Memo

A fatal problem that occur when operating a blockchain

Problem 1:
Since the public key does not have the quantum resistant, it is possible to calculate the private key data. The computing power of quantum computers can steal the crypto assets of others and turn them into their own.
Background:
Satoshi Nakamoto used an anonymous public key as a means of cutting off the flow of information.
Problem 2:
Hackers can steal online private keys directly without using the huge computing power.
Background:
A password limits access, but cannot cut off the flow of online private key information.

Verification

Satoshi Nakamoto used a public key without an X.509 certificate to allow recipients to verify the chain of ownership. Quoting (10. Privacy) from his treatise:

> 10. Privacy
> The traditional banking model achieves a level of privacy by limiting access to information to the parties involved and the trusted third party. The necessity to announce all transactions publicly precludes this method, but privacy can still be maintained by breaking the flow of information in another place: ***by keeping public keys anonymous.*** The public can see that someone is sending an amount to someone else, but without information linking the transaction to anyone.

The figure below is the New Privacy Model quoted from "10. Privacy". The blue text and lines in the figure are what I added. *By making the public key anonymous*, the flow of information from "Identities" to "Public" is cut off at the boundary defense line (bitcoin address).

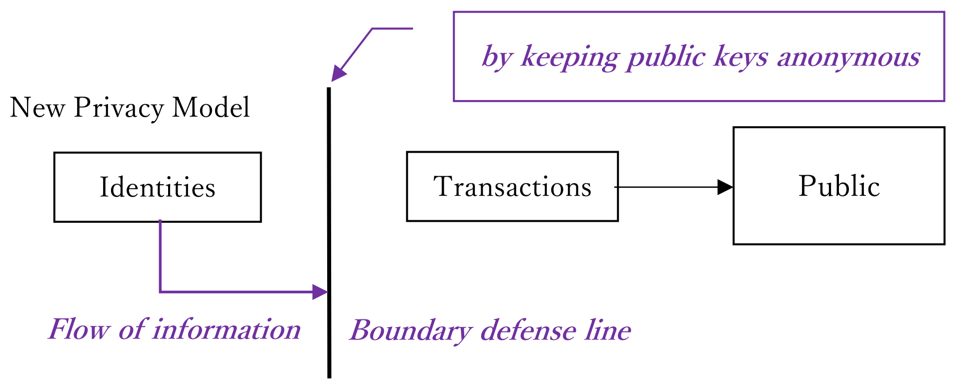

Fig.1: Boundary Defense Line = Anonymous Public Key = Bitcoin Address

The “Identities” of the public key with a certificate are certificate authorities. The “Identities” of public keys that do not have a certificate are “Private keys” data. This “Identities” allows the payee to verify the signature and verify the ownership chain (ownership of electronic coins). ☞

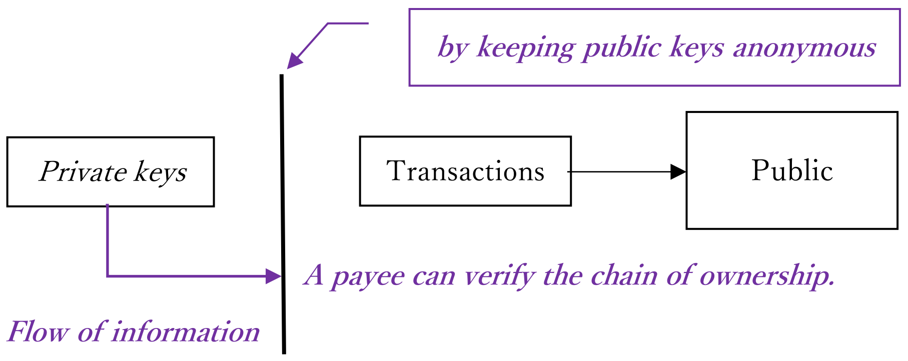
by keeping public keys anonymous
Private keys
Transactions
Public
A payee can verify the chain of ownership.
Flow of information

Fig.2: With this “Identities” the payee verifies the signature.

However, ownership of electronic coins is stolen due to the following two factors. One is that public keys are not quantum resistant: the other is that online private key data is leaked by cyberattacks.

Verification of problem 1

Since the commutative algorithm type public key does not have the quantum resistant, Fig. 2 becomes Fig. 3.

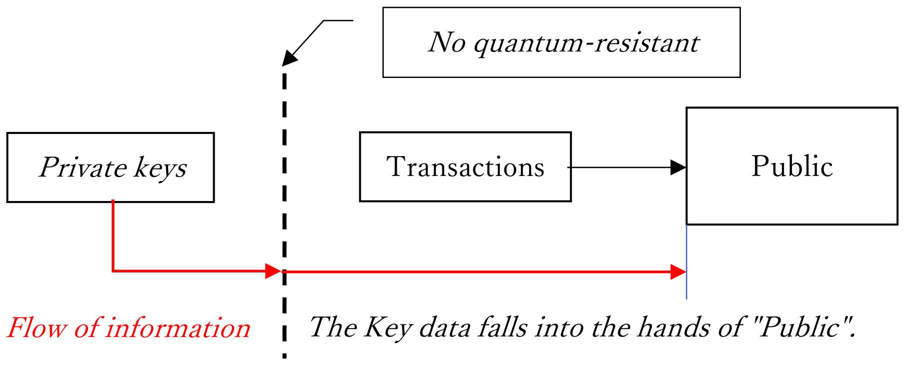

Fig.3: Ownership of electronic coins is stolen.

Verification of problem 2

ID password is used when operating the blockchain. The password originally limits access to information, and does not cut off information flow of the key data. The serious thing is that a password flows information from the definition itself. As shown in the figure below:

Memo

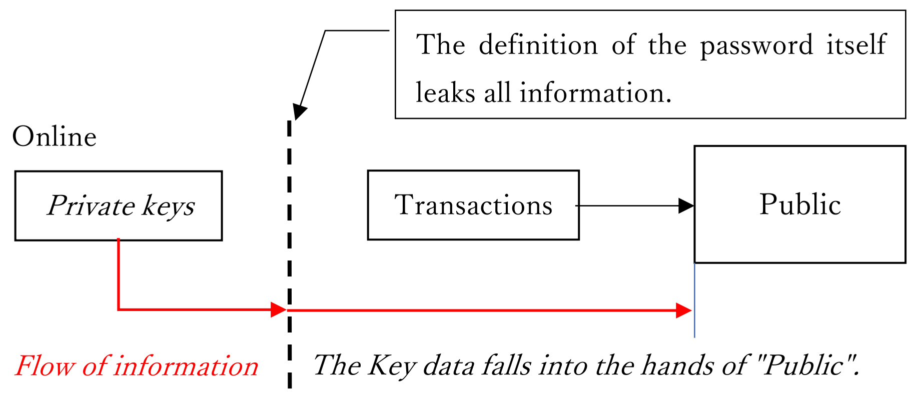
The definition of the password itself leaks all information.
Online
Private keys
Transactions
Public
Flow of information
The Key data falls into the hands of "Public".

Fig.4: Ownership of electronic coins is stolen.

Overlapping part of problems 1 & 2

Fig.3 and Fig.4 express that they are the same in that "the flow of information from" Private keys "to" Public" is not interrupted." That is, Fig.3 = Fig.4. This means that even if the public key is made quantum resistant, the Key data will still fall into the hands of "Public".

What is the real challenge?

Think about your account. We have long believed that user accounts are managed by the service provider: here password registration is also managed by the service provider. Therefore, even in the blockchain, Xchange and securities counter manage user accounts, and passwords limit access to the signing key. However, as long as a password is used, the flow of signature key information cannot be interrupted.

The real challenge is to operate the signing key without using a password. *Logically digging into this requirement: the system has no key data either online or offline, but can use it when signing.* What a mysterious scenario!

Non-commutative algorithms perform the mysterious scenario above.

Definition 1:

Do not require password registration.

Definition 2:

The private key data is "burned" as soon as it is generated.

Definition 3:

This implementation is also quantum resistant.

There is a realistic algorithm that covers all the three definitions: the non-commutative algorithm. The formalism of this non-commutative algorithm is noted in Appendix 1. Here, the function Y () corresponding to the cryptographic function and the collision function Y^{-1} () corresponding to the decryption function are cited and applied to Satoshi Nakamoto's New Privacy Model. As follows.

Private key data is "burned".

Apply the function Y () to Private key to "burn" the key data. Since the key data no longer exists anywhere, the information flow itself disappears. That is, there is no target for limiting access.

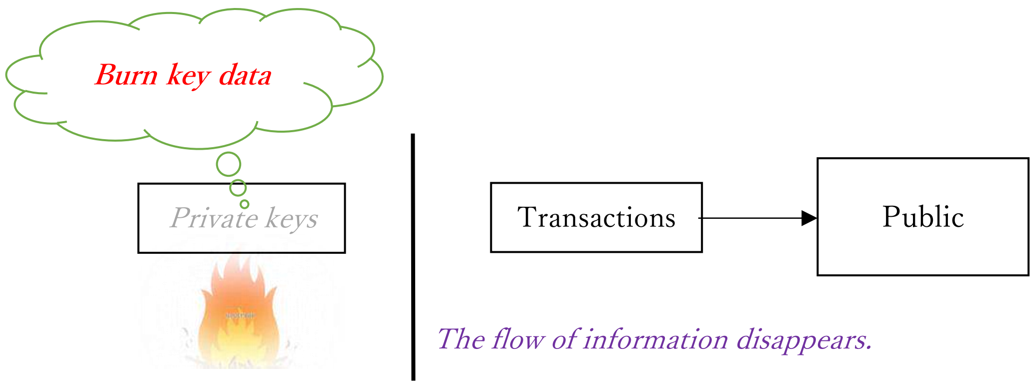

Fig.5: There is no target to limit access.

After burning the Private key data, three code IDs appear in the output of the function Y (). This will erase the key data from any memory, leaving only the signing task. The boundary defense line shifts to the left as shown in Fig.6.

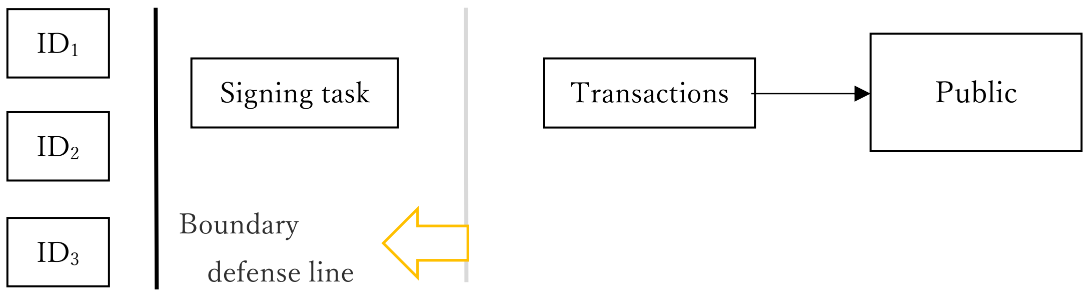

Fig.6: The boundary defense line moves to the left.

Three code above {ID_1, ID_2, ID_3} activates the signature task as follows.

Memo

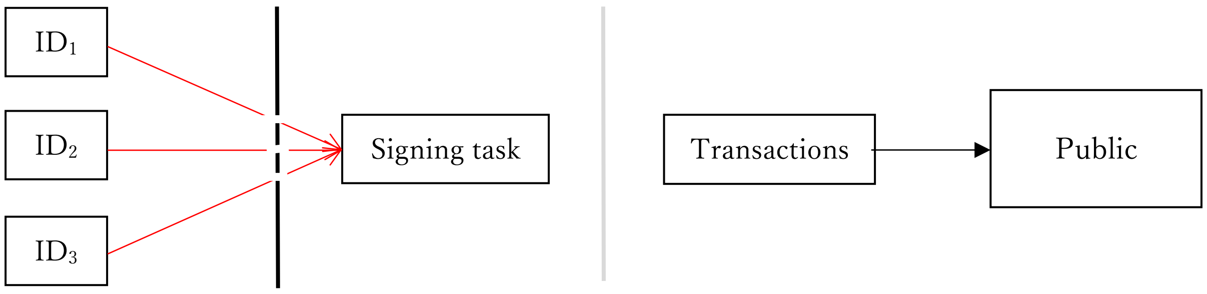

Fig.7: No password exists: No account exists.

That is, a user has ID_1, the Xchange has ID_2, and the third party has ID_3: When the user, the Xchange, and the third party each agree to sign, the key data is reproduced in the signature task: Immediately after the signature is complete, the key data is deleted. Details around this are given on the METEORA SYSTEM homepage. ☞ https://meteora-system.com ☞ Multivariable Digital Currencies

A point of interest

According to common sense, it is the service provider who manages the user's account. Therefore, as long as the password is used, the flow of key information cannot be interrupted. This is a fatal problem that occurs when operating a blockchain. The non-commutative algorithms solve this problem.

Suffice it to say, an account is an agreement protocol between a user, an exchange, and a third party. There is no DB that holds the ID password. This situation is shown in Fig.7.

Hackers use quantum computers to steal private key data from public key data. Once the private key data is available, it is easy to steal electronic coins. No one stops it. However, in Fig.6, the key data does not exist online, nor does it exist offline. The method for updating the signature chain (or stealing electronic coins) is as shown in Fig. 7. The private key data is finally hacked, but there is no use for Fig.7. That is, it has perfect quantum resistance. This non-commutative algorithm has no lifetime. There is no reason to wait for the NIST standardization process.

© Eiji Watanabe
© METEORA SYSTEM
January 28, 2021

Appendix1

The non-commutative algorithm is not compatible with cryptographic processing $K_1()$ and decryption processing $K_2()$. Here, K_1 and K_2 both represent random variables. Performing encryption processing for plaintext P is expressed as $K_1(P)$, and then performing decryption processing is expressed as $K_2(K_1(P))$. Here, the expression K_1K_2 is simply used.

If code C is created from plaintext P with the public key as the encryption key and returned to plaintext P with the private key as the decryption key, and vice versa, the result will return to the same P:

$$K_1K_2 = K_2K_1$$

This is a commutative algorithm. However, in non-commutative algorithms, the equal part is the inequality sign:

$$K_1K_2 \neq K_2K_1$$

In this non-commutative algorithm, the function corresponding to cryptographic processing $K_1()$ is represented by $Y()$: and $Y()$ is represented in a unique form:

$$Y() \equiv <Y_1(), Y_2(), Y_3()>$$

Three code IDs (n=3) appear in this output (Fig.6). Of these, if (n-1) IDs are leaked, it is difficult to calculate the secret information from those leaked IDs: because this calculation is a probability calculation, and the probability of hitting the secret information is $1/2^{256}$. This means that the dice are thrown 2^{256} times on the net, not inside the quantum computer.

On the other hand, there is a function named "collision function" as a function corresponding to the decoding process $K_2()$. This is also represented by the unique form $Y^{-1}()$:

$$Y^{-1}() \equiv <Y_1^{-1}(), Y_2^{-1}(), Y_3^{-1}()>$$

Thus, there is no key equivalent to the public key. However, as a convenience instead of the public key, even if (n-1) code IDs are leaked, the calculation of the collision function $Y^{-1}()$ cannot be deceived. The event of collision itself is supposed to be a brute force attack, so it cannot be deceived. *In this sense, the collision function $Y^{-1}()$ corresponds to the public key. Similarly, the function $Y()$ corresponds to the private key. This shows the existence and characteristics of non-commutative algorithms.*

There is a model that implements the functions $Y()$ and $Y^{-1}()$ on the server. In this case, it is a Static function. When applied to the blockchain, the function $Y()$ is used for One-time, while the collision function $Y^{-1}()$ has no limit on the number of times it can be used. Expressing $K_1K_2 \neq K_2K_1$ as $K_1K_2 - K_2K_1 = \Delta$, this is a form of quantum mechanics.

Memo

//