Secretation: Toward a Decentralised Identity and Verifiable Credentials Based Scalable and Decentralised Secret Management Solution

Zakwan Jaroucheh and Iván Abellán Álvarez

School of Computing, Edinburgh Napier University, Edinburgh, UK
z.jaroucheh@napier.ac.uk, ivan.abellan@pm.me

***Abstract*—Secrets such as passwords, encryption keys, and certificates are used to assist in protecting access to resources such as computing devices, customer data and other information. Unauthorised access to resources can cause significant disruption and/or disastrous consequences. Given the importance of protecting these secrets to the security and privacy of many software systems, many solutions have been proposed. These solutions take two main directions: either securely store the secret and implement an access control mechanism, or divide the secret into a set of shares and distribute them in different machines (such as the Shamir's secret sharing approach or multi-party computation MPC). However, apart from the MPC approach, they all share the same limitation: once the consumer receives the secret, it can be leaked and be used by any malicious actor. We believe that the secret management should not be centralised and that the secret should never be sent to the receiver. Therefore, in this paper we propose, Secretation, a new approach for managing the secrets in a decentralised way by leveraging decentralised identity concepts such as verifiable credential technologies, password-authenticated key exchange protocols and multi-party computation. The result is a more scalable and secure solution that significantly reduces the risk of leaking the secrets.**

I. Introduction

Secrets are sensitive data that can only be accessed by a limited number of users, applications, or processes. A secret can be virtually any sensitive information that we need to restrict access to, such as user credentials, cryptographic keys, API keys and configuration files. Secrets are used not only to authenticate users but also to authenticate applications, services, devices and any "thing" in the IoT world. Lack of security and proper access management can lead to devastating consequences, including an increased risk of cyber attacks and theft of certificates or user credentials, such as by Man-in-the-Middle (MITM) attacks and spoofing. Many solutions and approaches have been already implemented and proposed. These solutions can be categorised into two main categories:

- Centralised secret management systems where the secrets are stored encrypted in central storage.
- Distributed secret management systems where the secrets are divided into shares and stored several internal machines or in the cloud.

Most of the existing solutions implement a kind of access control mechanism to restrict access to the secrets to specific entities. However, they share the following limitations: 1) Most of them are based on role-based access control which is not adequate for fine-tuned access control [1]. 2) Some of them relies on a third-party (e.g. in the cloud) to stores the secrets which increase the risk of leakage. 3) The common factor of all of them (except the solutions based on multi-party computations such as [2]) is that the eligible secret consumer is able to access the actual (non-encrypted) value of the secret. We argue here that this poses a security risk because any malicious insider (who is eligible to access the secret) can leak the secret or use it maliciously. Rotating the secret regularly could indeed mitigate that risk, but it is not enough, and it is not adequate in all scenarios. For example, rotating an encryption key requires re-encrypting the encrypted data using the new key which might require an outage until all data are entirely re-encrypted.

In order to remedy these limitations, we propose a new secret management system that takes another perspective when it comes to protecting secrets. Instead of securely storing the secrets and delivering the secrets in their plain forms to the recipients, our approach is to create an encapsulation layer that protects the secret in a way that the secret must not leave its location at any point in time to the recipients. Instead, for each secret, there will be an agent responsible for fulfilling the requests to access the secret. We call these requests Usage Requests (URs). In this respect, the requester should specify in the request how they intend to use the secret and in which context. Once all information is available to the agent - the only entity who knows the secret value - they can trigger the action specified in the UR, receive the results and send them back to the requester. For that objective, the requester must be able to prove to the secret agent that they meet the secret policy available to the agent and specified during the secret creation.

978-0-7381-1420-0/21/$31.00 ©2021 IEEE

II. Background

A. Multi-party computation

Multi-party computation is, at least, a two-party protocol that protects participants privacy and data security. Two parties aim to compute the output of a function without revealing their inputs and which first was described by Yao [3]. Modern MPC protocols are complex designs in the sense that rely on many cryptographic tools and assumptions to achieve their objective. Therefore, garbled circuits, oblivious transfers and zero-knowledge proofs are some of the preferred primitives to use. Garbled circuits notion refers to a cryptographic technique that encrypts inputs with different keys to avoid computing functions over plain data, thus providing privacy, through a Boolean circuit. A formalisation attempt on garbled circuits or garbling schemes is provided in Bellare et al. [4]. An oblivious transfer, introduced by Rabin whose transcript is found in [5], is a message exchange cryptographic tool to privately output the result of one of the sender's inputs with no external trusted party involved. A zero-knowledge proof is defined as a probabilistic cryptographic function that allows for a verifier to verify that a concrete construction (e.g. a proof) is valid over a value without requiring the prover to expose the private value [6].

Due to inefficiencies and computational power limits, many MPC approaches are not yet implementable except for digital signatures schemes. Two-party digital signature schemes allow for computing a joint signature over a message between two participants without ever revealing their private keys nor reconstructing private keys in the clear. These techniques are widely known as threshold cryptography. Common MPC signing algorithm includes 2-party ECDSA by Lindell [7] and 2-party Schnorr signatures by Nicolosi et al. [8].

B. Password-Authenticated Key Exchange

1) Shared secrets: Bellovin and Merritt [9] presented the first Key exchange protocol to be used with passwords, namely the Encrypted Key Exchange (EKE). EKE achieves secure authentication by leveraging symmetric and asymmetric cryptography. This work permits to establish an authenticated and secure channel between two parties from a weak password, which is resistant to dictionary attacks. The password is considered a shared secret which both sides need to know before running the protocol. Since the invention of EKE, many improvements have been proposed, for instance, SPEKE and DH-EKE. SPEKE presented by Jablon [10] is similar to EKE, but instead of using a fixed primitive base for the protocol, it uses a function to transform the password into one. DH-EKE by Steiner et al. [11] presents a modification on the EKE protocol to use the public key distribution system of Diffie and Hellman. By combining DH public key and passwords, both methods prevent man-in-the-middle and offline-attacks [10].

2) Asymmetric AKE: The technology evolved towards implementing asymmetric cryptography to avoid sharing secrets between parties which is the main concern of these protocols. Then, the Secure Remote Protocol, which is an Asymmetric Key Exchange (AKE) protocol, was introduced by Wu et al. [12]. SRP prevents from sharing a long term password by leveraging one-way functions as a verifier that creates a relationship between them. It is based on the random oracle model and mixing functions as well as a hashed-based MAC to confirm session keys. In general, such protocols provide forward secrecy but differs on the resistance to diverse attacks. This particular AKE instantiation is resistance to active attacks.

As it is not easy to model and proof security over Password-only Authenticated Key Exchange (PAKE) many proposals rely on digital signatures schemes to construct an AKE protocol, such as [13] and [14]. Sometimes it seems there is a trade-off between sharing secrets or relying on digital signatures to develop authenticated key exchange protocols. However, there are efforts in removing the dependence on PKI and maintain only the easy use of passwords to define secure authentication and key agreement protocols. Hao et al. [15] presents J-PAKE as an AKE protocol with no PKI requirements, although both parties share a secret password. This protocol prevents from sending the password in clear, thus follows a verified-based protocol, by using Schnorr signatures as a zero-knowledge proof tool providing resistance to both online and offline dictionary attacks.

Additionally, it is argued that pre-computation attacks on server compromise attacks are inevitable and that resistance to only offline attacks, without password file leaked, is sufficient. However, some research has shown that PAKE with additional server compromise attack resistance is possible and interesting, so no need to update user's passwords as shown in [16] and [17]. Gentry et al. [16] presents a mechanism to transform traditional PAKE into server compromise resistance protocols by relying on symmetric encryption and a digital signature scheme. Jarecki et al. [17] presented the first PKI-free PAKE protocol resistant to the same attack. Both methods are proved under the UC framework.

C. Public Key Infrastructure

Digital signatures schemes are used to prove identity, in which each participant generates a key-pair, in particular, a private key and a public key. Any encryption with a public key can only be decrypted with the corresponding private key. Likewise, any signature with the private key is only successfully verified with the corresponding public key. The public key system is powerful, yet introduces a key management problem. Private keys on PKI are sensitive information which represents ownership of identities in the digital world, and therefore they have to be protected. Any leak or breach on private keys introduces forgery and impersonation attacks. Key management is still a challenge; however, many different approaches are proposed and considered to mitigate and reduce risks. In general, solutions are in the range of securing private keys on secure enclaves or using secret sharing schemes. These approaches introduce some degree of complexity within a system, although the security significantly increases.

Recently, the introduction of Distributed Ledger Technologies (DLT) has sped up the use of PKI. Many systems are being proposed to rely on blockchain-like technologies to manage public keys, or public identifiers, and to communicate. DLT provide an immutable record of transactions and information exchange. Ownership of information is proved with digital signatures. Furthermore, DLT is a means of freely exchanging information between participants in a censorship resistance environment. Public identifiers are now shared openly while being managed by the blockchain underpinning the network. In general, private keys management tends to rely on physically restricted modules, such as secure enclaves, to reduce access and protect them.

Decentralised Identity (DID) is a concept in development that leverages DLT immutability and censorship resistance characteristics to set up uniquely and global identifiers for individuals identity. Anyone or anything is represented by a public identifier (a DID) which is securely stored and distributed in the DLT. Preventing, removing or altering identities in such systems are subject to costly computing power and consumption. Furthermore, DID introduces verifiable credentials as an assertion of attributes of a subject, which tries to imitate the identifiable and credentials systems of the physical world. However, this idea is not new because it resembles the certificates system, but the fact of having a powerful use case as self-sovereignty allows it to be used extensively.

D. Identity management

Before Blockchain, Identity management was addressed by reducing the number of identities a user may have. Identity services providers (IdP) are responsible for authenticating users, and therefore they may provide authentication services for users. New applications leverage third-parties to register and authenticate users rather than handling this process by themselves. First solutions to identity management were provided by Federated protocols which try to solve both authentication such as OpenID and authorisation such as UMA [18] or OAuth2.0 [19]. Generally, authorisation frameworks are accompanied by Identity service Providers (IdP) to tackle authentication. Although these solutions provide segregated tasks, and therefore individuals have increased control over their identities, it is not a fully decentralised approach. Hardjono [18] argues that a federated authentication and authorisation provides decentralised identity management because both tasks are separated within the system. This protocol relies on Identity service Providers to authenticate users, whereas Authorisation servers are responsible for defining and enforcing policies. There is a legal trust framework abided by all domains to agree on policy definition. Although it is discussed to be a decentralised approach to manage identity, there is no use of DLT nor consensus algorithm to agree on personal identities, besides, individuals do not own identities, but third-parties do.

III. Related work

A. Authentication and authorisation

There are many proposals for securing secrets, although yet none of this restricts clients access once it is granted. Generally, these approaches tackle either authentication and authorisation or secrets securitisation. It seems that combining and extending both systems is not widely explored. PKI, ZKP and IAM are different mechanisms to authenticate users within a system and therefore deny or grant access to them. Role-based access control is de facto the most extended granting access control allowing more higher degree of policy creation and access restriction compared to authentication only. It is based on a system that helps in handling user roles life-cycle such as create, assign, modify and remove. Any authenticated or authorised client still requires to be in possession of the secret to use it.

On the other hand, solutions protecting secrets are just methods or strategies and technologies to prevent unauthenticated or unauthorised participants from gaining access to these secrets. Many of these technologies are based on encryption systems and secret sharing schemes. Furthermore, these services are provided by cloud providers as SaaS, so any entity should hand out their data to be kept safely out of their networks.

Nonetheless, MPC is a secure form of computation in which data access is restricted to only the owner of these data. At this time, there are many specific solutions to multiple MPC problems, although practical usage is still limited by computational power. MPC is a so broader term that it is required to define problems with a particular specification to propose a satisfactory solution to them. Digital signatures are one of the first technologies that benefit from MPC settings. Current computational power alongside efficient algorithms is sufficient to compute two-party digital signature protocols.

Private keys of cryptocurrency systems can be protected by leveraging MPC signatures to decentralise trust required on a single system (e.g. Unbound Tech). Concretely, this technology allows for generating keys and signing messages with two complementary key shares owned by two different parties. Key shares are never combined, thus significantly reducing the attack surface on stolen credentials. This technology is easily combined with Hardware Secure Modules or cloud service providers without highly trusting third-parties. Other alternatives propose to leverage PKI to prove ownership and therefore authenticate users, and which private keys are locally stored (e.g. Beyond Identity) or based on cloud storage (e.g. Magic.Link). Traditional PKI based on certificates do not leverage the potential of DLT or DID; therefore, key management should be considered. Zero-knowledge proofs on passwords prevent the usage of the public-key system (e.g. NuID). HashiCorp relies on the extensively used IAM systems but has extended role-based access control to their own needs so that having more control and flexibility on policy definitions.

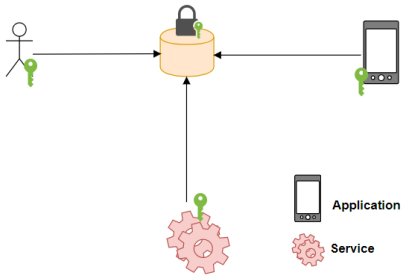

Fig. 1. The traditional setting

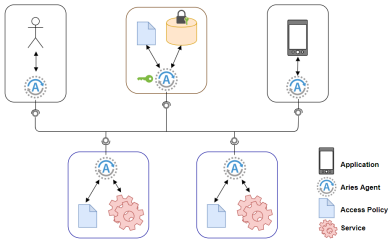

Fig. 2. The Secretation approach

IV. The Approach

Let us start with the scenario illustrated in Fig. 1 where we have a central database. Ideally, we should have a separate account with a specified set of privileges for each user of the database. In practice, the database admin creates a set of accounts, each of them is shared among a set of people/applications/services who meet some criteria (such as a set of roles). Having this access control leads to three issues: 1) it requires sharing and disclosing the secret (the database credential) to the recipients who potentially can leak that secret, 2) rotating these credentials becomes a challenge because this requires changing the credentials from the user and the database sides at the same time, and 3) the attacker will not be restricted to attack the database server in or get the required credential to access the information, they can attack any application/service that uses these credentials.

To remedy this situation, we believe in the idea that the secret should have one home location and that it should never leave that location. Database credentials should be located in the database server, and it should not leave that server. The question now is how the consumers will use these credentials. The answer is that the consumers should not receive the secrets; instead, they have to hand over their requests to an agent who is located on the same secret's machine. After being authenticated by the agent, it processes the requests on behalf of the consumers and returns the results as illustrated in Fig. 2. One of the advantages of this method is that the secret will be located in only one location, and that reduces the attack surface. Besides, as the agent is the only entity that has full access to the secrets, the agent should be able to rotate the secret regularly without the need to make any change in the application/services sides.

The problem now is how the agent authenticates the consumer. Here is where two approaches are proposed. One is based on passwords, precisely, a password-authenticated key exchange, whereas the other is based on the Decentralised Identity (DID). Concretely, OPAQUE is a user-centric password-only authentication protocol. It is the first PKI-free protocol secure against pre-computation attacks on *password file* leakage [17]. Particularly, the client generates a ciphertext, which the server stores, using a secure password that is computed by a two-party protocol between the client and server. With this authentication protocol, we keep the simplicity of passwords in front of users while providing secure authentication. Furthermore, any potential breach poses no risk on impersonation as the password is never disclosed. DID concept arise from distributed ledger technologies to leverage public key systems to identify and proof ownership of such identity. Public keys identifying agents are registered in DLT, so verifiers are capable of validating signatures over the identity of consumers.

In this respect, every entity (e.g. software component, application, service, agent) should register with another so that any two entities can communicate securely. For that aim, we need to encapsulate that entity with an agent that works as a representative of that entity (i.e. smart proxy). We differentiate here between two types of agents: 1) consumer agents (CAs) which acts on behalf of the consumers of services or secrets, and 2) provider agents (PAs) which act on behalf of a database, a service or even a secret.

The PA works as a smart gateway to grant or deny access to the corresponding protected entity (i.e. database, service, or secret). For that aim, we need to have an access policy in place. When the PA receives a request from the CA (on behalf of some application, for example), the PA checks if that agent fulfils the requirements of the access policy. The policy specifies the list of attributes the CAs should have and any other contextual information such as the request time. The PA is the only required entity to know the specific attributes, whereas CA should know which type of data or requirement is necessary for access control.

In order for the CA to proof to the PA that it holds the required attributes, it should get its attributes (credentials) signed by a trusted third-party. The access policy specifies the list of accepted public keys of the trusted third-parties for each attribute. Although it is necessary to rely on trusted third-parties to have attributes signed, the trust and risks that exist are reduced by leveraging multi-party computation signatures. Fig. 3 illustrates the communication protocol between the different agents regardless of authentication.

It is worth mentioning that the concept of granting access to the secret based on that policy provides a powerful and flexible mechanism which allows the software architecture to specify the access control rules based on the attributes of the consumers and on the specified contextual information. That is much powerful that merely granting access based on

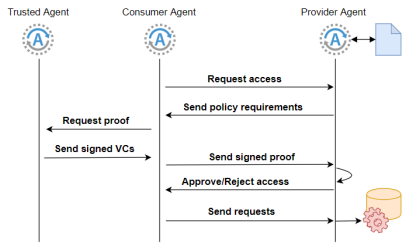

Fig. 3. The communication protocol between the agents

the role of the consumer. In addition, that would make the attacker's mission more challenging because, in the Secretation approach, it is not enough for the attacker to compromise the consumer account and impersonate it. The attacker must be able to "convince" the relevant trusted agents to sign its corresponding credentials in order for them to provide the proof to the PA. For instance, the policy can specify that the location of a consumer (location attribute) should be a specific city or a specific department location. The system can rely on a trusted service that can detect the location of the consumer (such as the mobile network operator or the door/building badge system) and sign the location information accordingly. This is similar to the multi-factor authentication concept, but here we do not have a centralised entity that manages the accounts; instead, we leverage a decentralised identity concept as verifiable credentials.

A. User life-cycle

We define how users, which are potential consumer agents, are created and managed in this system. For that reason, the user life-cycle is defined as 1) user and agent creation 2) user registration 3) credentials issuance and 4) revocation access. The life-cycle is dependent on the technology in place. Although there are slight differences in the process of managing consumer and provider agents, there are some similarities in the protocol for issuing and fulfilling policies.

Firstly, we propose to use an MPC signature scheme to increase the confidence in the user creation process. The PAKE approach would need an administrator and the current user to collaborate to create a valid agent. Therefore, each agent holds a signed credential of his identity. In the DID approach, the administrator and the user are responsible for creating key-pairs for each agent and registering the public key as identifies into a DLT. Agents in both methods are eligible for the user registration phase against provider agents. Furthermore, each time an agent is required to use its identity, it will need the collaboration server in order to compute signed messages.

Secondly, user registration stands for the process by which agents register to provider agents in order to access resources. This process is only needed for the OPAQUE approach, as PAKE protocols require, in general, a two-phase protocol, namely registration and login. The registration should be performed under an authenticated channel to assure that valid parties are being registered for later authentication. In general, this is approached by a legacy authentication system, but in our approach, we leverage digital signatures in an MPC setting to verify users. Conversely, in the DID approach agents are directly authenticated by the DID protocol and the registered public keys, which identifies them.

Thirdly, signed attributes for access control come in the form of verifiable credential. Verifiable credentials are a concept from the Decentralised Identity which represent signed assertions of the subject of the credential. Credentials are issued by trusted entities so that the public key of the issuer carries out verification. To reduce the confidence given to trusted entities, users register a key share through a two-party digital signature scheme. MPC signatures implicitly authenticate key shares owners with respect to a trusted entity. Therefore, credentials are issued only under the cooperation of both parties.

Finally, there are two ways of revoking access to resources in the password-only method 1) changing the attributes a consumer agent has to fulfil (e.g. modify the policy) or 2) dropping the authentication data of the OPAQUE protocol. The former approach let consumer agents authenticate, but the use of resources may be restricted, whereas the latter precludes consumer agents from authenticating against the provider agent. With regard DID approach, provider agents either create a deny list of public identifiers to prevent authentication or an allow list of eligible public identifiers.

B. An OPAQUE overview

Let us briefly present the authentication protocol we rely on which Jarecki et al. [17] presented, namely OPAQUE. It is an asymmetric PAKE meaning that the user password is never shared with the server. It is a two-party protocol in which the client and server engage in message exchange to create a mutually authenticated and encrypted channel. They first authenticate each other and then generate an ephemeral session key. It is defined by two phases, including registration and login. In registration, both run a function so that the client registers his authentication credentials. During login, the client authenticates with the use of a password, which discloses to him registered private and public keys to run a key exchange protocol in order to create a secure communication channel. Note that although saying someone registers a password, no password or hash at all is shared or stored on the server-side. Therefore, this secret should be only known by the client. This feature leads to a protocol resistant to pre-computation attacks.

In short, this protocol is built up from an Oblivious Pseudorandom function (OPRF) and an authenticated key exchange (AKE). OPRFs are a two-party protocol that allows computing cryptographically secure passwords from a low-entropy user password. Both the secret value used by the server and the password input by the user are not revealed to the other side. The OPRF output is only learned by the client. The concrete OPFR function of key k over input pwd is $F_k(pwd) = \mathsf{H}(pwd, \mathsf{H}'(pwd)^k)$, where H and H' represent a traditional hash function (e.g. SHA256) and a hash-to-curve function, respectively [17]. The generated random password

is used in the rest of the protocol. The construction of the protocol is followed by an AKE protocol such as HMQV or SIGMA. Any AKE protocol once combined with an OPRF leads to a strong asymmetric PAKE with KCI resistance [17].

*1) **Registration phase**:* This phase requires the user to run the registration over an authenticated channel to ensure only valid users. For example, users use their identity credential to send an authenticated registration request. The server validates the request with the corresponding public key, previously shared by the trusted server. For a detailed user identity creation refer to section IV-A.

Let G be an elliptic curve group of order q and g the generator. Both consumer and provider agents engage in the OPRF. The CA chooses a blinding factor $r \leftarrow_R \mathbb{Z}_q$ to hide the hash of his password when send it to the PA, whereas the PA chooses a random value $k \leftarrow_R \mathbb{Z}_q$ to be applied to the received value. The CA ends up with $rw \coloneqq \mathsf{H}(pwd, \mathsf{H}'(pwd)^k)$, where rw denotes a cryptographically secure password. Additionally, both perform a Diffie-Hellman key exchange such that:

$$\text{CA computes: } x \leftarrow_R \mathbb{Z}_q, X \coloneqq g^x$$

$$\text{PA computes: } y \leftarrow_R \mathbb{Z}_q, Y \coloneqq g^y$$

*2) **Login phase**:* The login phase is similar to the previous one as the OPRF deterministically generates the same random key if the user inputs the correct password. The random key is now used to decrypt the ciphertext which the PA has retrieved and sent back to the CA. Both run an authenticated key exchange protocol to obtain a session key. In particular, a 3-step SIGMA-I protocol because of the identity protection property [14]. First, each participant have an identity, in this case CA and PA, and a long-term key-pair from any digital signature scheme, for instance, denoted as $Sig(\cdot)$. Then, both perform a DH exponentiation to obtain a shared secret key k. A key derivation function is applied over k in order to get k_1 and k_2. One key is used to encrypt the channel, whereas the other to perform a MAC. Both exchange messages as follow:

$$\text{MSG 1: } X' \coloneqq g^{x'}$$

$$\text{MSG 2: } Y' \coloneqq g^{y'}$$

$$k_1\{Sig(X', Y'), PA, MAC_{k_2}(PA)\}$$

$$\text{MSG 3: } k_1\{Sig(X', Y'), CA, MAC_{k_2}(CA)\}$$

C. Authentication by DID

The introduction of blockchain and other DLT allowed for another approach to identity management. In this case, individuals are capable of managing their own identities which are ultimately verified by the DLT. Furthermore, such solutions solve the PKI management of public keys as they are stored and distributed by the DLT. This concept is widely known as Decentralised Identity or even Self-Sovereign Identity (SSI). Any pair of private-public key valid in the DLT represents the identity of and individual or entity. This cryptographic concept is represented in a URI-based syntax which depends on the DLT in which is being used. A Decentralised Identifier (DID) uniquely identifies a single entity that may have several corresponding verifiable credentials issued. Credentials assert characteristics from a subject while being tamper-proof and unforgeable by the use of digital signatures.

Now, many communities groups are building up the DID stack to extend and make it interoperable across several DLT and environments. They are proposing and developing standards to address issues related to privacy, security and portability, among others. W3C Decentralised Identifier [20] and Verifiable Credentials [21] working groups are standardising these new concepts of the SSI in DLT-like systems, respectively. They define and specify terms, their models and the necessary interactions between them.

In particular, each agent generates the corresponding DID according to the DLT in use (e.g. Bitcoin, Sovrin). Provider agents store a list of DIDs which are allowed to use the service. Every time a CA sends a signed request, the PA verifies it comes from a legitimate and valid source, thus authenticating the agent. This mechanism resembles traditional PKI authentication, although public keys are safely recorded in the DLT. Such technology allows creating self-sovereign identities, recording and storing public verifiers as well as enabling self-control of PII. Agents securely store their private keys which purpose is to identify themselves. DIDs provide easy to use PKI and identity management, therefore a ready to go authentication mechanism. DID is easily combined with verifiable credentials within the same system. Agents are uniquely identified and able to prove the ownership of their attributes through credentials. Note that we propose to strengthen this notion of verifiable credentials by having both parties, issuer and subject, cosigning the credential with an MPC digital signature setting. Instead of arguing in favour of trusted third-parties as issuers, both should have an understanding, thus a protocol, of collaborating to jointly compute credentials in order to reduce potential abuse or forgery.

V. Implementation

This system aims to simulate a real-world network that is composed of clients, several servers and data providers. This protocol allows for managing secrets within a network, and at the same time handling clients access rights. It is a solution that permits to have secrets in one place only and manage their life-cycle without interacting with other components of the network. Clients have usage rights of some data but not modification ones. All of these components of the protocol are modular and independent, so each one addresses a particular issue.

In Secretation, we create an agent for each type of secret we need to protect. The following are initial types we are focusing on: 1) A database agent and 2) a symmetric encryption key agent. The former focuses on protecting access to databases in such a way that the client request does not interact directly with the server. Instead, all request are passed to the database agents who, ultimately, pass the correct query. In this approach, clients are not required to know the string location of the database. Additionally, queries can be filtered

or analysed on the agent before passing it. The symmetric encryption agent is responsible for managing encryption keys to provide an encryption service. Encryption keys can be either locally stored or outsourced to cloud-service providers. Clients request the agent to encrypt or decrypt some data. The agent handles key generation, data encryption and decryption without permitting clients to interact with such secrets.

In particular, the solution built is based on an asymmetric password-authenticated key exchange protocol for the authentication, namely OPAQUE; an Attribute-Based Access Control approach along with verifiable credentials under an MPC digital signature scheme, which address policy creation and authorisation; and *usage requests* as a new approach to restrict access and manipulation rights to secrets.

Legacy authentication systems or identity credentials are used in the key generation process of both the Password-authenticated key exchange and the MPC signature scheme. This is set as the first step towards password registration in the proposed authentication mechanism as well as to distribute trust due to multi-party computation schemes. Both approaches make it possible to authenticate legitimate users within the network while reducing the exposure to intruders. Provider agents are able to create and manipulate the data they are holding as well as the access policies. Likewise, clients are capable of requesting verifiable credentials to issuers once they know the policy requirements. Both the issuer and client engage in a two-party signature scheme to generate verifiable credentials. Clients are authorised if their credentials satisfy the requirements imposed by the access policy. Note that, authorisation only happens if clients were previously authenticated against the server. Finally, consumer agents are capable of issuing usage request through an authenticated, authorised and encrypted channel.

A proof-of-concept validates the idea, which integrates *gopaque* and *blockchain-crypto-mpc* libraries. The former is a protocol implementation of the OPAQUE specification in [22], whereas the former is an implementation of multi-party computation digital signatures such as ECDSA and EdDSA. *Gopaque* acts as the main protocol between consumer agents and provider agents to both registration and login phases. CAs register passwords that are not stored nor seen by the service. *Blockchain-crypto-mpc* library allows for creating verifiable credentials that are signed by consumer agents and the issuer entity. It handles private key shares creation as well as signatures. Issuers are responsible for filling up credentials with appropriate consumer agent's data in JSON Web Token format as defined in RFC 7519 [23].

A. Protocol illustration - Use case scenario

Let us illustrate a scenario of an encryption service leveraging the PAKE approach. Note that the DID approach follows a similar behaviour in which an administrator handles DLT public identifiers registration and distribution. We describe three different participants, which are potential consumer agents in our network. There are two consumer agents in which one is eligible to use the service, whereas the other is just a valid user in the network but not eligible to use that service. A third participant is a malicious actor outside our control. Furthermore, there is a machine, known as the provider agent, providing several services from which one is an encryption service as well as two additional trusted servers. One of these servers is a collaborating server managed by an administrator, whereas the other is the issuance authority (which may be external to our network).

Both valid users will have undergone the user creation process, and therefore have their own identity certificates. This process resembles a legacy authentication system in order to create valid users within our network. Identity certificates try to replace legacy authentication systems when registering agents in multiple providers agents. Therefore, any legacy system is run only one time, instead of running them on each registration step of different protocols. As this process leverages MPC signatures, the collaborating server is able to share the corresponding users' public key with the encryption service and the issuance authority. Thus, these users will be registered in both services successfully. CAs register a *user* and a *password* with the PA through the OPAQUE protocol, see section IV-B. Registration concerning the issuance authority is defined by the creation process of key shares in the MPC digital signature scheme.

Upon authentication requests, both the consumer agent and the provider agent engage in the OPAQUE protocol. Only registered agents will be authenticated. Any attempt of a malicious actor trying to authenticate will fail as no password registration is stored in the service. After authentication, an ephemeral session key is generated which encrypts all traffic between CAs and the PA. After that, both agents request access to the encryption services, and therefore the PA presents the authorisation policy. This policy includes the requirements to be satisfied in order to use those specific services (e.g. department, role, location). In particular, the PA only presents the requirements but internally stores and handles the attributes assigned to each one. Each CA contacts the issuance authority and share the policy requirements to get their respective verifiable credentials. The issuance authority creates a credential based on their stored data regarding that CA. As it is a trusted entity, we assume the information set in the credential is correct. By leveraging MPC signatures both jointly compute a signature over the credential. As we have defined two different users, both have different verifiable credentials meaning that their attributes are different (e.g. distinct departments).

CAs hand over their credentials to the PA through the authenticated and encrypted channel. The PA validates the signature and checks the attributes in it with the attributes defined in the access policy. Only one CA have the access granted as having presented a matching credential. The unauthorised agent has the access denied, and therefore has two options 1) end session or 2) request access to another service that may be eligible to use. Note that credentials only have to be requested one time to the issuance authority if the information associated with that agent not changes over time. Finally, the authenticated and authorised CA is eligible to send usage

requests indicating which data has to be used in the service. In particular, the CA specifies either an encryption or decryption service. Upon receiving a usage request, the PA processes the request, in this case, the PA encrypts or decrypts data. Then, it returns either a ciphertext or a plaintext to the CA.

B. Corrupt users

As in any system, some information is susceptible to be compromised. In this setup, agents have several sensitive information to be kept safe. Depending on the approach taken agents may have a different kind of private information. Regardless of the authentication schema, agents have identity certificates, which identifies and validates their eligibility within the network. Agents hold either a DID key-pair or an internally valid key-pair share as specified in the creation step of the user life-cycle shown in section IV-A. Furthermore, they hold private key shares with regard to issuance authorities to create and manage verifiable credentials. Additionally, in the OPAQUE approach users should memorise or keep a password in order to use the authentication protocol.

Instead of managing private keys to compute digital signatures which upon compromise impersonation is possible, they manage key shares in the MPC setting. The advantage is that if a key share is compromised, credentials forgery is not possible as an honest entity still controls the other key share, for instance, the issuance authority. Identity credentials prevent unidentified agents from registering to any service. If all the key shares are compromised, a malicious actor may impersonate an agent by getting his credentials, but forgery is still not possible. If DID key shares are exposed, then an unregistered user will authenticate, whereas exposing key shares concerning an issuer will reveal credentials to an unintended party and lead to a possible impersonation. A password leak on OPAQUE leads to successful authentication, although the access control still has to be passed. Note that for a successful attack, most of the secrets need to be breached in order to bypass authentication and authorisation. Nonetheless, secrets of provider agents are not accessible due to the usage request proposal, introduced in section I.

Passwords and DID key shares prevent unauthenticated actors from accessing a service. Identity credentials and key shares identify agents and prevent impersonation and forgery, respectively. If a key share is leaked and someone tries to use it, their identity will not match with the identities handled by the issuance authority. Conversely, if the identity is compromised, but the key share is not controlled, then no valid, verifiable credentials over that entity will be generated. Note that incorrect paired key shares lead to invalid digital signatures.

VI. Evaluation

We do not believe that the central server is the best option for the secret management system. It is better to extend the storage of secrets to multiple servers in order to reduce the risk of exposure. In the Secretation approach, we take the decentralised approach to manage the secrets. The agent can be configured to log the secret usages to a central server for auditing purposes. For that reason, multiple specific agents are proposed depending on the use case they address, as shown in section V.

Several tests show that the protocol is lightweight, although having a significant degree of message complexity due to the nature of two-party protocols. Both authentication and authorisation process happens rapidly. However, getting issued credentials from third-parties may take longer depending on the issuer capabilities. Any of the verification processes are a set of computations that are performed on the provider agent side. Worth mentioning that each time a consumer agent is willing to make use of usage requests, this has to send a request even if using the same resource every time. That construction limits the number of executions a CA can make in a limited time but increases the security significantly. Although we leverage an MPC digital signature scheme to sign credentials which has an added cost in the form of computation and message exchange, key generation and signatures are performed in the range of milliseconds.

Known attacks on DLT may cause a temporary denial of service on a peer's registration process. For instance, an attacker could intercept and manipulate transactions such that the resulting identifier is different from the original. If the maliciously crafted transaction is confirmed into the blockchain, then the originally intended transaction will be rejected as double spend, this is known as a transaction malleability attack [24].

VII. Conclusion

This solution presents a new way of thinking towards storage and usage of sensitive information in which leaks poses a costly risk. Secretation proposes a new protocol which combines authentication, access control and multi-party computation schemes to offer a simple yet complete, flexible and secure secret management. It provides a non-friction and stand-alone technology, no dependencies needed to external entities, easy to use by users and complete and flexible by administrators. Password-authenticated key exchange provides mutual client-server authentication, whereas attribute-based access controls and verifiable credentials allow for a flexible alternative to controlling access conditions and producing unforgeable authorisation credentials. Alternatively, the Decentralised Identity in concert with multi-party computation digital signatures are leveraged to manage keys and authenticate users securely. Two-party digital signatures schemes reduce the attack surface on PKI and ease the key management while offering more controllable credentials issuing. Additionally, usage request, a delegated system approach, gives only authenticated and authorised users permission to use resources but limit their direct interaction and manipulation. With this approach, secrets do not leave the service machine. Registration processes prevent unintended parties to be authenticated, while issuance authorities and verifiable credentials ensure attributes not to be forged or faked, and therefore a reliable and strong access control.

REFERENCES

[1] D. R. Kuhn, E. J. Coyne, and T. R. Weil, "Adding attributes to role-based access control," *Computer*, vol. 43, no. 6, pp. 79–81, 2010.

[2] E. Syta, I. Tamas, D. Visher, D. I. Wolinsky, P. Jovanovic, L. Gasser, N. Gailly, I. Khoffi, and B. Ford, "Keeping authorities" honest or bust" with decentralized witness cosigning," in *2016 IEEE Symposium on Security and Privacy (SP)*. Ieee, 2016, pp. 526–545.

[3] A. C. Yao, "Protocols for secure computations," in *23rd annual symposium on foundations of computer science (sfcs 1982)*. IEEE, 1982, pp. 160–164.

[4] M. Bellare, V. T. Hoang, and P. Rogaway, "Foundations of garbled circuits," in *Proceedings of the 2012 ACM conference on Computer and communications security*, 2012, pp. 784–796.

[5] M. O. Rabin, "How to exchange secrets with oblivious transfer." *IACR Cryptol. ePrint Arch.*, vol. 2005, no. 187, 1981.

[6] U. Feige, A. Fiat, and A. Shamir, "Zero-knowledge proofs of identity," *Journal of cryptology*, vol. 1, no. 2, pp. 77–94, 1988.

[7] Y. Lindell, "Fast secure two-party ecdsa signing," in *Annual International Cryptology Conference*. Springer, 2017, pp. 613–644.

[8] A. Nicolosi, M. N. Krohn, Y. Dodis, and D. Mazieres, "Proactive two-party signatures for user authentication." in *NDSS*, 2003.

[9] S. M. Bellovin and M. Merritt, "Encrypted key exchange: Password-based protocols secure against dictionary attacks," in *Proceedings 1992 IEEE Computer Society Symposium on Research in Security and Privacy*, 1992.

[10] D. P. Jablon, "Strong password-only authenticated key exchange," *ACM SIGCOMM Computer Communication Review*, vol. 26, no. 5, pp. 5–26, 1996.

[11] M. Steiner, G. Tsudik, and M. Waidner, "Refinement and extension of encrypted key exchange," *ACM SIGOPS Operating Systems Review*, vol. 29, no. 3, pp. 22–30, 1995.

[12] T. D. Wu *et al.*, "The secure remote password protocol." in *NDSS*, vol. 98. Citeseer, 1998, pp. 97–111.

[13] I. R. Jeong, J. Katz, and D. H. Lee, "One-round protocols for two-party authenticated key exchange," in *International conference on applied cryptography and network security*. Springer, 2004, pp. 220–232.

[14] H. Krawczyk, "Sigma: The 'sign-and-mac'approach to authenticated diffie-hellman and its use in the ike protocols," in *Annual International Cryptology Conference*. Springer, 2003, pp. 400–425.

[15] F. Hao and P. Ryan, "J-pake: authenticated key exchange without pki," in *Transactions on computational science XI*. Springer, 2010, pp. 192–206.

[16] C. Gentry, P. MacKenzie, and Z. Ramzan, "A method for making password-based key exchange resilient to server compromise," in *Annual International Cryptology Conference*. Springer, 2006, pp. 142–159.

[17] S. Jarecki, H. Krawczyk, and J. Xu, "Opaque: an asymmetric pake protocol secure against pre-computation attacks," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2018, pp. 456–486.

[18] T. Hardjono, "Federated authorization over access to personal data for decentralized identity management," *IEEE Communications Standards Magazine*, vol. 3, no. 4, pp. 32–38, 2019.

[19] D. Hardt *et al.*, "The oauth 2.0 authorization framework," RFC 6749, October, Tech. Rep., 2012.

[20] D. Reed, M. Sporny, D. Longley, C. Allen, R. Grant, M. Sabadello, and J. Holt, "Decentralized identifiers (dids) v1.0," *World Wide Web Consortium*, 2020. [Online]. Available: https://www.w3.org/TR/did-core/

[21] M. Sporny, D. Longley, and D. Chadwick, "Verifiable credentials data model 1.0," *World Wide Web Consortium*, 2019. [Online]. Available: https://www.w3.org/TR/vc-data-model/

[22] H. Krawczyk, "The opaque asymmetric pake protocol," Internet Engineering Task Force, Internet-Draft, 2020. [Online]. Available: https://tools.ietf.org/html/draft-krawczyk-cfrg-opaque-06

[23] M. Jones, N. Sakimura, and J. Bradley, "Rfc 7519: Token web json (jwt)," 2015. [Online]. Available: https://tools.ietf.org/html/rfc7519

[24] C. Decker and R. Wattenhofer, "Bitcoin transaction malleability and mtgox," in *European Symposium on Research in Computer Security*. Springer, 2014, pp. 313–326.