A review of time-memory trade-off techniques

Liuteng, LiZelong
University of Jinan
Shandong province, China
Contact Email: 269810721@qq.com

山东产业技术研究院
Shandong Institutes of Industrial Technology

Liuteng

He is currently studying in the school of information science and engineering of Jinan University in Shandong Province, China, and his research direction is network security. This paper mainly focuses on the research of rainbow trade-off, a time-memory trade-off algorithm, and aims to improve the efficiency of rainbow trade-off in password attack.

LANJIANJUNXIN

山东产业技术研究院
Shandong Institutes of Industrial Technology

The Cryptography Cloud Computing Power Center is jointly established by Shandong Industrial Technology Research Institute and Jinan Blue Sword Junxin Information Technology Co., Ltd.

Its core function is to play a driving role in industrial aggregation for the development of cyberspace security, blockchain, and artificial intelligence industries in Shandong Province, connect the digital economy industry chain, create a new engine for the conversion of old and new kinetic energy, create a nationally influential information security and artificial intelligence industry demonstration base, and form an information security industry ecological community with a complete industrial chain and industrial advantages

01

PART

What Is Time-Memory Trade-Off?

What is time-memory trade-off?

Time-Memory Trade-Off

Time-memory trade-off is a concept in computer science and cryptography that refers to the balance between the time taken to perform a computation and the amount of memory space required to store precomputed data. This trade-off arises in various algorithms and techniques, particularly in cryptanalysis, where an attacker attempts to recover a cryptographic key or password.

- **Time Complexity**

  This refers to the amount of time an algorithm takes to perform a computation. It have a high time complexity because they require a lot of time to test each possibility.

- **Memory Complexity**

  This refers to the amount of memory space needed to store data. Techniques like time-memory trade-off involve precomputing hashes of many possible passwords and storing them in a trade-off table.

02

PART

Some Related Concepts

Some related concepts

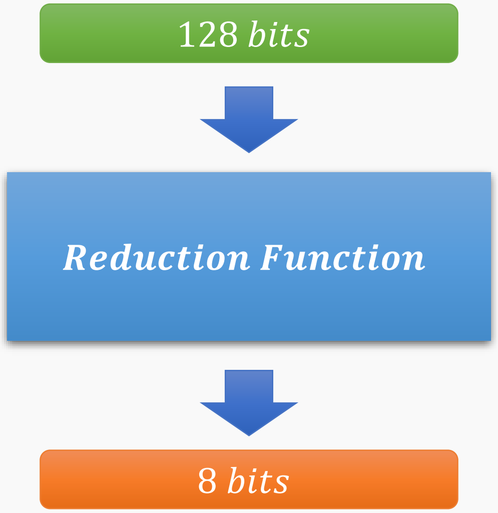
- **Reduction Function**

A reduction function is a deterministic algorithm that takes a hash value as input and produces a plaintext-password as output. The purpose of the reduction function is to reduce a large hash space (the set of all possible hash values) into a smaller plaintext space (the set of all possible passwords).

The reduction function consistently maps hash values to specific passwords. When building trade-off table, attackers apply the reduction function iteratively to generate a chain of plaintext-passwords from a given hash value.

Example:

Here is 128bits Hash string as input

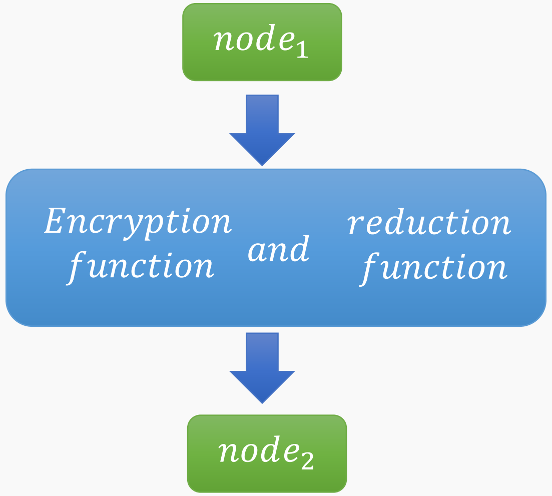


The output is 8bits ciphertext

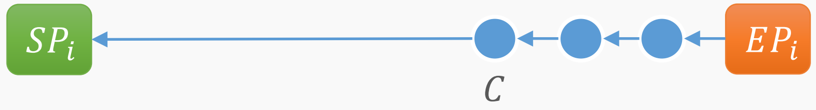
Some related concepts

● Pre-computation phase

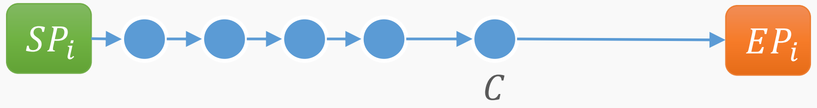
The pre-computation phase involves generating the precomputation table by creating precomputation chains according to reduction function. The starting and ending points of these precomputation chains are saved to form the precomputation table. Finally, for ease of retrieval, the table can be sorted based on the size of the ending points.



$Generating\ precomputation\ chains$:



$Precomputation\ chain\ reconstruction$:



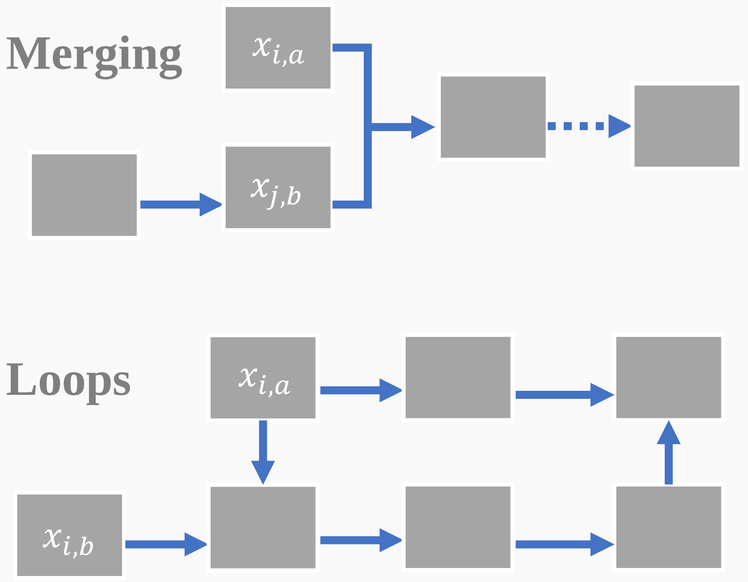
● Attack phase

The actual attack phase includes three components: generating precomputation chains, collision detection, and precomputation chain reconstruction.

Some related concepts

- **Collisions, Merging, and Loops**

Collisions refer to the occurrence of the same node in two different chains, for example, $x_{i,a} = x_{j,b}$, where $i \neq j$.



Comparison between Merging and Loops

If all nodes following the collision point in the precomputation phase are the same, a merging occurs.

When multiple collisions occur in a chain, a loop occurs.

03

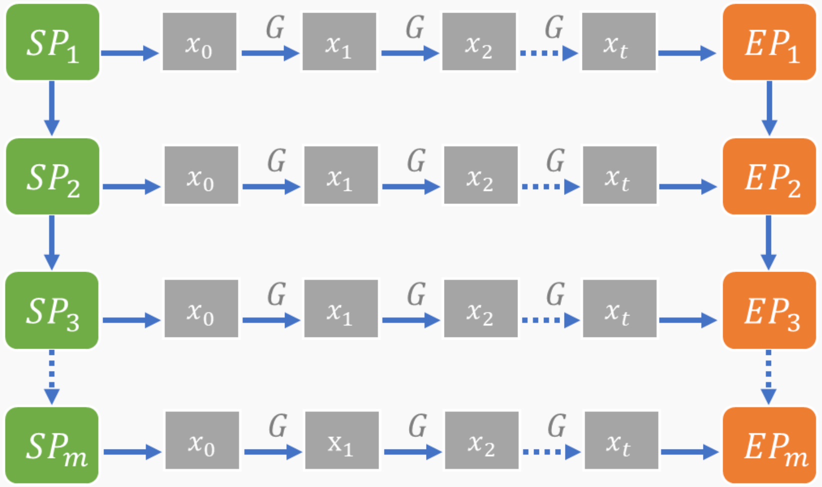
PART

Time-Memory Trade-Off

Time-memory trade-off

Hellman Trade-Off

In 1980, Hellman introduced a classic time-memory trade-off method [6] for attacking the Data Encryption Standard (DES) algorithm, a symmetric-key block cipher.



$Schematic\ of\ Hellman\ Trade-off$

- Starting points $(SP_1, SP_2, SP_3, \ldots, SP_m)$ generate multiple ending points$(EP_1, EP_2, EP_3, \ldots, EP_m)$ through t iterations. But only the key-value$(SP_i - EP_i)$ are stored in the Hellman table.

Time-memory trade-off

- **Hellman Trade-Off**

Within the framework of the Hellman trade-off, we assume the presence of m table entries in each table, with each pre-computed chain undergoing t iterations. We offer the probability of effectively locating the desired key within a specified table:

$$P_{single} \geq \frac{1}{N}\sum_{i=1}^{m}\sum_{j=0}^{t-1}\left(1-\frac{it}{N}\right)^{j+1}$$

Increasing the success rate of attacks using the Hellman trade-off usually entails the utilization of distinct G functions to produce multiple Hellman tables. Assuming the existence of l tables, the probability of successfully determining the target key is as follows:

$$P_{success} \geq 1-(1-\frac{1}{N}\sum_{i=1}^{m}\sum_{j=0}^{t-1}\left(1-\frac{it}{N}\right)^{j+1})$$

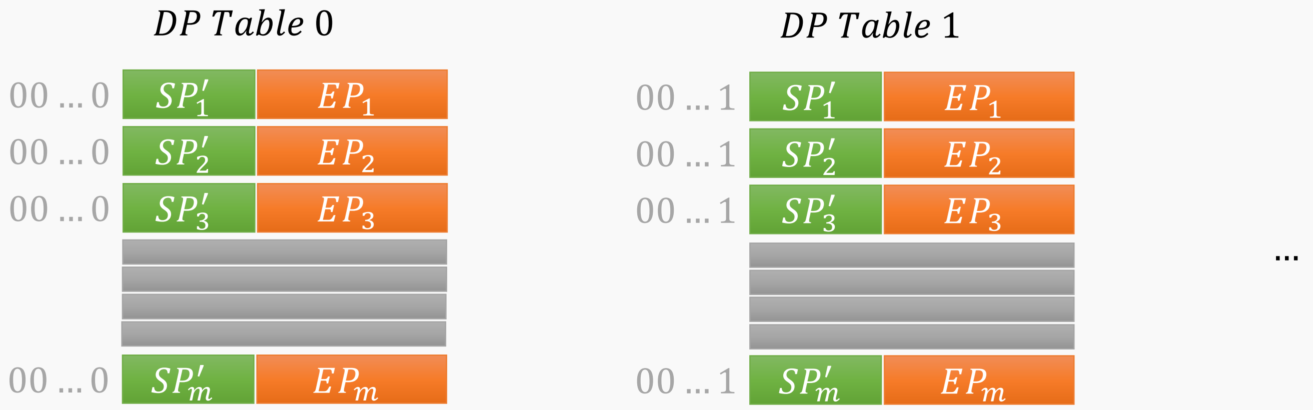
Hellman recommended setting $m = t = l = N^{1/3}$, The total storage space required for the precomputation phase of the Hellman trade-off is $M = ml$. The time complexity of the online phase can be represented as $T = tl$. resulting in a balanced trade-off curve with the equation:

$$TM^2 = N^2$$

Time-memory trade-off

- **DP Trade-Off**

Distinguishing Point (DP) method [7] in 1982. The DP trade-off involves introducing a discernible attribute into the key space with a fixed probability of 1/t. Typically, this attribute is easy to detect, such as setting the first logt bits of an element to 0.

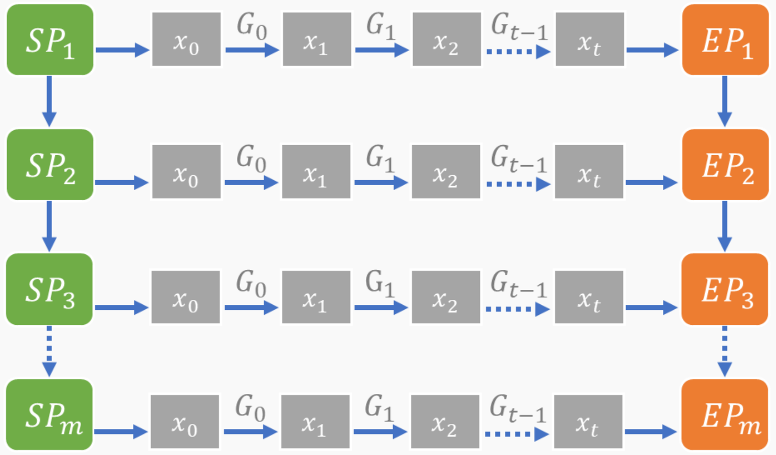

$DP\ Table\ 0$
$00 \ldots 0$ SP'_1 EP_1
$00 \ldots 0$ SP'_2 EP_2
$00 \ldots 0$ SP'_3 EP_3
$00 \ldots 0$ SP'_m EP_m
$DP\ Table\ 1$
$00 \ldots 1$ SP'_1 EP_1
$00 \ldots 1$ SP'_2 EP_2
$00 \ldots 1$ SP'_3 EP_3
$00 \ldots 1$ SP'_m EP_m
...

Precomputation phase, the average length of DP chains is t. Consequently, the total number of nodes covered by the m DP chains is mt.. Online phase, an average of t iterations is required to locate a DP point. Thus, the time complexity $T = tl$. Combining this with the size of the precomputed table $M = ml$, it can be deduced that the trade-off curve for the DP trade-off follows

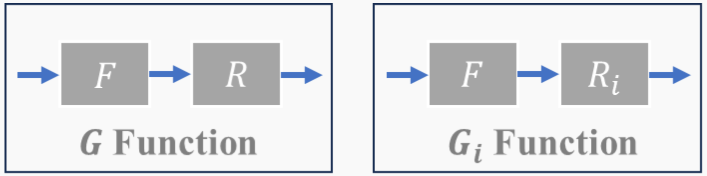
Time-memory trade-off

- **Rainbow Trade-Off**

The rainbow trade-off which was proposed by Philippe Oechslin in 2003 builds upon the foundation of the Hellman trade-off but introduces improvements. It applies different $G_i\ (1 \le i \le t)$ functions to each column of the precomputed rainbow matrix.



$Schematic\ of\ Rainbow\ Trade-off$



$Comparison\ of\ Reduce\ Function$
$with\ Hellman\ Trade-off$

Time-memory trade-off

- **Rainbow Trade-Off**

In terms of a singular table, the success rate of a rainbow trade-off is:

$$P_{single} = 1 - \prod_{i=0}^{t-1} (1 - \frac{m_i}{N})$$

When t is the length of a chain, $m_0 = m$ and $m_{i+1} = N(1 - \exp(-mi/N))(0 \leq i \leq t-1)$， assuming the existence of l tables, the success rate of a rainbow trade-off is:

$$P_{success} = 1 - \prod_{i=0}^{t-1} \left(1 - \frac{m_i}{N}\right)^l$$

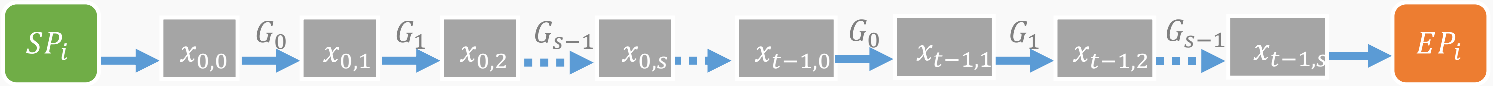
Oechslin suggested setting $m = N^{2/3}$, $l = 1$, and $t = N^{1/3}$. Therefore, in the pre-computation phase of the rainbow trade-off, the required space is $M = ml$, the number of iterations needed during the attack phase is $T = (t^2)/2$, so the trade-off curve for the rainbow trade-off is:

$$TM^2 = N^2/2$$

Time-memory trade-off

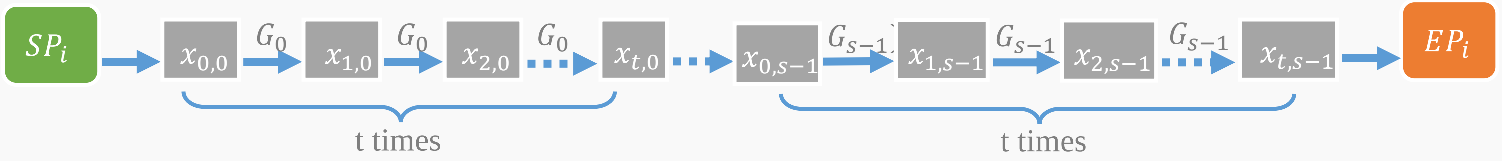
- **Other Rainbow Trade-Off**
  - Thin-rainbow trade-off

    The concept of a "thin-rainbow trade-off " involves reducing the number of distinct G_i functions to s and cycling through these s G_i functions in a periodic manner.



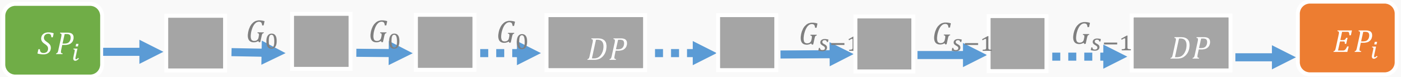
  - Thick-rainbow trade-off

    In this method, the thick-rainbow trade-off applies a different Gi function after each t repetition of a single Gi function.



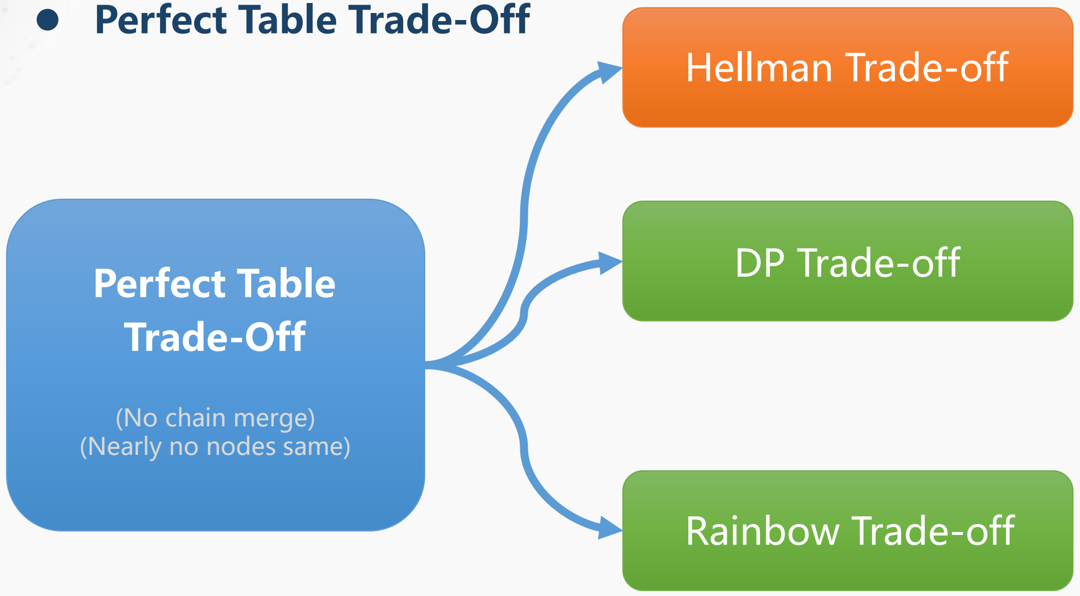
  - Fuzzy-rainbow trade-off

    This method enhances rainbow-based attacks by combining the DP trade-off with the thick-rainbow scheme.



Time-memory trade-off

- **Perfect Table Trade-Off**



Hellman precomputation table requires extensive checks, making its **cost prohibitively high** and **impractical**.

If a merge is detected within the current computation chain, the approach involves retaining the longer chain from multiple merged chains until m precomputed chains are generated.

This trade-off enables the creation of a perfect rainbow table without merges. However, in rainbow tables, the same node between different columns could be retained.

Note: Don't neglect the additional cost associated with generating perfect tables.

04

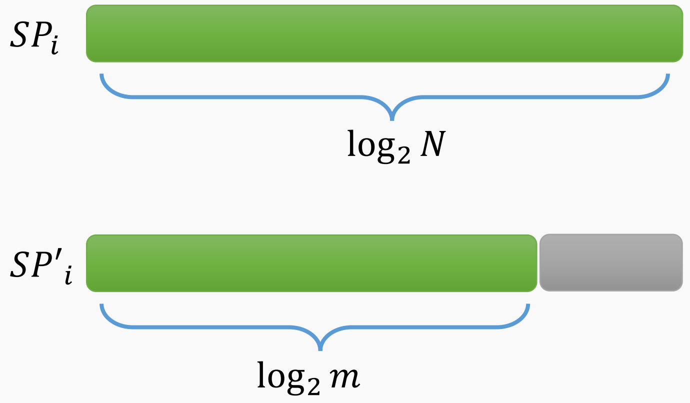
PART

Storage Optimization

Storage Optimization

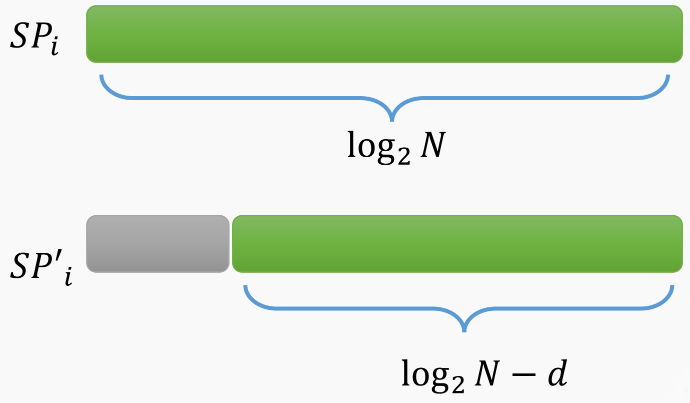
Consecutive Starting Points

A practical method of choosing starting points is to use consecutive integers [1]. The integers 0 through m − 1 will work for any (non-perfect) table.



DP Definition

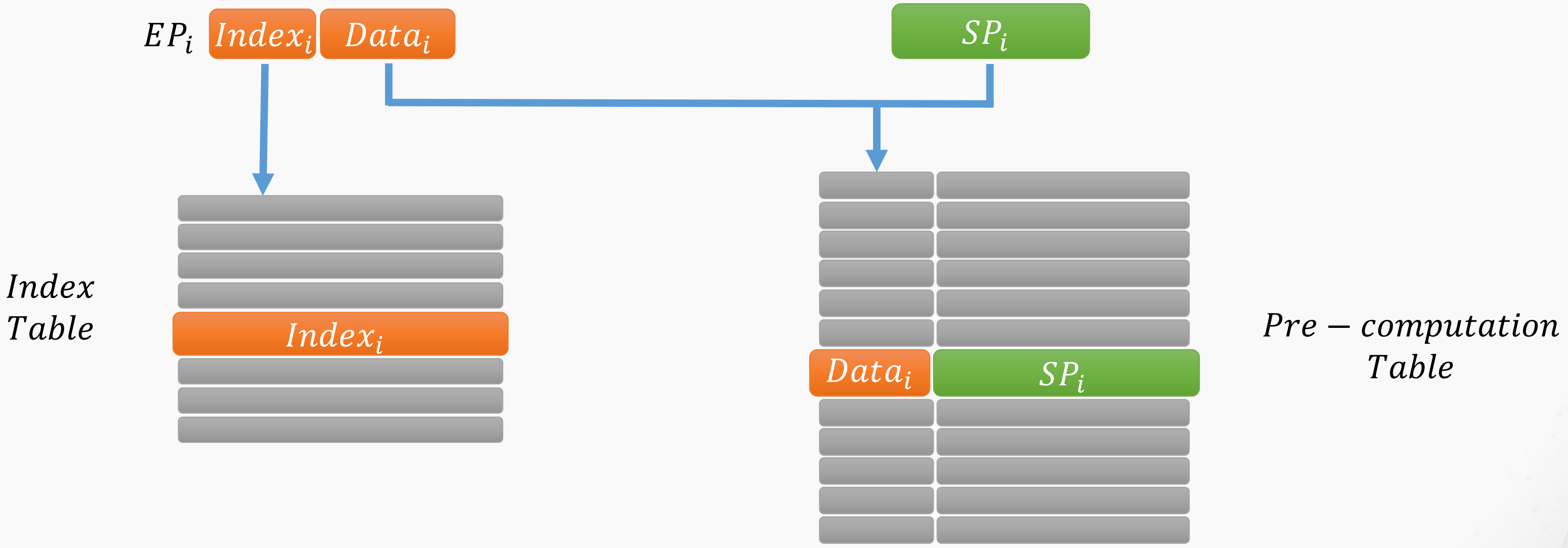
if the first d bits are defined as 0 for DP points, then eliminating these d bits when storing start point values would have no impact on the overall computational efficiency.



Storage Optimization

- **Index Table**

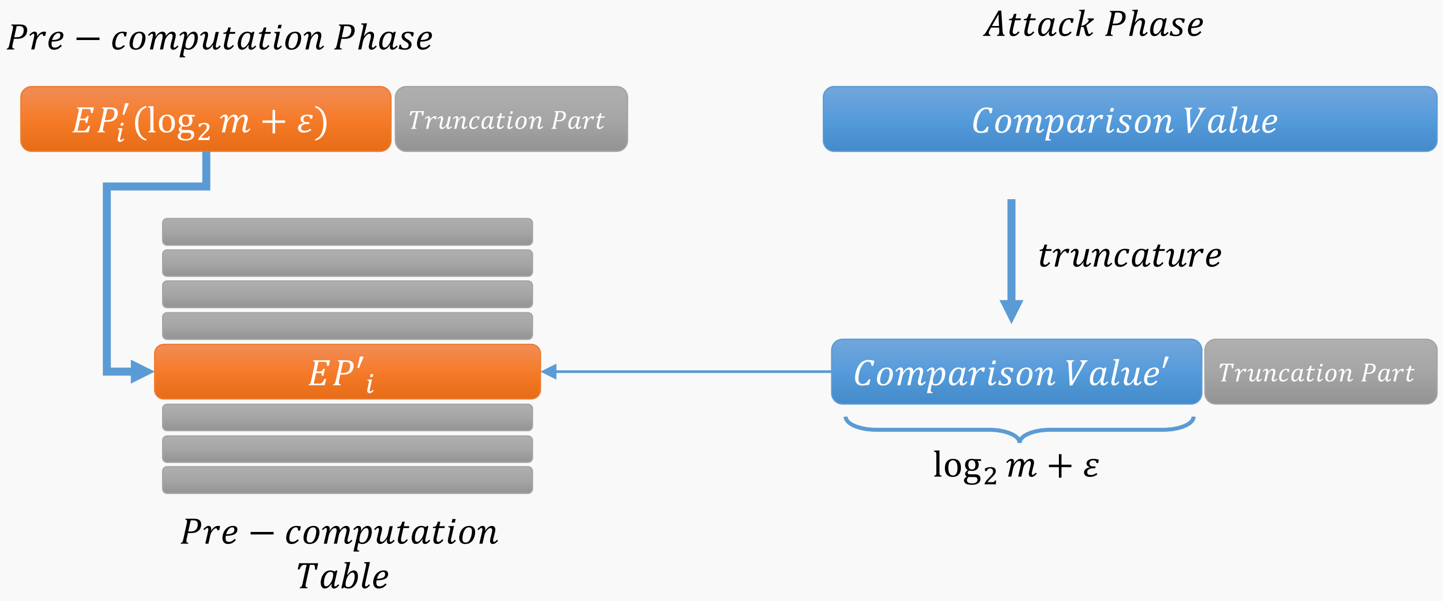
It dissects the endpoint into an **index part** and a **data part**. The data part is stored in the pre-computation table alongside the starting point, while the index part is retained within the index table.



Storage Optimization

- **Ending Point Truncation**

In the pre-computation phase, when truncating the endpoints in the table entries, it is essential to retain enough bits (slightly exceeding "$\log_2 m$") to uniquely identify each pre-computed chain. During the online phase, when conducting table lookups, the subject of the search will be truncated to the same length before being compared with the entries in the precomputation table.



05

PART

Implementation Platform

Implementation Platform

FPGA

The most distinctive feature of Field-Programmable Gate Arrays (FPGAs) lies in their configurational flexibility. Through programming, their application scenarios can be altered at will, greatly reducing the development time and costs of cryptographic attack accelerators.

GPU

GPU significantly enhances the efficiency of rainbow chain generation. Within the pre-computation phase of the rainbow trade-off, the calculation of rainbow chains is distributed to each GPU thread. This significantly enhances the efficiency of rainbow chain generation. However, the high performance of GPUs comes at the cost of high energy consumption.

CPU-GPU Heterogeneous Platform

Within the GPU, there exists a multitude of processing cores capable of simultaneously executing thousands of computational tasks. The CPU-GPU architecture is particularly well-suited for processing scenarios demanding high performance, making it an apt platform for password recovery.

CPU-FPGA Heterogeneous Platform

CPU-FPGA heterogeneous devices are also noteworthy heterogeneous platforms in the industry. The FPGA is responsible for accelerating data processing, while the CPU handles other minor computational tasks such as data interaction.

06

PART

Summary

Summary

- **Optimization Direction**
  - Trade-Off Algorithm Selection
  - Storage Optimization Selection
  - Implementation Optimization Selection
- **Optimization Principle**
  - Multi Strategy Cooperation
  - Success Rate Requirement
  - Budget Constraints

THANKS