Block Cipher, Hash Function, MAC, AEAD – Srimanta Bhattacharya

- Security of Pseudo Random Permutations –

$Perm_n : \{f : \{0,1\}^n \longrightarrow \{0,1\}^n, f \text{ is one-one}\}$

↑ Set of all possible permutations

Set $K \subseteq Perm_n$. K is the pseudo random permutation, which is a subset of all the permutations. K is also the set of keys which identify the pseudo random permutations.

We now find PRP-Advantage of K.

PRP-Advantage of $K = |Pr\{A^f = 1 : f \xleftarrow{\$} K\} - Pr\{A^f = 1 : f \xleftarrow{\$} Perm_n\}|$

A^f means that adversary is interacting with f. These adversaries are called oracle adversaries.

In the PRP-Advantage, the game is defined in the sense that first adversary chooses from the pseudo random permutation and then it selects from the random permutation.

The oracle adversaries just queries and does not look into After some time it outputs something.

In this case the output is either 0 or 1, as it is defined. A^f returns 1 if a certain permutation is selected from the set.

$Perm_n : \{0,1\}^n \longrightarrow \{0,1\}^n$

$K = Perm_n \setminus I$ → Identity Permutation (Whatever the input, we get the same output.)

↑ K is computed by removing the ~~Identity~~ Permutations ~~from the set of~~ all permutations

Number of Permutation in $Perm_n$ is $2^n!$

$I(00) \rightarrow 00$

$I(01) \rightarrow 01$

$I(10) \rightarrow 10$

$I(11) \rightarrow 11$

We now see that if we choose a string of only 0's, the probabability that it maps to all zeros is chosen from $Perm_n$ is $1/2^n$ and if chosen from K is

$\frac{(2^n-1)!-1}{2^n!-1}$

The cardinality of K is $2^n! - 1$

The cardinality of set which maps all zeroes to

• A Public key encryption and decryption : (Congruential Pu

1. Choose a large number q.
2. Secret f and g with $f < \sqrt{q/2}$ and with the bound $\sqrt{q/4} < g < \sqrt{q/2}$. GCD(f,q)=1 and GCD(f,q)=1.
3. $h = (f^{-1}g) \bmod q$
4. P.K. = (h,q)

PK : (h,q)

1. $0 < m < \sqrt{q/4}$, $0 < r < \sqrt{q/2}$
2. $C \equiv (rh + m) \bmod q$.

Now, $fc \equiv (rhf + fm) \bmod q$

$\equiv (rf^{-1}gf + fm) \bmod q$

$= (rg + fm) \bmod q$

Now, $rg + fm < \sqrt{q/2}\sqrt{q/2} + \sqrt{q/2}\sqrt{q/4}$

$< q$

Now, let $y = rg + fm$

$\therefore y \bmod g = (fm) \bmod g$

Now, $z \equiv y \bmod g = (fm) \bmod g$

or, $zf^{-1} \bmod g = m \bmod g = m$

$\therefore m = zf^{-1} \bmod g$.

This technique can be easily broken if a lattice is used.

• If p is prime p mod 4 gives 1 or 3, $p \neq 2$.

$p = 5 = 2^2 + 1^2$

$p = 13 = 3^2 + 2^2$

$p = 17 = 4^2 + 1^2$

$p = 29 = 5^2 + 2^2$

we can express p as sum of two squares (Not mathematically proven)

If p mod 4 = 1, $p = x^2 + y^2$.

• Lattice —

→ $v_1, v_2, \ldots, v_n \in \mathbb{R}^m$

All v_is are vector spaces.

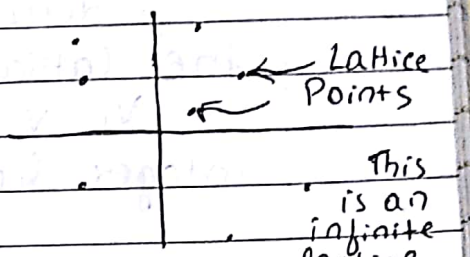
Lattice $L = \{\sum_{i=1}^{n} x_i v_i : x_i \in Z\}$

→ $v_1 = (1, 0, 0, -1) \in \mathbb{R}^4$

$v_2 = (0, 1, 2, 3)$

$v_3 = (0, 2, \ldots, 5, 1)$

$v_4 = (\pi, e, 1, 2)$



→ Now, let

$v_1 = (1, 0) \in \mathbb{R}^2$

$v_2 = (0, 1) \in \mathbb{R}^2$

$x_1 v_1 + x_2 v_2 = x_1(1, 0) + x_2(0, 1)$

$= (x_1, x_2)$

In this example, the lattice creates an xy-plane and there are certain lattice points which have to be integer points on the 'xy-plane. Here an infinite plane is created.

→ A lattice consists of linearly independent vectors. The integer combination of these vectors is called the lattice.

→ $u_1 = (2017, 1)$ $v_1 = (1, 0)$

$u_2 = (2018, 1)$ $v_2 = (0, 1)$

$L_1 = \{x_1 v_1 + x_2 v_2, x_1, x_2 \in Z\}$

$L_2 = \{x_1 u_1 + x_2 u_2, x_1, x_2 \in Z\}$

$u_1, u_2 \in L_1$ and $L_2 \subseteq L_1$

$u_1 = (2017, 1) = 2017(1, 0) + 1 \cdot (0, 1)$

$\in L_1$

$\therefore L_2 \subseteq L_1$

$u_2, u_1 \in L_2$

or, $u_2 - u_1 = (1, 0) \in L_2$

or, $2017(1, 0) = 2017$

or $2017 u_2 - 2017 u_1 \in L_2$

or $(2017, 0) \in L_2 \Rightarrow (0, 1) \in L_2$

$\therefore (2017, 1) \in L_2$

$\therefore L_1 \subseteq L_2$

Hence, we can represent any vector as an integer linear combination of L_1 and L_2.

We now consider the bassis vectors

$$B_1 = \{v_1, v_2\}$$

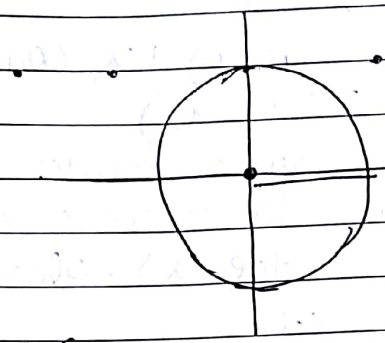
$$B_2 = \{u_1, u_2\}$$

Now, $\det(L) = |[v_1\ v_2 \ldots v_n]|$

$v_1, v_2, \ldots, v_n$ are are column vectors

If $\det(L)$ is large number of points in the lattice is less.

$v_1, v_2, \ldots, v_n$ are the vector spaces whose integer linear combination creates the lattice.



$$L = \{x(1,0) + y(0,1);\ x, y \in Z\}$$

$(1,0)$: Smallest non-zero vector in the lattice

$(0,1)$

Z: only integers

$$|(x,y)| = \sqrt{x^2+y^2}$$ ← Distance between two points x and y.

Finding the shortest vector in the lattice is a hard problem., the vector being non-zero

From any two given vectors, we can find the shortest vector of the lattice.

This is ——— exploited to create ... cryptosystems which are not only secure in the classical domain but also in the classical domain.

- $(p-1)! \equiv -1 \bmod p$, p being a prime
  [Check proof]

If $p \equiv 1 \bmod 4$, from the above result we can show $x^2 \equiv -1 \bmod p$

Stream Cipher — Santanu Sarkar

Lattices (contd...) —

$v_1, v_2, \ldots, v_n \in \mathbb{R}^n$

All $v_1, v_2, \ldots, v_n$ are vector spaces

$v_1 = (v_{11}, v_{12}, \ldots, v_{1n}) \in \mathbb{R}^n$

$\vdots$

$v_n = (v_{n1}, v_{n2}, \ldots, v_{nn}) \in \mathbb{R}^n$

Now, if

$$\left| \begin{bmatrix} v_{11} & \cdots & v_{n1} \\ \vdots & \cdots & \vdots \\ v_{1n} & \cdots & v_{nn} \end{bmatrix} \right| \neq 0$$

implies that $v_1, v_2, \ldots, v_n$ are linearly independent.

For $n=2$, $v_1 = (1,2)$ and $v_2 (2,3)$ (an example)

$$\therefore \begin{vmatrix} 1 & 2 \\ 2 & 3 \end{vmatrix} \neq 0$$

$\therefore v_1$ and v_2 are linearly independent.

We create a lattice L, such that $L = \{x_1 v_1 + x_2 v_2 + \ldots + x_n v_n, x_1, x_2, \ldots, x_n \in Z\}$, where say the vectors $v_1, v_2, \ldots, v_n \in \mathbb{R}^n$ creates the lattice and $x_1, x_2, \ldots, x_n$ are the lattice points.

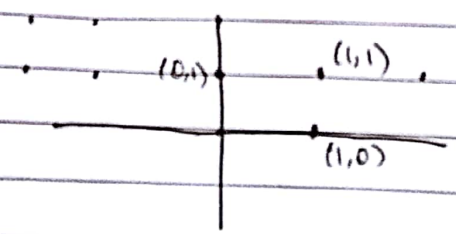
Also, for $n=2$, $v_1 = (1,2)$ and $v_2 = (2,4)$ (an example)

$$\therefore \begin{vmatrix} 1 & 2 \\ 2 & 4 \end{vmatrix} = 0$$

$\therefore v_1$ and v_2 are linearly dependent.

For $n=2$, let $v_1 = (1,0)$ and $v_2 = (0,1)$

$L = \{x(1,0) + y(0,1)\}$



← Lattice points forming an infinite set

To see if (a,b) is a lattice point, we solve the equation $(a,b) = x(2017,1) + y(2018,1)$ for x and y, which basically boils down to $a = 2017x + 2018y$ and $b = x + y$. If we see that x and y are integer values, then (a,b) is a lattice point.

In lattices, generating set is not unique.

Euclidean distance for (a,b) is $|(a,b)| = \sqrt{a^2+b^2}$ from origin.

If the distance of a vector from origin is small, it is said to be a good basis.

$v_1 = (1,0)$, $v_2 = (0,1)$ → Good Basis

$v_1 = (2017,1)$, $v_2 = (2018,1)$ → Bad Basis

Given $v_1 = (2017,1)$ and $v_2 = (2018,1)$ it is difficult to find the smallest possible non-zero vector for this basis.

NTRU is a cryptosystem based on lattice which is faster than the RSA cryptosystem.

- Minkowski's Convex Body Theorem —

Let Λ be a lattice of rank n, then for any centrally-symmetric convex set S, if $Vol(S) > 2^n \det(\Lambda)$, then S contains a non-zero lattice points.

For $n = 2$, Volume = Area

Centrally Symmetric: If $x \in S \Rightarrow -x \in S, \forall x$

Convex Set: $x, y \in S \Rightarrow \lambda x + (1-\lambda)y \in S$ where $\lambda \in [0,1]$

If we draw a line segment from x to y, the line segment should lie within the set, given x and y are part of the set.

- If $p \equiv 1 \bmod 4$, where p is a prime, p can be written as $p = x^2 + y^2$.

We know, $(p-1)! \equiv -1 \bmod p$.

From here, we can show $m^2 \equiv -1 \bmod p$

Definition for modulo:

$a \equiv b \bmod c \Rightarrow c \mid a-b$

(≡: Congruent operator; |: divides)

If $p \equiv 1 \bmod 4$, $\exists$ one m, such that $m^2 \equiv -1 \bmod p$

We consider the following generating set for a lattice:

$$v_1 = (1, m)$$
$$v_2 = (0, p)$$

Hence, the lattice $\Delta = \{x(1,m) + y(0,p) ; x, y \in \mathbb{Z}\}$

Now, $\det(\Delta) = \begin{vmatrix} 1 & m \\ 0 & p \end{vmatrix}$

$$= p$$

$\therefore w = x(1,m) + y(0,p) = (x, xm + yp)$,

where w is any lattice point.

Now, length of w $\|w\| = \sqrt{x^2 + (xm+yp)^2}$

$$\therefore \|w\|^2 = x^2 + (xm+yp)^2$$
$$= x^2 + x^2m^2 + 2xymp + y^2p^2$$
$$\equiv [x^2(1+m^2)] \bmod p$$

Now, $p \mid 1+m^2$, $\because m^2 \equiv -1 \bmod p$

$$\therefore \|w\|^2 \equiv 0 \bmod p$$

Hence, we can write $p \mid x^2 + (xm+yp)^2$

Let us consider S is a circle of radius $\sqrt{2p}$

Now, circle is a centrally-symmetric convex set.

Area of the circle $= \pi R^2$

$$= \pi 2p$$

In our case $n = 2$.

$\therefore$ Vol (S) = Area of the circle

$$= 2\pi p > 2^2 p.$$

Hence, there will exist a non-zero lattice point w in the circle according to Minkowski's Theorem.

Hence, for lattice point w,

$$\|w\|^2 < 2p$$

Now let $w = (z_1, z_2)$.

$$\therefore z_1^2 + z_2^2 < 2p$$

We have seen that $\|w\|^2 = x^2 + (xm+yp)^2$

$$\therefore x^2 + (xm+yp)^2 < 2p$$

We can thus write:

$$0 < x^2 + (xm+yp)^2 < 2p$$

From the equations $p \mid x^2 + (xm+yp)^2$ and $0 < x^2 + (xm+yp)^2 < 2p$, we can write

$$p = x^2 + (xm+yp)^2.$$

Hence, we have expressed p as the sum of two squares.

- "No efficient algorithm for factorization of $N = pq$ is known, p and q being primes" — RSA uses this assumption.

First equation: $N = pq \longrightarrow f(x,y) = N - xy$

Roots of the equation: $x = p, y = q$

$x = q, y = p$

$x = 1, y = N$

$x = N, y = 1$

Second equation:

$$ed = 1 + k(N+1-p-q)$$

$$\Rightarrow 1 + k(N+(1-p-q)) \equiv 0 \bmod e$$

$f(x,y) = 1 + x(N+y) \in \mathbb{Z}_e[x,y]$

with root $(k, 1-p-q)$

- Finding roots of a polynomial:

→ Univariate Integer Polynomial -

$f(x) \neq \in \mathbb{Z}[x]$ with root $x_0 \in \mathbb{Z}$

efficient methods available

→ Multivariate Integer Polynomial -

$f(x,y) \in \mathbb{Z}[x,y]$ with root $(x_0, y_0) \in \mathbb{Z} \times \mathbb{Z}$

→ Univariate Modular Polynomial -

$f(x) \in \mathbb{Z}_N[x]$ with root ~~or~~ $x_0 \in \mathbb{Z}_n$

The last two problems do not have an efficien method.

** Hilbert's Tenth Problem, 1900

- Using lattice we transform the last two problems (of the previous section) to the first problem and then solve it.

• LLL Algorithm:

Devised by Lenstra, Lenstra and Lovász (1982).

Main goal - Produce lattice basis 'short (bounded) and nearly orthogonal'.

$\{v_1 = \{1,2\}, v_2 = \{3,4\}\}$

$\Rightarrow \{r_1 = \{1,0\}, r_2 = \{0,2\}\}$

This is an approximation algorithm of the order 2^n.

• Minkowski showed that the lattice contains at least one non-zero vector r, s.t. the distance of r should be

$$\|r\| \leq \sqrt{\omega}\, \det(L)^{1/\omega}$$

, ω being the number of linearly independent vectors [$\omega = n$]

But LLL algorithm shows:

$$\|r\| \leq 2^n \det(L)^{1/n}$$

• Connecting LLL to Root Finding:

→ $a_0 + a_1 x + \ldots + a_\omega x^\omega \longleftrightarrow (a_0, a_1, \ldots, a_\omega)$

→ $f(x) \in \mathbb{Z}_N[x]$ with $f(x_0) \equiv 0 \bmod N$ with $|x_0| < X$

→ Generate $f_1(x) \ldots f_\omega(x)$ with $f_1(x_0) \equiv f_2(x_0) \equiv \ldots \equiv f_\omega(x_0) \equiv 0 \bmod N$

→ Construct L from $f_1(xX), \ldots, f_\omega(xX)$.

→ LLL: $\|r_1\| < (\det(L))^{1/\omega}$

→ $\det(L)^{1/\omega} < N \Rightarrow r_1'(x_0) = 0$ where $r_1'(x) = r_1(x/X)$.

→ This result comes from a theorem: Let

Let $h(x) \in \mathbb{Z}[x]$ be an integer polynomial with ω monomials. Let $h(x_0) \equiv 0 \pmod N$ with $x_0 < X$ and $\|h(xX)\| < N/\sqrt{\omega}$. Then, $h(x_0) = 0$ holds over integers.

Proof:

$$|hx_0| = \left|\sum_i h_i x_0^i\right| \leq \sum_i |h_i x_0^i| \leq \sum_i |h_i| X^i \leq \sqrt{n} \cdot \|h(xX)\| < N$$

Now, since N divides $h(x_0)$ by the first condition, $h(x_0) = 0$

- Problem: Find a root x_0 of $f(x) = x^2 + ax + b \equiv 0 \bmod N$
  We consider two more polynomials $g(x) = xN$ and $h(x) = N$.
  construct lattice from coefficient vectors of $f(xX)$, $g(xX)$ and $h(xX)$.

$$L = \begin{bmatrix} X^2 & aX & b \\ 0 & NX & 0 \\ 0 & 0 & N \end{bmatrix}$$

Use LLL algorithm to reduce this lattice.
$\det(L) = X^3 N^2$
we need $(\det(L))^{1/3} \leq N$
If the root is bounded by $|x_0| < N^{1/3}$, the reduction works.
For fourth order equation, if the root is bounded by $|x_0| < N^{2/5}$, the reduction works. This can be improved up to $|x_0| < N^{1/2}$ for higher order lattice, but it is impossible to reduce a lattice of size larger than 400.

- Howgrave-Graham —
  Suppose an approximation p_0 of p is known
  Let $x_0 = p - p_0$
  Consider $f(x) = p_0 + x$
  Hence $f(x_0) \equiv 0 \bmod p$
  Let $|x_0| \leq X$
  Aim: finding x_0 from $f(x)$ and N.
  we construct lattice from coefficient vectors of N^2, Nf, f^2, xf^2

$$L = \begin{bmatrix} N^2 & 0 & 0 & 0 \\ Np_0 & NX & 0 & 0 \\ p_0^2 & 2p_0X & X^2 & 0 \\ 0 & p_0^2X & 2p_0X^2 & X^3 \end{bmatrix}$$

We need $\det(L) = X^6 N^3 < (p^2)^4$
$\therefore p \approx N^{1/2}$
This implies that $X < N^{1/6}$

$$p_i(x) = N^{v-i} f^i, \quad 0 \leq i \leq v$$
$$p_i(x) = f^v x^{i-v}, \quad v \leq i \leq h$$

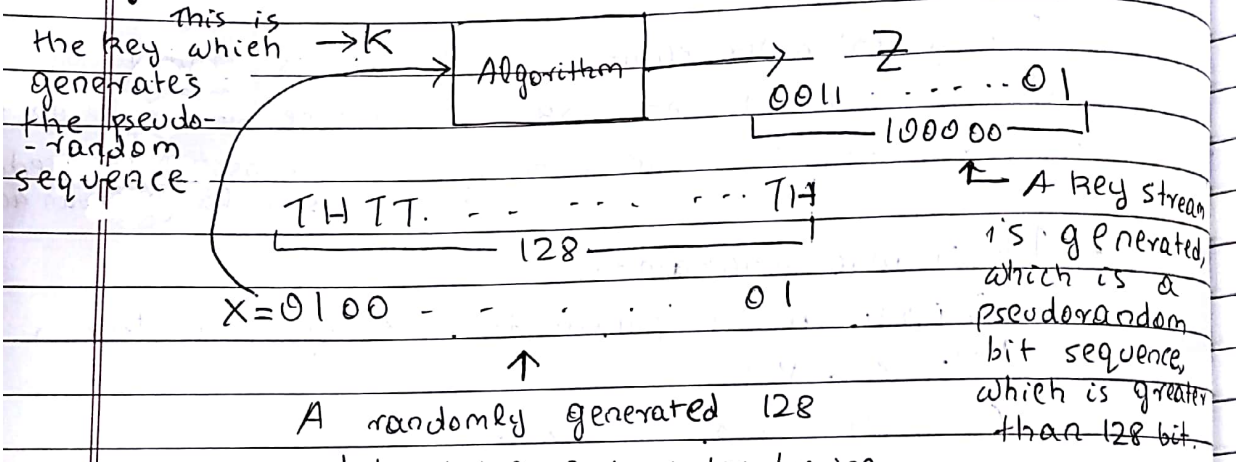
The equations $p_i(x)$ and $p_i'(x)$ are created from N, f and x; with the idea of solving f in polynomial time.

Now, coppersmith showed in 1996 that:

If $|p-p_0| < N^{1/4}$, one can solve f in polynomial time.

Thus, $q_0 = \lfloor \frac{N}{p_0} \rfloor$ is will be an approxiate of p. We derive $p = p_0 + x_0$ and $q = q_0 + y_0$. Hence, the polynomial is $f(x,y) = N - (p_0+x)(q_0+y)$.

Slide: [Stream Cipher]



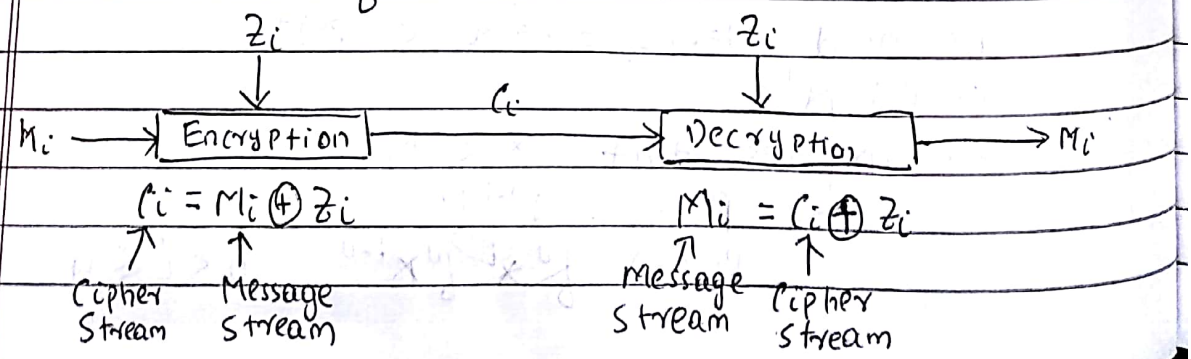
Message M = 0011 --- 01 --- 11

Key stream generated from the algorithm → Z = 0011 ----- 01

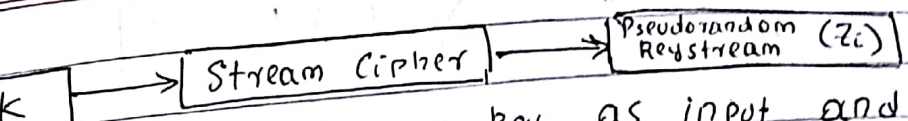
Cipher Text C = M ⊕ Z

To get back Message M, M = C ⊕ Z

The security of the message depends on the randomness of Z



Pseudorandom generator



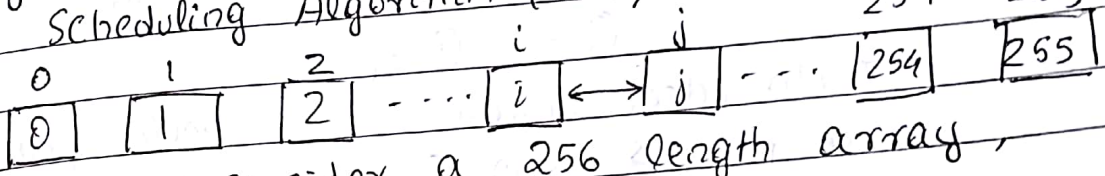
The stream cipher takes a key as input and keeps on generating a random-looking bitstream, the keystream bits.

Computationally, a pseudorandom bitsequence cannot be differentiated from a random sequence

- RC4 —

→ Designed by Ron Rivest in 1987

→ Key Scheduling Algorithm (KSA):



Initially we consider a 256 length array, where $a[i] = i$, i.e. the identity permutation.

Initialize index: $j = 0$;

for $i = 0 \ldots 255$ do

$j = (j + S[i] + K[i]) \bmod 256$

Swap $S[i] \leftrightarrow S[j]$;

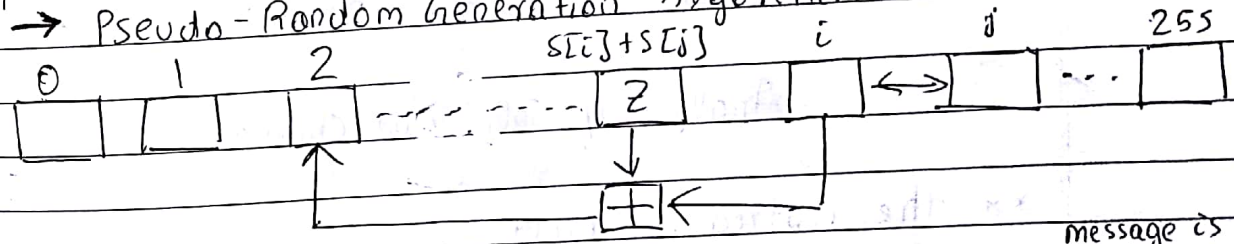
End.

Input: S-array initialized to identity permutation, and key K.

Output: Scrambled S-array.

The scrambled S-array is a pseudo-random permutation.

→ Pseudo-Random Generation Algorithm (PRGA):



Initialize indices: $i = j = 0$;

While TRUE do ← we are taking 8 bits.

$i = i + 1$;

$j = j + S[i]$;

Swap $S[i] \leftrightarrow S[j]$;

message is divide into 8 bit blocks.

Hence, loop terminates at (length of message)/8.

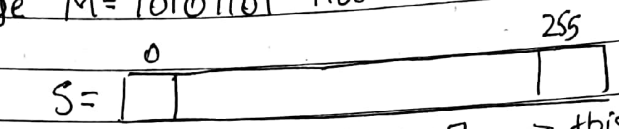
Output $Z = S[S[i] \oplus S[j]]$; Mod 256 domain

End.

Input: Scrambled S-array, obtained as the KSA output.

Output: Pseudo-random stream.

Message M = 10101101 1100 10110101



$Z = S[S[i] \oplus S[j]]$ → this is mod 256

$= \underbrace{10110110}_{8 \text{ bit}}$ ← we get 8 bit binary

$C_i = M_i \oplus Z_i$

→ An ideal stream cipher should generate all the numbers from 0 to 255 with equal probabilities

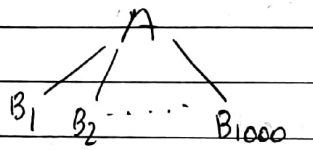
Biases in RC4: $Pr(Z_2 = 0) = 2/256$ (Mantin-2001)

$Pr(Z_r = r) > 1/256$ (Isobe-2013)

$Pr(Z_r = r - K[0]) < 1/256$ (Paterson-20

mod 256 ↑ ↑ First Block of the key

** Broadcast Attack exploits $Pr(Z_2 = 0) = 2/256$



A is sending same message to $B_1, B_2, \ldots, B_{1000}$. If we get the cipher texts and knows that they are from the same message, then the message can be cracked

Analysis of Salsa and Chacha

** The eStream Project.

- Salsa — (Slide: [Analysis of salsa and Chacha])
  - Designed by Bernstein
  - It has 20 rounds
  - The attacks are till 8 rounds.

$$X = \begin{pmatrix} x_0 & x_1 & x_2 & x_3 \\ x_4 & x_5 & x_6 & x_7 \\ x_8 & x_9 & x_{10} & x_{11} \\ x_{12} & x_{13} & x_{14} & x_{15} \end{pmatrix} = \begin{pmatrix} c_0 & k_0 & k_1 & k_2 \\ k_3 & c_1 & v_0 & v_1 \\ t_0 & t_1 & c_2 & k_4 \\ k_5 & k_6 & k_7 & c_3 \end{pmatrix}$$

→ Each word of 32 bits and 16 words

$k_0\ k_1\ k_2\ k_3\ k_4\ k_5\ k_6$ is the key

Known to the Hacker — $c_0\ c_1\ c_2\ c_3$ are constants

$v_0\ v_1$ to t_1 are nonce and counters respectively.

There is a 256-bit key

→ Quarter Round:

The basic nonlinear operation of Salsa 20 is the quarterround function.

Each quarterround (a,b,c,d) consists of four ARX rounds, each of which comprises of one addition (A), one cyclic left rotation (R) and one XOR (X).

$b = b \oplus ((a+d) \lll 7)$

$c = c \oplus ((b+a) \lll 9)$

$d = d \oplus ((c+b) \lll 13)$

$a = a \oplus ((d+c) \lll 18)$

} Each a, b, c and d are 32 bits and we rearrange each of them.

We use quarterround on each column.

$$Q\begin{pmatrix} a \\ b \\ c \\ d \end{pmatrix} = \begin{pmatrix} x \\ y \\ z \\ w \end{pmatrix}$$

Hence, after applying quarter-round to each column we generate a new matrix.

a	b	c	d	
c_0	k_3	t_0	k_5	← For Round 1
c_1	k_1	k_6	k_0	← For Round 2
c_2	k_7	k_1	v_0	← For Round 3
c_3	k_2	v_1	k_0	← For Round 4

} For column round

After four column-rounds we apply quarter-rounds to the four rows.

One columnround → quarter rounds on each column.

one row round — quarterround — on each row
columnround and rowrounds — are done one after the others.
In Salso 20, 10 column rounds and 10 row rounds are used.
20 rounds ≡ 80 quarter rounds.
Instead of row-rounds, we use inverse of a matrix.

→ Number of Rounds:
$X^{(r)}$: r rounds have been applied on initial state X
X^0: Initial state X.
X^R: State of x after R rounds.
A keystream block of 16 words or 512 bits is obtained as $Z = X + X^{(R)}$
Salsa 20; R=20
Salsa 20/12; R=12.
More Rounds: More security
Less Rounds: Higher Speeds.
** Salsa 8 can be aHacked with complexity 2^{243}. There are no such aHacks on Salsa12 or Salsa 20.

• Differential Analysis — (Slide: [Analysis of Salsa and Chacha
AHacker can control v_0, v_1, to and t_1 shown in the previous matrix X.
x_i: the i-th word of the matrix X
x_{ij}: we mean the j-th bit of x_i, the 0th bit being the least significant bit.
We consider two states $X^{(r)}$ and $X^{\prime(r)}$.
$\Delta_i^{(r)} = x_i^{(r)} \oplus x_i^{\prime(r)}$
$\Delta_{i,j}^{(r)} = x_{i,j}^{(r)} \oplus x_{i,j}^{\prime(r)}$, we mean the difference between two states at the j-th bit of i-th word after r many rounds.
$\Delta_{7,31}^{(0)}$ means that we have two initial states

$X^{(0)}$, $X'^{(0)}$ that differ at the 31-st most significant bit of the 7-th word (a nonce word) and they are same at all the other bits of the complete state.

We may obtain a significant bias after a few rounds.

We input differential at the initial state (called the Input Differential) and then try to obtain some bias corresponding to combinations of some output bits (called the Output Differential).

We can compute:

$$Pr\left(\Delta_{P,Q}^{(r)} = 0 \mid \Delta_{ij}^{(0)}\right) = \frac{1}{2}(1+\varepsilon_d)$$

If we are given two states $X^{(r)}$ and $X'^{(r)}$, we represent a differential state matrix, $\Delta^{(r)}$.

$$\Delta^{(r)} = \begin{bmatrix} \Delta_0^{(r)} & \Delta_1^{(r)} & \Delta_2^{(r)} & \Delta_3^{(r)} \\ \Delta_4^{(r)} & \Delta_5^{(r)} & \Delta_6^{(r)} & \Delta_7^{(r)} \\ \Delta_8^{(r)} & \Delta_9^{(r)} & \Delta_{10}^{(r)} & \Delta_{11}^{(r)} \\ \Delta_{12}^{(r)} & \Delta_{13}^{(r)} & \Delta_{14}^{(r)} & \Delta_{15}^{(r)} \end{bmatrix}$$

Here, $\Delta_i^{(r)} = x_i^{(r)} \oplus x_i'^{(r)}$

Lot of cryptanalytic strategies uses the differential analysis approach.

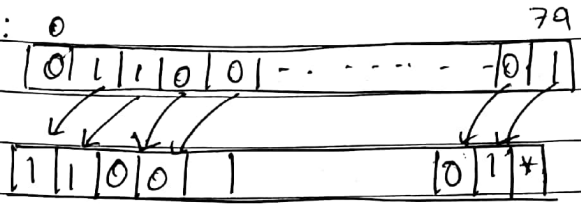
We may put some input differential at the initial state and then try to obtain certain bias in some output differential. Once, we can move forward a few rounds with the above strategy, try to come back a few rounds from a final state after a certain rounds obtaining further non-randomness.

Combining the above ideas, we may find some non-randomness in Salsa20 for forward plus backward many rounds.

Slide: [Stream Ciphers Grain and Trivium]

- Grain —
  - Proposed by Hell, Johansson and Meier (2005)
  - Grain 128 → 128 bit key
  - Grain 128a → 128 bit key with authentication
  - Grain v1 consists of an 80 bit LFSR and an 80 bit NFSR

LFSR: Linear Feedback Shift Register

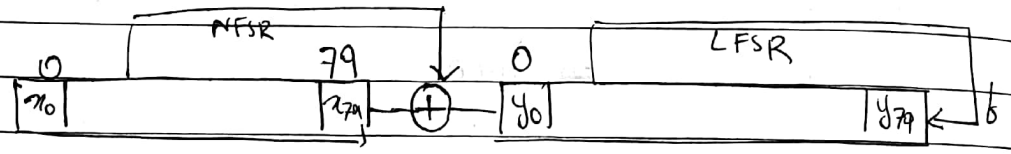


LFSR update function:

$$y_{t+80} = y_{t+62} + y_{t+51} + y_{t+38} + y_{t+23} + y_{t+13} + y_t$$

(y_{t+80}: round or clock)

NFSR: Non-Linear Feedback Shift Register — This uses some kind of product rule rather than only addition. We have product and addition



Output key stream:

$$z_t = \bigoplus_{a \in A} x_{t+a} + h(y_{t+3}, y_{t+25}, y_{t+46}, y_{t+64}, x_{t+63})$$

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

and $h(s_0, s_1, s_2, s_3, s_4) = s_1 + s_4 + s_0 s_3 + s_2 s_3 + s_3 s_4 + s_0 s_1 s_2 + s_0 s_2 s_3 + s_0 s_2 s_4 + s_1 s_2 s_4 + s_2 s_3 s_4$

→ Key Scheduling Algorithm :-

Grain v1 uses a 80-bit key K, and 64-bit initialization vector, IV.

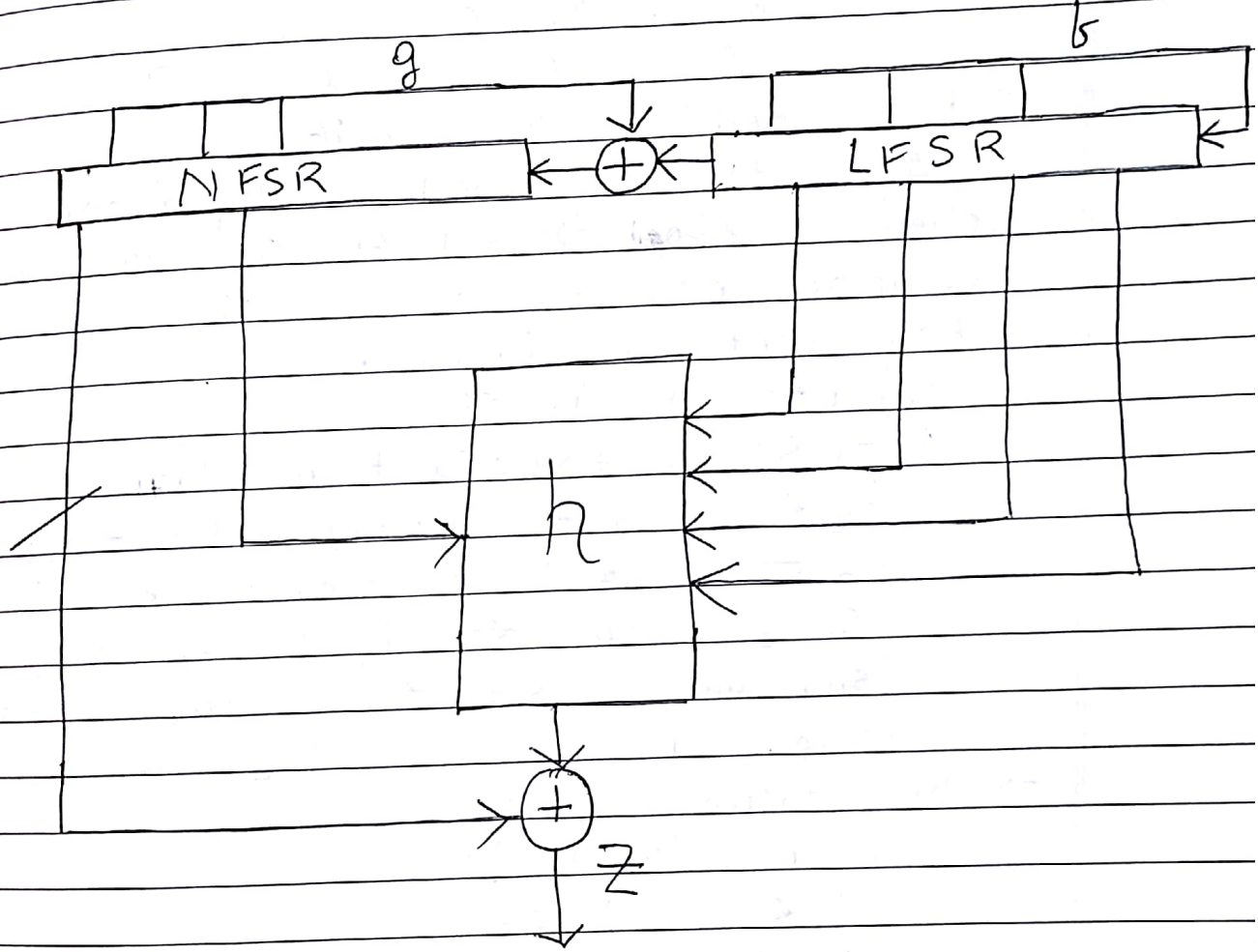
The Key is loaded in NFSR and the IV is loaded in the 0^{th} to the 63^{rd} bits of the LFSR. The remaining 64^{th} to 79^{th} bits of the LFSR are loaded with 1. Then, for the first 160 clocks, the Key-stream bit z_t is XOR-ed to both the LFSR and NFSR update functions.

→ Pseudo-Random Key-stream Generation Algorithm:

After the key scheduling Algorithm, $\therefore z_t$ is no longer XOR-ed to the LFSR and the NFSR

Thus, the LFSR and NFSR are updated as:

$y_{t+n} = f(Y_t)$, $x_{t+n} = y_t + g(X_t)$



This is the structure of Grain V1.

Trivium—

Key size: 80 bits
IV size: 80 bits
Internal state: 288 bits → Internally lots of arrays are required.

→ Key Loading Algorithm :-

The initialization routine works as following

$(s_1, s_2, \ldots, s_{93}) \leftarrow (K_1, \ldots, K_{80}, 0, \ldots, 0)$

$(s_{94}, s_{95}, \ldots, s_{177}) \leftarrow (IV_1, \ldots, IV_{80}, 0, \ldots, 0)$

$(s_{178}, s_{179}, \ldots, s_{288}) \leftarrow (0, \ldots, 0, 1, 1, 1)$

We denote the state after the key loading Algorithm as $S^{(0)}$

$S^{(0)} = [K_1, \ldots, K_{80}, 0, \ldots, 0, IV_1, \ldots, IV_{80}, 0, \ldots 0, 0 \ldots, 0, 1, 1, 1]$.

→ Key Scheduling Algorithm :-

After the key loading Algorithm, the cipher is clocked 4×288 times without producing any key-stream bits

Key Scheduling Algorithm:

for $i \leftarrow 1$ to 4×288 do

$t_1 \leftarrow s_{66} + s_{91} \cdot s_{92} + s_{93} + s_{171}$

$t_2 \leftarrow s_{162} + s_{175} \cdot s_{176} + s_{177} + s_{264}$

$t_3 \leftarrow s_{243} + s_{286} \cdot s_{287} + s_{288} + s_{69}$

$(s_1, s_2, \ldots, s_{93}) \leftarrow (t_3, s_1, \ldots, s_{92})$

$(s_{94}, s_{95}, \ldots, s_{177}) \leftarrow (t_1, s_{94}, \ldots s_{176})$

$(s_{178}, s_{179}, \ldots, s_{288}) \leftarrow (t_2, s_{178}, \ldots, s_{287})$

→ Key stream Generation:

for $i = 1$ to N do

$t_1 \leftarrow s_{66} + s_{93}$

$t_2 \leftarrow s_{162} + s_{177}$

$t_3 \leftarrow s_{243} + s_{288}$

$z_i \leftarrow t_1 + t_2 + t_3$

$t_1 \leftarrow t_1 + s_{91} \cdot s_{92} + s_{171}$

$t_2 \leftarrow t_2 + s_{175} \cdot s_{176} + s_{264}$

$t_3 \leftarrow t_3 + s_{286} \cdot s_{287} + s_{69}$

$(S_1, S_2, \ldots S_{93}) \leftarrow (t_3, S_1, \ldots, S_{92})$
$(S_{94}, S_{95}, \ldots, S_{177}) \leftarrow (t_1, S_{94}, \ldots, S_{176})$
$(S_{178}, S_{179}, \ldots, S_{288}) \leftarrow (t_2, S_{178}, \ldots, S_{287})$
end for

- Fruit :-
Designed by Ghafari, Hu and Chen in 2017. It is a cipher in which the state size is equal to the key size and it uses round key to update the NFSR and also for output keystream.

____ attacks haven suggested so far against it. One is the Birthday attack, which shows that the state twice can be twice the key size. Secondly, the sieving attack, where the ___ attack complexity is $2^{74.95}$ for all rounds.

** Lattices can break the congruential cryptosystem :

$h = f^{-1} g \bmod q$
$fh = g \bmod q$
$fh = g + kq$
$fh - kq = g$
$\therefore (g, f)$ or $(f, g) \rightarrow$ have to be shown as Lattice Points

Lattice : $v_1 = (q, 0)$ $v_2 = (h, 1)$
$\therefore kv_1 + fv_2 = (-kq, 0) + (fh, f)$
$= (g, f)$

Shortest Vector : $\sqrt{2}(\det L)^{1/2}$
$\approx \sqrt{q}$

Here, $|(g, f)| \approx \sqrt{q}$

** Subset Sum Problem :
$A = \{M_1, M_2, \ldots, M_n\}$
$= \{10, 12, 14, 17\}$
$S = 22 = 10 + 12 + 0.14 + 0.17$
$\rightarrow (1, 1, 0, 0)$

This is the knapsack Problem
↳ Shortest vector of the lines
Cryptosystems were built on this principle, but were broken as the problem + public key is not hard.