OVERVIEW

- Introduction to Stream Ciphers
- Linear Feedback Shift Registers (LFSR)
- Practical Stream Ciphers
- Cryptanalysis
- Case Study: A5/1 Algorithm and its Cryptanalysis

STREAM CIPHER

Stream Cipher encrypts individual characters
(usually *bits*)
of a plaintext message one at a time,
using an encryption which *varies with time*,
into a ciphertext code

- General approach: operate on the plaintext with a *secret keystream* to generate ciphertex

STREAM CIPHER …

- Standard version of a stream cipher works on bits (Z_2)
- Plaintext, keystream and ciphertext are all binary
- Example:

$$\begin{array}{lcl} P & = & 1\,0\,0\,1\,1\,0\,0\,1 \\ K & = & 0\,1\,0\,1\,0\,0\,1\,1 \\ \hline C = P \oplus K & = & 1\,1\,0\,0\,1\,0\,1\,0 \end{array}$$

XOR ($\oplus$) is standard on plaintext and keystream when using bits

- *Key generation function* generates keystream from primary key K_0
- Variations of stream cipher depend on *how we generate keystreams*

VERNAM CIPHER

- *Vernam Cipher* requires *True Random Number Generator* (TRNG)
- Keystream is a *true* random sequence of length equal to that of plaintext
- It is also called a *One Time Pad*
- Vernam Cipher is unconditionally secure

- **Big Problem:** Vernam Cipher is *impractical*
- Electronic software *does not* have TRNG
- Sender and Receiver should get the same random bits – *how?*
- Random sequence, that is the key, is as long as the plaintext!
- Keystream *cannot* be reused

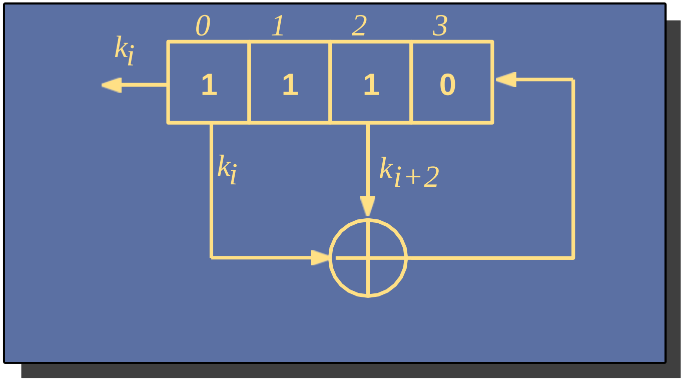
PRACTICAL STREAM CIPHER?

- Use a *Pseudo-Random Number Generator* (PRNG)
- Generate long key streams from a short key: *linear combination of the bits in the primary key*
    - Primary Key $K_0 = 1\ 1\ 1\ 0\ (k_0 k_1 k_2 k_3)$
    - $z_j = k_j$ for $0 \leq j \leq 3$
    - $z_{i+4} = z_i \oplus z_{i+2}$ for all $i \geq 0$
    - Keystream = $z_i = 1\ 1\ 1\ 0\ 0\ 1\ 1\ 1\ 1\ 0 \cdots$
- Periodicity is important – PRNGs repeat numbers after some-time
    - short periods easy to break
    - Above example is bad – repeats after only 6 bits

LFSR

- Practical stream ciphers based on *Linear Feedback Shift Register* (LFSR)
- Linear Feedback Shift Register (LFSR) implements linear combination of its contents

- LFSR for previous example:
  $c_0 = c_2 = 1$
- *Order of LFSR*: length m
- *Tap bits*: 0, 2 where $c_i = 1$



STREAM CIPHER WITH LFSR

- LFSR implements PRNGs for stream ciphers
- It is possible to get a periodicity of $2^m - 1$ for an LFSR of order m
  - Tap bits should be carefully chosen
- Fundamental concept: LFSR of order m is like an m^{th}–degree polynomial

$$P(x) = x^m + c_{m-1}x^{m-1} + \cdots + c_1 x + c_0$$

- The polynomial for previous example with LFSR of order 4 and tap bits of 0 and 2 is

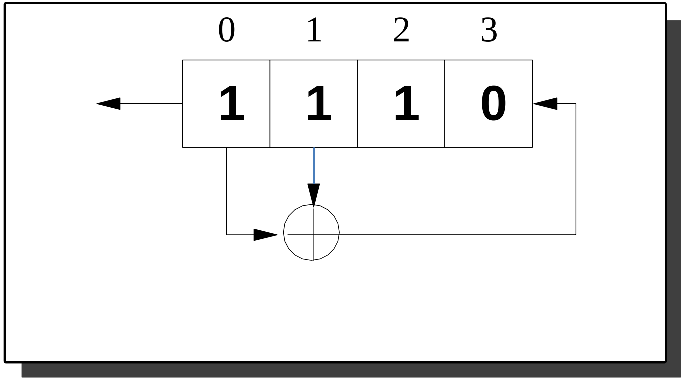
$$P(x) = x^4 + x^2 + 1$$

STREAM CIPHER EXAMPLE

- Primitive polynomial of degree-4

$$P(x) = x^4 + x^3 + 1$$

- LFSR constructed from $P(x)$



- Primitive polynomials are like prime numbers
- The keystream generated by $P(x)$ is

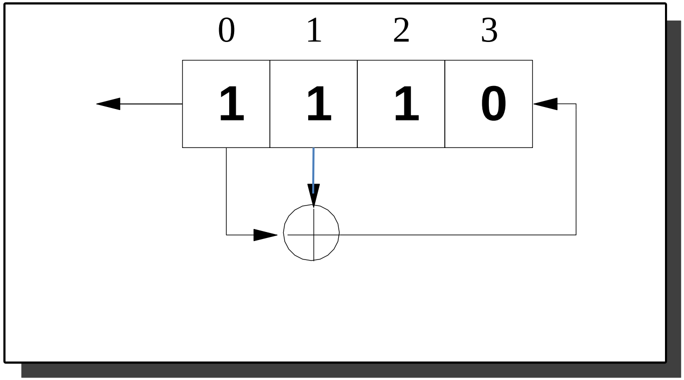
$$\underbrace{1\,1\,1\,0\,0\,0\,1\,0\,0\,1\,1\,0\,1\,0\,1}_{15\text{ bits}}\;1\,1\,1\,0\cdots$$

EXAMPLE

Encryption

- The keystream length = 15 bits
- We use 16 bit keystream for encryption
- Let P = pi
- Encryption as follows

P = 11100010 01101011
K = 11100010 01101011
C = 10010010 00000010



Decryption is reverse

SECURITY ASPECTS

- Never re-use the key
  - Let P_1 and P_2 be two plaintexts encrypted with the same key K
  - Let the two resulting ciphertexts be C_1 and C_2

$$C_1 \oplus C_2 = (P_1 \oplus K) \oplus (P_2 \oplus K) = P_1 \oplus P_2$$

  - This is pretty bad for certain types of messages, e.g., *images*
  - If we know one of the plaintexts, *all other* plaintexts are known!
- Even in LFSR, the initial keys K_0 should be different and randomly chosen each time

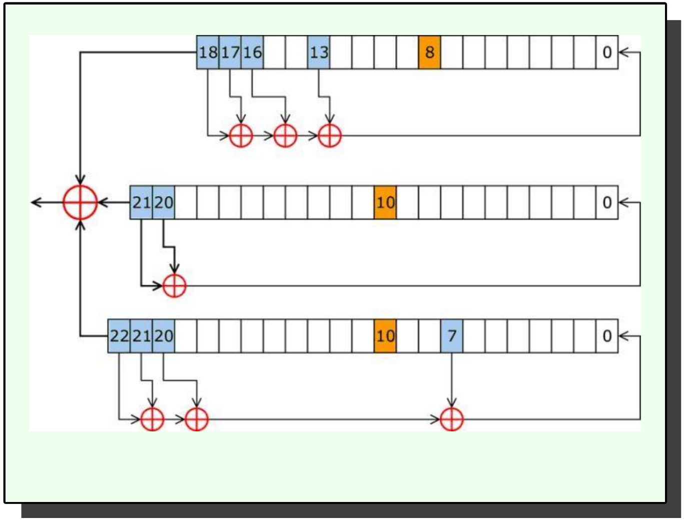
CASE STUDY: A5/1

- A5/1 is widely used in encrypting GSM communications
- It is a stream cipher using *three* LFSRs of lengths 19, 22 and 23 bits
  - Key length = 64 (19 + 22 + 23) bits
  - Periodicity is $2^{64} - 1$ bits
- Interesting *clocking* mechanism and XOR'ing the three outputs give non-linearity
- Exact details never revealed but reverse engineered
  - The reverse engineered algorithm matches *every output* from A5/1 algorithm

A5/1 ARCHITECTURE

LFSR	Len	CLK Bit	Tap Positions
R1	19	8	18,17,16,13
R2	22	10	21,20
R3	23	10	22,21,20,7

- A5/1 LFSRs initialised with 64 bit key *and 22 bit frame number*
- Bits not shifted every time but based on CLK bits
- Keystream used after throwing away first 100 bits
- Output is $b_{18} \oplus b_{40} \oplus b_{63}$



ENCRYPTION ALGORITHM

R1 = R2 = R3 = 0

Load the key K_c into R1, R2, R3 in 64 clock cycles

Load *Frame Number* into R1, R2, R3 in 22 clock cycles

(The contents of R1, R2, R3 define *Initial State*)

for cycle ← 1, 100 **do**

  if CLKbit(R_i) = *Majority* **then**

  Clock R_i (i.e., shift 1-bit to the left)

  else

    Do not change R_i

  end if

  Ignore output

end for

Now we are ready to start encryption