Time-Memory Trade-off Attack

Dr. Sugata Gangopadhyay [1]

Department of Computer Science and Engineering
Indian Institute of Technology Roorkee
Roorkee 247667 INDIA,
sugatfma@iitr.ac.in

ACM Winter School 2019 on Cybersecurity December 10 - 16, 2019 Hosted by : National Institute of Science Education and Research, Bhubanswar.

[1] This material is jointly created with Dr. Nishant Sinha, Security Researcher, Robert Bosch Engineering and Business Solutions Pvt. Ltd., India.

Symmetric Ciphers

- A symmetric cipher can be represented by a 5-tuple $(\mathcal{P}, \mathcal{C}, \mathcal{K}, \mathcal{E}, \mathcal{D})$ where
  - $\mathcal{P}$ is a set of plaintexts;
  - $\mathcal{C}$ is the set of ciphertexts;
  - $\mathcal{K}$ is the set of all keys;
  - The set of encryption functions $E : \mathcal{P} \times \mathcal{K} \to \mathcal{C}$;
  - The set of decryption functions $D : \mathcal{C} \times \mathcal{K} \to \mathcal{P}$,

  and for each $k \in \mathcal{K}$, $D(E(x, k), k) = x$, for all $x \in \mathcal{P}$.

- We may choose to denote $E(\cdot, k)$ by E_k, and $D(\cdot, k)$ by D_k.

Messages, Ciphertexts, and Keys

- A message is a finite binary string, i.e., an element of $\{0,1\}^*$.
- $x \in \{0,1\}^*$ can be written as a concatenation of substrings of a fixed length n as follows:

$$x = x_{\langle 1 \rangle} \| x_{\langle 2 \rangle} \| \ldots \| x_{\langle N \rangle}$$

  where $x_{\langle i \rangle} \in \{0,1\}^n$, for all $i \in \{1, \ldots, N\}$.

- Each $x_{\langle i \rangle} = x_{i1} \ldots x_{in}$, where each $x_{ij} \in \{0,1\}$, is said to be a block.

Notation

- $\mathbb{F}_2$ denotes the finite field with two elements 0 and 1.
- We replace $\{0,1\}$ by $\mathbb{F}_2$.
- We write $\mathbb{F}_2^n$ instead of $\{0,1\}^n$.
- $\mathbb{F}_2^n$ is an n-dimensional vector space over $\mathbb{F}_2$.

Block Ciphers

- A block cipher is a cryptosystem that transforms an n-bit plaintext block to any n-bit ciphertext block using an m-bit key.

- A block cipher is represented by a function $E : \mathbb{F}_2^n \times \mathbb{F}_2^m \to \mathbb{F}_2^n$ where $\mathcal{P} = \mathcal{C} = \mathbb{F}_2^n$ and $\mathcal{K} = \mathbb{F}_2^m$.

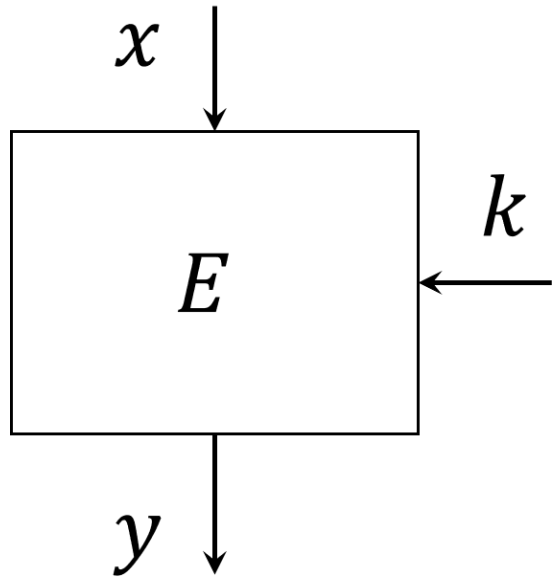

Figure 1: Schematic diagram of a block cipher

Challenges in Block Cipher Construction

- For each $k \in \mathcal{K}$, the function $E(\cdot, k)$ is a permutation on $\mathbb{F}_2^n$.
- There are $(2^n)! \approx 2^{2^n}$ permutations. To attain perfect secrecy, theoretically, we need to choose one permutation from 2^{2^n} permutations uniformly at random.
- The challenge is to construct E so that for each choice of $k \in \mathcal{K} = \mathbb{F}_2^m$, the function $E(\cdot, k)$ behaves as a function chosen uniformly at random from the space of all possible choices.

Block Cipher Designs

- Feistel Network.
- Substitution Permutation Network.

Feistel Network

- Feistel function

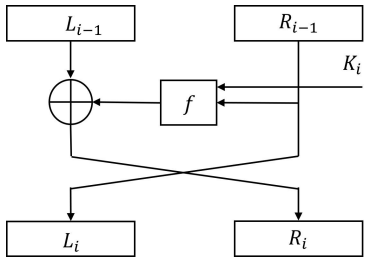

Figure 2: Feistel function

- $L_i = R_{i-1}$,
  $R_i = L_{i-1} \oplus f(R_{i-1} \oplus K_i)$.

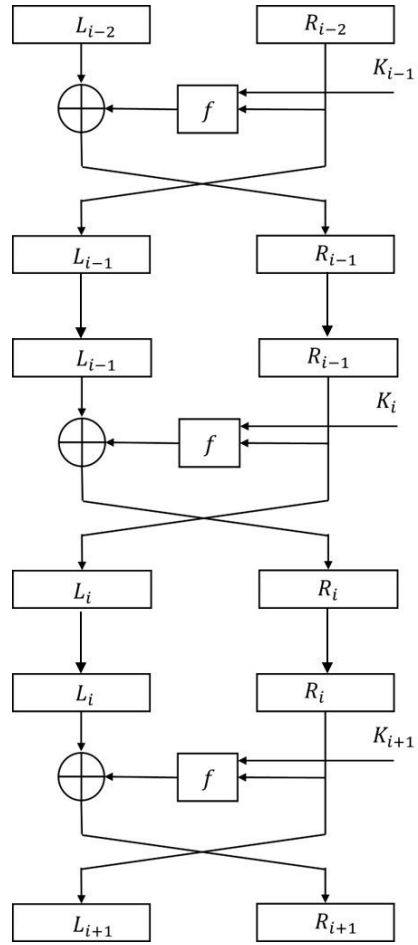

Figure 3: Feistel Network

Substitution-Permutation Network (SPN) 1/2

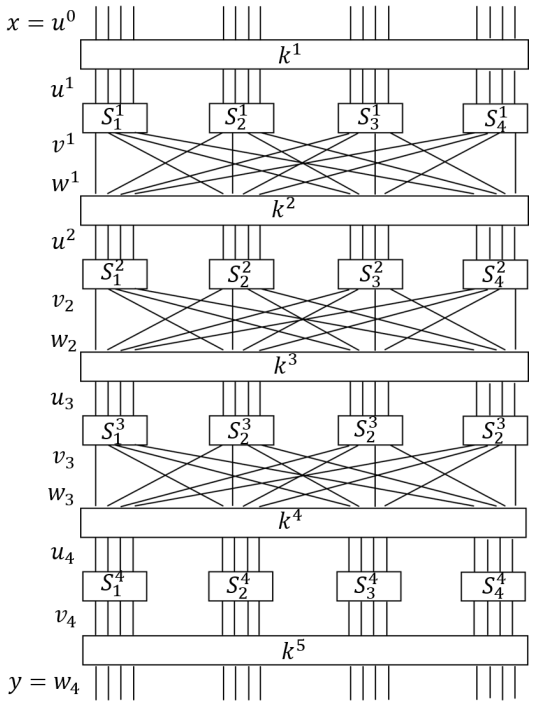

Figure 4: Substitution-Permutation Network having $\mathrm{Nr} = 4$ rounds with block length $\ell m = 4 \times 4 = 16$.

- Plaintext $x = (x_1, \ldots, x_{\ell m})$.
  Ciphertext $y = (y_1, \ldots, y_{\ell m})$.
  Key schedule $k = (k^1, \ldots, k^{\mathrm{Nr}+1})$.
  $\ell = 4, m = 4, \mathrm{Nr} = 4, [n] = \{1, \ldots, n\}$.

- Each S-box, S_i^j is associated with a permutation

$$\pi_{S_i^j} : \mathbb{F}_2^\ell \to \mathbb{F}_2^\ell.$$

- Each substitution layer is followed by a permutation layer specified by a permutation

$$\pi_P = [\ell m] \to [\ell m].$$

Substitution-Permutation Network (SPN) 2/2

- Key-mixing layer: $u^i = u^{i-1} \oplus k^i$.
- Substitution layer: The S-box function $\pi_S : \mathbb{F}_2^\ell \to \mathbb{F}_2^\ell$ is

x	0	1	2	3	4	5	6	7
$\pi_S(x)$	C	5	6	B	9	0	A	D

x	8	9	A	B	C	D	E	F
$\pi_S(x)$	3	E	F	8	4	7	1	2

- Permutation layer: $\pi_P : [\ell m] \to [\ell m]$ is

x	1	2	3	4	5	6	7	8
$\pi_P(x)$	1	5	9	13	2	6	10	14

x	9	10	11	12	13	14	15	16
$\pi_P(x)$	3	7	11	15	4	8	12	16

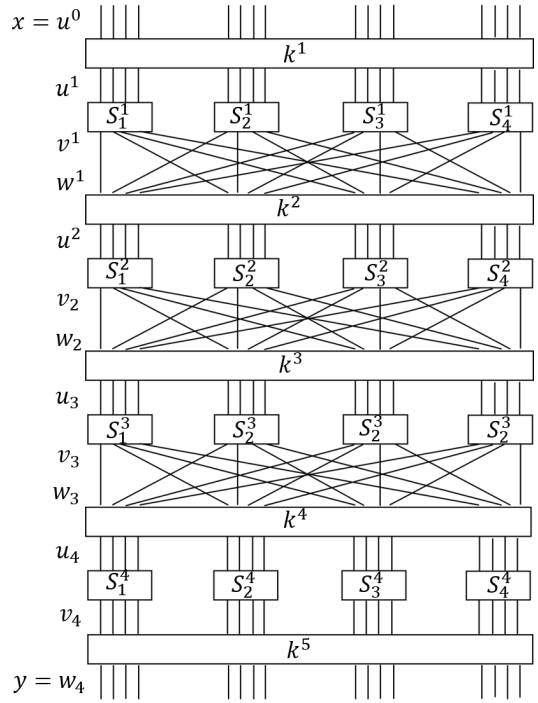

Figure 5: Substitution-Permutation Network

Attack Models[2]

- Ciphertext only attack.
- Known plaintext attack.
- Chosen plaintext attack.
- Chosen ciphertext attack.

[2] Stinson, D. R.: Cryptography – theory and practice. Discrete mathematics and its applications series, CRC Press (1995)

Known plaintext attack on a block cipher

- It is assumed that the adversary knows the plaintext-ciphertext pair (x_0, y_0), but does not know the key k_0.
- The goal of the adversary is to find the secret key k_0.

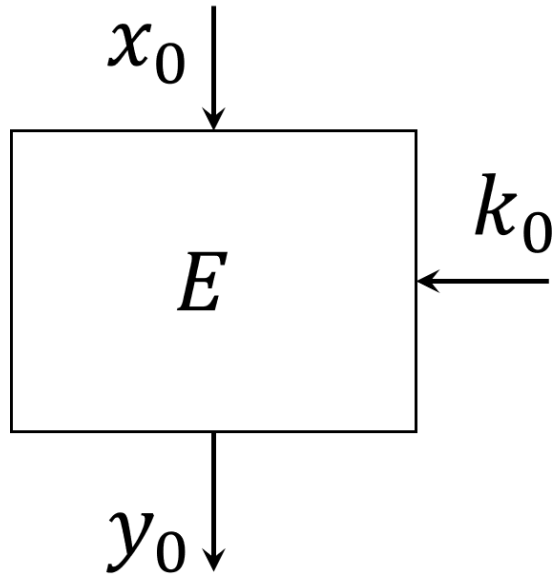

Figure 6: Schematic diagram of a block cipher

Exhaustive Search Attack (1/2)

- Precompute the following table

k_1	$E(x_0, k_1)$	$= y_1$
k_2	$E(x_0, k_2)$	$= y_2$
$\vdots$	$\vdots$	$\vdots$
k_{2^m}	$E(x_0, k_{2^m})$	$= y_{2^m}$

(1)

Table 1: Precomputation table.

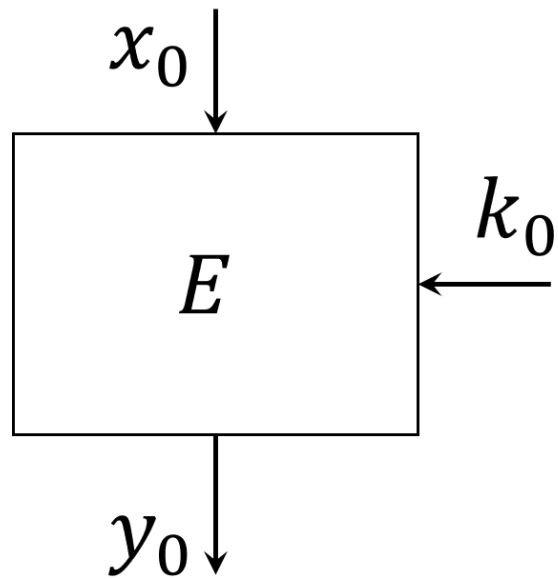

Figure 7: $E : \mathbb{F}_2^n \times \mathbb{F}_2^m \to \mathbb{F}_2^n$.

- In the online phase search for y_0 in the rightmost column of Table 1.
- If $y_j = y_0$, then $k_j = k_0$ the secret key.

Exhaustive Search Attack (2/2)
(The foundation of time-memory trade-off)

- The running-time complexity of the pre-computation phase is $P = O(2^m)$.
- The memory complexity is $M = O(2^m)$.
- The running time complexity in the online phase is $T = O(\log_2(2^m)) = O(m)$, if the list is sorted.
- If we do not store the preprocessing table then $M = O(1)$ and $T = 2^m$.

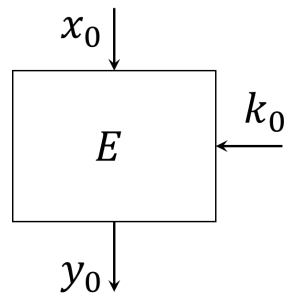

Figure 8: $E : \mathbb{F}_2^n \times \mathbb{F}_2^m \to \mathbb{F}_2^n$.

Time-Memory Trade-Off

- In the late seventies, after acceptance of DES as the encryption standard, Martin Hellman [3] proposed a time-memory trade-off attack on DES.

- In the previous slide, we have seen that if we have access to memory of $O(2^m)$, we can search for the key in $O(1)$ time. Conversely, if we allow computation of order $O(2^m)$, it is possible to mount an attack with $O(1)$ memory.

- Hellman proposed a middle path between these two extreme cases and demonstrated a practical attack on DES.

---

[3] Hellman, M.E.: A cryptanalytic time-memory trade-off. IEEE Trans. Information Theory 26(4), 401–406 (1980)

TMTO: Hellman's Approach

- Hellman considers block ciphers for which

$$\mathcal{P} = \mathcal{K} = \mathcal{C} = \mathbb{F}_2^n.$$

- Chosen plaintext attack: the attacker can choose a plaintext, x say, and obtain the ciphertext $y = E(x, k)$ during the actual operational phase of the cipher without the knowledge of the key $k \in \mathcal{K}$ in use.
- The goal of the cryptanalyst (attacker) is to find the key.

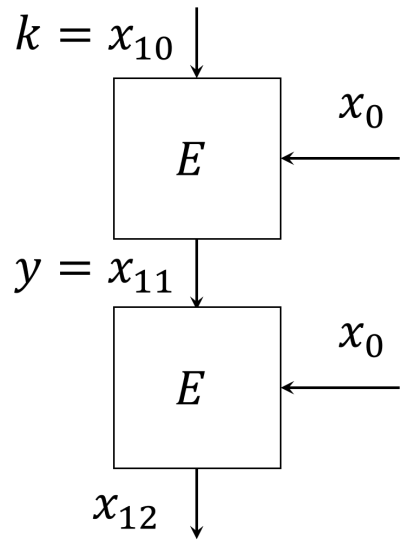

Figure 9: Assuming $n = m$, $f : \mathbb{F}_2^n \to \mathbb{F}_2^n$ defined by $f(x_{ij}) = E(x_0, x_{ij}) = x_{i(j+1)}$.

Preparing for the Precomputation Phase

- $\mathcal{P} = \mathcal{C} = \mathcal{K} = \mathbb{E}_2^n$. Keyspace size: $N = 2^n$.
- Block cipher: $E : \mathcal{P} \times \mathcal{K} \to \mathcal{C}$.
- $f : \mathcal{K} \to \mathcal{C}$ is $f(x_{ij}) = E(x_0, x_{ij}) = x_{i(j+1)} \in \mathcal{C} = \mathcal{K}$.

$$
\begin{array}{ccccccc}
x_{10} & \xrightarrow{f} & x_{11} & \xrightarrow{f} & x_{12} & \xrightarrow{f} \cdots \xrightarrow{f} & x_{1t} \\
x_{20} & \xrightarrow{f} & x_{21} & \xrightarrow{f} & x_{22} & \xrightarrow{f} \cdots \xrightarrow{f} & x_{2t} \\
\vdots & & \vdots & & \vdots & & \vdots \\
x_{m0} & \xrightarrow{f} & x_{m1} & \xrightarrow{f} & x_{m2} & \xrightarrow{f} \cdots \xrightarrow{f} & x_{mt}
\end{array}
$$

Collision Resistance

- For $i = 1$ to m choose x_{i0} at random. For $j = 0$ to $(t-1)$ evaluate $f(x_{ij}) = x_{i(j+1)}$. The table will "cover" mt keys.

- The probability of obtaining a collision is negligible if $(mt)(t) = mt^2 \leq N$. For details we refer to Hellman [4] proposed a time-memory trade-off attack on DES. Biryukov and Shamir [5].

$$
\begin{array}{ccccccc}
x_{10} & \xrightarrow{f} & x_{11} & \xrightarrow{f} & x_{12} & \xrightarrow{f} \cdots \xrightarrow{f} & x_{1t} \\
x_{20} & \xrightarrow{f} & x_{21} & \xrightarrow{f} & x_{22} & \xrightarrow{f} \cdots \xrightarrow{f} & x_{2t} \\
\vdots & & \vdots & & \vdots & & \vdots \\
x_{m0} & \xrightarrow{f} & x_{m1} & \xrightarrow{f} & x_{m2} & \xrightarrow{f} \cdots \xrightarrow{f} & x_{mt}
\end{array}
$$

---

[4] Hellman, M.E.: A cryptanalytic time-memory trade-off. IEEE Trans. Information Theory 26(4), 401–406 (1980)

[5] Biryukov, A., Shamir, A.: Cryptanalytic time/memory/data tradeoffs for stream ciphers. ASIACRYPT 2000. pp. 1–13.

TMTO Matrix

- All these mt vectors successively generated by f starting from randomly chosen vectors x_{i0}, $1 \leq i \leq m$, can be summarized in a single $m \times t$ matrix called TMTO-matrix whose elements are called keys.

- TMTO-matrix:

$$\begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1t} & \\ x_{21} & x_{22} & \cdots & x_{2t} & \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ x_{m1} & x_{m2} & \cdots & x_{mt} & \end{bmatrix} \tag{2}$$

How to cover all the keys?

- Each TMTO-matrix can store at most mt keys where $mt^2 = N$ without the risk of having to store the same key at several positions.

- Instead of f we use $h_j \circ f(x) = h_j(f(x))$, for all $x \in \mathbb{F}_2^n$ where h_j permutes the coordinates of the input vector. Let h_j for $j = 1, \ldots, t$ be t such distinct functions. h_1 is the identity permutation.

- Let the TMTO-matrix generated by $h_j \circ f$, jth TMTO-matrix, be denoted by

$$\begin{bmatrix} x_{11}^{(j)} & x_{12}^{(j)} & \cdots & x_{1t}^{(j)} & \\ x_{21}^{(j)} & x_{22}^{(j)} & \cdots & x_{2t}^{(j)} & \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ x_{m1}^{(j)} & x_{m2}^{(j)} & \cdots & x_{mt}^{(j)} & \end{bmatrix} \tag{3}$$

How to cover all the keys?

- Hellman envisioned that if each of these functions is used to generate a TMTO-matrix, the resulting t tables will cover all the $mt^2 = N$ keys, defying the Birthday Paradox.
- Hellman was proved to be correct. The time complexity of the preprocessing phase is $P = N$, since all the keys have to be visited.
- This computation may take several weeks or might be years! However, this is a one-time investment.

Memory Requirement to Store the Tables

- Instead of storing all the mt keys of each of the t matrices, only the starting and the ending keys are stored in a list.
- Instead of the jth TMTO-matrix we construct a list L_j of the pairs $(x_{i0}^{(j)}, x_{it}^{(j)})$ for $i = 1, \ldots, m$, sorted with respect to the second-component.
- Let us denote the list corresponding to the jth matrix by L_j.
- Therefore, we need mt memory locations to store t matrices.

TMTO Lists

- The TMTO lists are as follows:

$$L_1 = \begin{bmatrix} x_{11}^{(1)} & x_{1t}^{(1)} \\ x_{21}^{(1)} & x_{2t}^{(1)} \\ \vdots & \vdots \\ x_{m1}^{(1)} & x_{mt}^{(1)} \end{bmatrix}, L_2 = \begin{bmatrix} x_{11}^{(2)} & x_{1t}^{(2)} \\ x_{21}^{(2)} & x_{2t}^{(2)} \\ \vdots & \vdots \\ x_{m1}^{(2)} & x_{mt}^{(2)} \end{bmatrix}, \ldots, L_t = \begin{bmatrix} x_{11}^{(t)} & x_{1t}^{(t)} \\ x_{21}^{(t)} & x_{2t}^{(t)} \\ \vdots & \vdots \\ x_{m1}^{(t)} & x_{mt}^{(t)} \end{bmatrix}.$$

- The first column in each L_j is

Online Phase: Algorithm

Input: (x_0, y_0), L_j for $j = 1, \ldots, t$;
Output: The secret key k_0;
for $i = 0$ ***to*** $t - 1$ **do**
  for $j = 1$ ***to*** t **do**
    $y' \leftarrow h_j(y_0)$;
    $y'_i \leftarrow (h_j \circ f)^i(y')$;
    ▷ search in the sorted list L_j, for all j
    if $y'_i \in L_j$ **then**
      determine ℓ such that $y'_i = x_{\ell t}^{(j)}$;
      return $k = (h_j \circ f)^{(t-i-1)}(x_{\ell 0}^{(j)})$;
    end
  end
end

Algorithm 1: Time-memory trade-off

Analysis

- In case all the keys are covered in the tables, the attack will be successful.
- The memory requirement for each table is m. So for t tables the memory requirement is $M = mt$.
- If we assume that the tables are sorted, the time to search each table is $\log_2 m$.
- In case we have to search all t tables before finding a match, the time complexity if $t \times \log_2 m$.

Analysis

- To search in a table in the ith iteration, first f has to be applied $(i-1)$ times on the ciphertext. If there is a match, f has to be applied on the corresponding starting point (SP) $(t-i)$ times.

- The total number of times f has to be applied is $t-i+i-1=t-1$.

- The time complexity of the attack is

$$T=(t-1)\times t\times \log_2 m.$$

Complexity of Attack

- $M =$ the storage memory.
- For each table we need m locations to store the first and the last points of each row.
- For t tables we require mt locations.
- The preprocessing and online space complexity is $M = mt$.

Preprocessing Time Complexity

- While preparing the tables we have to cover all the N keys.
- The preprocessing time complexity is $P = N$.

Online Phase Time Complexity

- The function f is applied on the output t times for each table to determine whether the key appears in the table.
- Since there are t tables and for each table computation is of the order t we have the online phase time complexity $T = t^2$.
- In case we include the time complexity of the table search in the online phase $T = t^2 \log_2 m$.

The Trade-Off Curve

- $M = mt$
- $P = N$
- $N = mt^2$
- $T = t^2$
- $TM^2 = t^2 m^2 t^2 = (mt^2)^2 = N^2.$

Some Thoughts on Hellman's Trade-Off Curve

- For a more refined analysis we may argue as follows:
- $M = 2mt \log_2 N$
- $N = mt^2$
- $T = t^2 \log_2 m$
- $$\begin{aligned} TM^2 &= t^2 \log_2 m (2mt \log_2 N)^2 \\ &= (mt^2)^2 4 \log_2 m (\log_2 N)^2 \\ &= N^2 4 \log_2 m (\log_2 N)^2. \end{aligned}$$
- Therefore some refinements in certain cases may be possible.

Stream Ciphers: Introduction

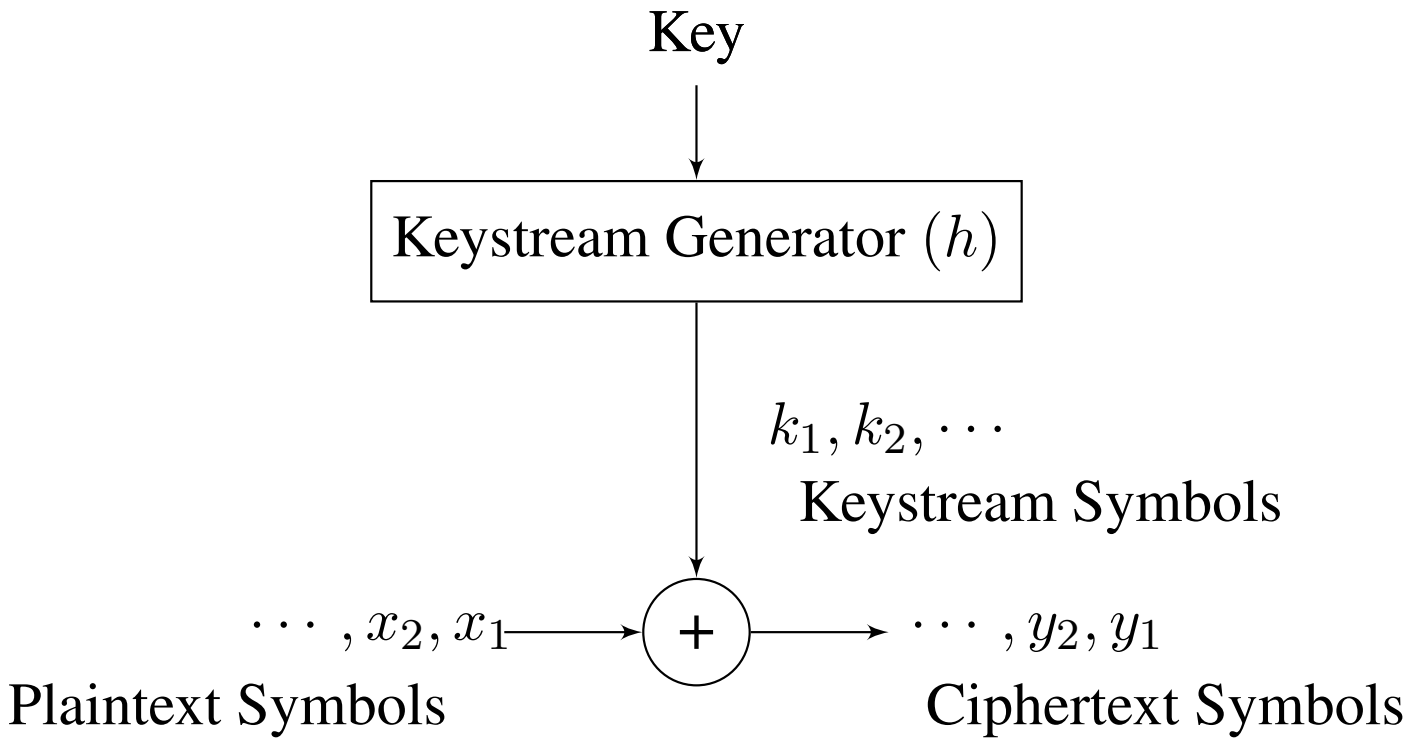

Figure 10: Encryption process of an additive stream cipher.

Feedback Shift Registers

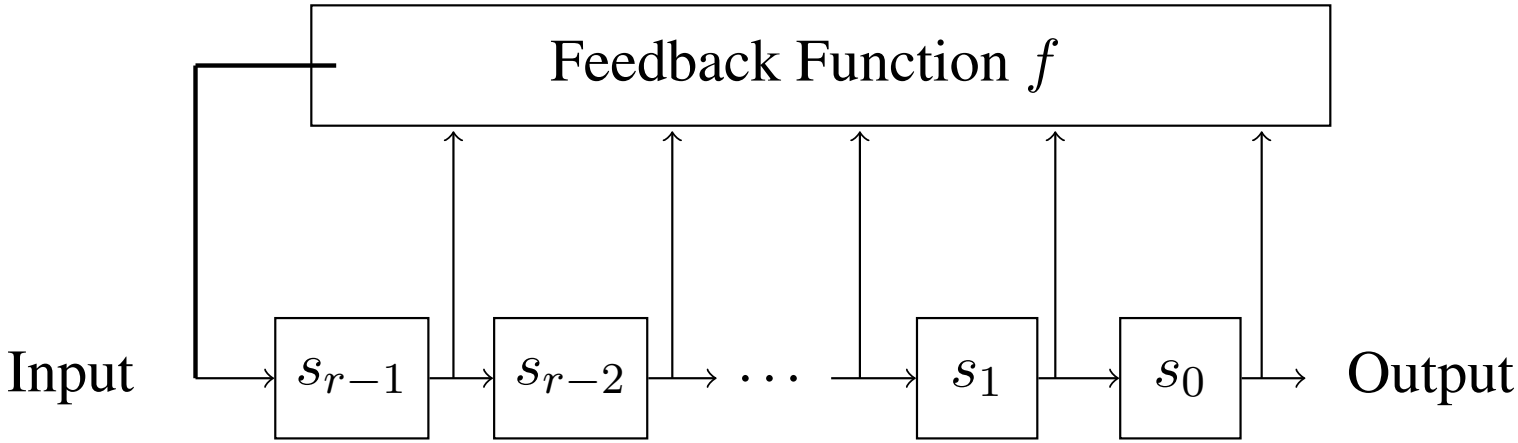

Figure 11: A block diagram of Feedback Shift Register (FSR).

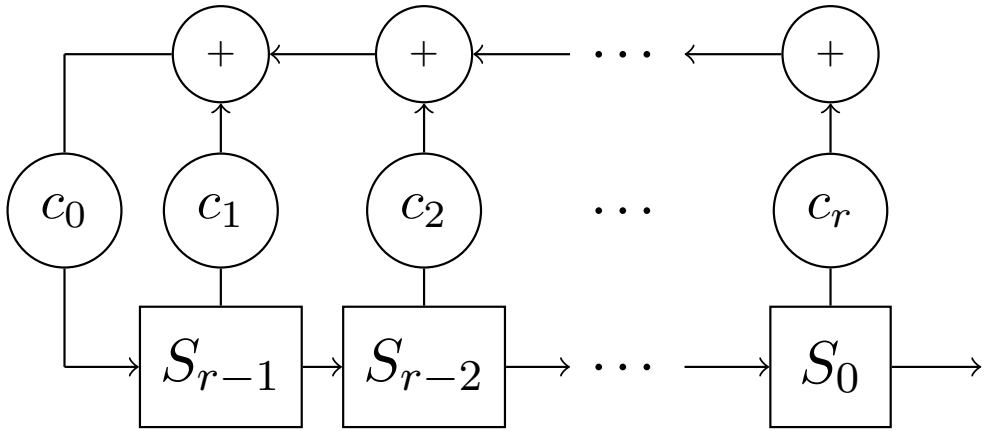

Figure 12: A block diagram of an LFSR.

Nonlinear Combiner Generator

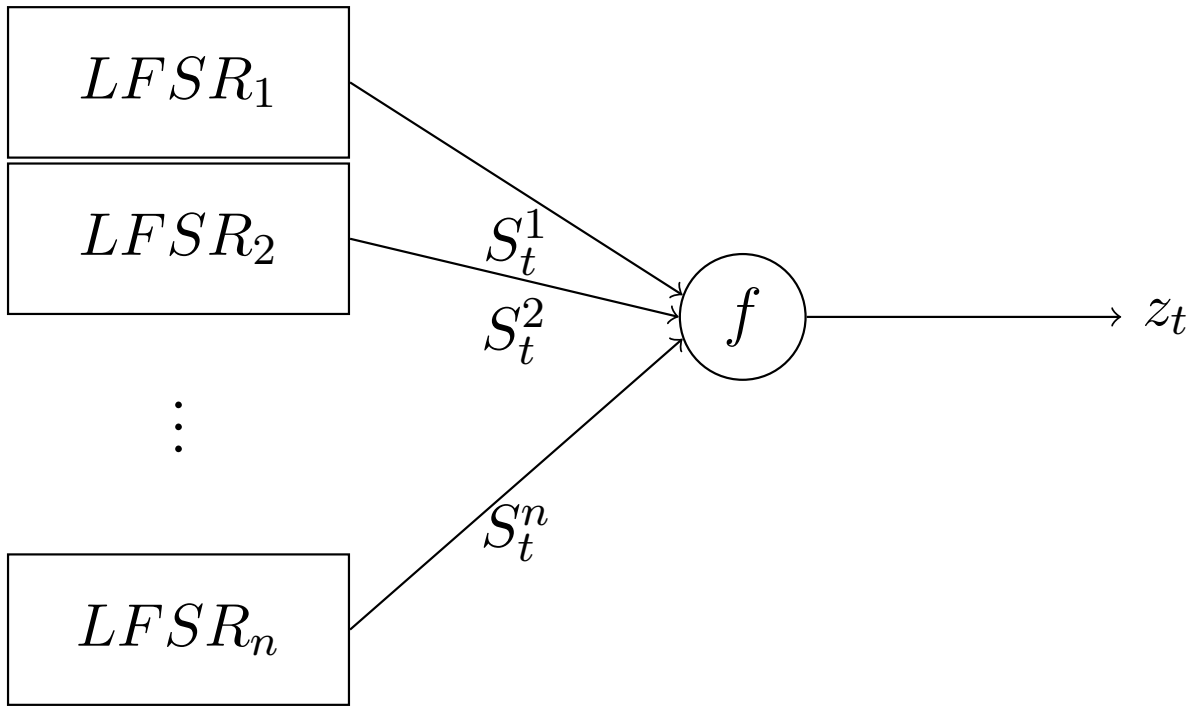

Figure 13: A block diagram of Nonlinear combination generator.

Nonlinear Filter Generator

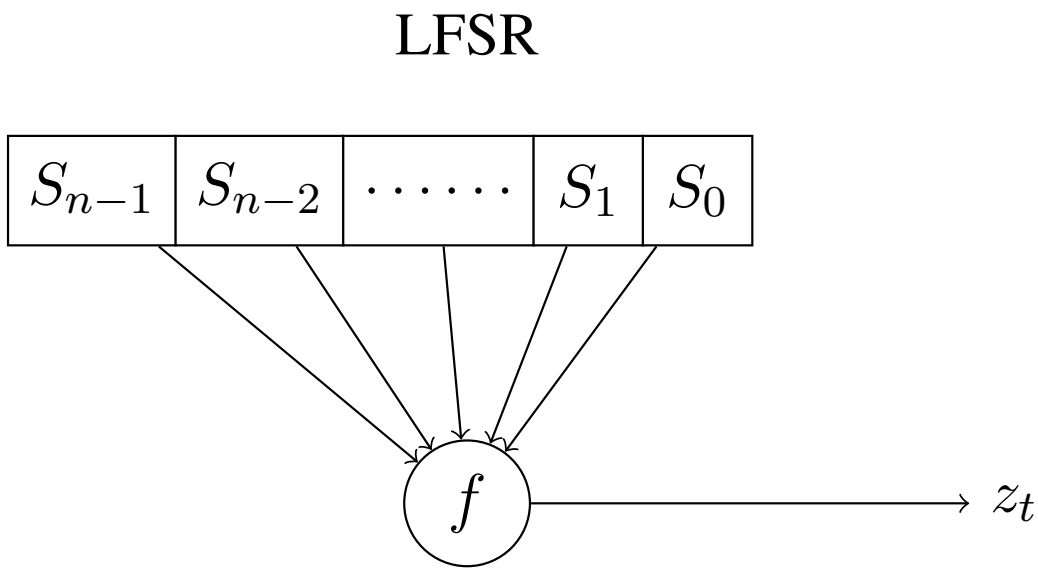

Figure 14: A block diagram of Nonlinear filter generator.

A modern stream cipher: Grain-v1 [6]

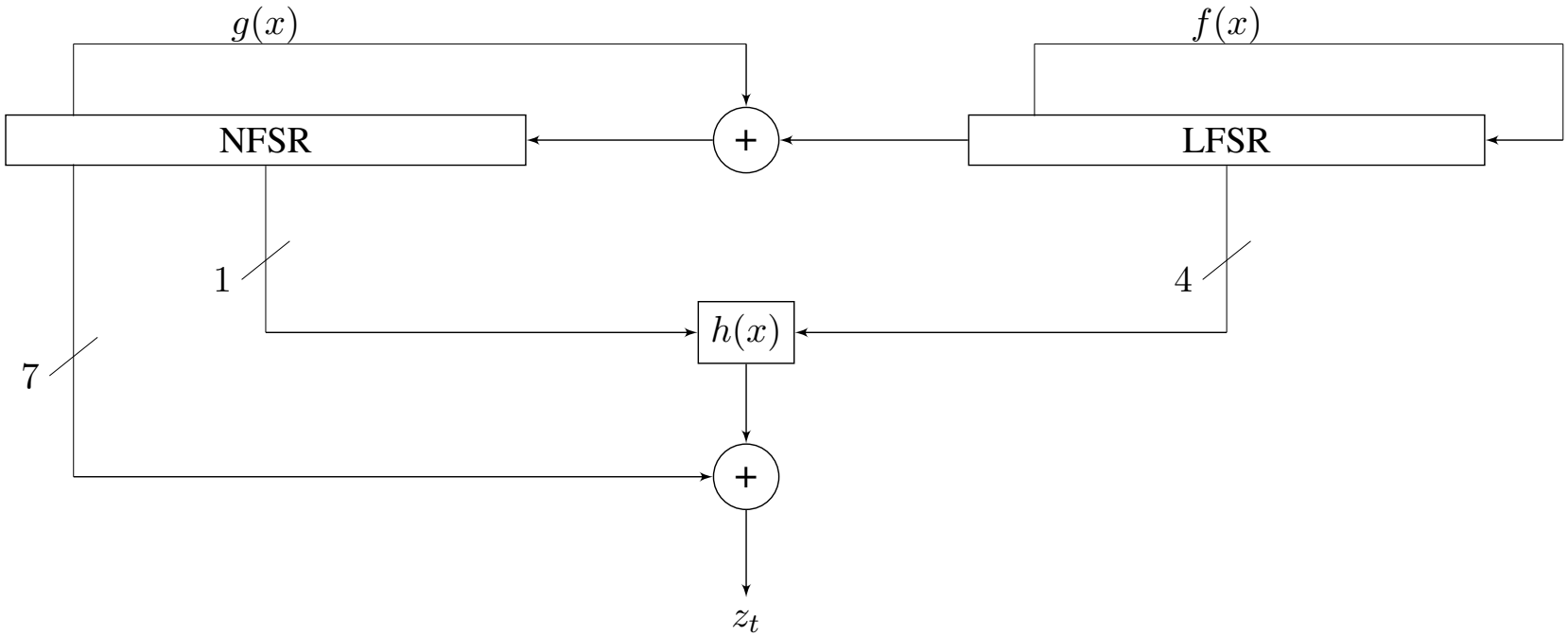

Figure 15: Grain-v1: Keystream generation

[6] Hell, M., Johansson, T., Meier, W.,: Grain: a stream cipher for constrained environments. International Journal of Wireless and Mobile Computing (IJWMC), vol. 2, No. 1, 2007. pp. 86-93. (2007)

A modern stream cipher: Grain-v1

- Let the vectors $b^{(i)}$ and $s^{(i)}$ be the states of the NFSR and LFSR, respectively, at the instant i.
- $$s^{(i)} = [s_i, s_{i+1}, \ldots, s_{i+79}],$$
$$b^{(i)} = [b_i, b_{i+1}, \ldots, b_{i+79}].$$
- Let $u^{(i)}$ be the internal state of Grain-v1, and accordingly,

$$u^{(i)} = [s^{(i)}||b^{(i)}] = [s_i, s_{i+1}, \ldots, s_{i+79}, b_i, b_{i+1}, \ldots, b_{i+79}].$$

A modern stream cipher: Grain-v1

- The feedback function of LFSR

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

- The feedback function of NFSR

$$\begin{aligned} b_{i+80} = {} & s_i \oplus b_{i+62} \oplus b_{i+60} \oplus b_{i+52} \oplus b_{i+45} \oplus b_{i+37} \oplus b_{i+33} \oplus b_{i+28} \oplus b_{i+21} \oplus b_{i+14} \\ & \oplus b_{i+9} \oplus b_i \oplus b_{i+63}b_{i+60} \oplus b_{i+37}b_{i+33} \oplus b_{i+15}b_{i+9} \oplus b_{i+60}b_{i+52}b_{i+45} \\ & \oplus b_{i+33}b_{i+28}b_{i+21} \oplus b_{i+63}b_{i+45}b_{i+28}b_{i+9} \oplus b_{i+60}b_{i+52}b_{i+37}b_{i+33} \\ & \oplus b_{i+63}b_{i+60}b_{i+21}b_{i+15} \oplus b_{i+63}b_{i+60}b_{i+52}b_{i+45}b_{i+37} \\ & \oplus b_{i+33}b_{i+28}b_{i+21}b_{i+15}b_{i+9} \oplus b_{i+52}b_{i+45}b_{i+37}b_{i+33}b_{i+28}b_{i+21}. \end{aligned}$$

A modern stream cipher: Grain-v1

- The function h is defined as

$$\begin{aligned} h(x) \quad = \quad & x_1 \oplus x_4 \oplus x_0x_3 \oplus x_2x_3 \oplus x_3x_4 \oplus x_0x_1x_2 \\ & \oplus x_0x_2x_3 \oplus x_0x_2x_4 \oplus x_1x_2x_4 \oplus x_2x_3x_4, \end{aligned}$$

- where the variables x_0, x_1, x_2, x_3 and x_4 correspond to the tap positions s_{i+3}, s_{i+25}, s_{i+46}, s_{i+64} and b_{i+63} respectively.

A modern stream cipher: Grain-v1

- The keystream is computed as

$$z_i = \left(\bigoplus_{k \in \mathcal{A}} b_{i+k} \right) \oplus h(s_{i+3}, s_{i+25}, s_{i+46}, s_{i+64}, b_{i+63}), \quad (4)$$

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

Time-Memory-Data Trade-Off for Stream Ciphers

- The fundamental difference between trade-off attack on block ciphers and stream ciphers is that a key stream is used in stead of a key. Therefore, the idea of a message being transformed by a unique unknown key has no meaning.

- This renders the known plaintext attack as we discussed above meaningless.

Time-Memory-Data Trade-Off for Stream Ciphers

- For stream ciphers cryptanalysis, we assume that the adversary has access to a key stream of length D.
- Let n be the number of bits in the internal state of the stream cipher under consideration.
- If D is sufficiently larger than n, that is if $n << D$, the adversary can obtain approximately D distinct consecutive key stream segments of length n. (How!)
- In that case we will say that the adversary has access to a data set of size D.

The Trade-Off Curve $TM^2D^2 = N^2$

- D is the number of keystream segments available to the adversary.
- Let a stream cipher have an n-bit internal state. So that total number of possible internal states is $2^n = N$.
- Instead of covering all the N internal states in the time-memory trade-off table,we cover $\frac{N}{D}$ internal states.
- We call this new strategy time-memory-data trade-off (TMDTO) instead of time-memory trade-off (TMTO).

The Trade-Off Curve $TM^2D^2 = N^2$

- $M = \frac{mt}{D}$
- $T = t^2$
- $N = mt^2$
- $TM^2D^2 = t^2 \frac{m^2t^2}{D^2} = m^2t^4 = N^2$
- Along with $t \geq D$ so $\sqrt{T} \geq D$
- $P = \frac{N}{D}$

Why $\frac{N}{D}$ internal states to be considered and not N?

- If we search $\frac{N}{D}$ tables by using a dataset of size D, then the expected number of match is 1.
- For this reason, it is enough to consider $\frac{N}{D}$ internal states rather than N internal states.
- The preprocessing time complexity reduces from $P = N$ to $P = \frac{N}{D}$.

Why $t \geq D$?

- Each table covers mt internal states.
- To avoid collision, while preparing a table, any optimal choice of m and t must satisfy $mt^2 = N$.
- We have to cover $\frac{N}{D}$ internal states. So we need $\frac{t}{D}$ tables. Since $mt \times \frac{t}{D} = \frac{mt^2}{D} = \frac{N}{D}$.
- Therefor $t \geq D$.

Memory in Preprocessing

- Memory requirement in the preprocessing phase is $M = m\frac{t}{D}$. That is, the product of the number of rows in each table with the total number of tables.

- Time complexity:
  - Searching $\frac{t}{D}$ tables requires $\frac{t}{D}$ time for 1 data point.
  - For the recursive use of f and computation after a hit we require t time.
  - So for a single data point $t \times \frac{t}{D} = \frac{t^2}{D}$.
  - For D data points $T = D \times \frac{t^2}{D} = t^2$.

A Warning?

- The final trade-off curve is $TM^2D^2 = N^2$.
- We often forget that there are two hidden parameters m and t such that $mt^2 = N$, and the number of tables is $\frac{t}{D}$ that forces $t \geq D$.
- These hidden variables impose considerable restrictions on the choices of T, M, D that should be always kept in mind.

Tutorial 1: TMTO Matrix Stopping Rule

The calculations are at best approximate

Birthday Paradox

- Any optimal choice of m and t that avoids collision must satisfy the equation $mt^2 = N$.
- Suppose that mt elements are generated in the table without any collision.
- We choose $x_{(m+1)0}$ to be distinct from the above and generate t elements by t times iteratively applying f.
- The probability that none of the elements computed in the last chain appear in the table is approximately $\left(1 - \frac{mt}{N}\right)^t \approx e^{-\frac{mt^2}{N}}$.

Birthday Paradox

- $\left(1 - \frac{mt}{N}\right)^t \approx e^{-\frac{mt^2}{N}}$.
- $e^{-\frac{mt^2}{N}} \geq \frac{1}{2}$; $\ln e^{-\frac{mt^2}{N}} \geq \ln \frac{1}{2}$;
- $-\frac{mt^2}{N} \geq -\ln 2$; $\frac{mt^2}{N} \leq \ln 2$; $mt^2 \leq (\ln 2)N$.

Birthday Paradox

- Suppose that we have generated mt keys in the table. If we generate t more keys one by one after that then the probability that no key is repeated is

$$\left(1-\frac{mt}{N}\right)\left(1-\frac{mt+1}{N}\right)\left(1-\frac{mt+2}{N}\right)\ldots\left(1-\frac{mt+t-1}{N}\right)$$
$$\approx e^{-\frac{mt}{N}}e^{-\frac{mt+1}{N}}e^{-\frac{mt+2}{N}}\ldots e^{-\frac{mt+t-1}{N}}$$
$$= e^{-\frac{2mt^2+t^2-t}{N}} \geq \frac{1}{2}$$

- $\frac{2mt^2+t^2-t}{N} \leq \ln 2$

- $mt^2 \leq (\ln 2)N$

Expected Number of Collisions !

- In order to avoid collision and cover all the keys Hellman (1978) employed a very sophisticated technique to generate multiple tables.

- Our question is If we use a single table so that $mt = N$, then what is the expected number of collisions?

Expected Number of Collisions !

- As long as $mt^2 \leq N$ i.e., $m \leq \frac{N}{t^2} = m_0$, the probability of collision is negligible.
- Let $m \geq m_0 + 1$.
- As each term of the mth row is generated, the probability p that it collides with a previously generated term should approximately satisfy: $\frac{(m-1)t}{N} \leq p \leq \frac{mt}{N}$.
- The expected number of collision in the mth row, E_m is $\frac{(m-1)t^2}{N} \leq E_m \leq \frac{mt^2}{N}$.

Expected Number of Collisions !

- Assuming $m_0 t^2 = N$, and summing up the expectations from $m = m_0 + 1 = \frac{N}{t^2} + 1$ to $N' = \frac{N}{t}$ we have:

- $\sum_{m=m_0+1}^{N'} \frac{m-1}{m_0} \leq \sum_{m=m_0+1}^{N'} E_m = E \leq \sum_{m=m_0+1}^{N'} \frac{m}{m_0}$

- $\sum_{m=m_0+1}^{N'} \frac{m}{m_0} \approx \frac{N'^2}{2m_0} - \frac{m_0}{2} + \frac{1}{2}$

- $\sum_{m=m_0+1}^{N'} \frac{m-1}{m_0} \approx \frac{N'^2}{2m_0} - \frac{m_0}{2} - \frac{1}{2}$

- An approximate value of the expected number of collisions is

$$E \approx \frac{N'}{2m_0} - \frac{m_0}{2} = \frac{N}{2} - \frac{N}{2t^2}.$$

Question

- What is the probability of successful cryptanalysis if some one uses a single table with $mt = N$ and applies TMTO?

Tutorial 2: Cryptanalysis of Reduced Grain-v1[7]

[7] Zhang, B., Li, Z., Feng, D., and Lin, D.: Near Collision Attack on the Grain v1 Stream Cipher, 518-538. doi: 10.1007/978-3-662-43933-3_27, (2014).

The Reduced Grain-v1

- The reduced version of Grain-v1 cipher consists of an LFSR of 32 bits and an NFSR of 32 bits.

- LFSR: $l_{i+32} = l_{i+30} + l_{i+25} + l_{i+16} + l_i$.

- $n_{i+32} = l_i + n_{i+25} + n_{i+23} + n_{i+15} + n_{i+8} + n_i$
  $+n_{i+25}n_{i+23} + n_{i+15}n_{i+8} + n_{i+25}n_{i+23}n_{i+15} + n_{i+23}n_{i+15}n_{i+8}$
  $+n_{i+25}n_{i+23} + n_{i+15}n_{i+8}$

The Reduced Grain-v1

- $$\begin{aligned} h(x) &= x_1 \oplus x_4 \oplus x_0x_3 \oplus x_2x_3 \oplus x_3x_4 \oplus x_0x_1x_2 \\ &\quad \oplus x_0x_2x_3 \oplus x_0x_2x_4 \oplus x_1x_2x_4 \oplus x_2x_3x_4, \end{aligned}$$

- $\mathcal{A} = \{1, 4, 10, 21\}$.

- $z_i = \sum_{k \in \mathcal{A}} n_{i+k} + h(l_{i+3}, l_{i+11}, l_{i+21}, l_{i+25}, n_{i+24})$

Question

- Set up a time-memory-data trade-off attack on the reduced version of Grain-v1.