Combining anonymity and KYC in cryptocurrency protocols

Pavel Kravchenko, Andras Kristof
pavel@distributedlab.com, andras@tembusu.sg

Abstract

During last 6 years several decentralized applications based on blockchain technology for financial industry were released. They have got some adoption level among users who are concerned about privacy of their spendings, clumsiness of financial institutions and uncontrolled inflation. But several issues prevent such systems to become mainstream - lack of compliance mechanisms, volatility that may be caused by anonymous malicious players, complexity of enforcing of KYC policies, low processing speed, obscurity for the non-technical users etc. We believe that blockchain technology should be supplemented by mechanisms that solve aforementioned issues. In this paper we describe the architecture and technology for "ideal" infrastructure that has built-in compliance mechanisms while ensuring privacy of user spending, sufficient speed and integrity of transaction history.

1 Introduction

The first viable electronic money system that doesn't have centralized authority was Bitcoin [1]. It solved the very important problem of achieving consensus and protecting from double-spending attacks [2] in a fully distributed network with anonymous participants. Bitcoin gained attention from people who were looking for an analogue of gold, but in electronic form - the main features of gold could be very attractive for certain conditions - amount of gold on the Earth is limited, it is not feasible to create more gold in the laboratory, nobody controls the gold itself, it is hard to mine gold. Having these features in an electronic form of gold allows building a financial system that is transparent and reliable - nobody can just print more money. Blockchain technology, which is the backbone of Bitcoin, stores all transactions in a public, distributed ledger. This makes financial history non-alterable, and makes it very difficult for a malicious actor to cheat. Strong cryptographic mechanisms like hashes and digital signatures are used to ensure integrity of the blockchain and protect transactions in the public network. Achievement of global consensus is done by using "mining" - this is a mechanism that protects the system from the Sybil attack [3]. Open-source codebase allows everybody verify that the

system doesn't have backdoors and free from malware. In addition, there is no central authority to take profits from the transactions. All these inventions make Bitcoin very cheap for the end user - it is not a company, but just a protocol to exchange value electronically.

However, Bitcoin has several drawbacks that prevented it from wide adoption. Price of bitcoin is very volatile, Bitcoin doesn't allow instant payments - full transaction confirmation takes some time (10+ minutes), which is unacceptable for offline businesses. Further, Bitcoin, as a relatively anonymous system could be used as a money laundering tool that puts law-abiding users at risk that government will prohibit it from usage. From the other side since all transactions in the blockchain are publicly viewable it is possible to trace transactions and identify users for anyone who has appropriate tools and processing power [4]. And finally, due to the incentive model properties that are used in Bitcoin, mining is rather an endless arms race that consumes hundreds of thousands dollars for electricity every day [5].

There are some reasons why Bitcoin was designed in this way. It tried to create a system where trust is distributed over the network. We believe in the importance of this feature, but the existing financial ecosystem needs technology that will eliminate weaknesses while keeping its strong sides. One of them is the ability to work with any kind of currency or asset. Another problem is the lack of tangible value. Government-issued currencies are backed by their respective issuers. The backing of a known and trusted entity gives a currency stability.

One attempt of creating a platform that is based on Bitcoin principles (transparency, security, open-source, decentralization) but at the same time is suitable for needs of existing financial ecosystem is a ledger-based systems (Ripple [6], Stellar [7], OpenTransactions [8], HyperLedger [9]). Some of them have consensus based mechanism for transaction processing, other use local consensus for private ledgers and rely on other platform to achieve global consensus, support of multiple currencies, trust relationships etc. These systems may replace existing transaction processing engines, such as Swift, Visa, Mastercard etc in the future due to open-source model and low cost of operation. But still, a lot of issues have to be solved - protection of users' privacy, clear mechanisms to enforce KYC/AML, risk management, and customer protection in case of fraud.

The goal of current research is implementation of the best aspects of cutting-edge technology while satisfying variety of regulatory requirements to KYC and AML, preserving privacy of individual users and making possible to implement sufficient level of customer protection.

2 Requirements and assumptions

Below we describe the requirements that are needed to ensure privacy, usability and compliance at the same time. We need to note that the system should be able to support hundreds of millions of users.

1. Privacy of payments - it should not be possible for external observer, merchant or bank to trace spendings of the the user, even if his/her payment address is publicly known.
2. AML procedures should be possible to be followed - in case of suspicious transactions designated auditor should be able to identify bad actors sender and recipient of funds.
3. KYC policies must be satisfied for users from any jurisdiction - requirements may vary from country to country and platform has provide an ability to enforce them.
4. Security - user should not trust anyone to store their private keys.
5. Customer protection - spending of stolen funds should be not worth doing it, identity theft should not be a disaster.

3 Architecture of the system

General principles

1. Our system is based on a ledger-based transaction engine, which allows to use any kind of currency or asset, perform a distributed exchange while having fast transactions confirmation.
2. Our system doesn't have its own cryptocurrency - therefore it is completely free of dependencies from some institution and has no risk of currency volatility. So basically it is just a protocol.
3. We introduce concept of *group* - set of users that are organized by some principle - for example citizens of a country, or employees of a company etc.
4. Each group has a group manager (could be an organization(s) or algorithm) which is in charge of establishing rules of the group, adding/removing members, making an audit of their transactions (if needed).
5. There could be as many groups as needed and each of them will have own rules that its members have to comply with. User can be a member of many groups.
6. Each member of the group has an identity, known only to the group manager (could be a pseudonym).
7. To initiate a transaction, user has to produce and attach to the transaction proof-of-identity (that can be verified by anyone) that proves that user really belongs to specified group, but doesn't reveal the identity. Miners (processing nodes) will only verify transactions that contain valid proof of identity.
8. User's wallet is still a key pair (so user is able to generate any wallet) and identity is also a key pair. Only user knows private keys.

So the main idea here is the so called proof-of-identity (POI). It allows only legitimate members of groups to transact, and it makes it possible for the group manager to reveal identities of both parties of the transaction in case of audit.

General Terms

Identity - is a public key that identifies legitimate user of a group. Each user has a corresponding private key (*identity key*). Only the group manager may reveal the relationship between registration data and public key. Group managers maintains list of all registered users and keeps record of revoked identities. Identity is obtained by the user during the registration process that complies with KYC requirements of the group.

Proof-of-identity (POI) is a cryptographic proof (block of data) that some user knows a private key that corresponds to an approved identity and cryptographically tied to a particular transaction.

The idea is that each member of some group is able to produce a proof-of-identify (just a piece of data) and present it to anybody (for example to the processing node). Proof of identity can be added to the transaction and stored in the blockchain/ledger. It will not reveal any information about the user for third party observers except 1) group user belongs to 2) the fact that user initiated the transaction.

All accounts in the ledger are anonymous, but each transaction carries proof of identity that may be audited later by group manager. It is like having a credit card (ability to spend money) and an ID (proof that this is your card and you are 18+).

Use cases

1. Government of CountryOne require all users to provide passport data, proof of residence, source of wealth to be able to open a account in crypto assets. All transaction about 10.000 (denominated in national currency) are subject to an audit. There are also daily/monthly limits for transaction size. Government outsources the registration process to approved financial institutions. Now each financial institution that operates in CountryOne establishes a group of its customers and requires needed documents to be provided. Financial institution is responsible for storing personal data and disclosure of details of particular transaction on government's request. So it keeps all the documents users provide safe and able to reveal the identity of sender/recipient of the particular transaction. So when the customer pays online to a merchant (or makes a trade with a customer of another organization) his/her privacy is still protected. Policies of CountryOne don't influence the rest of the users, who are citizens of other countries. Anyway, only approved users can transact in the system (it is still a question to consider which groups will be "compatible" with each other).

2. Government of CountryTwo has the same requirements to KYC but wants to register citizens itself, because all of them already have e-ID card. It means that there is only one group in CountryTwo. After the registration user is able to prove to financial organization that he was approved by the government to open the account (i.e. establish the trust to a gateway). Then user is able to transact and financial institution itself (as well as merchants or neighbours) will not be able to identify users by looking into the public ledger.

3. Private membership organization promises their customers full anonymity if they pass KYC procedure. In this case single group is established, amount of members may be fixed (users may or may not be revoked), but group rules are set in a way that doesn't allow organization to reveal identities of users by looking at proof of identity.

4. CountryThree is ruled by the principle that nobody alone is able to reveal identities behind a transaction. So it establishes a group where power of group manager (i.e. master key of a group) is shared between several parties - court system, ministry of internal affairs, ministry of taxation - they all need to cooperate to register/revoke users or reveal who produced proof of identity.

5. TransparentCompany is so liberate that is required to ask permission of its employees to perform identity disclosure request. Two options are available (and will be covered by following papers) - 1) only cooperation of majority of group members is able to reveal identity 2) TransparentCompany needs to send request to the network to obtain proof of identity that corresponds to the suspicious transaction - in this case proof of identity is not added to the blockchain directly, but stored in a distributed way using maidsafe approach.

6. It is also possible to create identities with different strengths that will be represented by different groups. For example the strongest A-level identities are created with a government ID, in a customer centre, in front of a clerk. One step weaker would be still with a government ID, but signed up online, without the presence of a trusted third party. Even weaker would be an identity created without using a government ID, but vouched for by 3 existing, A-level users. This system of multiple strength identities will necessary, because many people in our target audience (the unbanked) does not have a government ID. The multiple level identification system will be described in a separate paper.

3.1 Requirements to POI

Here we describe formal requirements to POI that has to be met in the real system.

1. Only the group manager is able to change the list of valid identities.
2. Provided POI shouldn't reveal identity used (e.g. some valid identity is used).
3. Identity owner should produce different (unlinkable) POI for each transaction performed.

4. Processing server should be able to verify the validity of POI themselves (without asking the group manager).
5. Group manager should be able to reveal identity behind each POI if this is stated in the rules of the group.
6. Group manager should not be able to create valid POI (in order to impersonate the user).
7. Only one identity should be issued for each user.
8. There must be a way to revoke a valid identity.

3.2 PKI as a method to maintain list of valid identities

List of valid identities may be signed by the group manager and published in the open database. This database can contain federated address of the user that will be used as an "official" receiving address. In this case government plays role of Certification Authority (CA) because it needs to maintain an integrity of this database and prevent unauthorized changes. This data can be stored in a blockchain/ledger to make it more robust and ensure community control over it. Identities of the members that have to be deleted from the group should be maintained in additional list - identical to CRL in PKI. When some member has to be deleted from the group, the list of revoked members are updated. Role of Registration Authority (RA) is represented by organizations that are entitled to perform registration of users.

3.3 Usability considerations

User has to keep only a few keys - master key from his wallet and identity key from groups that he joined - and these keys may be stored on a mobile device. Addresses for incoming payments will be generated from the master public key (HD wallet) and this process is managed by the application itself.

3.4 Consensus based on agreement of trusted parties

For our ecosystem we've chosen the ledger-based protocol [6, 7] that implements a different approach to financial transaction than Bitcoin. It inherits most of the features of cryptocurrencies, but unlike Bitcoin, it is ledger based and consensus is reached between servers that are publicly available. We selected it because it is fast, doesn't need mining and can work with any kind of currency. Consensus about legitimate transactions is reached automatically only between servers that know each other, but not obligatory trust. Any server can introduce a transaction to the network and these transactions are broadcasted between all active servers. Each server receives proposals from other trusted servers on the network about the transaction that

should be validated. List of trusted servers that provided public info about themselves is maintained by each server. Proposals from servers that are anonymous are ignored. This feature is needed for our system because we need only legitimate organizations to process transactions to prevent possibility from 51% attack from anonymous players. From the other side, decentralization of decision making will lead to the transparency. Consensus is reached when 80% of servers agree on a particular transaction set and transactions are applied to the ledger. Then the entire process is started again, and each round takes only a few seconds.

3.5 Ledger based storage

For our purposes we need a multi-currency solution. Instead of blockchain, we use a distributed ledger. It is a database that is shared with all the servers in the network. It stores information about all accounts. In addition to balances in any kind of currency, the ledger holds information about transactions, additional account information (such as messaging key), offers to buy or sell currencies and assets, creating the distributed exchange. Each transaction that is applied to the ledger can potentially change balances of several accounts atomically - this is happening when payment is initiated in one currency (for example USD) but has to arrive in the other one (for example SGD). The protocol automatically seeks for offers from market makers to exchange USD to SGD and then takes the offer with the best exchange rate. Then balances of the sender, recipient and market maker are changed simultaneously. If no offer to exchange USD to SGD is found, then USD could be exchange to EUR and afterwards, EUR to SGD - in this case balances of four people are affected by the transaction.

3.6 Greater Anonymity

With existing approaches to KYC in cryptocurrencies we may come to the point where only whitelisted addresses can transact. It is obvious then that financial institutions will get access to user's private data even faster than before - because blockchain is transparent. We suggest to use stealth addresses technique [10]. It means that each user has an account that is given to the public, but when some payment is initiated, a new address is generated and money goes there. At the same time the user's wallet can "find" the address where money went and later spend it. Addresses never used twice, if some change is left in one of the addresses, it is sent to the other one time account. In this case nobody can trace the wealth of the user, except the group manager.

3.7 Decentralised trust

One of the main requirements of the system is that users store their private keys themselves and don't trust anybody else to do it. That's why registration on POI server is done in the same way as in PKI - a key pair is generated on the user side and then public key is sent to certification in a protected way. The private key of the wallet is only known to the user.

From the transaction processing perspective trust is distributed since all processing nodes all known and their decisions are publicly verifiable (i.e. these servers are responsible for their actions).

Our intention is to maintain a list of verified identities in the ledger as well to ensure its integrity and immutability as well as the provable version control. Later on we will release the approach how to identify all inquires of the government to reveal the identity that is associated with some transaction.

3.8 Customer protection

Since each transaction contains POI, even if money (private key from the wallet) is stolen, it will be possible to trace it. There are still edge cases where 1) identity was stolen as well (it means that nobody can identify real hacker) 2) some anonymous goods are bought (for example Bitcoins) 3) two-factor authentication method was hacked. But If the user just lost the identity key she will a request to POI server to generate a new identity - previous identity will be revoked and all transactions that use it after revocation will be rejected.

There is a possibility to recover funds in case if user's private keys got destroyed and no recovery mechanism was used. Since group manager can identify transactions of the user (not only outgoing, but also incoming if "invoice" technique is used) it will be possible to see which addresses contain the money of the user. It is possible [11] to freeze funds forever if the user occasionally destroyed private keys and re-issue them again, since we are using gateway model. This decision will be made by each group manager and gateway themselves.

4 Solutions and technical requirements of POI

POI can be implemented using the group signatures technique. A Group signature scheme is a method for allowing a member of a group to anonymously sign a message on behalf of the group. The concept was first introduced by David Chaum and Eugene van Heyst in 1991 [12]. For example, a group signature scheme could be used by an employee of a large company where it is sufficient for a verifier to know a message was signed by an employee, but not which particular employee signed it.

Essential to a group signature scheme is a group manager, who is in charge of adding group members and has the ability to reveal the original signer in the event of disputes. All group signature schemes should follow these basic requirements:

- *Soundness and Completeness* - valid signatures by group members always verify correctly, and invalid signatures always fail verification.
- *Unforgeability* - only members of the group can create valid group signatures.
- *Anonymity* - given a message and its signature, the identity of the individual signer cannot be determined without the group manager's secret key.
- *Traceability* - given any valid signature, the group manager should be able to trace which user issued the signature.
- *Unlinkability* - given two messages and their signatures, we cannot tell if the signatures were from the same signer or not.
- *Exculpability* - even if all other group members (and the manager) collude, they cannot forge a signature for a non-participating group member.

There were a lot a group signature schemes introduced [13], but we need to select one that conforms to our very specific set of requirements:

1. Group key has to be updated without updating keys of the users.
2. Signature size should be independent from the group size (because a group could contain millions of users).
3. Signature itself should be short (because it will be stored in the blockchain).
4. Signature has to be generated and verified sufficiently fast.
5. If an identity was removed from the group, the user should not be able to generate a valid POI anymore.

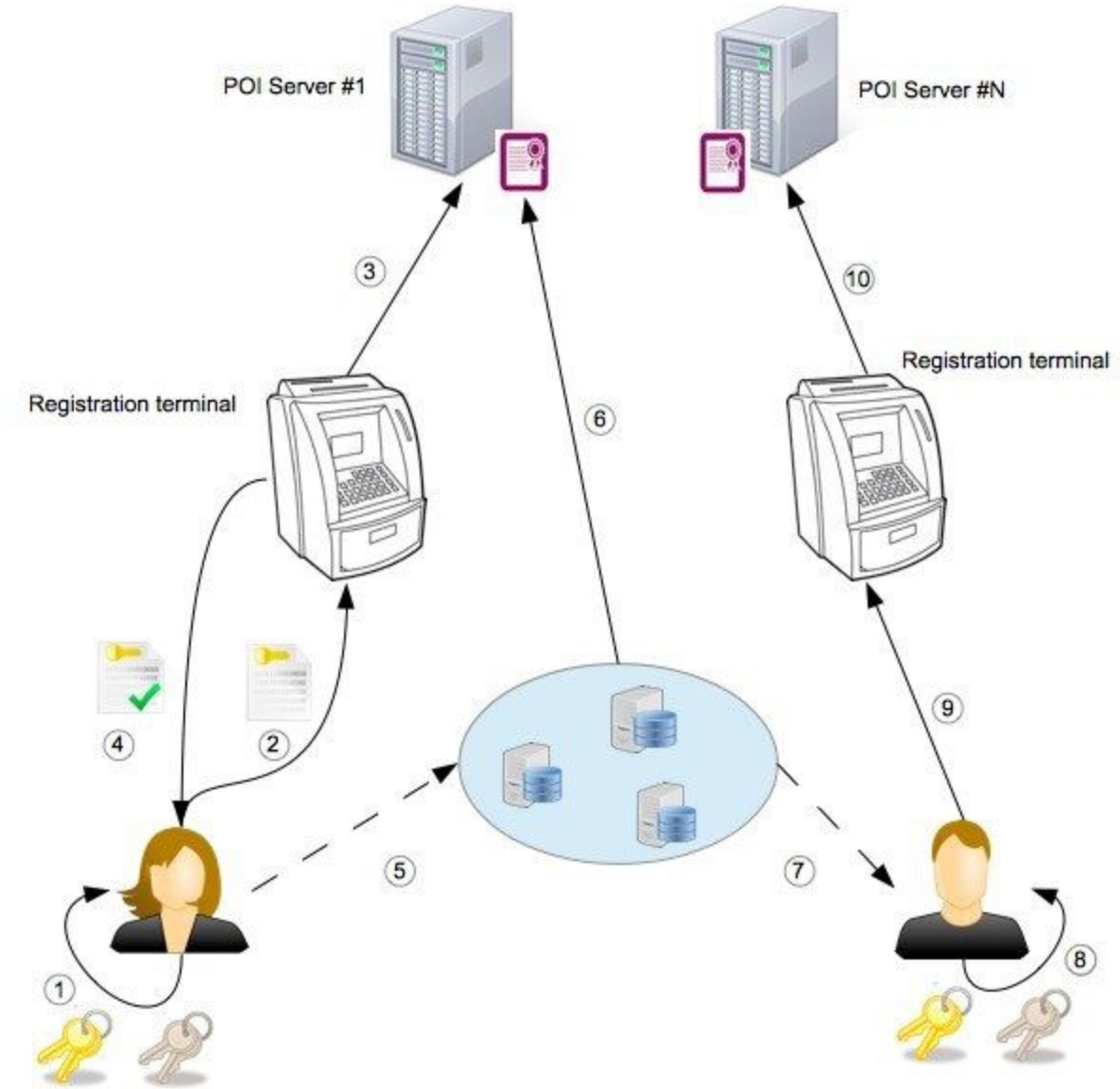
For the MVP of the system we selected The Bichsel-Camenisch-Neven-Smart-Warinschi group signature scheme [14] that can produce short signatures and has fixed size group key. Group manager is able to “open” proof of identities, add/revoke users. More group signature schemes will be supported in the future to satisfy requirements of usecases 1-5.

5 Example of registration process

General process of registration is quite straightforward (see pic. 1):

1. Alice selects the group she wants to join, generates an identity key and identity alongside with cryptocurrency wallet.
2. Alice provides ID documents alongside with her identity to the group manager.

3. Group manager verifies ID, updates group public key (if required by group signature scheme that is used) adds username and federated address to the public database, while storing identity of Alice in a secure way.
4. Group manager sends Alice confirmation and now she is able to generate her group private key.
5. Now Alice pays Bob (who is not a member yet), attaching POI to the transaction.
6. Processing node verifies that POI of the **Alice** is valid and approves transaction.
7. Now Bob has funds, but cannot spend them because he doesn't have identity key to generate POI.



Pic. 1 - Process of user's registration

Important to note that only sender's POI may be verified because even if money went to an account of non-approved person, he will not be able to spend it since he cannot create valid POI for the transaction. It means that users can generate addresses themselves without exposing their private keys to some server and even start accepting payments **before** registering on POI server. At the same time, some group managers may require identification of recipient as well. In this case "invoice" technique will be used (something similar to BIP 0070 [15]) - recipient will first provide to the sender a proof of identity that corresponds to prepared transaction plus proof that she controls private key from the receiving address. In this case final transaction will include two POIs. Each group manager will be able to set limits for transactions that require double POI - this parameter could be included as a system parameter of the group.

Alice needs to know only the username of Bob to send the money. Username is connected to Bob's public key that is actually used to generate transaction. If Alice interacts with Bob for the first time she should ask the federation server about correspondence of username and public key. But public key of Bob is used to generate one-time address where money will actually be sent.

6 Description of group signature implementation

We selected the BCNSW scheme that can be extended to add verifier-local revocation. The BCNSW scheme allows for the shortest known signature size as well as comparably low computation time while keeping a strong security level. In the following we will describe algorithms of the BCNSW scheme [14].

The BCNSW group signature scheme is a dynamic scheme with verifiable opening employing a user PKI (connected with an unforgeable digital signature scheme $\Sigma = (Kg, Sign, Vrfy)$. It has a single security parameter η N and uses bilinear groups $G_1 = (g_1)$, $G_2 = (g_2)$, and G_T of prime order Q with $|Q|=\eta$ and a bilinear map $e: G_1 \times G_2 \rightarrow G_T$. Additionally, two hash functions $Hash_1, Hash_2 : \{0, 1\} \rightarrow Z_Q$ are defined.

$GSetup(1^\eta)$: The Group Manager chooses random $\alpha, \beta \leftarrow Z_q$, and computes $x\tilde{} \leftarrow g\tilde{}^{\alpha}$ and $y\tilde{} \leftarrow g\tilde{}^{\beta}$. It then sets the group public key of the scheme to $gpk \leftarrow (x\tilde{}, y\tilde{})$ and the group secret key to $gmsk \leftarrow (\alpha, \beta)$.

$PKIJoin(i, 1\eta)$: The CA certifies public keys of a digital signature scheme as defined in Section 2. The user generates $(upk[i], usk[i]) \leftarrow DSKeyGen(1^\eta)$ and sends $upk[i]$ to the CA for certification.

$GJoin = (GJoinM(i, upk[i], gmsk), GJoinU(i, usk[i], gpk))$: When a user i wants to join the group, she must have already run the PKIJoin algorithm. Then she runs the following protocol

with the Group Manager. We assume that this protocol is run over secure channels and, for simplicity, that the parties only run one instance at a time. We also assume that if a verification for a party fails, the party informs the other party about the failure and the protocol is aborted.

The GroupManager chooses a random $\kappa \leftarrow Z_q$, computes $t \leftarrow G(\kappa)$, and sends t to the user.
The user i chooses $\tau \leftarrow Z_q$, computes $s \leftarrow g^{\tau}$, $\tilde{r} \leftarrow \tilde{x}^{\tau}$, $k \leftarrow \hat{e}(g,\tilde{r})$, as well as $\sigma \leftarrow DSSign(usk[i], k)$, sends $(s, \tilde{r}, \sigma)$ to the Group Manager and executes $FPK\{(\tau): s=g^{\tau} \quad \tilde{r}=\tilde{x}^{\tau}\}$ with the GroupManager.
The GroupManager uses $DSVerify(upk[i], \hat{e}(g,\tilde{r}), \sigma)$ to verify the signature. If it verifies correctly he computes $z \leftarrow s \cdot g^{\kappa}$ and $\tilde{w} \leftarrow \tilde{r} \cdot \tilde{x}^{\kappa}$, stores $(\tilde{w}, \tilde{r}, \kappa, \sigma)$ in $reg[i]$, chooses $\rho \leftarrow Z_q$, computes $a \leftarrow g^{\rho}$, $b \leftarrow a^{\beta}$, and $c \leftarrow a^{\alpha} \cdot z^{\rho\alpha\beta}$, and sends (a, b, c, κ) to the user. In addition, he executes $FPK\{(\alpha,\beta,\rho,\gamma): c=a^{\alpha}z^{\gamma} \quad a=g^{\rho} \quad \tilde{x}=\tilde{g}^{\alpha} \quad \tilde{y}=\tilde{g}^{\beta} \quad 1=b^{\alpha}/g^{\gamma}\}$ with her, where $\gamma = \rho\alpha\beta$. Note that this proof allows the user to verify that $\alpha, \beta \neq 0$.
The user computes $\xi \leftarrow \tau + \kappa \bmod q$, and checks whether $t = G(\kappa)$. She also verifies $\hat{e}(a, \tilde{y}) = \hat{e}(b, \tilde{g})$ and, if the verification is successful, stores the entry $gsk[i] \leftarrow (\xi, (a, b, c))$.

Remarks: The value of ω stored in reg[i] allows the Opener to identify a user within the group signature scheme. In addition, the Opener can provably attribute this ω to $k = \hat{e}(g, \tilde{r})$. Consequently, a group signature can be provably attributed to k. By the unforgeability of the external signature scheme, the signature on k allows to attribute a group signature to a user public key upk[i]. Furthermore, the FPK protocol that the Group Manager and the user execute in Step 3 of the protocol indeed proves that c was computed correctly w.r.t. a, b, $\tilde{x}$, and $\tilde{y}$. To this end, note that because of $\hat{e}(a, \tilde{y}) = \hat{e}(b, \tilde{g})$, we know that b = aβ and thus b = gβρ. Subsequently, from $1 = b^{\alpha}/g^{\gamma}$ we can conclude that γ = ραβ and hence that c was computed correctly by the Group Manager.

$GSign(gsk[i], m)$: Let a user i with signing key $gsk[i] = (\xi, (a, b, c))$ sign the message m. She first re-randomizes the signature by choosing $\zeta \leftarrow Z_q$ and computing $d \leftarrow a^{\zeta}, e \leftarrow b^{\zeta}$, and $f \leftarrow c^{\zeta}$, and then computes the SPK

$$\Sigma \leftarrow SPK\{(\xi) : \frac{\hat{e}(f,\tilde{g})}{\hat{e}(d,\tilde{x})}\hat{e}(e,\tilde{x})^{\xi}\}(m)$$

proving that she knows the "message" for which (d, e, f) is a valid CL-signature. Finally, she outputs $\sigma \leftarrow (d, e, f, \Sigma) \quad G^3_1 \times Z^2_q$ as the group signature on m.

$GVerify(gpk, m, \sigma)$: To verify a signature $\sigma=(d, e, f, \Sigma)$ on the message m, the verifier first checks that $\hat{e}(d, \tilde{y}) = \hat{e}(e, \tilde{g})$, where $\tilde{g}, \tilde{y}$ are retrieved from *gpk*. Secondly, the verifier checks that the proof Σ is valid. If either of the checks fail, output 0; otherwise output 1.

GOpen(gmsk, m, σ, reg): Given signature *σ=(d, e, f, Σ)* on m, the GroupManager verifies the signature using *GVerify*. Then, for all entries *reg[i] = (w̃ᵢ, r̃ᵢ, κᵢ, σᵢ)* he checks whether *e^(f, g̃) = e^(d, x̃) · e^(e, w̃ᵢ)* holds. For the $\tilde{w}_i$ where the equation holds, the Group Manager retrieves κ_i and σ $\bar{i}$, computes $k_i \leftarrow e\hat{}(g, r_i)$ and the SPK

$$\Pi \leftarrow SPK\{(\tilde{w}_i, \kappa_i) : \frac{\hat{e}(f, \tilde{g})}{\hat{e}(d, \tilde{x})} = \hat{e}(e, \tilde{w}_i) \wedge k_i = \frac{\hat{e}(g, \tilde{w}_i)}{\hat{e}(g, \tilde{x})^{\kappa_i}}\} \ ,$$

and outputs *(i, π = (kᵢ, σᵢ, Π))*. Note that the opening operation is linear in the number of users in the system, but we consider this reasonable as in most practical applications opening is a rather exceptional operation performed by a resourceful Group Manager.

The BCNSW group signature scheme is pretty efficient - signature cost is 1 operation in G_T and 3 operations in G_1, verification cost is 2 operations in P^2, 1 in G_1^2 and 1 in G_1. The size of the signature is 3 elements of G_1 and 2 in Z_q.

7 Conclusion

In this paper we tried to solve the issue how to build a platform where KYC requirements of different jurisdictions can be combined with privacy of individuals. We applied the idea of splitting users to groups and use group signature as a technique that allows signing a message on behalf of the group. We made the system coin-free by design.

8 References

1. https://bitcoin.org/bitcoin.pdf
2. en.wikipedia.org/wiki/Double-spending
3. en.wikipedia.org/wiki/Sybil_attack
4. https://eprint.iacr.org/2013/782.pdf
5. https://blockchain.info/stats
6. https://ripple.com/
7. https://www.stellar.org/
8. http://opentransactions.org/wiki/index.php/Main_Page
9. http://hyperledger.com/
10. https://sx.dyne.org/stealth.html
11. https://wiki.ripple.com/Freeze
12. http://p2sh.net/file/Group%20signatures.pdf

13. https://www.bsi.bund.de/SharedDocs/Downloads/EN/BSI/Publications/Studies/GruPA/GruPA.pdf
14. http://paedi.biche.ch/work/pub/BCNSW2010-Get_Shorty_via_Group_Signatures_without_Encryption.pdf
15. https://github.com/bitcoin/bips/blob/master/bip-0070.mediawiki