Nym Credentials: Privacy-Preserving Decentralized Identity with Blockchains

Harry Halpin
Nym Technologies
Neuchatel, Switzerland
Email: harry@nymtech.net

***Abstract*—Nym credentials solve the problem of privacy-enhanced authentication amongst a decentralized open-ended ecosystem of services. Many blockchain systems offer decentralized services, but at the cost of revealing all transactions in a public ledger, which is clearly not suitable for high-value transactions involving personal data. Nym credentials deploy anonymous authentication credentials as an identity system that maintains user privacy. Due to the validators being decentralized, the system is compatible with blockchain-based systems. Tokens can be used to show the "right to access" privacy-enhanced services. Nym validators transform these tokens into Nym credentials that validated publicly in a decentralized manner via a modified Coconut signature scheme capable of the open-ended embedding arbitrary attributes.**

I. Introduction

Privacy-enhanced services have at their heart what appears to be the impossible paradox of identity management: Users want to use these systems without revealing more information than is necessary about their personal identity to the service, but at the same time service providers need to use some form of authentication lest their limited resources be overrun by anonymous and possibly malicious users in a *sybil* attack [1]. Currently, services have to use techniques to prove a user's identity that violate the privacy of the users, such as using phone numbers as proxies for identity.

Our novel contribution is to use anonymous credentials that can be issued and verified in a decentralized manner with an open-ended number of attributes. This technique can be easily integrated with blockchain technology and tokenization for a wide variety of services. These credentials can be issued and validated in a decentralized manner via the use of a modified Coconut blind signature scheme [2], but unlike Coconut, Nym allows a flexible choice of user attributes that can differ between service providers.

Taking as inspiration OAuth 'tokens,' Nym credentials can replace authorization and authentication tokens in a distributed environment. In order to maintain privacy in token-based systems that use public ledgers, non-private tokens can be converted to Nym credentials in a manner that is also publicly verifiable. Tokens can be indexed to service provider capacity via the usage of a public ledger. We map in detail the use of the Coconut-based Nym anonymous authentication credential scheme to the paradigm of federated identity and test the implemented system for performance.

In this paper, we overview federated identity in Section II, including OAuth, W3C DIDs, and anonymous credentials. We outline the central problem to be solved in terms of the threat model and needed security and privacy properties of Nym credentials in Section III. The overall design and components of the usage of Nym credentials for federated identity are described in Section IV and the flow of the protocol in Section V. Nym credentials are evaluated in terms of performance in Section VI. Finally, next steps are outlined in Section VII.

II. Background

In practice, federated identity management is typically done via the standardized IETF OAuth (Web Authorization) framework, although the system is both centralized and not privacy-preserving

978-1-7281-9390-8/20/$31.00 ©2020 IEEE
DOI 10.1109/CVCBT50464.2020.00010

as detailed in Section II-A. In Section II-B, we show blockchain-based alternatives currently under standardization at the W3C suffer from considerable privacy issues. Instead, Nym credential use anonymous authentication credentials, whose large body of research is briefly reviewed in Section II-C.

A. Identity Management with OAuth (OpenID)

Due to well-known difficulties in managing multiple accounts across different services, there has been a move towards federated identity management. In the federated identity paradigm, the *identity provider* (IdP) manages the identity via controlling the personal data of a user, given by a finite number of *attributes*. *Service providers* (SPs) are any service that needs to authenticate a user, and possibly use their attributes for operations.

OAuth works by establishing a shared secret, called a 'token,' between the SP and IdP mediated by the user. This token does not have standardized cryptographic properties, but typically services hash the shared secret with a salt, such as a timestamp. The token can be considered a capability in terms of access control, yet these OAuth tokens do not have any well-defined privacy properties [3]. Although the user mediates the creation of the token, once an SP is authorized via the user's token, attributes such as personal data are then transferred directly to the SP from the IdP. At this point, the user is "out of the loop" such that the nature of the personal data being transferred remains opaque.

Facebook Connect, Google Sign-In, and Sign-in with Apple use the IETF OAuth 2.0 open standard for authorization [4]. Despite differences in attributes that prevent each proprietary deployment of federated identity from being compatible, the core protocol and flow of personal data is the same as given the OAuth 2.0 standard. Some services use the profile of OAuth given by OpenID Connect.[1]

The identity provider is a trusted third party, leading to privacy and issues with this identity management scheme. From a privacy perspective, the identity provider has complete visibility into all transactions of the user, and knows every service provider a user interacts with. In terms of security, a malicious IdP can impersonate a user who wants to use another IdP in the "IdP mix-up" attack as no user signatures are used in OAuth [5]. While the "mixed-IdP" attack can be prevented with additional information being added to the token,[2] the fundamental issue of enabling privacy requires advanced cryptography such as ring signatures [6], or algebraic MACs [7] as detailed in Section II-C. In brief, all of these solutions require a centralized identity provider and do not defend against a denial of service attack, where the IdP simply refuses to authenticate a user or authorize them.

B. Blockchain-based Identity

Decentralized identity management means that there is no single identity provider, but that identity is provisioned using a decentralized method such as blockchain technology [8]. Unlike in the federated setting, the user is explicitly represented by a keypair and a single blockchain is maintained via consensus between *validators*, nodes that reach consensus on the state of the blockchain using an algorithm. The user's public key is written to the chain, and statements signed with that key can thus be independently verified.

Although the core idea is simple, various companies such Evernym and the non-profit Sovrin[3] have created a convoluted "standardized" architecture called W3C Decentralized Identifiers (DIDs), supported by the U.S. Department of Homeland Security.[4] This architecture requires resolving the fragment of a public key to a document that specifies authentication methods, keys, and then so on that can then be used by services to verify W3C Verifiable Claims, a JSON document where the assertions are signed. However, W3C DIDs suffer from a number of weaknesses. First, the resolution from a key in a DID to a DID document is not specified, and in practice permissioned federations such as Sovrin resolve the DID to the necessary DID document. Second, although DIDs claim to support "zero knowledge proofs," there is no advanced cryptography used outside of RSA and elliptic curve Ed25519 signatures in the actual standard [9], although particular implementations like Hyperledger could use zero-knowledge proofs or anonymous authentication credentials.

[1] http://openid.net/specs/openid-connect-core-1_0.html

[2] https://tools.ietf.org/html/draft-ietf-oauth-mix-up-mitigation-01

[3] https://sovrin.org/

[4] https://www.sbir.gov/sbirsearch/detail/1302459

There is no simple way to enforce the privacy of any attributes attached to the DID document, only that "it is strongly recommended that DID documents contain no PII" (personally identifying information)[9]. As they are effectively permanent identifiers written to a public blockchain, W3C DIDs may of course be correlated. One-time use DIDs are not enough, as "the anti-correlation protections of pseudonymous DIDs are easily defeated if the data in the corresponding DID Documents can be correlated"[9]. Although DIDs do not easily suffer from a malicious IdP like OAuth and may have higher availability, as the blockchain is public, the leakage of metadata using DIDs is worse than traditional federated identity with OAuth, as correlation attacks may not only be done by a malicious IdP but by any actor due to the DIDs being stored in a public chain.

C. Anonymous Authentication Credentials

As privacy-enhancing technologies need the ability to authenticate users and authorize transfer of their data while simultaneously maintaining their privacy, there is a long tradition of deploying more advanced cryptography than currently used by IETF OAuth or W3C DIDs. The work of Chaum on *blind signatures* [10] has led to numerous schemes for selective disclosure credentials based on either the RSA or DH assumption, but typically these schemes require a centralized provider of credentials and are not publicly verifiable [11], and so are not usable within a decentralized environment. Likewise, another stream of research has used zero-knowledge proofs for selective disclosure credentials, leading to attribute-based credential systems such as Microsoft's U-Prove and IBM's Idemix [12] (currently used in Hyperledger Fabric[5]), but these schemes are both computationally expensive and centralized. Although some schemes use blockchain technologies to achieve decentralization (in conjunction with RSA accumulators), these are still too computationally expensive for real-world use and do not have decentralized issuance via threshold cryptography [13]. Furthermore, attribute-based credential systems have had very low deployment in comparison to centralized OAuth-style systems such as Facebook Connect [4]. Adding cryptographic constructions such as Algebraic MACs to OAuth has been done by the privacy-preserving unlinkable OAuth-compatible system UnlimitID. Yet UnlimitID still requires the issuer of the privacy-preserving credentials to be the same centralized entity that verifies the credentials [7].

One recent advance was the creation of the Coconut credential scheme, which via virtue of using publicly verifiable hashes of the server used in generation of the blind signatures, allows threshold blind signatures that are publicly verifiable [2]. By virtue of using the Coconut credential scheme rather than alternatives such as the algebraic MACs used in UnlimitID, Nym credentials are publicly verifiable and decentralized, while allowing the implementation of an OAuth-based flow for authorization and cryptographic authentication to preserve privacy. However, in order to maintain a constant length, Coconut uses a fixed number of elliptic curve points in each credential, so each attribute in Coconut has its own key. This means the number of attributes is fixed, so Coconut does not support a diversity of services with their own custom attributes. As each per-attribute key must be replicated by each validator, this also makes key management for Coconut unwieldy and difficult to deploy in practice.

III. Problem and Goals

A. Threat Model

In terms of security, the threat model is a malicious identity provider that can be unavailable and can impersonate a user on arbitrary services via replaying user attributes, as well as altering user attributes. In terms of privacy, we also want to prevent an honest-but-curious identity provider(s) or validator(s) from correlating a given user with the precise service providers the user is utilizing and when. We also want to guard against malicious users, so we also want to prevent users from replaying anonymous credentials in a sybil attacks on service providers.

B. Privacy and Security Properties

In order to counter these threats, Nym credentials should have the following properties:

[5]https://hyperledger-fabric.readthedocs.io/en/release-2.0/idemix.html

- **Undetectability:** Identity providers, validators, and other third-parties cannot link credentials to the owner of the respective credentials. Only the service provider can link a credential to the user if a user explicitly shows the credential to the service provider with a linkable attribute.
- **Unlinkability:** Identity providers and service providers cannot link the different credentials of a user across service providers or across different accounts on a single service provider.
- **Selective attributes disclosure:** Only a user can choose which subset of their attributes to reveal to the service provider each time in response to the request for attributes by a service provider.
- **Unforgability:** Identity providers and other third parties cannot impersonate a user to a service provider.
- **Persistent accounts:** Users can optionally have long-lived pseudonymous accounts on service providers.
- **Sybil resistance:** Service providers can enforce that users may create up to a certain number of credentials for a service provider or for the network as a whole.
- **Extensibility:** Service providers can ask for different and custom attributes from a user within a single identity framework.
- **Availability:** The authentication process of users to service providers should be resistant to byzantine faults and thus available when needed.
- **Decentralized:** There should be no single trusted-third party that is trusted by all the components of the system [14].

Similar to OAuth, user authentication methods to key material or how to securely store and synchronize secrets between devices are not specified. An operating environment with trusted cryptographic primitives and key storage for all parties in the protocol is assumed. In order to defend against network-level adversaries, an anonymous channel such as a mix-net should be used to deliver the credentials to the service provider and for all communication between the user and the service provider [15].

IV. Nym Credential Design

Nym credentials are part of a wider framework, the *Nym framework*, that can have another key component to prevent sybil attacks: The ***NYM token*** is a voucher for the usage of a privacy-enhanced service in a given epoch, such as the sending of a certain number of messages or the use of so much capacity of a VPN per month. The transfer of NYM tokens for authentication is essentially equivalent to proof of the right to use a service. Note the NYM token can be stored on its own chain or may be stored on an external blockchain for ease of use (for example, using a smart contract on Ethereum) or transferred to a sidechain of an existing blockchain (in Bitcoin).

So that a NYM token can be used to authenticate in a privacy-enhanced manner (as the token has no privacy properties), the NYM token has to be transformed into a credential that embeds any needed information for the service provider in an encrypted and unlinkable manner. After transfer to a smart contract, one or more tokens transform into a ***Nym credential***, an anonymous authentication credential based on a modified Coconut signature scheme [2] to support extensibility. This credential is shown to the service provider, revealing only the necessary user attributes for the service, giving the user access for an epoch of time. At the end of the epoch, the service provider can then prove they have been shown a given number of Nym credentials in a given epoch to receive the equivalent number of NYM tokens in exchange for provisioning the service. This allows the service to be rewarded for provisioning services and control the capacity of a service or network as a whole.

It would appear that the anonymity set of a given user is at least the set of all the users that authenticate, i.e. all Nym credentials that exist in a given epoch. In order to increase the anonymity set, we can force Nym credentials that are valid but not used in a given epoch to be refreshed, thus allowing the anonymity set to increase to those all Nym credentials that have not already been shown in *any* epoch. However, in order to enables persistent accounts, we allow users to allow selective *linking* of their identities by their own service providers in order to enable long-term accounts, but this linking is not be allowed for third-parties like validators and identity providers. Identity providers should only

have access to documents needed for the verification of user attributes, not knowledge of which services are being used by a user. Validators should only have access to knowledge that is explicitly stored on one or more blockchains.

We foresee different types of service providers, requiring different types of attributes to provide access, as well as different levels of linkability to long term identifiers. Nym supports all policies and ensures users are in control of revealing the type of information that service providers require from them. For fully anonymous short-term access to a service, only attributes necessary to run the service are embedded. Abuse can be dealt with on a per-service basis via techniques like privacy-preserving credential blacklisting. Some services, such as online collective decision making platforms, only need to ensure that the same user did not perform an operation, such as voting, more than once. For these, we can derive a per-action serial number distributed to users on registration and blinded before use, and ensure that a user performing the same action twice is detected. This does not provide any less privacy to non-cheating users than the previous option. Services that operate on the basis of user accounts, such as secure messaging or email providers, need a more permanent notion of identity to allow access to contact lists and media associated with an account. To support those, we ensure that Nym credentials can be used to derive per-service pseudonyms and prove that only a valid user can access this long-term identity via proof of possession of a private secret. These long-term identities are consistent between Nym credential shows from the same user but are unlinkable to the credential issuance or to pseudonyms in other services.

A. Components

The components of the entire system, including their communication over the network layer, are called the *Nym network*. The details of functions and mathematical terms for keys are defined in the Coconut paper [2].[6]

1) The NYM Token: The NYM token is an token that is simple fungible token, such as an ERC20-compatible token. By indexing the number of credentials to NYM tokens, the supply of active NYM tokens can be indexed to the amount of resources available to services using the Nym network, so that large scale service providers that require NYM tokens can provision services without being overrun with sybils or without compensation. A NYM token offers no privacy without being transformed into a Nym credential. The tokens can count the number of usages-per-epoch of a service provider and so may be used to reward the service provider, as well as to prevent sybil attacks. NYM tokens themselves do not expire.

2) The Nym Credential: The Nym credential is the cryptographic credential used for privacy-enhanced authentication that embeds both service-specific and possibly account-specific information. The Nym credential currently uses a modified Coconut anonymous aggregate signature scheme [2] in order to let each access with the Nym credential be unlinkable to the user and any other transaction by outside observers. By virtue of being an aggregate credential operating with a threshold signature, it is assembled by the user from per-validator partial credentials. A user may combine certificates of attributes to issue a Nym credential, and the user may combine attribute certificates from one or more identity providers, via linking these credentials via a proof of possession of a key, as well as a serial number if needed. A serial number, in combination with a blockchain that is accessible to all service providers that stores blinded serial numbers, allows the *one time* use of Nym credentials and so prevents double-spending attacks, i.e. the same user using the same credential across different service providers. Service providers and users may opt not to use serial numbers if they wish to allow double-spending of certain attributes in credentials. For example, the transfer of a token should likely have double-spending protection, but the use of a verified birth certificate likely should not.

The validator has no access to user attributes, as the validator only can prove whether or not one or more token has been converted to a credential and that the credential has a valid form. In order to simplify key management and allow extensibility (unlike Coconut), each credential is *actually* a number of Coconut credentials of a single user-given attribute, where multiple attributes are linked via proof of key possession (and optionally, a serial

[6]The formal definitions will need to be modified to enable split and merge operations. This is future work.

number) to form a single Nym credential. Each issued credential consists of aggregated signatures from a threshold number of validators ("issuing authorities" in the anonymous credential literature [2]). The verifiable attributes consist of:

- **Account secret value:** (*private*) k. This may be fresh for each Nym credential, but services that require persistent accounts and enable abuse protection will want this to be a long-term secret. Credentials may use this value as a key or a key derived from this value, including in order to bind the credential to a particular service. Proof of this long-term key material prevents transfer of attributes between users and is needed for multi-attribute credentials.
- **Serial number:** (*private*) s that has to be fresh for each new Nym credential generated to prevent double-spending attacks, in particular those on the transfer of NYM tokens by users to service providers. The serial number is a nonce that is only revealed in a blinded form, to prevent linking to initial creation of credential.
- **Attribute:** (*private* or *public*) m from a user that needs to be shown to a service provider and can be optionally signed by IdP idp.
- **Epoch** (*public*) v, representing a time period, used to restrict the usage of credential to specific epoch.

All attributes are mandatory so that each credential remains a constant size, and unused attributes (such as a serial number for a multi-show attribute) can simply be left as default unused constants. Multiple attributes are done via linking these credentials via proof of the account secret value. New attributes may be added to the base schema, such as type of credential, as long as all credentials are refreshed to the new constant size.

3) User: A user wants to access a privacy-enhanced service via the Nym network. It is assumed they have a wallet containing NYM tokens that contains at least the amount in NYM tokens needed by the service providers that the user wants to access for a given epoch. They have a number of attributes $[m_1, \ldots, m_q]$, a long-term secret k, and an encryption and decryption keypair (sk_u, pk_u). El Gamal is used as re-randomization is needed [2].

4) Identity Providers: A user may generate their own identity and public key material. For services with long-term accounts, a possibly pseudonymous per-service user identity and public key material can be issued by identity providers directly to the user. These identity providers may have an existing relationship with the user, such as a government entity or an affinity group of activists, and so can verify their identity for their pseudonymous account. An attribute certificate may contain both attributes that are public to the service provider, as well as private attributes given by commitments (or another form of NIZK), and these can be signed under the key of the identity provider. Identity providers have a signing keypair (sk_{idp}, pk_{idp}) and may use this to sign $[m_1, \ldots, m_q]$.

5) Blockchains: In order to keep track of amount of tokens in the Nym network, an account system is kept on a blockchain that corresponds to the amount of tokens possessed by each user. Each statement in the chain X_i is either a mapping between account address (keys) and a value ($X_i = addr \rightarrow val$) that represents the amount of NYM tokens. These tokens may be kept on an *external chain* that is not maintained by the Nym network, or it may be kept on the *Nym blockchain* that also maintains the proofs of double-spending. We assume that the Nym blockchain is maintained by the validators of the Nym network, although certain flexible blockchains (such as Ethereum) could function as both the external and Nym blockchain. The Nym credential framework is neutral to the underlying blockchain technology and even number of blockchains used.

These tokens can be sent by users to an address watched by the validators or a smart contract controlled by the validators, called the *Nym smart contract*. The user sends NYM tokens to a smart contract and receives the ability to get credentials of equivalent value in NYM tokens. The validators then can then transform the NYM token(s) into a Nym credential and keep track of the amount of token in a user's account that has been transformed into Nym credentials. The Nym blockchain serves as a public record where service providers record blinded serial numbers for the credential such that $X_i = \zeta$ where $\zeta = g^s$, where s is the private serial number of a user's credential (to prevent double-spending). Once a service provider is given a Nym credential, it writes ζ to the Nym blockchain. Only

service providers and validators may write to the Nym blockchain, although it is public so anyone can verify a Nym credential.

6) Validators: The validators issue Nym credentials and are also are responsible for controlling system-wide information, such as keeping track of amount of NYM tokens and consensus on the state of the Nym blockchain. If the user does not have enough NYM token to issue a credential for a service provider, the validators do not issue the credential. If the user has sent a NYM token to the Nym smart contract, the validators issue partial credentials to users of the system in response to attributes sent by the user. Each validator has its own public verification key vk for its Coconut signatures [2]. The validators can blindly sign credentials with the secret key corresponding to vk. They blindly sign user commitments to m, a proof of possession of the k as well a serial number s if needed, and attaching the epoch information v.

7) Service Providers: Each service provider accepts Nym credentials to authenticate, and can prove how many credentials they have received in order to receive NYM tokens for provisioning services. A service provider knows the verification keys vk of each validator, and so given a re-randomized credential *cred'* and the results of a user showing the credential, the service provider can *verify* if a credential is valid by checking the signature of with the vk keys of the validators.

For attributes that can only be used once, such as the transfer of NYM tokens to a service to authorize access, the service provider checks to see if a per credential blinded serial number ζ is on the Nym blockchain. Assuming tokens are needed to access the service, if ζ is not recorded on the Nym blockchain (and so there has been no "double-spend" attack), the service allows the user to access, and then writes the ζ to the Nym blockchain to prevent a double-spend. It may then accept other attributes from the user and ignore the re-occurrence of the same ζ. If the ζ is already on the chain, the user may not be allowed access. A credential that shares the same proof of k (and possibly the same s) with a different m is considered a single Nym credential and so ζ is written only once on the Nym blockchain for that credential.

V. Information Flow

The next sections describe the entire flow of the Nym credential framework within the Nym network. The flow starts with the transformation of NYM tokens into a credential by a user, the user embedding attributes from identity providers and obtaining the Nym credential from the validators, and finally the showing of the Nym credential by the user to gain access to a service provider, which the service provider can then redeem at the end of the epoch for NYM tokens.

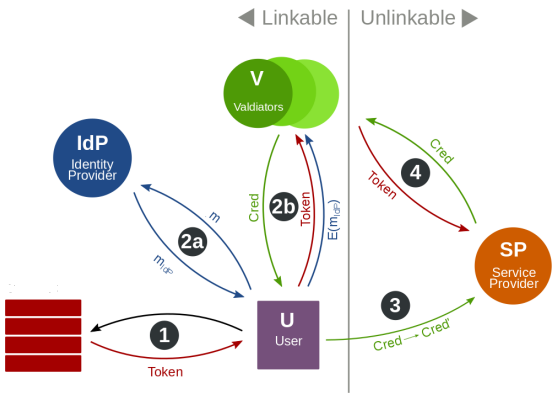

Fig. 1: Flow of NYM tokens

The movement of NYM tokens and credentials between actors in the framework is displayed in Figure 1. *Linkable* means that a third-party or entity within the system can link a user to their transactions with other entities in the Nym network, while *unlinkable* means entities cannot link a user with their actions simply by observing and inspecting the flow of information (without the explicit show by a user of their credentials). A user wants to use a Nym-capable service provider, and so the user determines what attributes (m) are needed to access the service provider. The service provider can advertise these attributes required by their service publicly. The system is assumed to be initialized with a distributed key generation phase that has been run for validators, IdPs, and users.

1. User obtains NYM tokens: Once the user transfers NYM tokens into the Nym smart contract, the validators confirm the transaction via consensus protocol. As the Nym framework is supposed to be neutral to the underlying blockchain, the

consensus protocol is unspecified but is assumed to require consensus of more than λ validators (for example, λ in Tendermint PBFT is two-thirds). Once the transaction has been validated, partial Nym credentials can be issued to the user for an amount equal or less than than the amount in their account transferred to the Nym smart contract. This is not required for services that do not require NYM tokens. In order to prevent the linking of values of NYM tokens, given that the transfer of the tokens to the Nym smart contract is public, the amount of token transformed into an individual credential should be less than that in the user account. To remain unlinkable, a user may transfer a large amount of NYM token to the Nym smart contract and split it among multiple credentials (for multiple services). Likewise, a user send one amount of NYM token into the Nym contract and another amount later, and merge credentials in order to obtain at least the amount of NYM tokens needed by a service provider.

2. Validators transform tokens into a credential: Optionally in (**2a**), the user can send attribute certificates m with evidence to IdPs for verification, and so a signed attribute certificate m_{idp} is returned to the user. For service providers that require long-term accounts, the user may ask for key material after verification. If a user wishes to connect their own self-generated long-term key k to a pseudonym, they may ask for key material connected to a proof of possession of k where the IdP does not therefore gain knowledge of the long-term secret k of the user. The user can also simply generate their own long-term k without an IdP and self-sign attributes so that it can send m rather than m_{idp} to the validators.

In **2b**, the user at this point has collected all needed attributes m and any third-party identity provider verification of m is complete. The user sends encrypted El-Gamal commitments to the attributes to each validator. As multiple validators may see the same El-Gamal encrypted commitments $c = E(m)$, the user should re-randomize the El-Gamal ciphertext $c \rightarrow c'$ in order to prevent the validators from gaining knowledge of their attributes, even if hidden via commitments. Each validator returns to the user a blindly signed partial credential *cred* for each attribute. The valid epoch v can be set to the next epoch in order to prevent timing attacks.[7]

Once more than λ of the validators have returned the blindly signed partial credentials for m, the user combines the blindly signed partial credential into a valid Coconut credential via unblinding the credential for m, aggregating the signatures of the validators, and re-randomizing the Coconut credential to de-link it from any previous transactions. This is repeated per attribute m with proof of possession of the same k (and, if needed, the same blinded s) in order to compose a single Nym credential from multiple Coconut credentials, thus gaining extensibility.

The credential is privacy-preserving but bound to the user due to it containing proof of knowledge of user secret k, a blinded serial number ζ, as well as an public expiry epoch v, in addition to the public or private attributes needed by the service provider. This procedure is distributed and fault tolerant using threshold cryptography, so only more than λ of the validators need to be available and honest. In order to prevent timing attacks, once the token is transformed into a Nym credential, the user should wait for the next epoch v to use the credential.

3. User shows credential to an SP: Upon issuance the user *re-randomizes* the credential, ensuring that any subsequent showing of the credential for authentication is unlinkable to any validators or any other third-party from previous transactions. Blinding happens currently via re-randomization of the Nym credential ($cred \rightarrow cred'$ in Figure 1). Now that the signed credential is unlinkable, the user may then perform the selective disclosure credential *show procedure* to prove possession of a valid Nym credential to a service provider and, if needed, reveal service provider-specific attribute m. Showing a credential produces a publicly verifiable Coconut signature that convinces anyone of the validity of the credential. Further attributes or may also be revealed by revealing Coconut credentials linked by a proof of k, and their validity is guaranteed. Note that this does, unlike Coconut,

[7]Note that the number of m sent by the same user to the same validator is revealed if the user is not using an anonymous channel or allowing an attribute of their credential to be linked, so making sure attributes are unlinkable and even sending 'fake' or simply unused attributes to the validator is possible.

leak the number of shows to the number of attributes required by a service provider as well, but the number of attributes a service provider needs is public.

4. Validators confirm transfer of tokens to SPs: Immediately, the service provider verifies if the credential has been spent before (via checking if ζ has been marked as spent by another service provider on the Nym blockchain) and if it is within its valid epoch v. If it has not been spent, the service provider writes ζ to the Nym blockchain along with a proof it did so. The user may then proceed to use the service provider for the epoch v. At the end of the epoch v, each service provider presents all the valid credentials they have received from users and the collection of proofs per each ζ they inscribed. At the end of the epoch the validators confirm the transaction of tokens from their aggregate users to the SPs, i.e. releasing the tokens from the Nym smart contract to the account of the service provider on the external blockchain at the end of the epoch corresponding to the amount sent by the user. As the number of NYM tokens are mixed together for every user that accessed the service during epoch v, tokens are sent as one large sum at the end of the epoch. Thus the transfers are unlinked both over time and in amount. The service provider may then exchange the resulting sum of NYM tokens to fund their operations if necessary.

Nym credentials are unlinkable both on the level of ciphertext and communication through the information flow. At first it may seem that the Nym credential system simply places the validators in the place where the identity provider was before in OpenID and so validators can link a user to their credential via observing the information flow. However, as the validator is only blindly signing data encrypted by the user and creating partial credentials that the user then merges to create a full Nym credential, the validator cannot link the credential to the specific attributes of a user. The user re-randomizes the credential so that the validator cannot link the ciphertext of a credential to its use with a colluding service provider. As the user then shows the credential directly to the service provider after holding the credential for a user-specified amount of time unknown to the validator, the validator does not mediate the usage of the credential. Validators sign attributes blindly unless it involves token transfer, and so they cannot link users to service providers. Given that double-spends are blinded before being recorded on the Nym blockchain, the validators and other third parties cannot link a particular double-spend to the creation of a credential. As detailed earlier, the flow of tokens in and out of the smart contract are not directly correlated with the amount of token in a credential or time it was spent, or even redeemed by a service provider. A user may link themselves to a service provider via showing attributes, but that is under control of the user and beyond the scope of the network. At the end of epoch, the user refreshes their credential, which is especially needed if the set of validators has changed, and re-randomizes it to preserve privacy.

VI. Evaluation and Comparison

We implemented the entire Nym credential framework and required actors in the network, including transfer of a NYM Token on the external Ethereum blockchain and the usage of Tendermint for the Nym blockchain.[8] The implementation consists of 739 lines of Go for the validators, including 382 lines for the IdP, 1518 lines for the user client, and 1892 lines of a generic zero-knowledge proof library. The underlying cryptographic and elliptic curve library used is *Milagra* due to its native support of pairing-friendly curve BLS12-381 for Go, although for future work it should be noted that this library is not highly performant as competitors in languages like Rust. For testing, a loop framework was implemented in order to limit computations to a single core. All testing was done on an Intel i7-9700K, 32GB DDR4-2666 RAM, with timing profiles taken using Go's native performance *testing* benchmark framework with compiler optimizations disabled and benchmarks stored in local variables. Servers were hosted on Amazon EC2 dedicated hosts model C3. All benchmarks are reported in milliseconds (ms) over an average of 100 runs of the protocol.

One could think that ZCash can be used for identity. While simply using ZCash would allow users private payment for services, ZCash is not an authentication credential and so would not be able to provide extensible structured attributes needed

[8]https://github.com/nymtech/nym-validator

for service operations and identity management easily (only the 'encrypted memo' field), and proof generation is still expensive despite their being recent progress [16]. Furthermore, the timing information of ZCash is also revealed, as currently no ZCash wallets integrate an anonymous channel like a mixnet. Nym credentials allow authentication for finite periods of time (epochs), where the credential is created before the epoch starts, sent to the smart contract and stored for the epoch (breaking the timing correlation between credential issuance and usage), and refreshed if not used during an epoch so it may be used in future epochs.

Nym credentials were tested for performance. First, in terms of key generation there was slightly worse than linear scaling with the number of validators, as would be expected, with the time for key generation going over 1 second when the number of validators increased to 70, taking up to 1750 ms for 100 validators. Given in Figure 2, validators blindly signing public attributes scales well, remaining under 200 ms for 100 public attributes. In contrast blindly signing private attributes scales worse, taking 600 ms for 100 private attributes. The user-side aggregating partial credentials to form a single attribute Nym credential, given in Figure 3, scales very similarly to key generation, taking up to 350 ms for 100 validators. In contrast, client-side credential re-randomization is basically constant in terms of number of attributes with the very fast speed of 2.3 ms. As shown in Figure 4, the showing of attributes scales linearly as expected, taking up to 300 ms for 100 private attributes. The verification of credentials takes only 44 ms per verification, of which 10 ms is required for cryptographic operations and the remaining for network communication to the validators. Nym credentials are in general competitive with some of the deployed zkSNARK systems in generation, whose generic time to generate is often 1 to 2s. Verification is close to equivalent to zkSNARKS and slightly slower than Algebraic MACs showing time of 29ms and 38 ms for issuing the credential [7]. Overall, the comparison between Coconut-based Nym credentials and the family of zkSNARK systems can be thought of as the trading of the availability of the validators versus the trusted setup of zkSNARKS. Precise comparisons would require a test suite over the same circuit and gates, which we leave for future work.

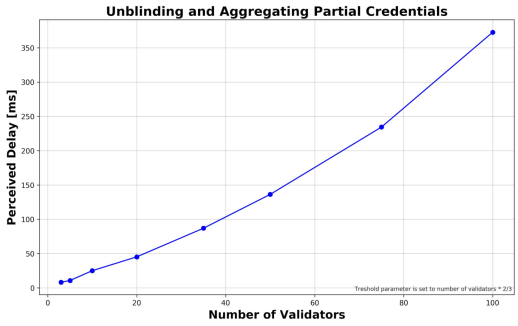

Fig. 2: User Credential Aggregation Performance

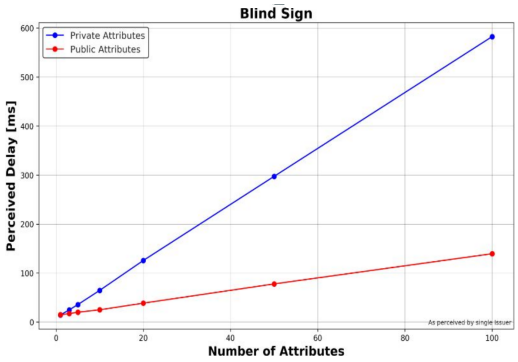

Fig. 3: Validator Signing

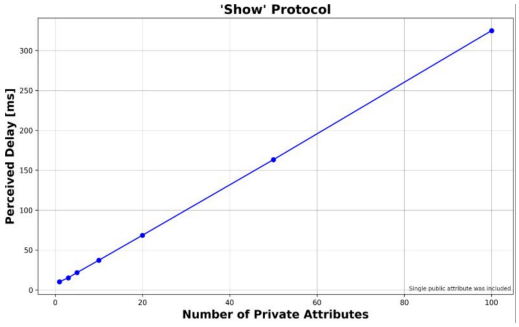

Fig. 4: User Selective Disclosure

The comparison of Nym versus other systems surveyed in Section II is given in Table 1. The general conclusion is that Nym credentials provides better privacy than widely deployed OAuth solutions as well as blockchain identity solutions under standardization. It is equivalent in privacy to systems such as UnlimitID, but provides better properties in terms of availability and resistance to byzantine failures, although at the cost of increasing

Property	OAuth	W3C DID	UnlimitID	Coconut	Nym
Undetectability	✗	✗	✗	✓	✓
Unlinkability	✗	✗	✓	✓	✓
Selective attributes	✓	✓	✓	✓	✓
Unforgability	✗	✓	✓	✓	✓
Persistent Accounts	✓	✓	✓	✓	✓
Sybil Resistance	✓	✗	✓	✗	✓
Extensibility	✓	✓	✓	✗	✓
Availability	✗	✓	✗	✓	✓
Decentralization	✗	✓	✗	✓	✓

TABLE I: Comparison of Privacy-Preserving Federated Identity Systems

the number of components (like validators) and so increasing network latency. Even with 100 validators, Nym credentials operate in under a second for all major operations involving validators with linear scaling or constant time operations for operations that do not involve communication with validators.

VII. Conclusion

This paper outlines Nym credentials for privacy-preserving identity management based on cryptographic credentials that both offer verified attributes and can be issued and verified in a decentralized and public setting. As opposed to systems based on zkSNARKS [16], Nym credentials are based on anonymous authentication credentials. Previous work in anonymous authentication credentials has been either privacy-preserving and centralized in issuing and verification [7] or centralized in issuing [11] or without extensibility. To a large extent, Nym credentials are an application of the Coconut [2] credential scheme, where issuing and verification is decentralized, to identity management. In a nutshell, the main difference between Coconut [2] and Nym credentials is that a single key is required per multiple attributes by Nym while Coconut requires each attribute to have its own key and be agreed upon by all the issuing authorities and users. This design only works for homogeneous services where all attributes can be "hard-coded" in advance, but that does not make sense in terms of open-ended federated identity management. Nym bypasses this issue with Coconut by requiring the linking of multiple per-attribute constant-sized Coconut credentials (that also contain mandatory information) into a single multi-attribute Nym credential. Nym credentials give users all the functionality of federated identity while enabling privacy and decentralization. The following design design goals have been achieved:

- **Undetectability:** Nym credentials are undetectable as the user mediates their transfer to the service provider directly, not via an identity provider or validator.
- **Unlinkability:** Due to re-randomization, Nym credentials are unlinkable.
- **Selective attributes disclosure:** Nym uses the Coconut signature scheme to accomplish selective disclosure.
- **Unforgability:** Only the user can present a NIZK of possession as a private attribute in a Nym credential.
- **Persistent accounts:** A secret, such as the aforementioned NIZK proof of key ownership, can be used for persistent accounts.
- **Sybil resistance:** Private attributes showing proof of a service-provided secret or registered key material prevents certain kinds of sybil attacks and sybil attacks are discouraged by costing tokens.
- **Extensibility:** Nym credentials can be issued with service and user-defined attributes dynamically, without new verification keys for new attributes.
- **Availability:** As multiple validators with a threshold signature are used, it can withstand byzantine faults of up to λ of validators.
- **Decentralized:** A single trusted third parties is not required via the use of blockchains and threshold cryptography.

There are number of areas for improvement and open avenues for future research in the Nym framework. The decentralized aspects of the Nym framework so far assume a set number of validators using threshold signatures, but the Nym framework should evolve over time to allow new validators to join the set of existing validators or non-performing validators to be removed. This could be accomplished by an incentive system that allows permissionless joining and removal of validators [17]. Regarding NYM tokens, future work will need to add spend and merge operations to the Coconut scheme to allow it to work without reducing the anonymity set by having different services charge different amounts and these amounts being paid in denominations. Lastly, the Nym framework should be adapted to not only track the usage of service providers in the Nym network, but reward an underlying anonymous overlay network like a mixnet capable of preventing network-level attacks and obscuring metadata.

Identity seems to directly oppose privacy via the linking of attributes to a user, in essence the *reduction* of a human to their attributes and their authorized actions. Identity then becomes a nexus of control over human life. Privacy, on the other hand, appears in contradiction to identity via the unlinkability of users to attributes and actions. As put by Eric Hughes, "Privacy is the power to selectively reveal oneself to the world."[9] The solution put forward by Nym credentials is to selectively allow the linking and unlinking of identity without any centralized control. One may even be recognized without revealing any attributes at all, being such as it is.

REFERENCES

[1] J. R. Douceur, "The sybil attack," in *International workshop on peer-to-peer systems*. Springer, 2002, pp. 251–260.

[2] A. Sonnino, M. Al-Bassam, S. Bano, and S. M. G. Danezis, "Coconut: Threshold issuance selective disclosure credentials with applications to distributed ledgers," in *Proceedings of the Network and Distributed System Security Symposium - NDSS'19*. Internet Society, February 2019. [Online]. Available: https://arxiv.org/abs/1802.07344

[3] H. Halpin and B. Cook, "Federated identity as capabilities," in *Annual Privacy Forum*, 2012, pp. 125–139.

[4] D. Hardt, "The OAuth 2.0 authorization framework," 2012, https://tools.ietf.org/html/rfc6749.

[5] D. Fett, R. Küsters, and G. Schmitz, "A comprehensive formal security analysis of OAuth 2.0," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2016, pp. 1204–1215.

[6] J. Maheswaran, D. I. Wolinsky, and B. Ford, "Cryptobook: An architecture for privacy preserving online identities," in *Proceedings of the Twelfth ACM Workshop on Hot Topics in Networks*. ACM, 2013, p. 14.

[7] M. Isaakidis, H. Halpin, and G. Danezis, "UnlimitID: Privacy-preserving federated identity management using algebraic MACs," in *Proceedings of the 2016 ACM on Workshop on Privacy in the Electronic Society*. ACM, 2016, pp. 139–142.

[8] P. Dunphy and F. A. Petitcolas, "A first look at identity management schemes on the blockchain," *IEEE Security & Privacy*, vol. 16, no. 4, pp. 20–29, 2018.

[9] D. Reed, M. Sporny, and M. Sabadello, "Decentralized Identifiers (DIDs)," 2019, https://w3c-ccg.github.io/did-spec/.

[10] D. Chaum, "Security without identification: Transaction systems to make big brother obsolete," *Communications of the ACM*, vol. 28, no. 10, pp. 1030–1044, 1985.

[11] F. Baldimtsi and A. Lysyanskaya, "Anonymous credentials light," in *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*. ACM, 2013, pp. 1087–1098.

[12] J. Camenisch and E. Van Herreweghen, "Design and implementation of the Idemix anonymous credential system," in *Proceedings of the 9th ACM conference on Computer and communications security*, 2002, pp. 21–30.

[13] C. Garman, M. Green, and I. Miers, "Decentralized anonymous credentials," in *Proceedings of the Network and Distributed System Security Symposium - NDSS'14*. Internet Society, February 2014. [Online]. Available: https://www.ndss-symposium.org/ndss2014/programme/decentralized-anonymous-credentials/

[14] C. Troncoso, M. Isaakidis, G. Danezis, and H. Halpin, "Systematizing decentralization and privacy: Lessons from 15 years of research and deployments," *Proceedings on Privacy Enhancing Technologies*, vol. 2017, no. 4, pp. 404–426, 2017.

[15] A. M. Piotrowska, J. Hayes, T. Elahi, S. Meiser, and G. Danezis, "The Loopix anonymity system," in *26th USENIX Security Symposium (USENIX Security 17)*, 2017, pp. 1199–1216.

[16] E. B. Sasson, A. Chiesa, C. Garman, M. Green, I. Miers, E. Tromer, and M. Virza, "Zerocash: Decentralized anonymous payments from bitcoin," in *Security and Privacy (SP), 2014 IEEE Symposium on*. IEEE, 2014, pp. 459–474.

[17] A. Kiayias, A. Russell, B. David, and R. Oliynykov, "Ouroboros: A provably secure proof-of-stake blockchain protocol," in *Annual International Cryptology Conference*. Springer, 2017, pp. 357–388.

[9] http://www.activism.net/cypherpunk/manifesto.html