参赛队员姓名：Jiayi Liu

中学：Archbishop Mitty High School

省份：California

国家/地区：USA

指导教师姓名：WenWu Wang

论文题目：A Privacy Secure Login System with a New Hierarchical Deterministic Key Pair Generation Method

# A Privacy Secure Login System with a New Hierarchical Deterministic Key Pair Generation Method

Jiayi Liu

## Abstract:

In an always-online world, login systems are presented to users by almost all information storage systems that need authentication and authorization. However, the current login systems are not free from flaws: a normal registration process can take quite some time; using single user-id across platforms (i.e. Single Sign-on) often raises privacy concerns; and even the current login process of entering username and password is still not convenient and secure, with the chance of user forgetting login information and login information being stolen. To address these issues, a new login system based on asymmetric cryptography technology, with the advantage of being fast, easy to access, and highly secure, is proposed. To make the system more user-friendly, a new deterministic key pair generation method is proposed with which, instead of providing users the mnemonics in the form of random words, users can provide their own seed phrases. This allows users to safely and easily retrieve lost login information.

## Keyword:

Login System, Hierarchical Deterministic Key Pair, Bitcoin, Blockchain, Mast Seed, Mnemonic, Key Pair Derivation Path.

# Table of Contents

# 1. Existing Login Systems and Problems

In contemporary society, people use various information systems every day and in many, if not most cases, accessing personal information in these information systems requires the user to go through a login system. This is because the information systems need to know a user's identity before that user can access his or her personal information.

This procedure is called authentication. Authentication as a process of determining whether someone is who it is declaring to be. It is a key component of any online system which handles sensitive data or transactions. As internet technology has evolved, a diverse set of network authentication methods have been developed, including both general authentication techniques (passwords, two-factor authentication [2FA], tokens, biometrics, transaction authentication, computer recognition, CAPTCHAs, and single sign-on [SSO]) and specific authentication protocols (Kerberos and SSL/TLS). [1]

Therefore, end-users have to adapt to these different authentication methods. Worse, people usually have different credentials for different systems, as shown in figure one:

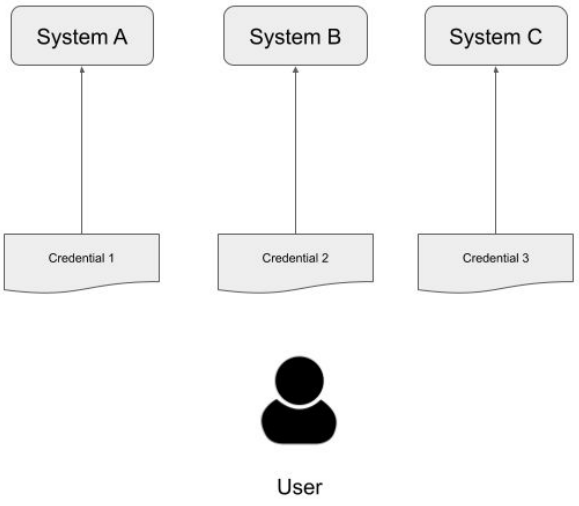


*Figure 1: Different credentials for different systems*

The increasing prominence of information technology and the internet in fields such as employment, finance, recreation, transaction, and many more have led people to accumulate a proliferation of accounts and passwords. This phenomenon is called password fatigue - a feeling experienced by many people who are required to remember an excessive number of passwords as part of their daily routine.

Then a technology called Single Sign-On (SSO) was introduced and it seems to be able to solve the problem of password fatigue because single sign-on manages user's login credentials in a centralized system and user can log into multiple related yet independent software systems with the access being granted by the centralized server.

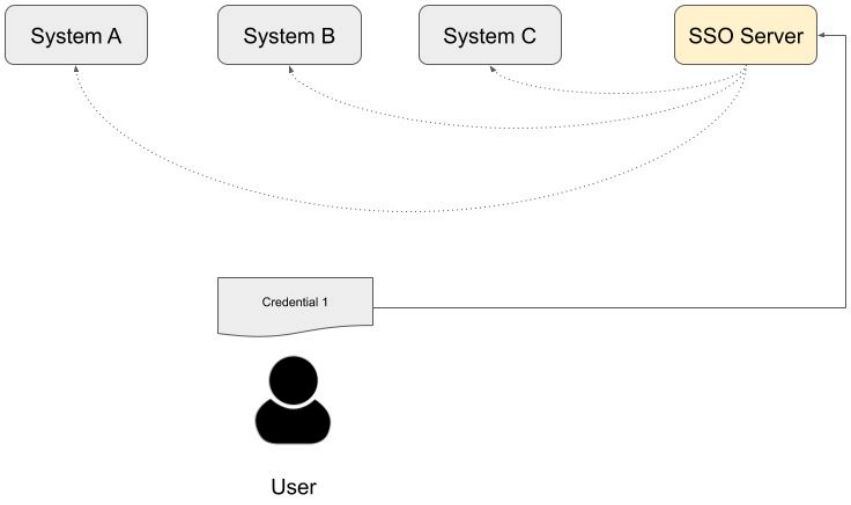



*Figure 2: Single Sign-On System*

However, since single sign-on provides access to many resources once the user is initially authenticated ("keys to the castle"), it increases the negative impact in case the credentials are leaked or misused. Therefore, single sign-on requires an increased focus on the protection of the user credentials, and should ideally be combined with strong authentication methods like smart cards and one-time password

tokens.[2]

Yet, regardless of a user uses different credentials for different systems, or uses the single sign-on method, these authentication methods have one thing in common:  the system that the users try to log in has our credentials stored in their servers - a password, biometric characteristics, or certificate. Some two-factor authentication methods will also have user's emails or phone numbers for passcode delivery, this on one side improves security, but on the other side sacrifice the user convenience as well users' personal information privacy.

Therefore, one significant downside of these login systems is that storing user credentials in servers poses severe security concerns. For example, "the Internet service company Yahoo! reported two major data breaches of user account data to hackers during the second half of 2016. The first announced breach, reported in September 2016, had occurred sometime in late 2014 and affected over 500 million Yahoo! user accounts"[3].  "Yahoo! later affirmed in October 2017 that all 3 billion of its user accounts were impacted in a separate data breach, occurring earlier around August 2013".[4] Both breaches are considered the largest discovered in the history of the Internet.

People need a new login system that is more secure, convenient, and privacy-protective.

# 2. Blockchain Transaction Verification and Account Management

Blockchain technology has developed rapidly for the past ten years since the unidentified person Satoshi Nakamoto invented Bitcoin in 2008. Bitcoin is a cryptocurrency that can be owned by different accounts with all transactions kept in a public blockchain. Blockchain enables people to anonymously perform secure and trustworthy transactions that are traceable and verifiable. Its anonymity ensures that a person will be linked to a public Bitcoin address, but no one will recognize the actual

name or address of that person. Blockchain's verifiability guarantees anybody can check all of the transactions and hashes all the way back to the genesis block. This type of transaction verification is based on asymmetric cryptography, also known as public cryptography.

Public cryptography: when owner 1 sends a BTC transfer transaction to owner 2, owner 1 has to sign the transaction with owner 1's private key. In the new transaction, it has the reference to the previous transaction in which owner 1' public key is included. Once the transaction is published, anyone can trace back to the previous transaction, get owner 1's public key, and verify this transaction is signed by owner 1's private key. Thus the transaction is verified and owner 2 will be the new owner. The diagram from the original Bitcoin white paper illustrated this verification process:

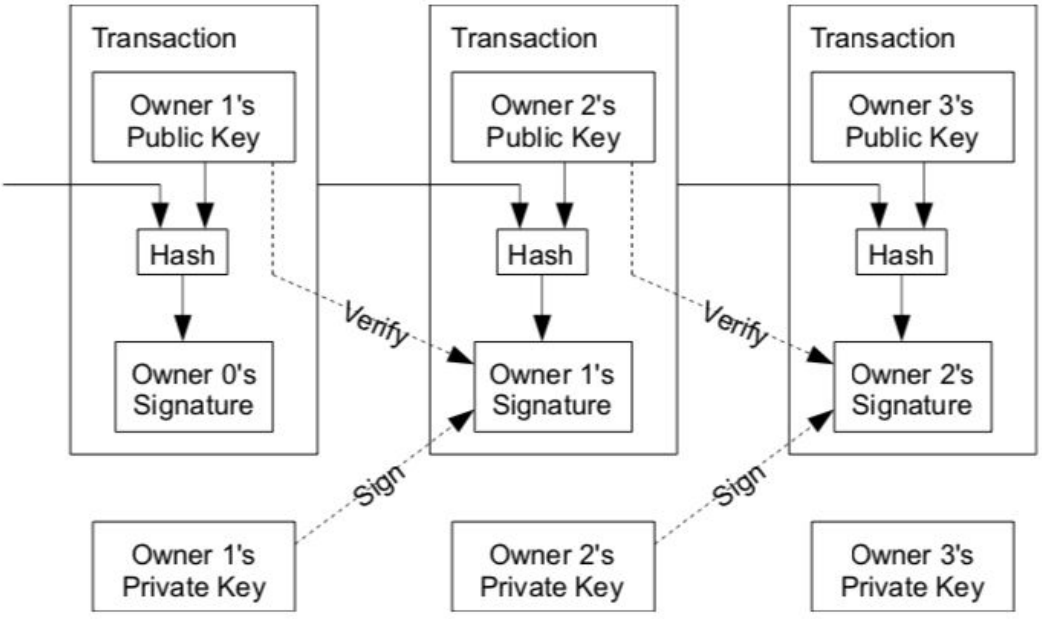



*Figure 3: Bitcoin transaction verification* [5]

As shown in the diagram above, Blockchain has the "owner" concept, yet other users never know who this user actually is. All they know is that this user has a certain amount of bitcoins via verification. This method is traceable, indisputable, and

incorruptible.

In a blockchain, a user's private key is managed by the user's Cryptocurrency Wallet. A cryptocurrency wallet is a device, physical medium, program, or service which stores the public and private keys and can be used to track ownership, receive or spend cryptocurrencies [6]. A cryptocurrency wallet contains one or multiple pairs of public and private cryptographic keys. Public keys are used as the user's identity in blockchain, and the private keys are used to sign blockchain transactions.

"BIP32 - Hierarchical Deterministic Wallet" is a Bitcoin specification that describes a general structure of a hierarchical deterministic wallet (HD wallet). In particular, it defines how to derive private and public keys of a wallet from a binary master seed (m) and an ordered set of indices (so-called BIP32 path): [7].

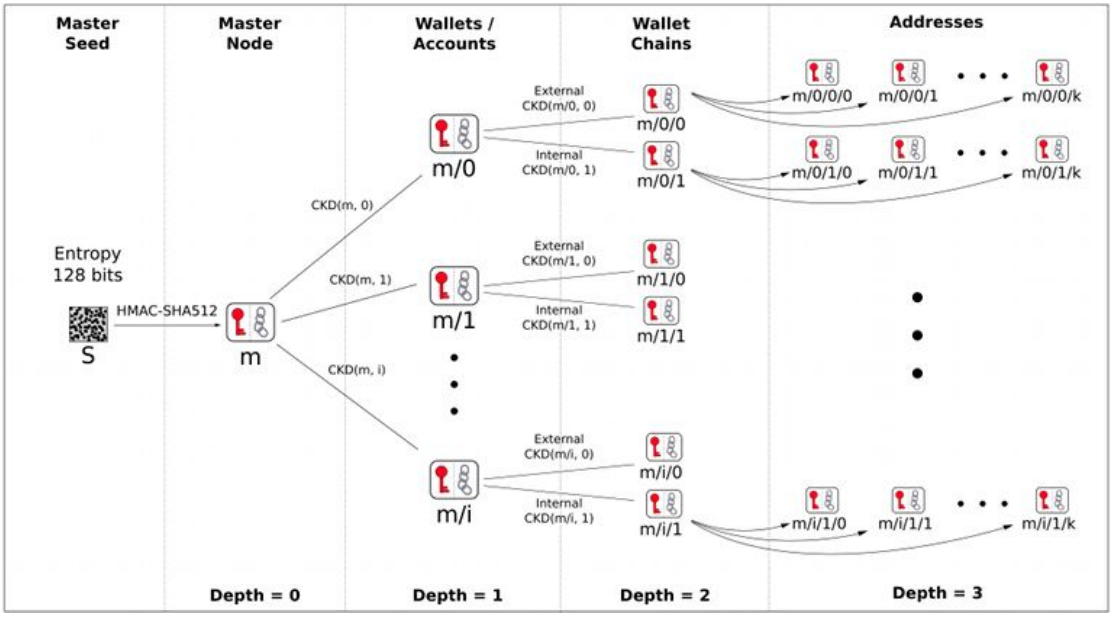



*Figure 4: Hierarchical Deterministic Wallet [7]*

Following BIP32, a blockchain user can have a cryptocurrency wallet with multiple wallet accounts (multiple public/private key pairs), derived from different BIP32 paths. "BIP44 - Multi-Account Hierarchy for Deterministic Wallets" further defines the following 5 levels in BIP32 path[8]:

*m / purpose' / coin_type' / account' / change / address_index*

where “Purpose” is a constant set to 44'; “Coin type” is a constant set for each crypto coin; “Account” are numbered from index 0 in sequentially increasing manner; “Change” is either constant 0, used for external chain, or constant 1 for internal chain; and “Address_index” are numbered from index 0 in sequentially increasing manner.

For example, path “*m / 44' / 0' / 0' / 0 / 0*” is for Bitcoin, first account, external chain, and first address.

However, the wallet is only functional if there is a master seed that can be used to generate all these different key pairs. The method that derives the master seed is called BIP39 - Mnemonic code for generating deterministic keys.

This BIP describes the implementation of a recovery seed and its relation to BIP32 binary master seed. It provides an implementation of a mnemonic code or mnemonic sentence -- a group of easy to remember words -- for the generation of deterministic wallets. It consists of two parts:

1) generating the recovery seed: from computer-generated randomness to a mnemonic sentence.

2) Converting mnemonic sentence to a BIP32 wallet master seed: this conversion includes an optional application of a passphrase during the conversion.

With BIP39, a user can then use a cryptocurrency wallet to generate a master seed from one device, and use a mnemonic sentence (for example, a 12-word sentence “aspect fortune into know outer pass drastic couple hello rebel domain box”) to recover the wallet data with the 12 words on a different device.

BIP32, BIP39, and BIP44 work together define a complete user account system that works perfectly with Bitcoin and other blockchains. Summarization of the functions of these specifications are listed below:

- BIP32: allow users to have multiple key pairs for different blockchains
- BIP39: defines how to generate the master seed of the key pairs, and how to easily remember and recover by human beings.
- BIP44: defines the derive paths for different blockchains.

The Bitcoin account system, along with the Bitcoin protocol, make blockchain an anonymous yet verifiable systems. Anonymous means privacy protected, and verifiable means secure. That is the goal to design a new login system.

# 3. Hierarchical Deterministic Key Pair Based Login System

## 3.1 Overview

Inspired by the blockchain account management practice, we define a new login system with hierarchical deterministic key pairs. This login system will have the following high-level architecture:

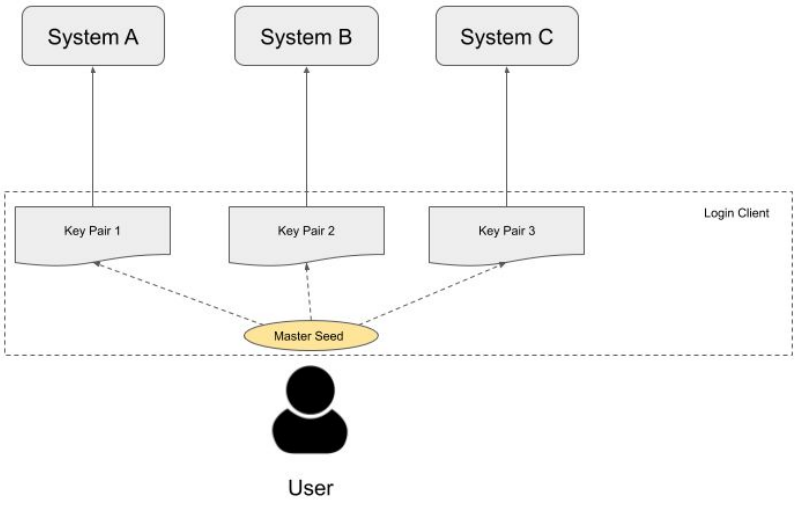



*Figure 5: hierarchical deterministic key pair based login system*

On the user's end, a login client, a software program or device, manages all user's login credentials. A user has a master seed and the login client which will be used to derive multiple key pairs that can be used to log into different systems. The

keys are managed in a similar way with the cryptocurrency wallet for blockchain: hierarchical, and deterministic.

The login client also triggers the login process when requested by the user. The login procedure for hierarchical deterministic key pair based login system is as follows:

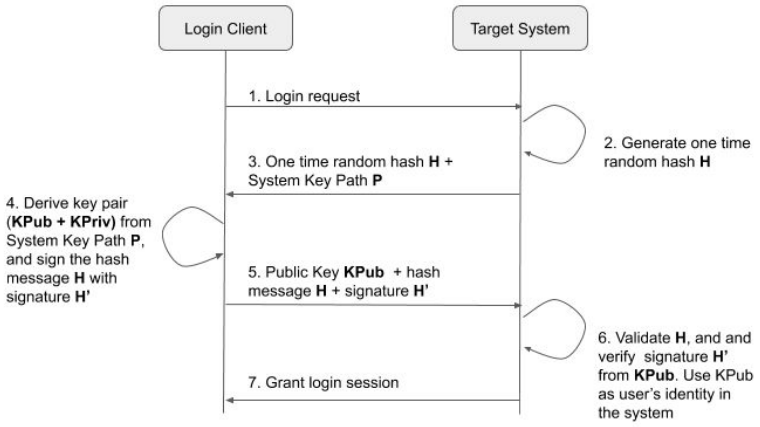



*Figure 6: Login Procedure*

(1) On behalf of the user, the login client requests to access the target system.

(2) The target system generates a one-time random hash **H**

(3) The target system sends the one-time random hash **H** back to the client, along with the Key Drive Path **P** of the system.

(4) The login client derives the key pair for the target server from key derivation path **P**. The derived key pair has public key **KPub** and private key **KPriv**. It signs the server hash **H** with signature **H'**, using the private key **KPriv**.

(5) The login client sends its public key **KPub**, the one-time hash **H**, and the signature **H'** back to the server.

(6) The target server validates **H** and verifies the signature **H'** with the given public key **KPub**. After all of the information is verified, the user is logged in with identity **KPub**.

(7) The login session is granted to the client.

Overall, this login system has the following characteristics:

- **Secure**:

  Users own the credentials (master seed and all private keys). There is no breach possible from the server side for user login information

- **Privacy protecting**:

  Users can log in with different identities (different public keys) to different systems so servers cannot track the users' behaviors across systems. The two-factor authentication process also makes emails/phone numbers unnecessary.

- **Fast**:

  No user name/password typing and no two-factor authentication. If permission is granted, the login client can handle all the login procedures for the user.

- **Transparent and open**:

  All the procedures involved during login are transparent to all parties involved. So as long as they follow the same procedures, login can be executed between any login clients and target systems.

  More details of login procedures are illustrated in the subsequent sections.

## 3.2 Mnemonic and Master Seed Generation

The master seed is the nucleus of the user's entire login credentials in the

hierarchical deterministic key pair login system. It is used to derive every individual key pair (account) that a user will use to login to different online systems. It is stored only on the user's own devices and should be stored as encrypted and securely protected. While the implementation of the login client and how it protects the master seed and key pair might vary, how users memorize and recover the master seed needs to be standardized.

BIP39 is the answer in the blockchain domain for how to recover a cryptocurrency wallet seed with a 12-word mnemonics. For a 128-bit long entropy with 4-bit checksum, the master seed is totally 132-bit long and it is divided into 12 11-bit long segments. (132/11 = 12).

Each 11-bit segment of the binary data has 2**11 = 2048 possible permutations and each permutation is mapped to a unique word. The word list is smartly selected, similar words avoided, and sorted. [9]

With the mnemonic as the 'password', and mnemonic + a possible passphrase as the 'salt', it is used to derive a longer (512-bit) seed through the use of the key-stretching function PBKDF2. The iteration count is set to 2048 and HMAC-SHA512 is used as the pseudo-random function. The seed produced is then used to build a deterministic wallet and derive its keys.

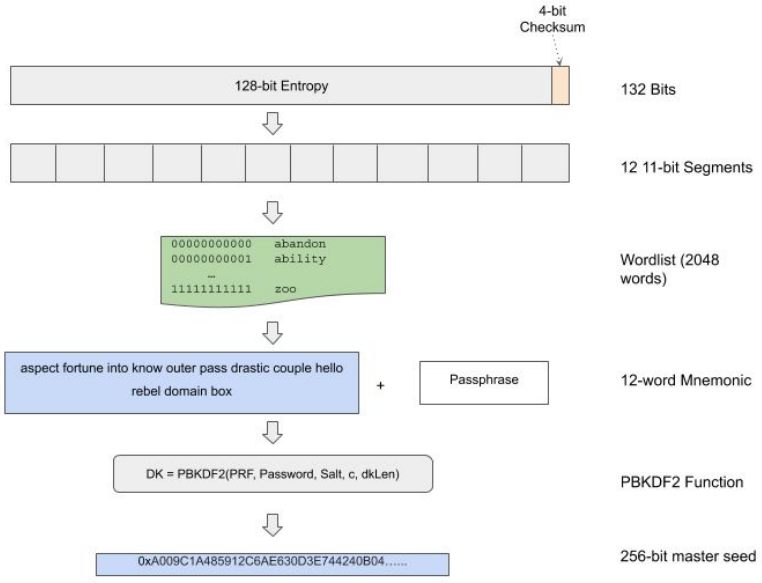


*Figure 6: BIP39 Mnemonic and master seed generation*

As defined by BIP39, an entropy is generated first, and then it is mapped to a 12-word mnemonic. The 12 words in the mnemonic are unrelated and the mnemonic sentence is not a human memorable sentence but a collection of unrelated words. This on one side improves the security, but on the other side expose complexity for most people to memorize it. So it is suggested that the mnemonic should be written down and kept safe.

However, there is a way that people can have a mnemonic that is easy to remember and guarantees uniqueness to different people: using personal experiences and opinions, and biometric characteristics.

A mnemonic generated from user's personal experiences and opinions can work as depicted in the following diagram:

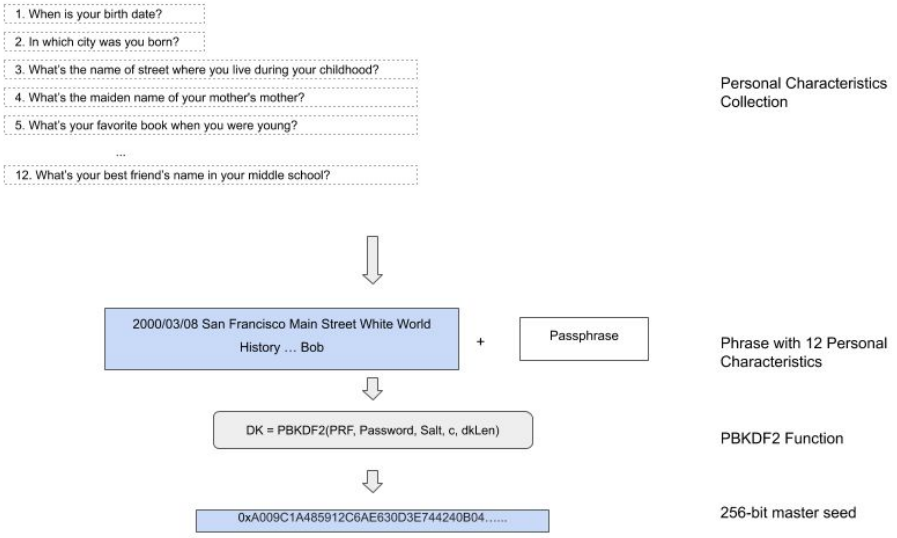



*Figure 7: Mnemonic generation based on answers to personal questions*

Each user is asked 12 questions and the user's personalized answers are collected. We use the answers as the input to the same PBKDF2 function as in BIP39 to generate the master seed.

There are two issues for the above mnemonic generation method:

1. The 12 questions can be guessed by people who know a user very well.
2. The uniqueness of the mnemonic is not guaranteed between different users

Therefore, the following enhancements are proposed for the above-mentioned method:

1. Require a passphrase when generating the master seed.

   Unlike the optional passphrase in BIP39, a passphrase from a user is mandatory in this process. This does not only protect the master seed to be tampered by acquaintances but it also greatly avoids the duplication of the generated master seed.

2. Involve biometric features in personal characteristics.

   Biometrics is the use of a person's unique physiological, behavioral, and morphological characteristics to provide positive personal identification. Because of its features of uniqueness, universality, performance, measurability, and user-friendliness, it is widely used in authentication systems.

   With the popularization and increasing computer power of smart devices such as smartphones and personal computers, user's biometric characteristics can be collected and translated into unique data and be put into the mnemonics.

   The following table summarizes some common biometric features used for authentication[10]:

| **Biometric** | **Trait** |
|---|---|
| Fingerprint | Finger lines, pore structure |
| Signature (dynamic) | Writing with pressure and speed differentials |

| | |
|---|---|
| Facial geometry Distance of specific facial features (eyes, nose, mouth) | Iris Iris pattern |
| Retina | Eye background (pattern of the vein structure) |
| Hand geometry | Measurements of fingers and palm |
| Finger geometry | Finger measurement |
| Vein structure of the back of the hand | Vein structure of the back of the hand |
| Ear form | Dimensions of the visible ear |
| Voice | Tone or timbre |
| DNA | DNA code as the carrier of human hereditary features |

*Table 1: List of Biometric Features* [10]

Assuming 10 personal questions and two biometric characteristics (for example, facial geometry [the distance between two eyes] and voice pitch [the average pitch value]) are asked and collected for the mnemonic, the enhanced Mnemonic and Master Seed Generation method is shown in the following diagram.

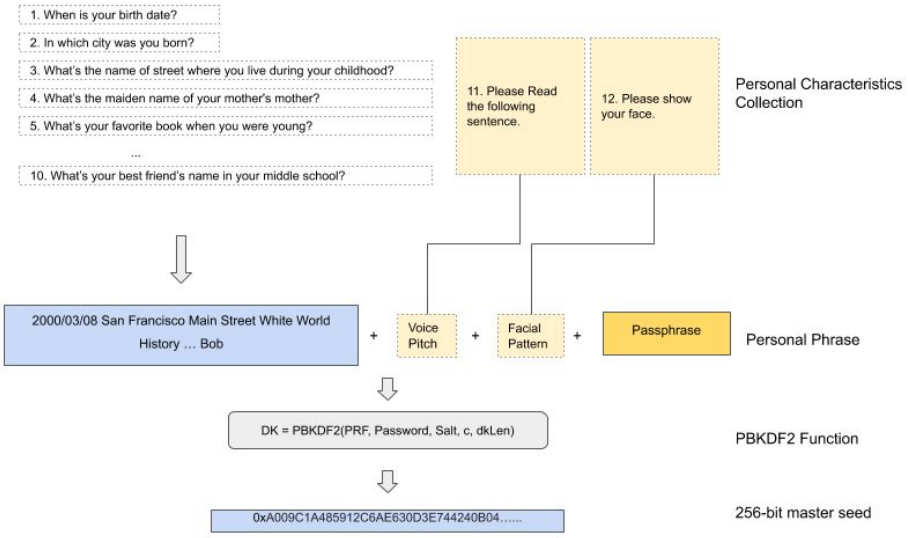



*Figure 8: Mnemonic generation based on question answers and biometric characteristics*

## 3.3 System Key Derivation Path

In the Hierarchical deterministic key pair based login system, a user logs into different systems with different key pairs. For one system, a user will always login with a single key pair. This requires the key derivation path for a given system to be unique and permanent.

A target system, in most formal cases, has a fully qualified domain name (FQDN). (for example, www.example.com and mail.example.com.) The FQDN for a given server is stable and will be used as the key derivation path in the Reverse domain name notation with "/" as delimiters between domain components. The key derivation paths for some example FQDN are listed below:

| Doman | Key Derivation Path |
|---|---|
| www.example.com | com/example/www |
| mail.example.com | com/example/mail |
| example.org | org/example |

The target server will present it's key derivation path to the login client so the login client can derive the right key pairs for the target server and help the user to login.

According to the definition of BIP32, the derive path is in the following format:

*m / purpose' / coin_type' / account' / change / address_index*

Other than the leading segment, each subsequent segment in the derivation path should be a designating the index of each segment.

The following product sum algorithm, following the Java hashCode implementation[11], is used to map the reverse domain name notation to an index based derive path:

$$index = s[0]*31^{(n-1)} + s[1]*31^{(n-2)} + ... + s[n-1]$$

where 's' is the string of the domain name segment, s[i] is the *i*th character of the string, n is the length of the string, and ^ indicates exponentiation. "The coefficient of 31 is adopted for two reasons: First, 31 is a prime number. According to the characteristics of a prime number, the result of multiplication with a prime number is much more likely to be unique than with other methods, namely, the repeatability of the obtained hash value is relatively small. Second, this specific prime number was chosen because when computing a hash address. we want to minimize the amount of data with the same hash address, which is the so-called "conflict". If a large amount of data has the same hash address, the hash chain Of these data will be too long, thereby reducing the query efficiency. Consider the above two points synthetically, under the premises of no overflow and the coefficient is a prime number. With the higher coefficient, the so-called "conflict" will decrease and the search efficiency will increase." [12]

Applying the above algorithm, the derive path for the reversed domain names are:

| Doman | Key Derivation Path | Actual Derive Path |
|---|---|---|
| www.example.com | com/example/www | m/98689/1322970774/118167 |
| mail.example.com | com/example/mail | m/98689/1322970774/3343799 |
| example.org | org/example | m/110308/1322970774 |

Since the Derive Path **P**—ie. the FQDN—is given by the server, as shown in step 3 of the login procedure in figure 6, a vulnerability arises: problems may occur if a server returns a fake derive path. This might happen in some fishing attacks—a fraudulent attempt by a malicious site to obtain users' sensitive information. In this case, the public key to another server disguises itself as the real target system that the user tries to log in.

The solution to this issue is the server certificate, which is commonly used in TLS based communication. To prove the server really 'owns' the given derive path P (ie. owns the given FQDN), the server needs to present a server certificate, issued by trusted Certificate Authority (CA).

If the whole communication between the client and the server is already under the TLS encrypted channel, the login client can obtain the server's certificate from the TLS handshake procedure. However, the client can issue a separate challenge procedure to verify the server is the owner of the derivation path of **P**. An illustration is presented below:

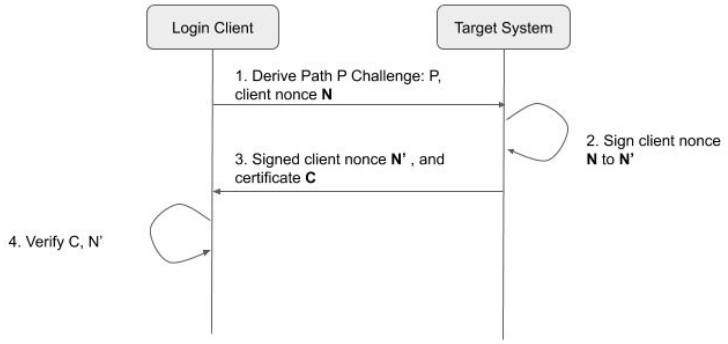



*Figure 9: Derive Path Challenging Procedure*

(1) The client generates a client nonce **N** (one-time random hash) and sends it to the server, to challenge the derive path **P**

(2) The server signs the message **N** with the private key of the certificate **C** corresponding to derive path **P**

(3) The server sends the signed message **N'** and the certificate **C** back to the client

(4) The client verifies the certificate **C**, and also uses the certificate to verify the signature of **N'**.

# 3.3 Replay Attack and One-time Random Hash

The hierarchical deterministic key pair based login needs a message from the server and the client has to return the signed message. This can easily cause a replay attack if the message and the signature are used again by hackers. This is why the one-time random hash is used.

In step 2 of the hierarchical deterministic key pair based login procedure, a one-time random hash is generated by the target server. This hash will be returned to the login client as the message that needs to be signed by the user's private key. In

step 6 of the login procedure, the target server needs to verify that the user was signing a valid hash issued by the server and it is used only once. This one-time hash is commonly called nonce in many cryptographic protocols.

Additionally, the one-time random hash should also include time-based information to prevent the system from a delayed message attack.

# 4. Possible Issues and Resolutions

## 4.1 Man-In-The-Middle Attack

Since the login and authentication procedure happens at the beginning of the client/server communication session, if not used with other protocols, such as TLS/SSL, it is very easy for the whole system to subject to man-in-the-middle attack—where the attacker secretly relays and possibly alters the communications between two parties who believe they are directly communicating with each other.

This problem can be solved by utilizing the already-existed key pair between the server and the client. Since the client has its own key pairs, after the initial login procedure, the two parties can continue to communicate utilizing the client's key. This is achieved by asking the client to sign each message it sends to the server, and the server verifying the client's signature with the client's public key.

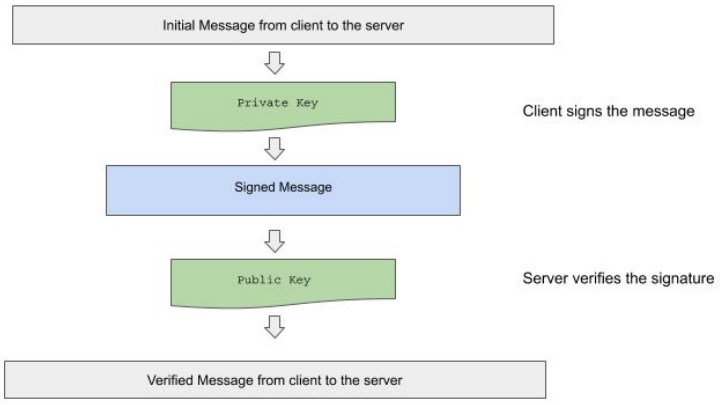



*Figure 10: Sign and verify messages for all client requests*

## 4.2 Fake and Spam Account

Fake and spam accounts invasion is a big problem faced by many service providers. Facebook announced in May 2019 that it removed 2.19 billion fake accounts between January and March of that year. [13].

With the introduction of a hierarchical deterministic key pair based login method, one can argue that the easier login method could lead to more fake accounts because the login client can act as a spam bot and interact with the server automatically.

The login method itself is never a key factor that can hinder fake account creation—software can control the browser and type in username/password easily. The server has to implement some fake account prevention methods, for example, two-factor verification during account creation, manual procedures during user login, or fake account pattern recognition with artificial intelligence on user's behavior analysis. All these approaches can be applied independently on the login system.

## 4.3 Master Seed Change

There are two aspects to changing a master seed:

1. Changing master seed on the client-side:

A user might want to change the master seed, due to his or her original mnemonic information being hacked or other reasons.

The easiest way to change the master seed is to generate a new master seed with a new passphrase, with all the questions and biometric characteristics remain the same.

To improve security, the sequence of the answered questions or collected biometric features can be shuffled with different passphrases.

If all of the 12 questions/biometric feature collections steps are numbered, then the following procedure can be applied as the Master Seed generation procedure:

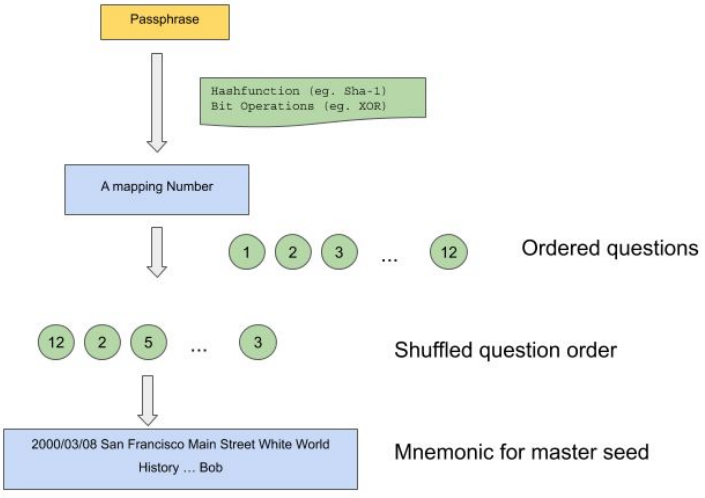



*Figure 11: Shuffle Mnemonic Question Order*

2. Swapping accounts on the server side

The target servers also need a procedure for the client to replace its account with new keys generated by the new master seed. The key to this procedure is to

correlate the two public keys to the same account. The following diagram illustrates one of the many ways to do this - the server issue an account swap token to correlate two accounts.

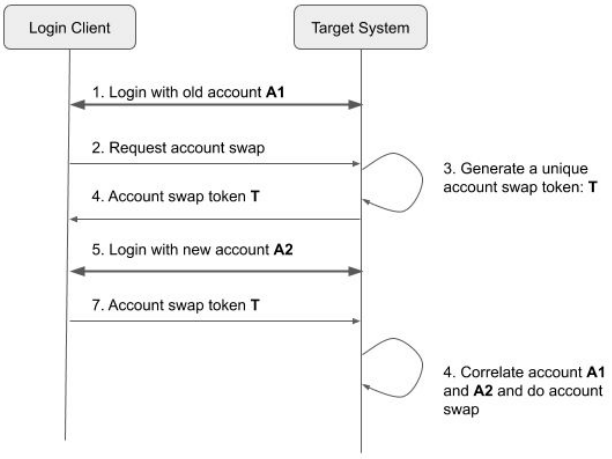



*Figure 12: Server-Side Account Swap*

(1) User logs into the server with its old account **A1**

(2) User request account swap to the server.

(3) The server generates a unique account swap token **T** for account A1

(4) Account swap token **T** is returned to the client.

(5) User logs into the server with it's new account **A2** (new key generated by new master seed)

(6) The client sends the account swap token **T** to the server.

(7) With the account swap token T, the server correlate A1 and A2, and associate A1's information to A2. Account swap completes.

## 4.4 Login Client on a Different Device

There is another common use case that needs to be covered: a user wishes to log into a system on a new device, such as a public computer, but the user's login client (the software that manages all the keys) is installed on a different device, for example on user's smartphone.

Two key procedures are involved in login with a different device:

1. Transfer login information from the other device to login client

The login information mentioned here includes the one-time random hash **H** returned from the server and the key derivation path **P** of the server. The login client needs the information to generate the login response.

There are many ways to transfer information between devices, but one intuitive way is using QR codes. the server can present the login information in text, as well as in a QR code picture. The user can use his or her login client on the smartphone to scan the QR code to extract the required information.

2. Send login response from login client to the target server

The login response is the signed message **H'** and the user's public key. Since the login client is not in the same login session where the user is doing the original login, it doesn't know where the login response shall be sent.

An enhancement will be made to the original login procedure - when a login is requested, along with the one-time random hash **H** and the key derivation path **P**, the server will respond with a server address **L** where login response shall be pushed back.

This solution is illustrated in the following diagram.

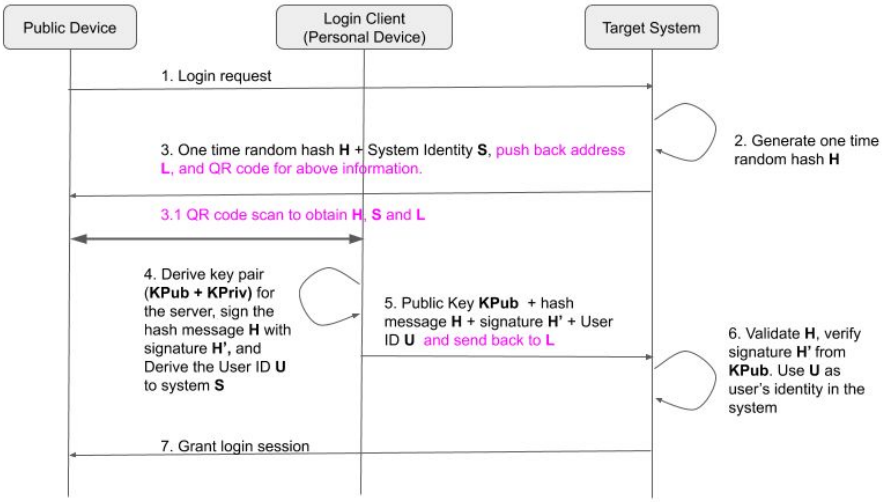



*Figure 13: Login from a different device*

Differences with the normal login procedure in figure 6 are highlighted. In step 3, a push back address **L** is given to the client side which will be used by the login client to send the login response back to the address, as shown in step 5. A new step 3.1 is added to depict how login information is transferred, from the original device where the user initiated the login, to the device where the login client resides.

# 5. Sample System

To prove the feasibility of the deterministic key pair based login system, a sample system is developed with the following components:

1. Login Client - Master Seed Generation

```
$ node 01-masterGenerate.js
? 1/12. When is your birth date? (yyyymmdd format) 20030828
? 2/12. In which city was you born? Beijing
? 3/12. What's the name of the street you lived in during your childhood? Yuquan Road
? 4/12. What's the maiden name of your mother's mother? Wang
? 5/12. What's your favorite book? pride and prejudice
? 6/12. What's your favorite color? black
? 7/12. What's your favorite food made by your families? jiaozi
? 8/12. What song do you like best when you are in elmentary school? Can You Feel the Love Tonight
? 9/12. Which sports team are you in favor of most? Lakers
? 10/12. Which person do you admire most in history? Washington
```

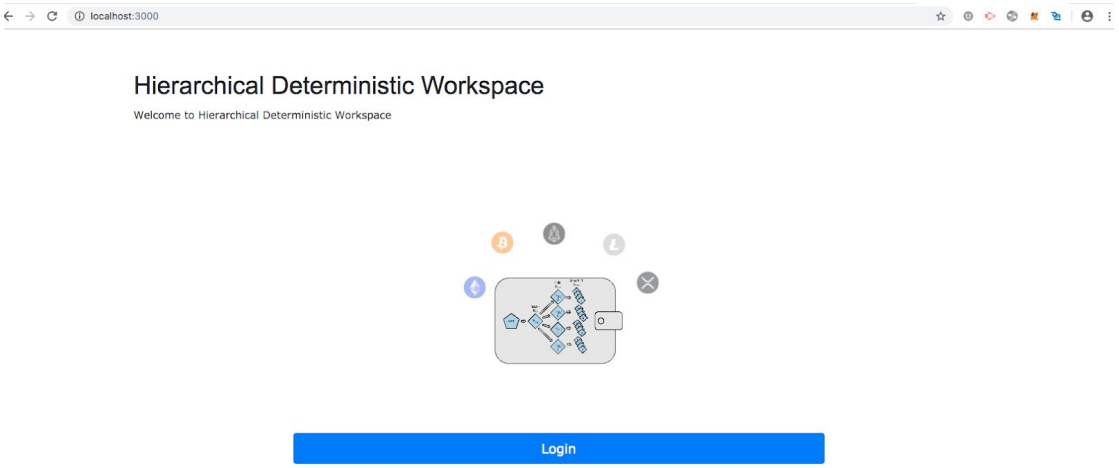
? **11/12. Please take a video and upload. (Input file path)** 2.8185871927930677
? **12/12. Please read the following text and upload the audio recorded: Secure, Private, Fast, Transparent! (Input audio path)** 2657.311671363327


```
{
 "birthday": "20030828",
 "city": "Beijing",
 "street": "Yuquan Road",
 "maiden": "Wang",
 "book": "pride and prejudice",
 "color": "black",
 "food": "jiaozi",
 "song": "Can You Feel the Love Tonight",
 "sport": "Lakers",
 "people": "Washington",
 "facial": 2.8185871927930677,
 "voice": 2657.311671363327
}
```

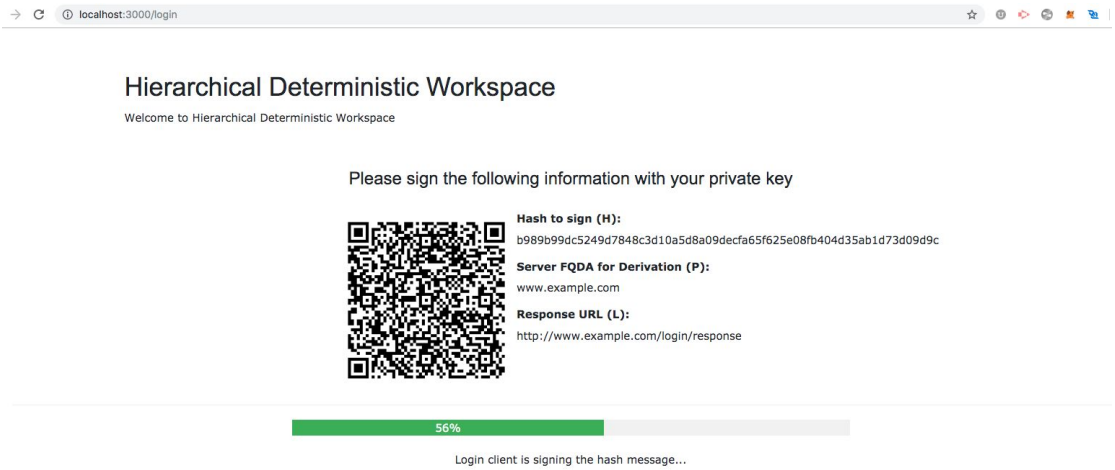

? **Please check the input. Are your answers correct?** Yes
Your nemonic is: 20030828 Beijing Yuquan Road Wang pride and prejudice black jiaozi Can You Feel the Love Tonight Lakers Washington 2.8185871927930677 2657.311671363327
Your master seed is:
3134e79456004aeb087b8f84daa1bf26c9beb4b30fd79bacf57bc39d5773e823b5cfd9c52d765ea3ccff89e83db1b937fdef2ee3aead8f7a3a085a951a452719
Master seed is saved securely!

## 2. Web Server and Login Client - Login Procedure

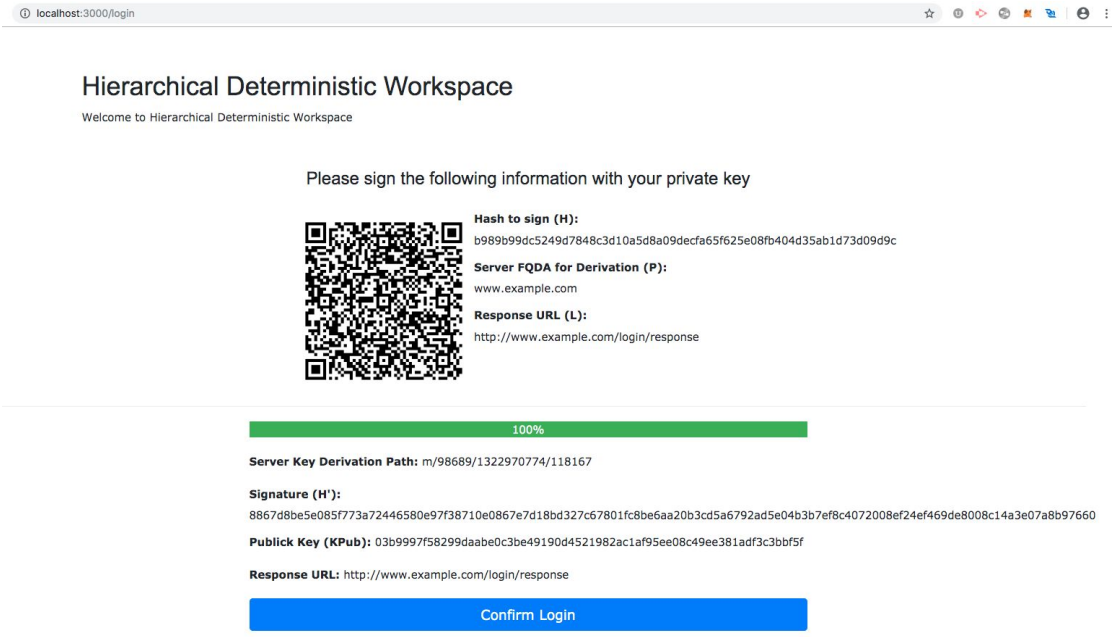
### (1) Login Page



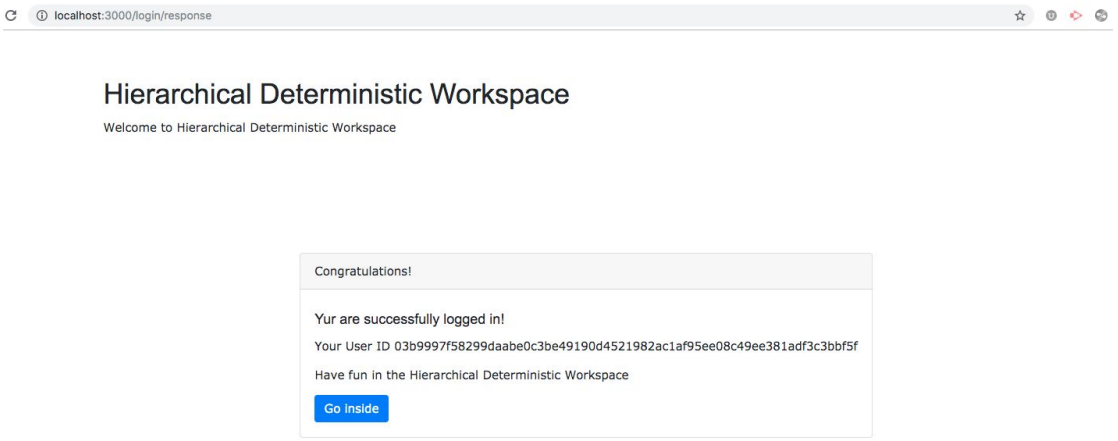

### (2) Login information is presented to the client / end user, after user clicks login button:




(3) Login Client signs login message, and ask the user to confirm:




(4) User is successfully logged in:




# 6. Conclusion

In this article, to resolve the issues in traditional login systems, a new login method—deterministic key pair based login system is proposed. It is secure, privacy-protecting, fast, transparent, and open. Deterministic key pairs are generated from a master seed, based on the Bitcoin BIP32 definition but with a new key derivation path specifically assigned to different target servers. The master seed of a user is generated from a modified BIP39 mnemonic generation process, with emphasis on the user's ability to remember and recover it with ease.

Various security considerations to use this login method are discussed: replay attack, server fishing, man-in-the-middle attack, fake and spam accounts. In addition, some use cases of a login system, such as account change and login from different devices, are discussed with solutions.

Finally, a sample system is developed to prove the concept of the login system presented in this article.

# 7. References

[1] “Common Network Authentication Methods.” *Solarwinds MSP*, 29 Apr. 2019, https://www.solarwindsmsp.com/blog/network-authentication-methods.

[2] “Single Sign On Authentication.” *Archive.is*, 15 Mar. 2014, https://archive.is/20140315095827/http://www.authenticationworld.com/Single-Sign-On-Authentication/.

[3] Perlroth, Nicole. “Yahoo Says Hackers Stole Data on 500 Million Users in 2014.” *The New York Times*, The New York Times, 22 Sept. 2016, https://www.nytimes.com/2016/09/23/technology/yahoo-hackers.html.

[4] McMillan, Robert, and Ryan Knutson. “Yahoo Triples Estimate of Breached Accounts to 3 Billion.” *The Wall Street Journal*, Dow Jones & Company, 4 Oct. 2017, https://www.wsj.com/articles/yahoo-triples-estimate-of-breached-accounts-to-3-billion-1507062804.

[5] Satoshi Nakamoto. “Bitcoin: A Peer-to-Peer Electronic Cash System.” *bitcoin*. https://bitcoin.org/bitcoin.pdf.

[6] “Cryptocurrency Wallet.” *Wikipedia*, Wikimedia Foundation, 30 July 2019, https://en.wikipedia.org/wiki/Cryptocurrency_wallet.

[7]Bitcoin. “Bitcoin/Bips.” *GitHub*, https://github.com/bitcoin/bips/blob/master/bip-0032.mediawiki.

[8] Bitcoin. “Bitcoin/Bips.” *GitHub*, https://github.com/bitcoin/bips/blob/master/bip-0044.mediawiki.

[9] Bitcoin. “Bitcoin/Bips.” *GitHub*, https://github.com/bitcoin/bips/blob/master/bip-0039.mediawiki.

[10] K P Tripathi, “A Comparative Study of Biometric Technologies with Reference

to Human Interface." *International Journal of Computer Applications*, January 2011. https://pdfs.semanticscholar.org/e5cb/5927efe46dc10f258ca2a326bee7fe9287b0.pdf

[11] *String (Java Platform SE 6)*, 19 Nov. 2015, https://docs.oracle.com/javase/6/docs/api/java/lang/String.html#hashCode().

[12]Huang, Xinyi. *Green, Pervasive, and Cloud Computing: 11th International Conference, GPC 2016, Xian, China, May 6-8, 2016: Proceedings*. Springer, 2016.

[13] Silverman, Craig. "Facebook Has More Fake Accounts Than Ever Before, Even After Removing 2 Billion Profiles." *BuzzFeed News*, BuzzFeed News, 25 May 2019, https://www.buzzfeednews.com/article/craigsilverman/facebook-fake-accounts-afd.

此页开始为致谢页

请说明每一个队员在研究报告撰写中承担的工作以及贡献；并对他人协助完成的研究成果进行说明。

如果有必要，最后可以列出团队成员和指导老师的简历。

Jiayi Liu

- Research and study current login systems
- Study cryptocurrency and its account management
- Study Bip32, 39 and 44
- Propose the new key pair generation method
- Create the new login system
- Writing the report

Wenwu Wang (Instructor)

- Professor of Signal Processing and Machine Learning. Co-Director of A-Lab (Machine Audition Lab) at the University of Surrey.
- Instructions, discussions, and support the author with his research.

此页为学术诚信声明

本参赛团队声明所提交的论文是在指导老师指导下进行的研究工作和取得的研究成果。尽本团队所知，除了文中特别加以标注和致谢中所罗列的内容以外，论文中不包含其他人已经发表或撰写过的研究成果。若有不实之处，本人愿意承担一切相关责任。

参赛队员：Jiayi Liu　　指导老师：Wenwu Wang

2019年 09 月 07 日