = x^{32} + x^{29} + x^{20} + x^{15} + x^{10} + x + 1$$

over $\mathbb{F}_2$, and $\pi(\alpha) = 0$.

- Let $s(1), s(2), \ldots, s(16) \in \mathbb{F}_{2^{32}}$ be the state of the LFSR.

- The output of the FSM, called FSM_{out}, is calculated as follows:

$$FSM_{out} = (s(1) \boxplus R_1) \oplus R_2$$

- The output of the FSM is XORed with $s(16)$ to form the key stream, i.e.,

$$\text{running key} = FSM_{out} \oplus s(16).$$

- The key stream is finally XORed with the plaintext, producing the ciphertext.

- Inside the FSM, the new values of R_1 and R_2 are given as follows,

$$\text{new } R_1 = ((FSM_{out} \boxplus R_2) <<<) \oplus R_1$$

$$R_2 = S(R_1),$$

$$R_1 = \text{new } R_1.$$

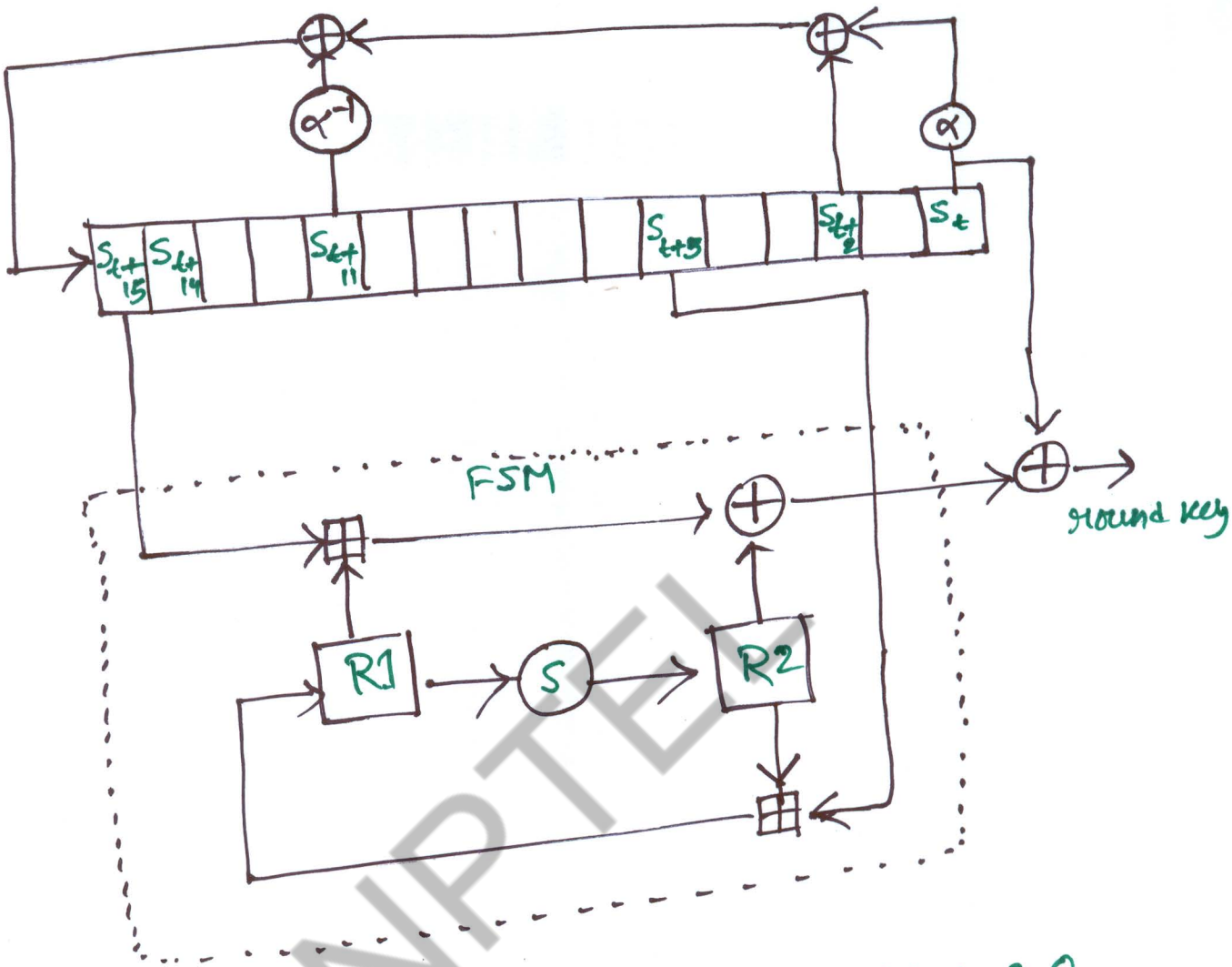
- $x \boxplus y$ means integer addition of x and y mod 2^{32}.

- $x <<<$ means cyclic shift of x, 7 steps to the left.

- $x \oplus y$ means bitwise XOR.

- The S-box $S(x)$ consists of four identical 8-to-8 bit S-boxes and a permutation of the resulting bits.

SNOW 2.0 :



Design specification of SNOW 2.0

- The feedback polynomial of SNOW 2.0 is given by

$$\pi(x) = \alpha x^{16} + x^{14} + \alpha^{-1} x^5 + 1 \in \mathbb{F}_{2^{32}}[x]$$

where α is a root of $x^4 + \beta^{23} x^3 + \beta^{245} x^2 + \beta^{48} x + \beta^{239} \in \mathbb{F}_{2^8}[x]$, and β is a root of $x^8 + x^7 + x^5 + x^3 + 1 \in \mathbb{F}_2[x]$

- The input to the FSM is (s_{t+15}, s_{t+5}) and the output of the FSM, denoted by F_t, is calculated as

$$F_t = (s_{t+15} \boxplus R1_t) \oplus R2_t, \quad t \geq 0.$$

- key stream bit at t-th clocking

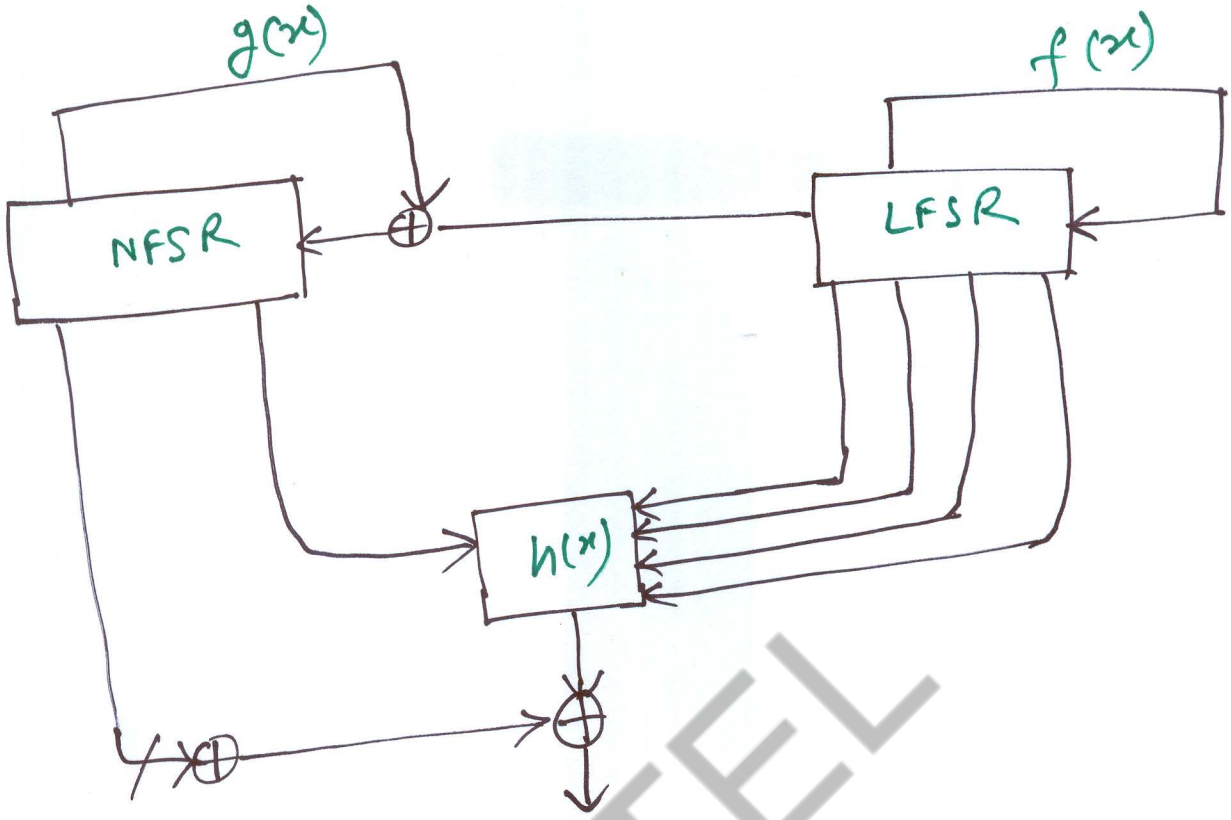
$$Z_t = F_t \oplus s_t, \quad t \geq 1.$$

- The registers R1 and R2 are updated with new values according to

$$R1_{t+1} = s_{t+5}$$
$$R2_{t+1} = S(R1_t).$$

- The S-box S is a permutation on $\mathbb{Z}_{2^{32}}$ (which is different from SNOW 1.0)

Grain V1 Stream Cipher :



Design specification of Grain V1

- Linear feedback function of the LFSR is

$$s_{i+80} = s_{i+62} + s_{i+51} + s_{i+38} + s_{i+23} + s_{i+13} + s_i$$

- Non linear feedback function of NFSR is given below,

$$b_{i+80} = s_i + b_{i+62} + b_{i+60} + b_{i+52} + b_{i+45} + b_{i+37} + b_{i+33} + b_{i+28} + b_{i+21} + b_{i+14} + b_{i+9} + b_i + (b_{i+63} b_{i+60}) + (b_{i+37} b_{i+33}) + (b_{i+15} b_{i+9}) + (b_{i+60} b_{i+52} b_{i+45}) + (b_{i+33} b_{i+28} b_{i+21}) + (b_{i+63} b_{i+45} b_{i+28} b_{i+9}) + (b_{i+60} b_{i+52} b_{i+37} b_{i+33}) + (b_{i+63} b_{i+60} b_{i+21} b_{i+15}) + (b_{i+63} b_{i+60} b_{i+52} b_{i+45} b_{i+37}) + (b_{i+33} b_{i+28} b_{i+21} b_{i+15} b_{i+9}) + (b_{i+52} b_{i+45} b_{i+37} b_{i+33} b_{i+28} b_{i+21}).$$

- The algebraic normal form of this non-linear filter function $h(\cdot)$ is given by the expression,

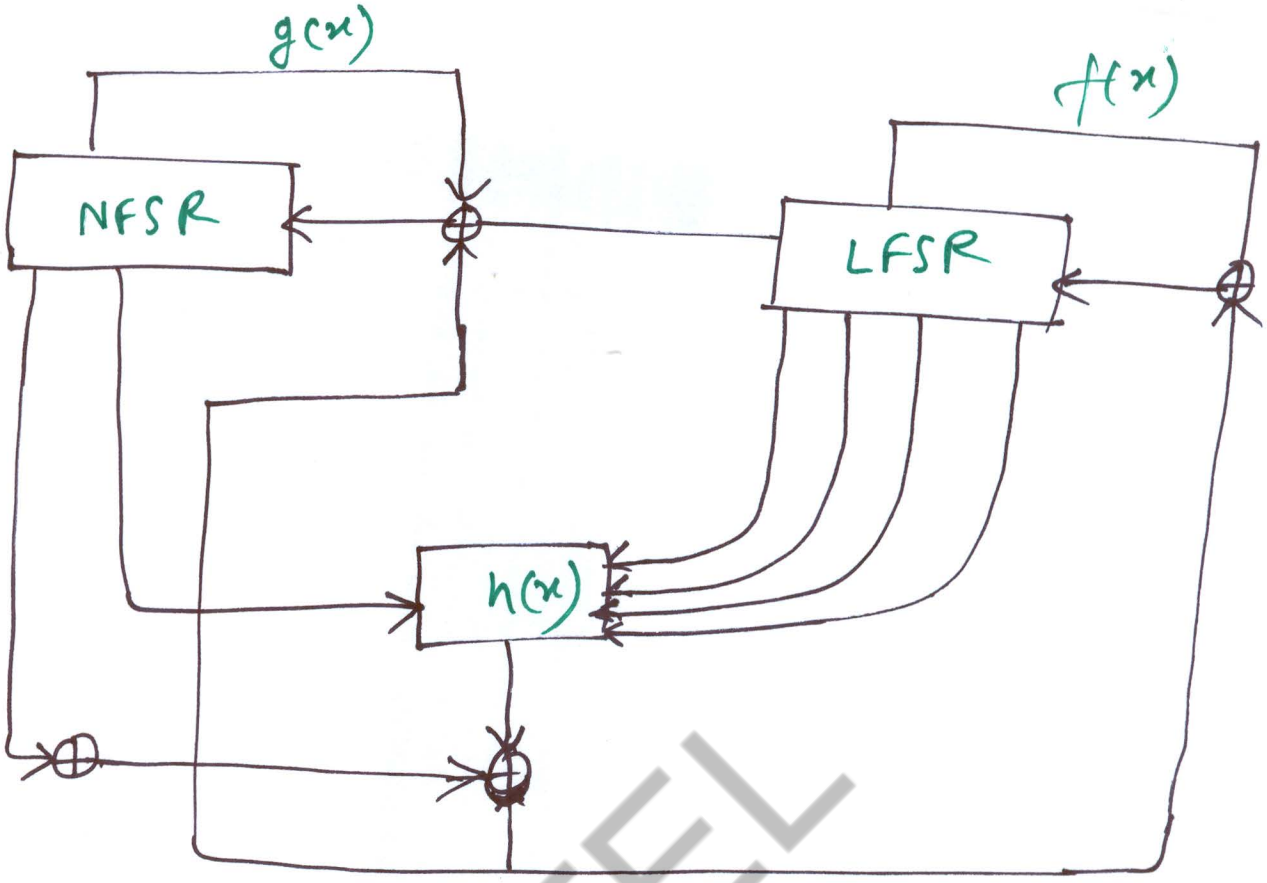
$$h(x) = x_1 + x_4 + x_0x_3 + x_2x_3 + x_3x_4 + x_0x_4x_2 + x_0x_2x_3 + x_0x_2x_4 + x_1x_2x_4 + x_2x_3x_4.$$

where x_0, x_1, x_2, x_3 variables correspond to the LFSR state bits s_{i+3}, s_{i+25}, s_{i+46}, s_{i+64} and the variable x_4 corresponds to the NFSR state bit b_{i+63}.

- The algebraic normal form of the keystream bits of the cipher at any clocking is given by the following expression,

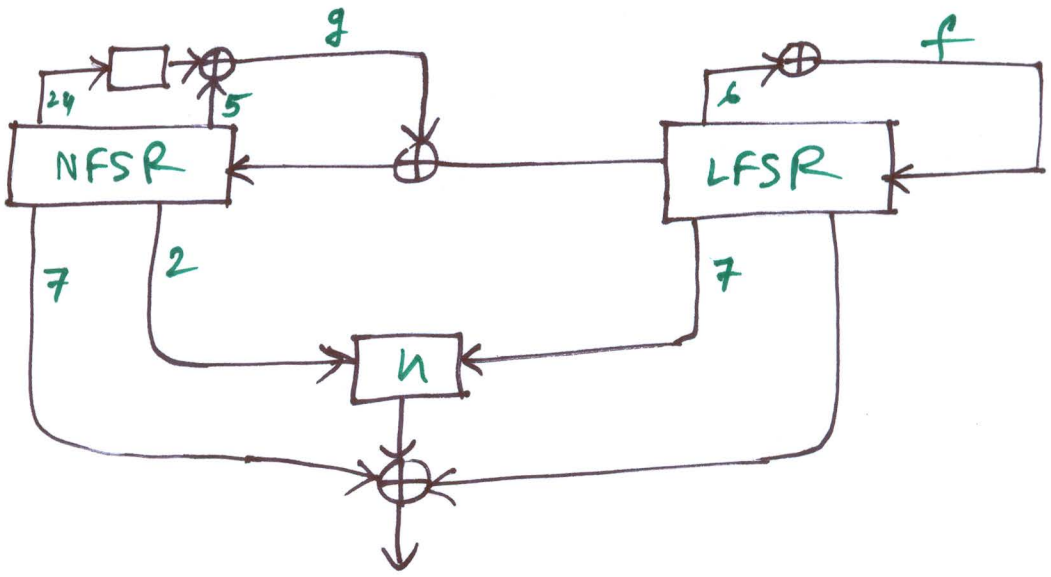
$$z_i = \sum_{k \in A} b_{i+k} + h(s_{i+3}, s_{i+25}, s_{i+46}, s_{i+64}, b_{i+63}),$$

where $A = \{1, 2, 4, 10, 31, 43, 56\}$



Key-IV initialization of Grain V1

Grain-128a Stream Cipher:



Design specification of Grain-128a:

- Linear feedback function of the LFSR is,

$$s_{i+128} = s_i + s_{i+7} + s_{i+38} + s_{i+70} + s_{i+81} + s_{i+96}$$

- Nonlinear feedback function of the NFSR is,

$$b_{i+128} = s_i + b_i + b_{i+26} + b_{i+56} + b_{i+91} + b_{i+96} + (b_{i+3}\, b_{i+67}) + (b_{i+11}\, b_{i+13}) + (b_{i+17}\, b_{i+18}) + (b_{i+27}\, b_{i+59}) + (b_{i+40} b_{i+48}) + (b_{i+61}\, b_{i+65}) + (b_{i+68}\, b_{i+84}) + (b_{i+88}\, b_{i+92}\, b_{i+93}\, b_{i+95}) + (b_{i+22}\, b_{i+24}\, b_{i+25}) + (b_{i+70}\, b_{i+78}\, b_{i+82})$$

- The algebraic normal form of the non-linear filter function is,

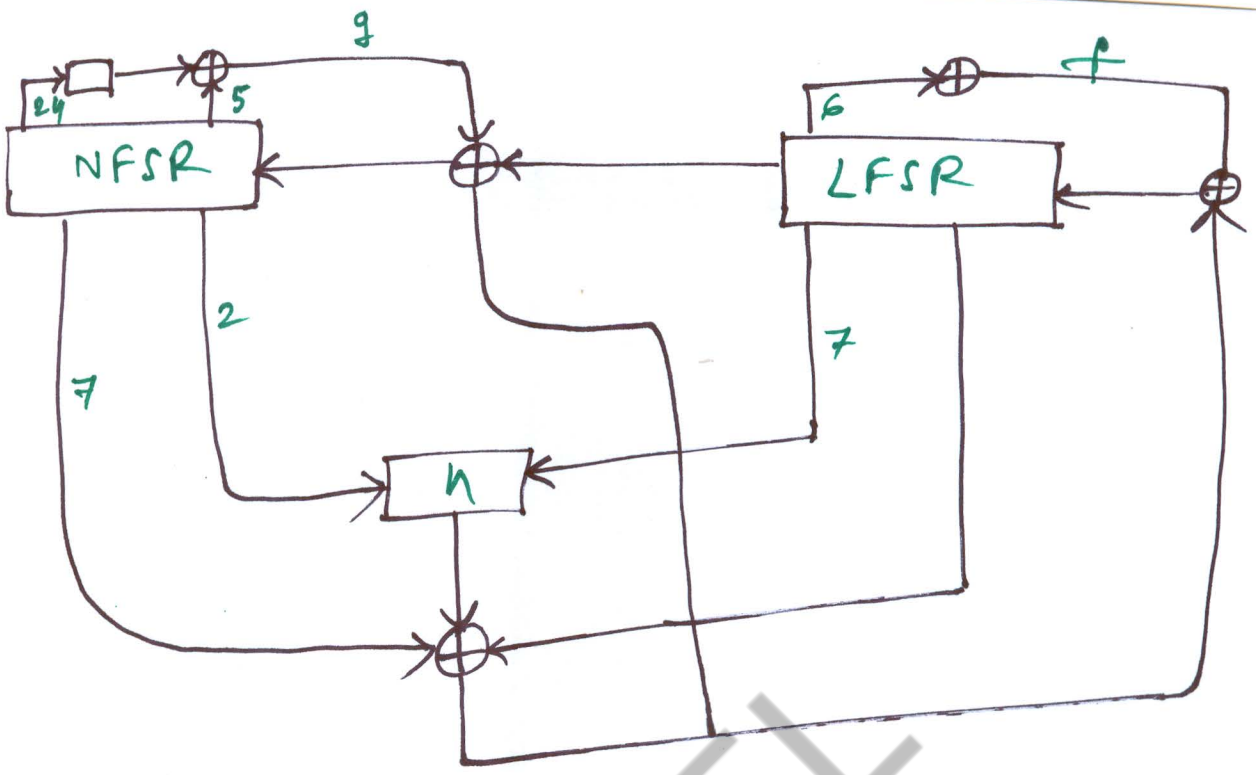
$$h(x) = x_0 x_4 + x_2 x_3 + x_4 x_5 + x_6 x_7 + x_0 x_4 x_8.$$

where $x_0, x_4, \ldots, x_8$ correspond to b_{i+12}, s_{i+8}, s_{i+13}, s_{i+20}, b_{i+95}, s_{i+42}, s_{i+60}, s_{i+70} and s_{i+94} respectively.

- output bit expression →
  The output function of the cipher is given by,

$$y_i = h(x) + s_{i+93} + \sum_{j \in A} b_{i+j}$$

where $A = \{2, 15, 36, 45, 64, 73, 89\}$



key - IV initialization of Grain - 128a

- Keystream generation phase of Grain-128a.
  - If $iv_0 = 1$ then the authentication mode is on and the keystream bits will be $z_i = y_{2i+64}$.
  - If $iv_0 = 0$ then the authentication mode is off and the keystream bits will be $z_i = y_i$.

Helix stream Cipher :

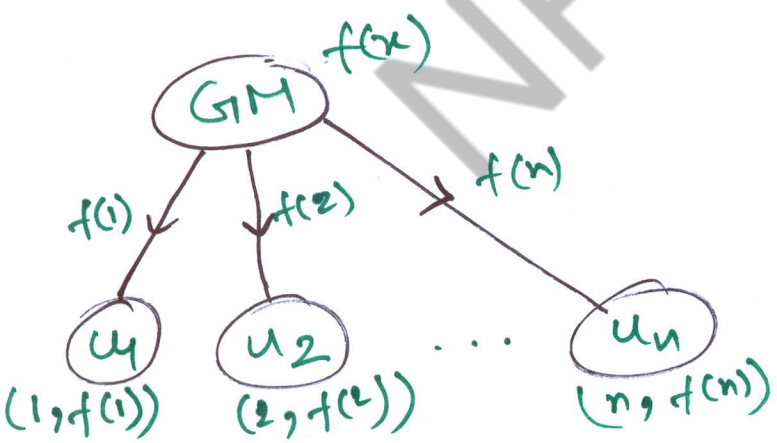
- Helix operators on 32-bit words.
- At the start of block i the state consists of 5 words: $Z_0^{(i)}, \ldots, Z_4^{(i)}$.
- At the end of the block the state consists of $Z_0^{(i+1)}, \ldots, Z_4^{(i+1)}$ which form the input to the next block with number $(i+1)$.
- Block i also uses as input two key words $X_{i,0}$ and $X_{i,1}$, and the plaintext word P_i. It produces one word of key stream $S_i := Z_0^{(i)}$.
- The ciphertext words are $C_i := P_i \oplus S_i$.

Shamir's Secret Sharing and BE :

- A (K, n) threshold scheme is a scheme where a secret s can be divided into n pieces, in such a way that the knowledge of k or more of the n pieces allows s to be recovered, but knowledge of (k-1) or fewer provides no information about s.

- Consider Shamir's secret sharing:

  (t, n) threshold scheme.



K → secret key to be divided into n pieces

GM → Group Manager

Ui → ith user

i → ID of user Ui.

- GM choose a polynomial

$$f(x) = K + a_1 x + a_2 x^2 + \cdots + a_{t-1} x^{t-1}$$

of degree (t-1) such that f(0) = K is the secret key to be shared.

- Let $u_1, \ldots, u_n$ are n users and for simplicity we take ID of user u_i is "i".
- u_i sends his/her ID to GM to get $f(i)$. Hence u_i has $(i, f(i))$.
- Let $A = \{u_{i_1}, \ldots, u_{i_t}\}$ be a group of t-users.
- The A has t points $(i_1, f(i_1)), \ldots, (i_t, f(i_t))$.
- Since these points may not be equidistance points, we can apply Lagrange's interpolation formula to recover $f(x)$ from these t points.
- Therefore, A can get $f(x)$ from those t points. Hence A can obtain $f(0) = K$, the secret key.
- Also we note that if the size of A is less than t then A can not get $f(x)$ and hence K is hidden.

Example: Let $K = 1234$, $n = 6$, $t = 3$.

Take, $(t-1)$ - degree polynomial

$$f(x) = 1234 + 166x + 94x^2.$$

$u_1 = (1, f(1)) = (1, 1494)$

$u_2 = (2, 1942)$, $u_3 = (3, 2578)$,

$u_4 = (4, 3402)$, $u_5 = (5, 4414)$,

$u_6 = (6, 5614)$.

Let $A = \{u_2, u_4, u_5\}$. Then compute,

$$l_0 = \frac{x - x_1}{x_0 - x_1} \cdot \frac{x - x_2}{x_0 - x_2} = \frac{1}{2}x^2 - \frac{3}{2}x + \frac{10}{3}.$$

$$l_1 = \frac{x - x_0}{x_1 - x_0} \frac{x - x_2}{x_1 - x_2} = \frac{1}{2}x^2 + \frac{7}{2}x - 5$$

$$l_2 = \frac{x - x_0}{x_2 - x_0} \frac{x - x_1}{x_2 - x_1} = \frac{1}{3}x^2 - 2x + \frac{8}{3}.$$

where we took, for $i = 1, 2, \ldots$, $u_i = (x_i, f(x_i)) = (x_i, y_i)$.

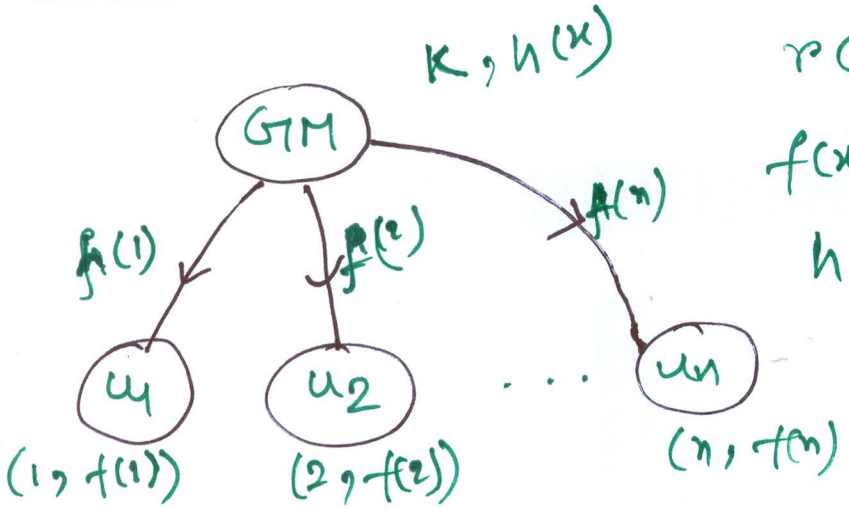
where we took, $u_2 = (x_0, y_0)$, $u_4 = (x_1, y_1)$

$u_5 = (x_2, y_2)$

Therefore, $$f(x) = \sum_{j=0}^{2} y_j l_j(x) = 1234 + 166x + 94x^2$$

Hence, the secret is $f(0) = 1234 = K$

Application to Broadcast Encryption (BE) :



$r(x) = (x-i_1)(x-i_2)\cdots(x-i_k)$

$f(x) = a_0 + a_1 x + \cdots + a_{t-1} x^{t-1}$

$h(x) = K\, r(x) + f(x)$

- Let K = session key (secret key).
- Let revoked users $R = \{u_{i_1}, u_{i_2}, \cdots, u_{i_k}\}$. So, users in R should not get the message or in particular cannot achieve the secret key K.
- GM computes revoked polynomial $r(x) = (x-i_1)\cdots(x-i_k)$ and choose a $(t-1)$-degree polynomial $f(x)$ to compute a polynomial $h(x) = K r(x) + f(x)$ to hide the secret key K.
- GM broadcasts $h(x)$ and R.
- Therefore an user U_i such that $i \notin R$ can compute K :

- U_i has $(i, f(i))$.
- U_i computes $h(i)$
- from R, U_i can compute the revoked polynomial $r(x)$. Therefore U_i computes $r(i)$ $(\neq 0)$
- Hence U_i gets $K = \frac{h(i) - f(i)}{r(i)}$

- But if, $i \in R$, then $r(i) = 0$ and U_i can not get the secret key K.

- Note, that if $f(x)$ is known to a group of users then they can together compute the secret key K by computing $\frac{h(x) - f(x)}{r(x)}$ for some x such that $r(x) \neq 0$.

- Hence a group of t or more users can compute the secret key K, via the Shamir Secret Sharing technique.

Identity-Based Encryption (IBE):

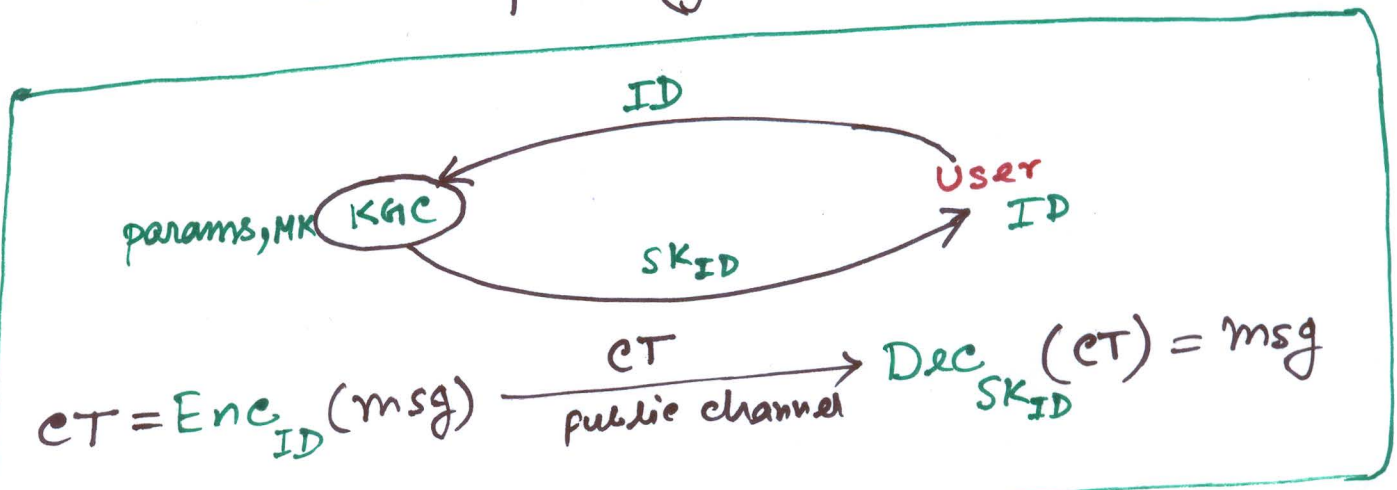
- Public key ID $\in \{0,1\}^*$
- An ~~IB~~ ID-based encryption scheme has four algorithms.

1. Setup: Creates system parameters and master key.

2. Extract: Uses master key to generate the private key corresponding to an arbitrary public key string ID.

3. Encrypt: Encrypts messages using the public key ID.

4. Decrypt: Decrypts the message using the corresponding private key of ID.



Bilinear Map :

- Let p be a prime number and G_1, G_T be multiplicative cyclic groups of order p. A map $e: G_1 \times G_1 \to G_T$ satisfying the following properties is called a bilinear map or bilinear pairing.
  - $e(u^a, v^b) = e(u, v)^{ab}$, $\forall\, u, v \in G_1$, $a, b \in Z_p$
  - if g is a generator of G_1, then $e(g, g)$ is a generator of G_T,
  - $e(u, v)$ is efficiently computable for all $u, v \in G_1$.
- We denote this bilinear map by (p, G_1, g, G_T, e), where g is a generator of G_1.
- Example: Tate or Weil pairing from elliptic curves.

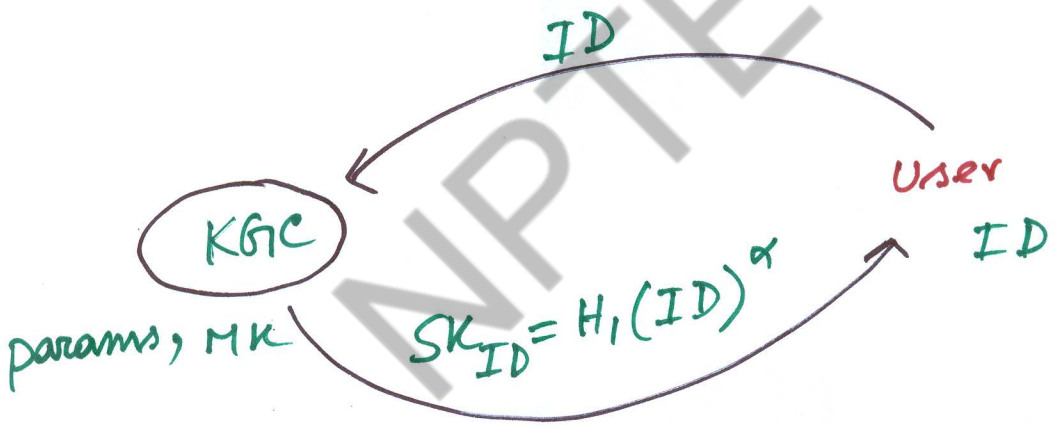
IBE (Boneh and Franklin)

- (p, G_1, g, G_T, e)
- $params = [p, g, Q, H_1, H_2]$

where $Q = g^\alpha$, $H_1 : \{0,1\}^* \rightarrow G_1$,

$H_2 : G_T \rightarrow \{0,1\}^\ell$.

- $MK = \alpha$



where $ID \in \{0,1\}^*$

- Encrypt($params = [p, g, Q = g^\alpha, H_1, H_2]$, $ID \in \{0,1\}^*$, $msg \in \{0,1\}^\ell$)

$= CT = [C_1, C_2]$

where $C_1 = g^r$, $C_2 = msg \oplus H_2(e(H_1(ID), Q)^r)$,

$r \xleftarrow{R} Z_p$

- Decrypt (params, $SK_{ID} = H_1(ID)^{\alpha}$, CT)

$$= msg = C_2 \oplus H_2(e(SK_{ID}, C_1))$$

- Correctness →

$$C_2 \oplus H_2(e(SK_{ID}, C_1))$$

$$= msg \oplus H_2(e(H_1(ID), g)^{r}) \oplus H_2(e(H_1(ID)^{\alpha}, g^{r})).$$

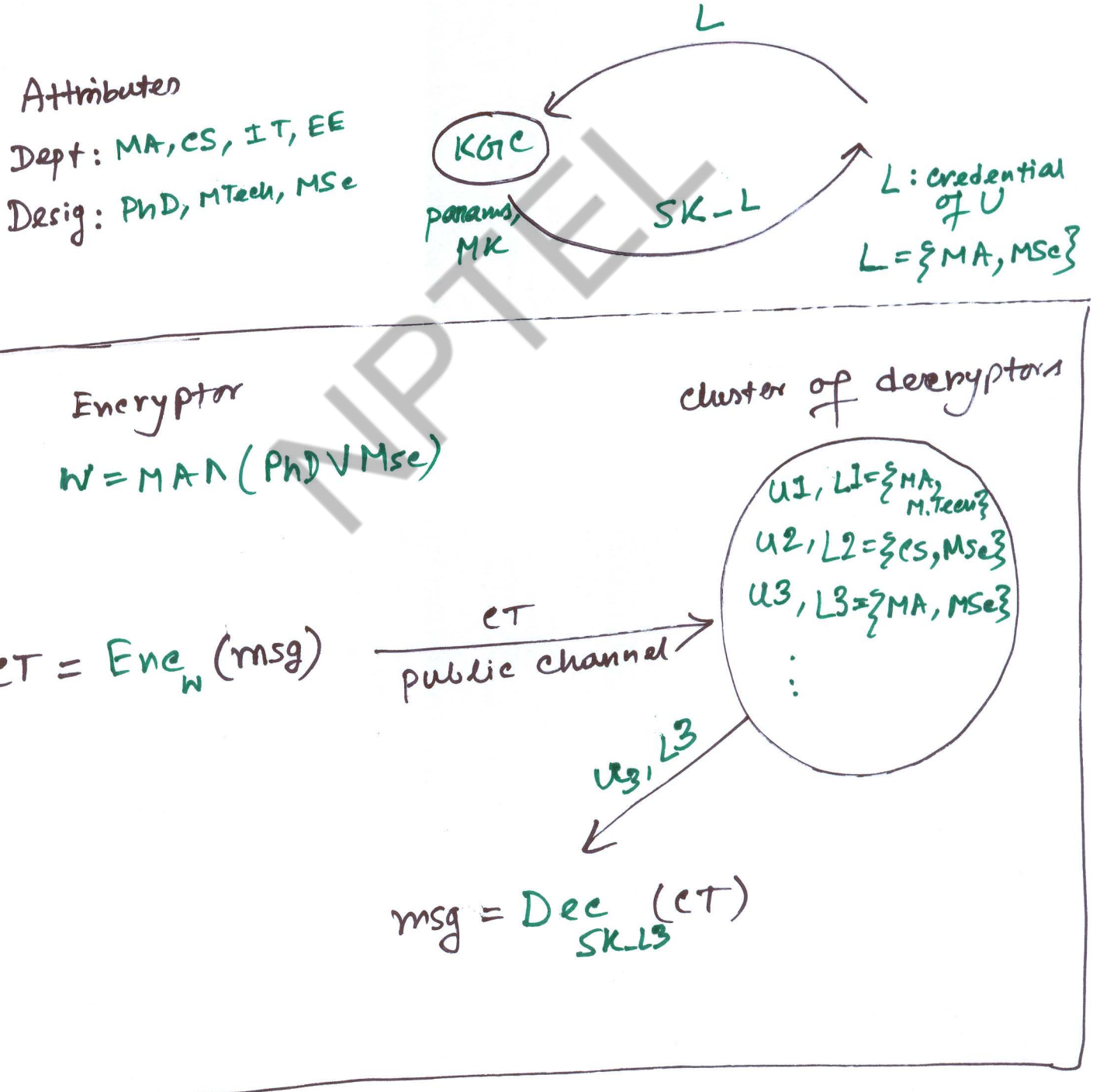
$$= msg \oplus H_2(e(H_1(ID), g^{\alpha})^{r}) \oplus H_2(e(H_1(ID)^{\alpha}, g^{r}))$$

$$= msg \oplus H_2(e(H_1(ID), g)^{\alpha r}) \oplus H_2(e(H_1(ID), g)^{\alpha r})$$

$$= msg.$$

• Attribute Based Encryption (ABE) :

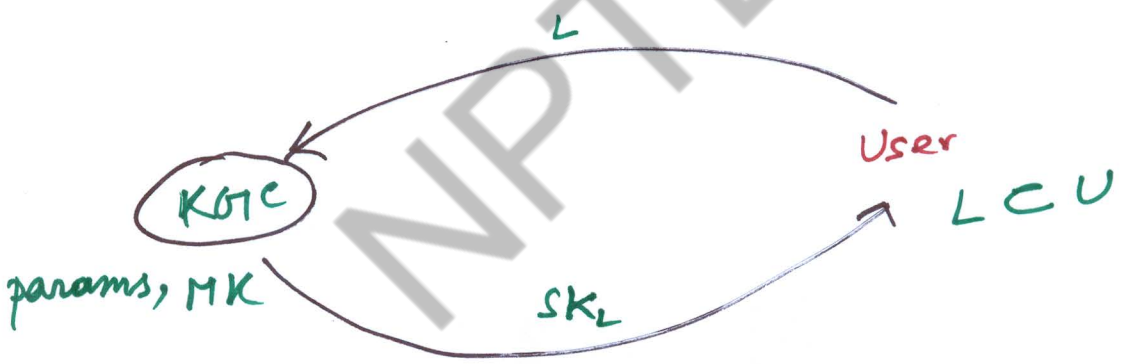
- 1-to-many
- fine grained access control.



ABE (Sahai and Waters) :

- (p, G_1, g, G_T, e)
- $U = \{att_1, att_2, \ldots, att_n\} = \{1, 2, \ldots, n\}$.
- $params = [p, g, e, g_2, h, Y = e(g, g_2)^{\alpha}, T_1, T_2, \ldots, T_n]$,

  where $\alpha \xleftarrow{R} Z_p$, $g_2, h, T_i \xleftarrow{R} G_1$.
- $MK = \alpha$



$$SK_L = [L, d_1 = g^r, d_2 = g_2^{\alpha} h^r, d_i = T_i^r \ \forall i \in L]$$

where $r \xleftarrow{R} Z_p$.

- Encrypt ($params[p, g, e, g_2, h, Y = e(g, g_2)^{\alpha}, T_1, T_2, \ldots, T_n]$, $W = i_1 \wedge i_2 \wedge \ldots \wedge i_k$, $msg \in G_T$)

  $= CT = [W, C_1, C_2, C_3]$,

where $C_1 = msg \cdot Y^s$, $C_2 = g^s$, $C_3 = \left(h \prod_{j=1}^{k} T_{i_j}\right)^s$, $s \xleftarrow{R} Z_p$

- Decrypt (param, $SK_L = [L, d_1, d_2, d_i, \forall i \in L]$, $CT = [W, C_1, C_2, C_3]$)

  if $\{i_1, \ldots, i_k\} \notin L$, decryption fails

  if $\{i_1, \ldots, i_k\} \subset L$, then $d = d_2 \cdot \prod_{j=1}^{k} d_{i_j}$

  and $$msg = \frac{C_1 \cdot e(d_1, C_3)}{e(d, C_2)}$$

- Correctness

  $SK_L = [L, d_1 = g^r, d_2 = g_2^{\alpha} h^r, d_i = T_i^r, \forall i \in L]$

  $CT = [W, C_1 = msg \cdot Y^s, C_2 = g^s, C_3 = (h \prod_{j=1}^{k} T_{i_j})^s]$,

  where $Y = e(g, g_2)^{\alpha}$ and $d = d_2 \cdot \prod_{j=1}^{k} d_{i_j}$

$$\frac{C_1 \cdot e(d_1, C_3)}{e(d, C_2)} = \frac{msg \cdot e(g, g_2)^{\alpha s} \cdot e(g^r, (h \prod_{j=1}^{k} T_{i_j})^s)}{e(g_2^{\alpha} h^r \prod_{j=1}^{k} T_{i_j}^r, g^s)}$$

$$= \frac{msg \cdot e(g, g_2)^{\alpha s} \cdot e(g, h \prod_{j=1}^{k} T_{i_j})^{rs}}{e(g, g_2)^{\alpha s} \cdot e(g, h \prod_{j=1}^{k} T_{i_j})^{rs}}$$

$$= msg$$