Fuzzy Systems and Soft Computing
ISSN : 1819-4362

TO ENRICH THE SPEED AND SECURITY OF CHACHA20 ALGORITHM THROUGH ROBUST KEY EXCHANGE PROTOCOLS

S. Julia little sha Research Scholar, PG and Research Department of Computer Science, Government Arts College (Autonomous) Nandanam, Chennai – 600035.
Dr. M. Ramesh Kumar Associate Professor & Head, PG and Research Department of Computer Science, Government Arts College (Autonomous) Nandanam, Chennai – 600035.

Abstract –
In today's world, where digital communication and data storage are booming, there's a huge demand for cryptographic tools that are both fast and secure. In this study, we suggest a new way to make the ChaCha20 algorithm faster and safer by teaming it up with the key exchange protocol. By putting them together, we hope to make a cryptographic tool that's both speedy and tough. We've done tests and analysis by python code to show that our idea works well, making ChaCha20 more secure without slowing it down. This could be a big step forward in keeping our digital stuff safe while keeping it fast.

I. Introduction

In today's digital world, where we share information online and store data on computers and phones, keeping our information safe is really important. Cryptographic algorithms are like secret codes that protect our messages and data from being seen by people who shouldn't see them. One popular algorithm, called ChaCha20, is known for being fast and efficient. It's used in things like messaging apps and securing data on our devices.

However, even though ChaCha20 is fast, there are some concerns about how safe it is against certain kinds of attacks. Also, with the rise of super-powerful quantum computers, there's a worry that even our best cryptographic methods might not be safe anymore.
To tackle these challenges, we came up with an idea; why not combine ChaCha20 with another key exchange mechanism X25519? X25519 with ChaCha20 is considered robust against quantum attacks due to the security properties of ECC and the symmetric encryption provided by ChaCha20. These algorithms are currently among the recommended choices for post-quantum security in cryptographic protocols, we hope to create a new system that's both quick and super secure.
In this paper, we'll explain how ChaCha20 work separately, and also with X25519 together. We'll also test our new combined system to see if it's faster and safer than just using ChaCha20 on its own. Our goal is to make it easier for people to keep their information safe in the digital world, even as technology keeps advancing. By mixing ChaCha20 with X25519, we're aiming to create a new tool that's both speedy and super secure for protecting our digital lives.

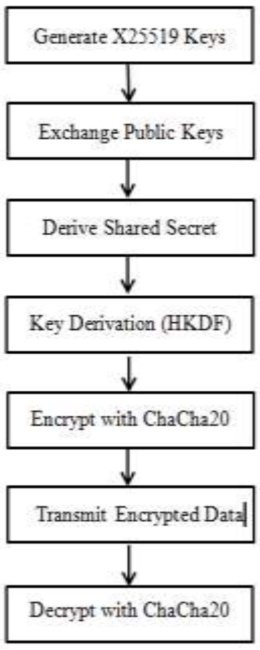
II. Literature Survey

Nasratullah Ghafoori, Atsuko Miyaji "Higher Order Differential-Linear Cryptanalysis of ChaCha20 Stream Cipher" The paper delves into advanced differential-linear cryptanalysis techniques applied to the ChaCha20 stream cipher, revealing vulnerabilities in reduced-round versions. It rigorously examines the cipher's susceptibility to higher-order attacks, offering valuable insights into its security margins.
Limitations: The higher-order differential-linear attacks on ChaCha20 can be computationally intensive, making them impractical for real-world applications due to their high complexity. These analyses may slow down the encryption/decryption process due to the additional overhead required. Thus restricting their general applicability and impact on ChaCha20's overall security.

Nitin Kumar Sharma, Sabyasachi Dey "Analyzing the Probability of Key Recovery in the Differential Attacks against ChaCha20"
This paper focuses on probabilistic analysis to assess the likelihood of key recovery in differential attacks against ChaCha20. While it contributes to understanding ChaCha20's security profile, its direct applicability to real-world cryptographic practice may be constrained by the scope of probabilistic bounds versus practical exploitability.
Limitations: Analysing the probability of key recovery in differential attacks against ChaCha20 can be computationally demanding, reducing the feasibility of these attacks in real-world scenarios. The required computational resources can significantly slow down the encryption and decryption processes. Additionally, the practical success of such attacks relies heavily on specific conditions and assumptions that may not always be met. As a result, the general applicability and impact of these attacks on ChaCha20's overall security remain limited.

III. Proposed Research

For improved speed and security in cryptographic applications, the approach we propose combines the X25519 elliptic curve key exchange protocol with the ChaCha20 encryption algorithm. For symmetric encryption operations, ChaCha20 is used because of its high performance and adaptability to timing attacks, which guarantees effective data processing. Simultaneously, X25519 enables safe key exchange between parties in communication, utilizing its resilience against known classical and possible future quantum cryptography attacks. Our solution enhances computing performance and strengthens data secrecy and integrity across various communication channels and applications by merging these technologies.

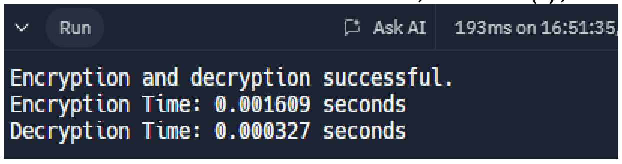


IV. Performance and Analysis

Let's see the performance of ChaCha20 with X25519 respect to its speed and security in cryptographic operations by implementing it on python.

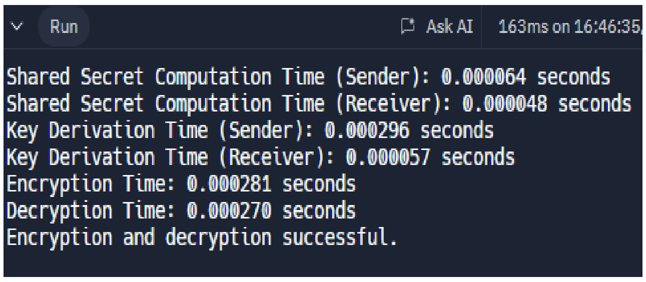
Execution time:

The speed and effectiveness of secure communications are significantly impacted by the execution time of cryptographic algorithms. Reduce latency, boost system performance, and enhance user experience with faster encryption and decoding. Strong security is ensured by evaluating execution time, which allows algorithms like ChaCha20 in addition to X25519 to be used in high-demand situations without sacrificing performance.



```
Encryption and decryption successful.
Encryption Time: 0.001609 seconds
Decryption Time: 0.000327 seconds
```

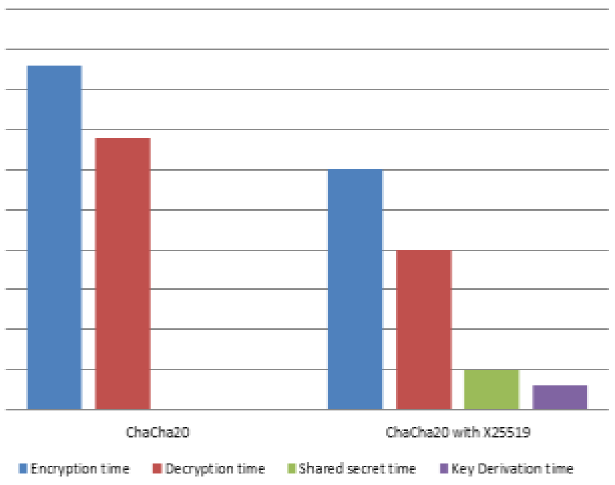
This is the execution time in python using only ChaCha20 algorithm without X25519. When using ChaCha20 algorithm there is only encryption and decryption time.



```
Shared Secret Computation Time (Sender): 0.000064 seconds
Shared Secret Computation Time (Receiver): 0.000048 seconds
Key Derivation Time (Sender): 0.000296 seconds
Key Derivation Time (Receiver): 0.000057 seconds
Encryption Time: 0.000281 seconds
Decryption Time: 0.000270 seconds
Encryption and decryption successful.
```

This is the execution time in python using ChaCha20 algorithm with X25519. When using both ChaCha20 algorithm and X25519 there is Shared secret time, Key derivation time, Encryption and Decryption time.

Comparison:



While comparing both ChaCha20 alone and ChaCha20 with X25519 we can see the time taken for ChaCha20 alone is more when compare to ChaCha20 with X25519.

Security Measures:

Cryptographic algorithms need security features to guarantee the authenticity, integrity, and confidentiality of data. Strong security measures guard against intrusions including tampering, eavesdropping, and illegal access. System security is improved by putting safe key exchange protocols like X25519 and powerful encryption algorithms like ChaCha20 into practice.

Feature	**With X25519**	**Without X25519**
Key Exchange Security	Strong	N/A
Forward Secrecy	Yes	No
Key Distribution	Securely Derived	Pre-shared Key Needed
Resilience to Attacks	High	Moderate
Man-in-the-Middle Protection	Yes	No
Key Management Complexity	Low	High

The above table shows that by offering powerful elliptic curve-based key exchange that ensures forward secrecy and good defence against man-in-the-middle attacks. The system depends on pre-shared keys in the absence of X25519, which can complicate key management and make it less resistant to some assaults. All things considered, adding X25519 greatly improves the system's cryptographic security.

Security with X25519: Forward secrecy and a safe shared key are assured by the X25519 key exchange. Previous conversation is safe even if one key is stolen. This configuration survives standard attacks on key exchange protocols.

Security without X25519: The same degree of protection is not offered when using a pre-shared key without X25519. There is no forward secrecy and the key needs to be transferred safely. Every communication is decryptable in the event that the key is compromised. Attacks are more likely to succeed using this strategy.

V. Conclusion

In this paper, a reliable and effective method for secure communications is offered by the combination of the X25519 key exchange protocol with the ChaCha20 encryption algorithm. High performance and improved security are ensured by combining the robust security features of X25519 with the speed and ease of use of ChaCha20. The implementation shows how current cryptographic approaches can be used to effectively safeguard sensitive data from attackers. In general, this strategy highlights how crucial it is to combine many cryptographic techniques in order to attain the best security and efficiency in modern digital communication systems.

References

[1] “ChaCha, a variant of Salsa20,” in Workshop Record of The State of the Art of Stream Ciphers, ECRYPT Network of Excellence in Cryptology, 2008, https://cr.yp.to/papers.html#chacha.

[2] Y. Nir and A. Langley, “ChaCha20 and Poly1305 for IETF Protocols,” RFC 7539, Internet Engineering Task Force, may 2015. Available: http://www.ietf.org/rfc/rfc7539.txt

[3]Jeeva, A. L, Dr V. Palanisamy, and K. Kanagaram. "Comparative analysis of performance efficiency and security measures of some encryption algorithms." International Journal of Engineering Research and Applications (IJERA) ISSN (2012): 2248-9622.

[4]Elminaam, DiaaSalama Abdul, Hatem Mohamed Abdul Kader, and Mohie Mohamed Hadhoud. "Performance evaluation of symmetric encryption algorithms" IJCSNS International Journal of Computer Science and Network Security 8.12 (2008): 280-286.

[5] Bursztein. E: Speeding up and strengthening https connections for chrome on anddroid . tech. rep (april 2014)

[6] Sadourny, Yulen, and Vania Conan. "A proposal for supporting selective encryption in JPSEC." IEEE Transactions on Consumer Electronics 49.4 (2003): 846 849.

[7] Puech, William, and José M. Rodrigues. "Crypto-compression of medical images by selective encryption of DCT." 2005 13th European signal processing conference. IEEE, 2005.

[8] Massoudi, Ayoub, et al "Secure and low cost selective encryption for JPEG2000." 2008 Tenth IEEE International Symposium on Multimedia. IEEE, 2008.

[9] Z Vahdati, SM Yasin, A Ghasempour, M Salehi, “ Comparison of ECC and RSA algorithms in IoT devices,” Journal of Theoretical and Applied Information Technology,Vol.97, No.16, 2019

[10]Singh, S Preet and Maini, Raman. “Comparison of Data Encryption Algorithms”, International Journal of Computer Science and Communication, vol. 2, No. 1, January-June 2011 pp. 125-127.