TrustVault: A privacy-first data wallet for the European Blockchain Services Infrastructure

Sharif Jacobino
Department of Software Technology
Distributed Systems
Faculty of Electrical Engineering, Mathematics & Computer Science
Delft University of Technology

Johan Pouwelse
Department of Software Technology
Distributed Systems
Faculty of Electrical Engineering, Mathematics & Computer Science
Delft University of Technology

I. Introduction

Internet users today have very little control over where and how their data is stored and used online. Big Tech companies store gigabytes of data about you, and know which online services you use [1]. User data is an extremely valuable asset and is the main source of income for such companies. Billions of people rely on Big Tech monopolies to store their data and voluntarily give up control and ownership over that data. Much of this data is deeply personal and valuable, such as intimate photos of our friends and family. Public and policy trust in Big Tech has been breaking down in recent years (also called the ”techlash”) following major scandals, rampant misinformation campaigns, and a perceived consolidation of power [2]. Nearly five decades after the invention of public key cryptography, we still lack a good solution for people to manage their digital identity and efficiently share encrypted data directly with each other, certainly at a massive scale. There are various movements aiming at halting the power of Big Tech and giving back control to the users. These movements are powered by technologies like blockchains and self-sovereign identity which promise to improve the way we interact with online services and with each other.

The European Commission is ramping up its efforts for bringing transformation in the digital sphere with projects such as ”Path to the Digital Decade”. One of their goals is to improve the way citizens, businesses and public administrations share information and trust each other, and simplify verification processes for cross-border services using blockchain technology [3]. Its proposed solution to reduce our reliance on Big Tech is the European Blockchain Services Infrastructure (EBSI). As at May 2022, there was €57 million in funding for large scale EBSI trials [4]. EBSI uses verifiable credentials to reduce the time and cost of verifying the authenticity of documents and information shared on the network. Each European citizen will have its own digital wallet to interact with EBSI.

This work aims to answer one question: *How can we give user back control of their own identity and their data?* We want to accelerate the European identity and data self-sovereignty movement by providing a EBSI-certified data wallet with advance data sharing capabilities. We show that European Commission’s EBSI initiative is a viable way to give control to the people. It is possible to take back ownership of digital identity and data. It is possible to have a fairer and more competitive system than the for-profit infrastructure of Big Tech, that is public, transparent, and open source.

The contribution of this work is a proof of concept called TrustVault: A privacy-first data wallet deployed on the TrustChain Super App. TrustVault consists of secure data vault and an EBSI conformant wallet. The data vault stores the users data locally and provides fine-grained access control for the stored files and the wallet exchanges credentials on the EBSI network. These verifiable credentials contain attribute claims that function as access tokens to the data vault. Using verifiable credentials as a basis for attribute-based access control for personal data storage is a novel concept that extends the notion of self-sovereignty over personal identity to personal data. This proof of concept is a photo sharing social app that demonstrates TrustVault’s ability to be used for zero-server applications. Users connect directly to your TrustVault and their credentials are automatically matched against your predefined access policies. Users only see the photos that you allow them to see. Our openness-by-design ecosystem encourages permissionless innovation and competition. Anyone is able to develop new applications that can interact with data in your TrustVault.

In section IV related work is discussed.

II. System Architecture and Design

A. Architecture

TrustVault is a component of the the TrustChain Super App and uses the IPv8 protocol for peer-2-peer communication. There is a specific IPv8 overlay for discovering and interacting with peers using TrustVault. TrustVault uses the Android app-specific directories to store data. Every file has a unique path down the a directory tree starting from the root directory. The user is able to create, delete and move files and folders, much like a traditional file system. The wallet is designed to support EBSI's Verifiable Credentials lifecycle [5]. As a holder, the wallet is capable of generating EBSI did's, store credentials and key material, request new credentials from issuers and present credentials to verifiers. As a verifier, the wallet is capable of verifying credentials and issuers.

IG-SSI?

B. Access control

Files and folders, including the root folder, have an associated meta-data file that includes the access policy rules. To access a file, every policy along the file's path must be satisfied. Practically this means that policies are inherited from parent folders. An effective way of setting access policies is to have minimal or no restrictions on the root folder and have increasingly specific and restrictive policies for sub-folders.

An access policy is a binary boolean expression tree and the leaves are attribute rules that are evaluated at access time. Attribute rules are triplets in the form of $(attribute, operator, value)$. Figure 1 depicts the access policy: *Age* $\geq$ *18 AND (AlumniOf TU Delft OR isFriend)*. To satisfy this policy, the requester has to present a verifiable credential that asserts that their age is 18+, e.g. a government ID with a predicate proof over the age, and either a proof-of-enrolment from the TU Delft or a friendship credential from the TrustVault's owner. The authenticity of the credential is verified by the wallet and then evaluated against the policy tree.

The requester first makes an accessible-files-request that contains a set of verifiable credentials. The TrustVault returns a directory sub-tree consisting of the paths with satisfied access policies along with a randomly generated session token. The directory sub-tree is used to dynamically populate the *peer vault view* on the requesting device. The actual files are retrieved once their parent folder is opened on the requesting device to avoid wasting bandwidth. File requests include the provided session token so that whole credentials don't need to be sent and verified with each file request. Session tokens are cached on the serving TrustVault, mapped to the corresponding sub-tree, with an expiry time that can be extended with each new request. A request with an expired session token fails and the requester is required to make a new accessible-files-request. The session token to sub-tree mapping ensures that no files are served that aren't covered by the original credentials. The user interface lets the user add or remove nodes to the policy tree. Additionally, the user can define read+write access policies or separate read and write policies for more fine-grained control.

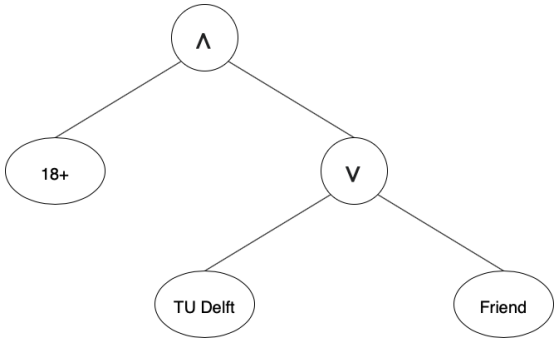

Fig. 1. Access policy tree

C. Self-issued credentials

Using verifiable credentials enables the requesting party to selectively disclose attributes, and allows the TrustVault to verify the authenticity of the credential, including verifying if the credential was issued by the controller of the vault. Self-issued credentials (SIC) can serve a similar but more expressive function than friend connections in traditional social networks. Extra attributes can be added to SICs to make a distinction between different types of friendships. These distinction can be used to define even more targeted policies. The friendship token in the previous example is a self-issued credential that grants vault access to those that are given one. This makes TrustVault well-suited for social applications.

D. Self-Sovereign-Identity

TrustVault is designed for decentralised applications, therefore decentralised identities (DIDs) are needed. We not only wanted to give user fine-grained control over access to their data based on attributes, but we wanted attribute claims to be verifiable. While DIDs like WebID make it easy to make arbitrary claims about yourself, those claims are not trustworthy because they are not verifiable. At most they provide authentication by proving possesion over a private key. SSI with verifiable credentials let users keep control over their identity but also increase trust in their claims, without intermediaries. Authorisation in access management is where SSI truly shines [6]. The ability to selectively disclose only the subset of claims necessary for verification enhances the privacy for users.

E. Tamper-proof access logs

Access control is completely automated without intervention of the TrustVault owner. This makes it impossible to keep track of who has been given access to which files. This is remediated by recording accessible-files-requests on TrustChain for each session. The owner sends a transaction to the requester with a bloom filter containing the accessible files from the request. This forms a timestamped, tamper-proof and irrefutable access log that can be used to verify with high probability if a given file was served to a user.

F. Encryption

Data in the vault is encrypted both at rest and in transfer. A password is required to ”lock” and ”unlock” the TrustVault by using a PBKDF derived symmetric key to encrypt and decrypt every file when starting or closing the TrustVault. IPv8 supports end-to-end encryption which is used when transmitting files between peers.

III. Implementation and Evaluation

This section describes our method for evaluating TrustVault's implementation, performance, privacy and security, and outlines the results.

A. Implementation

TrustVault is made for Android and is thus completely implemented in Kotlin. The codebase includes a fork of walt.id SSI kit. The open source code for SSI kit is also written in Kotlin. However it is not set up to work with Android. Major modifications were made to make it compatible with Android, including changing IO operations and replacing code that uses packages and libraries not supported by Android.

B. EBSI data wallet

EBSI will serve as the main network to issue and receive credentials to and from. This makes TrustVault a secure data wallet for EBSI users. The process of getting TrustVault EBSI conformant was not straight forward. The early prototypes were built using the early versions of the TypeScript cef-ebsi packages for EBSI v1 [7] as part of the EBSI Early Adopters programme. In v1, all operations were API calls to test endpoints. In v2, critical actions including creating, signing, and verifying credentials were moved from the endpoints to libraries running on the user's device. At this point, there were 3 documentation sources for implementing an EBSI that were out of sync in several places meaning that there was a lot of trial-and-error to get the API connection working. In the end, walt.id SSI kit was chosen to handle the issuing and verification of credentials as it is more feature complete in that area.
When initializing TrustVault, the user needs to complete the EBSI on-boarding process which entails scanning a QR-code on the on-boarding page to get an authentication token that is used to get permanent authorisation. In subsequent sessions, the user needs to provide the authorisation token to get short-term session token. The initial authentication step will be dropped once EBSI goes into full production. The ontology used for EBSI credentials can be used to devise appropriate access policies.

C. Performance

Measuring performance of evaluating policies with full credentials vs derived access token, transfer rate of file.

D. Privacy

Privacy in TrustVault can be analysed from the perspective of the TrustVault owner and from the requesting party. One of the main goals of this work is to give users control over their data and thus over their privacy. The first step is to enable users to self-host their data, stopping data-hungry companies from running machine learning algorithms over user data and learning users' behavioral patterns. This has the added benefit of disrupting Big Tech from monetizing user data. Giving the user fine-grained access control allows the user to have specific disclosure policies at the desired granularity level, down to file level. This comes with great responsibility, as there is the opportunity making mistakes when defining access policies and disclose data to unintended parties. The challenge is to make user experience simple and intuitive to minimise the chances for mistakes. The requesting party on the other hand has full control over its identity. Selective disclosurability allows the requesting party to only present claims it is comfortable disclosing.
Identification by static public keys does present the possibility of learning information over time. The host can keep record of every time a certain public key wants to access data, which arguably is a sensible thing for the host. However, the host is able to link different access requests over time while collecting the credentials presented at each request, possibly accumulating a more revealing, or even identifiable set of attributes of the requesting party. TrustChain does not support private/anonymous transactions. By logging access request on-chain, interactions between parties become publicly visible. Anyone can keep track of when and how often one public key requests access from another public key, potentially leaking information.

E. Security

The Android internal file storage protects files from being accessed from outside the Super App [8]. This offers the first layer of protection from unauthorized access. Additionally, the data vault is encrypted using AES when the application is closed. When opened, a password is required to decrypt the data vault. This prevents prevents unwanted access even if someone gets physical access to the phone and launches the application. Data is also encrypted during transfer with IPv8. Packets are split into chunks that are encrypted with the public key of the recipient.

Evaluate security in terms of data protection, potential attacks. Security properties inherited from SSI in comparison to traditional access control in Solid?

Availability of data

IV. Related Work

A. Solid

Solid is a protocol developed at MIT that let's people store their data securely in decentralised data stores called Pods [9]. Pods are personal web servers that can store any kind of data as Linked Data. Linked Data is data with semantic links to

other data recorded in its metadata such that computers can explore these links using semantic queries. Pod owners have granular control over who has access to the data. Solid uses the WebAccessControl system, which is based on access control lists with user identification by WebID, to grant and revoke access to any slice of data contained in a Pod to individuals, organizations, or applications. WebID is a protocol to allows persons, organizations or other types of agents to create their own unique identities and embed links to other people or objects using Resource Description Framework [10]. Access control rules can be made based on the agent's properties found in their profile document.

B. Self-Sovereign Identity

Self-sovereign identity (SSI) is a decentralised model of digital identity developed to address the shortcomings of the previous internet identity models [6]. With centralised identities, centralised institutions such as governments and banks issue credentials that allow citizens to interact with services and each other. On the internet you would establish an account with every website, service or application. In this model, all the data about you belongs to the issuing party, can't be reused, and is out of your control.

The federated identity model introduces identity providers (IDP). IDPs allow you to have one account that can be used to interact with any service that supports that IDP. This is the mechanism behind the social login buttons (Login with Facebook) widely found on the internet today. Federated identity simplified managing accounts for every service to managing a few accounts at a few IDPs. All our identity data, and information about when or how we use our federated identities is now concentrated in these Tech Giants, raising a lot of privacy concerns.

The rise of blockchain technology inspired the decentralised identity model. This model is not based on accounts with centralised institutions or IDPs but on direct relationships between peers. No party controls or owns the relationship. Users are in full control of their identity data, how it shared and with whom. Peers establish private connections by securely exchanging public keys whereby blockchains serve as decentralised public key infrastructures. This model closest resembles how we manage our identities in the real world: with wallets containing credentials obtained from trusted parties which can shown to other parties to initiate an interaction.

Verifiable Credentials are the building blocks of SSI. Much like physical credentials, VCs contain claims about your identity that some authority claims is true about you. You can then use this VC to convince others that trust said authority of these claims. Trust is established by the so called trust triangle formed between issuers, holders/provers, and verifiers. Issuers put a digital signature on issued credentials that are verifiable with their public keys. Verifiers request proofs about identity claims they need to be convinced of. Holders have the choice to respond with a proof that can be verified the verifier. This proof is usually in the form of a Verifiable Presentation: a verifiable credential with a digital signature of the prover. The verification off the issuer's digital signature is what convinces the verifier of the claims made by the prover. Verifiers don't need to have any direct relationship with issuers. They just need to trust an issuer's ability to make correct assertions.

Credentials and cryptographic keys need to be stored securely in a digital wallet. Digital agents wrap users' digital wallets and establish communication with other agents to exchange credentials.

C. European Blockchain Services Infrastructure

The European Blockchain Services Infrastructure (EBSI) is a distributed network that runs a public blockchain to host public and private services that want to leverage the benefits of blockchain technology. The main services that EBSI aims to facilitate are:

1) Notarization: using the blockchain to make digital audit trails and automate compliance checks.
2) Diplomas: giving citizens control over their educational credentials and lowering the cost of verifying documents.
3) European Self-Sovereign-Identity Framework (ESSIF): serve as a verifiable registry and communication channel for an SSI framework across Europe.

Most relevant to this work is ESSIF, as it serves as the network used to issue and receive verifiable credentials. This service encourages European citizens to adopt SSI to improve the identity verification process with public services and private companies across European borders. The EBSI blockchain serves as the verifiable registry where public keys of users and trusted applications can be looked up.

D. Walt.id SSI Kit

The SSI Kit by walt.id is a Self-Sovereign-Identity open source solution, primarily focused on the European EBSI/ESSIF ecosystem [11]. It provides building blocks for key management, issuing, presenting and verifying credentials, and specific EBSI-related functions. Walt.id developed one of the earliest EBSI conformant wallets.

E. Attribute-Based Access Control

Attribute-based access control (ABAC) is an access control model that controls access to objects by evaluating rules against attributes of entities [12]. This allows for more precise access control because of the large set of possible combinations of attributes and consequently large set of possible rules for policies, only limited by the available set of attributes.

F. Tribler IPv8

IPv8 is a peer-to-peer communication protocol developed by Tribler for private and authenticated communication. IPv8 abstracts away physical addresses and allows peers to be identified by their public keys[13]. IPv8 is used for communication in the decentralised applications on the TrustChain Super App. Explain communities/overlays https://github.com/Tribler/kotlin-ipv8/blob/2fb96d22d34411ed0f5375c22ed34218d90b87df/doc/INDEX.md https://py-ipv8.readthedocs.io/en/latest/

V. Conclusion and Future Work

A public blockchain could be used to keep record of every access request made to the TrustVault in a manner that is irrefutable. This further improves the security of TrustVault by making it possible to trace back malicious behavior on an auditable, immutable, publicly verified access log. TrustChain does not support writing arbitrary data to the chain. This functionality would have to be implemented in TrustChain, or a separate ledger could be used solely for this purpose. The storage costs and transaction costs for logging every request on chain may be prohibitive without proper scaling solutions.

Future work Support other SSI networks Audit access logs More privacy preserving solutions (predicate proofs, zkp) Build more dapps on top of TrustVault

References

[1] D. Curran. (2018) Are you ready? Here is all the data Facebook and Google have on you. [Online]. Available: https://www.theguardian.com/commentisfree/2018/mar/28/all-the-data-facebook-google-has-on-you-privacy

[2] K. Birch, D. Cochrane, and C. Ward, "Data as asset? the measurement, governance, and valuation of digital personal data by big tech," *Big Data & Society*, vol. 8, no. 1, p. 20539517211017308, 2021.

[3] European Commission. (2022) European Blockchain Services Infrastructure. [Online]. Available: https://ec.europa.eu/digital-building-blocks/wikis/display/EBSI/Home

[4] ——. (2022) EBSI Grants. [Online]. Available: https://ec.europa.eu/digital-building-blocks/wikis/display/EBSI/EBSI+Grants

[5] ——. (2022) EBSI's Verifiable Credentials Lifecycle. [Online]. Available: https://ec.europa.eu/digital-building-blocks/wikis/display/EBSIDOC/Verifiable+Credentials+Lifecycle

[6] A. Preukschat and D. Reed, *Self-sovereign identity*. Manning Publications, 2021.

[7] European Commission. cef-ebsi packages. [Online]. Available: https://www.npmjs.com/search?q=cef-ebsi

[8] Android. Access app-specific files. [Online]. Available: https://developer.android.com/training/data-storage/app-specific

[9] E. Mansour, A. V. Sambra, S. Hawke, M. Zereba, S. Capadisli, A. Ghanem, A. Aboulnaga, and T. Berners-Lee, "A demonstration of the solid platform for social web applications," in *Proceedings of the 25th International Conference Companion on World Wide Web*, ser. WWW '16 Companion. Republic and Canton of Geneva, CHE: International World Wide Web Conferences Steering Committee, 2016, p. 223–226. [Online]. Available: https://doi.org/10.1145/2872518.2890529

[10] W3C. Webid. [Online]. Available: https://www.w3.org/wiki/WebID

[11] Walt.id. walt.id ssi kit. [Online]. Available: https://github.com/walt-id/waltid-ssikit/blob/master/src/main/kotlin/id/walt/services/jwt/WaltIdJwtService.kt

[12] V. C. Hu, D. R. Kuhn, D. F. Ferraiolo, and J. Voas, "Attribute-based access control," *Computer*, vol. 48, no. 2, pp. 85–88, 2015.

[13] Tribler. IPv8. [Online]. Available: https://github.com/Tribler/kotlin-ipv8