League of Identity

A cryptographic threshold network implementing distributed identity-based encryption and authentication from Google and other providers, with applications to

Anonymous Identity-Based Web3 Payments

Vincenzo Iovino and AZKR Team

September 2024 - ***Draft***

Abstract

Identity-based encryption and signatures (IBE/IBS) were introduced by Adi Shamir in 1984 as a method to simplify certificate management in email systems. Despite extensive research on the subject, IBE has never been adopted in real-world applications. The main challenges lie in the difficulty of establishing concrete IBE infrastructures for identity management and the absence of unified identity standards.

In this work, we address this gap by proposing and implementing the League of Identity (LoI), an IBE/IBS system and tooling that leverages existing real-world identity providers.

With LoI, users can obtain a cryptographic token linked to their Gmail or Facebook account, phone number (linked to their Google account), social security number (via their digital identity card), Ethereum account, and more. These tokens can be acquired through a practical Single Sign-On (SSO) process for Google, Facebook, and other social providers, or by signing a document with a digital ID card or an Ethereum wallet.

The token can be used to sign messages or transactions on blockchains, or to decrypt secret messages.

As an application of LoI, we propose the first **anonymous identity-based payment system for web3**, allowing crypto users to send crypto assets to non-crypto users, an advancement of independent interest.

1 Introduction

Background. Identity-based encryption and signatures (IBE/IBS) were proposed by Adi Shamir in 1984 [1] as a way to simplify certificate management in email systems. Despite extensive research on this topic, IBE has been never been used in the "real world". This is largely due to the difficulty of creating infrastructures for identity management and the absence of unified identity standards. To make IBE/IBS practical we propose the following problem statement for which we show practical solutions along with a proof of concept.

Problem statement. We wish to implement a system that satisfies the following properties. A user should be able to use his existing Google (or Facebook, Twitter, etc.) account to engage in other distributed systems, e.g. a blockchain or a Decentralized Autonomous Organization (DAO). Moreover, we would also like to add identity-based encryption functionalities to these systems: a user Alice should be able to encrypt a message for the email address bob@gmail.com without interacting with Bob. Bob in turn should be able to decrypt a ciphertext associated with his email address just by using his ability to log into Google.

Why do we need cryptography? Consider the following scenario. We want to allow people to use a Decentralized Autonomous Organization (DAO) only if the users own a valid Gmail address belonging to the domain of some organization. A trivial solution is to have the user to prove to a "gateway" that he is able to log into Google and then the gateway can authorize the user to sign transactions on the blockchain. For example, the gateway can provide a valid public/secret key pair to the authorized user, and add the public key to the DAO. In this case the gateway should be fully trusted, as it can authorize any user to participate in the DAO and can associate any user to any Gmail address, even if the address is not owned by the user. Moreover, it is not clear here how to efficiently enable identity-based encryption which is one of our main goals.

Our model To solve the issue we envision the following model and we will later show how to implement it.

We assume that there is a set of n nodes who share a master secret key MSK, without actually knowing MSK. Each node $i \in [n]$ keeps a share MSK_i of MSK. If some threshold number t of nodes (e.g., the majority) were to collude, they could reconstruct MSK. The assumption, however, is that most nodes act correctly and do not collude. Corresponding to MSK there is a master public key MPK that is a public parameter of the system.

A user Bob with email bob@gmail.com can log into Google and obtain an OAuth 2.0 access token. This is a well-defined string that is associated with the address bob@gmail.com and which can be verified using public Google APIs. The user Bob then sends his access token to a subset S of the nodes of cardinality

t. Each node $i \in S$ replies with a share $Token^i_{Bob}$ and from the t shares Bob can reconstruct a token $Token_{Bob}$. Such token can be used in two ways.

- For encryption/decryption. A user Alice can encrypt a message m using just the master public key MPK and the email address bob@gmail.com to produce a ciphertext CT. Only Bob, having $Token_{Bob}$ can decrypt CT and get m.
- Authentication and signatures. The token can be used to sign a message m in the following sense. The signature algorithm takes as input the token $Token_{Bob}$ associated to the email bob@gmail.com, the master public key MPK and a message m (e.g., the transaction to sign) and produces a signature σ. This signature is a well-defined string that can be transmitted with a transaction. The verification algorithm takes as input the master public key MPK, the email bob@gmail.com and the signature σ and outputs 1 or 0 to denote acceptance or rejection of the signature. The signature should satisfy the usual properties of digital signatures adapted to this context, namely completeness and unforgeability (e.g., hardness of computing signatures of messages for which no signature was observed before). The token can be also used to authenticate interactively in a way that a user Alice is able to verify the identity of the user Bob but Alice is unable to reuse the transcript of the protocol to impersonate Bob (that is, Bob can convince Alice to be the actual Bob but after this interaction Alice is unable to convince Charlie of the false claim "I am Bob"). Furthermore, we envision that the verification of the signature should be efficient for web3 applications. For instance, it is efficient to verify signatures of digital identity cards off-chain but verification on Ethereum of such signatures consumes too much GAS to be practical.

The previous properties enable very powerful applications. It can allow users with access tokens from Google and similar services to participate in a blockchain or DAO. Moreover, one can extract useful information from access tokens, for example pictures of the users. Such pictures and additional info can be profitably used in the blockchain. As an example the access token for Instagram could be used to extract info about the number of followers of users and create a DAO of influencers. As another example profile pictures could be exploited in the following way: a group of people can add the same banner to their own profile pictures (e.g., a banner about some social cause) to create a DAO of people united by the some cause.

The token validity. For some applications a token should not be valid forever (accounts can be deleted or blocked). One can associate e.g. a month+year to the token so it can be publicly visible for which period a token is associated. An alternative solution would be to resort to more advanced functional encryption systems like wildcards IBE or Hidden Vector Encryption that would allow to decrypt past ciphertexts with a single token.

Verifiability and privacy. In the context of our system there are different levels of verifiability. The best property would be to guarantee that even if all nodes in the League of Identity network collude together, they cannot forge valid signatures under identities they do not control.

Instead, for encryption and privacy the security must rely on threshold assumptions: a sufficiently large set of nodes can always break the privacy.

Having verifiability based on threshold assumptions is more general and offers more flexibility. Some providers can only offer online deniable verification for their access tokens or for particular features (e.g. linking the identity to a phone number). Therefore, in this work we assume that the providers's access tokens are verified online by the LoI nodes and thus both the verifiability and privacy are under a threshold assumption. In our system the threshold can be completely arbitrary (not necessarily majority). Choosing a large threshold offers more security but less robustness in case of faults of nodes and adds more communication costs to the users.

Threshold security against providers. In the basic setting providers like Google must be trusted. It is possible to consider the following variant. Let us assume for simplicity that Bob has a Facebook account with email address *bob@gmail.com*. Bob logs into both his Google and Facebook account to request the respective access tokens and sends both access tokens to the League of Identity nodes which grant to Bob a token for a joint $google + facebook$ identity. Abstractly, the provider can be seen as the intersection of Google and Facebook. That way, the security of Bob can be broken only if Google and Facebook collude, or of both accounmts are compromised simultaneously.

Applications. In Section 3 we propose our anonymous payment system that enables crypto users to send crypto assets to non crypto users.

Other applications include the creation of DAOs based on real-world identities or organizations or e.g. the DAO of influencers having > 500k followers, or the DAO of all members of the company *oldcrypto.com* etc. These DAOs can have at same time verifiability for the members and *encrypted proposals*.

Related work IBE/IBS were proposed by Shamir in '84 [1] but concretely implemented only in 2001 by the breakthrough work of Boneh and Franklin [2]. Recently, there has been interest in deploying identity-based solutions for the web3 [3, 4, 5]. We remark that none of them implements IBE and as such cannot implement the applications that we propose in Section 3. Moreover, the latter solutions require expensive SNARK proofs for circuits relative to JSON Web Token computation and as such are not easy to adapt to providers' APIs which do not provide verifiable tokens.

2 Our implementation

We are in a bilinear group (G_1, G_2, G_T, e) of type 3 and we assume the groups to be of prime order p with generators resp. $g_1, g_2, g_T = e(g_1, g_2)$.

We assume that the nodes perform a Distributed Key Generation (DKG) procedure to compute shares $MSK_i \in Z_p$ of the master secret key $MSK \in Z_p$ in a way that the master public key can be set to $MPK = g_2^{MSK}$ and a subset S of t shares allow to reconstruct $MSK = \sum_{i \in S} MSK_i \ mod\ p$.

We don't discuss any specific Distributed Key Generation (DKG) protocol here since the previous properties can be achieved using several known DKG protocols, e.g., one can use the same DKG protocol used for the drand [6] system.

We can additionally assume that when new nodes enter or leave the system, a new DKG procedure is run to re-share the same master secret key and master public key. This is not really necessary but simplifies the treatment. We will also skip details regarding the validity of tokens.

Token generation In our system the share computed by each node $i \in S$ in response to a valid access token for email bob@gmail.com is $Token^i_{Bob} = H(bob@gmail.com)^{MSK_i}$, where H is an hash to point function that maps to G_1 and is indifferentiable from a random oracle.

From the t values $Token^i_{Bob}$ for $i \in S$, Bob can reconstruct $Token_{Bob} = \prod_{i \in S} Token^i_{Bob} = H(bob@gmail.com)^{MSK}$.

This can be used for encryption and authentication/signatures as follows.

2.1 Encryption

Our system is the Boneh and Franklin AnonIBE [2] that we recall next. Alice, on input a message m to encrypt and email bob@gmail.com and master public key MPK can choose random value $s \in Z_p$ and compute: $A = g_2^s, B = m \oplus H_T(g_{id})$, where $g_{id} = e(H(bob@gmail.com), MPK^s)$ and H_T is a function that maps G_T into bit strings of some length compatible with the message that is indifferentiable from a random oracle. Decryption of a ciphertext $CT = (A, B)$ can be done computing the following: $e(Token_{Bob}, A)$ and, by construction of the token and by the bilinear property, the latter value equals g_{id} so m can be recovered as $H_T(g_{id}) \oplus B$. The previous system is IND-CPA secure and can be done IND-CCA secure using the Fujisaki-Okamoto's transform as explained in [2].

2.2 Authentication and signatures

Authentication/signatures. We first focus on the signature case. The user has a token $Token_{Bob}$, the master public key $MPK = g_2^{MSK}$, chooses a random value $r \in Z_p$ and computes the following. $C = MPK^r = g_2^{MSK \cdot r}, E = g_1^r$, and $F = Token^r_{Bob} = H(bob@gmail.com)^{MSK \cdot r}$. Moreover, the user computes a proof π of knowledge of the exponent r in E, that is a non-interactive Schnorr's proof

except that in the proof the challenge is set to the hash of the concatenation of E with the first message of the Schnorr's protocol and the message m. This binds m to the proof π. The signature σ of the message m consists of tuple (C, E, F, π). The verifier needs to verify the following. It first verifies that $e(E, MPK) = e(g_1, C)$. This convinces the verifier that $C = g_2^{MSK \cdot r}$ and $E = g_1^r$. Then the verifier verifies that $e(H(bob@gmail.com), C) = e(F, g_2)$. This convinces the verifier that $F = H(bob@gmail.com)^{MSK \cdot r}$. Then the verifier verifies that π is a Schnorr proof computed including m in the challenge as explained above. Since a valid Schnorr proof can be computed only with knowledge of r this convinces the verifier that the user knows a valid token $Token_{Bob}$ without disclosing the token and since only Bob can get the token from the nodes this is a proof that the signature was from Bob. Authentication can be done using the Schnorr protocol in an interactive way and without binding the challenge to any message. .

3 Anonymous Identity-Based Payments (AIBP)

Recently, some financial apps enable payments via phone numbers and other real world identities. Similarly, we propose what we term an *Anonymous Identity-Based Payment* (AnonIBP) system.

Using an AIBP Alice can deposit any crypto assetin favor of the Bob's email address, social identity or phone number.

Bob, who is not a crypto user, can check that there are crypto assets in favor of him without the need of installing any wallet by just using the ability of logging into his Gmail, Facebook etc. accounts . The assets are securely and anonymously deposited into a smart contract.

Bob in the future can decide to become a crypto user and to install a wallet and he will be able to withdraw in favor of any Eth address - his own or someone else's address. The system can be also extended to enable Bobto withdraw in favor of other Gmail, Facebook, etc. accounts without even having a wallet - the GAS fees will be paid by paymasters that can take a fee for this service. Both deposits and withdrawals are fully anonymous, that is they do not reveal the Gmail, Facebook, etc. account.

The trivial system. We can implement a trivial AIBP as follows. If Alice wants to send n coins to Bob, she can do the following.

Use our CCA-secure IBE system to compute a ciphertext CT encrypting a random nonce x for the identity id, compute $y = Hash(x)$ and send to a smart contract n coins along with the pair (CT, y).

Bob can get from the LoI nodes the token relative to Bob's identity id and decrypt CT to get x and send to the smart contract x. The smart contract verifies that $Hash(x) = y$ and if the check is successful it transfers the n coins to Bob (that is, the coins are sent to the Eth account that invoked the smart contract).

Notice that, being our IBE system anonymous, the ciphertext also hides the identity.

An issue with this trivial AIBP system is that the sender can always withdraw on behalf of the receiver at any time. This is not a problem only in applications in which sender and receiver can communicate at deposit time: in that case the receiver Bob can request the sender Alice to deposit a ciphertext computed by his device.

3.1 Fully-secure AnonIBP via ZK proofs of correct decryption

We can obtain a fully secure AIBP system that does not suffer the above problem as follows.

EVM-friendly approach. The sender Alice (who wants to deposits crypto assets in favour of the receiver Bob) can choose a random value $r \in Z_p$, compute a CCA-secure IBE ciphertext CT for the Bob's identity id and message r, compute $D = H(id)^r$, and deposits on-chain the values CT and D. Bob can verify that a payment is withdrawable by him performing the following check: use his own token for the identity id to decrypt CT and get r and check that $D = H(id)^r$. The check is carried out off-chain in e.g. Bob's browser. The property we want to ensure is that if this check passes then Bob should be convinced that the payment is withdrawable by him and only by him.

In order to withdraw, Bob chooses a random value $s \in Z_p$, compute $E = D^s$, a non-interactive Schnorr's proof of knowledge π of dlog of E in base D (i.e., knowledge of s), the re-randomized token $Token'_{Bob} = Token^{r \cdot s}_{Bob} = H(id)^{msk \cdot r \cdot s}$ and sends to the smart contract the values $E, \pi, Token'_{Bob}$. (Note that Bob knows r since Bob can decrypt the IBE ciphertext CT as above.) The smart contract verifies π and checks that $e(Token'_{Bob}, g_2) = e(E, MPK)$. As before, Bob needs to include the withdrawal address as part of the FS challenge of the proof π. The idea is that the security property we wish is satisfied due to the fact that if these checks pass then Tok'_{Bob} is a correct re-randomized token, where the re-randomization is done with the values r, s resp. corresponding to CT and π and thus Bob knows a valid token for identity id and this is enough to conclude that the smart contract was invoked by the legitimate Bob.

Withdrawals in favor of non-crypto users. We supposed for simplicity that the withdrawal will be in favor of an Eth address. A natural extension is to enable withdrawal in favor of other non-crypto users (that is, Gmail, Facebook, phone numbers, etc.). Indeed, a user Bob will be need to withdraw in favor of an Eth address only when he will need to swap the token/coins or exchange for fiat.

Of course, this withdrawal in favor of non-crypto users consumes GAS. The GAS cost could be paid by a service that retains a fee for that so that the users will be able to make this withdrawal without having crypto and even without having a Wallet at all.

Batch withdrawals. The system can be generalized so that Bob can withdraw n deposits with a much lower cost than n independent withdrawals.

4 PoC implementations

In [7] we propose a PoC implementation of the system described in this paper along with the corresponding documentation. In [8] we additionally implemented smart contracts of the AIBP system of Section 3 and of a DAO for organisations using Google accounts. A demo of our AIBP system is accessible at `https://demo.azkr.ch` and supports payments in favour of Gmail, Facebook users and phone numbers (it is sufficient for the receiver to link any Gmail account to his phone number).

References

[1] Adi Shamir. Identity-based cryptosystems and signature schemes. In *Advances in Cryptology - CRYPTO 1984*. Springer, 1985.

[2] Dan Boneh and Matthew K. Franklin. Identity-based encryption from the weil pairing. In *Advances in Cryptology - CRYPTO 2001*. Springer, 2001.

[3] Foteini Baldimtsi, Konstantinos Kryptos Chalkias, Yan Ji, Jonas Lindstrøm, Deepak Maram, Ben Riva, Arnab Roy, Mahdi Sedaghat, and Joy Wang. zklogin: Privacy-preserving blockchain authentication with existing credentials. arXiv, 2401.11735, 2024.

[4] Privacy Preserving Exploration Team. Anon Aadhaar. `https://github.com/anon-aadhaar/anon-aadhaar`, 2023.

[5] Florent Tavernier. proof-of-passport. `https://github.com/zk-passport`, 2024.

[6] Wikipedia contributors. drand. `https://drand.love`.

[7] Vincenzo Iovino. League of Identity - PoC. `https://github.com/aragonzkresearch/leagueofidentity`, 2024.

[8] Vincenzo Iovino. LoI.SmartContracts. `https://github.com/vincenzoiovino/LoI.SmartContracts`, 2024.