A-SIT

Secure Information Technology Center – Austria

Design Considerations of Ex-post TETs for Credential Transparency

Design Considerations of Ex-post TETs for Credential Transparency

Autor:
Edona Fasllija
Mail:edona.fasllija@iaik.tugraz.at

Ex-post Transparency-Enhancing-Technologies aim to help data subjects exercise their right to transparency by providing them with visibility on the sensitive data that is collected, processed, or disclosed to different agents. This project goes through the main design considerations when developing such a tool for digital credential systems, starting from custom-fit secure logging mechanisms, sanitization, querying, monitoring and auditing mechanisms, and external mechanisms that interact with the data subjects.

1. Introduction 1
2. Goals and Challenges 2
3. Background 3
3.1. Credential Systems 3
3.2. Transparency Systems 4
Ecosystem Actors 4
3.3. Privacy-Preserving Cryptographic Techniques 5
Zero-Knowledge Proofs 5
Blinding Mechanisms 5
4. Design Considerations 6
4.1. Secure Logging Mechanisms 6
Verifiable Data Structures 6
4.2. Sanitization Mechanisms 8
4.3. Querying, Monitoring, and Auditing Mechanisms 8
5. Auditability of Credential Stages 9
5.1. Credential Issuance 10
5.2. Credential Presentation 10
5.3. Credential Revocation 10
5.4. Wallet Setup Phase: Binary Transparency 10
6. Designing a Credential Issuance Transparency Log 11
6.1. CIT Overview 11
6.2. Protocols 12
6.3. Performance Evaluation 13
7. Conclusions: 15

1. Introduction

The adoption of digital identity solutions in Europe has gained momentum with the rollout of the European Digital Identity Wallet (EDIW) [1] and the updated eIDAS 2.0[2] regulation, which expands on the original Electronic Identification, Authentication, and Trust Services (eIDAS) regulation. These initiatives aim to provide European citizens and residents with a standardized, interoperable, and secure means of proving their identity and sharing verifiable credentials across borders. A key challenge in this ecosystem is ensuring that these credentials are trustworthy, tamper-proof, and transparent.

Transparency Enhancing Technologies (TETs) are technical solutions designed to foster transparency

within systems or processes. In particular, "ex-post" TETs [3] provide information only after an event has occurred, enabling retrospective analysis. This project focuses on the design, development, and evaluation of ex-post TETs that can be integrated into various systems to deliver provable transparency. We illustrate how TETs can be applied to digital credential systems, to give users visibility and facilitate trust through cryptographic proof that data remains unaltered.

Specifically, this project examines the design considerations for implementing log-based TETs in user-centric digital identity frameworks, with a focus on Verifiable Credentials [4]. In this context, log-based transparency systems provide a foundation for creating tamper-evident, verifiable records that can be independently audited. Drawing on concepts from established transparency systems like Certificate Transparency and Key Transparency, we explore how transparency logs can support digital credentialing throughout the credential lifecycle, including issuance, verification, usage, and revocation. Each stage benefits from transparency logs, which enable independent auditability and accountability while preserving user privacy.

2. Goals and Challenges

Credential Transparency is a concept we introduce in this report to describe a privacy-preserving, audit-friendly framework for managing and verifying digital credentials. Credential Transparency ensures that each stage in the lifecycle of a credential—such as issuance, verification, or revocation—can be tracked in a publicly auditable manner without compromising sensitive information. Credential Transparency is designed to enhance transparency and accountability within credential systems while preserving user privacy. This approach introduces the need for advanced cryptographic methods that enable the verification of a credential's authenticity and validity without revealing any sensitive information about the user.

One prominent example of transparent audit logging is Certificate Transparency (CT) [4], which has proven the feasibility and utility of public logs for tracking certificates. However, this approach cannot be directly applied to digital credentials because of fundamental differences in privacy requirements. Unlike the fully public certificates in CT, the content of digital credentials is private. Therefore, logging a complete credential, as is done with certificates, would expose sensitive user information, which Credential Transparency seeks to protect.

The solution lies in using a blinding mechanism that addresses two main challenges. First is user discoverability: a user should be able to determine whether their credential is associated with a particular entry in the credential log. Second is verifier verifiability: a verifier, when presented with a credential and a corresponding log entry, should be able to confirm whether the log entry matches that specific credential.

In addition to these functional requirements, Credential Transparency must meet two crucial privacy requirements. The first is credential privacy: it should be practically impossible to derive any information about the credential's claims from the public log entry. The second is credential unlinkability: even if two different credentials and their log entries are compared, it should remain challenging to deduce if they belong to the same individual.

By adhering to these privacy-preserving principles and solving the challenges of discoverability and verifiability, Credential Transparency aims to bring the benefits of public auditability to credential systems without compromising user privacy.

3. Background

3.1. Credential Systems

A credential is a statement about a user (or holder) issued by an entity, which can include attributes like the user's name, physical traits, affiliations, or other verified personal details. The user can present this credential to a verifier, who, if they trust the issuer, can be confident that the user possesses the stated attributes. The combined processes of issuing and presenting a credential form what is known as a credential scheme.

Traditional credential schemes are typically centralized, with the issuer's server holding all user information. In contrast, user-centric digital identity approaches, such as Self-Sovereign Identity (SSI), allow individuals to manage their own digital attestations of identity attributes and cryptographic keys for authentication. This is often done on personal devices, like smartphones, through a "wallet" application. These attestations carry cryptographic proof of integrity, usually in the form of a digital signature from the issuer, making the information verifiable by machines. This concept has given rise to the term "verifiable credentials" (VCs)[5] for these attestations.

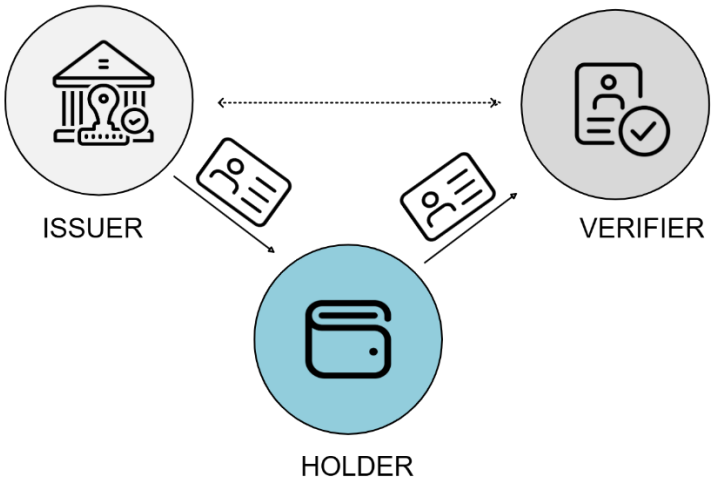

Figure 1 Verifiable Credentials Trust Triangle

The term Verifiable Presentation (VP) describes the process where users disclose specific identity attributes from their credentials to verifiers, also known as relying parties. When the user receives the presentation request, their digital wallet app automatically searches for stored credentials that contain the requested attributes and match the request's criteria. With the user's consent, the wallet app generates a cryptographic proof verifying the correctness of these attributes and sends both the attributes and proof to the verifier.

Privacy-focused VPs allow selective disclosure, so users can share only the necessary attributes with the verifier. VPs also include cryptographic evidence—such as blinded proofs—originating from the credential, which verifies that the displayed attributes genuinely come from a credential issued by a trusted entity. These proofs can be built using various techniques, including Merkle hash trees or zero-knowledge proofs. The verifier can then automatically validate the proof and confirm that the credential was issued by a trusted issuer, ensuring the authenticity of the user's claimed identity attributes, which enables access to services.

3.2. Transparency Systems

The core idea behind transparency systems is the public recording of all attestations issued by an authority, which promotes visibility and accountability. This transparency enables easier detection of unauthorized or fraudulent actions by these authorities. One of the earliest and most widely implemented transparency systems is Certificate Transparency (CT) [4], which was developed to bring accountability to the frequently mismanaged WebPKI ecosystem. Recently, there has been a shift toward adopting CT-like systems for other applications outside WebPKI, such as "key transparency" for secure messaging services [6] .

Transparency systems generally rely on externally auditable data structures that are verifiable by the public and exhibit two primary properties: (i) they are append-only, meaning that data cannot be deleted or modified, and (ii) they are consistent, providing a uniform view of the data for all users. These characteristics ensure that the system delivers the same results for a given query, thereby enhancing trust and integrity by making it difficult to alter or misrepresent the data.

Ecosystem Actors

This section describes the main participants typically involved in transparency systems.

The log provider is an untrusted entity responsible for maintaining a public, append-only audit log that stores information units called log entries. Its primary role is to store and keep this data publicly accessible— a task that traditional actors may not want to take on. Since the log provider is untrusted, it does not have access to any private information and can be cryptographically restricted from altering or removing data.

Auditors oversee the log provider to ensure it appends information properly and maintains a consistent global view. Auditors play a key role in detecting any dishonest behavior by the log provider. Any auditor has the ability to identify and irrefutably demonstrate if a log provider has cheated. Since anyone in the public can act as an auditor without needing special access, log providers must operate under the assumption that their published data is continuously monitored and audited. This setup, coupled with a secure gossip protocol for sharing log fingerprints, enforces honest behavior by the log provider.

Monitors frequently check the log provider's published data to verify the internal consistency within the structure. By validating that the log remains free from inconsistencies, monitors help detect misconduct. They review individual log entries to identify improper issuance and may also serve as log mirrors, offering search and alert services to users.

Finally, sources are the entities that generate the events recorded in the data structure managed by the log provider. End-users, to whom the log entries are relevant, query the log provider to retrieve this information.

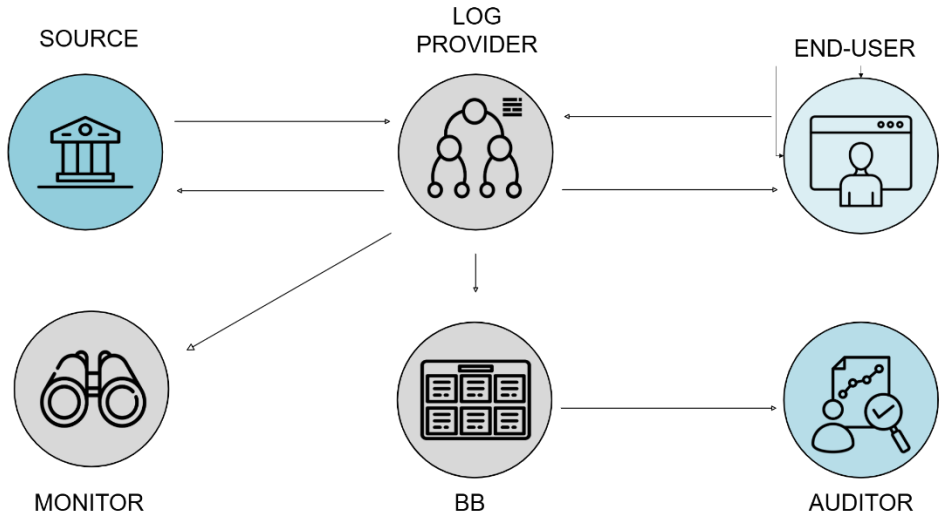

Figure 2 Transparency Systems Overview

3.3. Privacy-Preserving Cryptographic Techniques

Zero-Knowledge Proofs

Zero-Knowledge (zk) proofs are advanced cryptographic techniques that enable a "prover" to demonstrate to a "verifier" that they possess certain information meeting specific properties—without ever revealing the actual information itself. This approach is highly valuable in scenarios where privacy is essential, as it allows the verifier to be assured of the prover's knowledge without requiring them to expose sensitive data.

Among the various types of zk proofs, Zero-Knowledge Succinct Non-Interactive Arguments of Knowledge (zkSNARKs) [7] are particularly notable. zkSNARKs are compact proof constructions that eliminate the need for repeated back-and-forth communication between the prover and the verifier. This non-interactive nature makes zkSNARKs efficient and scalable, allowing for faster verification processes, which is crucial in large-scale or resource-limited environments.

In this work, we employ zkSNARKs to demonstrate that a specific privacy-preserving record, housed within a transparency log, corresponds to and could be traced back to a particular privacy-sensitive piece of information. By doing so, we ensure that while the record's existence and its correlation with certain information are verifiable, the sensitive details themselves remain confidential. This application of zkSNARKs provides a powerful balance between transparency and privacy, enhancing trust in systems that rely on privacy-sensitive data.

Blinding Mechanisms

Blinding mechanisms in cryptography are methods that conceal certain information from specific parties while preserving its underlying authenticity or usability. These techniques are essential in privacy-preserving cryptographic systems, allowing data or identity to be hidden, yet verifiable. Key blinding mechanisms include commitment functions and user-stable pseudonyms.

Commitment Functions are cryptographic primitives that allow a user (the "committer") to commit to a chosen value or statement while keeping it hidden from others (the "receiver"), with the assurance that the value cannot be changed later. This process has two main steps:
(i) *Commit Phase*: The committer selects a value and computes a commitment using a commitment function, producing a commitment output that conceals the original value. This commitment is then

shared with the receiver.
(ii) *Reveal Phase*: When the committer is ready to disclose the hidden value, they reveal both the value and any associated randomness used in the commitment. The receiver can verify that the revealed value matches the commitment by recomputing the commitment and checking it against the original commitment output.

Commitment functions are useful in protocols where trust and integrity are needed, such as digital voting or blockchain systems. They provide properties like *hiding*, where the original value is secret, and *binding*, where the committer cannot alter the value after committing to it.

User-Stable Pseudonyms are identifiers assigned to users that remain consistent across multiple interactions or sessions, but do not directly reveal the user's real identity. This mechanism enables repeatable, authenticated interactions without compromising the user's privacy. In practice, user-stable pseudonyms are created using cryptographic methods like hashing or keyed pseudorandom functions, where only certain trusted parties know the underlying identity or are able to link it to the pseudonym.

User-stable pseudonyms can be used in contexts such as anonymous surveys, decentralized identity systems, and privacy-preserving authentication, where an individual's behavior or reputation needs to be trackable without exposing their actual identity. They strike a balance between accountability and privacy by allowing traceability of interactions under a consistent pseudonym while keeping personal data obscured.

4. Design Considerations

4.1. Secure Logging Mechanisms

Transparency requires information to be recorded in a way that is both traceable and verifiable. This is often achieved through a structured, chronological record of events, actions, or data used by a system. Such a record could even capture the complete history of any data point or byte in its current or past states. To support transparency, logging mechanisms are often built on authenticated data structures and transparency overlays, such as Merkle Trees or blockchains. These structures ensure that logs are not only append-only and verifiable but also queryable and consistent. In this setup, logs meet critical criteria, including:
(i)*Consistency*: Ensuring that a potentially dishonest log server cannot present different, conflicting versions of the log to different parties.
(i)*Non-frameability*: Preventing parties from blaming a log server for misbehavior if it has, in fact, acted honestly.
(i)*Accountability*: Providing evidence to hold log servers accountable if they fail to include events as promised.
By combining these properties, transparency logs enable a trustworthy system for tracking actions and data in a manner that enhances accountability and protects against tampering.

Verifiable Data Structures

Transparency systems are built around *data structures* designed to generate verifiable proofs, guaranteeing the data's immutability and integrity. These structures differ in their construction, which also impacts the types of proof they can generate. Moreover, the data structure affects the efficiency of operations like lookups, inserts, and auditing from different parties, which comprise critical metrics for evaluating the suitability of a data structure in transparency systems.

Overview of Verifiable Data Structures and Cryptographic Proofs

The design of the underlying data structure is crucial for Transparency Systems as it directly influences the system's ability to fulfill its core security, efficiency, and transparency requirements. Below we summarize some of the properties that various underlying verifiable data structures can offer:

1. Proof of Inclusion: Verifies that a particular item is included in the data structure.
2. Proof of Non-Inclusion: Verifies that a particular item is not included in the data structure.
3. Value Retrieval for Key (Key Lookup): Retrieves the (provable and possibly current) value associated with a given key.
4. Proof of Append-Only: Ensures that data can only be appended, not altered or removed.
5. Entries Enumeration: Allows enumeration of all entries in the data structure.
6. Proof of Correct Operation: Verifies that operations (like state transitions) were performed correctly.
7. Split-View Detection: Enables auditors to detect if different parties are shown inconsistent versions of the data structure.

Core Data Structures in Transparency Systems

To illustrate the impact that the design of data structures have on the efficiency of the proofs, we explore some of the most widely deployed verifiable data structures and their key properties:

Binary Merkle Hash Trees are the underlying verifiable data structure for Certificate Transparency. They are binary trees where each leaf node represents a hash of a data item, and each non-leaf node is the hash of the concatenation of its child nodes. Merkle trees provide proofs of inclusion and consistency. These trees are optimized for logarithmic proof generation and verification, but their proof sizes can become cumbersome as the number of entries grows.

Sparse Merkle Trees (SMTs) extend the Merkle tree concept to handle large, potentially sparse datasets where many leaves might be empty or unassigned. SMTs are particularly useful in key transparency systems where the key space is large but sparsely populated. This data structure, also named as a transparency dictionary, serves the purpose of mapping a set of keys (or labels) to the corresponding set of values. This structure enables these trees to be efficiently queried for a particular value. The same is not true for append-only logs, as they require to traverse each leaf until the data is found iteratively. Furthermore, SMTs also allow for efficient proof of the absence of a value (a.k.a. non-membership proofs) by making use of the leaves that are labelled as empty. On the other hand, an append-only log as the one used in CT requires recomputing the whole tree to generate such a non-membership proof, as all the leaves and their corresponding inclusion proofs must be transferred to the auditor(s).

Combination of Trees This hybrid approach combines a prefix tree for efficient key lookups with a history tree that maintains a chronological order of changes. Such structures can support both efficient key-value lookups and historical queries, making them versatile in transparency systems where both current state and historical data need to be verified. This combination is particularly effective in systems like key transparency, where both the current state of keys and their historical validity must be provable.

Merkle Patricia Tries (MPTs) They combine a radix trie (or prefix tree) with a Merkle tree to optimize for key-value store operations. Proof generation is more complex due to the trie structure, but this complexity can be offset by optimizations that exploit common prefixes in the data, reducing the overall size of the proof. The trie structure allows for more efficient proofs in terms of space, especially for operations involving key prefixes. While the proof verification remains efficient, it can be more computationally intensive than standard Merkle trees due to the additional trie overhead. Merkle Patricia

tries are highly suitable to applications where the dataset is hierarchical or where prefix operations are common.

Merkle Forests Merkle forests are collections of Merkle trees used to manage multiple independent logs or datasets. Each tree in the forest operates independently but can be linked to a common root or aggregate proof. Merkle forests allow for parallel proof generation, which can significantly reduce the time required to produce proofs in systems with multiple logs. Proofs from different trees can be aggregated, providing a way to combine multiple independent proofs into a single, verifiable proof. Merkle forests are advantageous in multi-tenant environments where different users or systems maintain separate logs but require a unified verification mechanism.

4.2. Sanitization Mechanisms

The sanitization mechanism defines how information in a log is managed to balance privacy and transparency. This mechanism determines whether data is processed in plaintext (unsanitized) or through various privacy-preserving techniques to restrict access to sensitive details. Privacy-preserving options include:

• Noise Addition or Synthetic Data: Data can be "sanitized" by adding controlled noise or generating synthetic versions, obscuring specific details while maintaining statistical properties. This approach enables safe data release without exposing raw information.

• Encryption: In some cases, logged data is encrypted and made accessible only to specific parties, such as trusted auditors who can decrypt and examine the raw data. Similarly, individuals may be granted selective access to evidence related to them, preserving privacy while allowing for targeted verification.

• Cryptographic Techniques: Advanced methods like zero-knowledge proofs allow verification of certain properties of logged data without disclosing the actual content. For instance, a zero-knowledge proof might confirm that certain events occurred or that data meets a specific requirement, without revealing the data itself.

In cases where no suitable sanitization mechanism exists that can satisfy transparency requirements, limited access to unsanitized data may be essential. For example, if differential privacy cannot be achieved without adding excessive noise or if cryptographic proofs are unable to convey the necessary assurances, access to raw data may be granted to designated auditors. The auditors can then publish verified reports for public scrutiny, while the general public only has access to a sanitized, privacy-preserving version of the data that reflects the audit's findings.

Additionally, identifiers or metadata that enable individuals to verify their personal data in the log may also require careful sanitization. Systems like CONIKS employ verifiable random functions (VRFs) to generate user identifiers that obscure identities while allowing verifiability. Recent approaches [8] have further introduced append-only zero-knowledge sets that limit information exposure from queries, enhancing privacy while enabling transparency.

Through such sanitization mechanisms, logged information can maintain transparency and verifiability while protecting sensitive details, supporting a balance between accountability and individual privacy.

4.3. Querying, Monitoring, and Auditing Mechanisms

Transparency systems employ querying, monitoring, and auditing mechanisms to ensure that stakeholders can trust the system's contents and detect tampering or unauthorized actions. Below we list a breakdown of these mechanisms:

1. Querying Mechanisms

In transparency systems, querying allows users to check the presence and validity of specific records. For

example, Certificate Transparency (CT) allows anyone to query logs to verify the presence of a certificate, ensuring it has been publicly logged and is therefore accountable. Key Transparency (KT) supports secure lookup of cryptographic keys associated with a user, allowing clients to verify they are using authentic keys for secure communication. Binary Transparency enables users to check if a particular software binary has been verified and logged, helping to ensure its integrity and authenticity.

Querying mechanisms need to be designed to be efficient and privacy-preserving. Their efficiency is strongly dependent on the underlying data structure (i.e. Merkle Tree variation), which allow users to verify entries without needing access to the entire log.

2. Monitoring Mechanisms
Monitoring systems are employed to actively watch for and detect inconsistencies, unauthorized changes, or other suspicious activities within logs. Log Monitors are independent services or tools that track updates to logs in real-time, scanning for any discrepancies such as duplicate or missing entries. They can alert users or administrators if potential tampering or irregularities are found. In systems like Certificate Transparency, users or third-party services can continually monitor certificate logs, checking for any rogue certificates issued by a Certificate Authority (CA) without the subject's knowledge.

Monitoring is essential in transparency systems to quickly identify and address threats, providing additional layers of security beyond periodic audits.

3. Auditing Mechanisms
Auditing in transparency systems ensures that logs remain trustworthy and have not been altered over time. Auditing is typically performed by both the system's users and external auditors. Key aspects include:
• Consistency Checks: Auditors verify that logs have maintained their append-only nature and have not been tampered with or rewritten. This is often achieved through proofs that ensure log consistency across different points in time.
• Proofs of Inclusion and Consistency: Transparency systems frequently use cryptographic proofs to demonstrate that an entry has been logged and remains consistent with past entries. These proofs are particularly common in Certificate Transparency and Key Transparency, where they help verify that data (e.g., certificates, keys) have been correctly logged.
• Third-Party Audits: Transparency systems often invite independent auditors to periodically verify the entire log or specific entries, reinforcing the system's trustworthiness. For instance, in Key Transparency, third-party auditors can verify that users' public keys are logged correctly, ensuring that malicious actors cannot impersonate legitimate users.

Through these querying, monitoring, and auditing mechanisms, transparency systems help detect unauthorized activities, ensure data integrity, and provide verifiable accountability. These mechanisms are crucial in building trust by allowing independent verification while preserving privacy and enabling secure, consistent access to critical records.

5. Auditability of Credential Stages

Transparency logs can record the entire credential lifecycle (e.g., issuance, usage, revocation), offering independent proofs of each action taken, and they use privacy-preserving methods to ensure compliance with data protection regulations.

5.1. Credential Issuance

Credential issuance is the process by which trusted entities (issuers) create and distribute verifiable credentials to users. While the credentials themselves are cryptographically signed, adding transparency to this process can further build trust.

Users need to be sure that the credentials they receive are from legitimate and trusted sources. Credential issuance transparency ensures that any credential created is logged in a public or distributed ledger, allowing users to verify the issuer's identity and track the history of credential issuance.

Transparency helps prevent malicious or fake issuers from creating fraudulent credentials. A transparent issuance log would expose such activities, allowing issuers to maintain their reputation and ensuring verifiers trust the credentials presented by users.

Finally, Credential issuance logs can provide an auditable trail of when, where, and by whom credentials were issued, providing accountability for issuers and reinforcing the overall trust in the ecosystem.

5.2. Credential Presentation

When users share their credentials with verifiers (e.g., employers, service providers), transparency in the presentation process is equally important. This ensures that both the user and the verifier have confidence in the data being exchanged.

One of the key advantages of verifiable credentials is the ability to share only the necessary information (e.g., proving you are over 18 without revealing your full birth date). Transparency in credential presentation ensures that only the agreed-upon data was shared, and no more [9].

Furthermore, Presentation transparency verifies that the credential presented has not been altered and that the user consented to share it with the verifier. By providing a transparent trail, the system ensures that users retain control of their data while verifying that the presentation process was valid.

5.3. Credential Revocation

Credential revocation is a mechanism that allows issuers to invalidate a previously issued credential (e.g., if a driver's license is revoked or a professional certification expires). Revocation transparency ensures that both users and verifiers can easily track the status of a credential.

Revocation transparency provides a clear, real-time method for verifying the status of any credential, preventing the misuse of outdated or invalid credentials. If a credential has been revoked (due to an error or a legitimate reason), the user should be immediately informed. Transparency ensures that revocations are publicly logged and that the holder is aware of any changes to their credential status.

Furthermore, Verifiers need to be confident that the credential they are accepting is still valid. A transparent revocation system allows them to check the current status of any credential in real time, ensuring that they only accept valid and trustworthy information.

5.4. Wallet Setup Phase: Binary Transparency

Digital wallets are the cornerstone of user-centric identity ecosystems, as they store and manage verifiable credentials. Ensuring the integrity of the wallet is fundamental, because it is the primary tool users rely on to manage their credentials.

Binary Transparency offers advantages such as:

• Preventing tampering and malware: Binary transparency ensures that the wallet app the user downloads has not been tampered with or compromised by malicious actors. Users can verify that the binary corresponds to the original source code, providing them with confidence that the wallet's functions are trustworthy.

• Open-source validation: In some cases, wallets may be open source, meaning anyone can review the source code. Binary transparency allows users to ensure that the compiled version of the app they download matches the original code reviewed by the community.

• Trust in updates: Regular updates are essential for improving functionality and security, but they can

also introduce risks. Binary transparency logs verify that new releases come from trusted developers and have not been altered, ensuring that security remains intact over time.

6. Designing a Credential Issuance Transparency Log

6.1. CIT Overview

This section introduces Credential Issuance Transparency (CIT) [10], a privacy-preserving audit logging mechanism specifically designed for the issuance of identifying credentials. CIT builds upon the concept of append-only logs, as seen in Certificate Transparency (CT), but adapts it to protect sensitive information. By fully blinding log entries, CIT removes all identifying details, thereby maintaining user privacy while preserving public auditability of the log's append-only structure and enabling individuals to verify credentials related to them.

Instead of displaying a publicly visible domain name or other identifying information, CIT uses a randomly blinded commitment to conceal the credential subject's identity. This ensures that only authorized parties can link a credential to its subject while the public can still validate that the credential issuance was properly logged.

CIT also inherits CT's trust model, relying on trusted verifiers to enforce logging mandates. To be accepted as valid, each credential must have a corresponding entry in the CIT log, ensuring that all credential issuances are properly recorded. However, the blinded nature of these log entries introduces an additional validation challenge: verifying the authenticity of a credential without revealing sensitive information. To address this, CIT employs non-interactive zero-knowledge proofs (NIZKPs). These proofs allow the credential issuer to demonstrate that a log entry is consistent with the associated credential, without exposing any private data.

As illustrated in Figure 3, CIT enables users to monitor the issuance of any identifying credentials that could potentially be used to impersonate them. By providing a secure, privacy-preserving framework for credential issuance, CIT supports the principles of self-sovereign identity and moves closer to a model acceptable to legislative and regulatory bodies.

This audit and logging approach promotes accountability while safeguarding privacy, bridging the gap between transparency and individual security in digital identity frameworks. In doing so, CIT addresses a critical need for verifiable, privacy-centric auditability, essential to building trust in self-sovereign identity systems and supporting broader adoption across both public and private sectors.

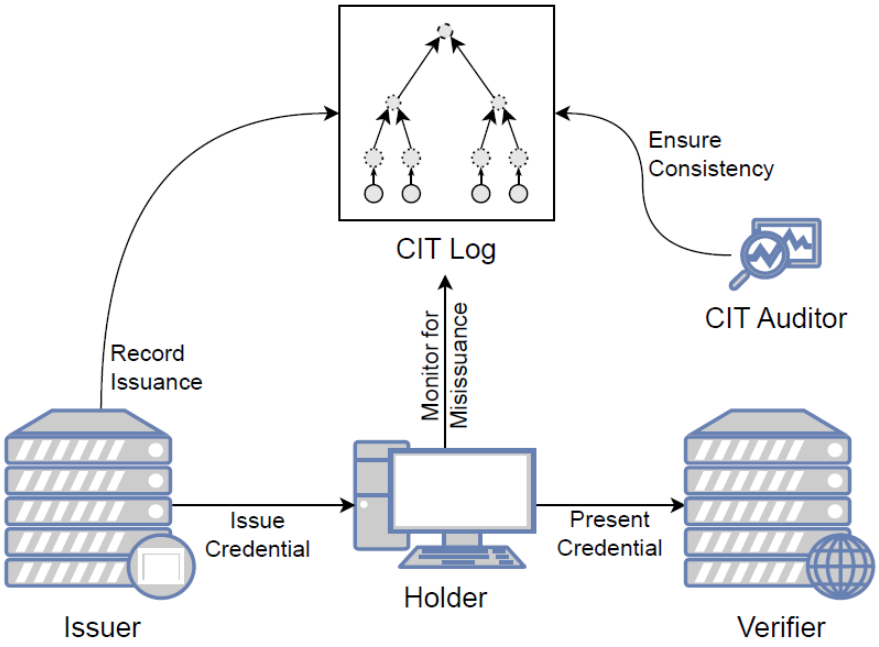

Figure 3 CIT Overview

6.2. Protocols

To address these challenges, we propose the Credential Issuance Transparency (CIT) protocol, a privacy-preserving framework for auditing credential issuance. This protocol is designed to ensure that identifying credentials are logged in a secure and verifiable manner without revealing sensitive user information, enabling users and verifiers alike to confirm the issuance of credentials in a privacy-conscious way.

In CIT, each log entry consists of two main elements: the hash digest of the credential itself and a blinded commitment tied to the user's unique identifier (UserID). This approach ensures that no sensitive details are exposed, while the commitment enables users to identify and verify log entries relevant to them. By knowing their UserID, users can simply perform a trial calculation to determine if a particular log entry corresponds to their credential, enhancing transparency without compromising privacy.

To ensure that verifiers can authenticate the log entries without accessing sensitive user data, CIT employs a non-interactive zero-knowledge pre-image proof. This cryptographic proof allows verifiers to confirm that the blinded commitment in a given log entry corresponds to the UserID underlying a user's pseudonym—without revealing the actual UserID itself. As a result, verifiers can ascertain the legitimacy of the credential issuance without gaining access to personal information. Table 1 summarizes the protocols introduced by CIT in more detail.

The CIT protocol thus strikes a balance between auditability and privacy, enabling users to monitor their credentials for unauthorized issuance while allowing verifiers to confirm credential validity. By providing a secure, privacy-preserving mechanism for credential audit logging, CIT supports a transparent, verifiable system that aligns with privacy-first principles, making it a valuable component in secure digital identity ecosystems. This innovation paves the way for more robust self-sovereign identity

frameworks, where privacy and transparency are harmoniously integrated to enhance trust and compliance in digital identity management.

Credential Issuance Protocol: 1. Issuer issues Credential: x based on $UserId$ 2. CIT appends log entry : $l := (n, C(n, UserId), H(x))$ 3. CIT returns signed inclusion proof 4. Issuer appends l and inclusion proof to x and returns it to the User
Credential Presentation Protocol: 1. User shows x, l, inclusion proof to Verifier 2. Verifier learns $p := P(s, userId)$ 3. User provides NIZKP of $C(n, UserId) \wedge p = P(s, UserId)$ 4. Verifier checks $H(x)$ and NIZKP to ensure correct logging
Discovery Protocol: 1. User requests all new audit log entries 2. For each entry $(n, C(n, UserID'), H(x))$, the user trial calculates $C(n, UserID) == C(n, UserID')$ a. User compares H(x) against its credentials b. If no-match is found, the user has identified a misissued credential

Table 1 CIT Protocols for Issuance, Presentation, and Discovery

6.3. Performance Evaluation

We analyze the practicality of using Credential Issuance Transparency (CIT) in real-world settings and conclude that it is feasible. Below we provide a summary of the key performance considerations:

Log Size and Downloading Requirements: In CIT, users (acting as monitors) need to download new log entries appended since their last check. For comparison, in Certificate Transparency (CT) within the Web PKI system, an average log entry size of 5.93 KB and daily growth of about 7.8 million records result in 43.99 GB of new log data per day—a scale manageable only by large organizations. By contrast, CIT uses fully blinded data, resulting in a constant log entry size of just 96 bytes. Even at CT's high growth rate, this would generate only 746 MB of new data daily, which is manageable for individual users on a typical home internet connection.

Verification Computation: After downloading entries, users must compute trial hashes to compare their UserID commitments. Using SHA-256, modern consumer hardware with hash rate acceleration can achieve over 3 million hashes per second, meaning users could feasibly handle the 8 million daily records reported by CT studies with ease, even after accounting for system overhead.

Log Entry Submission and Append Proofs: Each entry submission requires obtaining an append proof, which involves minimal data transfer. The Merkle Tree append proof size grows logarithmically with the number of entries. Assuming SHA-256 hashes and daily entry rates similar to CT, the proof size remains under 1 KB per entry, which is insignificant in terms of bandwidth.

Overall, even at a global adoption scale comparable to CT, CIT's monitoring requirements remain

feasible for individual users on standard home PCs. Additionally, CIT's logging operations introduce no noticeable overhead during the credential issuance process, making it practical for widespread use.

Proofs of Logging Correctness: We benchmarked our zero-knowledge proof (ZKP) implementation on an Intel i7-8550U laptop. Table 1 presents the results, and Figure 3 provides a visualization. The process is divided into three phases:
1. Initiation Phase: Performed once for the entire system, including the trusted setup and circuit compilation.
2. Preparation Phase: Executed once per nonce-userID-spID tuple, which users can pre-compute.
3. Showing Phase: Completed by the verifier each time a credential is shown. This phase is fast enough that caching at the verifier is generally unnecessary, though feasible.

Figure 4 compares the performance of different hash functions for each phase of the NIZKP proof. Table 2 provides the evaluation results obtained for each of the phases.

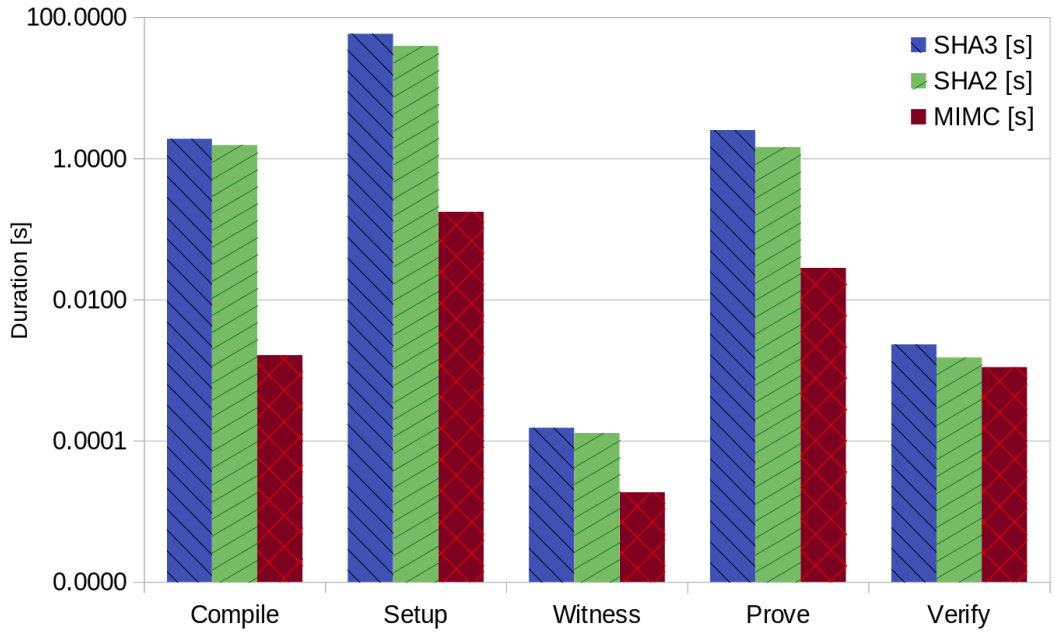

Figure 4 NIZKP Evaluation

		BN254		
		MiMC	SHA256	SHA3-256
	security		102 bit	
	#constraints	1322	170157	235032
initiation	compile policy [s]	0.002	1.541	1.891
	setup (gen. keys) [s]	0.176	39.134	58.189
prepare	gen. witness [s]	0.00002	0.0001	0.0002
	prove statement [s]	0.028	1.444	2.515
show	verify proof [s]	0.001	0.002	0.002

Table 1 Evaluation Results

These results demonstrate that the ZKP implementation is efficient, particularly in the showing phase, making it suitable for real-time verification.

Overall, our evaluations demonstrate that CIT is highly practical, introducing minimal impact on system performance. During the credential presentation process, CIT adds an overhead of less than 2 milliseconds, making it virtually unnoticeable in real-time applications. Additionally, users who wish to monitor for unauthorized credential issuance need only download about 1 GB of data per day, a manageable amount for most internet connections. The verification process is equally efficient, requiring less than 10 seconds of computational effort to complete. These low resource requirements make CIT a feasible solution for widespread adoption, even for individual users on standard devices.

7. Conclusions:

This project strived to show that Transparency Enhancing Technologies (TETs), especially log-based transparency logs, play a critical role in ensuring that digital identities and verifiable credentials can be audited and verified independently.
It creates an open and verifiable framework where users, issuers, and verifiers can interact securely and confidently. By integrating transparency into every phase—wallet binaries, credential issuance, presentation, and revocation—digital identity ecosystems can achieve several key goals:
• Increased Accountability: All actors, including issuers and wallet developers, are accountable for their actions through auditable and verifiable logs.
• Enhanced Security: Transparency adds layers of security, preventing tampering, fraud, and misuse of credentials, ensuring that data remains secure and protected.
• User Empowerment: By providing insight into the systems and processes, users are empowered to make informed decisions about how they manage and share their identities.

Furthermore, we presented Credential Issuance Transparency (CIT), a framework designed to enhance transparency and trust in the issuance of identifying credentials within digital identity systems. CIT enables users to monitor the issuance of their credentials, ensuring that any mis-issuance or unauthorized issuance can be quickly detected and audited. Importantly, CIT achieves this while maintaining efficiency and usability: it adds an imperceptible overhead of less than 2 milliseconds during credential presentation, requires only minimal daily data downloads of under 1 GB for monitoring, and completes computational verification in under 10 seconds.

These features make CIT a practical solution for real-world adoption, supporting secure and privacy-preserving auditability without burdening users or systems. As digital identity becomes increasingly central to secure online interactions, CIT provides a foundational step toward accountable, verifiable credential management that aligns with privacy principles. This framework thus has the potential to foster greater trust in digital credentialing ecosystems and advance the adoption of self-sovereign and user-centric identity models.

References

[1] "EU Digital Identity Wallet Home - EU Digital Identity Wallet -." Accessed: Nov. 11, 2024. [Online]. Available: https://ec.europa.eu/digital-building-blocks/sites/display/EUDIGITALIDENTITYWALLET/EU+Digital+Identity+Wallet+Home

[2] “The European Digital Identity Regulation (eIDAS 2): Framework and Insights.” Accessed: Nov. 11, 2024. [Online]. Available: https://www.european-digital-identity-regulation.com/

[3] A. Hicks, “SoK: Log Based Transparency Enhancing Technologies,” May 2023, Accessed: Nov. 11, 2024. [Online]. Available: http://arxiv.org/abs/2305.01378

[4] “Certificate Transparency : Certificate Transparency.” Accessed: Nov. 11, 2024. [Online]. Available: https://certificate.transparency.dev/

[5] “Verifiable Credentials Data Model v2.0.” Accessed: Apr. 02, 2024. [Online]. Available: https://www.w3.org/TR/vc-data-model-2.0/

[6] “Deploying key transparency at WhatsApp - Engineering at Meta.” Accessed: Nov. 11, 2024. [Online]. Available: https://engineering.fb.com/2023/04/13/security/whatsapp-key-transparency/

[7] “What are zk-SNARKs? - Z.Cash.” Accessed: Nov. 11, 2024. [Online]. Available: https://z.cash/learn/what-are-zk-snarks/

[8] M. Chase, E. Ghosh, A. Deshpande, and H. Malvai, “Seemless: Secure end-to-end encrypted messaging with less trust,” *Proceedings of the ACM Conference on Computer and Communications Security*, pp. 1639–1656, Nov. 2019, doi: 10.1145/3319535.3363202/SUPPL_FILE/P1639-MALVAI.WEBM.

[9] S. More, J. Heher, E. Fasllija, and M. Mathie, “Service Provider Accreditation: Enabling and Enforcing Privacy-by-Design in Credential-based Authentication Systems,” *ACM International Conference Proceeding Series*, p. 86, Jul. 2024, doi: 10.1145/3664476.3669934.

[10] E. Fasllija, J. Heher, and S. More, “Credential Issuance Transparency: A Privacy-preserving Audit Log of Credential Issuance,” Sep. 04, 2024, *Springer*. Accessed: Nov. 11, 2024. [Online]. Available: https://graz.elsevierpure.com/en/publications/credential-issuance-transparency-a-privacy-preserving-audit-log-o