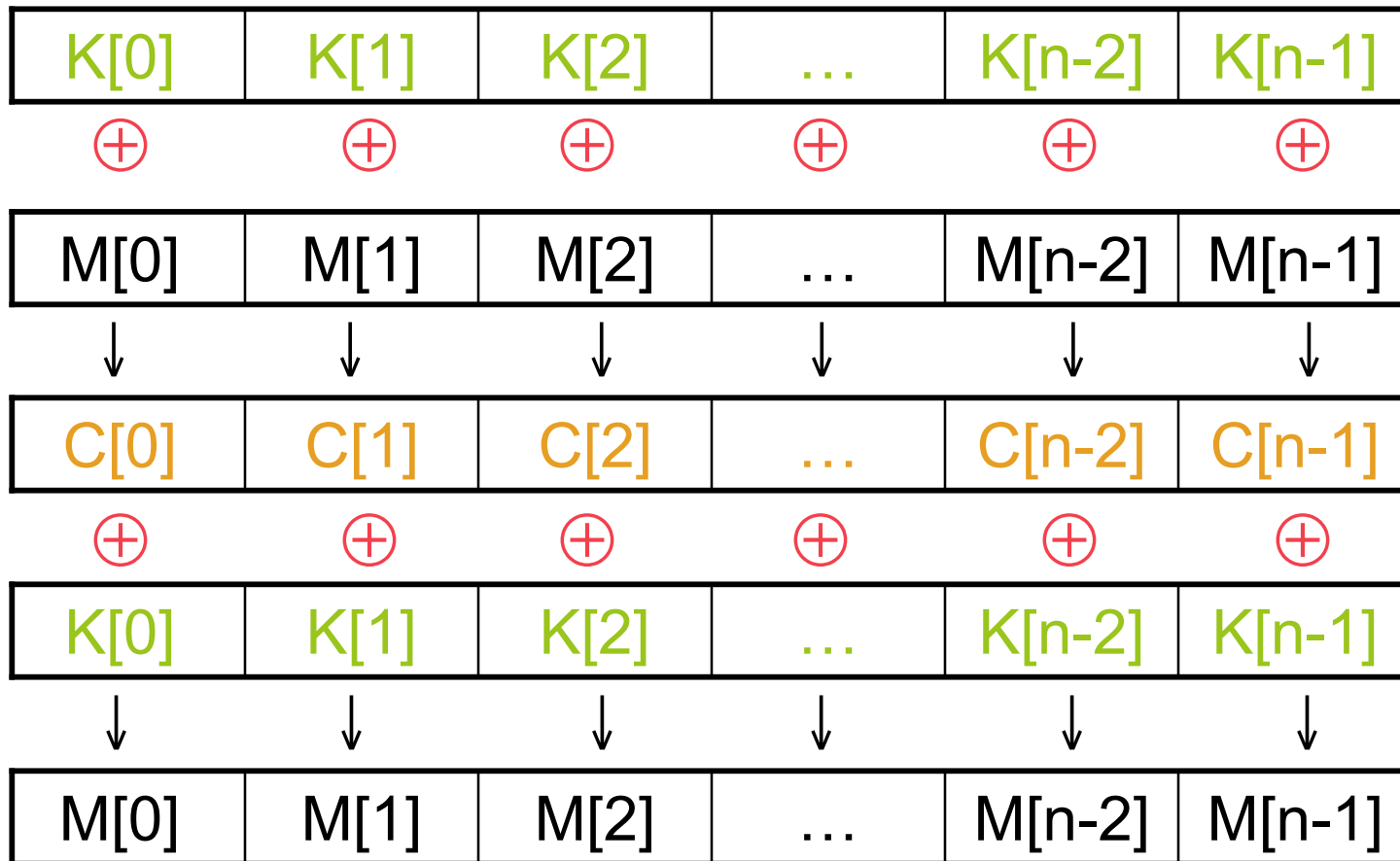


Stream Ciphers

Making the one-time pad practical

Binary One-Time Pad



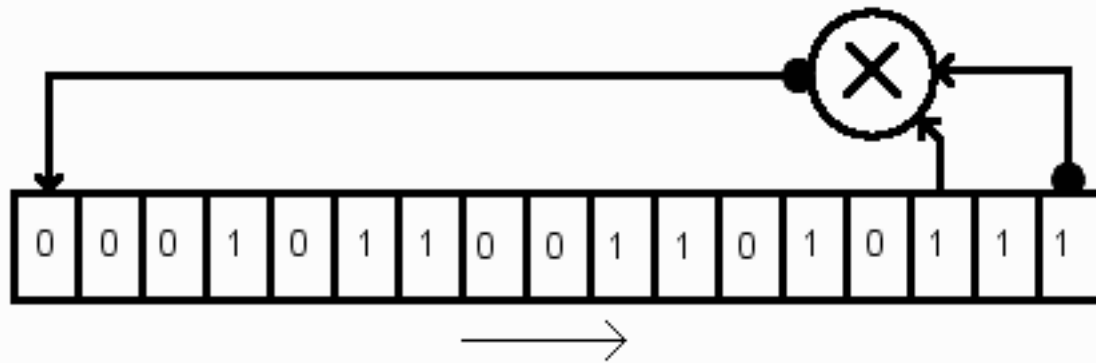
Idea behind stream ciphers

- Start with a fixed-length, shared secret. This is generally called the **seed** s .
- Use a procedure that, with the seed as input, generates a stream of bits that seems random, but which is in fact deterministically computable (from s).
- Use this stream (**keystream**) as the one-time pad: XOR it with the plaintext.

Types of Stream Ciphers

- A stream cipher is a finite state machine (finite input, fixed memory size, deterministic). Two main types:
 - **Key-auto-Key** (KAK, synchronous) -- state determined by last bits of keystream
 - **Ciphertext-auto-key** (CTAK, self-synchronizing) -- state determined by last bits of ciphertext

Linear Feedback Shift Registers



- Compute the parity of “tap” entries in a register
- Shift the register (right shift in the picture)
- Enter the parity bit in the new space (leftmost in picture)

Properties of Shift Registers

- Very efficient generators of pseudo-random sequences. Think of the newly computed bits as the output
- Generate provably long sequences before cycling
- Shift registers in crypto often results in weak ciphers, such as A5/x. However, the **shrinking generator** seems strong.

Shrinking Generator

- Two LFSR generate keystreams s_k and t_k
- If $s_k = 1$, output t_k
- If $s_k = 0$, output nothing; increase k
- Buffer output in order to disguise timing delays which reveal information about the state of keystream s .

RC4

- A state array of 256 bytes: $S[0, \dots, 255]$
- A key K , from 2 to 256 bytes: $K[0, \dots, n]$
- Initialize state as $S[i] = i$
 - FOR $i = 0 \dots 255$
 - $j = j + S[i] + K[i \bmod n] \bmod 256$
 - $\text{SWAP}(S[i], S[j])$

RC4 Stream Generator

- $i = 0; j = 0;$
- WHILE(TRUE)
 - $i = i + 1 \text{ mod } 256;$
 - $j = j + S[i] \text{ mod } 256;$
 - SWAP($S[i]$, $S[j]$);
 - OUTPUT($S[S[i] + S[j] \text{ mod } 256]$);

RC4 Weaknesses

- Initialization starts from a known state.
 - The first bytes of the RC4 keystream are distinguishable from random output, and reveal information about the key.
- Recommendations:
 - Use random IVs, changed everytime.
 - Generate strong pseudo-random keys
 - Drop initial bytes of keystream (768...3072)

Is RC4 Insecure?

- RC4 has been well-studied.
 - No known methods to break RC4 (except brute-forcing the key) when used according to recommendations.
 - E.g.: Robust implementation of RC4 within SSL.
- RC4 was used in a very insecure way in the WEP protocol:
 - No method to distribute initial keys
 - Poor handling of IVs
 - No dropping of the first keystream bytes
- 802.11.x took unnecessary risks.

Stream Cipher Summary

- Stream ciphers are efficient
 - Useful for secure communication with constrained devices (cell phones, smartcards)
- Good stream ciphers available (?)
 - Even as theoretical framework develops
- Never re-use keystreams: must provide mechanism to change IV **EVERYTIME.**