A Novel Method for Secure Key-Sharing in 5G and Beyond

*Debjyoti Bagchi
Computer Science and Engineering
Calcutta Institute of Engineering and Management
Kolkata, West Bengal, India
Email: debjyotibagchi1982 [AT] gmail.com

Sanjib Sil
Computer Science and Engineering
Calcutta Institute of Engineering and Management
Kolkata, West Bengal, India
Email: sanjib.sil [AT] gmail.com

Soubhik Sen
Computer Science and Engineering
Calcutta Institute of Engineering and Management
Kolkata, West Bengal, India
Email: soubhiksen100 [AT] gmail.com

Abstract—**In today's world, online communication is essential, and infrastructure for cutting-edge mobile technologies like 5G, is growing daily to meet the demand. So, information sharing security needs to be safeguarded as electronic communications spread. To implement this, cryptography is typically used, and most commonly symmetric key cryptography, due to its many advantages over other crypto-systems. However, one significant disadvantage of symmetric key system is that the single-key-sharing is exposed to all entities in a network communication system, which makes the subsequent communications vulnerable to unauthorized access. There are many approaches for securing the single-key-sharing transmission, but each has its own drawbacks. In this paper, we propose and present a novel approach to secure key-sharing over a communication network in symmetric key cryptography system which makes the single-key immune to unauthorized access. A total of four messages are exchanged between two devices for our secure key-sharing method. To implement the key sharing process, our method employs a few techniques, including asymmetric key cryptography, hash functions, machine learning-based pseudo random number generators, and timers. Our analysis shows that, besides providing similar level of confidentiality as the existing approaches, it also provides other significant improvements over the current ones, such as enhanced integrity maintenance, and authenticity verification of the two devices involved in the process. The short latency of modern 5G networks helps to balance the increased network demand caused by sending four independent messages. To determine timer duration and key validity, we propose applying AI algorithms and extending the security of our method; nevertheless, these applications fall within the purview of our upcoming study.**

Keywords- Cryptography; Symmetric key cryptography; Asymmetric key cryptography; Key sharing

I. INTRODUCTION

Presently, we are heavily dependent on communication network for our day to day activities, such as, an ordinary message exchange, an online banking transaction, a pizza order from a favorite outlet, streaming video program of our choice, an online meeting, etc. This improved communication services have been achieved gradually, from 1G to present 5G mobile communication technology. We are currently living in the 5G communication age, which provides three main application services, namely –

- URLLC: Ultra-Reliable and Low Latency Communications
- eMBB: Enhanced Mobile Broadband
- mMTC: Massive Machine-Type Communications

Secure and reliable data communication is one of the prime features of 5G and beyond communication systems.

To accomplish the desired level of secure communication, "Cryptography" can be used to safeguard any online interactions. It is a process of securing information and communications using codes, so that only those whom the information is intended for, can access, read, or process it. However, there are still certain security issues associated with the cryptography technology in some situations. Therefore, it should be properly planned before applying any cryptography technology, and attention is to be given to the type of communication and the degree of security to be provided.

The figures 1, 2 & 3 below illustrates the basic idea of cryptography and its classifications:

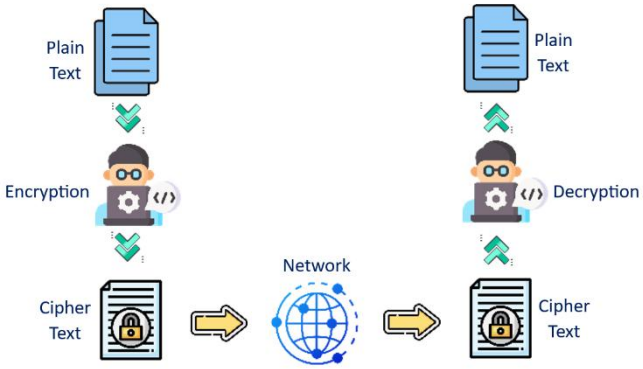

Figure 1. Basic concept of cryptography

Cryptography can be classified into two types:

1. Symmetric key cryptography –

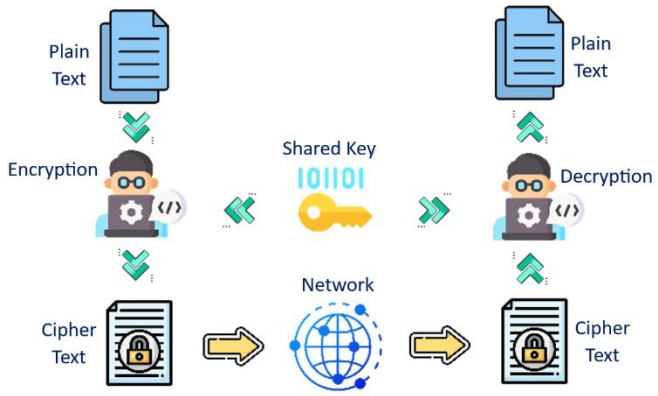

Figure 2. Basic concept of symmetric key cryptography

2. Asymmetric key cryptography –

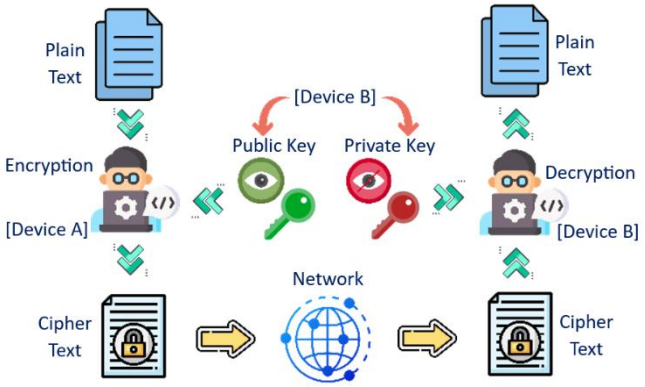

Figure 3. Basic concept of asymmetric key cryptography

Symmetric Key Cryptography: This cryptography process uses a single key (called as secret key) to encrypt the plain-text (information) and decrypt the cipher-text, i.e., to decrypt the information, one must have the same key that was used to encrypt it. The key, in practice, represents a shared secret between two or more parties that can be used to maintain a private information link.

Asymmetric Key Cryptography: This process requires two different keys, one to lock or encrypt the plain-text, and the other to unlock or decrypt the cipher-text. Neither key will do both the functions. One key is published (public key) and the other is kept private (private key). If the lock/encryption key is the one published, the system enables private communication from the public to the unlocking key's owner. If the unlock/decryption key is the one published, then the system serves as a signature verifier of documents locked by the owner of the private key.

Symmetric key cryptography has a wide range of practical applications, including –

- Large-scale, secure data transfer across the internet between devices, making sure that the data's contents are only accessible by approved devices.
- Disk encryption, which safeguards locally stored data, and stops malicious individuals from accessing personal data even if they physically possess the machine.
- Guarding Personally Identifiable Information (PII) during credit card transactions to stop fraud and identity theft.

Likewise, there are numerous practical applications for asymmetric key cryptography, including –

- Improving email security and safeguarding message contents.
- Offering enhanced security in Secure Socket Layer (SSL) and Transport Layer Security (TLS), which are utilized to create safe connections between a web browser and a trusted website.
- Ensuring the legitimacy of exchanges utilizing digital currency, such as Bitcoin.

However, there are advantages and disadvantages to each strategy. To start, asymmetric key cryptography offers the following advantages:

- It provides great key security because the public key is accessible to all parties, but the private key is exclusive to the host device [1] [2] [3].
- It offers a high degree of protection and defense against various kinds of attacks [2] [4].

One of its disadvantages is as follows:

- It is complex and requires more computational time compared to other cryptographic techniques [4] [5] [6] [7].

Conversely, symmetric key cryptography is more often used because it has the following benefits –

- It is quicker and requires much less computational time than other cryptographic techniques [4] [7] [8] [9] [10].
- It can process massive volumes of data quickly and effectively [5] [11].
- Because both the encryption and the decryption processes share a common key, they are easier to implement [1] [3].

Disadvantage of symmetric key: The requirement that both parties need to have access to the same single key (secret key) is one of the main drawbacks of symmetric key encryption.

Many studies are being conducted to improve the symmetric key and asymmetric key cryptographic techniques for secure communications in the present 5G and the upcoming 6G communication systems. As symmetric key techniques offer simpler implementations and lesser compute time, it can be used to facilitate quicker and more effective communication. This motivates us, the authors, to concentrate our research work on symmetric key cryptography and find the practical problems

associated with the system. A literature survey on different works carried out by researchers in this area are as follows:

In their study [8], E. Rawat has discussed the advantages and disadvantages of cryptographic methods, such as symmetric key cryptography. They have found that although symmetric cryptography is quicker and effective, it is susceptible to both active and passive attacks, such as repudiation and masquerade.

In their work [5], S. Bera looked at symmetric key cryptography, and found it to be useful for securing large data collections. However, the issue of sharing the single secret key is mentioned as being one of its disadvantages.

According to M. Zubair's research [1], symmetric key cryptography is simple to use because it only needs one key for both encryption and decryption; nevertheless, sharing the key is considered as the trickiest part of the process.

M. Biswas discussed in their study [2], the different ways in which different encryption techniques, such as symmetric key cryptography, contribute to network security. While symmetric key cryptography works very well, the need for secure key exchange is its major drawback.

As noted by M. Sharma in their study [6], since symmetric encryption uses the same key on both the sender's side and the recipient's side, a robust and secure way needs to be used for key transmission.

In their publication [12], S. Kumar describes a technique for sending symmetric keys. It involves transforming the key on the sender side into a hash code, and then on the receiver side back into the original key. Although the converted key is more difficult for an adversary to decrypt, the conversion procedure is deterministic and therefore lacks resiliency.

In their study [4], A. Kakkar looked at several cryptographic techniques that could be used in 5G networks. Despite the great security and quick processing of symmetric encryption, it may not be able to resist powerful attacks because it only uses one key.

In their study [3], S. Chandra covered a number of both novel and established symmetric key cryptography techniques. Symmetric key cryptography operates with a single key; nonetheless, security of the symmetric key itself is important, to ensure that it is not compromised, as this could allow an attacker to readily decrypt any subsequent encrypted messages.

Among the several symmetric key cryptography algorithms currently in use, S. Vyakaranal has examined and contrasted them in their study [13]. The main disadvantages of AES is that its performance is lower than DES and Blowfish in terms of throughput and bandwidth consumption respectively, although it is still extremely good overall and adequate for most applications.

In their study [7], F. Maqsood contrasted and examined symmetric and asymmetric key cryptography methods. When it comes to computational cost, symmetric key cryptography is less expensive than asymmetric key cryptography. However, because symmetric key cryptography techniques employ smaller keys, they are less safe when used with extremely sensitive material.

We, the authors, have also reviewed several other research works [9] [11] [14] [15] [16] [17] [18] [19] [20] [21] [22] [23] [24] [25] on the topic of symmetric key cryptography. It has been found from the review that key transmission across unreliable channels remains a problem in symmetric key cryptography, which leaves it vulnerable to unauthorized access attacks. *Therefore, we concentrate our research efforts on safeguarding symmetric key cryptography's key transmission procedure.*

Key Exchange Procedures in Cryptography:

The generic key exchange (sharing) mechanism in symmetric key cryptography through an unreliable channel is shown in figure 4.

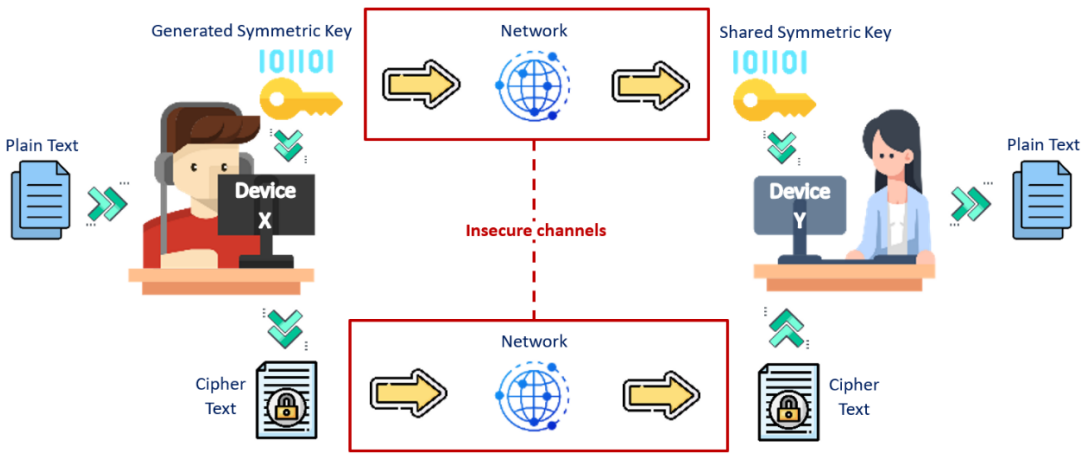

Figure 4. Issue with symmetric key cryptography: exchanging keys across an unreliable channel

Currently, the two most often utilized procedures for safe key sharing in unreliable channels are:

1. Asymmetric encryption for key sharing – It safeguards the confidentiality of the secret key by encapsulating it using an asymmetric key scheme such as RSA [24]. Figure 5 explains how this technique operates.

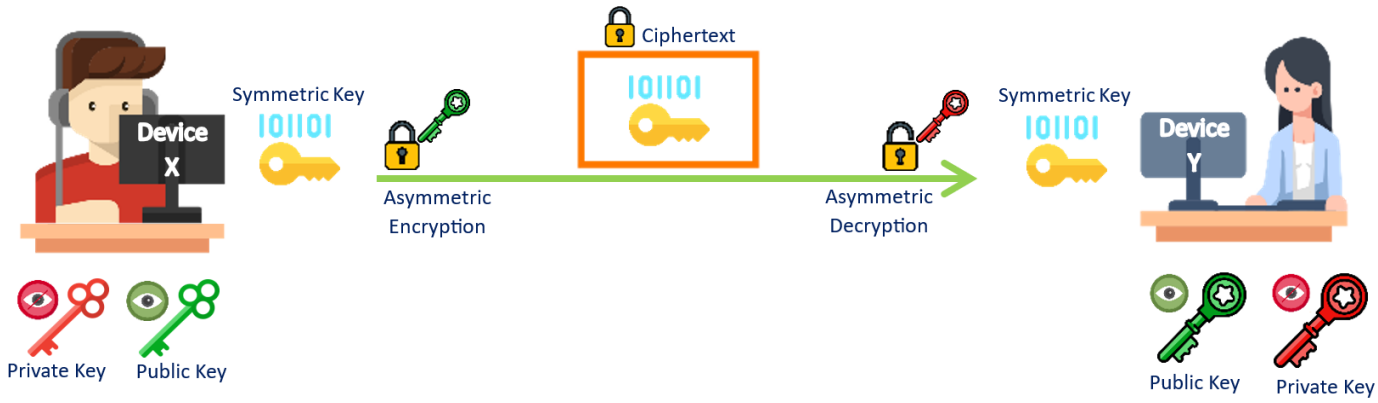

Figure 5. Method of distributing keys via asymmetric encryption

Nevertheless, this procedure has the following drawbacks –

- Since no hash function is utilized during key transmission, it offers no defense against integrity attacks, which might compromise message integrity covertly [20].
- There is no way to confirm that Device X sent the symmetric key, and not someone else. This is because anyone can obtain Device Y's public key, and use it to encrypt and send data to Y. Thus, authenticity is not maintained here.

2. Key exchange protocol (Diffie-Hellman) – The technique is shown in figure 6. It guarantees confidentiality by building the secret key locally after exchanging two generated result values [24].

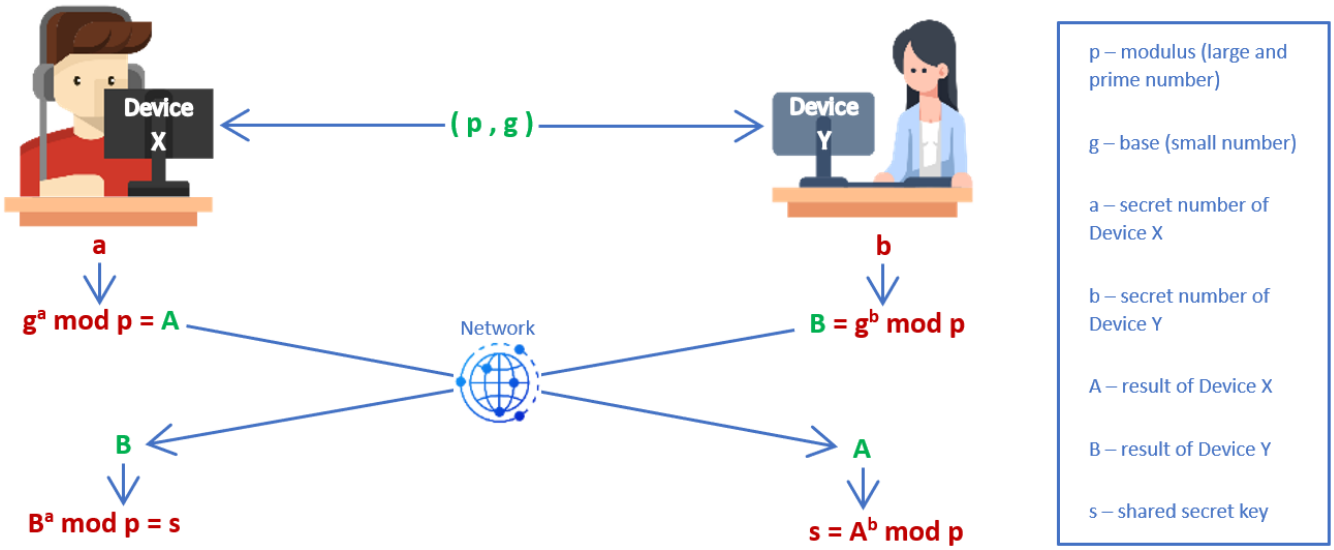

Figure 6. Diffie-Hellman key exchange procedure

There are certain drawbacks to this procedure as well –

- When transferring the values of A and B, no hash values are attached to them. So, there is

no defense against integrity attacks, which could potentially jeopardize the integrity of these values without being detected by either of these devices [20].

- It is impossible for Device X to verify that Device Y sent B, and impossible for Device Y to verify that Device X transferred A. Thus, authenticity is not maintained here [21].

Therefore, based on the procedures covered above and the literature review, *we, the authors, conclude that symmetric key cryptography's secure key sharing issue has not yet been adequately resolved. Thus, we propose a novel approach to solve this issue in this research paper.*

II. METHODOLOGY

Our method enables the safe transmission of a produced symmetric key between devices, enabling fast communication between them via symmetric key cryptography with the same shared symmetric key. The main goal of our method is to offer a secure key transmission procedure.

Our method requires a total of four communication steps to safely transfer the *'key'* in symmetric key cryptography. We use asymmetric key cryptography technique to guarantee message confidentiality. In this method, four signals C_1, C_2, C_3 and C_4 (in packet form) are used to transfer the key from the sender to the authentic receiver. These signals are sent in four steps. C_1 is the first start signal sent from the sender to the receiver. C_2 is the corresponding acknowledgment signal sent from the receiver to the sender. After receiving the acknowledgment signal from the receiver, C_3 is the 'key transmission signal' sent from the sender to the receiver. C_4 is the final 'key received acknowledgment signal' sent from the authentic receiver to the sender.

Our methodology includes a timer as a critical element, which is primarily used to verify the actual recipient as an authentic receiver. The four signal steps should be completed within a predefined stipulated time-period, i.e., in a predefined period, the sender should receive an acknowledgment from the authentic receiver. This will happen only when a valid receiver decodes the message within a specified short time and sends the correct acknowledgment. An unwanted receiver (or a hacker) will probably take a longer time to decode the message.

The signals C_1, C_2, C_3 each consists of three fields, while C_4 consists of two fields. The fields are marked as R, S, H & K respectively. The field R consists of Random (number/character) strings, field S consists of Synchronization bits, field H consists of Hash value, and field K consists of the Key value that is to be shared. The signal formats are shown in figure 7. The random character sequences or strings are utilized for maintaining authenticity, the synchronization bits are required for sender & receiver synchronization, and hash functions are used to boost integrity.

The overview of the method is shown in figure 7, and the flow diagrams of the corresponding method are shown in figure 8, 9, 10, and 11. The method is explained as follows – considering key sharing between two devices, Device X and Device Y, i.e., they are attempting to exchange a key, with X acting as the sender and Y as the recipient. The symmetric key is to be shared from X to Y using four distinct signal packets.

At first, the Key Sharing Start Signal (C_1) packet is initiated by X, with the packet containing the hash value H_1, the synchronization bit S, and the random number/character R_x. The packet is then encrypted using Device Y's public key, and then transmitted to Device Y.

After receiving C_1, Device Y decrypts C_1 packet by using Device Y's Private Key, and extracts and retains R_x. Device Y then creates an Acknowledgment Signal packet (C_2) for Device X, which is subsequently encrypted using Device X's public key. This packet contains the hash value H_2, the random number R_y created by Device Y, and the R_x received from Device X. The packet is then transmitted to Device X.

Now if the packet C_2 reaches Device X within the predicted or predetermined timeout value T_1, then Device X uses its private key to decode the message, otherwise the transaction is considered invalid. After decoding the packet, Device X extracts and retains R_y. Device X then forms the signal packet C_3, and encrypts it using Device Y's public key. The packet contains the symmetric key *'K'*, the hash value H_3, and the random number R_y of Device Y. The packet C_3 is then transmitted to Device Y.

Device Y expects to receive C_3 in a reasonable length of time T_2 after sending the acknowledgement C_2. Once it has received C_3, it decodes it using its private key, and retrieves the key K, and R_y which was previously sent. Device Y stores the key K, and compares the received R_y with its own R_y for authenticity. Once the comparison is satisfied, Device Y then forms the Acknowledgment Packet (C_4), with the previously received R_x of Device X, and H_4. This packet is then transmitted to Device X after encrypting it with Device X's public key.

Device X receives C_4 within a specified time T_3, and then decodes it using its private key, and compares the received R_x with its own R_x for authenticity of key sharing. Once satisfied, the key sharing transaction is considered a success.

The following diagram shows the general overview of our suggested approach:

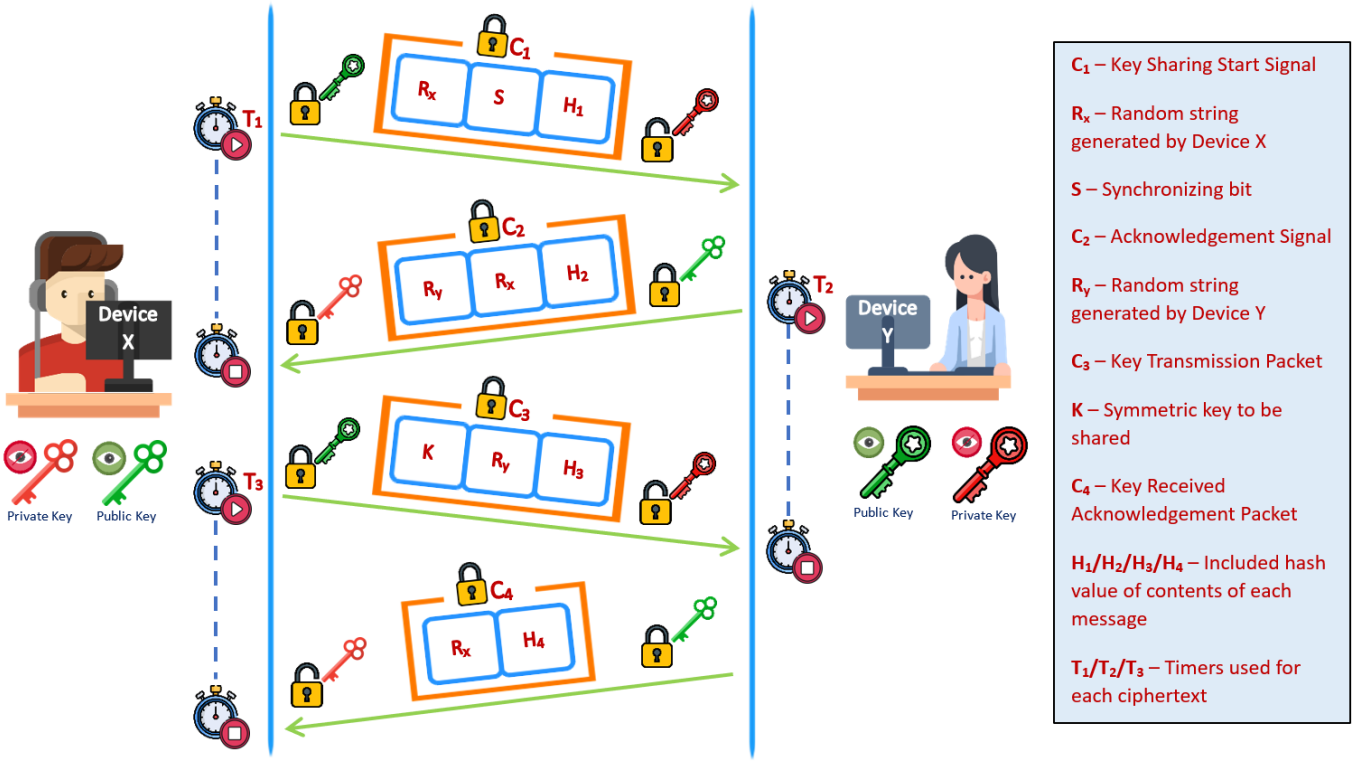

Figure 7. An overview of the suggested approach

Here in this procedure, we consider that the hash generator, pseudo-random number generator, and encryption/decryption techniques used by both devices to be the same. A modified machine learning framework, Generative Adversarial Network (GAN) is used as a pseudo-random number generator (PRNG) [26]. We utilize it here because it has the benefit of producing more unpredictable sequences. This feature is shared by the two devices. Additionally, we have used SHA-256 as the hash technique to generate hash values, due to its popularity, and it is also considered to be more secure by modern standards. Following are the advantages of this technique:

- Because of its strong resistance to several integrity related attacks, our system is safer against these kinds of attacks.
- The method's overall ability to ensure communication integrity is significantly improved.

The flow diagram of the four steps are as follows:

Step 1: Generation and Decoding of Key Sharing Start Signal (C_1):

Device X creates the packet C_1, which Device Y receives and decodes in the way that follows –

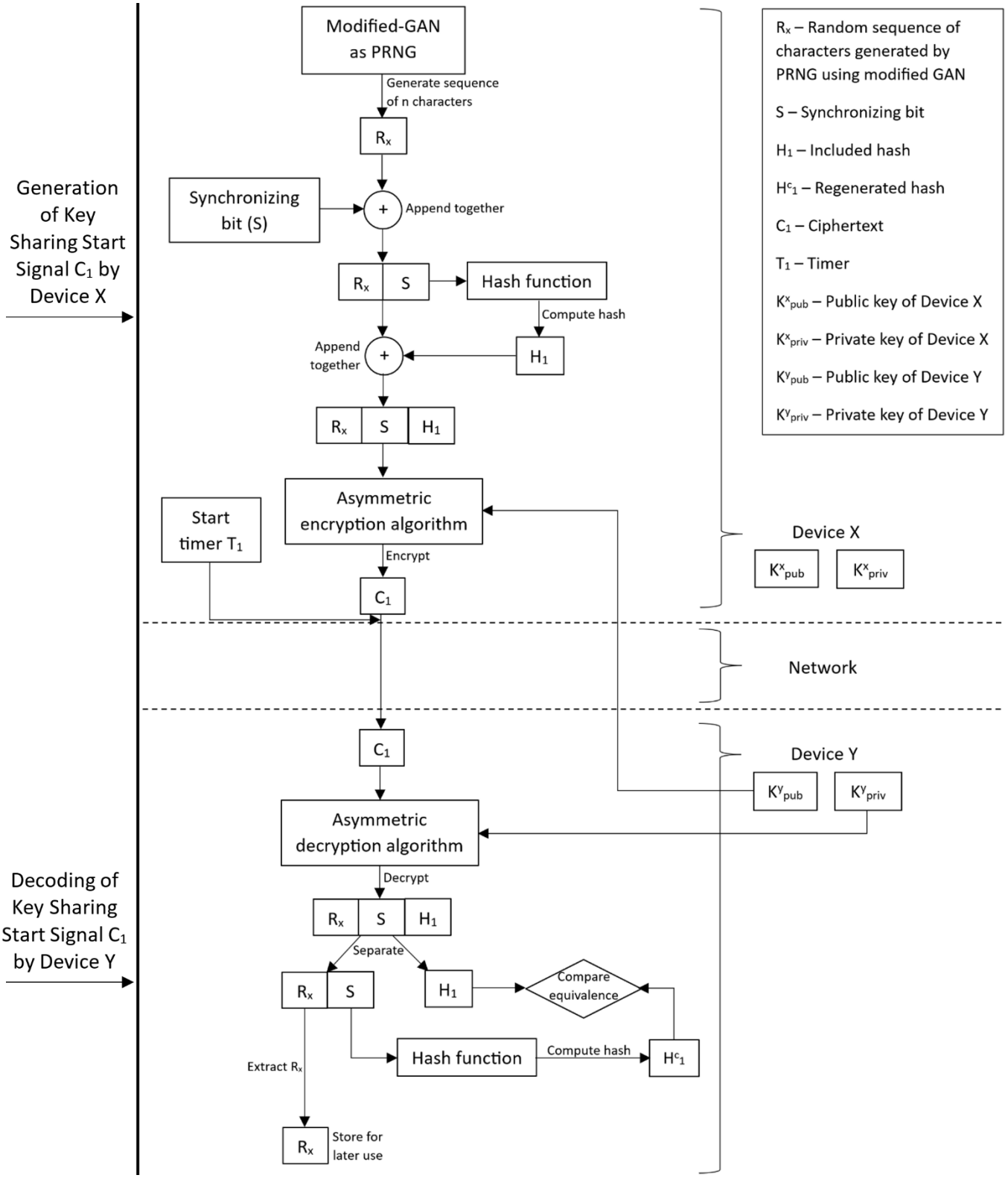

Figure 8. The detailed flowchart showing how Device X contacts Device Y via C_1 to initiate the key sharing process

Step 2: Generation and Decoding of Acknowledgement Signal (C_2):

Device Y creates the packet C_2, which Device X receives and decodes in the way that follows –

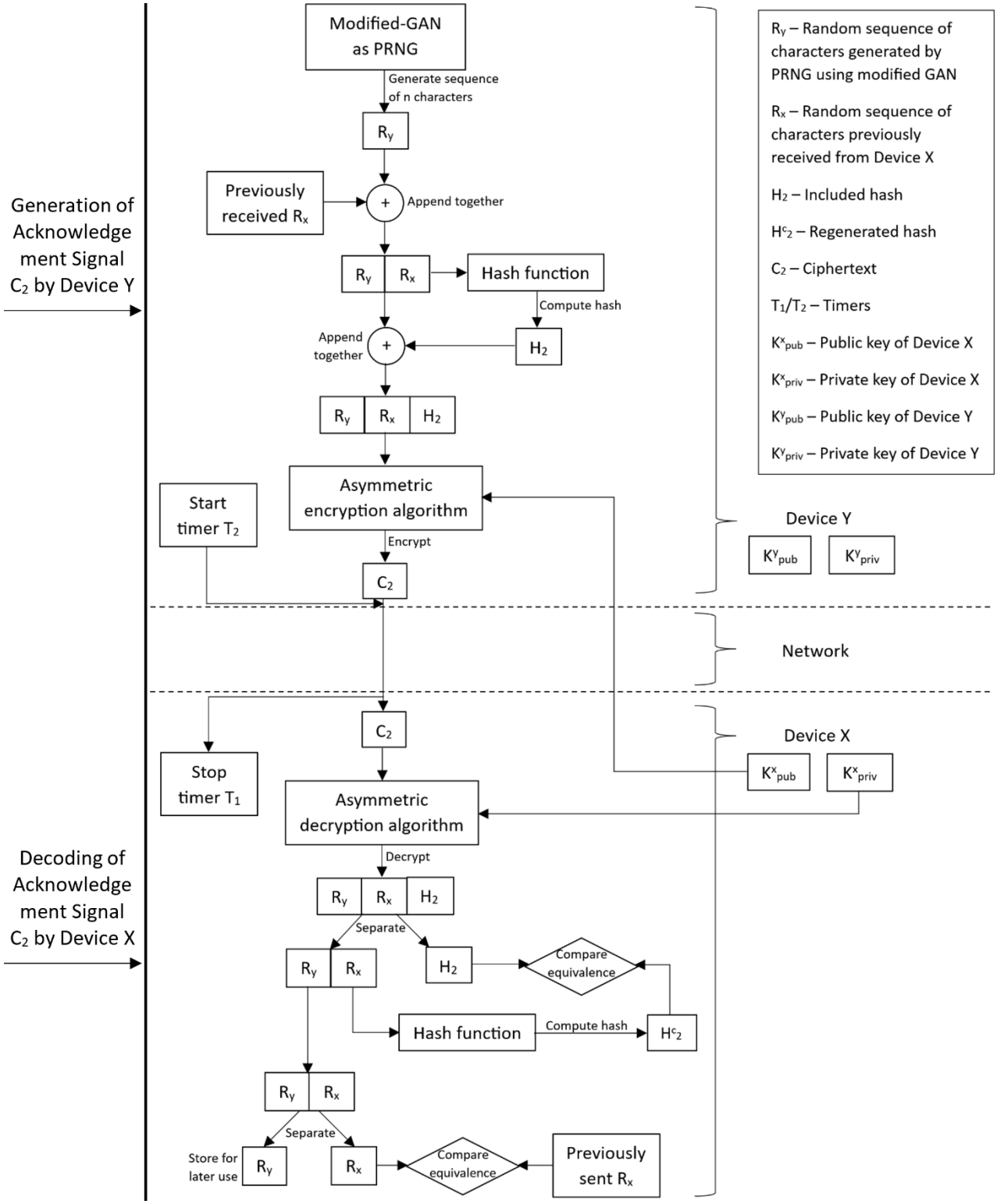

Figure 9. The detailed flow diagram demonstrating how Device Y uses C_2 to acknowledge Device X

Step 3: Generation and Decoding of Key Transmission Packet (C_3):

Device X creates the packet C_3, which Device Y receives and decodes in the way that follows –

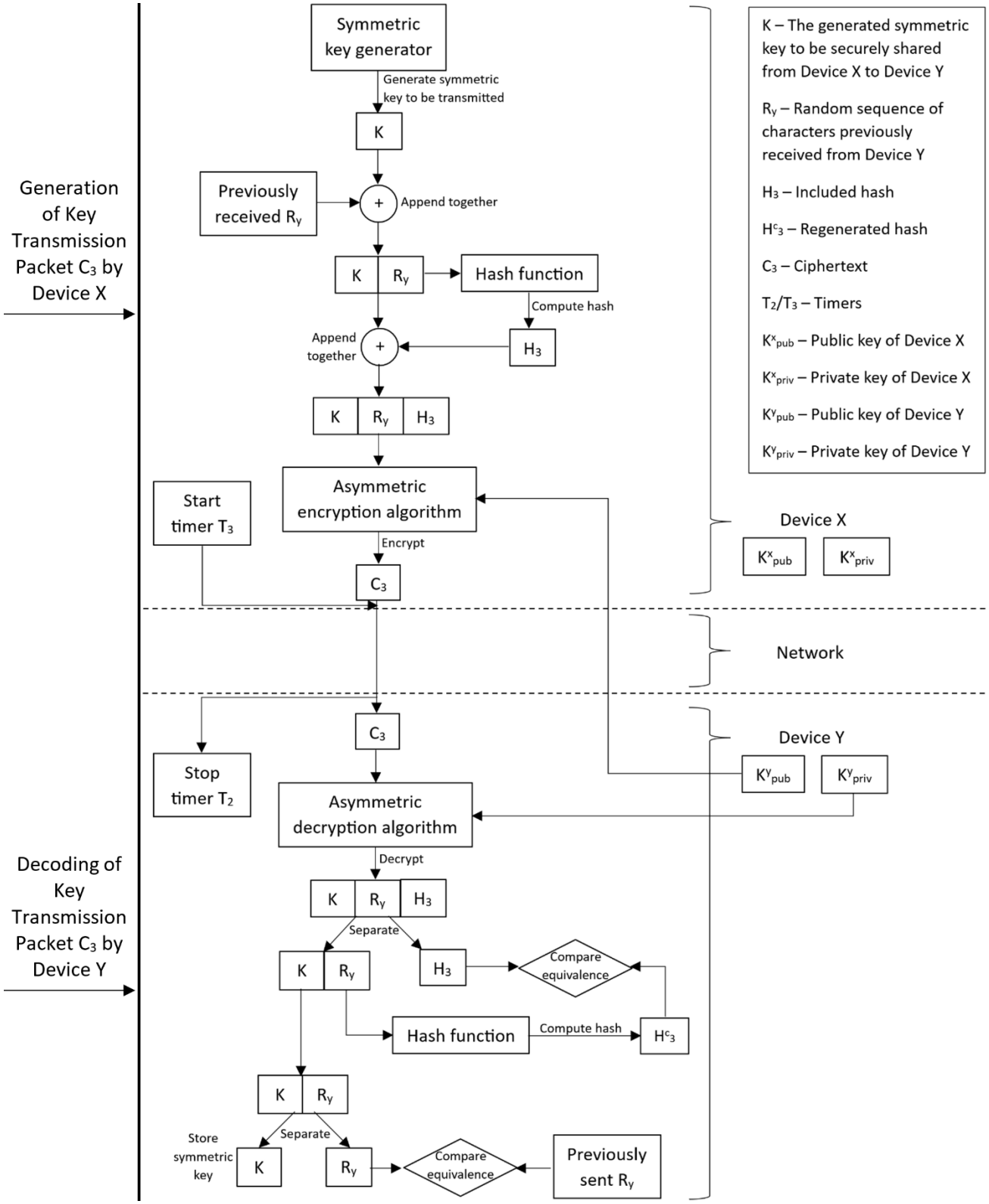

Figure 10. The detailed flow diagram that shows how Device X sends the key K to Device Y via C_3

Step 4: Generation and Decoding of Key Received Acknowledgement Packet (C_4):

Device Y creates the packet C_4, which Device X receives and decodes in the way that follows –

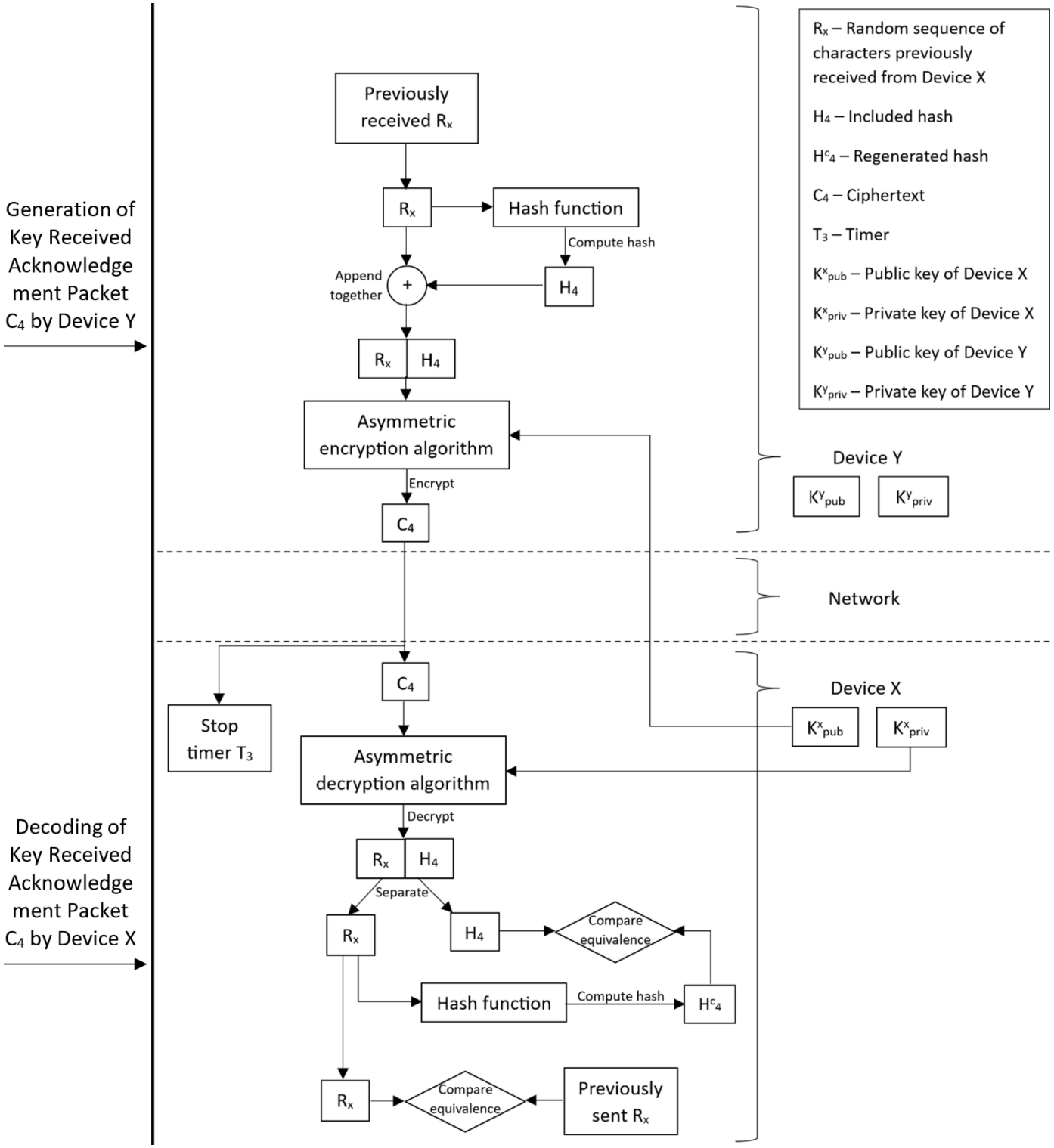

Figure 11. The detailed flowchart demonstrating how Device Y uses C_4 to acknowledge that Device X has transmitted the key K

III. Discussions on Methodology

Prior to being transmitted over the network, each message's contents are encrypted to create the cipher-texts C_1, C_2, C_3, and C_4. Using the public key of the recipient, encryption is carried out using asymmetric key cryptography. By utilizing its own private key, the recipient decrypts this cipher-text. Device X, for instance, encrypts the messages it sends to Device Y using the latter's public key, and Device Y uses its own private key to decrypt them. With our approach, we can use any asymmetric technique, but we have used RSA as it is more popular.

The benefit of employing asymmetric key cryptography in this situation is –

- The message's confidentiality is preserved since, aside from Device Y, no other person has access to the private key needed to decrypt its contents.

R_x and R_y both are random strings or random sequences of characters. These are generated by each device using a modified Generative Adversarial Network (GAN) that serves as a Pseudo-Random Number Generator (PRNG), as shown by M. D. Bernardi in their study [26]. R_x and R_y are produced, respectively, by devices X and Y.

So, in our method, Device X sends R_x to Device Y using the first message, and Device Y sends R_y to Device X using the second message. The integrity and confidentiality of both messages guarantee that only the devices X and Y are aware of the real values of R_x and R_y. Both devices utilize encryption, which makes it impossible for any other device to decipher them, and the hash values are used to identify any tampering. Once the values of R_x and R_y are established, these can be used to verify each other's identity as the true sender of the message, by simply repeating these values back to each other.

For instance, Device X can authenticate itself by including R_y in a message that it sends to Device Y. Since only devices X and Y are aware of R_y's value, only Device X can be the message sender if Device Y compares the R_y value found in the message to its own R_y value, and determines that the two are equivalent.

The benefit of utilizing random strings is –

- Since X and Y are the only devices that know the values of R_x and R_y, they can verify each other's identities by repeating these values back to each other. As a result, our approach preserves the message sender's authenticity, so that the recipient can confirm it.

Timers T_1, T_2, and T_3, and acknowledgments are employed here to further give the message sender an additional means of authenticating, and confirming whether a message has reached the intended recipient.

Apart from the final message, the sender initiates a timer upon sending, and ends it upon getting a reply message. Senders can presume that their communication was not received by the recipient if they do not receive a response within a predetermined time frame. It is important to carefully select the timer's duration such that it gives the recipient enough time to receive, process, and reply, but not enough time for an attacker to brute force the cipher-text. For example, artificial intelligence (AI)-based channel estimation techniques can be used to establish the timer's duration, such as by predicting the network channel's characteristics and modifying the timer accordingly.

Furthermore, a message that comes after is interpreted as an acknowledgment of the one that came before. As a result, the initial message from Device X to Device Y is acknowledged in the second message from Device Y to Device X. The second message from Device Y to Device X is acknowledged by the third message from Device X to Device Y, and the third message from Device X to Device Y is acknowledged by the final and fourth message from Device Y to Device X.

Device X's timer will expire and it will become aware of the situation, for instance, if an attacker prevents a communication from Device X from reaching Device Y. In case the attacker not only intercepts, but also attempts to send a reply message pretending to be Device Y, Device X will be able to determine that the message was not sent by Device Y, but rather by someone else by comparing the random strings. In the unlikely event that the attacker attempts to pose as Device Y more successfully by brute-forcing the cipher-text, obtaining data, and then sending a reply message, the timer will expire well before it is feasible to accomplish so, because it is designed to prevent situations like this from happening.

The benefit of our method's use of timers and acknowledgments is –

- The sender can determine if the messages are reaching the intended recipient. Our approach thereby preserves the message recipient's authenticity, allowing the sender to confirm it.

There is no mention of a validity time for the shared symmetric key in our method. However, because keys must be changed on a regular basis, adding a validity period can increase security. For maximum security, AI-based methods could be used to forecast the length of this validity period. Once more, we leave this aspect open for any further research.

IV. Results and Conclusion

Finally, we provide a few straightforward real-world examples to show how our unique technique outperforms the current approaches. Together with the two current methods, we evaluate our innovative approach in similar conditions to demonstrate how our approach may effectively address these problems, and how the prior ways can fail in specific situations.

- Asymmetric encryption for key sharing (using RSA) –

  Figure 12 illustrates the issue with traditional RSA asymmetric key encryption, in which an integrity attack on the compromised node modifies the message, but leaves the final checksum unaltered. As a result of the checksum validation failing to detect the change, Device Y is given the wrong key value. Therefore, this approach does not identify the integrity violation.

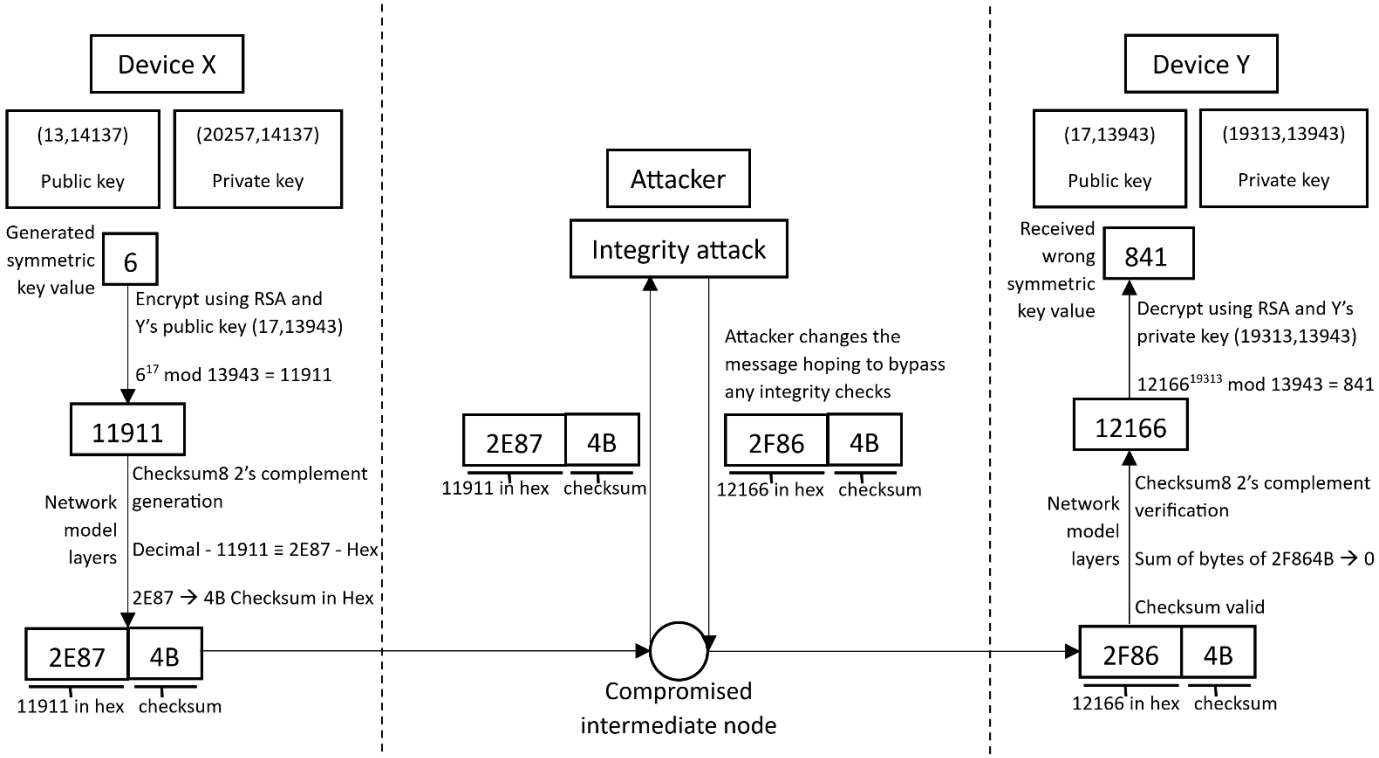

Figure 12. An instance of key sharing using asymmetric encryption

- Diffie-Hellman key exchange protocol –

  Figure 13 illustrates the issue with DH key exchange. The message containing result A is altered by the integrity attack in the compromised node, but the resultant checksum remains unchanged. As a result, Device Y receives the incorrect value of A, due to the checksum validation failing to identify the change in value, which ultimately causes different values of s to be generated on both sides. Thus, the integrity breach was also missed by this method.

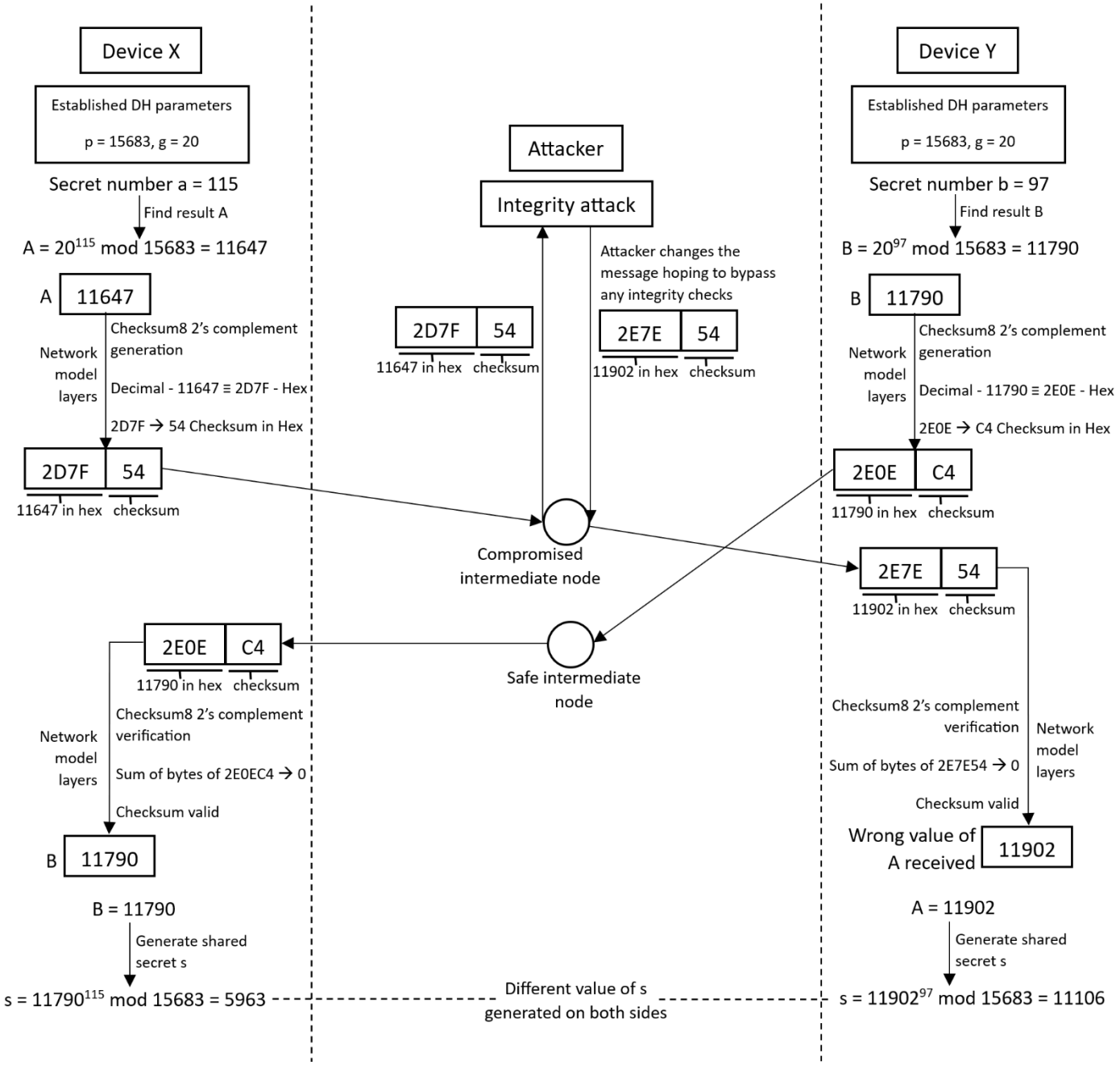

Figure 13. An illustration of the DH key exchange mechanism in use for key sharing

- Our suggested approach (using RSA and SHA-256) –

  Figure 14 shows our suggested approach. Here, the third message is illustrated for a circumstance that is comparable to the other approaches. Once more, the integrity attack modifies the message while maintaining the same checksum as a result. The addition of a hash value, and the subsequent regeneration and comparison of the hash can, nevertheless, discover the change in the message, even though the checksum validation is unable to identify it. Device Y thus notices the integrity breach and deletes the changed message.

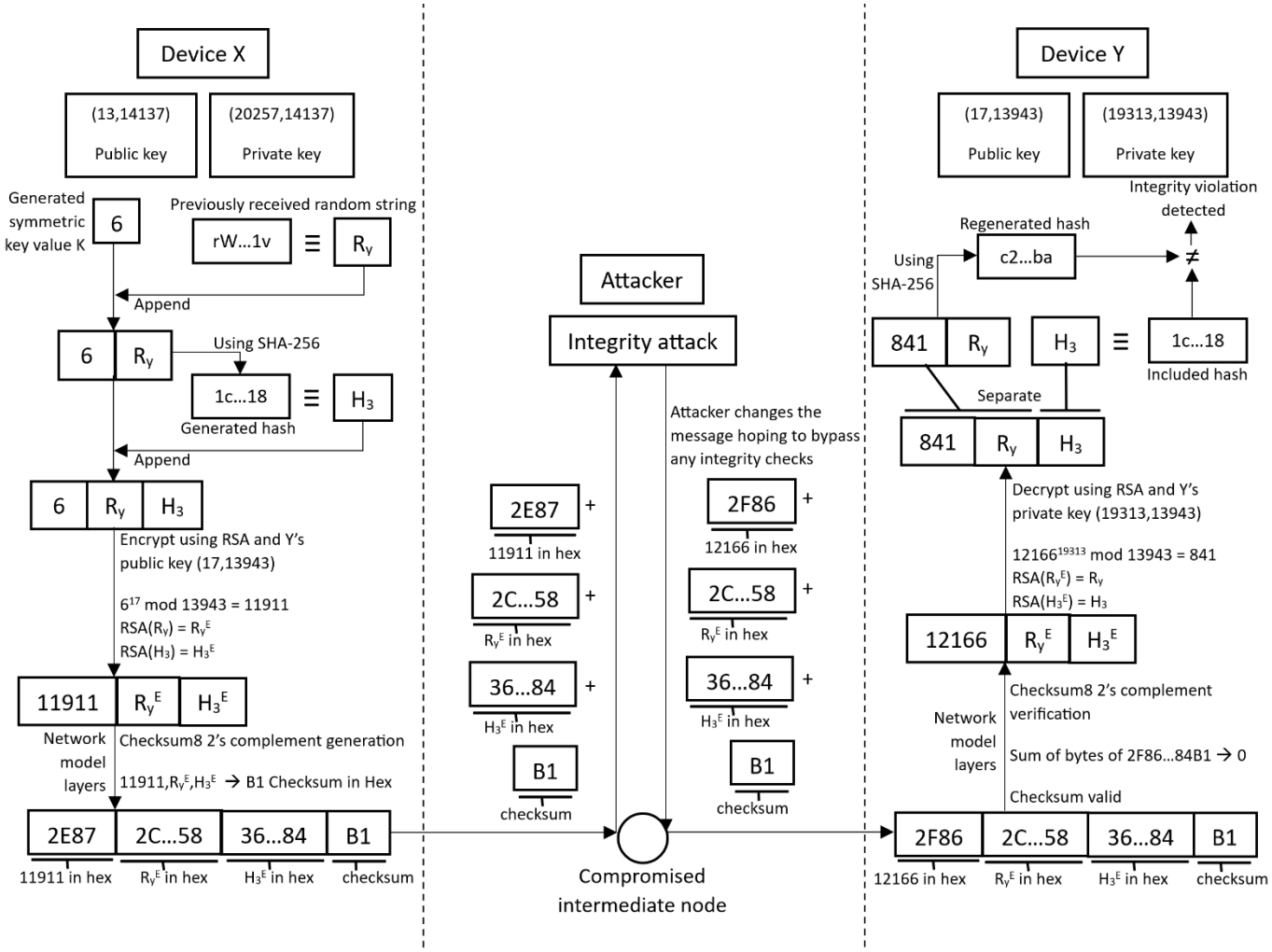

Figure 14. An illustration of key sharing utilizing our brand-new, suggested approach

The examples demonstrate how our approach, which employs hash functions, can offer resilience to integrity related attacks, and so improve integrity overall, something that the other approaches are unable to achieve.

Additionally, by using random strings, our approach may detect the possibility that an attacker has altered the entire message, by removing the original message and delivering a new one in its place. As an attacker's chances of successfully guessing the random string is extremely low, they will be unable to assume the identity of the original sender.

Our technique's timers and implicit acknowledgments – which allow the sender to confirm whether a message has reached the intended recipient or not – are configured such that, even with network delays, the timer's duration is sufficient for the intended recipient to respond by sending back a subsequent message that serves as an acknowledgment. However, it is not long enough for an attacker to attempt any kind of deception, or compromise the method's security in any way. Thus, setting up the timer correctly can further improve our method's security.

Table I summarizes the results of our analysis. It compares the two currently used methods against our new suggested method, in terms of how well they maintain these three important properties of network security – confidentiality, integrity, and authenticity. It demonstrates the advantages provided by our suggested approach.

TABLE I. COMPARISON OF THE TWO EXISTING METHODS AND OUR NEW SUGGESTED METHOD

Properties maintained	**Key-sharing methods**		
	Asymmetric encryption for key sharing	***Diffie-Hellman key exchange protocol***	***Our suggested method***
Confidentiality	Yes (Completely)	Yes (Completely)	Yes (Completely)
Integrity	Yes (Partially)[a]	Yes (Partially)[a]	Yes (Completely)
Authenticity	No	No	Yes (Completely)

a. Fails to detect integrity attacks as shown in figure 12 and 13.

Even though our approach requires more messages than other current approaches to finish the key sharing process, we think that the additional network usage is not a problem, because current 5G infrastructure enables fast and low-latency communications, which will be even faster in 5G and beyond networks. Thus, the drawback of our method's usage of more network resources is outweighed by the expansion of communication infrastructures. By utilizing 5G's benefits to offset its possible drawbacks, our approach enhances current 5G infrastructure.

Based on the analysis, we can conclude that our proposed method not only offers confidentiality in the same way as the currently used methods, but also offers several improvements over the currently used methods, including improved integrity, and authenticity, which allows the recipient to confirm the sender's identity, and the sender to determine whether their messages are received. Therefore, in symmetric key cryptography, our suggested method offers a considerably stronger process for safe symmetric key sharing.

REFERENCES

[1] M. Zubair, M. Ahmed, A. Ali, S. Naeem, and S. Anam, "Network Security and Cryptography Challenges and Trends on Recent Technologies," Journal of Applied and Emerging Sciences, vol. 13, no. 1, pp. 1–8, 2023.

[2] M.Biswas, "THE ROLE OF CRYPTOGRAPHY TOWARDS NETWORK SECURITY," International Journal of Multidisciplinary Educational Research, vol. 11, no. 3(7), pp. 59–64, 2022.

[3] S. Chandra, S. Paira, S. S. Alam, and G. Sanyal, "A comparative survey of Symmetric and Asymmetric Key Cryptography," 2014 International Conference on Electronics, Communication and Computational Engineering (ICECCE), Hosur, India, pp.83–93, 2014, doi: 10.1109/ICECCE.2014.7086640.

[4] A. Kakkar, "A survey on secure communication techniques for 5G wireless heterogeneous networks," Information Fusion, vol. 62, pp. 89–109, 2020, doi: 10.1016/j.inffus.2020.04.009.

[5] D. Bera, B. Roy, and D. Pal, "Network Security and Cryptography: A Review," International Journal of Advances in Engineering and Management, vol. 3, no. 7, pp. 4172–4176, 2021.

[6] M. Sharma, "Review on Cryptography in Network Security," International Journal of Engineering Research & Technology, vol. 2, no. 3, pp. 206–213, 2014.

[7] F. Maqsood, M. Ahmed, M. M. Ali, and M. A. Shah, "Cryptography: A Comparative Analysis for Modern Techniques," International Journal of Advanced Computer Science and Applications, vol. 8, no. 6, pp. 442–448, 2017, doi: 10.14569/IJACSA.2017.080659.

[8] E. Rawat, A. Singh, A. Mahar, and A. Agarwal, "A Review- Paper on Cryptography and Network Security," EasyChair Preprint no. 7952, 2022, https://easychair.org/publications/preprint/2f5q.

[9] M. Al-Shabi, "A Survey on Symmetric and Asymmetric Cryptography Algorithms in information Security," International Journal of Scientific and Research Publications (IJSRP), vol. 9, pp. 576–589, 2019, doi: 10.29322/IJSRP.9.03.2019.p8779.

[10] C. Singh, and L. Kaur, "A REVIEW OF DIFFERENT APPROACHES FOR IMPROVING NETWORK SECURITY IN CRYPTOGRAPHY," Turkish Journal of Computer and Mathematics Education, vol. 12, no. 1, pp. 819–823, 2021.

[11] M. U. Bokhari, and Q. M. Shallal, "A Review on Symmetric Key Encryption Techniques in Cryptography," International Journal of Computer Applications, vol. 147, no. 10, pp. 43–48, 2016, doi: 10.5120/ijca2016911203.

[12] S. Kumar, M. S. Gaur, P. S. Sharma, and D. Munjal, "A Novel Approach of Symmetric Key Cryptography," 2021 2nd International Conference on Intelligent Engineering and Management (ICIEM), London, United Kingdom, pp. 593–598, 2021, doi: 10.1109/ICIEM51511.2021.9445343.

[13] S. Vyakaranal, and S. Kengond, "Performance Analysis of Symmetric Key Cryptographic Algorithms," 2018 International Conference on Communication and Signal Processing (ICCSP), Chennai, India, pp. 0411–0415, 2018, doi: 10.1109/ICCSP.2018.8524373.

[14] M. Sohal, and S. Sharma, "BDNA-A DNA inspired symmetric key cryptographic technique to secure cloud computing," Journal of King Saud University - Computer and Information Sciences, vol. 34, no. 1, pp. 1417–1425, 2022, doi: 10.1016/j.jksuci.2018.09.024.

[15] V. Cortier, and G. Steel, "A generic security API for symmetric key management on cryptographic devices," Information and Computation, vol. 238, pp. 208–232, 2014, doi: 10.1016/j.ic.2014.07.010.

[16] A. Soleimanian, and S. Khazaei, "Publicly verifiable searchable symmetric encryption based on efficient cryptographic components," Des. Codes Cryptography, vol. 87, no. 1, pp. 123–147, 2019, doi: 10.1007/s10623-018-0489-y.

[17] R. Masram, V. Shahare, J. Abraham, and R. Moona, "Analysis and Comparison of Symmetric Key Cryptographic Algorithms Based on Various File Features," International Journal of Network Security & Its Applications, vol. 6, pp. 43–52, 2014, doi: 10.5121/ijnsa.2014.6404.

[18] S. Kansal, and M. Mittal, "Performance evaluation of various symmetric encryption algorithms," 2014 International Conference on Parallel, Distributed and Grid Computing, Solan, India, pp. 105–109, 2014, doi: 10.1109/PDGC.2014.7030724.

[19] P. Singh, and P. Shende, "Symmetric Key Cryptography: Current Trends," International Journal of Computer Science and Mobile Computing, vol. 3, no. 12, pp. 410–415, 2014.

[20] A. Maetouq, S. M. Daud, N. A. Ahmad, N. Maarop, N. N. A. Sjarif, and H. Abas, "Comparison of Hash Function Algorithms Against Attacks: A Review," International Journal of Advanced Computer Science and Applications, vol. 9, no. 8, pp. 98–103, 2018, doi: 10.14569/IJACSA.2018.090813.

[21] A. Taparia, S. K. Panigrahy, and S. K. Jena, "Secure key exchange using enhanced Diffie-Hellman protocol based on string comparison," 2017 International Conference on Wireless Communications, Signal Processing and Networking (WiSPNET), Chennai, India, pp. 722–726, 2017, doi: 10.1109/WiSPNET.2017.8299856.

[22] M. A. Latif, M. B. Ahmad, and M. K. Khan, "A Review on Key Management and Lightweight Cryptography for IoT," 2020 Global Conference on Wireless and Optical Technologies (GCWOT), Malaga, Spain, pp. 1–7, 2020, doi: 10.1109/GCWOT49901.2020.9391613.

[23] M. AlRoubiei, T. AlYarubi, and B. Kumar, "Critical Analysis of Cryptographic Algorithms," 2020 8th International Symposium on Digital Forensics and Security (ISDFS), Beirut, Lebanon, pp. 1–7, 2020, doi: 10.1109/ISDFS49300.2020.9116213.

[24] A. Hamza, and B. Kumar, "A Review Paper on DES, AES, RSA Encryption Standards," 2020 9th International Conference System Modeling and Advancement in Research Trends (SMART), Moradabad, India, pp. 333–338, 2020, doi: 10.1109/SMART50582.2020.9336800.

[25] H. Xu, K. Thakur, A. S. Kamruzzaman, and M. L. Ali, "Applications of Cryptography in Database: A Review," 2021 IEEE International IOT, Electronics and Mechatronics Conference (IEMTRONICS), Toronto, Canada, pp. 1–6, 2021, doi: 10.1109/IEMTRONICS52119.2021.9422663.

[26] M. D. Bernardi, M. H. R. Khouzani, and P. Malacaria, "Pseudo-Random Number Generation Using Generative Adversarial Networks," ECML PKDD 2018 Workshops, Cham, Springer International Publishing, pp. 191–200, 2019, doi: 10.1007/978-3-030-13453-2_15.

[27] A. Argabi, and M. Alam, "A new Cryptographic Algorithm AEDS (Advanced Encryption and Decryption Standard) for data security," International Advanced Research Journal in Science, Engineering and Technology, vol. 6, pp 1–7, 2019, doi: 10.17148/IARJSET.2019.61001.

[28] L. Zhou, J. Chen, Y. Zhang, C. Su, and M. A. James, "Security analysis and new models on the intelligent symmetric key encryption," Computers & Security, vol. 80, pp. 14–24, 2019, doi: 10.1016/j.cose.2018.07.018.

[29] M. Asgari-Chenaghlu, M. A. Balafar, and M. R. Feizi-Derakhshi, "A novel image encryption algorithm based on polynomial combination of chaotic maps and dynamic function generation," Signal Processing, vol. 157, pp. 1–13, 2019, doi: 10.1016/j.sigpro.2018.11.010.

[30] B. Zahednejad, L. Ke, and J. Li, "A Novel Machine Learning-Based Approach for Security Analysis of Authentication and Key Agreement Protocols," Security and Communication Networks, pp. 1–15, 2020, doi: 10.1155/2020/8848389.

[31] N. A. Gunathilake, W. J. Buchanan, and R. Asif, "Next Generation Lightweight Cryptography for Smart IoT Devices: : Implementation, Challenges and Applications," 2019 IEEE 5th World Forum on Internet of Things (WF-IoT), Limerick, Ireland, pp. 707–710, 2019, doi: 10.1109/WF-IoT.2019.8767250.

[32] A. Shah, and M. Engineer, "A Survey of Lightweight Cryptographic Algorithms for IoT-Based Applications," Smart Innovations in Communication and Computational Sciences, Springer Singapore, pp. 283–293, 2019, doi: 10.1007/978-981-13-2414-7_27.

[33] E. Jincharadze, "CRITICAL ANALYSIS OF SOME CRYPTOGRAPHY ALGORITHMS," Scientific and Practical Cyber Security Journal (SPCSJ), vol. 1, no. 2, pp. 60–68, 2017.

[34] Y. Alemami, M. A. Mohamed, and S. Atiewi, "Research on Various Cryptography Techniques," International Journal of Recent Technology and Engineering, vol. 8, pp. 395–405, 2019, doi: 10.35940/ijrte.B1069.0782S319.

[35] M. F. Mushtaq, S. Jamel, A. H. Disina, Z. A. Pindar, N. S. A. Shakir, and M. M. Deris, "A Survey on the Cryptographic Encryption Algorithms," International Journal of Advanced Computer Science and Applications, vol. 8, no. 11, pp. 333–344, 2017, doi: 10.14569/IJACSA.2017.081141.

[36] S. Soomro, M. R. Belgaum, Z. Alansari, and R. Jain, "Review and Open issues of Cryptographic Algorithms in Cyber Security," 2019 International Conference on Computing, Electronics & Communications Engineering (iCCECE), London, UK, pp. 158–162, 2019, doi: 10.1109/iCCECE46942.2019.8941663.

[37] S. Thorat, R. Sharma, and S. Pansare, "ANALYSIS OF SYMMETRIC KEY CRYPTOGRAPHIC ALGORITHMS," International Research Journal of Engineering and Technology (IRJET), vol. 4, no. 2, pp. 1628–1630, 2017.

[38] R. Andriani, S. E. Wijayanti, and F. W. Wibowo, "Comparision Of AES 128, 192 And 256 Bit Algorithm For Encryption And Description File," 2018 3rd International Conference on Information Technology, Information System and Electrical Engineering (ICITISEE), Yogyakarta, Indonesia, pp. 120–124, 2018, doi: 10.1109/ICITISEE.2018.8720983.

[39] M. Abu-Faraz, and Z. A. Qadi, "Using Highly Secure Data Encryption Method for Text File Cryptography," International Journal of Computer Science and Network Security, vol. 21, pp. 53–60, 2021, doi: 10.22937/IJCSNS.2021.21.12.8.

[40] J. A. P. Artiles, D. P. B. Chaves, and C. Pimentel, "Image encryption using block cipher and chaotic sequence," Signal Processing: Image Communication, vol. 79, pp. 24–31, 2019, doi: 10.1016/j.image.2019.08.014.