DEUTERIUM: A SECURE PROTOCOL FOR GROUP MESSAGING WITH ROTATING KEYS AND IDENTITY VERIFICATION

Xiyou Jin[1] and Jonathan Sahagun[2]

[1]Northwood High School, 4515 Portola Pkwy, Irvine, CA
[2]Computer Science Department, California State Polytechnic University, Pomona, CA

ABSTRACT

Deuterium is a protocol for instant messaging that allows users to join a channel, securely exchange messages, and rotate the group key for security purposes. When a user wants to join a channel, they must first send their public key, wallet address, and a digital signature to verify their identity. If the user's identity is successfully verified, the channel's creator will perform an elliptic Elliptic Curve Diffie-Hellman key exchange with the user using curve25519,generating a group key for encrypting messages in the channel. The group key is periodically rotated for security purposes. Users can send messages to the channel by encrypting them with the X25519-XSalsa-Poly1305 algorithm, including a Galois Message Authentication Code instead of an index after keys are exchanged, and attaching a digital signature to verify the authenticity of the message. The protocol also includes a "Termination event" for handling errors or exceptions that may occur during key exchange or message exchange.

KEYWORDS

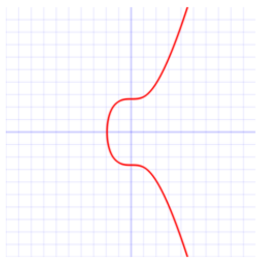
MetaMask, Ethereum, PubSub, Encryption, Protocol

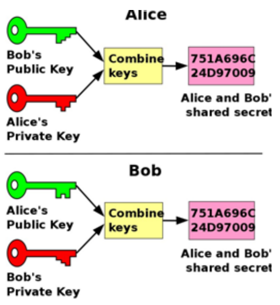
1. INTRODUCTION

In today's digital age, where much of our personal and professional lives are conducted online, it is increasingly important to consider the security and privacy of our communication and to take steps to protect ourselves and our sensitive information. This includes choosing secure communication channels and following best practices for protecting our personal information and devices. It is difficult to quantify the amount of attention that has been raised about the importance of security and privacy for people, however, it is clear that these issues have become increasingly important in recent years due to the pandemic, the amount of personal and sensitive information that is shared online has increased. There are many organizations and individuals who are working to raise awareness about the importance of security and privacy and to educate people about how to protect themselves and their sensitive information online. These efforts may include public campaigns, educational resources, and advocacy for stronger privacy laws and regulations. There are a number of laws and technologies that have been developed to protect people's online privacy and security, from acts and laws that were enached, like the California Consumer Privacy Act (CCPA), and the Health Insurance Portability and Accountability Act (HIPAA), to technologies that we have developed, such as the Transport Layer Security (TLS), and Virtual Private Networks (VPN) [3][1][6]. These resources regulate the collection, use, and storage of personal data, protect the confidentiality of electronic protected health information, and secure communication over the internet, among other things. Even though many privacy laws and security methods were introduced, many online communication platforms still collect users' data, which can be not ideal for some users. Therefore, many have sought alternatives that are transparent and privacy-respecting. One of these examples is matrix, which according to their website, is "an open network for secure, decentralized communication". It is a good alternative, but it relies on hosted servers. The deuterium protocol allows a connection to be established over any communication protocol, such as PubSub service on the InterPlanetary File System(IPFS), websockets, https, etc.

2. CHALLENGES

One of the most important considerations when creating a protocol is ensuring the security of the system. This involves designing and implementing secure mechanisms for authentication, encryption, and integrity protection, as well as protecting against various types of attacks and vulnerabilities. Authenticating the identity of users and devices is important for preventing unauthorized access and protecting against impersonation attacks. This may involve using mechanisms such as passwords, digital certificates, or biometric authentication. Encryption is equally important. Encrypting messages and data helps to protect against eavesdropping and other types of attacks on the communication channel. This may involve using symmetric or asymmetric encryption algorithms, as well as key management protocols. In the case of ethereum, it uses a combination of symmetric and asymmetric encryption techniques to secure data and communication. One of the main encryption techniques used by Ethereum is the Secure Hash Algorithm (SHA) family of hash functions [7]. These hash functions are used to generate a fixed-size hash value (also known as a digest) from a message or data. The hash value is unique to the message or data, and can be used to verify the integrity of the data. Ethereum uses SHA-256 and Keccak-256 (also known as SHA-3) as its primary hash functions [4]. In addition to hash functions, Ethereum also uses public-key cryptography for secure communication and authentication. Public-key cryptography uses a pair of keys - a public key and a private key - to encrypt and decrypt messages. The public key is used to encrypt the message, and the private key is used to decrypt it. Ethereum uses the Elliptic Curve Digital Signature Algorithm (ECDSA) and the Secure Remote Password (SRP) protocol to generate and manage these keys. In simpler words, the Elliptic Curve Digital Signature Algorithm (ECDSA) is a cryptographic algorithm that is used to generate and verify digital signatures[10]. In ECDSA, a pair of keys - a public key and a private key - are generated from a randomly chosen point on an elliptic curve[10]. The private key is kept secret and is used to sign messages, while the public key is made publicly available and is used to verify the signature. The Ethereumblockchain uses the secp256k1 curve for its Elliptic Curve Digital Signature Algorithm (ECDSA). The secp256k1 curve is a standardized elliptic curve that is widely used in cryptography, and is the curve that is used for Bitcoin as well[5]. It is defined as y2=x3+7. In ECDSA, the secp256k1 curve is used to generate a pair of keys - a public key and a private key - from a randomly chosen point on the curve[5]. The private key is kept secret and is used to sign messages, while the public key is made publicly available and is used to verify the signature. The use of the secp256k1 curve in Ethereum helps to ensure the security and efficiency of the blockchain's cryptographic operations. It also enables Ethereum to be compatible with other systems and protocols that use the same curve. After the key pairs are generated, when parties need to communicate securely and want to establish a shared secret key without exchanging the key directly, a key exchange is required. Ethereum uses the Elliptic Curve Diffie-Hellman (ECDH) key exchange algorithm to establish secure communication between parties [9]. ECDH is a variant of the Diffie-Hellman (DH) key exchange algorithm, which allows two parties to establish a shared secret key over an insecure communication channel without exchanging the key directly [9]. In ECDH, the parties use their own private keys and each other's public keys to calculate a shared secret key. The shared secret key can then be used to encrypt and decrypt messages sent between the parties. To perform ECDH, both parties need to have a public-private key pair, which can be generated using the Elliptic Curve Digital Signature Algorithm (ECDSA). The private key is kept secret and is used to sign messages, while the public key is made publicly available and is used to verify the signature.





(a) Graph of secp256k1's elliptic curve (b) Simplified process of the Elliptic Curve y2 = x3 + 7 Diffie-Hellman (ECDH) key exchange

Figure 1: Visualization of the encryption methods used in Deuterium

When designing a protocol, it is important to strike the right balance between complexity and simplicity. A protocol that is too complex may be difficult to implement and use, while a protocol that is too simple may not provide the necessary functionality and security. Keeping a simple protol design also leaves less space for user error, and gives uers more area to make secure choices.[2] Some key considerations in finding this balance include ensuring that the protocol provides the necessary functionality, is easy to use, and is easy to implement. For example, a protocol that is too complex may require extensive documentation or have a steep learning curve, which can make it difficult for non-technical users to understand and use. On the other hand, a protocol that is too simple may not be able to support advanced features or capabilities that are necessary for the system. By carefully considering these factors and finding the right balance, it is possible to create a protocol that is both effective and easy to use. Interoperability refers to the ability of different systems or devices to work together and exchange information or data. In the context of protocols, interoperability means that different implementations of the protocol can communicate and exchange data with each other. This is important because it allows different systems or devices to work together and share data or resources, regardless of the specific technology or platform they are using. In today's world, there are many different devices, with different requirements and capabilities. That is why the interoperability of a protocol is also important during its development. There are a number of factors that can impact the interoperability of a protocol. First, the compability. Protocols need to be compatible with the systems or devices that will be using them. This means that the protocol should use technologies or algorithms that are supported by the systems or devices. For example, a protocol that uses a specific encryption algorithm may not be interoperable with systems or devices that do not support that algorithm. Second, standardization: Protocols that are standardized are more likely to be interoperble because they are widely accepted and supported by different systems or devices. For example, the HTTP protocol, which is used for web communication, is standardized and is widely supported by web browsers, servers, and other web-based systems [8]. Third, Flexibility: Protocols that are flexible and can be customized or configured to meet the needs of different systems or devices are more likely to be interoperable. For example, a protocol that supports different encryption algorithms or authentication mechanisms may be more interoperable because it can be adapted to work with a wide range of systems or devices. And finally, the documentation and support are equally important. Protocols that have comprehensive documentation and support are more likely to be interoperable because they are easier for developers to understand and implement. Protocols with good documentation and support are also more likely to be adopted and used by a wider range of systems or devices.

3. Solution

This protocol provides a method for secure communication in a group setting, possibly over a network. It involves the use of encryption and digital signatures to verify the identity and authenticity of messages. There are several types of events that can occur within this protocol:

Join Event When a user wants to join a channel, they must send their public key, wallet address, and a digital signature to prove their identity. If the identity cannot be proven, a termination event is sent.

Accepted Event When a user is accepted into the channel, a key exchange is performed and a group key is generated and sent back to the user, encrypted using
their public key. A digital signature is also required to verify the identity of the sender.

Termination Event If a user is denied entry, fails to prove their identity, has a key exchange failure, is kicked by the channel creator, or fails to send a beat event for longer than 5 seconds, a termination event is sent, along with a message indicating the reason for the termination.

Request for Rotation Event This event is sent by the channel creator to all users in the room, and prompts them to send a join event. A new group key is then generated.

Beat Event This event is sent periodically by the user to the channel creator every second.

Message Event When a user sends a message, it is signed, then encrypted using the group key, and sent as a message event. A digital signature and message authentication code (MAC) are also required, as well as a timestamp.

Overall, this protocol is designed to provide secure communication in a group setting, with a focus on encryption and identity verification, as well as being flexible, as it can be used on any network. In the sign in process described in this protocol, a user must provide their public key, wallet address, and a digital signature to prove their identity in order to join a channel. The digital signature is used to verify that the message (in this case, the request to join the channel) was actually sent by the owner of the private key associated with the public key provided. This helps to prevent unauthorized access and protect the security of the channel. To create a digital signature, the user creates a hash of the message using a secure hashing algorithm, and then uses their private key to encrypt the hash. The resulting encrypted hash (the digital signature) can then be sent along with the message, and anyone with the corresponding public key can use it to verify the authenticity of the message by decrypting the signature and comparing it to a hash of the message that they themselves have calculated. If the hashes match, the message is verified as authentic. In this protocol, the digital signature is included in the MessagePack blob sent as part of the join event. It is used by the channel creator to verify the identity of the user requesting to join the channel. If the signature cannot be verified, a termination event is sent. The most interesting part in this protocol might be the accepted event. When a user is accepted into the channel, the channel's creator will perform an ECDH key exchange, using the curve25519, with the user that wants to join the channel, as well as updating the group key that was generated for the rest of the group members if it is present. Curve25519 was chosen due to its small key size (256 bits) and resistance to various types of attacks. It is also relatively efficient to compute, making it well-suited for use in systems where computational resources may be limited. After performing the key exchange, the server will send the generated group key back to the user, encrypted using the public key of the user. A digital signature is also required to verify the identity of the sender. The GMAC of the group key is also calculated to protect the originality of the group key. After providing the group key, the clients can encrypt and decrypt messages that were sent using the group key. Using a group key in a cryptographic protocol can provide improved security for communication within a group, as well as being more convenient and efficient than establishing a separate key for each pair of users. A group key can be used to encrypt and decrypt messages sent between all members of the group, protecting against eavesdropping and other types of attacks. It is also easier to manage and requires less computation than establishing multiple individual keys. More information about the protocol can be found here: https: //github.com/isotope-app/deuterium, and the reference implementation written in Typescript can be found here: https://github.com/isotope-app/hydrogen.

4. CONCLUSIONS

Overall, this protocol described is a communication protocol designed for use in a group messaging application. It includes a number of different events, such as join, accepted, termination, request for rotation, beat, and message, each of which serves a specific purpose in the communication process. The protocol uses a combination of symmetric and asymmetric encryption techniques, including the Secure Hash Algorithm (SHA) family of hash functions and the Elliptic Curve Digital Signature Algorithm (ECDSA), to ensure the security of communication and data [10]. It also uses the Elliptic Curve Diffie-Hellman (ECDH) key exchange algorithm to establish a shared secret key between parties, which is used to encrypt and decrypt messages[9]. While the protocol has a number of strengths, including its use of strong encryption techniques and its support for secure communication and identity verification, it also has some limitations, such as the potential for performance and scalability issues, which may need to be addressed in the future to improve the overall effectiveness of the protocol. The protocol may have performance limitations that could impact the speed or efficiency of the messaging system. For example, the use of encryption and key exchange protocols may add overhead and latency to the communication process, which could slow down the delivery of messages. Another might be the compatibility: The protocol may not be compatible with all systems or devices that may be used for instant messaging. This could limit the ability of the messaging system to work with a wide range of clients or devices, or could require additional work to support different platforms or technologies. The most important might be the scalability: The protocol may not be designed to handle a large volume of messages or a large number of users, which could impact the scalability and reliability of the messaging system. This could be an issue if the messaging system is intended to be used by a large number of users or if it is expected to handle a high volume of messages. Due to the nature of the encryption in the protocol, there are several aspects that may affect the scalability of the protocol. First, the key size: The size of the group key could affect the efficiency and performance of the system. As the number of users in the system increases, the number of keys that need to be exchanged and used in the ECDH key exchange process will also increase. This could potentially impact the performance of the system, depending on how efficiently the key exchange process is implemented and how well the system is optimized to handle the additional keys. However, it is important to note that the performance impact of adding more keys to the ECDH key exchange process may not be linear. This means that the performance of the system may not necessarily degrade at a constant rate as the number of keys increases. For example, the system may be able to handle a larger number of keys more efficiently if the keys are smaller in size, or if the cryptographic algorithms being used are more efficient. Second, key management: Managing the group key and distributing it to all members of the group could also be a challenge, particularly if the group is large or dynamic. This could involve additional overhead and complexity, which could impact the scalability and reliability of the system. Third, key updates: If the group key needs to be updated or rotated on a regular basis, this could also add overhead and complexity to the system. This could be particularly challenging if the group is large or if there are many messages being exchanged. There are a number of ways that the protocol could potentially be addressed in the future to improve its performance and scalability. One approach is to optimize the key exchange process by using more efficient cryptographic algorithms or implementing optimization techniques. Improving the key management process could also help, by designing more efficient mechanisms for distributing and updating the group key, or using key management protocols that are better suited to the system's needs. Another option is to redesign the protocol itself, revising it to make it more efficient, scalable, or flexible, or adding new features and capabilities to meet the system's needs. Ultimately, the most effective approach will depend on the specific needs and requirements of the system, and by carefully considering the limitations of the current protocol and working to address them, it may be possible to improve its performance, scalability, and reliability.

REFERENCES

[1] Health insurance portability and accountability act of 1996, 1996.
[2] Protocol design principles - ncsc, Dec 2020.
[3] California consumer privacy act (ccpa), Feb 2023.
[4] Guido Bertoni, Giles Van Assche, Micha¨elPeeters, and Joan Daemen. The k sha-3 submission - keccak team, Jan 2011.
[5] Daniel R. L. Brown. Sec 2: Recommended elliptic curve domain parameters, Jan 2010.
[6] T. Dierks and E. Rescorla. The transport layer security (tls) protocol version 1.2, Aug 1970.
[7] Morris Dworkin. Sha-3 standard: Permutation-based hash and extendable-output functions, 2015-08-04 2015.
[8] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee. Hypertext transfer protocol – http/1.1, Jun 1999.
[9] RakelHaakegaard and Joanna Lang. The elliptic curve diffie-hellman (ecdh), Dec 2015.
[10] Don Johnson, Alfred Menezes, and Scott Vanstone. The elliptic curve digital signature algorithm (ecdsa) - international journal of information security, Jan 2014.