IRAQI Academic Scientific Journals

Al-Rafidain Engineering Journal (AREJ) Vol.31, No.1, March 2026, pp. 69-78

Discrete Markov Chain Modeling for Efficient Key Processing in Anonymous Networks

Hothayfa Rabea Agha* huthaifa.mohammed@uomosul.edu.iq

✉Tanygin Maxim Olegovich** tanygin@yandex.ru

Salah Abdulghani Alabady* salah.abdulghani@uomosul.edu.iq

*Computer Engineering Department, College of Engineering, University of Mosul, Mosul, Iraq
**Department of Information Security, Faculty of Fundamental and Applied Informatics, Southwest State University.Kursk, The Russian Federation.

Received: July 25th, 2025 Received in revised form: September 13th, 2025 Accepted: November 13th, 2025

ABSTRACT

Anonymous networks are designed for confidentiality, while simultaneously keeping the sender's identity hidden from all nodes except the designated receiver. The decryption process requires the actual receiver to try out all possible keys stored in memory, thereby making computation very time-consuming, especially in noisy conditions where errors and hash collisions occur. This paper proposes a probabilistic algorithm to reduce the number of keys that must be processed, without compromising accuracy. The algorithm models the decryption process as a discrete Markov chain the Bernoulli formula in all its facets to calculate the probability of correct key detection and thereby determine an optimal stopping condition. Simulation results with 15 sources and an error probability of 0.15 exhibited up to a 14% reduction in the number of key trials required compared to brute-force search, with expected higher efficiencies for larger network sizes. This research contribution introduces a lightweight, scalable model that reduces computational cost, saves energy, and extends node lifetime. These findings stress the importance of providing a practical means for enhancing the performance of anonymous communication systems, especially while working under a resource-constrained environment such as the IoT and decentralized networks.

Keywords:

Anonymous networks, discrete Markov chains, decryption, Bernoulli, hash collision, error probability.

https://jamh.uomosul.edu.iq/index.php/rengj
Email: alrafidain_engjournal3@uomosul.edu.iq

==========

1. INTRODUCTION

The receiver must identify the proper source when decoding messages transmitted by several sources. Several potential source identifiers remove this stored in memory are checked sequentially throughout the decoding process. In the most basic scenario, there are N+1 nodes in the system, and N sources produce messages for a single recipient. The system must carry out N consecutive decoding operations, each employing a distinct stored identifier from the set $\Omega = \{\Omega_1, \Omega_2, \ldots, \Omega N\}$, to determine the proper source of a received message.

Computational efficiency is one of the main issues in such a system, particularly when working with battery-operated, low-power computing devices with constrained processing capability[1,2]. Multiple decoding attempts are necessary for every received message, which might result in a very high overall computing cost. The vast number of nodes in a peer-to-peer system can have a substantial effect on the receiver's power consumption and message processing performance, even when utilizing lightweight cryptographic techniques. The system can include feedback mechanisms in the message-processing workflow to reduce the number of decoding processes. The implication is that previously decoded messages stored in a queue can affect how new messages are processed. By using previously decoded data, the system can reduce unnecessary computations and maximize decoding choices. The dynamic structure-building process is another essential component.

When a message is successfully decoded, its descriptor (metadata) is added to a tree-based structure associated with its source. This descriptor is stored under two possible conditions:

If the message was generated by a known source, its descriptor is added in sequence after the

previous message from that source. In the event of a hash collision, which occurs when the message is erroneously linked to the incorrect source, the descriptor is still added to the tree, but it is given a different structure. Mathematically speaking, the decoding procedure is modeled by a discrete Markov chain. Transitions between states occur when a new source identifier is examined, with each state in this paradigm corresponding to a distinct stage of decoding. There are three parameters that characterize the system's current state:

i (whether the correct source has been checked so far, taking values {0,1})

k (the number of source identifiers tested at a given moment, from 0 to N)

j (the number of incorrect sources that have mistakenly matched the current message, ranging from 0 to N-1)

This independent behavior in transitioning between states at each step of the process is the premise of the Markov chain model where the probability of going from one state to another is independent of how we arrived at that state. The total number of tested source IDs plus the initial state is N+1, which is the number of discrete time steps in the model. The system only changes states if a decoding operation is performed, and the sum of all transition probabilities from a given state of the system always has to equal 1. The Markov chain that represents this key-checking process is illustrated in Figure 1.

Five basic equations modify how likely one decoding step is to lead to the next step; those define the transition probabilities from state to state. So, the algorithms consider how a message may be appropriately decoded or how an error may occur due to hash they also consider the options of continuing to decode until the correct source is reached or how many hashes would be passed before realizing a match. The model ensures that a message gets assigned to a source at the end, by making sure that with every step, the system converges to a conclusion.

Framing message decoding as a probabilistic Markov process enables the system to accurately consider errors and maximize likelihood functions, eliminating redundant computations to improve overall system use. Applications like sensor networks, cryptography systems, and decentralized communication protocols, Require signals to be decoded efficiently and accurately [3,4].

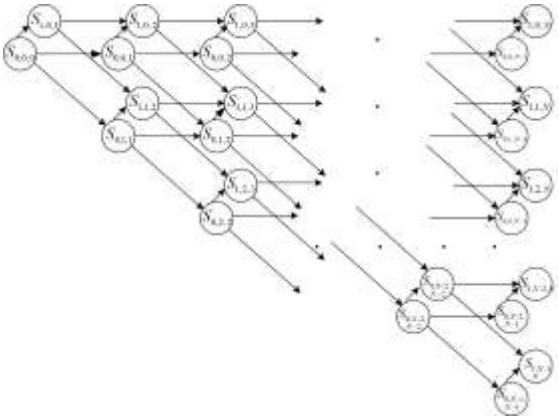

Fig 1 A graph of a Markov process that simulates the processing of keys.

The probabilities of transitions between states are:

$$
\begin{aligned}
&p\left(S_{0,j,k} \to S_{1,j,k+1}\right) = \frac{1}{N-k}, \\
&p\left(S_{0,j,k} \to S_{0,j+1,k+1}\right) = \frac{N-1-k}{N-k} p_{\text{err}}, \qquad \text{..... 1} \\
&p\left(S_{1,j,k} \to S_{1,j+1,k+1}\right) = p_{\text{err}}, \\
&p\left(S_{0,j,k} \to S_{0,j,k+1}\right) = \frac{N-1-k}{N-k}\left(1-p_{\text{err}}\right), \\
&p\left(S_{1,j,k} \to S_{1,j,k+1}\right) = 1-p_{\text{err}}.
\end{aligned}
$$

Transitions between other pairs of states are Impossible; the sum of the outgoing probabilities of transition from each state is equal to 1, that is, at each next moment, the system is guaranteed to change its state.

Using the discrete Markov chain and Bernoulli formula to predict the probability of finding j correct keys among k processed keys in decryption is the original this approach calculates the required number of found successes before halting the key searching process, then selecting those that exceed the predefined threshold. As IoT devices and anonymous network applications dominate the tachnology market , we need to ensure they operate with extreme performance, such as reducing processing, and conserving energy to extend the lifetime of the node.

However, more emphasis has been placed on various areas such as routing efficiency, optimization of encryption techniques, and resistance to traffic analysis. The decryption procedures have not properly modeledhence the decryption process has been ignored [5,6]. Particularly, no study has formally modeled the decryption process using a probabilistic approach, specifically by employing discrete Markov chains combined with Bernoulli processes, in order to reduce the number of key trials during the act of decryption.

The goal of this paper is to propose a lightweight probabilistic algorithm that models key decryption using discrete Markov chains and

Bernoulli formulas, thereby reducing the number of keys treated. The contribution of this particular research is that it provides a scalable solution to minimize computational costs, save energy, and enhance performance in anonymous networks, particularly for IoT and decentralized communication applications.

2. THETHEORETICALBASES

Anonymous networks have evolved significantly since their inception, primarily aimed at ensuring privacy and secure communication. Early implementations, such as Mix Networks and Onion Routing, laid the foundation for modern anonymous communication systems[]. The Tor network, which utilizes multi-layer encryption and relay nodes, remains one of the most widely adopted anonymous networks. Over time, researchers have explored various enhancements to improve anonymity, scalability, and resistance to attacks[1]. The role of anonymous networks has expanded beyond traditional secure browsing. They are now utilized in decentralized finance (DeFi), whistleblowing platforms, censorship resistance, and secure communication for journalists and activists[2]. Recent developments have also seen their application in blockchain-based privacy solutions and dark web marketplaces. Additionally, law enforcement and cybersecurity researchers are studying these networks to understand and mitigate their misuse in cybercrime activities[8]. One of the major challenges in anonymous networks is optimizing performance while maintaining strong anonymity. Several studies have focused on reducing network latency, improving routing algorithms, and enhancing encryption efficiency[5]. Researchers have proposed adaptive relay selection, probabilistic packet forwarding, and traffic obfuscation techniques to reduce the risk of deanonymization while maintaining high throughput[6,7].

Processing overhead remains a key concern, especially in resource-constrained environments. Techniques such as selective encryption, lightweight cryptographic protocols, and parallel processing architectures have been explored to minimize computational costs[3]. Load balancing among relay nodes and predictive caching have also been proposed to optimize bandwidth usage and reduce processing delays[4]. Markov chains and Bernoulli processes have been applied to model and optimize the behavior of anonymous networks[9]. Researchers have utilized Markov decision processes (MDPs) to predict traffic patterns, optimize routing paths, and detect potential vulnerabilities. Bernoulli trials have been employed to assess packet loss probabilities and analyze the effectiveness of probabilistic forwarding techniques[1,10].

Several studies have examined the trade-off between processing load reduction and network security. While minimizing computational requirements can improve efficiency, it may introduce vulnerabilities such as increased susceptibility to traffic analysis attacks. Research has explored threshold-based mechanisms to balance security and performance, ensuring that anonymity guarantees are not compromised while optimizing resource utilization The study of anonymous networks continues to evolve, with a strong focus on optimizing performance and reducing processing overhead. The integration of Markov chains and Bernoulli processes has provided valuable frameworks for analyzing and improving network behavior. Future research should explore the intersection of machine learning and probabilistic modeling to further enhance the efficiency of anonymous communication systems.

3. THE PROPOSED METHODOLOGY

How the algorithm works: At first, working of the algorithm, we have zero key processed (k=0), zero false success j=0, and we have not seen the true success key , which is of course unique (i=0), so we at state in grey in figure 2 that simulates a brute-force of keys. With i=0 in the grey color.

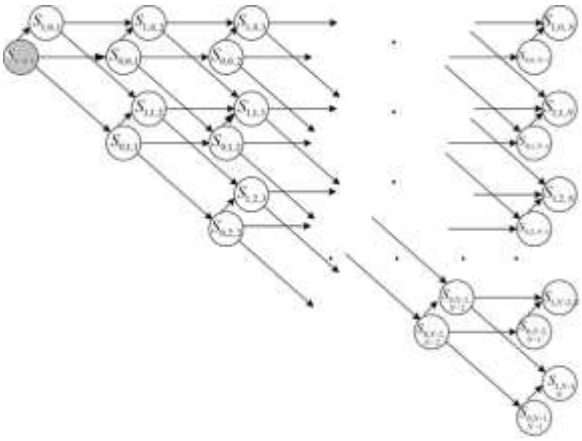

Fig. 2 – $S_{i,j,k}$ graph of a Markov process

Each state with i=0, has 3 possible transitions

I. In the next state,we find the true key , so that i would be changed to 1, processed key counter (k) will be incremented by one (as we processed the next key), and j will remain the same as we did not conduct false success

$$next\ i\ = 0 \rightarrow 1, next\ \ j = \ the\ same, next\ k = current\ k + 1$$

II. In the next state, we will process a false key (real false with no collision) so that the k will be

incremented as we process the next key and j will remain the same as we do not find a false miss match and i will remain the same

$$next\ i\ = the\ same\ previous\ i\ , next\ \ j$$
$$= \ the\ same\ of\ previous\ j, next\ k$$
$$= previous\ k + 1$$

III. In the next state we will conduct a false key. Still, because of collision, the receiver will consider this key to be the correct key, so k will be incremented as we process the next key, i will remain the same because this is not true false to change the value of i, and j will be incremented because we conduct another false success.

$$next\ i\ = the\ same\ previous\ i, next\ j$$
$$= previous\ j + 1, next\ k$$
$$= previous\ k + 1$$

Each state with i=1, has 2 possible transitions

Now, let's consider that we are at the state colored with red color in Figure 3 , which represents the state where we found the correct key for true success.

Each state with i=1 has two possibilities, one of them is that the next state is true false (incorrect key without collisions) , or it will find another successful (which is false because we have a unique true key for successful decryption)

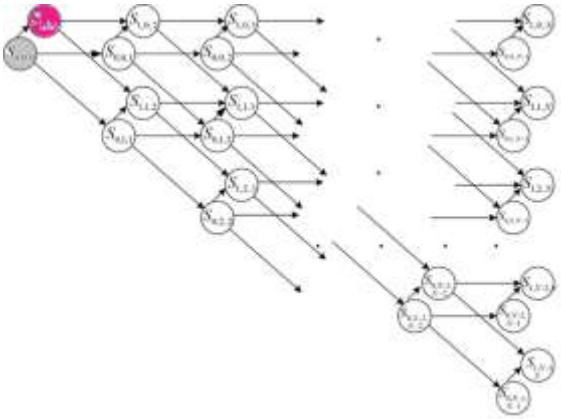

Fig. 3 A graph of a Markov process that simulates a brute-force of keys. With i=1 in red state

IV. In the next state, we will process a false key (a real false with no collision) so that the k will be incremented as we process the next key and j will remain the same as we do not find a false miss match , and i will remain the same

$$next\ i\ = the\ same\ as\ previous\ i, next\ j$$
$$= \ the\ same\ as\ previous\ j, next\ \ k$$
$$= privousk + 1$$

V. In the next state we will conduct a false key. Still, because of collision, the receiver will consider this key to be the correct key, so k will be incremented as we process the next key, i will remain the same because this is not true false to change the value of i, and j will be incremented because we conduct another false success.

$$next\ i\ = the\ same\ as\ prevous\ i, next\ \ j$$
$$= prevous\ j + 1, next\ k$$
$$= previous\ k + 1$$

In the beginning of the algorithm , we have zero key processed (k=0), zero false successes j=0, and we have not seen the true success key , which is of course unique (i=0), so we at The state in grey in figure 3

Each state where i = 0 (indicating that the correct key has not yet been found) has three possible transitions:

1. Finding the True Key:

In the next state, we successfully identify the correct key.

- As a result, i is updated to 1 (indicating a successful identification)
- k is incremented by 1 , since we processed the next key.
- The value of j remains the same because no false success (caused by a hash collision) has occurred.

$$Transition: i = 0 \rightarrow 1, j = same, k$$
$$= k + 1$$

Accumulated probability of this state = the previous state probability that led to this state multiplied by the probability (Bernoulli formula) of selecting the correct key among the remaining (unprocessed keys yet) keys ($\frac{1}{N-k}$)

So that

$$the\ probability\ of\ next\ state\ \ (1\ \ , j\ , k+1) +$$
$$=$$
$$= \ the\ probability\ of\ previous\ state\ (\ 0\ , j\ , k)$$
$$\times \frac{1}{N-k})$$

2. Processing a False Key (Without Collision):

The next key is incorrect, but no hash collision occurs.

- k is incremented because we processed another key.
- i remains 0, as the correct key has not yet been found.
- j also remains the same since no false success has occurred.

$$Transition: i = same, j = same, k$$
$$= k + 1$$

Accumulated The probability of this state = the previous state probability that led to this state ultiplied by the probability of selecting the incorrect key among the remaining keys ($\frac{N-K-1}{N-k}$) without probability of existing error of collision,

and as the possibility of collision is Err, so the absence of collision is $1 - Err$
So that

$$The\ probability\ of\ next\ state\ (0\ ,j\ ,k+1\ ) += = the\ probability\ of\ previous\ state\ (\ 0\ ,j\ ,k-1)\ \times (\frac{N-K-1}{N-k}) \times (1-Err)$$

3. Processing a False Key (With a Collision):

The next key is incorrect, but due to a hash collision, the receiver mistakenly considers it correct.

- k is incremented because we processed another key.
- i remain 0 since this is not a true match.
- j is incremented, as we now have an additional false success.
-

$$Transition: i = same, j = j + 1, k = k + 1$$

Accumulated probability of this state = the previous state probability that led to this state multiplied by the probability of selecting an incorrect key among the remaining keys ($\frac{N-K-1}{N-k}$) but here we will consider the existance of error, so we will multiply it by error, so that:

$$the\ probability\ of\ next\ state\ (0\ ,j+1\ ,k+1\ ) += = the\ probability\ of\ previous\ state\ (\ 0\ ,j\ ,k-1)\ \times \frac{N-K-1}{N-k}) \times Err$$

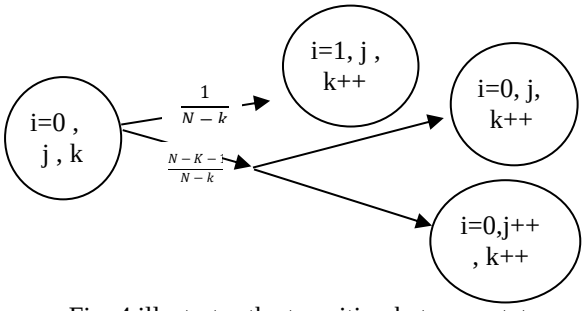

Fig. 4 illustrates the transition between states when i=0

Now, consider a scenario where i = 1, meaning the correct key has already been found. At this point, the next state has two possible transitions:

let's assume that we are at the state marked in red in Figure 3, which represents the point where the correct key for true success has already been found. At this stage, every subsequent key processed must be incorrect since there is only one unique true correct key for successful decryption.

4. Processing a False Key (Without Collision):

The next key is incorrect, and no hash collision occurs.

- k is incremented because we processed another key.
- i remain 1, as the correct key has already been found.
- j remains the same since no false success has occurred.

$$Transition: i = same, j = same, k = k + 1$$

Here, since the correct key has already been found, the only remaining probability (100%) is to find the incorrect key. This key is either considered false with probability equal to the absence of error $(1 - Err)$ or it is considered false true due to collision with probability Err.

Accumulated probability of this state = the previous state probability that led to this state multiplied by the probability of the absence of error, so we will multiply it by the error probability, so that

$$the\ probability\ of\ next\ state\ (1\ ,j\ ,k+1\ ) += = the\ probability\ of\ previous\ state\ (\ 1\ ,j\ ,k) \times (1-Err)$$

5. Processing a False Key (With a Collision):

The next key is incorrect, but due to a hash collision, the receiver mistakenly considers it correct.

- k is incremented because we processed another key.
- i remain 1 since we have already found the correct key.
- j is incremented, as we now have an additional false success.

$$Transition: i = same, j = j + 1, k = k + 1$$

$$the\ probability\ of\ next\ state\ (1\ ,j+1\ ,k+1\ ) += = the\ probability\ of\ previous\ state\ (\ 1\ ,j\ ,k) \times Err$$

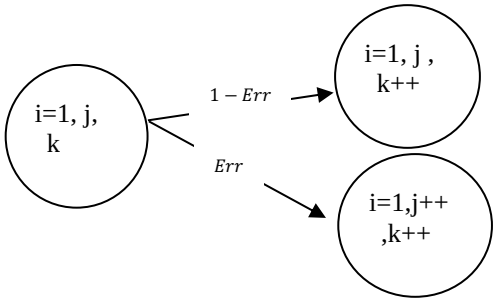

Fig. 5 State Transition from a State with i = 1

This structured transition system effectively models the behavior of sequential key

processing, distinguishing between true and false successes, and tracking how hash collisions influence the perceived correctness of a given key. In summary, each node with i=1 will birth 2 nodes, and each node with i=0 will birth 3 nodes. The value of children (i, j, k) in the next state will be calculated as mentioned in states 1 to 5

The proposed algorithm is indicated in Figure 6, it is written in Mathcad

Describing the Algorithm

1- We aim to compute the probability of finding j correct keys from k processed keys, without distinguishing whether The successful keys are genuinely correct or as a result of a hash collision in the received message. This probability is represented by the Res_good matrix. In the Res_good matrix, each columnorresponds to the number of processed keys, ranging from 0 to N, where N is the number of sources in the network, while each row represents the number of successful matches (without differentiating between true matches and those caused by hash collisions).

The values in each cell indicate the probability of obtaining j successful matches (whether genuine or due to a collision) after processing k keys.

The probability Res_good(j, k) is computed as the summation of two probabilities for each k (1 → N) and j (1 → k):

a) True Match – when a correct key is successfully identified.

b) False Match – when a hash collision leads to an incorrect key being mistakenly considered correct.

While Res_good(j, k) accounts for the total probability of both true and false matches, it excludes the probability of false matches. This ensures an accurate representation of the probability of obtaining j successful key identifications after processing k keys considering the presence of a collision error probability (Err). Figure 7 shows the description of Res_good.

2- In addition to calculating the total probability of success, we also need to determine the percentage of true successes proportional to the total probability of success (both true and false successes) when processing k keys with j successes in the decryption process. This probability is represented by the Res_bad matrix.

The Res_bad M^p matrix is computed for each k (number of processed keys) and each j (number of successful matches after processing k keys), providing insights into the likelihood of correct key identifications within the decoding process. The probability is defined as:

$$Res_{bad}(j,k) = Probability\ of\ obtaining\ truematches\ after\ processing\ k\ keys\ /\ Total\ probability\ of\ success\ \binom{true\ +\ false}{matches}$$

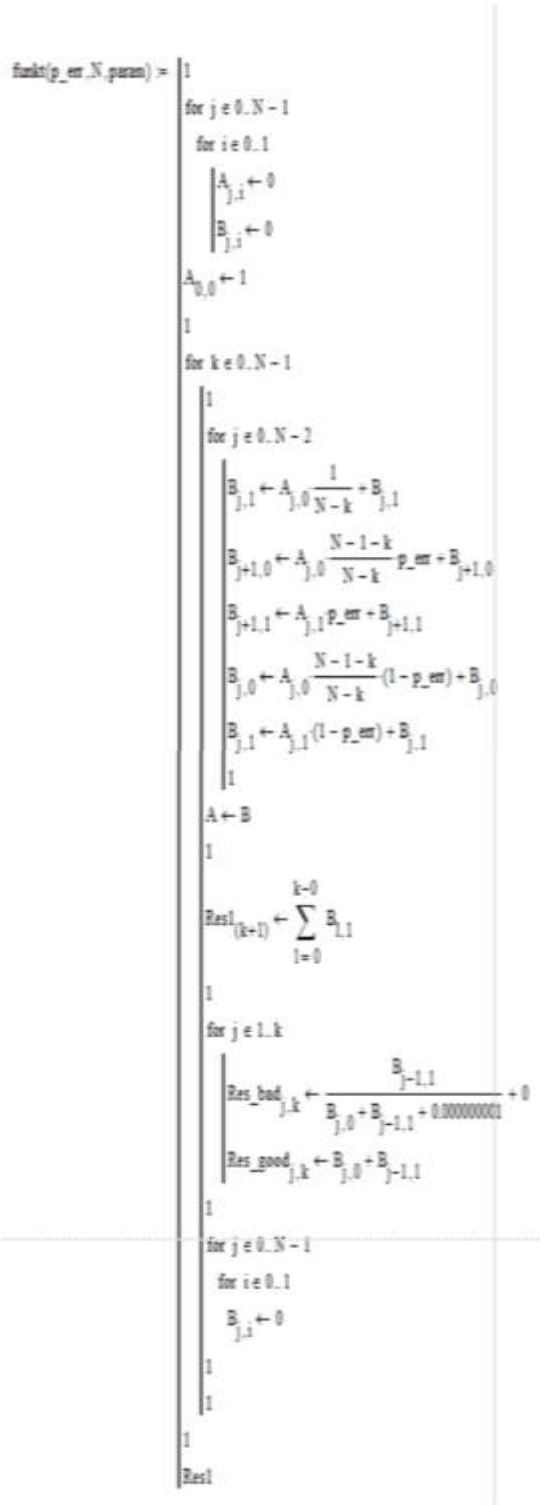

Fig. 6 The proposed algorithm

	Processed keys for finding if they are match /mismatch →					
Number of matches (successful decoding) →						

Fig. 7 Res_good matrix

This probability can be interpreted as a posterior probability, representing the adjustment in probability after incorporating new information. Specifically, it reflects the change from the initially computed probability in Res_good once we account for the fact that some of the identified keys resulted from hash collisions rather than being truly correct. Even though the decoding process initially considered them correct, Res_bad quantifies the probability of these false successes within the total observed successes.

Each cell in the Res_bad matrix corresponds to the percentage of probability that we will find a truly correct key if we find j correct keys after processing k keys, relative to the total number of successes (both true and false) at that j,k . This metric provides valuable insight into the impact of collision errors and helps in evaluating the reliability of the decoding process in the presence of errors.

At this point, we have the probability of finding j correct keys after processing k keys, which we named Res_good. We will denote Res_good as Mg and Res_bad as Mb Then, we find the percentage of finding j true keys among the total successes (true and fake), which we name Mb.

So, if we process k keys, we have to know how many successes we have to find before stopping looking for a correct key.We know that there is only one correct key, but due to collisions, this key may be incorrect . Therefore,we continue looking for successes in decryption operation until a stop condition is satisfied. We need to determine the probability that each success key is truly the correct key and not fake. This can be achieved by checking Mb and assessing the likelihood that the found key is genuine (not a fake success due to collision).

The stop condition is that for each column of Mb, we need to find the probability above the threshold with the minimum j. Reaching this probability will be the stop condition for the number of processes keys denoted as keys k.

The threshold should be between 0.9 and 0.99

As soon as this condition is satisfied, we will have one or more keys, and all of them will answer that they are the correct keys. We need to distinguish which of them is the correct key. This process will be done using two steps:

Step 1: Build a confidence matrix. Each key used to find the correct match has its counter incremented. Finally, a confidence score is attached to each key associated with sender i (Si).

$$P\_confidence(key\ of\ Si) = Number\ of\ successful\ keys\ from\ Si\ /Total\ successful\ keys\ found$$

Step 2: After ranking senders, apply extra checks to confirm correctness.

1. A. Re-Evaluate Stored Messages by Retrieving previously received messages from storage, then comparing candidate keys against past verified keys. If a sender's key has appeared in prior verified messages, they gain higher confidence.

2. Use Additional Metadata like

If messages from one sender arrive earlier, they are more likely to be correct because they were received more recently. If messages include CRCs or digital signatures, we need to validate them. If a sender's key matches previous data trends, the key is more likely to be correct.

Step 3: Decision-Making

After all filtering steps:

If one sender dominates (highest probability), accept it as the true key ; if two or more senders have similar probabilities, choose the one with a lower false success rate. Or that appears more frequently in independent checks or whose message structure aligns best with prior verified messages.

4. RESULTS AND DISCUSSIONS(10 PT)

If we pick these parameters to be applied by the algorithm , Number of sources =15, Probability of Error =0.15

The Mb matrix will be shown in Figure 8

	0	1	2	3	4	5	6	7	8	9	10	11	12	13
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0.209	0.357	0.377	0.4	0.406	0.455	0.488	0.526	0.571	0.625	0.69	0.769	0.87
2	0	0	0.526	0.548	0.571	0.597	0.625	0.656	0.69	0.727	0.769	0.816	0.87	0.93
3	0	0	0	0.645	0.667	0.69	0.714	0.741	0.769	0.8	0.833	0.87	0.909	0.952
4	0	0	0	0	0.727	0.748	0.769	0.792	0.816	0.842	0.87	0.899	0.93	0.964
5	0	0	0	0	0	0.787	0.806	0.826	0.847	0.87	0.893	0.917	0.943	0.971
6	0	0	0	0	0	0	0.833	0.851	0.87	0.889	0.909	0.93	0.952	0.976
7	0	0	0	0	0	0	0	0.87	0.886	0.903	0.921	0.94	0.959	0.979
8	0	0	0	0	0	0	0	0	0.899	0.914	0.93	0.947	0.964	0.982
9	0	0	0	0	0	0	0	0	0	0.922	0.937	0.952	0.968	0.984
10	0	0	0	0	0	0	0	0	0	0	0.94	0.956	0.971	0.985
11	0	0	0	0	0	0	0	0	0	0	0	0.94	0.969	0.986
12	0	0	0	0	0	0	0	0	0	0	0	0	0.865	0.967
13	0	0	0	0	0	0	0	0	0	0	0	0	0	0.569
14	0	0	0	0	0	0	0	0	0	0	0	0	0	0
15														

Fig. 8 Mb with parameter N=15, Err=0.15

And we can simulate it using a 3D plot, as shown in Figure 9

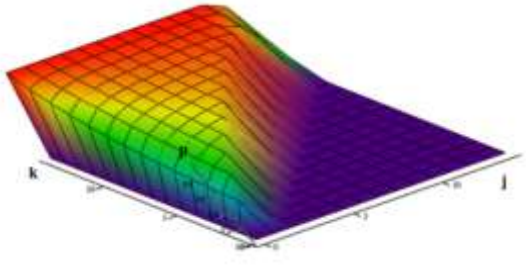

Fig. 9 Mb 3D representation for parameter N=15, Err=0.15

From Figures 8 and 9, if we set a threshold of probability to consider the success key to be true to be over 0.95, we find j successes after processing k keys then we can stop searching for the correct key since there is 0.05 probability (the remaining from 0.95), so there is a possibility that the correct key is not found yet, which will cause an error of dropping correct message came to the receiver, figure 10, depicts how changing the size of the network, change the number of requires successes to stop reaching for a correct key.

In Figure 10 we plotted the stopping j versus network size and process keys for various error and threshold values.

Some decoding operations would not benecessary if we implement our lgorithm allowing us to calculate the reduction in compression operations for each case.

The reduction will be equal to the number of decoding processes using our proposed algorithm (A) /number of decoding processes without using our method (B) minus one, as indicated in Table 1

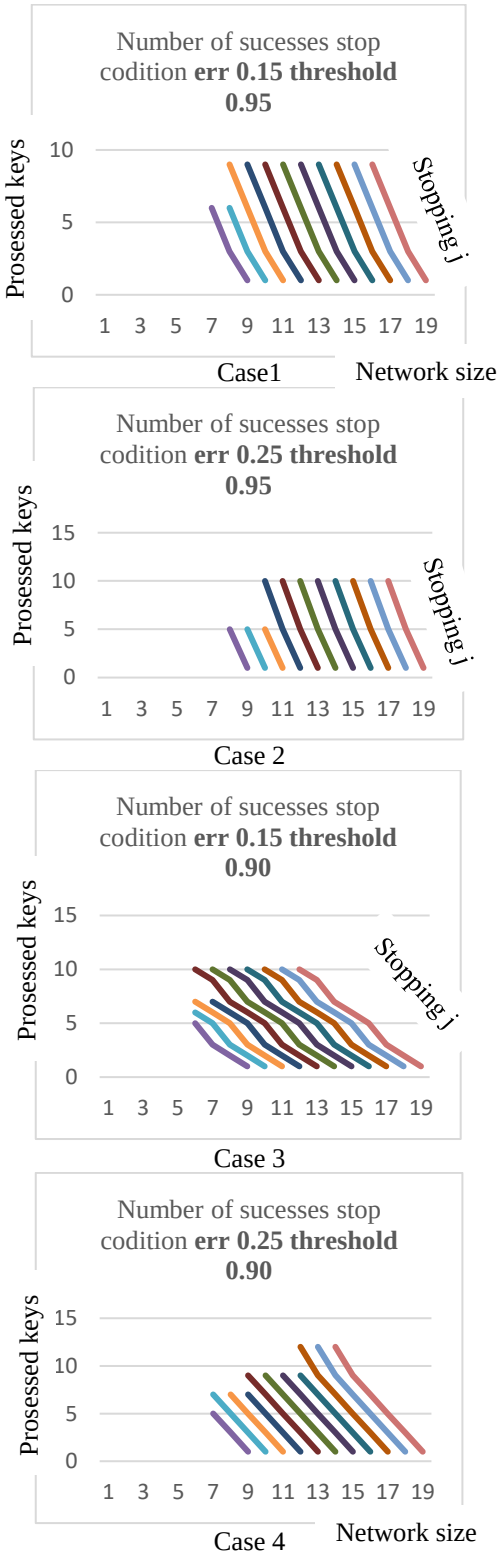

Fig. 10 The relation of stopping j with threshold, N, k, Err

Table 1: Reduction for each case study

case	A	B	Reduction Percentage
1	84240	96000	0.14
2	87800	96000	0.09
3	75080	96000	0.28
4	81820	96000	0.17

The results seem reasonable becausedecreasing the threshold stops the search process earlier as fewer false successes are accepted as a stop condition, Thus,we Do not need to deploy more iterations to improve the probability that this key is correct. Also, increasing the error probability will require us to proceed further in decrypting processes to cross the threshold and satisfy the stop condition.

While the results emphasize that the probabilistic approach proposed by the authors can considerably lessen the number of keys to be processed when compared with brute force decryption, the values of reduction percentages in Table 1 (between 9% and 28%) further indicate that performance gains vary largely as a function of both the error probability and the threshold. The user interprets an earlier cut of the key search, saving on computational efforts, but this may increase the likelihood of accepting some false successes. In contrast, higher error probabilities delay the stopping condition, which in turn occurs when the system must process a larger number of keys before reaching the desired confidence level. This trade-off highlighhts the downside of both efficiency and so far-perfect reliability-an argument that appears more than once in previous works on anonymous networks [5,6].

Nevertheless, unlike certain previous studies where optimizing for routing or encryption was the primary concern, the present work focuses on decryption itself and thus filling an obvious research gap. From a practical point of view, therefore, reducing the number of keys processed directly affects energy consumption in nodes and, consequently, their lifetimes, which is vital for for both IoT and resource-constrained settings. As the network grows, however, substantial efficiency gains may be realized, enhancing the scalability of the model. These findings confirm that not only does the proposed algorithm reduce complexity, but it also makes anonymous communication systems more sustainable and reliable.

5. CONCLUSION (10 PT)

This article discussed a probabilistic lightweight algorithm for key processing that can be used in anonymous communication networks. The work arose due to the ever-increasing computational cost imposed on the receiver, as the latter has to exhaust virtually all conceivable keys for any given incoming message. To solve this problem, a discrete Markov chain model for the decryption process was considered together with Bernoulli trials. This allowed for the analysis of the probability for correctly detecting the key and defining an optimal stopping criterion.

Simulation showed that the potentially proposed method gave good results while reducing processed keys by around 9% to 28% compared to brute-force search. These savings translate to lead to energy conservation and an extension of node lifetime, which is a very important consideration in constrained environments such as IoT devices and decentralized telephony. Moreover, as the number of nodes increases from relatively low to infinite, the higher the gains, and the more scalable the model becomes.

The novelty of the presented research hinges on the shift from optimizing routing and encryption-which is predominantly emphasized in the literature-to focusing on the decryption process-an area hardly ever addressed. By integrating the probabilistic study of decryption, the research fills a critical niche.

Still, some limitations remain. Foremost, the error probability is presently uniform, and dynamic network conditions such as node churn or adversarial strategies are not considered. Future works should try to overcome these limitations by lifting the algorithm into heterogeneous environments, employing machine learning techniques for adaptive threshold tuning, and validating the model over real-world datasets. In a nutshell, these findings show how probabilistic modeling of decryption provides a great potential for the sustainability, reliability, and efficiency of anonymous networks-a workable solution for next-generation communication systems.

REFERENCES

[1] H. Zhao and X. Song, “TOAR: Toward Resisting AS-Level Adversary Correlation Attacks Optimal Anonymous Routing,” Nov. 2024, [Online]. Available: https://doi.org/10.3390/math12233640

[2] D. Liu and Y. Park, “Anonymous Traffic Detection Based on Feature Engineering and Reinforcement Learning,” Apr. 2024, [Online]. Available: https://doi.org/10.3390/s24072295

[3] K. Khalil, H. Idriss, T. Idriss, and M. Bayoumi, Lightweight Hardware Security and Physically Unclonable Functions. Springer Nature, 2023.

[4] R. Aghazadeh, A. Shahidinejad, and M. Ghobaei-Arani, “Proactive content caching in edge

computing environment: A review," Software: Practice and Experience, Sep. 2021, doi: https://doi.org/10.1002/spe.3033 .

[5] K. Zhang and S.-Y. Chang, "Performance or Anonymity? Source-Driven Tor Relay Selection for Performance Enhancement," pp. 1–9, Jul. 2024, doi: https://doi.org/10.1109/icccn61486.2024.10637610.

[6] E. M. Ghourab et al., "Moving Target Defense Approach for Secure Relay Selection in Vehicular Networks," Oct. 2023, doi: https://doi.org/10.36227/techrxiv.24428176.

[7] A. Al, "Cybersecurity and personal privacy: Protecting yourself in the digital age," Open Access Research Journal of Science and Technology, vol. 12, no. 1, pp. 131–135, Oct. 2024, doi: https://doi.org/10.53022/oarjst.2024.12.1.0122.

[8] B. P. Pokharel and N. Kshetri, "blockLAW: Blockchain Technology for Legal Automation and Workflow -- Cyber Ethics and Cybersecurity Platforms," Oct. 2024, [Online]. Available: https://doi.org/10.48550/arXiv.2410.06143

[9] J. Zeng, S. Wu, Y. Chen, R. Zeng, and C. Wu, "Survey of Attack Graph Analysis Methods from the Perspective of Data and Knowledge Processing," Security and Communication Networks, vol. 2019, pp. 1–16, Dec. 2019, doi: https://doi.org/10.1155/2019/2031063.

[10] D. Rusinek, B. Ksiezopolski, and A. Wierzbicki, "Security Trade-Off and Energy Efficiency Analysis in Wireless Sensor Networks," International Journal of Distributed Sensor Networks, vol. 11, no. 6, p. 943475, Jan. 2015, doi: https://doi.org/10.1155/2015/943475.