International Journal of Applied Engineering and Technology ISSN: 2277-212X
An Open Access, Online International Journal Available at http://www.cibtech.org/jet.htm
2022, Vol.12, pp.66-73/Sathya
Research Article

A NOVEL APPROACH FOR WIRE AND CACHE TAPPING IN ARTIFICIAL INTELLIGENT BASED TO ACHIEVE FAULT TOLERENCE MECHANISM

***R. Sathya**
Master of Engineering in Computer Science & Engineering,
A.R.J College of Engineering and Technology, Mannargudi,
**Author for Correspondence: sathyarengaraj028@gmail.com*

ABSTRACT
The concept of cache-tapping is introduced into the information theoretic models of coded caching in this paper. The wire-tap channel II is introduced in the presence of multiple receivers with fixed-size cache memories and an adversary who selects symbols to tap into from cache placement and/or delivery. The legitimate terminals have no idea whether they are tapped for placement, delivery, or both, or what positions they are tapped in. Only the total size of the tapped set is known. The strong secrecy capacity– the maximum achievable file rate while keeping the overall library strongly secure– is identified for two receivers and two files. When the library contains more than two files, lower and upper bounds on the strong secrecy file rate are calculated. Achievability is dependent on a code design that incorporates wiretap coding, security embedding codes, one-time pad keys, and coded caching. The converse for the two-files case is established by a genie-aided upper bound, in which the transmitter is provided with user demands prior to placement. When there are more than two files, the upper bound is built using three successive channel transformations. Our findings provide provable security guarantees against a powerful adversary who optimises its tapping across both communication phases in a cache-aided system.

INTRODUCTION
Caching attempts to reduce network traffic congestion by proactively storing partial contents in end-user cache memory during off-peak hours, resulting in a local caching benefit. The seminal work in demonstrated that careful design of cache contents in a multi-receiver situation allows the transmitter to provide delivery messages that are useful to multiple users at the same time, delivering an additional benefit known as the global caching gain. This increase is dependent on the aggregate cache memory of the network and demonstrates the ability to code over delivery transmission and/or cache contents. Coded caching is a technique that was first introduced in for the single-stream bottleneck broadcast channel (BC) in order to deliver cacheable content to a large numberof users at the same time. Initially, this technique wasused in a scenario in which a single-antenna transmitter has access to a library of N files and serves(via a single bottleneck link) K receivers, each with a cache the size of M files.
The procedure included a novel cache placement method and a subsequent delivery phase in which each user simultaneously requests one library file, while the transmitter uses cache-dependent coded multicasting to deliver independently requested content to many users at the same time. Numerous studies have been conducted on coded caching under various modelling assumptions and network configurations, including decentralised caching, non-uniform demands, more users than files, heterogeneous cache sizes, improved bounds, hierarchical caching, interference networks, combination networks, device-to-device communication, coded placement, and delivery over noisy channels.The wiretap channel II (WTC-II) model gives an adversary partial access to valid communication in the form of a temporal fraction during which the adversary can listen in. The model, in particular, assumes a noiseless legal channel and an adversary who selects a fixed-size subset of the sent symbols to observe noiselessly. According to references, despite the ability to select the locations ofthe tapped symbols, the adversary can be made no more powerful than nature through proper coding, i.e., the WTC-secrecy II's capacity is identical to that of a binary erasure wire tapper channel with the same fraction of erasures. The WTC-II has been enhanced to include a discrete memory less–noisy–legitimate channel, as well as inner and outer boundaries for its capacity-equivocation region. The ability of this model to maintain secrecy has been acknowledged. We proposed a generalised wiretap model that includes both the classical wiretap and wiretap II channels as special circumstances. This generalised model has been applied to networks with a large number of transmitters and receivers.
A caching problem has two phases: the prefetching phase, in which a portion of files is stored in caches, and the delivery phase, in which the server fulfils the users' requests. The prefetching schemecould be centralised, in which

case the central server supervises the caching process, or decentralised, in which case the files are randomly cached by the users without the involvement of the central server. In comparison to centralised schemes, which require knowledge of the number of active users during the prefetching phase, the decentralised approach is more practical, especially in large networks.
The resilience of stochastic wiretap encoding against a type II attacker who can choose where to tap is the common thread in all of these cases. The genuine terminals are only aware of the total size of the tapped set. The difficulty in caching stems from the fact that the transmitter, who has access to a file library, has no knowledge of future end-user requirements when constructing their cache contents. When it comes to security against an external enemy, this is still true. Furthermore, the adversary may tap into cache placement, delivery, or both, and the location of the tapping is unknown to legitimate terminals under the novel paradigm proposed in this research. We show that even under these adverse conditions, strong secrecy guarantees that are invariant to the locations of the tapped symbols changing between cache insertion, delivery, or both phases can be guaranteed.

RELATED WORK

The system responsible for error correction in the event of a large number of RAID failures. For a long time, there have been error-correction systems. However, the method of distributing data across multiple storage devices in order to provide high-bandwidth input and output, as well as the use of one or more error-correcting devices for failure recovery, is relatively new. It rose to prominence with "Redundant Arrays of Inexpensive Disks" (RAID), which combines batteries of tiny, low-cost discs with large storage capacity. The method has since been used to develop high-reliability, high-bandwidth multicomputer and network file systems, as well as fast distributed check pointing systems. Tolerating concurrent failures with exact checksum devices is a general method. Its maximum distance separable (MDS) property ensures an optimal check word-to- data word ratio in terms of tolerable failures. It has a low computational cost, a high reliability and bandwidth, and a low number of xor operations.

MATERIALS AND METHODS

Approach:

Erasure and network coding concepts have recently been used in various types of networks to improve system throughput and reliability. Erasure codes, when used in distributed storage systems such as P2P or RAID-based systems, can improve data reliability and persistence by implementing Rabin's Information Dispersal Algorithm (IDA). This scheme is used in a large number of proposals for P2P systems.
Furthermore, another advantage of erasure codes in this context is that they reduce average downloading time by increasing data diversity in the network. Erasure codes are also used in multimedia transmissions where packet losses cannot be recovered using traditional retransmissions due to real-time constraints. We reduce the number of data copies from three to one using erasure coding. As a result, when a block is written, we only assign one Data Node to the Client. After writing the data blocks to the assigned Data Nodes, we save a copy of the data to the Data Queue to be encoded. While there are k data blocks ready in Data- Queue, we encode them using the CaCo schedule. We get coding blocks after encoding and add them to the Parity Queue. Then, for the coding blocks, we allocate some Data Nodes. It should be noted that those data blocks and coding blocks should be stored on separate data nodes. Otherwise, a single node failure could result in multiple faults in the same group of blocks.

Algorithm: Erasure Coding Approach (ECA)

Step 1: The Client sends a write request to the NameNode.

Step 2: The Name Node assigns some Data Nodes to the Client.

Step 3: Write the data blocks into Data Nodes.

Step 4: Make a duplicate of data and put it into DataQueue.

Step 5: Encrypt data with the schedule selected by CaCo.

Step 6: Write the coding blocks into Data Nodes.

Step 7: Data Encrypting finishes.

Step 8: Eliminate the copies of data from Data Queue.

Architectural View:

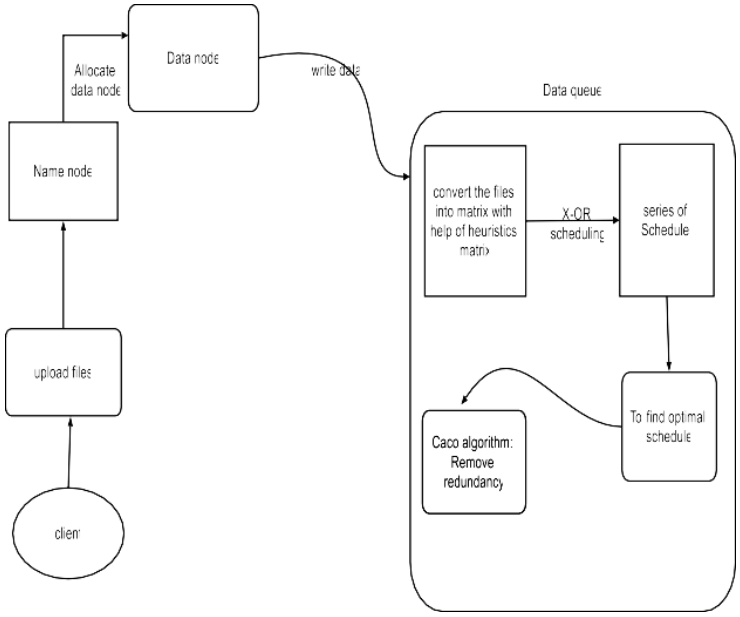

Figure 1: Erasure Coding approach

Approach:

For cloud storage systems, the Cauchy Coding approach is used. The decentralised shared caching problem is divided into three phases: prefetching, user to cache association, and delivery. This algorithm describes the prefetching phase as well as the user to cache association phase. The prefetching phase is identical. Similar methods were used to demonstrate the optimality of the traditional decentralised caching problem. A coded caching problem's delivery phase corresponds to an index coding problem. The prefetching phase's cached contents serve as side information. Each file is assumed to be of F bits in the shared caching problem. Each of these bits is regarded as a separate message. When a data or parity chunk becomes unavailable due to device failure, a decoding operation is initiated to restore the missing chunk. The input for recovering a parity chunk is w row vectors from the generator matrix plus all data chunks. To recover a data chunk, an inverse matrix is generated from the generator matrix (which can be done offline in advance), and k alive chunks are fed into the inverse matrix with w row vectors. In terms of data access pattern and computation, the encoding and decoding operations are essentially the same. As a result, the term coding refers to both encoding and decoding.

Algorithm: S-QuaLShare Algorithm (SQLA)

Inputs: Nil(b$\hat{}$$\vec{\imath}\imath$;Lil)

Output: Gj*

1: Appeal Misbehavior Detection algorithm for singleuser
share as shown in Algorithm 22: **if** Ui(H + 1) -U^{init}_{i} < 0 **then** 3: **if** user not assign **then**
4: Bandwidth Sharing required.5: **end if**
6: **else**
7: **if** (max(dj; de)≤ 1**then**
8: Gi shares the bid p with the member9: **if** any Gj_ accepts the price **then** 10: Assign the user to Gj_
11: **else**
12: **if** multiple gateways accept the price **then** 13: Assign the user to Gj with minimum de j 14: Gi pays #i to the assignee gateway
15: **else**
16: Revise the bid price p(H + 1) = p(H) + Δp17: Goto Step 2
18: **end if**
19: **end if**
20: **end if**
21: **end if**

Architectural View:

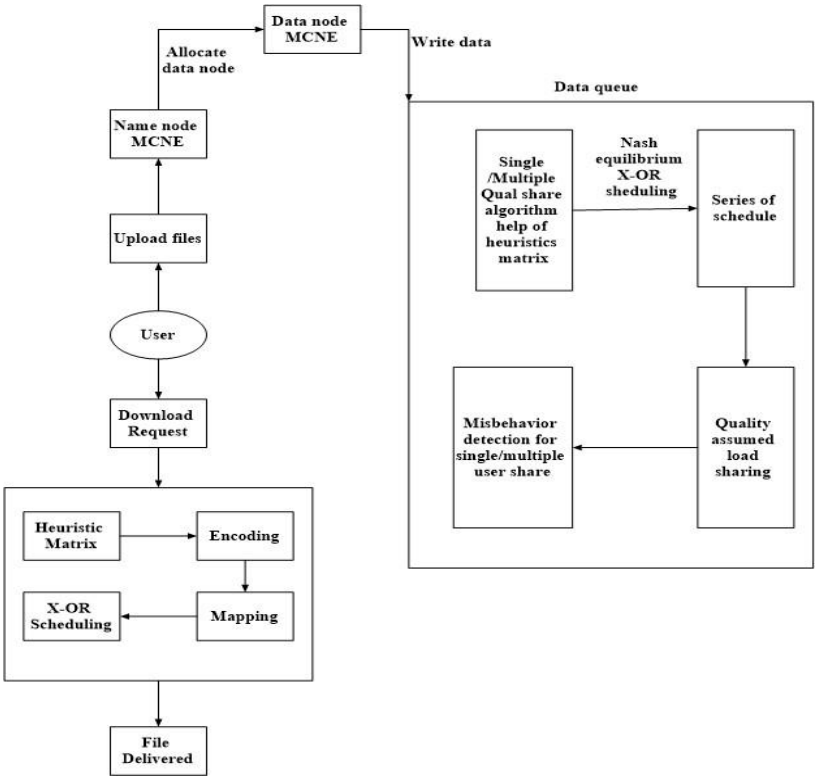

Figure 2: S-QuaLShare Approach

Approach:

We begin by providing a high-level overviewof G-CRS and defining our terminologies. Array codes are error and burst correcting codes that have been widely used to improve data reliability in communication and storage systems. So after that, we examine the potential drawbacks of this fundamental design and present a set of optimization strategies for accelerating CRS coding on GPUs. Following that, we present a baseline implementation of CRS coding on GPUs that is directly compatible with the CPU version. All of the optimization strategies described inthis section are used to implement our G-CRS. The bitmatrix stores the bottom mw rows of the CRS code's generator matrix.

The input data is divided into k equal-sized data chunks, and the output includes m equal-sized parity chunks. On the target hardware platform, we uses to represent the number of bytes of a long data type,

i.e., s = sizeof (long). A packet is defined as s consecutive bytes. A data block is defined as w consecutive packets, where w is the CRC parameter and must be greater than log2(k + m). N denotes the number of data blocks in a chunk. CaCo can obtain anoptimal coding scheme in a reasonable amount of timethanks to our efficient decoding approach.

Algorithm: G-CRS Workflow(G-CRS)1: Input: k, m, w, bitmatrix, dataSize
2: Calculate round from k, m, w, dataSize3: Create bitmatrix
4: Assign GPU memory 5: Copy bitmatrix to GPU6: for each I [1, round] do
7: Copy k data chunks of the i-th round to GPU
8: Launch the encoding kernel function to generate mcoding chunks
9: Copy back m parity chunks of the i-th round to mainmemory
10: end for
11: Free memory resources

Architectural View:

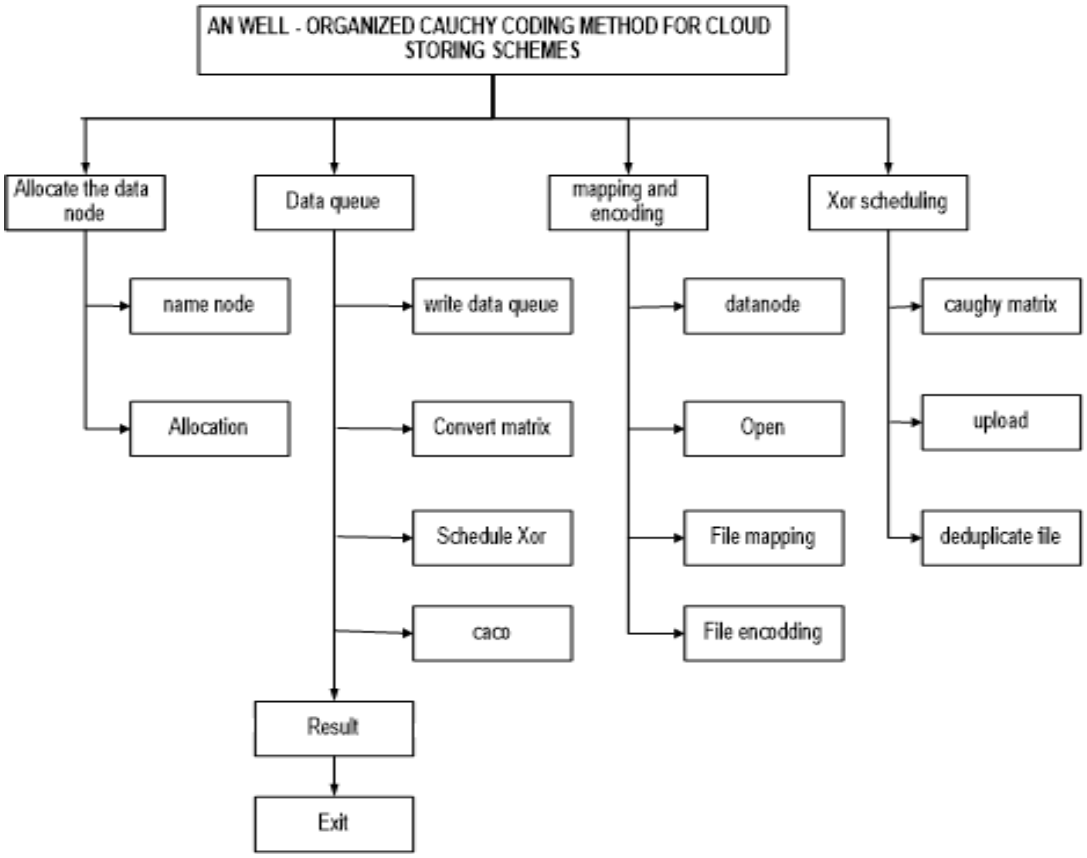

Figure 3: Workflow of G-CRS

Proposed Approach:

An efficient Cauchy Coding method is used for cloud storage systems. Cauchy Coding uses Cauchy matrix algorithms to generate a matrix set. Cauchy Coding then generates a series of schedules using XOR scheduling heuristics for each matrix in this collection and selects the smallest one. As a result, each matrix is associated with a schedule that is locally optimal. Finally, Cauchy Coding selects the globally optimal schedule from a set of locally optimal schedules. This globally optimal schedule, along with its corresponding matrix, will be saved and used during data encoding and decoding.

Cauchy Coding, by utilising all available matrix and schedule heuristics, is capable of identifying an optimal coding scheme for any given redundancy configuration within the capabilities of the current state of the art. In this architecture, the user must first login to the system, so we must register withit. In the registration form, the user enters his or her personal information. After entering the information, the user is able to process the system. The administrator verifies the information provided by the user; if the username and password are correct, the administrator grants access to the data. If it is incorrect(or invalid), the administrator denies the user access tothe data. The main data queue process employs the homographic hash algorithm to determine the series ofschedules. Find the assumed load sharing quality andthe misbehaviour detection. This process can be used to upload files by using four processes: wiretap loading, security embedded codes, coded caching, andone-time pad keys. Following that, the user can send adownload request to download the processed file. Then proceed to the main data queue process. The file has been downloaded twice by the two processes. Thefile can then be delivered to the user.

Algorithm: Homomorphic Hash Algorithm (HHA)
Step 1: Key Generation
Step 2: Select two large primes p and q, such that p ≠q.
Step 3: N = p* q.
Step 4: ϕ (n) = (p-1)*(q-1), where ϕ is Euler's totientfunction.
Step 5: Select an integer e such that 1 <e< ϕ(n) andgcd (e,ϕ(n))=1 (comprime).
Step 6: d = e-1 mod ϕ(n)
Step 7: Public key ← (e,n)
Step 8: Private key ← d
Step 9: Encryption
Step 10: C = me mod n
Step 11: Decryption
Step 12: m = cd mod n

Architectural View:

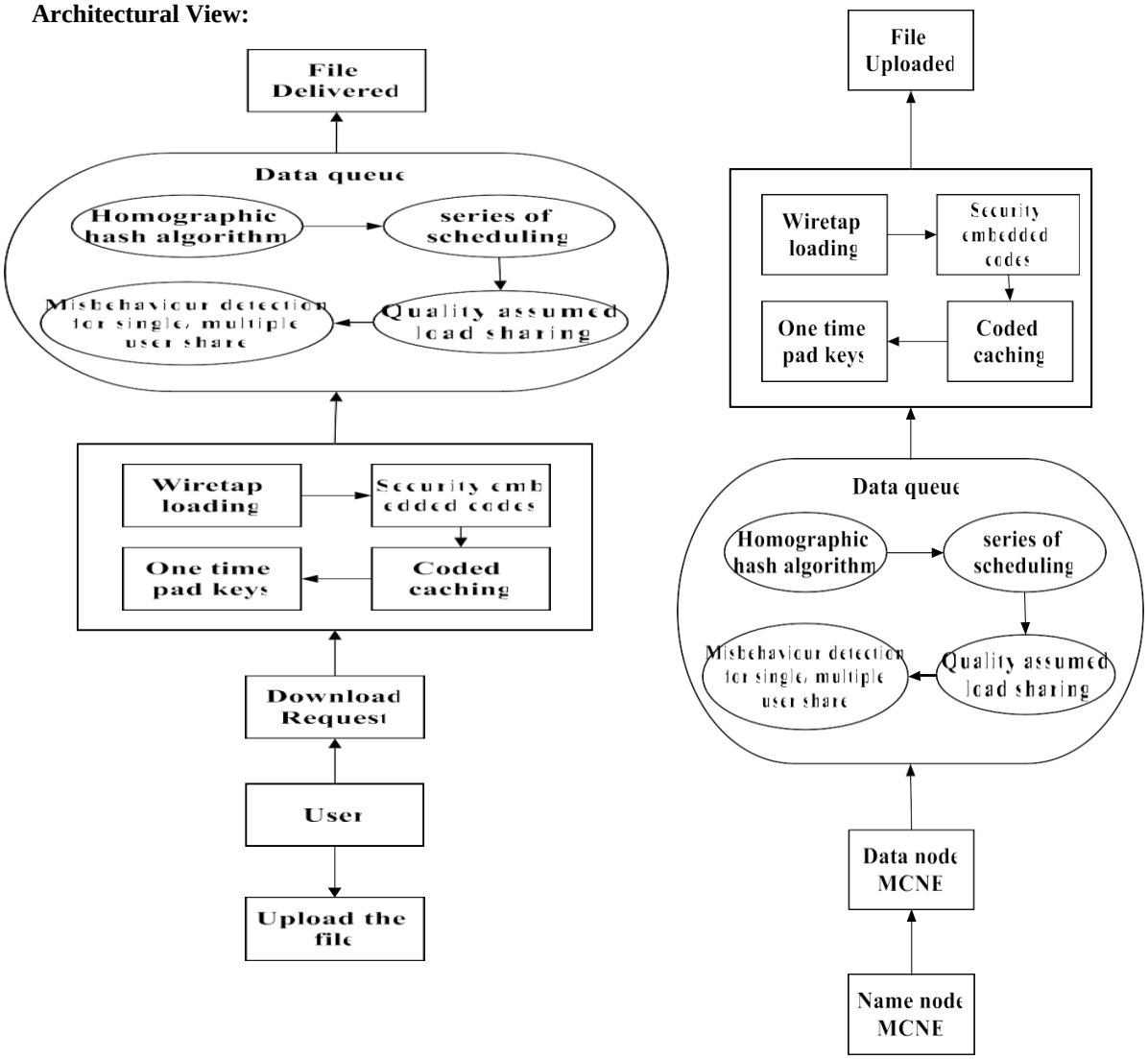

Figure 4: Homomorphic Hash Approach

RESULTS

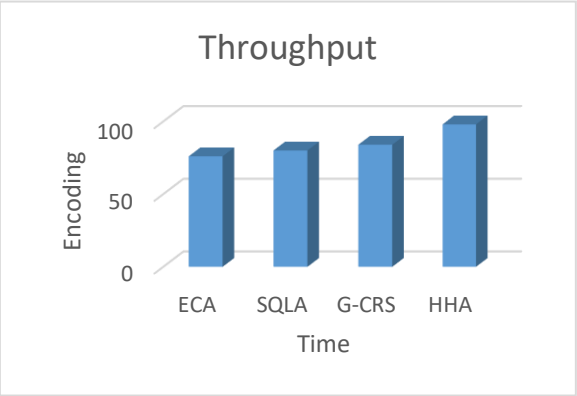

Figure 5: Throughput

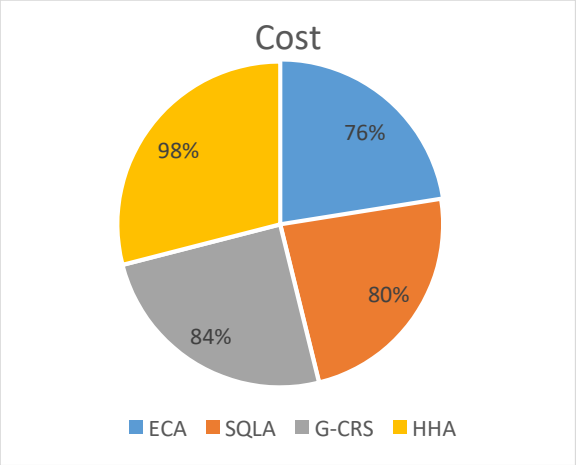

Figure 6: Cost

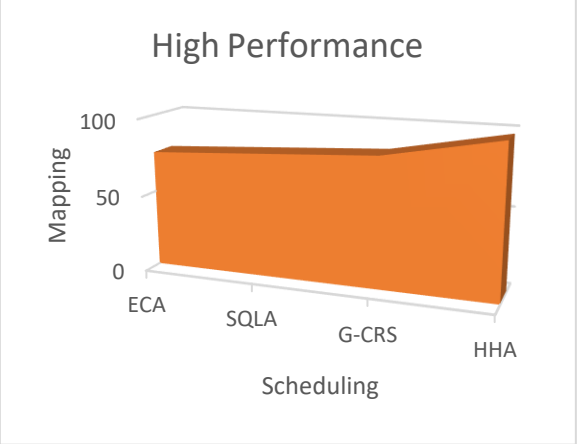

Figure 7: High Performance

CONCLUSION

We introduced the caching broadcast channelusing a wire and a cache tapping adversary of type II. Each receiver has a fixed-size cache memory, and the adversary can access a subset of the transmitted symbols during cache placement, delivery, or both. The legitimate terminals are unaware of the fractions of the tapped symbols in each phase, as well as their positions. Only the total size of the tapped set isknown. We discovered this model's strong secrecy capacity– the maximum achievable file rate whilekeeping the overall library secure– when the transmitter's library contains two files. When the transmitter has more than two files in its library, we calculated lower and upper bounds for the strong secrecy file rate. We created an attainable scheme that combines wiretap coding, security embedding codes, one-time pad keys, and coded caching techniques.

REFERENCES

1. Device-to-Device Secure Coded Caching Ahmed A.Zewail, Aylin Yener year: 2020
2. The Caching Broadcast Channel With A Wire AndCache Tapping Adversary Of Type Ii Mohamed Nafea, Aylin Yener Year :2018
3. Noisy Broadcast Networks With Receiver Caching Shirin Saeedi Bidokhti, Michèle Wigger, Roy Timo year : 2018
4. The Caching Broadcast Channel with a Wire and Cache Tapping Adversary of Type II: Multiple Library Files Mohamed Nafea, Aylin Yener year : 2018
5. Generalizing Multiple Access Wiretap and Wiretap II Channel Models: Achievable Rates and Cost of Strong Secrecy Mohamed Nafea, Aylin Yener year: 2019
6. Coded Caching in the Presence of a Wire and a Cache Tapping Adversary of Type II: Mohamed Nafea, Aylin Yener_2021
7. A Coded Caching Scheme with Linear Sub- packetization and its Application to Multi-Access Coded Caching: Anjana Ambika Mahesh, B. Sundar Rajan_2021
8. Linear Network Coded Wireless Caching in Cloud Radio Access Network: Long Shi, Kui Cai, Tao Yang, _2021
9. On Coded Caching With Private Demands: Kai Wan, Giuseppe Caire_2021
10. Cache Compression with Golomb-Rice Code and Quantization for Convolutional Neural Networks: Seung-Hwan Bae, Hyuk-Jae Lee_2021