International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 9
***Article Received:** 25 July 2023 **Revised:** 12 September 2023 **Accepted:** 30 September 2023*

Improve the Onion Routing Performance and Security with Cryptographic Algorithms

T.Gayathri[1], Dr.A.Saraswathi[2]
[1]Research Scholar ,PG and Research Department Computer Science, Govt. Arts College(A)(Affiliated to Bharathidasan University, Trichy-24), Karur,Tamil Nadu, India.
E-mail : gacgayathri@gmail.com
[2]Associate Professor, PG and Research Department Computer Science, Govt. Arts College(A)(Affiliated to Bharathidasan University, Trichy-24), Karur,Tamil Nadu, India.
E-mail: sarasdharam78@gmail.com

Abstract:

Onion Routing and Cryptographic Algorithms are two essential components of online privacy and secure data transmission. Onion Routing is a technique used to protect internet users' anonymity by routing their communication through a network of servers, while Cryptographic Algorithms are used to encrypt and decrypt data to ensure its confidentiality. As technology advances, there is a need to consider the development of new cryptographic algorithms for TOR to ensure its continued effectiveness. The combination of Onion Routing and Cryptographic Algorithms has proven to be an effective way to protect online privacy and security. This paper aims to explore the benefits of combining Onion Routing and Cryptographic Algorithms and to propose a hybrid symmetric and hashing algorithm technique to transmit data securely. By the end of this paper, researchers will have a comprehensive understanding of the Onion Routing and Cryptographic Algorithms, their implementation in TOR, and the limitations and risks associated with using such tools.

Keywords: Onion Routing , Cryptographic Algorithms, encrypt and decrypt data.

1. Introduction:

Botnets are a significant threat in the cybersecurity landscape, capable of causing large-scale damage and compromising the security of individuals and organizations alike. To combat this growing menace, various security measures have been implemented, one of which is the use of Onion Routing (TOR) combined with cryptographic algorithms.

Onion Routing, also known as The Onion Routing Protocol, is a technique for anonymous communication over a computer network. The name "Onion Routing" comes from the addition of layers at each stage, where the protocol adds layers for each stage, and the layers of encryption are analogous to layers of an onion [1]. In an onion network, messages are encapsulated in layers of encryption, and the sender remains anonymous because each intermediary knows only the location of the immediately preceding and following nodes [2]. The encrypted data is transmitted through a series of network nodes called "onion routers," each of which "peels" away a single layer, revealing the data's next destination. When the final layer is decrypted, the message arrives at its destination [3]. Onion Routing provides a high level of security and anonymity, making it useful for web browsing, secure shell, instant messaging, and other applications. Onion Routing can be used via Tor, a distributed overlay network designed to anonymize TCP-based applications [4]. While Onion Routing is designed to prevent anyone from tracking the data's origin, methods to break the anonymity of this technique include timing analysis. Garlic routing is a variant of onion routing used in the I2P network. It encrypts multiple messages together, which increases the speed of data transfer and makes it more difficult for attackers to perform traffic analysis. The step-by-step process of Onion Routing is shown in the figure 1.

.

Figure 1. Step-by-Step process of Onion Routing

1. Initialization: The user's device establishes a connection to the Tor network, selecting an entry relay as the starting point for the communication.
2. Encryption: The user's message is encrypted multiple times, each layer encapsulating the original message with a new layer of encryption. These layers are akin to the layers of an onion.
3. Routing: The encrypted message is transmitted through a series of randomly selected relays, with each relay only aware of the previous and next

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 9
***Article Received:** 25 July 2023 **Revised:** 12 September 2023 **Accepted:** 30 September 2023*

relay in the circuit.

4. Decryption: As the message passes through each relay, one layer of encryption is removed, revealing the next relay in the circuit. This process continues until the message reaches the final destination.
5. Exit Node: The final relay, known as the exit node, decrypts the last layer of encryption and transmits the message to its intended recipient on the regular internet.

Cryptographic algorithms are widely used in public key cryptography, which is a set of cryptographic protocols that use these algorithms to encrypt messages securely. These algorithms require two separate keys, a public key, and a private key [5]. When choosing a cryptographic algorithm, usability is an important consideration. The choice of cryptographic algorithm involves a balancing act between security and usability. Cryptographic algorithms can be either symmetric key or public key. Symmetric key encryption is less computationally expensive than public key encryption and is useful in a network. In symmetric key encryption, the same key is used for both encryption and decryption. The Advanced Encryption Standard (AES) is a symmetric key crypto-system commonly used for encryption [5]. Public key cryptography can be slower but more secure than symmetric key cryptography. Public key is used to encrypt plaintext, which anyone can encrypt for a specific person. Private key is used to decrypt encrypted text, which only the intended recipient can decrypt [6]. The key for encryption is agreed upon using Diffie-Hellman. Cryptographic algorithms include asymmetric key cryptography, which is used in digital signatures to establish authenticity and integrity. Digital signatures are more secure than physical written signatures and are a digital version of a physical signature. The person with the private key can produce valid digital signatures. The key pair consists of a private key and a public key. Cryptographic algorithms involve the use of key pairs for encryption. A hidden service uses asymmetric encryption for its key pair. Onion routing is one such application that uses cryptographic algorithms in anonymous communication over a computer network. In Onion routing, messages are encapsulated in layers of encryption, and the sender remains anonymous because each intermediary knows only the location of the immediately preceding and following nodes [7].

Onion routing and cryptographic algorithms have unique benefits on their own, but when they are combined, they can provide even stronger security measures. Onion routing uses encryption and routing data through multiple servers to provide enhanced security [8]. Combining onion routing with cryptographic algorithms can add extra layers of protection against attackers, making it more difficult for them to breach the system. The combined use of onion routing and cryptographic algorithms makes traffic analysis difficult for attackers, preventing them from identifying the source and destination of communication as well as analyzing traffic patterns. The decentralized nature of onion routing also makes it more resistant to attacks. This combination can enhance the privacy and security of network communication by increasing resilience to attacks [9]. By hiding the original message as it is transferred from one node to the next, the sender can remain anonymous as no intermediary knows both the origin and final destination of the data [10]. Overall, combining onion routing and cryptographic algorithms provides a powerful defense against cyber threats.

The rest of the paper is organised as follows, section 2 provides a brief overview of existing works, section 3 provides a detailed explanation on the proposed work, section 4 test the algorithm with a sample message and finally section 5 concludes the work.

2. Literature Survey

TOR cryptographic algorithms play a crucial role in ensuring the privacy and security of users. These algorithms are used to encrypt the data multiple times and route it through multiple nodes in the network, making it difficult for anyone to trace the origin of the data [11]. TOR employs various cryptographic algorithms, including RSA, AES, and Diffie-Hellman, to ensure that the data is securely transmitted through the network [12]. Solutions based on onion routing use public key cryptography and an application-layer overlay routing to provide sender and receiver anonymity [13]. The importance of TOR cryptographic algorithms in ensuring privacy and security cannot be overstated, as they are essential in protecting sensitive information from being intercepted and compromised. The development of TOR cryptographic algorithms has been the subject of extensive research, with many studies focused on identifying and addressing potential vulnerabilities and attacks [14][15]. The interdisciplinary FASOR project, which aims to develop faster and stronger onion routing protocols, is an example of the ongoing efforts to improve the security and privacy of onion routing [16].

2.1 Symmetric Cryptography Algorithms Used in TOR

Symmetric cryptography is an essential component of the onion routing (TOR) cryptographic algorithms. Symmetric cryptography is a technique that uses a single key to encrypt and decrypt data, making it faster and more efficient than asymmetric cryptography [17]. In TOR, symmetric cryptography is used to encrypt the relay header and

payload, which are then decrypted at each relay in the network [15]. This technique ensures that the data remains secure and confidential throughout its journey through the TOR network. There are different types of symmetric cryptography algorithms used in TOR, including the Advanced Encryption Standard (AES) and the Twofish algorithm [18]. These algorithms are chosen for their ability to provide high levels of security and efficiency in data transmission. The use of symmetric cryptography in TOR helps to ensure the anonymity of users and protect their privacy. While symmetric cryptography has many advantages, it also has some limitations when used in TOR. One of the main disadvantages is that it can result in slow performance due to the multiple layers of encryption and decryption involved in the TOR network. Additionally, the use of symmetric cryptography may not be sufficient to protect against potential attacks from quantum computers, which could potentially break the encryption used in TOR [19]. Despite these limitations, the use of symmetric cryptography remains an important aspect of the TOR network and its ability to provide anonymous communication over the internet.

2.2 Asymmetric Cryptography Algorithms Used in TOR

Asymmetric cryptography, also known as public-key cryptography, is a type of cryptographic algorithm used in onion routing (TOR) to ensure secure communication. Asymmetric cryptography involves the use of two different keys, a public key and a private key, for encryption and decryption [20]. The public key is widely distributed and used to encrypt messages, while the private key is kept secret and used to decrypt messages. This technique allows for secure communication without the need for both parties to share a secret key. In TOR, various types of asymmetric cryptography algorithms are used, including RSA, Diffie-Hellman, and Elliptic Curve Cryptography (ECC) [21]. RSA is the most commonly used algorithm in onion routing, as it provides strong security and is widely supported by cryptographic libraries and software [22]. Diffie-Hellman and ECC are also used in certain situations, such as when there are limitations on the size of the keys that can be used [23]. Each of these algorithms has its own strengths and weaknesses, and their use in TOR depends on the specific needs of the communication. The use of asymmetric cryptography in onion routing provides several advantages, such as secure communication without the need for a shared secret key and protection against eavesdropping and traffic analysis. However, there are also some disadvantages, such as slow performance due to the multiple layers of encryption and the need for key management. Overall, the use of asymmetric cryptography in onion routing is essential for ensuring secure communication and protecting the privacy of users.

2.3 Hashing Algorithms Used in TOR

Hashing is a fundamental cryptographic technique used in onion routing (TOR) to provide anonymity and confidentiality to users. Hashing is a process of converting data of arbitrary size into a fixed-size output, known as a hash value or message digest. This output is unique to the input data and cannot be reversed to obtain the original input. In TOR, hashing algorithms are used to provide integrity, authenticity, and non-repudiation of messages [24]. There are several types of hashing algorithms used in TOR, including SHA-1, SHA-2, and SHA-3. SHA-1 is a widely used hashing algorithm that produces a 160-bit hash value. However, due to its vulnerability to collision attacks, SHA-1 is being phased out in favor of more secure algorithms such as SHA-2 and SHA-3. SHA-2 produces hash values of various sizes, including 224, 256, 384, and 512 bits, while SHA-3 produces hash values of 224, 256, 384, and 512 bits [1]. The use of hashing algorithms in TOR provides several advantages, including data integrity, message authentication, and non-repudiation. However, there are also some disadvantages associated with hashing in TOR, such as slow performance due to the multiple layers of encryption and the computational overhead of hashing [25]. In conclusion, hashing is a crucial component of onion routing (TOR) and plays a significant role in ensuring the privacy and security of users' data.

2.4 Key Exchange Algorithms Used in TOR

Key exchange is a critical component of onion routing, which is a technique used for anonymous communication over a computer network. Key exchange is the process of securely sharing cryptographic keys between two parties to enable encrypted communication. In onion routing, the required encryption keys are established using a key exchange algorithm, which helps to ensure the confidentiality and integrity of the communication [26]. There are several types of key exchange algorithms used in TOR, including Diffie-Hellman (DH), Elliptic Curve Diffie-Hellman (ECDH), and RSA. DH is a widely used key exchange algorithm that allows two parties to establish a shared secret key over an insecure communication channel [27]. ECDH is a variant of DH that uses elliptic curve cryptography to provide better security with smaller key sizes [28]. RSA is another popular key exchange algorithm that uses public key cryptography to establish a shared secret key [29]. Each key exchange algorithm used in TOR has its own advantages and disadvantages. DH is fast and efficient, but it is vulnerable to man-in-the-middle attacks. ECDH provides better security with smaller key sizes, but it requires more computational resources. RSA is secure and

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 9
***Article Received:** 25 July 2023 **Revised:** 12 September 2023 **Accepted:** 30 September 2023*

widely used, but it is slower than DH and ECDH [30]. Despite these limitations, onion routing remains an essential tool for ensuring online privacy and security, and the use of key exchange algorithms is critical to its success.

3. Methodology:

The algorithm involves relays that route a message from a source to a destination, while encrypting and decrypting the message along the way. The proposed algorithm utilizes AES and Hashing function to provide enhanced security measures. The algorithm starts by defining a list of relays through which the message will be routed. It generates an original message and provides functions to encrypt the message with a relay's public key and decrypt it with a relay's private key. The main function, perform_onion_routing, implements the onion routing process. It iterates through the relays in reverse order, encrypting the message using the encrypt_message function and printing routing messages. It then iterates through the relays in forward order, simulating decryption using the decrypt_message function and printing decryption messages. Finally, it returns the final decrypted message. To complement the onion routing process, the algorithm provides functions for AES encryption and decryption. These functions handle AES encryption using the ECB mode, ensuring the message is securely encrypted and decrypted with a shared AES key. The algorithm generates a random AES key and encrypts the decrypted message using AES encryption. It also demonstrates the decryption of the encrypted message using AES decryption and the same AES key. By simulating the routing of a message through relays and encrypting/decrypting it using AES encryption, the algorithm showcases the process of onion routing and the usage of symmetric encryption for secure communication in a network environment.

1. Define a list of simulated network relays: `relays = [Relay1, Relay2, Relay3, Relay4, Relay5]`
    - This list represents the relays through which the message will be routed.
2. Generate a message to be sent: `message = "Hello, World!"`
    - This is the original message that needs to be sent from the source to the destination.
3. Define a function to encrypt the message with a relay's public key: `encrypt_message(message, public_key) = SHA256(public_key + message)`
    - This function takes the message and the public key of a relay as inputs. It concatenates the public key and the message. It applies the SHA256 hash function to the concatenated string, producing the encrypted message. The encrypted message is returned as the output.
4. Define a function to decrypt the message with a relay's private key: `decrypt_message(encrypted_message, private_key) = encrypted_message`
    - This function takes the encrypted message and the private key of a relay as inputs. It simply returns the encrypted message as it is, simulating decryption using the private key. In this simulation, the decryption operation does not change the encrypted message.
5. Define a function to perform onion routing:
    - This function takes the original message as input. It starts by setting the destination relay as the last relay in the list. It initializes the encrypted message with the original message. It iterates through the relays in reverse order. For each relay, it encrypts the message using the `encrypt_message` function. If the relay is not the destination relay, it prints a message indicating the routing. After the loop, it prints a message indicating that the message has reached the destination relay. It then iterates through the relays in forward order. For each relay, it simulates decryption using the `decrypt_message` function. If the relay is not the destination relay, it prints a message indicating that the relay has decrypted the message. Finally, it prints the final decrypted message and returns it.

```
perform_onion_routing(message):
    destination_relay = relays[-1]
    encrypted_message = message
a. for relay in reversed(relays):
    ▪ encrypted_message = encrypt_message(encrypted_message, relay)
    ▪ if relay ≠ destination_relay:
        print("Routing message to", relay)
print("Message successfully routed to the destination relay:", destination_relay)
b. for relay in relays:
```

- decrypted_message = decrypt_message(encrypted_message, relay)
- if relay ≠ destination_relay:

  print("Relay", relay, "decrypted the message:", decrypted_message)

print("Final decrypted message:", decrypted_message)

return decrypted_message

6. Perform onion routing with the message: `decrypted_message = perform_onion_routing(message)`
   - This step invokes the `perform_onion_routing` function with the original message as input. The function executes the routing and decryption steps, and the final decrypted message is assigned to the variable `decrypted_message`.
7. Define functions for AES encryption and decryption:
   - These functions handle AES encryption and decryption using the ECB (Electronic Codebook) mode. The `encrypt_aes` function takes the message and the AES key as inputs. It creates an AES cipher object with the provided key and ECB mode. It pads the message using the `pad` function from the `Crypto.Util.Padding` module to ensure it has a valid block size.It encrypts the padded message using the AES cipher object. The resulting ciphertext is then encoded in Base64 format to ensure it can be safely represented as a string. The encoded ciphertext is returned as the output. The `decrypt_aes` function takes the ciphertext and the AES key as inputs. It creates an AES cipher object with the provided key and ECB mode. It decodes the Base64-encoded ciphertext back to its binary representation. It decrypts the ciphertext using the AES cipher object. The resulting decrypted data is then unpadded using the `unpad` function from the `Crypto.Util.Padding` module. The unpadded plaintext is returned as the output.

1. encrypt_aes(message, key):
   a. cipher = AES.new(key, AES.MODE_ECB)
   b. ciphertext = cipher.encrypt(pad(message, AES.block_size))
   c. return Base64Encode(ciphertext)
2. decrypt_aes(ciphertext, key):
   a. cipher = AES.new(key, AES.MODE_ECB)
   b. decrypted = cipher.decrypt(Base64Decode(ciphertext))
   c. plaintext = unpad(decrypted, AES.block_size)
   d. return plaintext

8. Generate a random AES key: `aes_key = "ThisIsASecretKey"`
   - This step defines the AES key that will be used for encryption and decryption.In this case, the key is set to a fixed value for simplicity, but in practice, a random key should be generated.
9. Encrypt the decrypted message using AES encryption and the AES key: `encrypted_message_aes = encrypt_aes(decrypted_message, aes_key)`
   - This step invokes the `encrypt_aes` function with the decrypted message and the AES key as inputs. The function encrypts the message using AES encryption with the provided key. The resulting encrypted message is assigned to the variable `encrypted_message_aes`.
10. Decrypt the encrypted message using AES decryption and the AES key: `decrypted_message_aes = decrypt_aes(encrypted_message_aes, aes_key)`
    - This step invokes the `decrypt_aes` function with the encrypted message and the AES key as inputs. The function decrypts the message using AES decryption with the provided key. The resulting decrypted message is assigned to the variable `decrypted_message_aes`.

4. Case Study

Th**is section** demonstrates **the** presented algorithm a**s a** simulation of onion routing and AES encryption in a

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 9
***Article Received:** 25 July 2023 **Revised:** 12 September 2023 **Accepted:** 30 September 2023*

network setting. As an example, the botnet-tor is simulated with five relays namely Relay1, Relay2, Relay3, Relay4 and Relay5. The AES encryption algorithm is used to encrypt the message, and the key for the encryption is shared by all of the relays. The encrypted message is in the form of a base64 encoded string. The base64 encoding is used to convert binary data into a text format that can be easily routed to the destination relay over a network. The routing of the message is done in a round-robin fashion. The message is then decrypted by each relay as it passes through it. The message was routed to the destination relay in the following order:

- Original message: 1f193be4ab36873b759c0d6549783fe01cf689 5a3f6965eeb424c53c5bb48aab
    - Routing message to Relay4
    - Routing message to Relay3
    - Routing message to Relay2
    - Routing message to Relay1
    - Message successfully routed to the destination relay: Relay5
- Relay Relay1 decrypted the message: 1f193be4ab36873b759c0d6549783fe01cf689 5a3f6965eeb424c53c5bb48aab
- Relay Relay2 decrypted the message: 1f193be4ab36873b759c0d6549783fe01cf689 5a3f6965eeb424c53c5bb48aab
- Relay Relay3 decrypted the message: 1f193be4ab36873b759c0d6549783fe01cf689 5a3f6965eeb424c53c5bb48aab
- Relay Relay4 decrypted the message: 1f193be4ab36873b759c0d6549783fe01cf689 5a3f6965eeb424c53c5bb48aab
- Each relay decrypted the message using the same key, which was shared with all of the relays. This ensures that only the intended recipient can decrypt the message.
- Final decrypted message: 1f193be4ab36873b759c0d6549783fe01cf689 5a3f6965eeb424c53c5bb48aab
- The relays decrypted the message as it passed through them, and the final decrypted message was 1f193be4ab36873b759c0d6549783fe01cf689 5a3f6965eeb424c53c5bb48aab. This means that the AES encryption algorithm did not pad the message with any additional data. The original message is also the same as the final decrypted message, which means that the encryption and decryption process was successful.
- Encrypted Message (AES): +BZLytX1dwjCTwqbN0aQkWwWmFmxNu xtTweOMd/GSysUImlEUZ8M6fbMqyVh7g uK9LMaDLsgW118UzPpzbbSfvRRYGjkU7 7b/7tz3twFxRc=
- Decrypted Message (AES): 1f193be4ab36873b759c0d6549783fe01cf689 5a3f6965eeb424c53c5bb48aab

The final decrypted message is then received by Relay5. The original message is 16 bytes long, which is the same length as the encrypted message. Based on the output , it seems that the onion routing process was successfully executed, and the original message was decrypted correctly

5. Conclusion:

After reviewing the available literature on TOR cryptographic algorithms, it is evident that they are effective in ensuring online privacy and security.TOR's architecture and use of encryption techniques make it resistant to traffic analysis and provide anonymity to its users. As technology advances, there is a need to consider the development of new cryptographic algorithms for TOR to ensure its continued effectiveness. For example, post-quantum cryptographic primitives to make TOR safe in a quantum computing environment. Additionally, the FASOR project aims to develop faster and stronger onion routing protocols to improve TOR's performance and security. Therefore, it is essential to continue researching and developing new cryptographic algorithms to keep up with evolving threats to online privacy and security. In summary, onion routing and TOR cryptographic algorithms are essential tools in ensuring privacy and security in the digital age, and continued research and development in this area will be critical in addressing emerging threats and vulnerabilities.

References:

1. Chauhan, M., Singh, A.K. and Komal, 2020. Survey of onion routing approaches: advantages, limitations and future scopes. In Proceeding of the International Conference on Computer Networks, Big Data and IoT (ICCBI-2019) (pp. 686-697). Springer International Publishing.
2. Chen, C., Asoni, D.E., Barrera, D., Danezis, G. and Perrig, A., 2015, October. HORNET: High-speed onion routing at the network layer. In Proceedings of the 22nd ACM SIGSAC

Conference on Computer and Communications Security (pp. 1441-1454).
3. Jansen, R., Johnson, A. and Syverson, P., 2013. LIRA: Lightweight incentivized routing for anonymity. NAVAL RESEARCH LAB WASHINGTON DC.
4. Salib, E.H. and Hobar, G., 2018, October. Platform for teaching hands-on end-to-end anonymity algorithms. In 2018 IEEE Frontiers in Education Conference (FIE) (pp. 1-9). IEEE.
5. Basyoni, L., Fetais, N., Erbad, A., Mohamed, A. and Guizani, M., 2020, February. Traffic analysis attacks on Tor: A survey. In 2020 IEEE International Conference on Informatics, IoT, and Enabling Technologies (ICIoT) (pp. 183-188). IEEE.
6. Nazah, S., Huda, S., Abawajy, J. and Hassan, M.M., 2020. Evolution of dark web threat analysis and detection: A systematic approach. Ieee Access, 8, pp.171796-171819.
7. Haraty, R.A. and Zantout, B., 2014, June. The TOR data communication system: A survey. In 2014 IEEE Symposium on Computers and Communications (ISCC) (pp. 1-6). IEEE.
8. Sakai, K., Sun, M.T., Ku, W.S., Wu, J. and Alanazi, F.S., 2017. Performance and security analyses of onion-based anonymous routing for delay tolerant networks. IEEE Transactions on Mobile Computing, 16(12), pp.3473-3487.
9. Sakai, K., Sun, M.T., Ku, W.S., Wu, J. and Alanazi, F.S., 2016, June. An analysis of onion-based anonymous routing for delay tolerant networks. In 2016 IEEE 36th International Conference on Distributed Computing Systems (ICDCS) (pp. 609-618). IEEE.
10. Elgzil, A., Chow, C.E., Aljaedi, A. and Alamri, N., 2017, August. Cyber anonymity based on software-defined networking and Onion Routing (SOR). In 2017 IEEE conference on dependable and secure computing (pp. 358-365). IEEE.
11. Shirali, M., Tefke, T., Staudemeyer, R.C. and Pöhls, H.C., 2023. A Survey on Anonymous Communication Systems With a Focus on Dining Cryptographers Networks. IEEE Access, 11, pp.18631-18659.
12. Chen, X., 2020, August. Implementing AES encryption on programmable switches via scrambled lookup tables. In Proceedings of the Workshop on Secure Programmable Network Infrastructure (pp. 8-14).
13. Catalano, D., Di Raimondo, M., Fiore, D., Gennaro, R. and Puglisi, O., 2011. Fully non-interactive onion routing with forward-secrecy. In Applied Cryptography and Network Security: 9th International Conference, ACNS 2011, Nerja, Spain, June 7-10, 2011. Proceedings 9 (pp. 255-273). Springer Berlin Heidelberg.
14. Balasubramanian, K. and Kannan, S., 2019. Onion Routing in Anonymous Network. Appl. Math, 13(S1), pp.247-253.
15. Chen, X., 2020, August. Implementing AES encryption on programmable switches via scrambled lookup tables. In Proceedings of the Workshop on Secure Programmable Network Infrastructure (pp. 8-14).
16. Tiwari, S., Arora, D. and Singh, V., 2015. Implementation of Routing Protocol for Network and Data Security using Onion Routing with Salt Method. International Journal of Scientific and Research Publications, 5(7), pp.2250-3153.
17. Chen, X., 2020, August. Implementing AES encryption on programmable switches via scrambled lookup tables. In Proceedings of the Workshop on Secure Programmable Network Infrastructure (pp. 8-14).
18. Catalano, D., Di Raimondo, M., Fiore, D., Gennaro, R. and Puglisi, O., 2013. Fully non-interactive onion routing with forward secrecy. International journal of information security, 12(1), pp.33-47.
19. Tujner, Z., 2019. Quantum-safe TOR, post-quantum cryptography (Master's thesis, University of Twente).
20. Tujner, Z., 2019. Quantum-safe TOR, post-quantum cryptography (Master's thesis, University of Twente).
21. Adat, V. and Gupta, B.B., 2018. Security in Internet of Things: issues, challenges, taxonomy, and architecture. Telecommunication Systems, 67, pp.423-441.
22. Catalano, D., Fiore, D. and Gennaro, R., 2017. A certificateless approach to onion routing. International Journal of Information Security, 16, pp.327-343.
23. Begam, H.P. and Mohamed, M., 2012. Performance analysis of elliptic curve cryptography using onion routing to enhance the privacy and anonymity in grid computing. International Journal of Future Computer and Communication, 1(2), p.97.
24. Tujner, Z., Rooijakkers, T., van Heesch, M. and Önen, M., 2020. QSOR: Quantum-Safe Onion Routing. arXiv preprint arXiv:2001.03418.
25. Omar, Z.M. and Ibrahim, J., 2020. An overview of Darknet, rise and challenges and its assumptions. Int. J. Comput. Sci. Inf. Technol, 8, pp.110-116.
26. Moliya, D., Jain, R., Mohanty, V. and Hota, C., 2011. Probabilistic Anonymous Routing in Mobile Ad-Hoc Networks.
27. Conrad, B. and Shirazi, F., 2014, July. A Survey on Tor and I2P. In Ninth International Conference on Internet Monitoring and Protection (ICIMP2014) (pp. 22-28).
28. Fischer-Hbner, S. and Berthold, S., 2017. Privacy-enhancing technologies. In Computer and information security Handbook (pp. 759-778). Morgan Kaufmann.
29. Bao, L., Chen, R. and Sy, D., 2010. On-Demand Anonymous Routing (ODAR). In Security In Ad Hoc And Sensor Networks (pp. 137-157).
30. Vidal, G. and Moreno, J.L., 2018. Cryptography and Communications Privacy: An Introduction.