Bitcoin Like Messaging System

Supervisor: Wang Cong
Student: Zhang Yan
email: yzhang775-c@my.cityu.edu.hk

July 19, 2016

Abstract

Motivated by recent revelations of widespread state surveillance of personal communication, especially the Snowdon Event. Users' privacy should be seriously concerned, but the traditional messaging system cannot provide efficient solution to solve the problems. Moreover, some service providers collude with the goverment and reveal users' privacy. That's why we want to propose a reliable decentralized messaging system to address the problem.

We propose a decentralized messaging system which implements the blockchain protocol to provide the peer-to-peer network to prevent the surveillance of the server, also implements the Signal protocol to provide the Forward Secrecy and Deniabliity to achieve the desirable demands of the messaging systems and ensure the security. And we also propose another scheme to combine the decentralized messaging system with the searchable encryption and the scheme provides efficient updates and security insurance.

1 INTRODUCTION

Since the break of the Snowdon Event, there has been a growing concern about the violation of privacy in the digital world, especially the ubiquitous mobile messaging conversations, which has been dominating the online communication since the inception of mobile phones. Although people know that their privacy is quite import, the available solutions to achieve the privacy protection are still lacking. Some popular mobile messaging systems even don't provide the end-to-end encryption, user's privacy leaks without informed and controlled. Furthermore, some App producers claim that their end-to-end encryption scheme will ensure no one else can read the messages except the participants of the conversation, but with the change of encryption demands, the performances of some Apps are still not satisfied peoples needs. Another problem of existing messaging system, the server is hidden trouble in the communication. If the server becomes malicious and leaves trapdoor of the encryption, user's conversation will be also collected transparently. According to the PRISM Plan, the huge company provides the users' information to the NSA (National Security Agency), which means every users' online move monitored by the NSA. Also, assuming the encryption scheme is reliable, user's metadata can be collected by the server at least. A phone number or an E-mail address will leak a lot of information about user's identity. So, the decentralized messaging system with reliable encryption is very suitable for current needs of users.
Basing on the background of the existing messaging systems and the problems they have, we want to introduce a decentralized messaging system with reliable encryption scheme. Most of the P2P protocols just provide service for the online user, which

means user cannot receive the messages when they are offline in the messaging system. The Bitcoin protocol gives us a practical example of the anonymous P2P network without server, which can also hide user's identity and provide an asynchronous network. In the part of encryption, we focus on two properties, Perfect Forward Secrecy and Deniability, which are desirable in the messaging systems. And we also propose a scheme which combines the P2P messaging system with searchable encryption to provide better service for end users. Our main contribution is that we propose a fully decentralized system which provide Forward Secrecy that the existing decentralized messaging systems don't achieve.

2 RELATED WORK

2.1 SIGNAL

Signal is developed by the Open Whisper Systems which got a partnership with the WhatsApp to improve WhatsApp's security, WhatsApp is the most successful messaging system in the world which owns one billion users. The Axolotl protocol also implemented by some other instant messaging systems like Wire. It's a strong evidence that Axolotl protocol is secure which is an upgrade version of the Off-the-Record protocol(OTR), it means to provide the main features which mentioned by the Sok: Secure Messaging [Unger et al.'15]: Perfect Forward Secrecy, Deniability and Asynchronicity.
In this system, all the keys generated over the Curve25519, keys are separated into one identity key and several Prekeys. It uses 3-DH key exchange protocol to generate the shared secret for the KDF (Key Derivation Function), the shared secret will go through the Axolotl Ratchet protocol (the KDF's name) to produce the encryption keys and HMAC keys. The messages will be encrypted by the symmetric encryption AES in CTR model without padding to generate the cipher. The cipher, freshness and one of the Prekey (which will be used in the next session) will be used to obtain the MAC. Final message contains freshness, Prekeys, identity key, MAC and other identifier. Final message will be also encrypted by the key shared with the server.
The encryption scheme will go like this:

1. $(\bar{x}_{a,0}, g^{\bar{x}_{a,0}}) \in_R \mathcal{Z}_p \times$ Curve25519
2. secret $= (g^{\bar{x}_{b,z} \cdot a}, g^{b \cdot \bar{x}_{a,0}}, g^{x_{b,z} \cdot \bar{x}_{a,0}})$
3. $(k_{BA,r}, k_{BA,c}) = f(\text{secret}, const_0, const_R)$
4. $(\bar{x}_{a,1}, g^{\bar{x}_{a,1}}) \in_R \mathcal{Z}_p \times$ Curve25519
5. $(\bar{x}_{a,2}, g^{\bar{x}_{a,2}}) \in_R \mathcal{Z}_p \times$ Curve25519
6. $k_{shared} = g^{x_{b,z} \cdot \bar{x}_{a,2}}$
7. $(k_{AB,r}, k_{AB,c}) = f(k_{shared}, k_{BA,r}, const_R)$
8. $(k_{Enc}, k_{MAC}) = f(MAC_{k_{AB,c}}(const_1), const_0, const_K)$
9. $k_{AB,c} = MAC_{k_{AB,c}}(const_2)$

10. m $\in M$

11. c = $ENC_{k_{Enc}}(m)$

12. $ctr_a = 0$

13. $pctr_a = 0$

14. $\chi = (v,\ g^{\bar{x}_{a,2}},\ ctr_a,\ pctr_a,$ c)

15. tag = $MAC_{k_{MAC}}(\chi)^*$

Final message includes: χ, tag, z, $g^{\bar{x}_{a,0}}$, g^a.
($const_0 = 0x00^{32}, const_1 = 0x01, const_2 = 0x02, const_R = "WhisperRatchet", const_K = "WhisperMessageKeys", v = 2$)

Table 1: The key derivation fuction

Algorithm 1 f(input, key, string)
$k_{pr} \longleftarrow MAC_{key}(input)$
$k_0 \longleftarrow MAC_{k_{pr}}(string, 0x00)$
$k_1 \longleftarrow MAC_{k_{pr}}(k_0, string, 0x01)$
return (k_0, k_1)

There are three core parts of this system to provide the properties that we want: Prekeys, 3DH, Axolotl protocol.

2.2 THREEMA

Threema is another popular centralized system, but there is a small difference between the traditional messaging system that Threema uses three central servers to provide service for the users. The three servers are chat server, directory server, media server. Directory server provides service for the trust establishment which indicates user's public key will be stored at this server after the registration. The chat server will store and forward the messages between the participants, also the media server will deliver the media messages in the middle.
The message encryption scheme can only satisfy the Deniability but no Forward Secrecy, Threema provides the Forward Secrecy in the transport layer. After registration, the participants will generate asymmetric key pairs. When sender wants to communicate with the others, sender will connect to the directory server and ask the public key of the receiver. Then sender will use his private key and receiver's public key go through the ECDH key exchange protocol to generate the shared secret, this shared secret will be hashed with HSalsa20 to derive a symmetric key that both of the sender and receiver can generate it. Sender will use this symmetric key to encrypt the plaintext and nonce and get the ciphertext. Then sender uses the Ploy1305 to generate the MAC code and prepends it to the ciphertext. The figure (figure 1) illustrates the encryption scheme of the Threema.
The encryption scheme will go like this:

1. Sender's key pair: $(x, g^x) \in_R \mathcal{Z}_p \times$ Curve25519

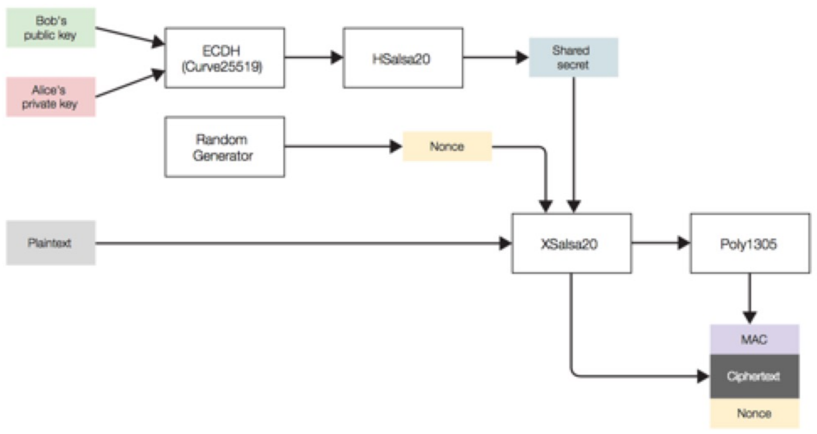


Figure 1: Threema encryption scheme

2. Get receiver's public key: g^y
3. Shared secret = $(g^{y \cdot x})$
4. k_{shared} = Hash$(g^{y \cdot x})$
5. Ciphertext = $Enc_{k_{shared}}(nonce + plaintext)$
6. MAC = $MAC_{Ploy1305}(ciphertext)$

Final message will contains: ciphertext, MAC, nonce.

2.3 BLEEP

BLEEP is developed by the BitTorrent Inc. which gives up the traditional Client-Server model and changes to the P2P network. BLEEP uses its own DHT (Distributed Hash Table), which makes the network lookup only happened when the user first accesses the network, the following conversation will use the DHT to find the receiver. This DHT maintains a list of online users in the format of public key and IP, it can also reduce the network burden. There are not sufficient academic paper about the BLEEP, so we don't know the encryption scheme, just know it uses the ratchet to provide the Perfect Forward Secrecy.
All the messages in this system will be delivered by the peers, there is no server participants the messages transportation. The puzzle of the P2P network is how to solve the Asynchronicity, because there is no central server provide the service for the offline users. BLEEP provides the asynchronicity in this way, if the receiver is offline, the messages will be set 3 days TTL (time to live), just keep delivering in the DHT until the receiver is online or the messages are dead. During my test, there is another problem of this system that before two participants establish the trust (the receiver is not on the contract of the sender), the message sent from the sender will be never received by the receiver. I think this may cause by the sender doesn't have the key materials of the receiver which means the participants exchange their key materials during the trust establishment.

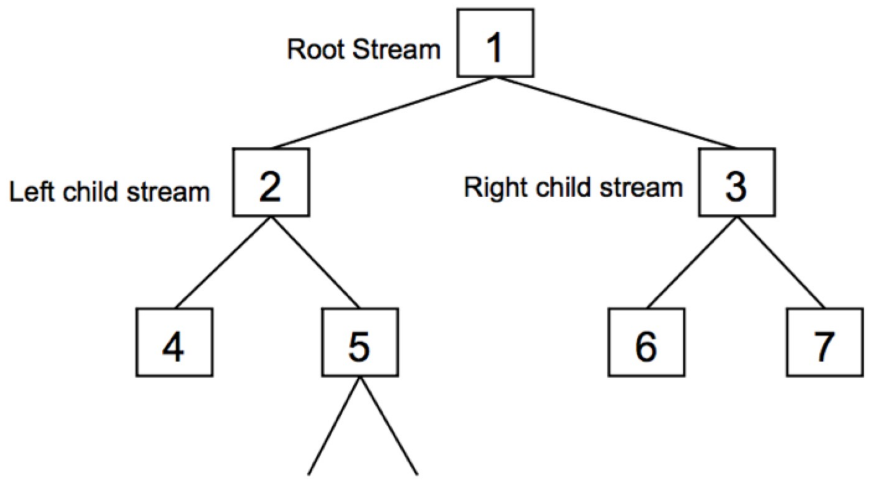
But BLEEP still inspire the messaging systems companies a lot, because the P2P model will prevent the attack from the server which is the biggest treat of the messaging systems.

2.4 BITCOIN

Bitcoin is the prototype of our system, it's the most successful digital cash all over the world. Bitcoin is a fully decentralized electronic cash system, the money represented by the transactions, and each transaction will be signed by the sender's private key that only the owner of the key pair has, to ensure the money is sent by a valid user. Every peer will verify this signature and deliver the transaction to the next hop. So there is no central server participants in the transactions' transfer to make this server is not controlled by any specific node.
The biggest problem of the peer-to-peer system is that the asynchronous network are not easy to ensure, so the Bitcoin proposed the blockchain scheme. In this system, every transaction will be delivered by the peers and the peers will also store the transactions in the unconfirmed block at the local storage if the transaction is valid. Normally, peers will verify two things: whether the signature is valid and whether the amount of the money is valid. Meanwhile, every full node has the mining function which indicates to find the proof-of-work. The proof-of-work is finding a difficult hash collision. When one miner finds this hash collision this miner has the right to share his unconfirmed block to other peers include the transactions stored in this block. Peers will verify the hash of this block, if it's valid for this block will be added to the existing blockchain and the other peers will stop their current computing and begin to find the next block's hash. The miner who mined this block will win a reward that's also how the money of this system create. We should be awareness of two things, the miner stores the transactions in a special data structure Merkle hash tree, which is fast for the peers check whether the sender has enough money to send, the other thing is that every transaction can be traced back to the create money transaction. So this is trustful for the user to guarantee the fairness of the trade.
But the system is not perfect still has some problem. The first problem is the double-spend, which means sender use the money twice or more. According to Satoshi's white paper, he suggests user commits the business deal after six blocks are mined to avoid the money lose. Another problem is that if the two miners mines the same height block at the same time, it may lead the fork of the main blockchain. This problem we will illustrate in the section 4. The last problem of the Bitcoin is the 51% attack, which means if the attackers have over 50% computing power of the whole network, the attackers will have the ability to modify the main blockchain, this kind of attack will be illustrate in the section 4. In conclusion, Bitcoin is the best example to build a system with a central server and still guarantee the reliable communication between the participants.

2.5 BITMESSAGE

BitMessage is a fork of the Bitcoin, the message transfer mechanism just similar to the Bitcoin but need another proof-of-work for each messages to avoid the malicious user to rebroadcast his message in a short time interval, the time of the proof-of-work can



$$Parent\ of\ n = \begin{cases} \left\lfloor \frac{n}{2} \right\rfloor & if\ n > 1 \\ null & if\ n \leq 1 \end{cases}$$

$$Children\ of\ n = [n \cdot 2, (n \cdot 2) + 1]$$

Figure 2: The look-up scheme of the BitMessage

be adjusted. BitMessage also provides a network routing mechanism to reduce the burden of the network. As Bitcoin does, the public key will be used as the receiver's address, when the network reaches a certain threshold, users will be self-segregate into large cluster called stream. The stream number is hard coded into users' public key address. Before When the conversation begins, sender will run a look-up to check whether the receiver is online, if the receiver is online, the message will be sent to the target stream with high priority to reduce the network delay, otherwise, the message will be broadcasted to all the peers and stored into the blockchain. The figure (figure 2) shows the look-up scheme of the Bitcoin. Also the message has a TTL in two days, so it will be deleted from the blockchain after two days. If two days later the sender still does not receive the ACK from the participants, the sender will send this message again to ensure the receiver can get this message.
BitMessage's encryption scheme can provide Deniability but no Perfect Forward Secrecy. BitMessage uses ECIES (Elliptic Curve Integrated Encryption Scheme) to generate the asymmetric key pair. It uses the ECDH (Elliptic Curve Diffie - Hellman) to generate the shared secret, the part of the shared secret will be hashed through SHA512, and the first 32 bytes of the hash result will be used as the encryption key and the last 32 bytes of the hash result will be used as the HMAC key. The plaintext will be encrypted by the encryption key in AES-CBC model with PKCS7 and IV will be used. The HMAC key will be used as the slat and the cipher, IV and sender's public key as the data to calculate the MAC by the HMACSHA256.

The encryption scheme will go like this:

1. It uses the hash of public key as the address, the key pairs generated over ECIES.
2. The key exchanged by user include version number, Stream number, and checksum.
3. The destination public key is called K.
4. Generate 16 random bytes using a secure random number generator. Call them IV.
5. Generate a new random EC key pair with private key called r and public key called R.
6. Do an EC point multiply with public key K and private key r. This gives you public key P.
7. Use the X component of public key P and calculate the SHA512 hash H.
8. The first 32 bytes of H are called k_e and the last 32 bytes are called k_m.
9. Pad the input text to a multiple of 16 bytes, in accordance to PKCS7.
10. Encrypt the data with AES-256-CBC, using IV as initialization vector, k_e as encryption key and the padded input text as payload. Call the output cipher text.
11. Calculate a 32 byte MAC with HMACSHA256, using k_m as salt and IV + R + cipher text as data. Call the output MAC.

This scheme can provide Deniability because all the encryption key can be generated both by the sender and receiver and it does not implement the signature to provide the authentication, but instead the MAC code. But the ephemeral key does not use in this case, the encryption key is static. Once the encryption key is compromised, the conversation between the two participants will be not secure anymore.

2.6 SHADOWCHAT

ShadowChat is the extension application of the ShadowCash which is a kind of altcoin of the Bitcoin. It provides a high level privacy and anonymity network and also preserving the core part of the decentralized system. The transactions will be also store in blockchain at each node's local storage to provide the fully peer-to-peer system. Comparing to the Bitcoin, the main difference is the signature scheme, Bitcoin uses the EdDSA digital signature but the ShadowCash uses the ring signature scheme which has better performance for prevent the double-spend and anonymity preserving.
ShadowChat uses the ECDH for the key exchange, and the public key will be used as the contact address which is similar with the Bitcoin and BitMessage. Sender can get the receiver's public key from the blockchain if the receiver is a ShadowCash user or get the public key via out of band channel and type it manually, because the blockchain had already downloaded at the local storage, both of the two ways can prevent the Man-in-the-Middle attack (MitM), this is the hassle of the trust

establishment of the messaging systems. After that, sender will use his private key and this public key go through the ECDH key exchange to generate a shared secret. Then the shared secret will be hashed by the SHA512. The hash result will be splited into two part, the first 32 bytes will be used as the encryption key k_e, the last 32 bytes will be used as the HMAC key k_m. The sender can optionally use his private key to generate a signature from his public address and the plaintext. Using the encryption key k_e to encrypt the plaintext and the signature to generate the ciphertext. Then generating the MAC code with HMACSHA256, using the k_m as salt to authenticate (timestamp + destination + ciphertext).
The encryption scheme will go like this:

1. Sender's key pair: $(x, g^x) \in_R \mathcal{Z}_p \times$ Curve25519
2. Get receiver's public key: g^y
3. Shared secret = $(g^{y \cdot x})$
4. Hash result = SHA512(Shared secret)
5. Hash result = $\underbrace{\underbrace{k_e}_{32bytes} + \underbrace{k_m}_{32bytes}}_{64bytes}$
6. Signature = $(g^x, plaintext)_x$
7. Ciphertext = $ENC_{k_e}(plaintext, Signature)$
8. MAC = $MAC_{k_m}(timestamp, destination, ciphertex)$

The final message will contains: MAC, Ciphertext, sender's public key, destination(receiver's public key).

2.7 STORJ

StorJ is another fork application of the Bitcoin, it's a Blockchain-Based decentralized file storage application. It can provide the opportunities for users to rent their storage to others to earn money. In this case, if user has the needs to upload their file, they will broadcast the request to ask someone's storage. When the sender receives the at least 3 Acks, sender will encrypt and transmit the file to the storage providers whiling the file will be replicated to 3 copies. If there are more than 3 Acks, the file will be partitioned to several shards and each shard will be replicated to 3 copies. After uploading the file, the sender will send an evidence which will be stored into the blockchain, there is no full detail about the evidence in their whitepaper, we only know that the evidence at least contains the uploader's public key, the file name, file hash, each shards hash and a Merkle root of the file shards' hashes. The figure (figure 3) shows the part of the smart contract information of the StorJ. It provides us an idea of how to reduce the size of the blockchain in the messaging system and the searchable encryption might be also used in the future.
For the file storage system, people usually care about the files' availability and confidentiality. The confidentiality of the files will be ensured by the standard encryption

and the availability will be ensured by the original scheme of the StorJ. Before sender uploads the files to the service provider, sender will add the seeds and generate unique hash. Then the user will upload the file and periodically issue the seeds if the service provider return the correct hash which means the files are not deleted or modified. Another problem is the case that the service provider is not always rent his storage to tenants, he can leave anytime or just stop provide service. This will affect the availability of the files to the owner. So, StorJ design a protocol that once the service provider wants to stop his service, he cannot leave until he transfers the files to another available service provider and broadcast the smart contract to the peers. People also concern about how does the owner search their files, the owner can search over the blockchain by the file name or file hash to find the file's location then generate request to service provider, then the service provider will return the file that the owner requests. So, there is a point that storJ can improve is that provide the searchable encryption to provider higher level secure for the files.

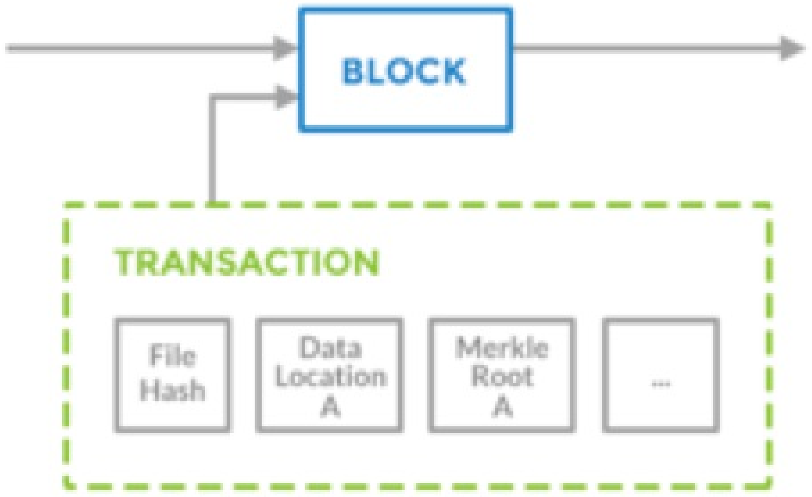


Figure 3: The evidence of the StorJ system

3 EXISTING PROBLEMS (OUR CONTRIBUTION)

3.1 Problems of the centralized messaging system

Previous messaging systems usually considered the server is not malicious, they can just partially prevent the attack from the server. If the server is malicious or attacked by the malicious attacker, the messages or keys will be not delivered successfully. And the messages which stored at the server also faced the risk from the attackers. We can take Signal as an example, there are at least two types attacks:

1. When sender asks the server for the receiver's public Prekey to start a conversation, the malicious server can return a wrong key to the sender, although the malicious server cannot decrypt the message, but the receiver can not neither. This problem

caused by the public Prekey cannot be verified. Despite the fact that Signal provides the fingerprint to help the participants verify their identity via out of band channel, this kind of attack will also influence the service between the participants.

2. Servers also have the possibilities to perform DoS attack, once the servers are hacked, they reject to provide any service for the users, the system will be paralyzed. For example, rejecting to deliver the keys or messages will affect the conversation.

The two reasons show us the servers are extremely powerful in the centralized messaging systems which might be the bottleneck of the systems sometimes. Even some tiny mistake will lead the system collapsed. But, there is a tradeoff that centralized server have better performance to provide the store-and-forward service for the users. The decentralized messaging system should be considered if they can provide same level service as the centralized system and reduce the risks from the attacked or malicious servers.

3.2 PROBLEMS OF THE DECENTRALIZED MESSAGING SYSTEM

I want to separate this part into two parts: The decentralized messaging systems with blockchain and without blockchain.

3.2.1 DECENTRALIZED MESSAGING SYSTEM WITHOUT BLOCKCHAIN

With the development of the hardware and technology, the P2P networks improve their performance, but the kinds of systems still have two main problems:

1. Wasting the network resources. Most of the P2P systems use the Distributed Hash Table (DHT) which has good performance of the file sharing, because all the participants are online users and everyone help upload the target files to reduce the total time. But the DHT is not reasonable for the messaging systems, because there are large number of offline messages which will be transferred in the DHT circle until the receiver is online or the messages are out of TTL(Time To Alive). Due to this kind of property, the offline messages will occupy the network resource for a long time. Also, according the nature of the messaging systems, the amount of users is extremely large and the users login and exit frequently, so the DHT is difficult to accurately record the current number of online users.

2. Hard to provide the Forward Secrecy. Because this kind of system does not have the server or other parts of storage to store and forward the keys which will be used in the message encryption, so the participants cannot get the keys directly to do the key exchange to generate symmetric keys. So the Forward Secrecy will be provided with additional step, via out-of-band channel or additional messages for key exchange.

3.2.2 DECENTRALIZED MESSAGING SYSTEM WITH BLOCKCHAIN

There are few Bitcoin like messaging systems like BitMessage, ShadowChat, and these systems have their unique encryption scheme developed by their designers. But encryption schemes are not meeting the current demands of the messaging system, because it cannot provide the Forward Secrecy. These systems at least tell us the blockchain can be also used in the messaging systems and provide reliable service as the traditional systems, and blockchain might be a better choice than the other decentralized scheme.

3.2.3 OUR CONTRIBUTION

According to the existing problems of the centralized and decentralized messaging systems, we issue the new demands of the messaging system to meet current users' needs and these properties are our design baseline. We propose several properties that the messaging system must achieve that can provide much more reliable secrecy of the system.

1. Forward/Backward Secrecy: If a system uses the same static key for all the conversations, when this key is compromised that all the messages which are encrypted by this key are insecure. That's why we want Forward/Backward Secrecy to avoid this problem. A protocol provides Forward Secrecy that when one key is compromised and this compromised key cannot decrypt previous sessions' ciphertexts. A protocol provides Backward Secrecy that when one key is compromised and this compromised key cannot decrypt the subsequent sessions' ciphertexts. So the Backward Secrecy also referenced as the Future Secrecy. As the figure (figure 4) shown.

2. Message Deniability: Deniability, also called reputability, which is desirable goal for the secure messaging systems. The high-level goal of the messaging system like the real-world conversation, we can consider a scenario that two people talk in the room, and an eavesdropper tapping out of the room, there is no evidence to prove which words were said by one specific person and the eavesdropper cannot accuse the word is said by a specific person of the participants. In the messaging systems, the valid cryptographic digital signature provides strong evidence that the message belonged to whom, so called non-repudiation. So in order to achieve the Deniability, the signature scheme cannot be used, instead of using the MAC or HMAC to provide the authentication of the messages.

3. Asynchronicity: Due to the mobile systems' features, the user always login and exit frequently, like switched-off devices and weak signal, so we cannot ensure the receiver is online when he has incoming messages, that why we need the asynchronous system. From example, the P2P architecture provide the synchronous network in most case that needs the participants must be online at the same time during the message transmission or file sharing. To provide the asynchronous system, the messaging systems usually apply the store-and-forward model that employ the server buffer the messages when the recipient is offline.

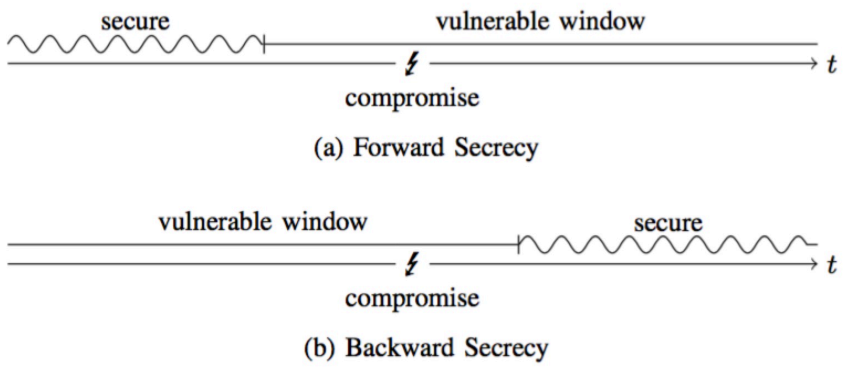


Figure 4: Session keys are protected from long-term key compromise.

4. Decentralization: A decentralized messaging system indicates there is no central server provides the service for the users, the service includes store-and-forward messages and keys. So, a decentralized messaging system is a highly autonomic system which can prevent the attacks from the malicious server. But according to the nature of the Peer-to-Peer network, the decentralized architecture is hard to provide the asynchronicity unless the blockchain scheme is implemented.

4 SYSTEM MODELING AND STRUCTURE

4.1 NODES

Nodes means the online users in the P2P network, because the P2P network needs every online user takes part in the communication, so each node needs to know which nodes are near to him and can help him forward the message to the next hop until arrive the destination. Each node may have different role in the network which means some nodes may have more duty than other nodes that indicates the full nodes.
Bitcoin is a good example, which can provide users various roles with diverse functions. We also employ the same method in our system. A node contains all the functions called full node, which has the functions:

1. Maintaining a list of a part of the online users (not all the online users) and the list will be dynamically updated when receives other node's message or request. It means this router plays a role as the network router.
2. Basic messaging function, each node can send and receive the message. But distinct from the traditional messaging system, message will be sent with a specific header.

When node receives the message, node verifies the header first and then records the messages in the block whether the message belongs to him, then forward the message to the next hop.

3. Having all the blockchain's history, this function can help the offline users who reconnect the network can get the messages when they are online. In a simple word, this is the main part to provide the asynchronous network, because all the the online users download history of the messages when they reconnect, and find the messages belongs to them locally.

4. Mining, simplify this process, it's a computation of the hash collision to compete who has the right to broadcast the history in this time interval. It's the process of finding the proof-of-work.

4.2 MESSAGES

There are several types of messages and request in this system. Different messages or requests have different headers and roles.

4.2.1 UPDATE LIST

When user just accesses the network, user has zero knowledge of the peers in the network, so user needs a default IP who is always in the network to help him to update the list of the online users. This request will contain sender's public key which used as the address, sender's IP, timestamp and type: a specific string as "list". This request happened between two online users, it's an end-to-end connection, so the IP address should be shown, thus the receiver can send back the list immediately. Timestamp will help the receiver distinguish whether this request is out-of-date. We need claim that the default IP can be considered as a server which only provides the list of the online users, not participants in the messages and keys transmission.

4.2.2 UPDATE BLOCKCHAIN

Node broadcasts another new request to peers after getting the list of online users(user knows the peers now). This new request means to get the part of history which happened when the user is offline. This request's header will contain the sender's public key, sender's IP, timestamp, type: a specific string update, and the payload is the highest height of sender's existing blockchain. This request also happened between two online users, after receiver verifying the type, receiver will check the payload, calculate how much data does the sender lack, and send the data to the sender. So every user will get a full blockchain history since he updates the blockchain every reconnection. This is also why the blockchain scheme can make the Peer-to-Peer network achieve the asynchronicity.

4.2.3 MESSAGE (UPLOAD KEYS)

This message will be stored into the blockchain. This message's header will contain sender's public key, timestamp, type: a specific string keys and the payload will contain the Prekeys will be used in the conversation's encryption. According to existing Client-Server model messaging system, the server saves users' key(s) without encryption, this is one of the core parts to provide the Forward Secrecy. Also, in our system, the users need to have the social knowledge, thus they can establish the trust between each other. Because we don't have a server so there is no one match the ID or phone number with their public address, so if participants want to establish the trust they need social knowledge. Social knowledge means they need to exchange the public key address in person or via out-of-band channel like via other email or other app. This scheme cost more user effort, but we don't want user links their phone number or email address to expose their privacy.

4.2.4 MESSAGE (CONVERSATION MESSAGE)

This message will be stored into the blockchain. This message's header will contain sender's public key, receiver's public key, type: a specific string "bmsg". The payload is the encrypted conversation. The sender's public key and receiver's public key will be utilized to generate the envelope key and as the address of the sender and receiver. When the peers receive this message, they will verify the type and receiver's public key whether belongs to themselves, if yes, try to decrypt the message; if not, store the message into their current unconfirmed block and forward this message to their peers, their peers will do the same step. And this encrypted message will be delivered to the receiver finally if the receiver is online or the receiver can download the blockchain and find his messages when he is reconnected.
So, the Bitcoin-like messaging system looks like a public wall that every users' encrypted messages are written on this wall, but only the right person can decrypt it. Actually, this scheme also can be enhanced in future, we will discuss in the section 8.

4.3 BLOCKCHAIN

We can consider the blockchain is a linked list like database, every block is a database with a fixed header. The block header contains blockhash, timestamp, nonce, height, difficulty, previous blockhash, next blockhash. The height represents the ID of the block and also indicates how many blocks are owned by this user currently, because there are several blocks in the blockchain, these blocks are arranged in the order of the height. The blockhash is a result of the hash collision, the difficulty means how many bits need to be collided, it's a standard of the random target generation. The nonce shows how many times does the computation to collide with the target. So the users can use the timestamp to estimate how main blocks does the main blockchain have, use the previous blockhash, blockhash and next blockhash to form the blocks in the order, use the nonce and blockhash to verify whether the block is valid. And every block will contain the history took place in that time interval. And the difficulty is helping the justify the main blockchain because the main blockchain always has the highest difficulty.

4.4 PROOF OF WORK

It is the process of the hash computation. The meaning of the proof-of-work, all the peers have the history in this time interval, but the history may have a little

differences caused by the network delay or other reason like the miner just reconnects and doesn't record the full time interval's messages, so all the peers need a unified history. Because this is a decentralized system, no one can determine who always has the right to broadcast his history. So the proof-of-work gives every peer an equal right to compete for the opportunity to write the blockhash and share this time interval's history they recorded with other peers.
Describing the proof-of-work in detail, it looks like to find a hash collision. When each user creates their new unconfirmed block, a target will be generated automatically and it can be considered as a random number. Then the user will use a constant string with increasing number do the hash in the loop, like hash(I am Groot + 0), hash(I am Groot +1), every hash result will be compared with the target, when they are collided with the target which means this block is mined, and this block will be broadcasted to all the users of the network.
This step is the key part of the decentralized system with blockchain, every user has the opportunity to play the role of server and the blockchain can provide the store-and-forward service like central server. That's why we choose the decentralized network to reduce the risks from the server, because the attacker cannot determine the next block mined by the whom. The next miner who mines the block, based on the computation abilities among the users are not very different, so everyone might be the winner. But there is also a risk that once someone or party has over half of the computation power of the network, he can easily control the whole network which called 51% attack, but it's hard to achieve in fact.

4.5 FORKING

According to the Bitcoin system, there is the possibility that two users mine the same block at the same time, so the fork will happen. To avoid forking, every block's target will be changed into integer called difficulty. The main blockchain always has the highest difficulty, if two users mine the same block at the same time with similar difficulty, then the two block will be confirmed and added to the blockchain, so there are two blockchain exists. All the users need to choose which fork they will follow, after the new blocks mined, we can compare the two new blocks' difficulties, the block with highest difficulty will be added to the main blockchain, the other one will be dropped. If the two new blocks still have similar difficulties, we can compare the following blockchains' difficulty, until we find a blockchain with higher difficulty, the other one will be dropped. Because the hash collision is extremely hard to get, the fork will be happened rarely.

4.6 51% ATTACK

The 51% attack is theoretically risk of the blockchain but never happened, it indicates some entities get over 50% computation power of the whole network, then they can influence the existing blockchain. Explaining this problem in detail, the malicious user gets the block's difficulty is higher than others but he doesn't broadcast it. Then, the following ten blocks' difficulties are still higher which calculated by this malicious user, then this attacker broadcasts the eleven blocks to the peers. The peers will find that this fork has higher difficulty than the existing blockchain, so the eleven blocks

will be added to the blockchain from where the height it begins by the malicious user, and this fork will be considered as the main blockchain. So the previous eleven blocks will be dropped, and the transactions which recorded by the honest parties will be dropped neither.
Taking Bitcoin as an example to illustrate this problem, somebody may use his bitcoin buy some food during that time interval, but this transaction might be not recorded by the malicious attacker, so the bitcoin is came back the purchaser he can spent this money again. This attack will also affect the messaging system, when the 51% attack happens, some important messages might be dropped by the attacker that the receiver cannot receive them anymore, it will influence the availability of the system. However, this attack had never happened before, and according to the real-world situation, it's hard to achieve this attack.

5 THE WORKFLOW OF THE SYSTEM

When a new user Alice accesses the system and wants to communicate with her friend Bob.

1. Alice is required to communicate with default IP, send a request "list" to update the list of the other online users.
2. After getting the list, Alice will broadcast the request "update" and the block height he owns to the peers and download the blockchain's history from the peers.
3. Meanwhile, Alice will generate 50 Prekeys, and send to peers. Peers will verify this message header and store the Prekeys into the blockchain. Actually the Prekeys will be stored into the unconfirmed block, after the unconfirmed block is mined, this unconfirmed block will be added to the blockchain.
4. Alice will search one of Bob's Prekeys in the blockchain, the blockchain is downloaded, the history of the blockchain becomes local data.
5. After Alice getting Bob's Prekey, Alice can generate the encrypted message, the encryption scheme I will discuss in the next section.
6. Alice broadcasts the encrypted message, each node collects the message into a block (unconfirmed block), the message will go through the entire network. If Bob is online, Bob can receive this message directly and decrypt the message. (the communication is established)
7. Each node works on finding the proof-of-work, to fight for the right to share the unconfirmed block change to confirmed and add it to the blockchain.
8. When a node finds the proof-of-work, it will broadcast this block to all peers in the network (sharing).
9. Nodes accept this block only if the blockhash is matched with the computation of the nonce. This step may have bug, attacker can just use the same header and generates the useless messages which the receiver cannot decrypt it. But the fact

is the competition of the computation is fierce, if the attacker does this additional step, his computation will be slower than others so that attacker might lose the opportunity.

10. Nodes express their acceptance of the block by creating the next block in the blockchain following this accepted block, and using the hash of the accepted block as the previous hash.

Depends on whether the receiver is online or offline, there are two type situations. If the receiver is online, the following conversation will go like the normal P2P network obviously. On the other hand, if the receiver is offline, when the receiver reconnects the network, he will download the blockchain automatically and search the messages which belongs to him then check and decrypt them. The figure (figure 5) illustrates the simplify message flow of the system.

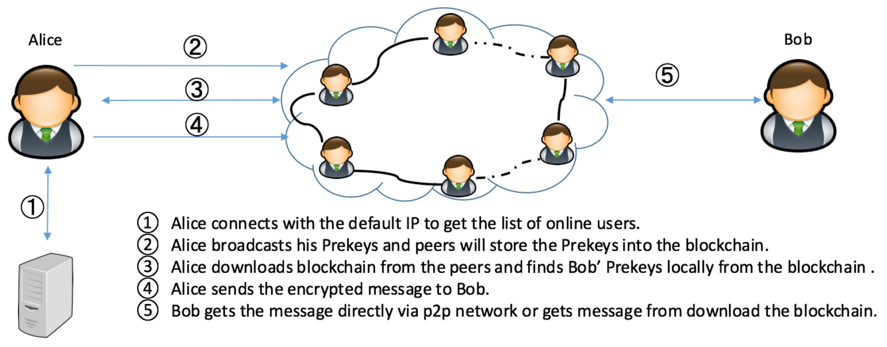

Alice
Bob
②
③
④
⑤
①
① Alice connects with the default IP to get the list of online users.
② Alice broadcasts his Prekeys and peers will store the Prekeys into the blockchain.
③ Alice downloads blockchain from the peers and finds Bob' Prekeys locally from the blockchain .
④ Alice sends the encrypted message to Bob.
⑤ Bob gets the message directly via p2p network or gets message from download the blockchain.

Figure 5: Message flow

6 ENCRYPTION SCHEME

Basing on existing messaging systems which can provides the Forward Secrecy, the users need to upload their keys to the server when they register in the system, that's why we have to store the public Prekeys into the blockchain which we discussed in the section 4.2.3, this is called Prekey scheme. The difference between the centralized messaging system, the public Prekeys are transparent to all the users, everyone can access them but cannot modify them. So the sender can get the receiver's public Prekeys without server or additional effort. After getting the receiver's Prekey, the following key agreement and key management we use the Axolotl protocol, we will discuss this encryption scheme in detail in this section. To achieve the Forward Secrecy, before starting a conversation, the sender must get the public identity key g^b of the receiver via out of band channel and the public Prekey $g^{x_{b,z}}$ belongs to the receiver which searched in the blockchain locally. Now, the sender will use his own private identity key a, one of his private Prekey $x_{a,0}$, receiver's public identity key g^b and receiver's public Prekey $g^{x_{b,z}}$ to go through the 3-Diffie Hellman key exchange protocol to generate the shared secret. After sender completing the key agreement, sender will use the shared secret go through the Axolotl protocol to derive two symmetric keys $k_{BA,r}$ and $k_{BA,c}$. Now sender will choose another private Prekey $x_{a,1}$ which will be used with $g^{x_{b,z}}$ to generate the k_{shared}, the k_{shared} and $k_{BA,r}$ will go through the Axolotl protocol again to generate the $k_{AB,r}$ and $k_{AB,c}$, then the $k_{AB,c}$ will go through the Axolotl protocol to generate the k_{ENC} and k_{MAC} which will be used as the encryption key and MAC key in the message encryption. After this, the $k_{AB,c}$ will be used as the MAC key to encrypt a String to generate a new $k_{AB,c}$, this step to ensure every message will be encrypted with a different key and both of the sender and receiver can generate every time. Replying the message by the receiver doesn't need to search for a new public Prekey of the sender, just use the public Prekey in this session to derive new k_{ENC} and k_{MAC}. If the receiver doesn't reply the message, the sender can just go through the Axolotl protocol to derive the new k_{ENC} and k_{MAC} for the following-up messages. We can use a diagram (figure 6)to illustrate the process of the encryption scheme.

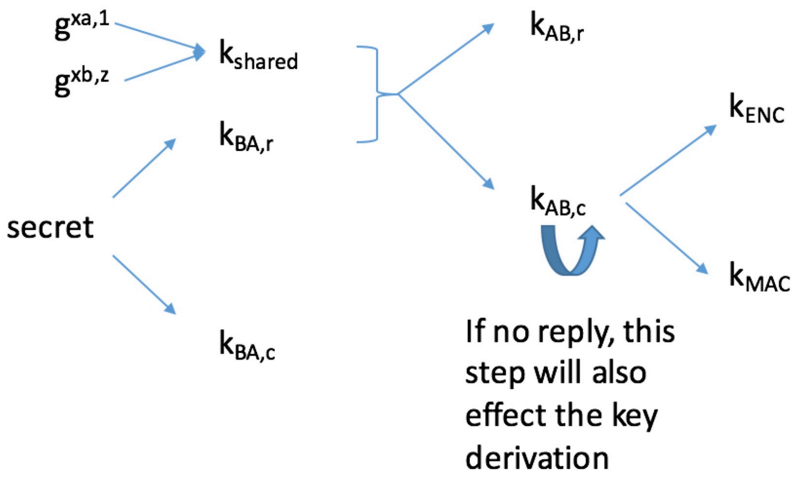


Figure 6: The key derivation of the system

Describing the encryption scheme in detail:

1. Sender gets the recipient's public key g^b out of band channel.

2. Sender gets the recipient's public Prekey $g^{x_{b,z}}$ from the blockchain.

3. Sender chooses one of his Prekeys pair $(\bar{x}_{a,0},\ g^{\bar{x}_{a,0}})$ then does the 3DH key exchange: secret $= (g^{\bar{x}_{b,z} \cdot a},\ g^{b \cdot \bar{x}_{a,0}},\ g^{x_{b,z} \cdot \bar{x}_{a,0}})$

4. Sender chooses another Prekey pair $(\bar{x}_{a,2},\ g^{\bar{x}_{a,2}})$, then does the ECDH: $k_{shared} = g^{x_{b,z} \cdot \bar{x}_{a,2}}$

5. Then, using the secret go through the Axolotl protocol:

   $k_{pr} \longleftarrow MAC_{0x00^{32}}(secret)$

$k_{BA,r} \longleftarrow MAC_{k_{pr}}("WhisperRatchet", 0x00)$
$k_{BA,c} \longleftarrow MAC_{kpr}(k_{BA,r}, "WhisperRatchet", 0x01)$

6. Using the k_{shared} go through the Axolotl protocol again:
   $k_{pr} \longleftarrow MAC_{k_{BA,r}}(k_{shared})$
   $k_{AB,r} \longleftarrow MAC_{k_{pr}}("WhisperRatchet", 0x00)$
   $k_{AB,c} \longleftarrow MAC_{k_{pr}}(k_{AB,r}, "WhisperRatchet", 0x01)$
7. Using $k_{AB,c}$ go through the Axolotl protocol again:
   $k_{pr} \longleftarrow MAC_{0x00^{32}}(MAC_{k_{AB,c}}(0x00))$
   $k_(ENC) \longleftarrow MAC_{k_{pr}}("WhisperMessageKey", 0x01)$
   $k_{MAC} \longleftarrow MAC_{k_{pr}}(k_{ENC}, "WhisperMessagekey", 0x01)$
8. Then, generating a new $k_{AB,c}$: $k_{AB,c} \longleftarrow MAC_{k_{AB,c}}(0x02)$

Why we choose this encryption scheme, there are some reasons:

1. Both of the sender and the receiver can generate the k_{ENC} and k_{MAC}, so the message Deniability can be ensured, because both of the participants can deny they encrypt the message and no signature is used, which contrasts with the non-repudiation.
2. Every message will be encrypted by different symmetric keys which can be derived by both of the sender's and receiver's key materials, to ensure the confidentiality of messages and provide the Forward Secrecy. As the figure (figure 6) shown, each of $g^{\bar{x}_{a,1}}$, $g^{xb,z}$, $\bar{x}_{a,0}$ will change the value of the k_{ENC}. Also the $k_{AB,c}$ also change itself every single time.
3. Comparing the existing P2P messaging systems, we construct our system with blockchain, we can store the Prekeys into the blockchain that the user can just search them locally for the key exchange without additional effort.

7 EVALUATION

Recently, DARPA put forward new requirements of the messaging system, they want the new generation messaging system which developed with the blockchain and provide secure communication. Our design can meet the demands of the DARPA, provides reliable encryption, desirable properties and avoids the problem of the traditional client and server model system. As we mentioned in section 3.2.3, we consider the 4 aspects as our design baseline. But there are some other properties that the instant messaging system needs to achieve considered. We also use a table (table 2) visualizing our evaluation of the apps within the problem area.

- Confidentiality: Only the intended recipients are able to read a message. Specifically, the message must not be readable by a server operator or anyone else who is not conversation participant.

- Integrity: No honest party will accept a message that has been modified in transit.
- Authentication: Each participant in the conversation re- ceives proof of possession of a known long-term secret from all other participants that they believe to be participating in the conversation. In addition, each participant is able to verify that a message was sent from the claimed source.
- Network MitM Prevented: Prevents Man-in-the-Middle (MitM) attacks by local and global network adversaries.
- Privacy Preserving: Anybody can see the message belong to which sender address and receiver address, but they don't know these address belongs to which person.
- Operator MitM Prevented: Prevents MitM attacks executed by infrastructure operators.
- Operator MitM Detected: Allows the detection of MitM attacks performed by operators after they have occurred.
- Operator accountability: It is possible to verify that operators behaved correctly during trust establishment.
- In-band: No out-of-band channels are needed that require users to invest additional effort to establish.
- Immediate Enrollment: When keys are (re-)initialized, other participants are able to verify and use them immediately.
- No Service Provider Required: Trust establishment does not require additional infrastructure (e.g., key servers).
- Participants Consistency: At any point when a message is accepted by an honest party, all honest parties are guaranteed to have the same view of the participant list.
- Destination Validation: When a message is accepted by an honest party, they can verify that they were included in the set of intended recipients of the message.
- Causality Preserving: Implementations can avoid displaying a message before messages that causally precede it.
- Global Transcript: All participants see all messages in the same order.
- Message Unlinkability: If a judge is convinced that a participant authored one message in the conversation, this does not provide evidence that they authored other messages.
- Out-of-Order Resilient: If a message is delayed in transit, but eventually arrives, its contents are accessible upon arrival.
- Dropped Message Resilient: Messages can be decrypted without receipt of all previous messages.

- Trust Equality: No participant is more trusted or takes on more responsibility than any other.
- Contractible Membership: After the conversation begins, participants can leave without restarting the protocol.

Table 2: The other desirable properties. $\bullet : fullyachieved \circ partiallyachieved- : nonachieved$

apps properties	Signal	Threema	Bitmessage	ShadowChat	our design
Confidentiality	●	●	●	●	●
Integrity	●	●	●	●	●
Authentication	●	●	●	●	●
Network MitM Prevented	○	●	●	●	●
Privacy Preserving	●	●	●	●	●
Operator MitM Prevented	○	○	●	●	●
Operator MitM Detected	○	●	●	●	●
Operator Accountability	○	●	–	–	–
In-band	●	●	-	○	–
Immediate Enrollment	●	○	●	●	●
No Service Provider Required	●	●	●	●	●
Participants consistency	●	●	●	●	●
Destination Validation	●	●	●	●	●
Causality Preserving	●	●	●	●	●
Global Transcript	-	-	●	●	●
Message Unlinkability	●	-	-	-	●
Out-of-Order Resilient	●	●	●	●	●
Dropped Message Resilient	●	●	●	●	●
Trust Equality	●	●	–	–	–
Contractible Membership	●	●	●	●	●

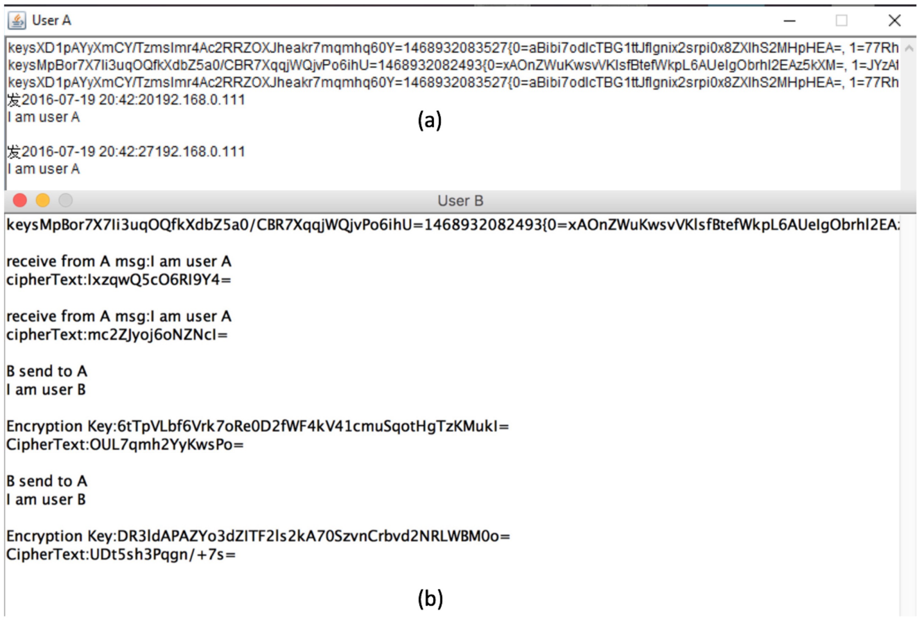
Despite achieving all the aspects, our design also has some performance need to be improved considerably.

1. The problem of the blockchain itself, the resources of the storage will be wasted. Every user must store all the transcripts of the whole network, which sounds not reasonable, but it's the price of decentralization. According to the Bitcoin, it costs over 20 GB storage of each user, the ledger is from 2009 to 2015. So the cost of our design is affordable basing on the existing hardware.
2. Basing on existing design, all the cipher message will be stored into the blockchain. There is the potential risk of the ciphertext-only attack. If the attacker can determine the public identity key belongs to which entity, attacker can find all the messages related to the victim. Although the encryption scheme is secure in nowadays, but this problem should be considered.
3. Our system uses the public key as the participants' addresses, it also has some problems. According to the Bitcoin, the public key used as the address of the users can provide privacy preserving theoretically. But the reality is that the public key is related to the IP address, and the IP address reveals a lot information of the users,

this is the TCP/IP network's problem, unless we change the network architecture that we can solve the problem. The second issue is that we don't band the public key with the phone number, because we consider the phone number contains much more information about users, any malicious user can use the phone number to find the victim's privacy even the attacker is not an expert of the computer. That's why we give up the scheme with will cost more user effort.

4. As I mentioned in this report, to provide the Forward Secrecy must store the public Prekeys where the sender can access that the normal P2P network is hard to achieve. But blockchain can solve this problem but not perfectly, after user registers and upload his public Prekeys, the sender must wait at least one block confirmed, then the sender can just start the conversation, this is the price of the decentralization too. Because the sender cannot receive the public Prekeys from the receiver or server directly. Sometimes, the conversation message may also have this problem due to the same reason. We can consider the scenario that the sender just sent the message to his peers and peers recorded this message but this block is not mined yet, and meanwhile the receiver is reconnected. This message is not delivered over the P2P network and the block which contains this message is not mined neither, so the recipient needs to wait a few minutes until this block is confirmed.

7.1 PERFORMANCE RESULT



```
keysXD1pAYyXmCY/Tzmslmr4Ac2RRZOXJheakr7mqmhq60Y=1468932083527{0=aBibi7odlcTBG1ttJflgnix2srpi0x8ZXlhS2MHpHEA=, 1=77Rh
keysMpBor7X7li3uqOQfkXdbZ5a0/CBR7XqqjWQjvPo6ihU=1468932082493{0=xAOnZWuKwsvVKlsfBtefWkpL6AUelgObrhl2EAz5kXM=, 1=JYzAt
keysXD1pAYyXmCY/Tzmslmr4Ac2RRZOXJheakr7mqmhq60Y=1468932083527{0=aBibi7odlcTBG1ttJflgnix2srpi0x8ZXlhS2MHpHEA=, 1=77Rh
发2016-07-19 20:42:20192.168.0.111
I am user A

发2016-07-19 20:42:27192.168.0.111
I am user A
```

```
keysMpBor7X7li3uqOQfkXdbZ5a0/CBR7XqqjWQjvPo6ihU=1468932082493{0=xAOnZWuKwsvVKlsfBtefWkpL6AUelgObrhl2EA:

receive from A msg:I am user A
cipherText:IxzqwQ5cO6RI9Y4=

receive from A msg:I am user A
cipherText:mc2ZJyoj6oNZNcl=

B send to A
I am user B

Encryption Key:6tTpVLbf6Vrk7oRe0D2fWF4kV41cmuSqotHgTzKMukI=
CipherText:OUL7qmh2YyKwsPo=

B send to A
I am user B

Encryption Key:DR3ldAPAZYo3dZITF2ls2kA70SzvnCrbvd2NRLWBM0o=
CipherText:UDt5sh3Pqgn/+7s=
```

Figure 7: Snapshot of the system

According to the nature of the blockchain scheme, each user plays the both roles of client and server, and the user who computes the proof-of-work will play central

server's role temporarily, he will broadcast his current block when the computation complished. Thus every online user can get this block via the network. On the other hand, blockchain scheme provides the asynchronous network to make the Forward Secrecy accomplished. We will use the figure(figure 7) to illustrate what we want to achieve.
7.(a) and 7.(b) first line indicates the participants broadcast their Prekeys, and the Prekeys will be stored into the blockchain. The duplicate Prekeys occurs in the second and third lines of the 7.(a), because the user B had already finished the hash computation, and broadcast the content of his current block which contains user A's and user B's Prekeys. The following lines in the 7.(a) express user A sent two same messages to user B. We can see the result in the 7.(b) that these two same messages sent from user A are different ciphertexts. From the 7.(b) we can also find that user B sent two same messages are encrypted with different keys. So, we can find that our system achieves the Forward Secrecy successfully.

8 FUTURE WORK

Because of previous problems that our system has, we will continue our research in the future, mainly about reducing the blockchain's size, making the messages much more secure. So we consider combine the searchable encryption with the messaging system, but the problem is that the Forward Secrecy which is the most desirable property needs to change the encryption key for every message, but the searchable encryption is related with the encryption data without changing the key. Once this puzzle is solved, we can rebuild our system and make it more secure and more efficient.
The ideal work flow will be divided into two parts, online message and offline message, depends on whether the receiver is online. Users still need to upload their public Prekeys into the blockchain, and they can do the key exchange and key management as this report mentioned, the difference is that before sending the message, sender has a lookup scheme to check whether the receiver is online. If receiver is online, the following messages will be delivered to the receiver directly because of the lookup scheme, sender can build the connection with the receiver. If the receiver is offline, the sender will broadcast a request to find the storage providers. After sender receiving the replies from the storage providers, sender can generate the encrypted messages but the messages are not sent yet. After one new block is confirmed, sender will send batch of the encrypted messages to the storage provider, we should be aware of that the storage provider will be offline, so the encrypted messages should be duplicated and stored at several places. Then the sender will broadcast the evidence where the encrypted messages stored and who can decrypt the messages. So the receiver can scan the blockchain to find where his messages stored, then he can get the encrypted messages belonged to him. This scheme will reduce the size of the blockchain, and make the transcripts more secure because nobody can access the encrypted messages directly.
The ideal situation is that we can combine the searchable encryption with the messaging system and achieve both of the searchable encryption's and messaging system's demands. But, the reality is that the Forward Secrecy is a little bit conflict with

searchable encryption. Because the searchable encryption is searching over the history, but the Forward Secrecy of the messaging system makes each message encrypted with different key which makes the search difficult and inefficient. Apart this, we also need to concern about whether the searchable encryption scheme is appropriated, because comparing to the traditional scenario of the searchable encryption, messaging system will have a lot of operations of the inserting. Due to this reason, we prefer the scheme which can provide keyword search without building index, that will make the system more reasonable.

8.1 SYSTEM ARCHITECTURE

To provide the searchable encryption, our original system is not suitable, because the user needs to access the data from the remote server. The searchable encryption aims to prevent the server's attack and ensure the data security.
As we mentioned in the previous section, searchable encryption is conflicted with the definition of the Forward Secrecy of the messaging system. And we do not figure out the suitable scheme to solve this problem, so we give up the Forward Secrecy in this section. Maybe we can find a reasonable way to address this problem in the future, so that we can improve our work.
The new architecture we will combine the blockchain protocol, StorJ storage scheme and smart contract, a look-up scheme similar like the BitMessage, and Hahn's work[Hahn et al.'14] to provide the searchable encryption. In this part, we do not introduce the blockchain scheme again, because it's similar like the previous work.

8.1.1 WORKFLOW OF THE SYSTEM

We can consider the scenario that Alice wants to send messages to Bob. After Alice getting the list of the peers the following steps:

1. Alice runs look-up scheme to check whether the Bob is online. The look-up scheme we implement the Bitmessage's scheme.

2. If Bob is online, the message will be delivered to the recipient over the P2P network.

2. If Bob is offline, Alice will broadcasts a request to peers to find the fixed service providers to save the messages. The diagram shows the workflow if Alice knows Bob is offline (figure 8).

3. After receiving the ACK from the service provider, Alice encrypts the message in an appropriate way and send the encrypted to the service provider(Eve).

4. Then Alice will broadcast an evidence to the peers that he stores the message at Eve(service provider) and the message is belonged to Alice and Bob.

5. When Bob reconnects to the network, Bob will download the blockchain from the peers. Bob will check the history of the blockchain, Bob will send the request to Eve to download the message which belongs to him.

6. The following messages will be sent to Eve by both of Alice and Bob.

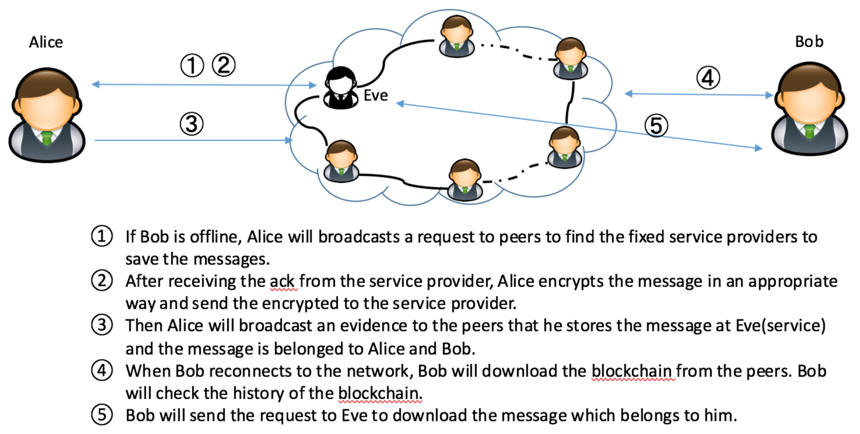

Alice
① ②
③
Eve
Bob
④
⑤
① If Bob is offline, Alice will broadcasts a request to peers to find the fixed service providers to save the messages.
② After receiving the ack from the service provider, Alice encrypts the message in an appropriate way and send the encrypted to the service provider.
③ Then Alice will broadcast an evidence to the peers that he stores the message at Eve(service) and the message is belonged to Alice and Bob.
④ When Bob reconnects to the network, Bob will download the blockchain from the peers. Bob will check the history of the blockchain.
⑤ Bob will send the request to Eve to download the message which belongs to him.

Figure 8: Message workflow that Bob is offline

The blockchain will store the evidence of the message's location that the recipient can find the service provider to download the his message via our system. The evidence will look like figure (figure 9).

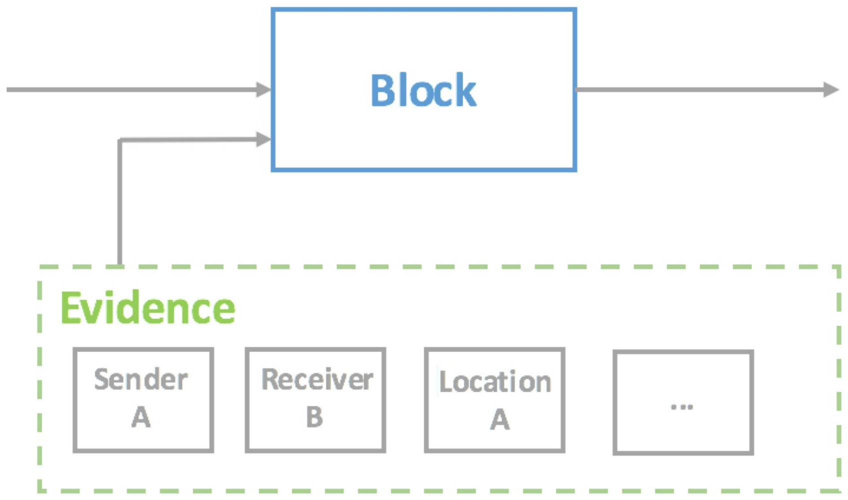


Figure 9: The evidence in the blockchain

Actually, Alice not only saves the encrypted message at Eve, but also saves the search index and search history. We will discuss this in the following part.

8.2 Keyword search

As we said, the encrypted messages just store at the blockchain is not reasonable, because every user can access the messages directly, it makes the message's security under huge potential risks. So we consider implement the symmetric searchable encryption to avoid this problem. There are at least three aspects we concern about the searchable encryption.

- *Dynamic*: Data can be added, deleted and hence changed after the initial outsourcing. In fact, we do not need a specific operation for initial outsourcing and assume all data is added incrementally.
- $Efficient$: the scheme must have asymptotically optimal, sublinear search time. We only require to store 2 cryptographic hash values per keyword and document.
- *Secure*: The security must use a simulation-based definition. The leakage functions which are significantly more restrictive. Loosely speaking, the scheme is semantically secure unless a search token has been revealed.

The three aspects are our baseline to employ the searchable encryption. We will use a table to demonstrate the efficiency of the existing works of the searchable encryption. Comparing to other schemes, Hahn's work [HK14] is much suitable for the messaging system, because this scheme leaks no more information than the access patter when updates, also has reasonable search time, index size and do not require the clients store their data locally.

Table 3: The existing work of searchable encryption

scheme	search time	Index Size	Client Storage	Update Leakage	Update Cost
[SPR12]	O(m/n)	O(m+n)	O(1)	ID(w)	O(m/n)
[NPG14]	O(m/n)	O(m+n)	O(1)	ID(w)	O(m/n)
[KP13]	$log(\|f\|\cdot$ m/n)	O($\|f\|\cdot$ n)	O(1)	-	O($log\|f\|\cdot$ n)
[CJJKRS14]	O(m/n)	O(m+n)	O(n)	-	O(m/n)
[HK14] 1	O(m/n)	O(m+n)	O(n)	-	O(m/n)
[HK14] 2	O(m/n	O(m+n)	O(1)	-	O(m)

8.2.1 KEY DERIVATION

Firstly, we need to claim again that we give up the Forward Secrecy of the messaging system. According to Hahn's work, we need to derive two symmetric encryption keys to encrypt the files and build the indexes, so the Signal protocol is not suitable anymore that we intend to implement another encryption scheme to build this system. And basing on the existing messaging systems' encryption scheme, we want to use the following scheme to derive the keys of this system.

1. Alice gets Bob's public key g^b via the out-of-band channel, the key pair (b, g^b) $\in$ Curve25519.
2. Alice uses the his private key a and Bob's public key g^b do the ECDH key exchange to derive the secret. (a,g^a) $\in$ Curve25519, secret = $g^b \cdot a$.
3. Alice hashes the secret by the SHA512 hash function. Using the first 32 bytes of the hash result as the key k, the last 32 bytes as the MAC key k_m. $\mathbf{H}$(secret) = k + k_m.
4. Alice hashes the key k again, and uses the first 32 bytes as the k_1 and the remaining 32 bytes as the k_2. $\mathbf{H}(k) = k_1 + k_2$.

5. Alice uses the k_1 and k_2 for the searchable encryption, the k_m will be used for the authentication.

8.2.2 Definition of the keyword search

- $(\mathrm{K}, \gamma, \sigma) \longleftarrow \mathrm{Gen}(1^\lambda)$: is a secure key generation fuction. λ is security parameter, γ is an empty search index and σ is an empty search history.
- c $\longleftarrow$ Enc(K, f): is a algorithm that using key K to encrypt the file f to generate the ciphertext c.
- $(\sigma', \tau_\omega) \longleftarrow$ SearchToken(K, ω, σ): is a algorithm that inputs key k, keyword ω and search history σ to output new search history σ' and search Token τ_ω.
- $(\mathbf{I}_\omega, \gamma') \longleftarrow \mathrm{Search}(\tau_\omega, \gamma)$: is a algorithm that uses the search token τ_ω, a sequence of encrypted files $\mathbf{c}$ and search index γ to generate the a sequence of identifier I_ω and new search index γ'.
- $\alpha_f \longleftarrow$ AddToken(K, f, σ): is a algorithm that uses key K, file f and search history σ to generate the add token α_f.
- $(\mathbf{c}', \gamma') \longleftarrow$ Add(α_f, c, $\mathbf{c}$, γ): is a algorithm that uses the add token α_f, encryped file c, a sequence of encrypted files $\mathbf{c}$, and search index γ to generate the new sequence of the encrypted files and new search index.
- $f \longleftarrow$ Dec(K, c): is the decryption that uses key K to decrypt the ciphertext c to output the plaintext f.

8.2.3 HAHN'S WORK

This scheme maintains an index γ_f to the encrypted keyword. All file identifiers will be transferred to inverted index γ_ω for the keyword if any keyword is searched. The updates of the unsearched keywords need to update the inverted index, otherwise, the the following search will over two indices and cost more time. This section describes the scheme that the search history store at the client, so the client can check the keyword whether been searched and tells the server add the searched keyword to the inverted index, which is pure performance optimization.

- $(\mathrm{K},\gamma, \sigma) \longleftarrow \mathrm{Gen}(1^\lambda)$: is a algorithm that generate key K= (k_1, k_2), two chained hash table γ_f, γ_w and an empty set σ.

- c $\longleftarrow$ Enc(K, f): is the encryption function that uses the key k_2 to encrypt file f to generate the ciphertext c.

- $(\tau_\omega, \sigma') \longleftarrow$ SearchToken(K, ω, σ): is a algorithm that $F_{k_a}(\omega) = \tau_\omega$ and $\sigma' = \sigma \cup \{\tau_\omega\}$.

- $(\mathbf{I}_\omega, \gamma') \longleftarrow$ Search(τ_ω, γ): to check if there is an entry for τ_ω in the γ_ω

  – If yes, then set $\mathbf{I}_\omega = \gamma_\omega[\tau_\omega]$ and $\gamma'_\omega = \gamma_\omega$.

  – Otherwise, create an empty list $\mathbf{I}_\omega$ and do for every $\bar{c} \in \gamma_f$:

  1. For every $c_i \in \bar{c}$ that is i $\in$ [1, **len**(c)], set $c_i = l_i \parallel \gamma_i$ and if $\mathbf{H}_{\tau_\omega}(\gamma_\omega) = l_i$. If yes then insert ID$(f)$ that corresponds to $\bar{c}$ into $\mathbf{I}_\omega$.

  Creating entry $\gamma_\omega[\tau_\omega] = \mathbf{I}_\omega$ to update the γ'_ω.
  Output $\mathbf{I}_\omega$ and $(\gamma' = \gamma', \gamma_f)$.

- $\alpha_f \longleftarrow$ AddToken(K, f, σ): creates a list $\bar{f}$ consists the unique words. **len**$(\bar{f})$ equals the unique words of f. Creating an empty list $\mathbf{x}$ and generating a sequence pseudorandom values s, the length of s equals the unique words of f. For every unique words do following:

  1. Computing the search token $\tau_{\omega_i} = F_{k_1}(\omega_i)$.
  2. If the search token was searched before: if $\tau_{\omega_i} \in \sigma$, put the τ_{ω_i} to the list $\mathbf{x}$.
  3. Set $c_i = \mathbf{H}_{\tau_{\omega_i}}(s_i) \parallel s_i$.

  Output α_f = (ID(f), $\bar{c}$, $\mathbf{x}$).

- $(\mathbf{c}', \gamma') \longleftarrow$ Add $(\alpha_f$, c, $\mathbf{c}, \gamma)$: output updated sequences of ciphertexts **c'** and the search index γ'

- $f \longleftarrow$ Dec(K, c): uses key K decrypts the ciphertext c to get the plaintext f.

9 CONCLUSION

In this report, We bridge the gap between the messaging system with blockchain scheme and implement reliable encryption schemes to provide desirable properties that the messaging systems must achieve. We implement the Prekeys scheme to ensure the asynchronicity, 3 DH key exchange protocol for the key agreement, and Axolotl protocol for the key derivation. We also implement the blockchain scheme to provide a P2P network which avoids the attacks from the server to provide user more privacy and security. And we also propose the decentralized messaging system combines with the searchable encryption, to reduce the storage of the blockchain size and improve the security of the messaging system. This searchable encryption scheme also ensures the efficient update and security.
The other improvement we can do in the next phase is that combining the Forward Secrecy of the messaging systems with the searchable encryption without reducing the security and efficiency.

References

[1] N. Unger, S. Dechand, J. Bonneau, S. Fahl, H. Perl, I. Goldberg, and M. Smith, "SoK: Secure Messaging," in Symposium on Security and Privacy. IEEE, 2015.

[2] NIkita Borisov, Ian Goldberg, Eric Brewer. ”Off-the-Record Communication, or, Why Not To Use PGP.”

[3] Matthew D.Green and Ian Miers. ”Forward Secure Asynchronous Messaging from Puncturable Encryption.”

[4] Sebastian Schrittwieser, Peter Kieseberg, Manuel Leithner, Martin Mulazzani, Markus Huber, Edgar Weippl. ”Guess Whos Texting You? Evaluating the Security of Smartphone Messaging Applications.”

[5] Williams Stallings, ”Cryptography and Network Security-Principles and Practices.”

[6] Confide. Confide-Your Off-the-Record Messenger. [Online]. Available: https://getconfide.com/

[7] Electronic Frontier Foundation. Secure Messaging Scorecard. [Online]. available: https://www.eff.org/secure-messaging-scorecard

[8] R.Stedman, K.Yoshida, and I.Goldberg, ”A User Study of the Off-the-Record Messaging,” in symposium on Usable Privacy and Security. ACM, 2008, pp. 95-104

[9] M.S.Melara, A.Blankstein, J.Bonneau, M.J.Freedman, and E.W.Felten, ”CONIKS:A Privacy-Preserving Consistent Key Service for Secure End-to-End Communication,” Cryptology ePrint Archive Report 2014/1004, 2014. [Online]. Available: https://eprint.iacr.org/2014/1004.

[10] T.Frosch, C.Mainka, C.Bader, F.Bergsma, J.Schwenk, and T.Holz, ”How Secure is TextSecure?” Cryptology ePrint Archive Report 2014/904, 2014 [Online]. Available: https://eprint.iacr.org/2014/904.

[11] O.W.Systems. Open Whisper Systems partners with WhatsApp to provide end-to-end encryption. [Online]. Available: https://whispersystems.org/blog/whatsapp/

[12] D.J.Bernstein. ”Curve25519: new Diffie-Hellman speed records,” in PKC, 2006. [Online]. Available: http://crypto/ecdh/curve25519-20060209.pdf

[13] S.Nakamoto. ”Bitcoin: A peer-to-peer electronic cash system,” (2009). [Online]. available: https://bitcoin.org/bitcoin.pdf.

[14] S. Wilkinson, J. Lowry. ”Metadisk: Blockchain-based decentralized file storage application,” (2014). [Online]. available: http://metadisk.org/metadisk.pdf.

[15] Jonathan Warren Bitmessage: ”A PeertoPeer Message Authentication and Delivery System.”

[16] S. Wilkinson, ”Storj A Peer-to-Peer Cloud Storage Network,” [Online]. available: http://storj.io/

[17] Rynomster, Tecnovert, "ShadowChat Secure Messaging: A P2P Encrypted Instant Messaging system," Online. available: https://github.com/shadowproject/whitepapers/blob/master/shadowcash-em.pdf

[18] Rynomster, Tecnovert, "Shadow: Zero-knowladge Anonymous Distributed Electronic Cash via Traceable Ring Signature," [Online]. available: https://github.com/shadowproject/whitepapers/blob/master/shadowcash-anon.pdf

[19] "Threema Cryptography Whitepaper," [Online]. available: https://threema.ch/press-files/cryptographywhitepaper.pdf

[20] "Ethereum: A next-generation smart contract and decentralized application platform," (2014). https: //github.com/ethereum/wiki/wiki/%5BEnglish%5D-White-Paper.

[21] M. Marlinspike, 2014. [Online]. Available: https:// openwhispersystems.org/

[22] D. Song, D. Wagner, and A. Perrig. "Practical techniques for searching on encrypted data," In IEEE Symposium on Research in Security and Privacy, pages 4455. IEEE Computer Society, 2000.

[23] R. Curtmola, J. Garay, S. Kamara. R. Ostrovsky. "Searchable Symmetric Encryption: Improved Definitions and E cient Constructions"

[24] S. Kamara, C. Papamanthou, and T. Roeder. "Dynamic searchable symmetric encryption." In Proceedings of the 19th ACM Conference on Computer and Communications Security, CCS, 2012.

[25] F. Hahn, F. Kerschbaum. "Searchable Encryption with Secure and Efficient Updates"

[26] "DARPA Is Looking For The Perfect Encryption App, and Its Willing to Pay." [Online]. available: http://motherboard.vice.com/read/darpa-decentralized-blockchain-encryption-messaging-app

[27] A. Kosba, A. Miller, E. Shi, Z. Wen, C. Papamanthou. "Hawk: The Blockchain Model of Cryptography and Privacy-Preserving Smart Contracts"

[28] R. A. Popa, E. Stark, J. Helfer, S. Valdez, N. Zeldovich, F. Kaashoek, and H. Balakrishnan. "Building web applications on top of encrypted data using mylar." In Proceedings of the 11th USENIX Symposium of Networked Systems Design and Implementation, NSDI, 2014.