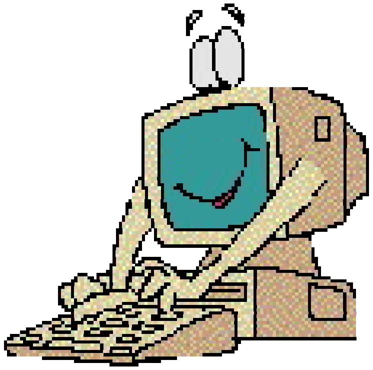


First Order Correlation Attack on a Geffe Generator

Sarad A Venugopalan

Email: sven250+1 @ aucklanduni.ac.nz

Ciphers

What is it?

An algorithm to encrypt and decrypt information.

Why do we need it?

To keep a secret to yourself

To share a secret with intended recipient(s).

Who uses it?

Anyone who keeps a secret

Everyday authentication(Email, online banking, SSL, SSH logins)

Confidential Military communication

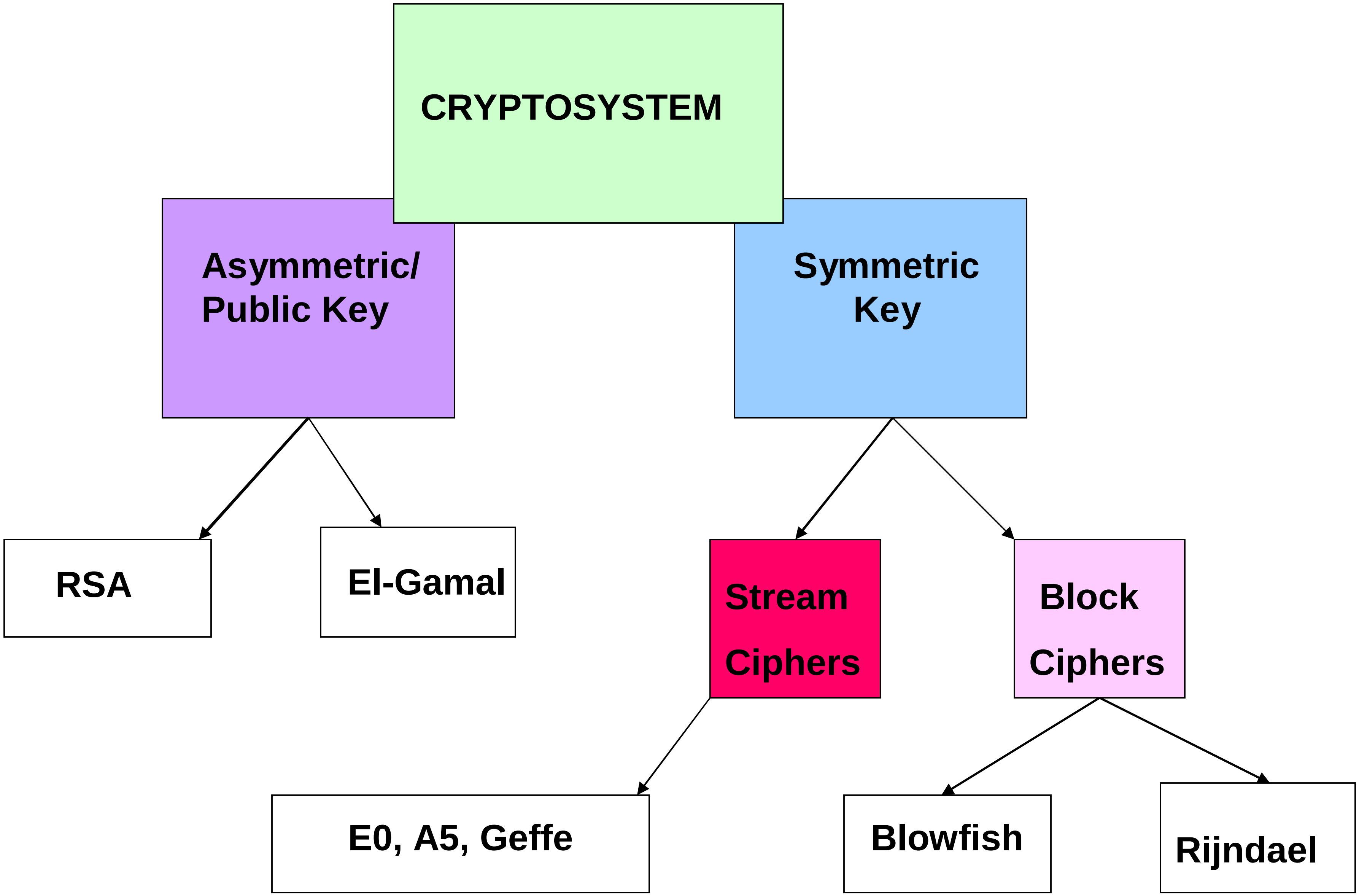
Who wants to decipher your secret?

Anyone wanting to use or sell VALUABLE information. For example

- Corporate secrets

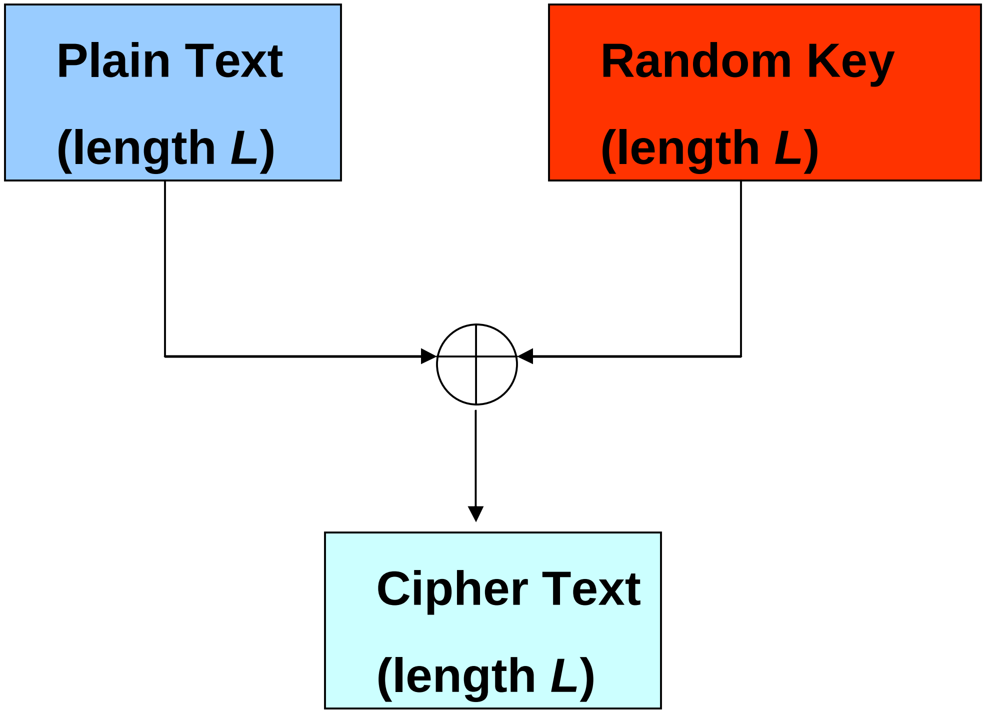
- Military secrets

- Your secrets


CRYPTOSYSTEM
Asymmetric/ Public Key
Symmetric Key
RSA
El-Gamal
Stream Ciphers
Block Ciphers
E0, A5, Geffe
Blowfish
Rijndael

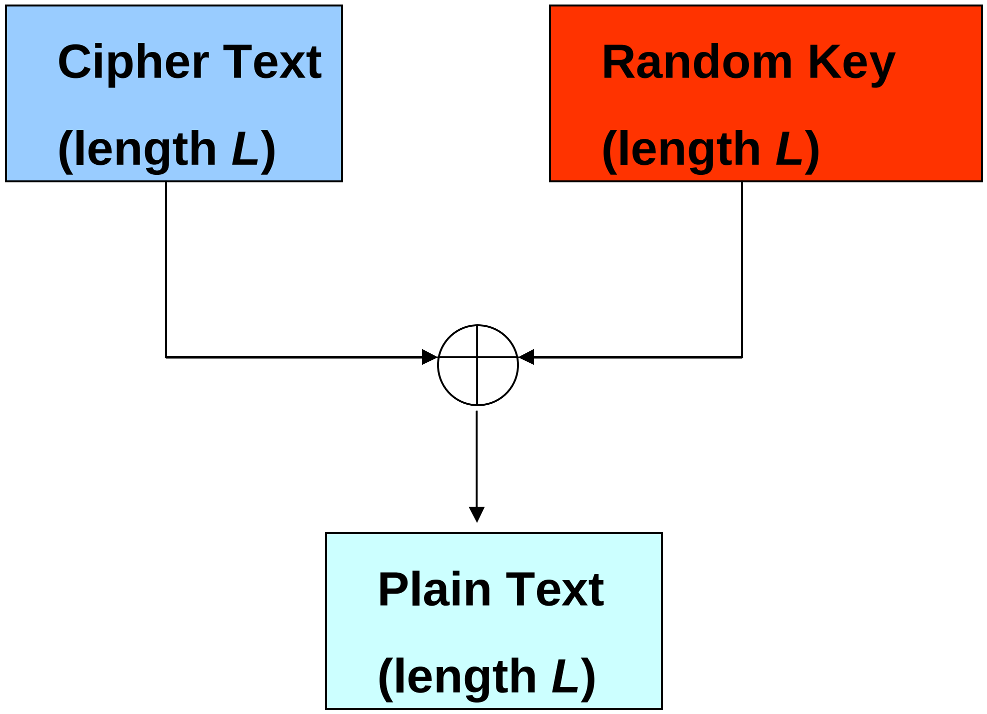
One Time Pad (OTP)

Encryption



One Time Pad

Decryption



One Time Pad

Proven to be unbreakable by Shannon if the keys are random and non-repeating

Key distribution and management are big dampeners to its use

Emphasis is to 'fix' management and the result is STREAM CIPHERS

Properties of Exclusive OR(XOR)

XOR is a linear operator

XOR is bitwise addition modulo 2. (a + b) modulo 2

XOR is symmetric. If a XOR b =c, then a XOR C =b

50% of the output bits are 0's and 50 % of O/P bits are 1's

A	B	A XOR B
0	0	0
0	1	1
1	0	1
1	1	0

What values of A and B gave O/P bit 0? We can guess with a 50% probability

What values of A and B gave O/P bit 1? We can guess with a 50% probability

Properties of bitwise AND

A	B	A AND B
0	0	0
0	1	0
1	0	0
1	1	1

What values of A and B gave O/P bit 0? We can guess with a 33.33% probability!

What values of A and B gave O/P bit 1? We can guess with a 100% probability!

We are at an advantage to correctly guess parts of the input given the output

Properties of bitwise OR

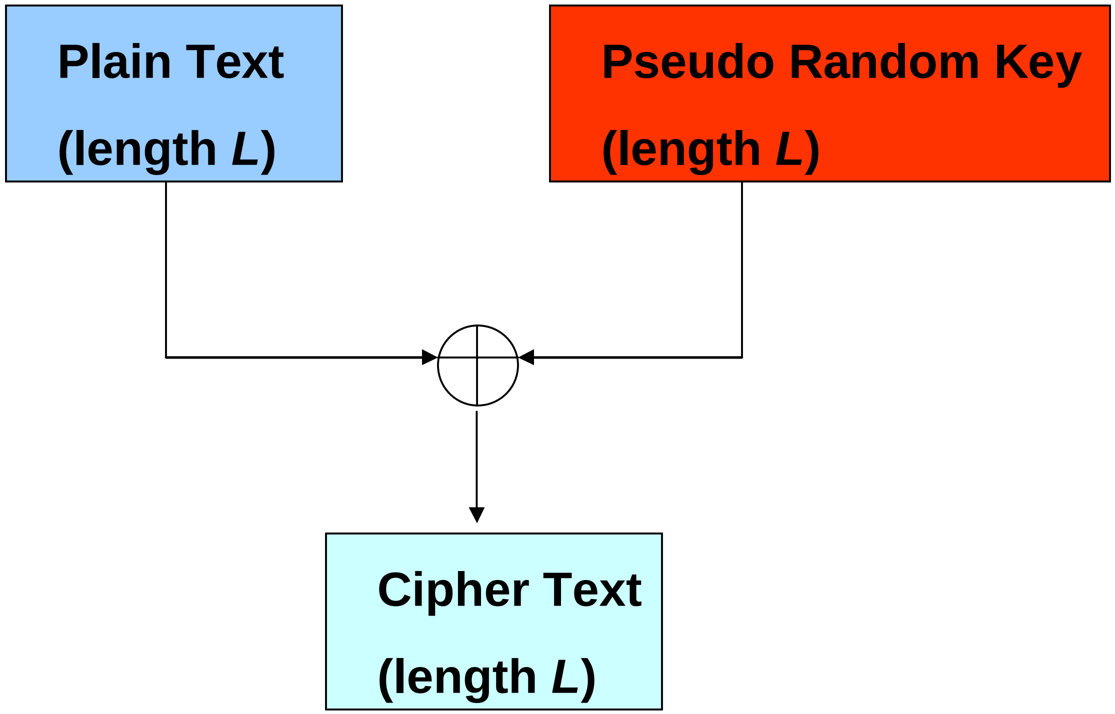
A	B	A OR B
0	0	0
0	1	1
1	0	1
1	1	1

What values of A and B gave O/P bit 0? We can guess with a 100% probability!

What values of A and B gave O/P bit 1? We can guess with a 33.33% probability!

Synchronous Stream Cipher

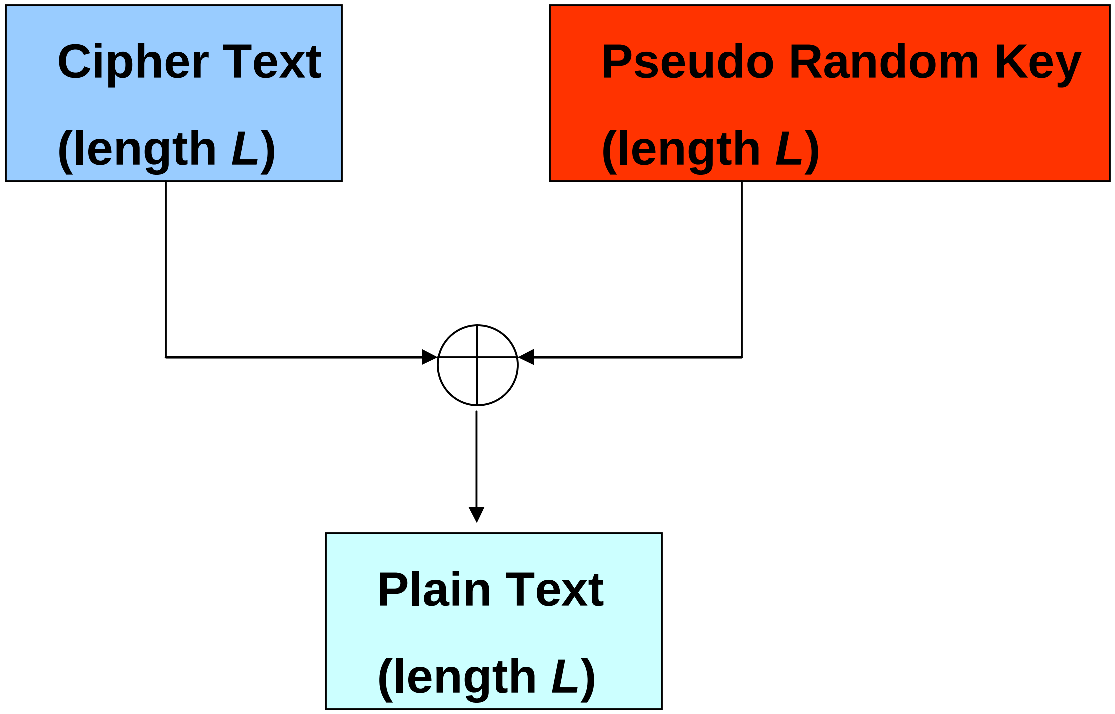
Encryption



Stream ciphers are usually much faster than block ciphers rendering it attractive

Synchronous Stream Cipher

Decryption

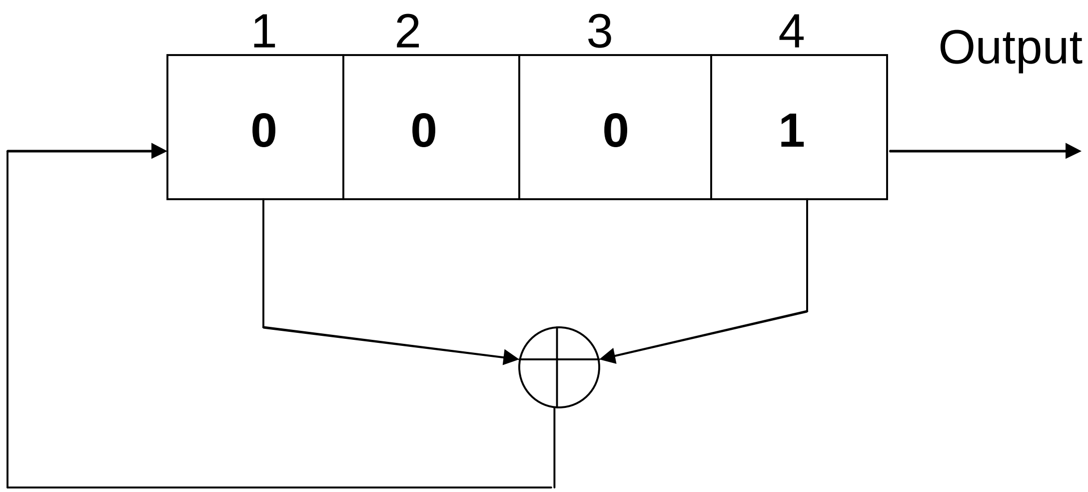


Pseudo Random Key

The generated key exhibit statistical randomness but is computed by a deterministic process.

A Linear Feedback Shift Register (LFSR) is a common building block in generating a Pseudo random Key.

Linear Feedback Shift Register



Primitive Polynomial: $x^4 + x + 1$

Choosing a Primitive Polynomial gives the LFSR a maximum period or $2^{poly\_degree} - 1$

0001=> 1
1000=>8
1100=>12
1110=>14
1111=>15
0111=>7
1011=>11
0101=>5
1010=>10
1101=>13
0110=>6
0011=>3
1001=>9
0100=>4
0010=>2

Geffe Generator

A Synchronous stream cipher with 3 LFSR's

A non-linear Boolean function F combines the three registers to provide the generator output

The symmetric key is the secret initial loading of each of the 3 LFSR's

I.e. 3.(32) = 96 bit key.

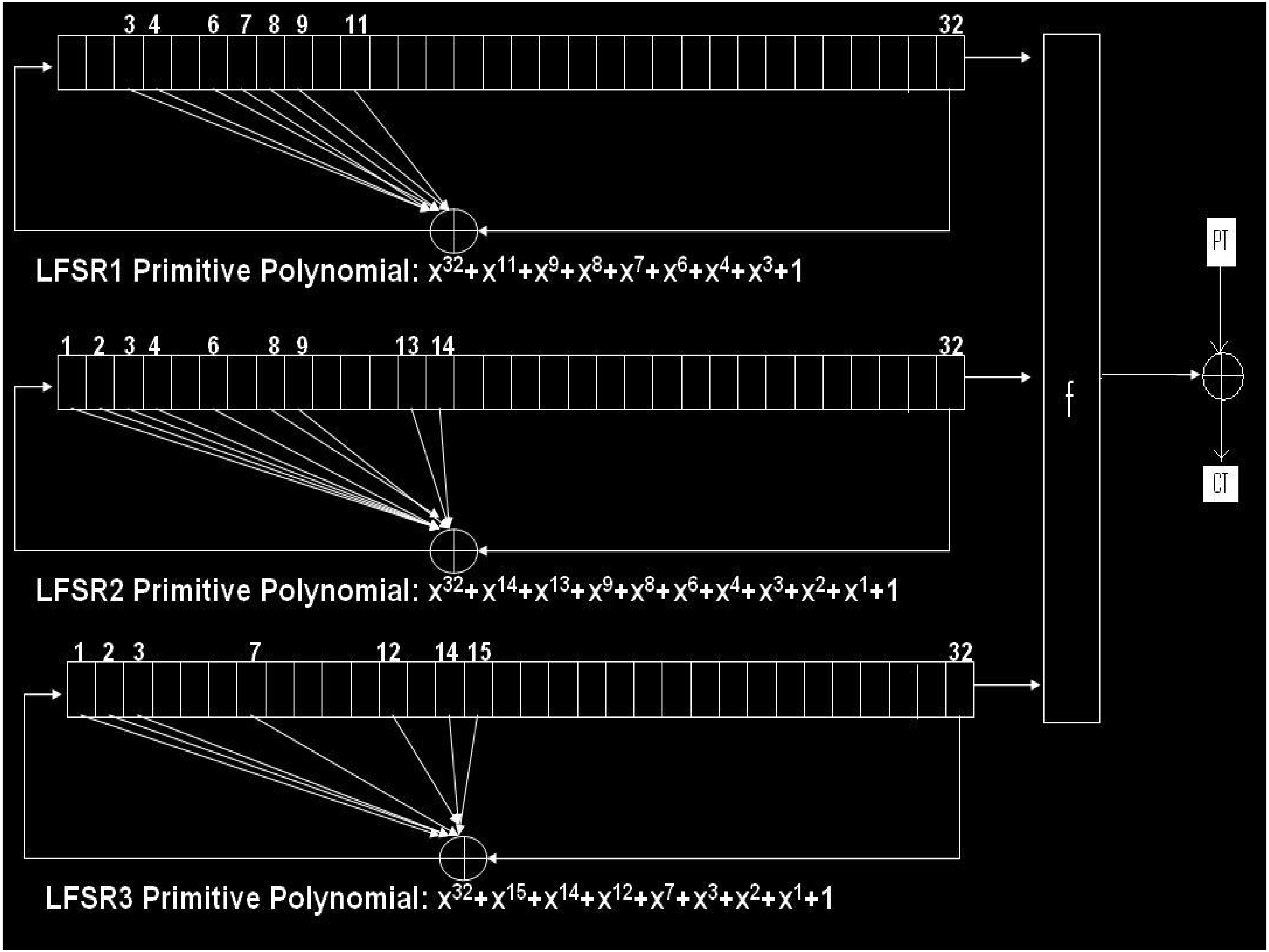
Boolean Function F

$F(x_1,x_2,x_3) = (x_1 \text{ AND } x_2) \text{ XOR } (\underline{\text{NOT } x_1} \text{ AND } x_3)$

x_1 = LFSR 1 O/P bit

x_2 = LFSR 2 O/P bit

x_3 = LFSR 3 O/P bit


3 4 6 7 8 9 11 32
LFSR1 Primitive Polynomial: x32+x11+x9+x8+x7+x6+x4+x3+1
1 2 3 4 6 8 9 13 14 32
LFSR2 Primitive Polynomial: x32+x14+x13+x9+x8+x6+x4+x3+x2+x1+1
1 2 3 7 12 14 15 32
LFSR3 Primitive Polynomial: x32+x15+x14+x12+x7+x3+x2+x1+1
f
PT
CT

Truth Table for non-linear function F

x1	x2	x3	F(x1,x2,x3)
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

6 of 8 bits(75%) of x3 (from LFSR 3) matches with F, the O/P of the Geffe Generator

First Order Correlation Attack

Step 1: Find Known PlainText

Known Plaintext attack

We assume that we know a few blocks of known plaintext and their corresponding ciphertext.

This is a reasonable assumption since WebPages may start with a <html> header or Network Protocols have a standard header.

Step 2: Recover available parts of the KeyStream F

With known plaintext $p_1, p_2, \ldots p_n$ and ciphertext $c_1, c_2, \ldots, c_n$, recover keystream

$$\mathbf{F(x_{1i}, x_{2i}, x_{3i}) = c_i \text{ XOR } p_i}$$

Step 3: Bruteforce LFSR 3

We know that when we 'hit' the right key for LFSR 3, 75% of its bits will match with the keystream F.

For all the incorrect keys of LFSR 3 brute-forced, we except only half(50%) of its bits to match with keystream F.

There are still a few false positive keys that would match 75% of the bits of keystream F. To eliminate them, use more keystream bits F if available.

Step 4: Bruteforce LFSR 2

From the Truth table for function F, we note that 6 of 8(75%) bits of LFSR 2 match with the keystream F.

By a similar argument from Step 3, we brute-force LFSR 2 to get the correct LFSR 2 key.

Step 5: Bruteforce LFSR 1

For entire LFSR 1 keyspace 1 to 2^{32}-1 and recovered LFSR 2 and LFSR 3 keys, compute

BEGIN FOR

BrutStrm=(LFSR1_32bit AND LFSR2_32bit) XOR (<u>NOT LFSR1_32bit</u> AND LFSR3_32bit)

If(keystream_recovered_from_known_plaintext == BrutStrm)

    Print(Probable LFSR 1 key = LFSR1_iteration_index)

END FOR

As a rule of the thumb, the greater the known plaintext available, fewer the false positive on the LFSR_1 key.

Time Complexity of the attack

The time complexity of the correlation attack on the Geffe Generator is reduced to that of bruteforcing 3 individual 32-bit LFSR's from a whopping $O(2^{96})$.

The time complexity of the attack is thus $O(2^{32})$.

Demonstration

Thank you!