Money Transfer on Transaction Signature-Based Ledger

Momoko Shiraishi(✉) and Hitoshi Aida

The University of Tokyo, Tokyo, Japan
shiraishi@os.ecc.u-tokyo.ac.jp

Abstract. Money transfer is indispensable in our daily lives. For providing the services, financial institutions bear great costs for verifying customers' identity and behaviors, so called KYC (Know Your Customer) costs. This paper proposes a comprehensive transaction scheme to reduce the KYC costs, by sharing customers' information among multiple financial service providers. Once a trusted service provider identifies a user using an image of his/her physical ID such as passports and residence certificates, the provider issues a digital certificate of the user's public key. When the user opens another account at different providers, the providers can identify the user with the certificate without proceeding the KYC step. The security analysis shows that the proposed transaction scheme is resistant even to harsh attacks in a network represented by Man-in-the-Middle (MITM) attacks, as long as the user's physical ID is tied to his/her public key appropriately. The performance analysis shows that the proposed scheme is applicable in terms of computation time and storage space.

Keywords: Public key infrastructure · Transaction signature · Digital identity

1 Introduction

Money transfer service is indispensable in our daily lives and also for economic growth. Meanwhile, for providing secure transfer services, financial institutions must meet the high-level security requirements. KYC (Know Your Customer) is the mandatory process of verifying the customer's identity at opening an account and during the service use over time, while the cost is high. According to a Thomson Reuters survey [24], the average cost for a bank to meet the KYC compliance is 60 million U.S. dollars a year, and some banks spend up to 500 million U.S. dollars annually.

This paper then proposes a comprehensive transaction scheme to reduce the KYC costs. Under the proposed scheme, service providers can share the users' identity information with stronger security and also verify the traceability of the transaction data in the long run. The underlying idea of the proposed scheme is that a provider issues a digital certificate of a user's public key. Different providers can confirm the user's identity, only by verifying the validity of the

G.-V. Jourdan et al. (Eds.): FPS 2022, LNCS 13877, pp. 338–354, 2023.
https://doi.org/10.1007/978-3-031-30122-3_21

certificate. Also, the traceability of a transaction data can be attained by its multi-signature between a sender and a receiver. As a result, it leads for providers to reduce their KYC costs.

The contribution of this paper is summarized as follows:

- We newly provide a transaction scheme, sharing individuals' public key as their identity information. A trusted host service provider issues each user's public key certificate after confirming his/her physical ID. When the user opens another account, providers only need to verify the certificate issued by the host provider. This mechanism reduces KYC costs for each provider, possibly resulting in low transaction fee.
- Under the proposed scheme, a transaction is multi-signed by the sender and the receiver, and stored in the host provider's database with the multi-signature. The signed ledger is called as Transaction Signature-based Ledger (TS-L). As a result, the traceability of the transaction data can be assured by its signature verification.
- The security analysis shows that the proposed mechanism is resistant to attacks in a network communication such as Man-in-the-Middle (MITM) attacks, as long as the user's physical ID is tied to his/her generated key pair appropriately. Particularly in remittance, under satisfying the condition, even if losing a secret key, unintended transactions due to its leakage can be traced and canceled with a rationale.
- The performance analysis shows that the scheme is easily applicable in terms of signing and verifying time.

The next section describes the architecture of the proposed scheme. After Sect. 3 presents its security analysis, Sect. 4 offers the performance analysis. Section then 5 provides related works. Lastly, Sect. 6 provides a conclusion.

2 System Architecture

This section describes the transaction scheme on the Transaction Signature-based Ledger (TS-L). A transaction indicates a money transfer between accounts. Also, as a premise, when each user digitally signs, it is assumed that the user possesses a secret key and a public key. The proposed scheme relies on FIDO2 (Fast Identity Online) protocol [1] for creating keys and signing in the user's device.

2.1 Related Parties

As shown in Fig. 1, exhibiting the transaction execution process, there are seven roles: a sender, a receiver, a trusted service provider, an inquirer, a conventional bank, a Time Stamping Authority (TSA), and a Certificate Authority (CA). Hereafter, "a user" indicates an entity who registers for the service, including the sender, the receiver and the inquirer. The user can represent either one individual or one organization. The following describe the parties respectively.

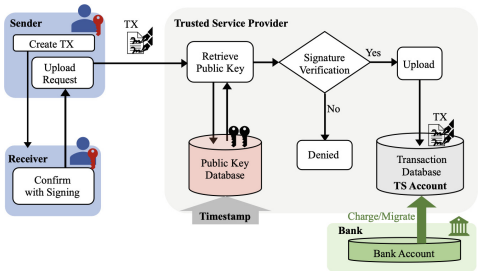

Fig. 1. Overview of the transaction execution process

Sender: An entity who registers for the transaction service, opens the Transaction Signature-based account (TS account), and sends money to a receiver.

Receiver: An entity who registers for the service, opens the TS account, and receives some money from the sender.

Trusted Service Provider: An entity who provides the transaction service, and manages the users' TS accounts. The trusted service provider generates a pair of secret and public keys, and the certificate of the provider's public key is issued by a CA. The provider is "trusted" in terms of verifying transaction signatures and managing databases appropriately. Specifically, the provider notifies administrative authorities of the money transfer business and establishes his/her trust. In a case that the sender and the receiver have opened their accounts under different providers, multiple providers will be involved in the transaction. Here, all of the providers are assumed to be trusted, for example, without considering a case of a collusion with unauthorized recipients to take money from a legitimate sender.

Inquirer: An entity who registers for the service and refers to a user's transaction records. It can be the data owner himself/herself or someone authorized by the data owner.

Conventional Bank: An entity who manages some users' deposits, when this transaction service is about to start. The first step to initiate the transaction service is to inflow money, from existing bank accounts to TS accounts. For this transfer operation, the conventional bank also registers for the service, generating a pair of secret and public keys. Then, the bank as a sender and a user who migrates his/her bank account into his/her TS account, as a receiver, conduct the transaction procedure in the same as a transaction between individual users. When the conventional bank switches to a TS account-based system, the trusted service provider is identical to this conventional bank.

Time Stamping Authority (TSA): An entity who issues timestamps to the users' public keys. A time information affixed to each transaction is also assumed to be conveyed from the TSA to the provider.

Certificate Authority (CA): A trusted party who issues digital certificates for public keys.

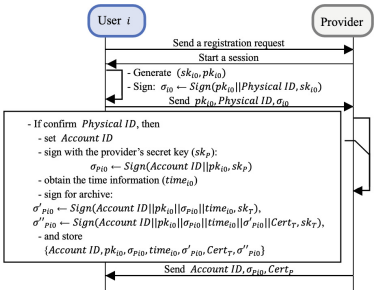

Fig. 2. Service registration protocol

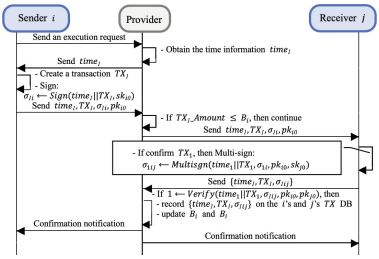

Fig. 3. Transaction execution protocol under identical provider

2.2 Protocol Design

Under the transaction service based on the TS-L, the six operations are prepared. The following subsections explain them.

1) Service Registration: First of all, a provider is assumed to hold a pair of secret and public keys (sk_P, pk_P). A certificate for the provider's public key $(Cert_P)$ is issued by a CA. Also, assume that a channel between the provider and a user is based on a Transport Layer Security (TSL) communication. Any certificate used for the communication is separated from the provider' certificate $(Cert_P)$ and the user's certificate created in the following registration process. As shown in Fig. 2, at the beginning, user i generates a pair of secret and public keys (sk_{i0}, pk_{i0}). The secret key is stored in a secure area in the user's device. The user then prepares his/her physical ID information and takes its photo, converting the physical ID into the digital information. The user also takes his/her profile photo. These digital information for the identity proofing are denoted as $Physical\ ID$ in this model. The method to obtain the $Physical\ ID$ relies on the digital identity guidelines by NIST [11]. For an individual, the $Physical\ ID$ is for example, passport, residence certificate and driver's license. For an organization as the user, it will be some legal documents. One novelty in the proposed scheme is that the user here creates a signature (σ_{i0}) for the $Physical\ ID$ and his/her public key (pk_{i0}). The signing process is based on the FIDO protocol, and assumes that a credential such as biometric information is input to the device. The matching of the credential is confirmed in the device, and the signature is created. The signing on the image of the physical IDs is possible because the image information and the signing process can be handled simultaneously on a device. Note that since the current FIDO protocol does not cover this scheme, for more specific mechanism for signing the image in the device, further analysis is expected. These information $(pk_{i0}, Physical\ ID, \sigma_{i0})$ are then sent to the provider. If the provider confirms the $Physical\ ID$ by image verification processing, first creates the user's account ID $(Account\ ID)$. Specifically, the provider here stores the user's personal information such as his/her address and birthday from the physical ID information, linking them to the $Account\ ID$.

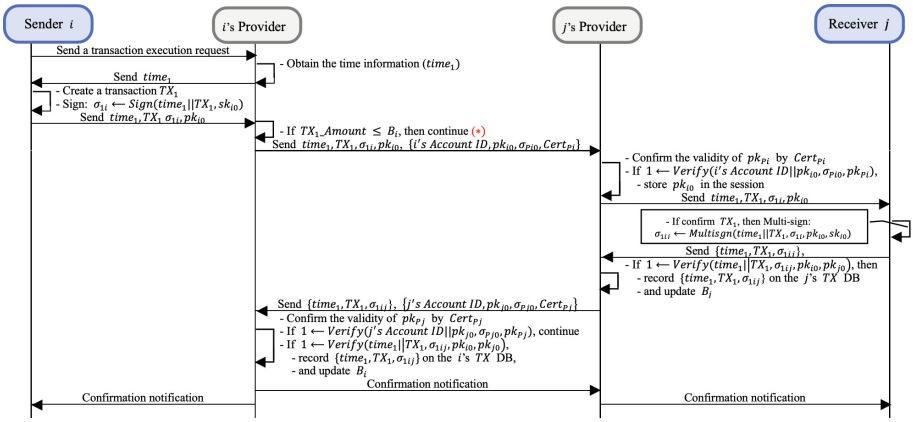

Fig. 4. Transaction execution protocol under different providers

Afterwards, the provider signs the user's public key (pk_{i0}) with the provider's secret key (sk_P), outputting the signature (σ_{Pi0}). This signature is the certificate of the user's public key (pk_{i0}) issued by the provider. The provider then asks its long-time signature to the TSA. The process relies on the long-term signature standard offered by ISO (International Organization for Standardization) [16]. The long-term signature (σ''_{Pi0}) is obtained using the TSA's secret key (sk_T). As a result, the provider stores $\{Account\,ID, pk_{i0}, \sigma_{Pi0}\}$ with its long-term signature in the user's public key database, and notifies the user of $Account\,ID$, σ_{Pi0} and $Cert_P$. Here, the user's public key registration is completed and the TS account is opened. The set of $\{Account\,ID, pk_{i0}, \sigma_{Pi0}, Cert_P\}$ can be used for identity proofing for different providers from this time on. Hereafter, "a host provider" for a user indicates the provider who issues the certificate for the user and records the user's public key updates.

2) Transaction Execution: As a premise, the proposed system does not provide operation of deleting or modifying once recorded transaction data, but only appending operation. The deletion or modification purpose is achieved in cancelling operation described in the following (2.2.3.Transaction Cancellation).

As shown in Fig. 3, consider a case where a sender i transfers money to a receiver j. Both of them are assumed to have opened the TS account under the same provider. After the provider receives a transaction request from the sender i, the provider obtains the time information ($time_1$) from the TSA and sends it to the sender. The sender then creates a transaction, i.e., $TX_1 = \{i'\,Account\,ID, j'\,Account\,ID, Amount\}$. The sender signs the TX_1 using his/her secret key (sk_{i0}), outputting the signature (σ_{1i}). After the provider receives the information ($time_1, TX_1, \sigma_{1i}, pk_{i0}$), first the provider confirms whether the transfer amount (TX_1_Amount) is less than the balance of the i's TS account (B_i). If it is satisfied, the provider sends the information to the receiver. The receiver confirms the content of the transaction and signs

TX_1 with his/her secret key (sk_{j0}), creating the multi-signature (σ_{1ij}). The provider verifies the multi-signature with the sender's and receiver's public keys (pk_{i0}, pk_{j0}), referring their account IDs. If it is successful, the transaction record $\{time_1, TX_1, \sigma_{1ij}\}$ is uploaded on the transaction database of the sender's and the receiver's account. At this time, their balance (B_i,B_j) are also updated. This is the whole process for transaction execution under one provider.

On the other hand, when the sender and receiver have opened the TS account under different providers, additional steps are required. Assume that a channel between the providers relies on a TSL communication. The server certificates used for the communication are separated from the providers' certificates for signing users' public key ($Cert_{Pi}, Cert_{Pj}$). As shown in Fig. 4, after the sender i's provider receives the transaction information from the sender i ((*) in Fig. 4), this time the provider sends the information ($time_1, TX_1, \sigma_{1i}, pk_{i0}$) to the receiver j's provider, adding the validity information about the sender's public key $\{i's\, Account\, ID, pk_{i0}, \sigma_{Pi0}, Cert_{Pi}\}$. The j's provider then confirms the validity of the i's provider's public key (pk_{Pi}) by using its certificate ($Cert_{Pi}$). If it is successful, next the j's provider verifies the signature (σ_{Pi0}) with that i's provider's public key (pk_{Pi}). If it is also successful, the j's provider stores the i's public key (pk_{i0}) during this session. The receiver then confirms the transaction TX_1, and creates the multi-signature (σ_{1ij}). The j's provider verifies it using the sender's and receiver's public key (pk_{i0}, pk_{j0}). The sender's public key is the one stored in the previous step. If it is successful, the j's provider inserts the transaction data $\{time_1, TX_1, \sigma_{1ij}\}$ on the j's transaction database. Afterwards, the i's provider conducts the same procedure as the j's provider did. When all the verification are completed, the transaction data $\{time_1, TX_1, \sigma_{1ij}\}$ is recorded on the i's transaction database Here, the execution process is completed. Although we have shown the two cases regarding whether the sender's and receiver's providers are identical or not, the mechanism does not vary. The transaction is signed with the sender's and receiver's secret keys, and the provider verifies and records it. Under the different providers' case, the additional step is only for the provider to confirm the validity of the user's public key by using the valid host provider's public key.

In the practical use, the first step to activate this system is to transfer money from the conventional bank account to the TS account. The conventional bank is now assumed to be different from the provider, therefore the bank registers for the service and generates the bank's secret and public key. When user i transfers money from the user's bank account to the user's TS account, he/she sends a transfer request to the bank via the provider. According to the request, the bank creates the transaction $TX_B = \{i's\, bank\, Account\, ID, i's\, TS\, Account\, ID, Amount\}$. The bank signs the TX_B with its secret key, creating the signature. When confirming the transfer (TX_B), the user signs with his/her secret key, creating the multi-signature. The multi-signature is verified with the bank's key and user's public key by the provider. If it is successful, the money transfer (TX_B) is executed. The money *Amount* is subtracted from the i's bank account in the conventional bank system, and the *Amount* is added to the TS account,

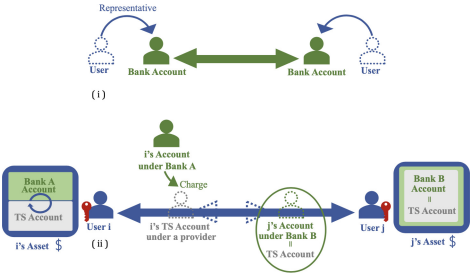

Fig. 5. Implementations of the TS account

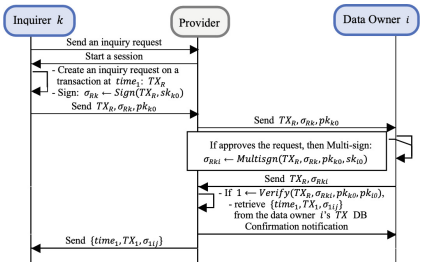

Fig. 6. Inquiry protocol

with the balance updated. Even when the provider is the conventional bank itself, and the bank migrates the existing system to the TS accounts-based system, the same procedure is conducted. In that case, the bank verifies the transaction which is multi-signed by the bank itself and the user. Figure 5 illustrates example cases to implement the TS accounts. Under (i), the conventional banks take the money transfer service. Under (ii), user i opens a TS account under a provider, totally holding two accounts, an account under the bank A and the TS account. On the other hand, the bank B migrates to the TS-based system, and the user j remains his/her asset under the bank B, while all of the records are stored with the transaction signatures.

3) Transaction Cancellation: If, for some reason, unintended transactions are executed, cancellation is possible by newly creating offset transactions. For example, when cancelling the transaction $TX_1 = \{i's\ Account\ ID, j's\ \ Account\ ID, \$10\}$, the offset transaction $TX_1' = \{j's\ Account\ ID, i's\ Account\ ID, \$10\}$ is executed. Although the execution procedure is completely identical between the TX_1 and the TX_1', when cancelling related to past transaction records, evidence is required. It is that the past transaction (TX_1) was definitely executed at a certain point.

The provider saves the transaction in the form of $\{time_1, TX_1, \sigma_{1ij}\}$, including the time information ($time_1$). Additionally, the provider records the history of the user's public keys. From the sender's (receiver's) account ID and the transaction time ($time_1$), the provider identifies the i's public key at that time as a certain public key (pk_{i0}). The validity of the user's public key is assured by the provider's signature (σ_{Pi0}). Here, the signature is supported by the validity of the provider's public key (pk_P), while the provider's public key is supported by the certificate issued by the CA ($Cert_P$). Furthermore, since all the users' public keys are stored with archive-stamps and the long-term signatures are updated by the TSA, it is possible to prove the existence of the user's public key for a long time into the future. As a result, using the pointed valid user's public key (pk_{i0}), the fact of the transaction (TX_1) is confirmed, which is described as $(1/\perp) \leftarrow \mathbf{Verify}(time_1||TX_1, \sigma_{1ij}, pk_{i0})$. If it is successful, then the fact is presented that the sender was consent to the transaction (TX_1) at the time ($time_1$).

The link between the TS account ID and the person in real is tied in the registration process using the physical ID, therefore, the individual's commitment for transactions can be investigated. Moreover, even if the sender's balance at the cancellation time is not enough, the offset amount can be charged with the rationale. In this way, it is shown that the proposed scheme satisfies traceability, accountability and non-repudiation.

4) Records Inquiry: Two types of users can refer the records in the database: the data owner himself/herself, or someone else authorized by the data owner. Referring to Fig. 6, consider that an inquirer k, different from the data owner i, inquires about the i's records. The inquirer request for the transaction of the data owner at a certain time ($time_1$) is described as $TX_R = \{k's\ Account\ ID, i's\ Account\ ID, time_1\}$. The inquirer signs the request with the inquirer's secret key (sk_{k0}), creating the signature (σ_{Rk}). The information ($TX_R, \sigma_{Rk}, pk_{k0}$) is sent to the data owner via the provider. When confirming the request, the data owner creates the multi-signature (σ_{Rki}). The provider verifies it with the inquirer's and data owner's public key (pk_{k0}, pk_{i0}). If it is successful, the provider retrieves the requested record $\{time_1, TX_1, \sigma_{1ij}\}$ from the data owner's transaction database, and sends it to the inquirer.

Under the case that the inquirer and the data owner are not identical, registering under different providers, then the providers exchange information about the validity of the inquirer's and data owner's public key one another. This is completely the same as it in the transaction execution process in Sect. 2.2. Moreover, when the inquirer is identical to the data owner, a single-signature with the inquirer's secret key is created, instead of a multi-signature.

5) User Side's Key Refreshing: The critical issue of the TS-L scheme is that an attacker obtains a legitimate user's secret key and uploads a false transaction. Note that there is a recovery option by cancellation in case of the leak as described in 2.2.3. When the user notices his/her secret key leakage, he/she conducts this refreshing process at first, and then cancels the unintended transactions. Moreover, as a precaution, it is recommended for each user to refresh the key pair periodically. Here, also in terms of compliance, it is desirable that customer information be kept inspected periodically in addition to the moment of opening accounts. This key exchange is assumed to be done with strict verification of identity, thereby, it is considered that continuous monitoring can be conducted with this. Furthermore, particularly if the user is a company rather than an individual and some periodic legal documents are additionally required, then the submissions can be signed by the host provider and shared with different providers. As for the process, based on Fig. 2 in the registration phase in Sect. 2.2.1, the user generates a new key pair (sk_{i1}, pk_{i1}), and creates a signature (σ_{i1}), given by $\sigma_{i1} \leftarrow \mathbf{Sign}(pk_{i1}||Account\ ID||Physical\ ID, sk_{i1})$. Here, also the user's account ID ($Account\ ID$) is added in signing. After the set of the information ($pk_{i1}, Account\ ID, Physical\ ID, \sigma_{i1}$) is sent to the provider, the same procedure as the registration phase is conducted. Finally, the user's new public key (pk_{i1}) is registered and the set of its certificate and the provider's public key certificate ($\sigma_{Pi1}, Cert_P$) is sent to the user. The updated public key (pk_{i1})

is also delivered from the host provider or the user himself/herself to different providers.

6) Provider Side's Public Key Database Validating: The host provider stores the history of the users' public keys with the long-term signatures, so that the transaction traceability, accountability and non-repudiation are satisfied, as described in cancellation process in 2.2.3. Here, the certificates in the archive timestamps attached to all public keys expire at some point in time, requiring to update them. The providers then ask the TSA to renew all the certificates at the expiration time. A new certificate is attached and a new signature is created, and they are stored affixed to each public key information. The storage size for the public key updates is provided in Sect. 4.

2.3 Identity Proof Sharing and Paths for Reducing KYC Costs

In addition to the purpose of ensuring the transaction traceability, the user's public key information dedicates to share his/her identity information among different service providers. Once a trusted host provider confirms user i's identity using his/her physical ID, then the information $\{Account\,ID, pk_i, \sigma_{Pi0}, Cert_P\}$ can be shared. When the user opens another TS account under a different provider, the provider first checks the validity of the host provider's public key (pk_P) using its certificate ($Cert_P$). Afterwards, the provider verifies the signature (σ_{Pi0}) using the confirmed host provider's public key (pk_P), described as $(1/\perp) \leftarrow \mathbf{Verify}(Account\,ID||pk_{i0}, \sigma_{Pi0}, pk_P)$. Note that the validity period of the provider's public key (pk_P) is longer than the validity period of the user's public key (pk_{i0}), recommending each user to refresh his/her key as a precaution for its loss and also for periodical monitoring checks. In this way, the user's public key information can be shared among different providers based on the trust to the host provider. This leads to save the different providers' KYC costs. When the different provider from the host provider starts the service, the legitimacy of the registration request is verified, only by verifying the user's public key and an affixed signature on the request. Unlike the other conventional authentication methods such as passwords, the different providers can verify requests directly with a public key as a credential.

3 Security Analysis

The security goal of the TS-L is to achieve data integrity in the sense that the transaction is intended for the sender and the receiver of the money. This section describes the possible vulnerabilities to break the integrity and the corresponding countermeasure. We consider four places: a channel between a user and a provider, the user side, the provider side, and a channel between the providers.

1) Channel between the User and the Service Provider: While the channel between the user and the provider implements the TSL communication, there

is a possibility of MITM (Man-in-the-Middle) attacks, in which an attacker positioned between two communicating parties intercepts or alters data traveling between them [6,11]. The attacker replaces an original server certificate with a modified certificate. If the user neglects a warning notification from a browser and inputs the required information for remittances, then the attacker can execute fraudulent money transfers using the legitimate user's account. Under the proposed scheme, transaction signatures are created in the user's device. Therefore, even if the attacker rewrites the transaction contents on the communication channel, the provider can detect the tampering in verifying the signature. This is the primary advantage of using transaction signatures. The signing is completed in the user's device with a credential such as biometric information and the credential is kept in the device, enabling to be resistant to the attacks arising in a network. Multi-signatures are used not only in transaction execution but also in inquiry. This is for the purpose of maintaining the message integrity of all the requests in the network. Since all networks are insecure, regardless of whether it is inside or outside an organization network, data is signed within individual devices and sent to a recipient. In other words, the zero trust [25] is assumed here. Since the transaction signatures inherently contributes to security in these attacks in the network, some methods were proposed. For example, IBM Zurich Research Laboratory invented a token-based transaction signature for online banking [34]. However, as a whole, these specialized devices take costs to prepare, becoming an obstacle to implement them [9]. A feature of the proposed method is that it assumes authentication with familiar devices such as smartphones and PCs, without preparing additional devices, dedicating for the provider's efficiency.

2) User Side: In the current FIDO authentication scheme, a mis-binding attack has been pointed out [12,14]. In a registration phase, attackers register their own device linking a legitimate user name, indicating that the attackers' public key is registered to the application provider. Communication within modules in a device is not authenticated for one another, allowing malware to perform unauthorized operations. For this attack, the proposed method requires the signature information of the physical ID at the registration. Therefore, unless the attacker obtains the image of the legitimate physical ID, the provider detects the fraud, and the attacker fails to register by impersonating the legitimate user. Ultimately, under the assumption that providers are trusted, the most serious problem is that the image of the physical ID is forged by the attacker. If the image of the digitized physical ID of the sender is forged and the receiver's physical ID is correctly submitted, it can be canceled by the cancellation process. However, if the digitized physical IDs for both the sender and the receiver are forged and the attacker's public keys are registered to the host provider, then the fraudulent transaction cannot be canceled, although it would be difficult for the attacker to succeed it. When the fraudulent transaction amount is converted to cash, the legitimate sender would lose the money. In order to prevent this physical ID forgery attack, it is recommended, for example, to improve the accuracy of image verification.

The following attacks might be successful. However, ultimately cancellations work for the attacks, as long as the linkage between a physical ID and a public key for the receiver is appropriately tied. A straightforward attack is that credentials used for signing such as biometric information and PIN codes, are stolen. However, even if these credentials are forged, fraudulent requests can be cancelled if the linkage between a public key and a physical ID is properly linked. Under MITB (Man-in-the-Browser) attacks, a malware infects the web browser. An attacker eavesdrops and alters the transaction content between the web browser and the web server, and executes unintended requests. Zbot is a representative malware, first identified in 2007 [33]. While the MITB attack is basically difficult to be observed and unsolvable, the proposed transaction scheme is resistant to this MITB attack. The current FIDO protocol defines the software module to create the signature called authenticator. The transaction signature is created in that authenticator, then the required signed information is passed from it to the browser. Therefore, even if a fraudulent transaction is created in a malicious browser, the legitimate signature cannot be created in the legitimate authenticator, enabling to be detected in the provider's signature verification. However, once if several parts in the device are infected with malware, unintended transactions can be successfully verified by providers. Under a clickjacking attack, a malicious software presents a false display and executes unintended transaction operation [13,15,23]. In this case, the legitimate user signs the fraudulent transaction with his/her own secret key, without being aware of it. The provider then approves the transaction. Note that the false transaction can be cancelled later. Specifically, a parallel session attack in software modules within a device supports the successful attack. Under the parallel session attack, a malicious software module exists between legitimate modules [12,14]. The attacker sends a request again and obtains random values generated for each session, leading the successful attack. A DoS (Denial of Service attack) attack is another possible attack [12]. A malicious software present in the device can halt transaction executions. Although these malware are difficult to be implemented, improvements in the protocol are expected. For example, it is recommended to authenticate each software module in the user's device for each session.

3) Provider Side: For appropriate transaction executions, the provider has mainly four roles under assuming that the provider is trusted. First, the provider surely confirms the authenticity of a person in real with his/her physical ID in the registration phase. The requirement of the physical ID in the account opening process will determine the level of security of the service. Second, when executing transactions, the provider properly verifies the transaction signature sent by the user and records it on the database. The provider prepares the control devices in a secure place, as well as appropriate allocation of its security managers within an organization. Third, the provider properly manages databases, i.e., the account information database, the transaction database, and the public key database. Against malfunction, cyber attacks, and natural disaster, the databases are located in multiple places, and hold both online and offline backup options for one another. Note that it does not imply that each user has the

Table 1. Parameters for Performance Analysis

Notation	Description	Benchmark
T_{R}	The average period for refreshing a user's key pair	-
T_{E}	The average validity period of certificates for long-term signatures ($Cert_T$)	10 [years] [10]
T_{TX}	The average frequency for executing a transaction	-
T	Arbitrary time	-
n_{pk}	The number of bits for a public key (pk)	96 [bytes] [4]
n_{sig}	The number of bits for a signature (σ)	48 [bytes] [4]
n_{id}	The number of bits for an account ID	80 [bits] [32]
n_{time}	The number of bits for a time information ($time$)	48 [bits] [2]
n_{TX}	The number of bits for a transaction (TX)	-
n_{cer}	The number of bits for a certificate ($Cert$)	1500 [bytes] [19]

databases, rather each host provider is responsible for the users in charge. For efficient resource management, the provider can deposit all of the data to a cloud, by encrypting with the provider's encryption key if it needs. Current cloud services have these security options [18]. Lastly, the provider keeps the provider's secret key securely. The secret key is used for signing the user's public key information. Originally, public key certificates are issued by CAs under strict security, while in the proposed framework, each provider takes its role for the users. The provider is therefore, assumed to own a HSM (Hardware Security Module). It is a hardware that securely stores keys and computes digital signatures. Similar to the conventional requirements for financial industries and government agencies, it is assumed to satisfy the requirements of FIPS 140-3 level 3, defined by FIPS (Federal Information Processing Standard) [8].

4) Channel between the Providers: The communication between the trusted providers is based on the TLS communication. They confirm the trust by verifying the public key certificates for one another. Furthermore, the host provider's registration requirements reflect to the following provider's trust, when the providers share the identity proofing with the user's public key. Therefore, each provider checks the other provider's requirements, so that they can attain their desired security level.

4 Performance Analysis

This section presents a computational evaluation for the TS-L. As a multi-signature schemes, BLS signature [3,5] is implemented. Its signature size is 48 bytes, indicating the BLS signature attains smaller in size [35]. Under the proposed scheme, all of the data is signed and stored with the signature. Prioritizing the storage size of the provider to manage the information, the BLS signature is here adopted. The public key size is 96 bytes referred to [4]. SHA-256 is assumed for the hash function. The execution time is measured on Apple M1 CPU with

Table 2. storage size of the transaction database for a user

Frequency	**Data Size**	**Time-span**		
T_{TX}	n_{TX}	10 years	50 years	90 years
5 times per a day	128 [B]	4 [MB]	18 [MB]	32 [MB]
	1 [MB]	18 [GB]	91 [GB]	164 [GB]
once per a month	128 [B]	23 [KB]	115 [KB]	207 [KB]
	1 [MB]	120 [MB]	600 [MB]	1080 [MB]

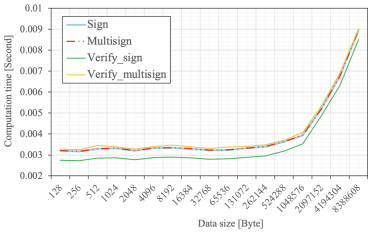

Fig. 7. Computation time for signing and verifying [second]

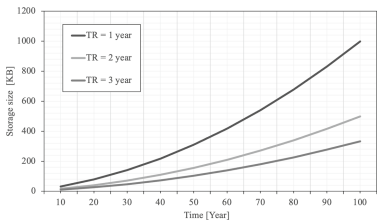

Fig. 8. Storage size of a user's public key database at time T [byte]

8 GB memory, and PBC library is used. The results are the average of the 100 times simulations. Table 1 summarizes the parameters for the model.

The user side's cost is considered as the signing time in uploading transactions (data items) or in requesting an inquiry. Figure 7 shows the computation time for them given a data size. Since it does not take much time to create an aggregate signature, the difference between **Sign** and **Multisign** is small. In executing a transaction, the transaction data size can be around 128 or 256 bytes, implying to take approximately 0.003 s. The result shows that the signing operation of each user is light for regular data size.

For the service provider, we analyze the verification time for signatures, the storage size of transactions (data items), and the storage size of users' public keys. Figure 7 shows the computation time for the verifying operation for a single-signature and for a multi-signature. Similar to the signing cost, the verification time is around 0.003 s and starts increasing from approximately 2 MB. For more frequent transaction operations, the verification time is simply increasing. However, the goal of the TS-L is to share KYC information, and it is assumed that there exist multiple providers. Consequently, the transaction verification will not be concentrated on one provider, resulting in a feasible verification time for a provider.

The storage size of the transaction database at time T for a user is defined as $S_{(TX,T)} = \frac{T}{T_{TX}} \cdot (n_{\mathrm{id}} + n_{\mathrm{time}} + n_{TX} + n_{\mathrm{sig}})$. Table 2 describes the examples of the storage size for a user. Even if frequent transactions are stored for a long time, the storage size is feasible.

The storage size of a user's public keys managed by the provider at time T is describes as $S_{(pk,T)} = \frac{T_{\mathrm{E}}}{T_{\mathrm{R}}} \cdot \left(\frac{T}{T_{\mathrm{E}}} \cdot (n_{\mathrm{id}} + n_{\mathrm{time}} + n_{pk} + 2n_{\mathrm{sig}} + n_{\mathrm{cer}}) + \left(\sum_{i=1}^{\frac{T}{T_{\mathrm{E}}}} i \right) \cdot (n_{\mathrm{cer}} + n_{\mathrm{sig}}) \right)$. The number of renewal time is simply determined by the average validity period of certificates for long-term signatures (T_E). At a renewal time, the storage size for a new certificate and a signature is added by $(n_{\mathrm{cer}} + n_{\mathrm{sig}})$ bits. Figure 8 describes the required size given a time. For example, when the user refreshes his/her key once per a year, it takes approximately 300 KB in 50 years. Even if it is 100 years, the size is around 1 MB. In the case of less frequent refreshing such as once per 2 years or 3 years, it requires much smaller. Note that as for the computation time for validating the certificates on the user's public key, the TSA needs to update certificates on long-term signature with signing by 10 years. Overall, the result shows that the public key updating mechanism is easily applicable to the providers.

5 Related Literature

Role-based access control using digital certificates was described in multiple works [7,29,30]. Digital certificates link to device IDs and they are used for controlling access to network resources. Comparing with them, the proposed scheme directly ties the digital certificate and the individuals' identity.

Regarding signed data, some literature proposed methods to deposit signed data in the cloud [17,28,32]. They assumed that the cloud is an untrustworthy entity, therefore the data owners check its integrity by verifying the digital signature on the retrieved data by themselves. In particular, [32] assumed financial and medical database, and adopts multi-signatures by multiple data owners. Since the proposed method assumes the trusted provider, the signature is used only to verify the validity of the data. As an extension, these methods will be incorporated if individuals desire to verify the integrity by themselves. In the other direction, under assuming a trusted verifier, the importance of determining database access policies based on signature verification was mentioned in [31], while any specific method was not provided.

Transaction signatures have been implemented in a wide variety of blockchain-based transaction schemes [21,22,27]. Particularly, Bitcoin [22] is a well-know payment scheme. While both the TS-L and the blockchain-based upload data after verifying the transaction signature, these systems do not assume a trusted central administrator of the system. Since the proposed scheme prioritizes the user authenticity tied to the physical person, over the other information security properties such as confidentiality and privacy, the trusted provider is assumed. For the other directions of blockchain, a permissioned scheme with trusted participants was suggested [26]. As for database resources, under the system, the database is shared among multiple network participants in an attempt to prevent a single point of failure. On the other hand, under the proposed scheme, the host provider who manages the user's public key information and issues certificates, is responsible for managing the databases, holding backup functions. If it is inefficient to prepare the database resources on-premise,

a cloud database can be chosen. The current cloud services can meet appropriate security standards, taking measures to prevent failures. For data privacy, the host provider can encrypt data and deposit it in the cloud. Regarding the integrity of the database, it is an extension of the proposed scheme to detect anomalies using a Merkle tree [20], as the blockchain scheme implements. However, the computational complexity needs to be examined. Overall, the proposed method focuses more on the message integrity of transmission channels in a network space, rather than the database integrity of start and end-points of the communications.

6 Conclusion

This paper proposes a comprehensive transaction scheme to reduce the KYC costs, by sharing customers' information among multiple providers. For further research, for example, improvements in detailed signature schemes to attain lighter computation complexity or additional verification features are expected. In addition, since the linkage between physical IDs and public keys is critical under this scheme, further analysis on that point is necessary.

References

1. World Wide Web Consortium, Web authentication: An API for accessing public key credentials level 2. https://www.w3.org/tr/webauthn/ (2021)
2. Adams, C., Cain, P., Pinkas, D., Zuccherato, R.: Rfc 3161: Internet x. 509 public key infrastructure time-stamp protocol (tsp) (2001)
3. Boneh, D., Drijvers, M., Neven, G.: Compact multi-signatures for smaller blockchains. In: Peyrin, T., Galbraith, S. (eds.) ASIACRYPT 2018. LNCS, vol. 11273, pp. 435–464. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-03329-3_15
4. Boneh, D., Gorbunov, S., Wahby, R.S., Wee, H., Zhang, Z.: BLS Signatures. Internet-Draft draft-irtf-cfrg-bls-signature-04, Internet Engineering Task Force (2020). https://datatracker.ietf.org/doc/html/draft-irtf-cfrg-bls-signature-04
5. Boneh, D., Lynn, B., Shacham, H.: Short signatures from the weil pairing. In: Boyd, C. (ed.) ASIACRYPT 2001. LNCS, vol. 2248, pp. 514–532. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-45682-1_30
6. Callegati, F., Cerroni, W., Ramilli, M.: Man-in-the-middle attack to the https protocol. IEEE Secur. Priv. **7**(1), 78–81 (2009)
7. Chadwick, D., Otenko, A., Ball, E.: Role-based access control with x. 509 attribute certificates. IEEE Internet Comput. **7**(2), 62–69 (2003)
8. Cooper, M.J., Schaffer, K.B., et al.: Security requirements for cryptographic modules (2019)
9. Delgado, O., Fúster Sabater, A., Sierra, J.: Analysis of new threats to online banking authentication schemes (2008)
10. DigiCert, I.: Certificate policy version 5.5 (2011)
11. DRAFT NIST: Special publication 800–63-3. Digital Identity Guidelines (2017)
12. Feng, H., Li, H., Pan, X., Zhao, Z.: A formal analysis of the FIDO UAF protocol. In: Proceedings of 28th Network and Distributed System Security Symposium (NDSS) (2021)

13. Fratantonio, Y., Qian, C., Chung, S.P., Lee, W.: Cloak and dagger: from two permissions to complete control of the UI feedback loop. In: 2017 IEEE Symposium on Security and Privacy (SP), pp. 1041–1057. IEEE (2017)
14. Hu, K., Zhang, Z.: Security analysis of an attractive online authentication standard: FIDO UAF protocol. China Commun. **13**(12), 189–198 (2016)
15. Huang, L.S., Moshchuk, A., Wang, H.J., Schecter, S., Jackson, C.: Clickjacking: Attacks and defenses. In: 21st {USENIX} Security Symposium ({USENIX} Security 12), pp. 413–428 (2012)
16. ISO: ISO 14533–2:2021 processes, data elements and documents in commerce, industry and administration - long term signature - part 2: Profiles for xml advanced electronic signatures (xades) (2021)
17. Juels, A., Kaliski Jr, B.S.: Pors: proofs of retrievability for large files. In: Proceedings of the 14th ACM Conference on Computer and Communications Security, pp. 584–597 (2007)
18. Mathew, S.: Overview of amazon web services: AWS whitepaper. Amazon Web Services, Seattle, WA, USA, White Paper (2020)
19. McGrew, D., Pritikin. M.: The compressed x.509 certificate format draft-pritikin-comp-x509-00 (2010)
20. Merkle, R.C.: A certified digital signature. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 218–238. Springer, New York (1990). https://doi.org/10.1007/0-387-34805-0_21
21. Miller, A., Juels, A., Shi, E., Parno, B., Katz, J.: Permacoin: repurposing bitcoin work for data preservation. In: 2014 IEEE Symposium on Security and Privacy, pp. 475–490. IEEE (2014)
22. Nakamoto, S.: Bitcoin: a peer-to-peer electronic cash system. Decentralized Bus. Rev. 21260 (2008)
23. Niemietz, M., Schwenk, J.: Ui redressing attacks on android devices. Black Hat Abu Dhabi (2012)
24. Reuters, T.: Thomson reuters 2016 know your customer surveys reveal escalating costs and complexity. Thomson Reuters **9**, 06–20 (2016)
25. Rose, S., Borchert, O., Mitchell, S., Connelly, S.: Zero trust architecture (2nd draft). Technical report, National Institute of Standards and Technology (2020)
26. Sajana, P., Sindhu, M., Sethumadhavan, M.: On blockchain applications: hyperledger fabric and ethereum. Int. J. Pure Appl. Math. **118**(18), 2965–2970 (2018)
27. Sasson, E.B., et al.: Zerocash: decentralized anonymous payments from bitcoin. In: 2014 IEEE Symposium on Security and Privacy, pp. 459–474. IEEE (2014)
28. Shacham, H., Waters, B.: Compact proofs of retrievability. In: Pieprzyk, J. (ed.) ASIACRYPT 2008. LNCS, vol. 5350, pp. 90–107. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-89255-7_7
29. Silva, E.F., Muchaluat-Saade, D.C., Fernandes, N.C.: Across: a generic framework for attribute-based access control with distributed policies for virtual organizations. Futur. Gener. Comput. Syst. **78**, 1–17 (2018)
30. Thompson, M., Johnston, W., Mudumbai, S., Hoo, G., Jackson, K., Essiari, A.: Certificate-basedaccesscontrol forwidelydistributedr esources. In: Proceedings 8th UsenixSecurity Symposium (1999)
31. Vegh, L., Miclea, L.: Access control in cyber-physical systems using steganography and digital signatures. In: 2015 IEEE International Conference on Industrial Technology (ICIT), pp. 1504–1509. IEEE (2015)
32. Wang, B., Li, H., Liu, X., Li, F., Li, X.: Efficient public verification on the integrity of multi-owner data in the cloud. J. Commun. Netw. **16**(6), 592–599 (2014)

33. Wazid, M., Zeadally, S., Das, A.K.: Mobile banking: evolution and threats: malware threats and security solutions. IEEE Consum. Electron. Mag. **8**(2), 56–60 (2019)
34. Weigold, T., Kramp, T., Hermann, R., Höring, F., Buhler, P., Baentsch, M.: The zurich trusted information channel – an efficient defence against man-in-the-middle and malicious software attacks. In: Lipp, P., Sadeghi, A.-R., Koch, K.-M. (eds.) Trust 2008. LNCS, vol. 4968, pp. 75–91. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-68979-9_6
35. Xiao, Y., Zhang, P., Liu, Y.: Secure and efficient multi-signature schemes for fabric: an enterprise blockchain platform. IEEE Trans. Inf. Forensics Secur. **16**, 1782–1794 (2020)