eISSN No. : 2976-2278

UTM Computing Proceedings
Innovations in Computing Technology and Applications
Volume: 6 | Year: 2024 | ISBN No. : 2976-2278

Password Manager Application Using XChaCha20 Encryption Algorithm

Danial Syafiq Sukhairul Nizam and Firoz Yusuf Patel Dawoodi
Faculty of Computing,
Universiti Teknologi Malaysia,
81310 Johor Bahru , Johor, Malaysia
danialsyafiq2000@graduate.utm.my, firoz@utm.my

Abstract— **This study details the development of a password manager system that addresses the security and usability concerns associated with traditional password management practices. In the digital age, password protection is more important than ever, with the use of strong and complex passwords becoming increasingly popular. However, remembering multiple passwords for different websites remains a challenge for users. Password managers offer a solution to this problem, but many existing applications use the widely used Advanced Encryption Standard (AES) algorithm, which may become vulnerable to future attacks. In addition, the hardware support required for AES encryption limits its use on certain devices. To address these issues, this system utilizes the XChaCha20 encryption algorithm, which enhances security while also reducing hardware requirements, making the system more widely available. Additionally, the system includes several features that not all password manager applications offer, such as account recovery and password generator. The development process of the system follows the SCRUM methodology to ensure an efficient and effective approach. This thesis contributes to the development of a more secure and user-friendly password management system for users in the digital age.**

Keywords— Mobile Application, Password Manager, XChaCha20

I. INTRODUCTION

With the advent of digital technology, numerous applications have emerged that enable people to connect with one another on various platforms. These applications include social networking sites, online banking platforms, video games, and many more. However, to gain access to these platforms, users are required to provide credentials to verify their identities. Passwords are a primary means of authentication used by these applications. A password is a combination of letters, numbers, and symbols that a user creates to confirm their identity during the authentication process. Usually, a password is paired with a username that the user selects, and together, they grant access to a device, application, or website.

The use of letters, numbers, and special characters in passwords makes them more secure, and their length can be increased to enhance their security further. Passwords are critical in securing personal and confidential information and preventing unauthorized access. Therefore, it is essential to choose strong passwords and keep them secure to protect personal data and avoid security breaches.

II. PROBLEM BACKGROUND

Passwords nowadays are well-known and too easy for attackers to guess in which the complexity of passwords have increased significantly over the years to counter it. Unfortunately, this has also made it more difficult for users to memorize or remember the password, as they now need to remember each password for each website that they registered with in case they have different passwords for each website. Because of these issues, users are more likely to use passwords that are easy to remember and to reuse such passwords across many accounts.

The Advanced Encryption Standard (AES) is the de facto standard for password managers due to its widespread adoption. The existence of more safe and efficient algorithms that can be employed across various platforms may not be as far-fetched as it sounds.

A potential issue is that users must remember one master password to access the app but if they forget that password, they will be unable to access any of their saved passwords from the password manager.

III. Overview of Algorithms

A. AES

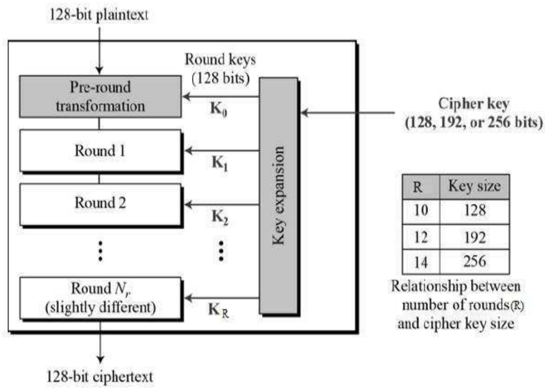


Figure 3.1 Flow of AES

The Advanced Encryption Standard (AES) is a widely used encryption algorithm that has become the industry standard for protecting data. AES was developed by two Belgian cryptographers, Joan Daemen and Vincent Rijmen, and was selected as the successor to the Data Encryption Standard (DES) in 2001 by the National Institute of Standards and Technology (NIST) in the United States.

AES uses a symmetric key encryption algorithm, which means that the same key is used for both encryption and decryption of the data. It operates on a block of data, typically 128 bits in size, and can use different key lengths, including 128, 192, and 256 bits.

B. ChaCha20

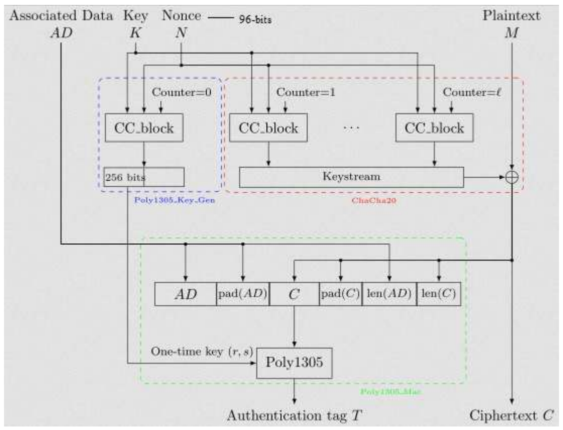


Figure 3.2: Flow of ChaCha20-Poly1305

ChaCha20 is a stream cipher that was originally designed to expands a 256-bits key into 2^64 randomly accessible streams, with each containing 2^64 randomly accessible 64-byte (512 bits) blocks. It is a variant of Salsa20 designed to offer improved data diffusion. Internally, ChaCha20 works like a block cipher used in counter mode, featuring an internal block counter that eliminates the need for incrementing nonce after each block despite being a stream cipher. In Figure 2.5.1, it shows the flow of ChaCha20-Poly1305 which is an authenticated encryption with associated data (AEAD) scheme that combines the ChaCha20 encryption function and Poly1305 message authentication (MAC) function. In this context, ChaCha20 provides the encryption as described above, while Poly1305 provides a way of checking the integrity and authenticity of the data. This means that not only is the data kept secret, but any modifications to the encrypted data can be detected. The difference between the two is that ChaCha20 only provides data encryption, while ChaCha20-Poly1305 provides both data encryption and a way to check that the data has not been tampered with.

C. XchaCha20

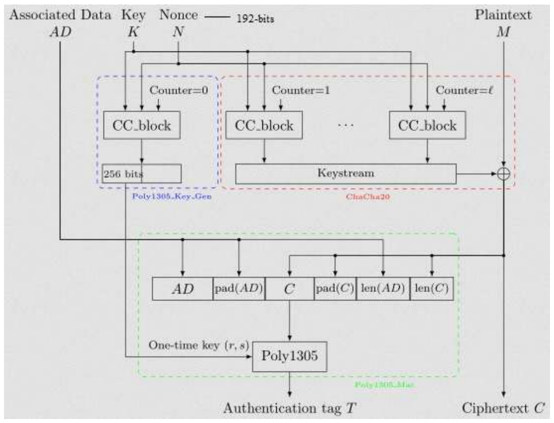


Figure 3.3: Flow of XChaCha20-Poly1305

XChaCha20 is a variant of ChaCha20 with an extended 192-bits nonce, allowing random nonces to be safe. XChaCha20 does not require any lookup tables and avoids the possibility of timing attacks. Internally, XChaCha20 works like a block cipher used in counter mode. It uses the HChaCha20 hash function to derive a subkey and a subnonce from original key and extended nonce and a dedicated 64-bit block counter to avoid incrementing the nonce after each block. XChaCha20 is generally recommended over plain ChaCha20 due to its extended nonce size and its comparable performance. Compared to Advanced Encryption Standard (AES) algorithm, XChaCha20 does not require hardware support. In Figure 2.6.1, it shows the flow of XChaCha20-Poly1305 which is an extension of ChaCha20-Poly1305 that uses the XChaCha20 stream cipher. The difference between XChaCha20 and XChaCha20-Poly1305 is the same as the difference between ChaCha20 and ChaCha20-Poly1305. For example, XChaCha20 provides only data encryption while XChaCha20-Poly1305 provides both data encryption and a way to verify the integrity and authenticity of the data.

IV. Technology Used

A. Android Studio

Android Studio, developed and maintained by Google, is a widely used integrated development environment (IDE) for creating Android applications. It offers a range of Android-specific development tools, including a layout editor, code editor, and debugger. With built-in support for version control systems like Git, developers can easily manage code changes. The IDE also provides debugging tools like memory monitoring and CPU profiling for issue identification and resolution. In terms of security, Android Studio includes a security scanner to detect potential vulnerabilities, performance testing tools, and stability testing capabilities. Overall, Android

Studio's comprehensive features, integration with Google Cloud Platform, and focus on security make it an ideal choice for developing password manager applications.

B. Firebase

Firebase, developed by Google, is a highly regarded platform for mobile and web application development. It offers various services and tools for building, testing, and deploying applications. The platform's real-time database is a standout feature, allowing developers to store and sync data across multiple devices in real-time. This feature is particularly advantageous for developing a 'Password Manager Application Using XChaCha20 Encryption Algorithm' as it enables users to access their passwords from any device with an internet connection through the database stored in Firebase. Firebase also offers analytics and reporting tools for tracking user behavior and identifying potential issues, along with cloud functions for executing server-side code in response to Firebase events. Overall, Firebase's comprehensive range of features, including its real-time database and authentication tools, make it a popular choice for password manager application development.

C. GitHub

GitHub is a web-based version control platform that is commonly used in the development of software systems. In the case of ‘Password Manager Application Using XChaCha20 Encryption Algorithm’, GitHub can be used to store and manage the source code of the system, as well as to track and resolve any issues that arise during development process. Overall, GitHub can be a vital resource in the system development process, helping to ensure that the project is well-organized and efficiently developed.

D. Bouncy Castle

Bouncy Castle is a powerful and easy-to-use software library that provides a wide range of cryptographic functions, including symmetric ciphers, asymmetric ciphers, stream ciphers, message digests, signature algorithm and etc. It is widely used in many different types of applications. In this project, Bouncy Castle is used to provide encryption algorithm which is then used to encrypt and decrypt users’ passwords.

V. METHODOLOGY

SCRUM, commonly employed for collaborative teams, can be adapted for projects with a sole developer. The methodology's key principle of short, iterative development cycles, called "sprints," remains valuable in this context. Breaking down the project into manageable tasks, the developer can focus on one aspect at a time and make consistent progress. By prioritizing essential features, this ensures prompt delivery of a functional product that meets user needs. While working alone, the developer can utilize SCRUM's emphasis on regular communication by keeping organized and updating a supervisor on progress. Daily stand-up meetings can aid in tracking progress and identifying any obstacles, while retrospective meetings allow for review and improvement planning. Implementing SCRUM enables efficient and flexible development while maintaining progress oversight and issue management.

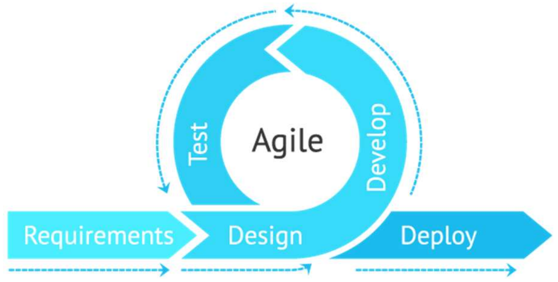


Figure 5.1: Process of SCRUM Methodology

A. Phases of SCRUM Methodology

There are five phases in SCRUM methodology that will be a guideline to the project which are requirements, design, development, testing and deployment.

B. Requirements

This phase in SCRUM methodology involves the process of identifying and prioritising requirements for the project. With the developer having a clear understanding of needs and objectives for the project, they can develop a proper application that fits with the goal. Any potential challenges faced, or resources needed will be analysed in this phase for example by conducting research on current system and technologies used.

C. Design

In this phase, the user interface will be designed as well as overall architecture of the password manager application. This may involve creating prototypes to visualize the application’s features and functionality by using tools such as Figma. Other tools such as diagrams.net will be used to create UML diagram and Sequence diagram.

D. Implementation

In this phase, actual coding and development of the password manager application will be done. This may involve writing and debugging code, integrating third-party libraries and APIs, and implementing the XChaCha20 encryption algorithm to secure user data.

E. Testing

In this phase, the password manager application will be tested to ensure that it meets the requirements and performs as expected. This may involve conducting unit tests, integration tests, and user acceptance tests to identify and fix any issues with the application.

F. *Deployment*

In this phase, the development of password manager application is finished and can be released to users and becomes available for download.

VI. Requirement Analaysis and Design

This chapter will discuss in detail about requirement analysis and project design that also includes database and interface of the proposed system. The project design will include UML diagrams which are use case diagram, activity diagram and sequence diagram of the project to understand the flow of the proposed system better.

A. *Requirement Analaysis*

In this phase, functional and non-functional requirements for the project system will be identified and defined. This is to understand what the system needs to do, what it should be able to accomplish and what constraints it must operate under. By clearly defining these requirements, this can ensure the finished system meets the needs of users. Requirements can be divided into two types which are functional and non-functional requirements. Functional requirements are the specific features and capabilities that the system must have such as the ability to store and retrieve users' saved passwords. Non-functional requirements are the broader characteristics of the system such as the performance of the system.

B. *System Architecture of The System*

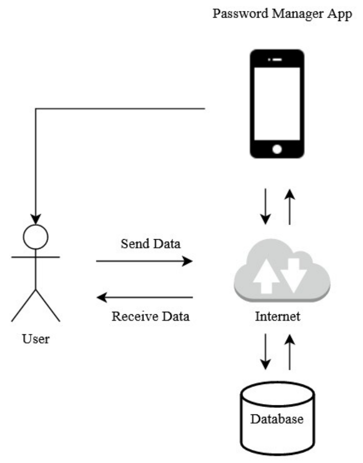


Figure 6.1: System Architecture of Password Manager Application

The password manager application system has a simple system architecture in which there is only one user interacting with the system. The system includes server to store users' information or data. Once the user logged into the system, they can request the server to send the information they need such as the password from the database. User can also add information or data to the system such as their saved username or password or website name, and it will send to the database through an online connection to be stored for future use when requested.

VII. Implementation and Testing

A. *Coding of System Main Functions*

This section focuses on the coding of the system's main functions. Password Manager application has a few functions to achieve the user's requirements such as login, main activity, add info, edit info, delete info and generate passwords.

B. *Login*

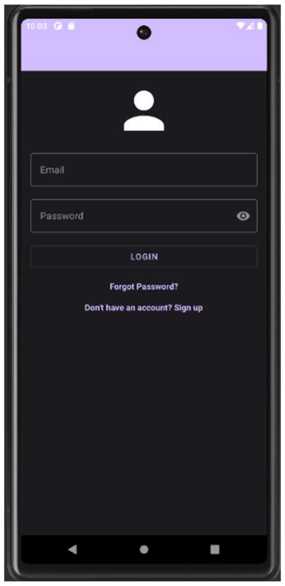
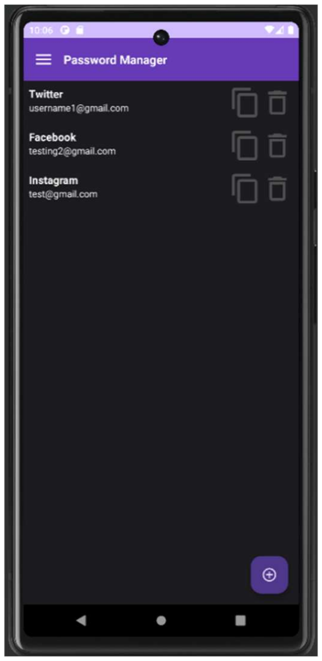


Figure 6.2: Login Interface

Figure 6.2 above shows the login interface which has two text box input for users to insert their email and password. This login procedure is needed every time they logged out to ensure the application is secure from being accessed by other people. There is also a link for registering and resetting password in case user forgot their password.

C. *Main Activity*



Twitter
username1@gmail.com

Facebook
testing2@gmail.com

Instagram
test@gmail.com

Figure 6.3: Main Activity Interface

Figure 6.3 shows the main activity interface once the user logged in which list necessary information that have been saved by the user.

D. *Add Info*

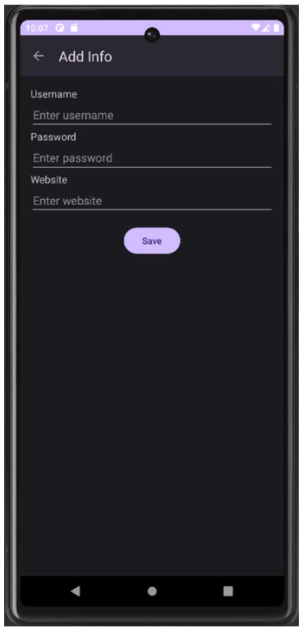
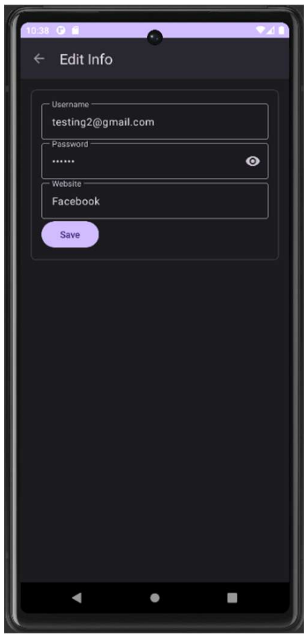


Figure 6.4: Add Info Interface

Figure 6.4 shows the add info interface from the floating action button that user pressed on main activity interface which allow them to add necessary information such as their username, password and website name.

E. *Edit Info*

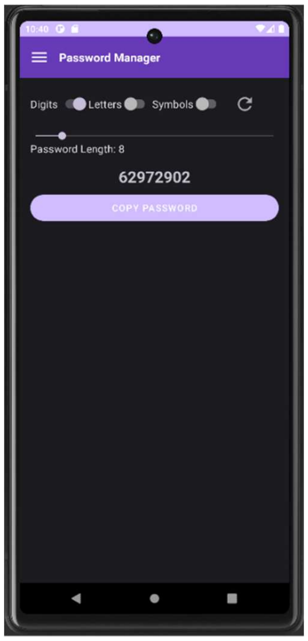


Username: testing2@gmail.com
Password: ••••••
Website: Facebook

Figure 6.5: Edit Info Interface

Figure 6.5 shows an example if the user selected and click one of the saved information that they have added before which will open an edit info page. In the edit info page there are three text box input which they can edit and save.

F. *Password Generator*



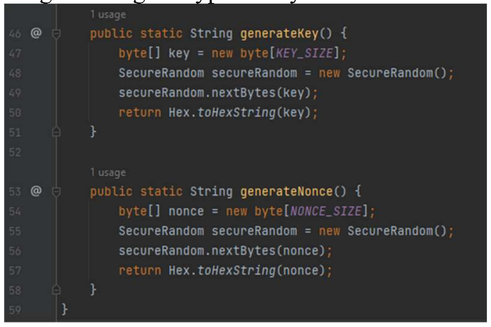
Password Length: 8
62972902

Figure 6.6: Password Generator

Figure 6.6 shows a password generator page for the user to generate their own password if they want to use a randomised password. They can pick to have digit only, letters only, symbols only or have them all toggled as well as copy the generated password into their clipboard.

G. *Encryption*

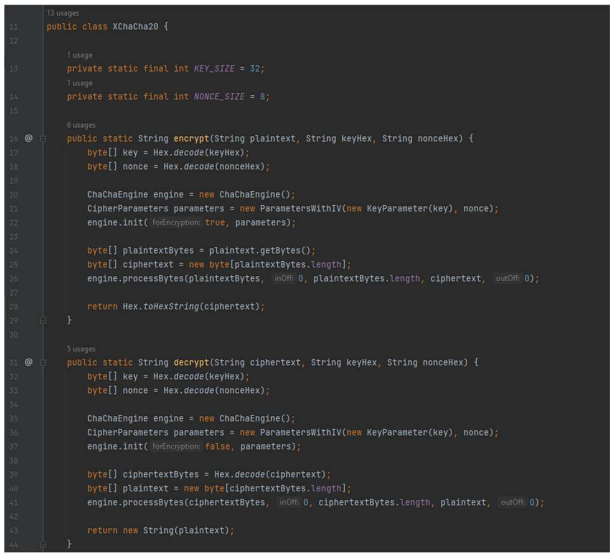
When creating the user's account, it will generate a unique encryption key and nonce for the user to be saved in the database so that it can be used for encryption and decryption of the user's data inside the application. In Figure 6.7 shows the coding for generating encryption key and nonce.



```java
1 usage
public static String generateKey() {
    byte[] key = new byte[KEY_SIZE];
    SecureRandom secureRandom = new SecureRandom();
    secureRandom.nextBytes(key);
    return Hex.toHexString(key);
}

1 usage
public static String generateNonce() {
    byte[] nonce = new byte[NONCE_SIZE];
    SecureRandom secureRandom = new SecureRandom();
    secureRandom.nextBytes(nonce);
    return Hex.toHexString(nonce);
}
}
```

Figure 6.7: Coding for Generating Encryption Key and Nonce



```java
13 usages
public class XChaCha20 {

    1 usage
    private static final int KEY_SIZE = 32;
    1 usage
    private static final int NONCE_SIZE = 8;

    6 usages
    public static String encrypt(String plaintext, String keyHex, String nonceHex) {
        byte[] key = Hex.decode(keyHex);
        byte[] nonce = Hex.decode(nonceHex);

        ChaChaEngine engine = new ChaChaEngine();
        CipherParameters parameters = new ParametersWithIV(new KeyParameter(key), nonce);
        engine.init( forEncryption: true, parameters);

        byte[] plaintextBytes = plaintext.getBytes();
        byte[] ciphertext = new byte[plaintextBytes.length];
        engine.processBytes(plaintextBytes,  inOff: 0, plaintextBytes.length, ciphertext,  outOff: 0);

        return Hex.toHexString(ciphertext);
    }

    5 usages
    public static String decrypt(String ciphertext, String keyHex, String nonceHex) {
        byte[] key = Hex.decode(keyHex);
        byte[] nonce = Hex.decode(nonceHex);

        ChaChaEngine engine = new ChaChaEngine();
        CipherParameters parameters = new ParametersWithIV(new KeyParameter(key), nonce);
        engine.init( forEncryption: false, parameters);

        byte[] ciphertextBytes = Hex.decode(ciphertext);
        byte[] plaintext = new byte[ciphertextBytes.length];
        engine.processBytes(ciphertextBytes,  inOff: 0, ciphertextBytes.length, plaintext,  outOff: 0);

        return new String(plaintext);
    }
```

Figure 6.8: Encryption and Decryption Process

VIII. CONCLUSION

In conclusion, Password Manager Application Using XChaCha20 is an application that can be used by people to store their passwords as well as generating them. With the XChaCha20 encryption algorithm, it can give options for user to pick from the rest of other applications that is similar.

ACKNOWLEDGMENT

We would like to express our sincere gratitude to Faculty of Computing and Universiti Teknologi Malaysia (UTM) for their invaluable support and resources, which played a crucial role in the successful research.

REFERENCES

[1] List of password managers. (2023, January 13). Retrieved January 19, 2023, from https://en.wikipedia.org/wiki/List_of_password_managers

[2] Key, K. (2023, January 26). The best password managers for 2023. Retrieved February 18, 2023, from https://www.pcmag.com/picks/the-best-password-managers?test_uuid=05n7gTzbSo0Sh5pVEDljnCi&test_variant=a

[3] Advanced encryption standard. (n.d.). Retrieved February 21, 2023, from http://www.tutorialspoint.com/cryptography/advanced_encryption_standard.htm

[4] Aumasson, J. (1970, January 01). Too much crypto. Retrieved January 19, 2023, from https://eprint.iacr.org/2019/1492

[5] Arciszewski, S. (n.d.). XCHACHA: Extended-nonce chacha and aead_xchacha20_poly1305. Retrieved January 19, 2023, from https://datatracker.ietf.org/doc/html/draft-irtf-cfrg-xchacha-03#section-2.3

[6] Mujezinovic, D. (2022, October 24). What is the xchacha20 encryption algorithm? Retrieved January 19, 2023, from https://www.makeuseof.com/what-is-xchacha20-encryption/

[7] Javainterviewpoint. (n.d.). Java CHACHA20 encryption and decryption example: Symmetric Encryption. Retrieved January 19, 2023, from https://www.javainterviewpoint.com/chacha20-encryption-and-decryption/

[8] Chacha20. (n.d.). Retrieved January 19, 2023, from https://libsodium.gitbook.io/doc/advanced/stream_ciphers/chacha20

[9] XCHACHA20. (n.d.). Retrieved January 19, 2023, from https://libsodium.gitbook.io/doc/advanced/stream_ciphers/xchacha20

[10] Salsa20. (2022, December 28). Retrieved January 19, 2023, from https://en.wikipedia.org/wiki/Salsa20#XChaCha

[11] Keeper password manager review (2023): How secure is keeper's app. (n.d.). Retrieved January 19, 2023, from https://proprivacy.com/password-manager/review/keeper

[12] Chacha20-Poly1305. (2022, October 13). Retrieved January 19, 2023, from https://en.wikipedia.org/wiki/ChaCha20-Poly1305

[13] The legion of the bouncy castle. (n.d.). Retrieved from https://www.bouncycastle.org/specifications.html