HALP: A Hybrid Anonymous Login Protocol

by

Zunaed Sazzad Tonay
22101416
Elham M. Ajmotgir
22301220

A thesis submitted to the Department of Computer Science and Engineering
in partial fulfillment of the requirements for the degree of
B.Sc. in Computer Science and Engineering

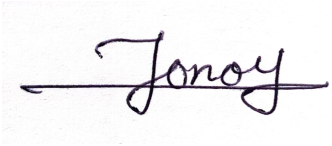
Department of Computer Science and Engineering
BRAC University
January 2026.

Declaration

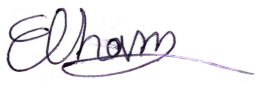
It is hereby declared that

1. The thesis submitted is my/our own original work while completing degree at BRAC University.
2. The thesis does not contain material previously published or written by a third party, except where this is appropriately cited through full and accurate referencing.
3. The thesis does not contain material which has been accepted, or submitted, for any other degree or diploma at a university or other institution.
4. We have acknowledged all main sources of help.

Student's Full Name & Signature:



Zunaed Sazzad Tonay
22101416



Elham M. Ajmotgir
22301220

Approval

The thesis/project titled "HALP: A Hybrid Anonymous Login Protocol" submitted by

1. Zunaed Sazzad Tonay (22101416)
2. Elham M. Ajmotgir (22301220)

of Fall 2025 has been accepted as satisfactory in partial fulfillment of the requirement for the degree of B.Sc. in Computer Science on 31st January, 2026.

Examining Committee:

Supervisor:
(Member)

Dr. Md Sadek Ferdous

Professor
Department of Computer Science and Engineering
BRAC University

Thesis Coordinator:
(Member)

Dr. Md. Golam Rabiul Alam

Professor
Department of Computer Science and Engineeringt
BRAC University

Head of Department:
(Chair)

Dr. Sadia Hamid Kazi

Chairperson
Department of Computer Science and Engineering
BRAC University

Abstract

The current research demonstrates the design and implemention of the Hybrid Anonymous Login Protocol (HALP), which attempts to overcome the main shortcomings of existing privacy-preserving authentication systems. Reliance on fixed identifiers in conventional authentication systems implies that the attributes associated with a user's privacy are exposed to tracking and profiling threats. HALP combines Zero-Knowledge Proofs (ZKPs), pseudonymous identifiers and selective revocation protocols to accomplish unlinkability, scalability, and privacy protection. The ability to log in to the system without revealing identifiable information allows the user to maintain their privacy, while selective revocation provides the ability to prevent compromised credentials from gaining access to services. HALP is designed modularly and conforms to decentralized identity specifications (most notably W3C Verifiable Credentials), enabling its use in privacy-oriented applications such as healthcare, e-voting and decentralized applications (DApps). The framework's performance and scalability are thoroughly evaluated through benchmarking tests, and security is analyzed in terms of cryptographic properties and security games, through which HALP was found to be effective for deployment in real-life scenarios.

Keywords: Zero-Knowledge Proofs (ZKPs), Hybrid Anonymous Login Protocol (HALP), Pseudonymous Identifiers, Selective Revocation, Unlinkability, zk-SNARKs, Anonymous-Credentials.

Acknowledgement

We are grateful for the assistance and advice of our supervisor Dr. Md Sadek Ferdous Sir, who played a critical role in guiding the direction towards which this research had to be taken through his expert knowledge and encouragement. We also want to thank Yeasin Ali sir and Istiaque Ahmed sir, whose help was very important in the development of our work.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

In today's digital age, user privacy is under constant threat. As surveillance has become a normal part of the human condition, centralized identity systems enable the constant tracking of users. Traditional authentication systems make it easy for both companies and hackers to track and link what a person does across different websites or apps over time. User profiles are now created faster and with more granular details than ever before. So, undoubtedly, the most obvious target of all this is our right to privacy. Because this loss of privacy threatens civil liberties. It also discourages people from engaging in sensitive online activities. These activities include whistleblowing, confidential communications, and secure voting.

Anonymous authentication allows individuals to access online services and then pass some kind of test to prove that they have access to the correct credentials, but not who they might be [5]. Unlike conventional systems, where authentication and identification are tightly coupled, anonymous login mechanisms decouple these layers, enhancing user privacy while still enforcing access control. That is why this is ideal to use in anything such as the anonymous chat forums, secure online voting, and decentralized finance (DeFi), where privacy is essential.

With the rapid advancement of privacy-enhancing technologies, we now think of login systems that allow an individual to remain untraceable, unlinked across their multiple usages, and only disclosure their identity when there is truly no choice. Some tools are assisting in bringing this type of secure, privacy-protecting access to reality, such as Zero-Knowledge Proofs (ZKPs), Anonymous Credentials (e.g., Idemix or U-Prove), or Blind Signatures[4], [5].

1.1 Background

Overall, the proposed system is concentrated on the privacy-preserving authentication in which user security will be advanced without loss of anonymity. The core behind it is the use of such technologies as Zero-Knowledge Proofs (ZKPs), Anonymous Credentials, and Pseudonymous Identifiers. Zero-Knowledge Proofs, and zk-SNARKs in particular, are a cryptographic solution that allows one to prove possession of some information in a way that does not compromise any other secrets or sensitive data. They are of great importance to systems that need fast and small proofs of the high-throughput and small-latency authentication protocols, e.g., de-

centralized finance (DeFi) and e-voting systems. They help to provide unlinkability, thus the actions of users cannot be linked between sessions or services. Selective disclosure of credentials will also be achievable through the zk-SNARK protocol to be incorporated into the system to reduce the amount of information revealed, depending on the nature of the required service.

The protocol proposed in this paper is a development of previous work on privacy-preserving identity management because it proposes a hybrid model that has the advantages of centralized-model systems as well as decentralized technologies. In essence, the framework is based on Anonymous Credentials so that users can establish a particular identifying characteristic, like age or citizenship, and, at the same time, hide their other personal details. An extra protection is provided by blind signatures, which have the property (e.g., in the Idemix system) that the issuer cannot re-link a credential with its holder. As such, the design is useful in addressing the tracking and profiling hazards of the conventional centralized infrastructures. However, scalability issues on current systems, such as those of Idemix and zkLogin, have been experimentally revealed: they both present significant computation overhead and do not provide effective revocation with separate and distinct mechanisms. As an alternative, HALP (Hybrid Anonymous Login Protocol) comes into play by overcoming these limitations with the help of a combination of centralised and decentralised techniques.

The suggested system utilizes Pseudonymous Identifiers and Nullifiers, which can be acquired to strengthen anonymity and assist in the effective denial of credentials. Pseudonyms are used dynamically for each session to prevent cross-session monitoring, and nullifiers enable the revocation of compromised credentials without disclosing the identity of the user. These methods fulfill the following desired principles of unlinkability, minimum disclosure, and the main requirements of modern privacy-preserving systems. The framework also allows on-chain as well as off-chain revocation mechanisms, which facilitate freedom based on platform-specific applications.

The key tools and technologies to be used in its implementation are Zero-Knowledge protocols Circom as a ZKP circuit generator, SnarkJS as an engine to generate proofs, and decentralized identifiers (DIDs) to associate credentials with self-sovereign identities. The system, combined with a modular architecture, will be able to interface with existing specifications of decentralized identity like W3C Verifiable Credentials and remain scalable to real-life applications. Through the use of robust cryptographic protocols and associated infrastructure, the suggested architecture seeks to balance robust privacy with efficient and safe authentication.

1.2 Motivation

Although systems such as Idemix and zkLogin provide high privacy guarantees, they are either not very scalable or do not support efficient means to remove malicious users without breaking anonymity[5], [14]. Furthermore, a large number of existing solutions are based on centralized trust models or require the use of additional computational load in real-world environments, such as federated login or

distributed identity networks[14]. The recent cases (abuses of anonymous credentials in anonymous QA services or decentralized markets) support the necessity of a balance between privacy and responsibility. It is becoming urgent to have selective revocation through which a certain pseudonymous identity can be banned without compromising their identity or putting other people at risk [16].
Anonymous authentication also separates the identity of a user and their authentication credentials so that people can demonstrate they have the right to use a service without having to identify themselves. This is an important part of privacy-preserving systems in the fields where confidentiality of information is required. The past studies identify the need for anonymous authentication on the following sectors: **Secure E-Voting:** It will provide anonymous authentication, which means that voters can assert their qualification to vote without any reference to their identity, stopping coercion and vote-purchasing [14]. **Healthcare:** Privacy-friendly access to health records must be possible in medical systems because a person should have a way to verify his or her eligibility to view records without revealing any sensitive health information. **Decentralized Finance (DeFi):** The DeFi sector stands out as privacy- preserving transactions plays a critical role because the Know Your Customer (KYC) policies need to ensure that transactions stay anonymous without re- vealing delicate financial information [18].

1.3 Problem Statement

Research challenges to privacy-preserving authentication exist that are a hindrance to its effectiveness and popularity. Systems like Tor and Zcash are effective in the case of anonymity through unlinkability, but do not have sufficient forms of revocation of credentials that have been compromised without being in violation of anonymity. On the other hand, more frameworks (such as OAuth) offer powerful revocation features, but cannot avoid using persistent identifiers. Scalability also makes it even more complex: idemix, for example, has high computation overhead and thus is inappropriate to run large-scale deployments. In addition, several current systems rely on centralized trust, thus bringing about more risk and compromising on user sovereignty. Moreover, inadequate modularity of the existing frameworks hinders their support of decentralized frameworks like blockchain, which limits their use in decentralized applications (DApps) and such new industries as decentralized finance (DeFi). All these difficulties demonstrate the need for a more flexible and efficient approach. In 2.1, we can see the limitations of the existing research.

Unlinkability vs. Revocation Trade-off The unlinkability vs. revocation trade-off is one of the most acute problems in the modern systems of preserving privacy during authentication. A significant number of systems providing unlinkable authentication (e.g., Tor, Zcash) guarantees that the user's privacy is achieved by separating actions of a user in different sessions. But they do not have an effective means through which they can avert the compromised credentials without violating the anonymity of the users. In contrast, systems such as OAuth support strong revocation but do this at the annus horribilis of privacy, by associating the sessions using identifiers that are persistent, like emails or IP addresses. An example of such a system is Zcash, which also provides unlinkability through the use of zk-SNARKs, but has no means of efficiently revoking compromised credentials, a major weakness

against malicious actors [25].

Scalability Issues in Existing Systems: Scalability is another major issue of current systems. As an example, one of the most applicable privacy- preserving authentication frameworks, Idemix, implements Camenisch-Lysyanskaya (CL) signatures, which have the advantage of supporting selective disclosure but are very computationally expensive and require complicated setups. That renders Idemix impractical in large systems that might have hundreds of thousands and millions of users. For instance, Idemix's high computational overhead due to CL-signatures makes it unsuitable for applications where fast verification is critical, such as large-scale web platforms or real-time services [12], [25].

Trust Assumptions in Current Systems: The other significant con- straint is the centralized assumptions of trust of various authentication systems. As an example, U-Prove (Microsoft) uses a central issuer, which generates a single point of failure in authentication. By analogy, zkLogin, which can be connected to commercial identity providers such as Google and Facebook, is susceptible to span-across tracking and may result in the centralization of control over user authentication data [20]. As an illustration, U-Prove is based on a centralized issuer, which compromises user sovereignty because it introduces the real possibility of compromising or manipulating credentials at the central hub point of control.

Lack of Modularity in Existing Frameworks: Current privacy-preserving authentication systems are usually monolithic and d:o not provide flexibility to work with recently developed decentralised platforms. As an example, the Idemix can not be compatible with Ethereum and other blockchain-type systems, which are not suitable for decentralized applications (DApps) [14]. The challenge imposed by the lack of a straightforward way to integrate with the new smart contracts and blockchains cripples the applicability of such systems in contemporary decentralized settings. Example: The Idemix protocol is too complex and incompatible with blockchain protocols such as Ethereum, so it cannot be used to implement decentralized finance (DeFi), as well as other promising applications that enhance the use of Idemix functionality in practice [12].

1.4 Objective

In this research, we aim to satisfy the following research objectives (ROs):

RO1. Develop Design of a Hybrid Anonymous Login Protocol (HALP) Architecture: Design a scalable, modular system of anonymous logins based on zero-Knowledge Proofs (ZKPs), single-use pseudonyms, and cryptographic nullifiers that are both on-chain and off-chain revocation registries.
RO2. Obtain Scalable and Secure Unlinkability: Ensure each login is unlinkable by creating new pseudonyms in each session, and ensure scalability of the system and high throughput in real-time high-density applications.
RO3. Facilitate Selective Revocation Mechanisms: Establish a scalable system of selective revocation, so that compromised or mischievous pseudonyms may be

blacklisted without sacrificing the privacy of the other users, with cryptographic nullifiers to secure and scale revocation.
RO4. Optimize the system to be used in the real world: Compare the performance of the benchmarked system, HALP, with other privacy-preserving systems, under the metrics of latency, throughput, and computational performance, so that the proposed system does well when used in real-time authentication systems.
RO5. Conduct Formal Privacy and Security Analysis: Demonstrate that HALP meets privacy properties such as unlinkability and revocation soundness using formal cryptographic arguments that resist various typical attacks, including Sybil, replay, and forgery attacks.
RO6. Develop and Evaluate an Open-Source Prototype: Develop an open- source reference implementation in Rust/TypeScript that will implement credential issuance, ZKP-based login, and revocation management, and real- world performance parameters, such as the gas cost, the size of proofs, and their scalability.

1.5 Methodology in Brief

The current research project will take on the scalability, the unlinkability, and the revocation constraints of current privacy-preserving authentication schemes the most notable of which is Idemix and zkLogin. A critical literature review is done to analyze the advantages and deficiencies of approaches like Zero-Knowledge Proofs (ZKPs) and anonymous credentials in order to make a policy decision of structuring the Hybrid Anonymous Login Protocol (HALP). The HALP aims at achieving unlinkability and selective revocation, as well as performance optimisation.

The methodology (Figure 4.1) involves a scientific evaluation of functional and non-functional requirements, i.e., secure issuance of credentials, scalability, etc., that need to be performed in conjunction with a threat model based on the STRIDE structure to check the possible security pitfalls. The outcome of system architecture is the incorporation of ZKPs, pseudonyms, and nullifiers through which the system achieves scalable and modular authentication properties of on-chain and off-chain revocation paths. The property of the system to preserve data privacy will be checked in a comprehensive security analysis, and an open-source realisation of a prototype in Rust/TypeScript will be accomplished. Gas expenditure and overall performance are some of the measures in which this prototype will be benchmarked.

1.6 Scopes and Challenges

This study examines design, implementation, and effectiveness of the Hybrid Anonymous Login Protocol (HALP). This protocol combines Zero-Knowledge Proofs (ZKPs), one-time pseudonyms, and cryptographic nullifiers around the task of developing an authentication protocol that would be scalable, unlinkable and revocable at the same time. HALP aims to alleviate some of the privacy and scalability limitations imposed on existing systems and, by protecting unlinkability, to allow revocations on demand, and to avoid the need to assume centralised trust. More so, it will be in real-time optimisation, thus it minimizes 12 delay and maximising big-scale throughput. To this effect, an open-source prototype will be created and benchmark tested

against the current solutions to gauge performance, scalability, and compatibility with standardised decentralised identity protocols like W3C Verifiable Credentials and DID-Core.

Numerous challenges have to be overcome to allow HALP, a scalable and high-throughput credential-based authentication protocol to achieve practical adoption. On the one hand, zero-knowledge proofs (ZKP) are very efficient to verify, yet introduce computational complexity to generate such a proof. To find the right balance between powerful performance and privacy is thus a vital design challenge. Second, due to large user traffic, HALP needs to have low latency, and this increases scalability issues. Third, safe selective revocation schemes are a requirement, and they should not affect the privacy of the users. Fourth, using blind signatures, nullifiers, and ZK- SNARKs would require quite a bit of cryptographic complexity, which in turn would require extensive security testing. Fifth, achieving interoperability with the current systems and decentralized identity standards are challenging since most current systems are centralized. Lastly, decentralized trust that can be made without compromising on security should be developed, and the difference will be whether it is used or not, which will depend on how usability issues are addressed, especially for end users in privacy-sensitive areas. Despite these difficulties, HALP can be used to significantly improve privacy and scalability when it comes to current authentication systems.

1.7 Report Structure

- In Chapter 2, we present the background terminologies and literature review of the relevant research works.
- Chapter 3 establishes HALP's foundation through research methodology, system architecture, functional/security requirements (anonymous authentication, unlinkability, revocation), and examines societal impacts, ethical considerations, and project management.
- In Chapter 4, we describe our protocol flow and system architecture.
- In Chapter 5, we do the performance and system evaluation of our system.
- In Chapter 6, we conclude our thesis report by mentioning the future works.

Chapter 2

Literature Review

The chapter reviews the basics, covering the principles and available frameworks, which are relevant to privacy-preserving authentication. The key concepts, i.e., unlinkability, minimal disclosure, and selective revocation, are discussed and emphasized as being important to anonymous login systems. Existing methods, such as Zero-Knowledge Proofs (ZKPs), blind signatures, and anonymous credentials, are compared in each of the following points with regard to the advantages and limitations of each method with regard to the preservation of privacy and scalability. The discussion is then focused on the major systems like Idemix, zkLogin, and U-Prove, with their deficiencies towards scalability and revocation capability being outlined. The discourse climaxes in the comparative analysis where gaps have been found in the existing methods, which the presented Hybrid Anonymous Login Protocol (HALP) is keen to address. This discussion plays a key role in the run-up towards presenting HALP in detail in the upcoming chapters.

2.1 Preliminaries

Authentication refers to a procedure by which a system confirms the identity of a user. It is usually done with techniques such as usernames and passwords, biometrics (fingerprints, face recognition), and multi-factor authentication (MFA). Nevertheless, the existing authentication mechanisms are also coupled with serious issues of privacy. During login, the system usually captures identifiable data (e.g., email, username), and this becomes a means of tracking the user or profiling the user amongst a multiplicity of services. This is a privacy compromise as it doles out more information than what is necessary concerning the user To address this, the concept of anonymous authentication came up. In anonymous authentication, the user can demonstrate his or her credibility to view a service without submitting personal data. This is vital in the security of user privacy, especially in the sensitive areas such as healthcare, e-voting, and whistleblowing sites [42].

2.1.1 Privacy Requirements in Anonymous Authentication

Unlinkability guarantees that the same user's identity cannot be tracked across different login sessions. For example, when Alice logs into the system in two different situations, for a consultation and for a prescription refill, the system will not be able to match these two sessions with Alice, even if they are the same person.

This means that even if the same user logs into a system repeatedly, she will not have any linkability. This is accomplished through means such as ephemeral key rotation (keys rotating between sessions) and zero-knowledge proof (ZKP) authentication, which allows authentication without leaking identifying characteristics[47]. Mathmatically unlinkability is defined as the inability to link items or actions based on available evidence. This concept is formalized through equivalence relations, probability distributions, and entropy measures [7].

Unlinkability within One Set: Let $A = \{a_1, \ldots, a_n\}$ be a set of items in a given system. We model the unlinkability of items within the set using an equivalence relation $\sim_{r(A)}$, which partitions A into equivalence classes $A_1, \ldots, A_l$, where:

$$A = A_1 \cup A_2 \cup \cdots \cup A_l \quad \text{and} \quad A_i \cap A_j = \emptyset \quad \forall i \neq j.$$

Two items $a_i, a_j \in A$ are *related* if $a_i \sim_{r(A)} a_j$, and they are *unrelated* if $a_i \not\sim_{r(A)} a_j$.

Degree of Unlinkability (Two Items): The *degree of unlinkability* between two items a_i and a_j in A is measured by the attacker's *a posteriori* probability of their relationship, given some evidence:

$$d(i,j) = -P(a_i \sim_{r(A)} a_j) \log_2 P(a_i \sim_{r(A)} a_j) - P(a_i \not\sim_{r(A)} a_j) \log_2 P(a_i \not\sim_{r(A)} a_j).$$

where:

- $P(a_i \sim_{r(A)} a_j)$ is the probability that a_i and a_j are related.
- $P(a_i \not\sim_{r(A)} a_j) = 1 - P(a_i \sim_{r(A)} a_j)$ is the probability that a_i and a_j are not related.

The degree of unlinkability ranges from:

$$d(i,j) = 1 \quad \text{(maximal unlinkability, i.e., } P(a_i \sim_{r(A)} a_j) = P(a_i \not\sim_{r(A)} a_j) = \frac{1}{2}),$$

to:

$$d(i,j) = 0 \quad \text{(full linkability, i.e., } P(a_i \sim_{r(A)} a_j) = 1 \text{ or } P(a_i \sim_{r(A)} a_j) = 0).$$

Unlinkability of Arbitrary Many Items Let $\{a_1, a_2, \ldots, a_k\}$ be a subset of A containing k items. The equivalence relation $\sim_{r(A)}$ on A induces a relation on the subset $\{a_1, a_2, \ldots, a_k\}$. The probability that the relation $\sim_{r(A)}$ restricted to $\{a_1, a_2, \ldots, a_k\}$ is the same as the original relation is denoted by:

$$P(\sim_{r(A)} |\{a_1, \ldots, a_k\}) = P(\sim_{r(A)}) \quad \text{(relation distribution remains unchanged)}.$$

Attacker Models Two types of attacks are considered for unlinkability:

1. **Existential break**: Any pair of items $(a_i, a_j) \in A$ experiences a change in their linkability probabilities due to the attacker's actions.
2. **Selective break**: A subset of items is chosen by the attacker, and the linkability probabilities between one item in the subset and all other items are altered.

Unlinkability Between Sets In some systems, items are linked to elements from a separate set, such as a set of users $U = \{u_1, \ldots, u_n\}$ and actions $A = \{a_1, \ldots, a_m\}$. We define an equivalence relation $\sim_{r(U,A)}$ that links the actions in A to users in U, forming equivalence classes on A based on the users who performed the actions.
For example, in a communication system, the set A represents messages sent, and U represents users. A message sent by user u_i is related to u_i but should not be linkable to u_j for $j \neq i$ from the attacker's perspective.

Optimal Unlinkability For an equivalence relation $\sim_{r(A)}$ on a set A of size $|A| > 1$, the degree of unlinkability between any two items in A cannot be guaranteed to be $d(i, j) = 1$ for all pairs if the sizes of the equivalence classes are known. Thus, perfect unlinkability is not achievable if the structure of the equivalence classes is revealed.

Anonymity via Unlinkability Anonymity is refined as unlinkability between subjects and actions. This is commonly applied in cryptographic protocols to protect privacy:

- **Sender/Receiver anonymity**: No message is linkable to any sender or receiver.
- **Relationship anonymity**: Unlinkability of "who communicates with whom".

Degree of Anonymity via Unlinkability Let $U = \{u_1, \ldots, u_n\}$ be a set of users, and $A = \{a_1, \ldots, a_m\}$ a set of actions. Anonymity can be defined by the unlinkability between elements in these sets. The degree of anonymity of a user u_i with respect to an action a_j is given by the degree of unlinkability between those items.
For two items $a_i \in A$ and $u_j \in U$:

$$d(a_i, u_j) = -P(a_i \sim_{r(A,U)} u_j) \log_2 P(a_i \sim_{r(A,U)} u_j) - P(a_i \not\sim_{r(A,U)} u_j) \log_2 P(a_i \not\sim_{r(A,U)} u_j),$$

where $P(a_i \sim_{r(A,U)} u_j)$ and $P(a_i \not\sim_{r(A,U)} u_j)$ are the probabilities of the relationship between the action and the user being related or unrelated, respectively [7].
Minimum Disclosure Minimum Disclosure is a privacy principle that ensures that users provide only the necessary information required to establish a claim, avoiding the sharing of extraneous details. This principle is crucial in maintaining user privacy, particularly in authentication scenarios.[7].

Mathematical Definition of Minimum Disclosure Let $U = \{u_1, u_2, \ldots, u_n\}$ represent the set of users, and $C = \{c_1, c_2, \ldots, c_m\}$ represent the set of credentials or attributes associated with users. Suppose a system verifies a claim Q (such as proving that a user is over 18 years old), to minimize the disclosed information. The principle of minimum disclosure can be formally expressed as:

$$\text{Claim} : Q(u_i) \quad \text{for} \quad u_i \in U$$

where the disclosed information $D(u_i)$ should satisfy:

$$D(u_i) \subseteq C \quad \text{and} \quad D(u_i) \text{ is minimal such that } Q(u_i) \text{ holds},$$

i.e., the disclosed information is minimized to only the necessary attribute(s) $c_k \in C$, such that:

$$D(u_i) = \{c_k | Q(u_i) \text{ holds with } c_k\}.$$

The principle ensures that unnecessary attributes (e.g., full birthdate) are not disclosed, and only the proof (e.g., proof of age) is shared, in compliance with data minimization standards (e.g., GDPR).

Implication for Data Minimization Let $\mathcal{D}$ represent a data collection set. The system must guarantee that for any claim $Q(u_i)$, the data collected is:

$$\mathcal{D} = \{c_k \in C | c_k \in D(u_i)\}.$$

where:

$$\forall c_k \in C \quad c_k \notin D(u_i) \quad \text{unless required by} \quad Q(u_i).$$

Selective Revocation There should be revocation mechanisms in case a user misbehaves or his/ her credential is compromised. The system will have to revoke the credential (or pseudonym) of a user without interfering with the privacy of other users. This may be done with either nullifiers (which are tokens based on credentials that have been revoked) or accumulating (where revocations are handled with cryptographic sets). Newer schemes, such as threshold revocation based on Publicly Verifiable Secret Sharing (PVSS), decentralize revocation control to prevent single points of failure [39].

Mathematical Definition of Selective Revocation Let $P = \{p_1, p_2, \ldots, p_n\}$ be a set of pseudonyms or credentials associated with users, and let $R = \{r_1, r_2, \ldots, r_m\}$ be a set of revocation tokens. Selective revocation can be expressed through a function $\text{Rev}(p_i)$ that allows the revocation of a specific pseudonym p_i without affecting the privacy of other users. This can be modeled as:

$$\text{Rev}(p_i) = \text{Nullifier}(p_i) \quad \text{for} \quad p_i \in P$$

where the *nullifier* $\text{Nullifier}(p_i)$ is a cryptographic token that is associated with the revoked credential, ensuring that the revocation does not leak additional information about the user or their other credentials.

Threshold Revocation with Publicly Verifiable Secret Sharing (PVSS) In more advanced schemes, such as threshold revocation, the control over revocation is distributed, ensuring no single point of failure. Let S represent the secret shares, and $T \subseteq S$ represent the threshold subset of shares required to revoke a credential. The threshold revocation scheme is modeled as:

$$\text{Rev}(p_i) = \text{Threshold}_T(S) \quad \text{where} \quad T = \{t_1, t_2, \ldots, t_k\} \subseteq S.$$

This ensures that revocation is only possible when the required threshold number of shares T is reached, thereby decentralizing control over the revocation process.

Implication for Privacy The selective revocation scheme ensures that:

$$\text{Rev}(p_i) \quad \text{does not leak information about} \quad P \setminus \{p_i\}.$$

Thus, the revocation of one credential or pseudonym does not expose the private information of other users or their credentials.

2.1.2 Cryptographic Building Blocks

Zero-Knowledge Proofs (ZKPs) are cryptographic protocols, which enable a prover to certify that he or she knows some particular piece of information (e.g., a valid credential or a mathematical statement) to a verifier, without leaking any further information about the information. This is particularly important in privacy-preserving systems, where authentication of legitimacy must be compatible without the exposure of any sensitive information.

Core Concepts
I. Completeness: When the statement is true, an honest verifier will be convinced by the honest prover.

$$P(\text{Verifier accepts} \mid \text{Statement is true}) = 1 \tag{2.1}$$

It follows that there is a probability of 1 when the statement is true that the verifier accepts the proof.
II. Soundness: In case the statement is false, no cheating prover can make the verifier believe it to be true.

$$P(\text{Verifier accepts} \mid \text{Statement is false}) \leq \epsilon \tag{2.2}$$

Where ϵ is a small probability (usually called negligible). This means that there is very little chance of a dishonest prover succeeding to convince the verifier.
III. Zero-Knowledge: The only thing that the verifier does learn is nothing more than whether the statement is true. The verifier does not get to know any more information about the underlying data[20][11].

$$\text{Verifier's knowledge} = \emptyset \tag{2.3}$$

The communication between prover and verifier does not reveal more than just the truth of the statement.

ZK-SNARKs(Succinct Non-Interactive Arguments of Knowledge) are one of the most popular types of ZKPs applied in modern systems; they allow generating proofs and verifying them rather efficiently. They are concise, i.e., the size of proof is small (approximately 200 bytes), and the verification is extremely fast (in the order of 300ms). This has made ZK-SNARKs suitable for real-time authentication systems, like in the one we are suggesting in HALP, where fast verification is desired due to the user experiences on systems like this need to be smooth [14].
Although ZK-SNARKs are efficient, they do require a trusted setup stage, the compromise of which may be a vulnerability. Such trade-offs are also what make ZK-SNARKs different to other ZKP systems, such as Bulletproofs, which only require a non-trusted setup but are associated with even greater proof sizes [22]. This kind of

trade-off is systematically addressed in HALP via the Circom, ZKP circuit designing framework, and SnarkJS, a proof generation library. These tools accelerate the use of ZK-SNARKs in the efficient implementation of HALP at the same time, making the system fast and secure[14].

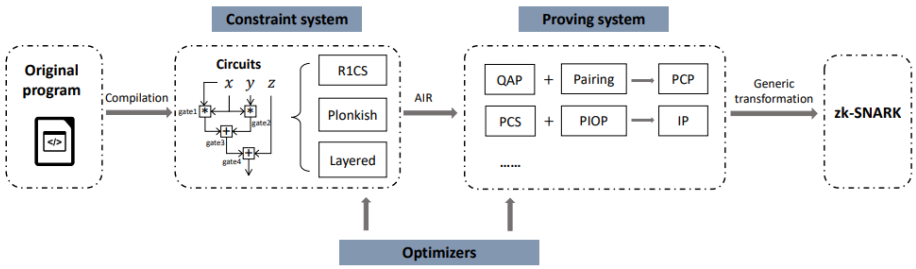


Figure 2.1: General steps of converting a high-level program to a ZK-SNARK

The above diagram (Figure 2.1) describes the procedure of how to encode a high-level program into ZK-SNARK (Zero-Knowledge Succinct Non-Interactive Argument of Knowledge). The process of the work starts with the assembly of the initial program in the form of a circuit. It is then placed in a constraint structure, where it is turned into a mathematical representation of the computation by the application of techniques like R1CS, Plonkish, or Layered circuits. The circuit obtained is subjected to a proving system, adopting methods such as QAP, PCS, pairing, or PIOP+IP to prove its correctness. Then, a number of optimizations are additionally carried out to make the proof more efficient (shorter proof size and faster to verify). The sequence is concluded by a generic transformation that turns the proof into a ZK-SNARK and achieves the desired result, efficient, succinct, and non-interactive verification (and thus finishes turning a high-level program into a cryptographically verifiable program).

Formal Languages A formal language L is a subset of the set of all finite strings Σ^*, where Σ is an alphabet and Σ^* is the set of all possible finite strings (including the empty string). The alphabet Σ consists of symbols or letters, and words in L are strings of symbols formed according to a grammar.

Definition 1. Let Σ be a finite alphabet and Σ^* be the set of all strings over Σ. A *formal language* L is a subset of Σ^*, i.e., $L \subseteq \Sigma^*$.

Decision Functions A decision function determines whether a given string belongs to a formal language. It maps strings from Σ^* to a boolean value indicating membership in the language.

Definition 2. A *decision function* R for a language L is a function:

$$R : \Sigma^* \to \{\text{true}, \text{false}\},$$

where $R(x) = \text{true}$ if $x \in L$ and $R(x) = \text{false}$ otherwise.

Definition 3. The associated language L_R of a decision function R is:

$$L_R := \{x \in \Sigma^* \mid R(x) = \text{true}\}.$$

Instance and Witness In zero-knowledge proofs, an *instance* is a publicly known input to a statement, while a *witness* is a private input that helps to prove the truth of the statement.

Definition 4. Let Σ_I and Σ_W be two alphabets, where Σ_I represents the instance alphabet and Σ_W represents the witness alphabet. The *decision function* R is defined as:

$$R : \Sigma_I^* \times \Sigma_W^* \to \{\text{true}, \text{false}\}, \quad (i, w) \mapsto R(i, w),$$

where $i \in \Sigma_I^*$ is an instance and $w \in \Sigma_W^*$ is a witness. The associated language L_R is:

$$L_R := \{(i, w) \in \Sigma_I^* \times \Sigma_W^* \mid R(i, w) = \text{true}\}.$$

Rank-1 Quadratic Constraint Systems (R1CS) A Rank-1 Quadratic Constraint System (R1CS) is a system of quadratic equations over a finite field that represents a computation or relation. It consists of a set of constraints that must be satisfied by an instance and a witness pair.

Definition 5. A *Rank-1 Quadratic Constraint System (R1CS)* is a set of k equations of the form:

$$\left(a_1 + \sum_{j=1}^{n} a_j I_j + \sum_{j=1}^{m} a_{n+j} W_j\right) \cdot \left(b_1 + \sum_{j=1}^{n} b_j I_j + \sum_{j=1}^{m} b_{n+j} W_j\right) = c_1 + \sum_{j=1}^{n} c_j I_j + \sum_{j=1}^{m} c_{n+j} W_j,$$

where $I_1, \ldots, I_n$ are instance variables, $W_1, \ldots, W_m$ are witness variables, and $a_i, b_i, c_i \in F$ for $i = 1, 2, \ldots, n + m$ are constants in the finite field F.

R1CS Satisfiability An R1CS is satisfiable if there exists an assignment of values to the instance and witness variables that satisfies all the constraints in the system.

Definition 6. The *R1CS satisfiability* problem is the decision problem of determining whether there exists a solution $(I, W) \in \Sigma_I^* \times \Sigma_W^*$ such that:

$$R_{\text{R1CS}}(I, W) = \text{true},$$

where R_{R1CS} is the decision function associated with the R1CS.

Quadratic Arithmetic Program (QAP) A *Quadratic Arithmetic Program (QAP)* is a generalization of the R1CS, where the constraints are quadratic polynomials over a finite field. These programs are used to describe computations in zero-knowledge proofs.

Definition 7. A *Quadratic Arithmetic Program (QAP)* consists of a set of polynomials $P_1(x), P_2(x), \ldots, P_k(x)$ over a finite field F, where each polynomial $P_i(x)$ describes a constraint in the program. The QAP is represented as:

$$P_i(x) = A_i \cdot B_i - C_i, \quad \text{for} \quad i = 1, 2, \ldots, k,$$

where $A_i, B_i, C_i \in F$ are polynomials over the field, and the program defines the set of valid assignments to the instance and witness variables.

QAP Satisfiability QAP satisfiability refers to the problem of determining whether there exists an assignment to the instance and witness variables that satisfies all the polynomial equations in the QAP.

Definition 8. The *QAP satisfiability* problem is the decision problem of determining whether there exists an assignment $(I, W) \in \Sigma_I^* \times \Sigma_W^*$ such that:

$$R_{\text{QAP}}(I, W) = \text{true},$$

where R_{QAP} is the decision function associated with the QAP [27].

Anonymous Credentials [5] enable users to prove their identity without disclosing it, and it is the key capability of privacy-preserving authentication systems. A Camenisch-Lysyanskaya (CL) signature is a widely known example of an anonymous credential system whose construction builds off the use of a variant of public key encryption, which supports selective disclosure of one attribute without disclosure of others. An example is that a user can be able to prove that they are over 18 years old without publishing their age number. Hyperledger Ursa Coconut scheme [25] is a threshold issuance and selective disclosure of credentials used in HALP. Threshold issuance enables credential issuance to be distributed, so a single authority can not take full control over the credentialing process, which increases the trust level and also makes the system more secure and reliable. Selective disclosure implies that users may not have to disclose their whole credentials (such as the correct date of birth) but can release certain characteristics of their credentials (such as the fact that they are over 18). In contrast to the U-Prove with centralized issuers, Coconut does not require a single centralized issuer, and the trust is spread among entities. This distributed model decreases the chance of a common point of failure, and has a more resistant structure against assaults or abuse [25].

Pseudonymous Identifiers & Nullifiers Pseudonymous identifiers [20] play an important role in privacy enhancing authentication systems, where they enable the privacy of users of the systems and avoid tracking in case of multiple sessions. Rather than applying true, persistent identifiers (e.g., email addresses or usernames), such systems are pseudonymous. A pseudonym is a new identifier issued specifically to a single session or interaction, where it is generated as a cryptographic hashing of the secret credentials of the user, and some form of randomization, which may be a nonce (a one-time per-interaction randomly-generated value). Such a process guarantees that as long as a pseudonym is intercepted or acquired by a third party, it remains completely anonymous to the user who only possesses a secret key or understands the requirement of the first credential.

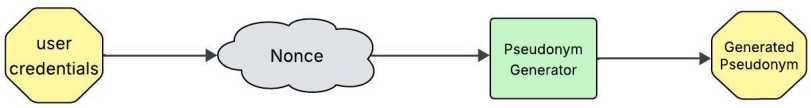


Figure 2.2: Pseudonym Generation

Core Concepts of Pseudonymous Identifiers: The main idea of pseudonym usage is to avoid connecting identities between sessions to guarantee privacy with the ability to authenticate. Essentially a pseudonym is a temporary, identifying text that can be associated with a particular action or session. Notably, pseudonyms are cryptographically derived and may be based on a master secret (e.g., a private key) with an appendage known as a nonce, ensuring that each time a user logs on, a different pseudonym will be produced.

Pseudonym Generation: pseudonyms are created by applying a nonce on a secret credential (normally a cryptographic key) of a user (Figure 2.2). This would give every session a distinct identifier and ensure that there is no identity relatability between sessions.

Linkability: Although the same person might use several pseudonyms across different time intervals, it would be impossible to connect the same to that individual unless a Priv key or some other sensitive information is accessed. This helps in keeping the activity of the user unlinkable between sessions [47].

Revocation: HALP employs nullifiers to handle revocation in ways that do not interfere with privacy. A nullifier is a cryptographic tag produced on a credential usage or revocation. It blocks the reuse of the credentials once it has been used. Nullifiers based on secret information of the user perform the task of handling revocation without uncovering the identity of the user or any other personal information. This system enables blocking malicious or compromised users efficiently because it does not violate user privacy[17]. This makes sure that a malicious user is not able to use the system by exploiting weak credentials repeatedly [22].

HALP uses Zero-Knowledge Proofs (ZKPs), anonymous credentials, and one-time pseudonyms with nullifiers to achieve its main objectives of strong privacy, support scalable authentication, and revocable authentication. Those cryptographic schemes are used to solve the fundamental problems of linkability, anonymity, and revocation in privacy-preserving systems.

Unlinkability: Every time a user logs in a different pseudonym is created and hence the activities of a user cannot be linked together in different login sessions, even when using the same service provider [31].

Privacy Preservation: As pseudonyms are used, through HALP, users can securely authenticate themselves without exposing their identity and other sensitive personal information [20].

2.1.3 Infrastructure Components

Revocation List: Revocation registry [44] is essential infrastructure to handle compromised or malicious credentials, retaining privacy of legitimate users. Revocation registry helps in a way that when a user credential is compromised or revoked, the user is not capable of authenticating using the credential again, leading to the prevention of abuse or misuse.

On-chain Revocation: As an example of transparency, on-chain revocation makes use of the blockchain, (e.g., Ethereum smart contracts or zkRollups (e.g., Aztec)). This makes revocations immutable because when a blockchain records a revocation, the consequence of such revocation is publicly trackable and cannot be changed using any central source. Smart contracts are applied in a transparent revocation algorithm, which allows all participants to check whether the credential is valid or not without knowing the identity of the user [14].

Low Latency Off-Chain Revocation: Off-chain revocation can be used to mitigate issues of scalability and performance, particularly in high-traffic systems. Revocation data is stored off-chain, usually in a database or server, attested to by signed statements that a credential was actually revoked. Off-chain systems have the benefit of low latency, since revocation can be used at a quick time without the overhead of engaging the blockchain on each and every transaction [39].
One solution where revocation is checked through attestation is by a trusted party allowing optimistic systems to exist, which is fast but has some degree of security. Revocation decisions are trusted in such a system, but can be disputed or checked against an on-chain registry in case of a discrepancy [53][39].

Decentralized Identifiers (DIDs): Decentralized Identifiers (DIDs) represent a new kind of identifier designed to support self-sovereign identity systems, where a user is always in control of their own identifiers and the data that those identifiers reference. The use of DIDs forms a vital part of this HALP infrastructure since it ties the credentials of the user to a user-controlled identity that permits privacy-friendly authentication. DIDs bind credentials to an identity controlled by the user on a decentralized platform. And unlike the traditional identifications (like email addresses or government-issued IDs), DIDs are not connected to some centralized entity, which is why users are in full control of their personal information. This decentralization denies third parties the ability to store data and follow users or benefit directly from their data, and guarantees that individuals may use authentication to access services without depending upon a core provider. DIDs frequently are complemented with Verifiable Credentials (VCs), in which users may provide selective disclosure of aspects of their credentials without divulging other sensitive data [1].

2.1.4 Performance Optimization Tools:

ZK Proof Systems: Privacy-preserving authentication fundamentally requires ZK Proofs an,d in particula,r imposes performance issues to large systems. A few achievements inZK proof systems have been mamademake these proofs as scalable and efficientk as possible.

Groth16: Groth16 is a well-known ZK-SNARK protocol that gproduces small proofs with fast verification times(300ms). It ho,wever, needs a trusted setup, which is a security risk, unless done properly. Groth16 is suitable to problem applications such as HALP wh,ere efficient, low-latency proofs are necessary [19].

Definition: Proof System A proof system consists of the following components:

- $\mathcal{P}$: The prover algorithm,
- $\mathcal{V}$: The verifier algorithm,
- $\mathcal{S}$: The simulator algorithm,
- $\mathcal{L}$: The language defined by a Rank-1 Constraint System (R1CS).

A proof system is designed such that the prover $\mathcal{P}$ convinces the verifier $\mathcal{V}$ of the validity of a statement, without revealing unnecessary information.

Definition: Completeness A proof system is **complete** if, for every instance $x \in \mathcal{L}$, the prover $\mathcal{P}$ can generate a valid proof $\pi = P(A)$ (where A is relevant portion of set of credentials $\{c_1, c_2, \ldots\}$) that the verifier $\mathcal{V}$ will always accept:

$$\Pr[\mathcal{V}(\pi) = \texttt{accept} \mid x \in \mathcal{L}] = 1.$$

This means that if the statement is true (i.e., $x \in \mathcal{L}$), the verifier will always accept the proof generated by the prover.

Definition: Soundness A proof system is **sound** if no malicious prover can convince the verifier to accept a false proof. Formally, for every $x \notin \mathcal{L}$, the probability that a dishonest prover convinces the verifier to accept a proof is negligible:

$$\Pr[\mathcal{V}(\pi) = \texttt{accept} \mid x \notin \mathcal{L}] \leq \epsilon(\lambda),$$

where $\epsilon(\lambda)$ is a negligible function of the security parameter λ.

Definition: Zero-Knowledge A proof system is **zero-knowledge** if, given a proof π, the verifier learns nothing beyond the validity of the statement being proven. More formally, there exists a simulator $\mathcal{S}$ that can generate a proof π indistinguishable from a real proof generated by the prover, without knowing the witness:

$$\text{Sim}(\mathcal{L}, x) \approx \mathcal{P}(\mathcal{L}, x, w),$$

where w is the witness, $\mathcal{P}$ is the prover algorithm, and $\mathcal{S}$ is the simulator.

Prover and Verifier Algorithms The prover $\mathcal{P}$ and verifier $\mathcal{V}$ algorithms are formally defined as:

- $\mathcal{P}(x, w)$ takes as input an instance $x \in \mathcal{L}$ and a witness w, and outputs a proof π.
- $\mathcal{V}(x, \pi)$ takes as input an instance $x \in \mathcal{L}$ and a proof π, and outputs either `accept` or `reject`.

The proof π is sent by the prover to the verifier, and the verifier uses the verification algorithm $\mathcal{V}$ to decide whether to accept or reject the proof.

Groth16 Parameters: Let G_1 and G_2 be two cyclic groups of prime order r, with generators $g_1 \in G_1$ and $g_2 \in G_2$, and let $e : G_1 \times G_2 \to G_T$ be a bilinear map. The Groth16 protocol involves the following parameters:

- G_1, G_2: Two groups of prime order r,
- g_1, g_2: Generators of G_1 and G_2,
- e: A bilinear map,
- $\mathcal{CRS}$: The common reference string (CRS) generated during the setup phase.

Setup Phase: The setup phase takes the Rank-1 Constraint System R and generates the common reference string $\mathcal{CRS}$ and the simulation trapdoor $\mathcal{ST}$. The setup algorithm is as follows:

$$(\mathcal{CRS}, \mathcal{ST}) \leftarrow \text{SETUP}(R).$$

5 random invertible elements $\alpha, \beta, \gamma, \delta,$ and τ are taken from the field Fr of the protocol to produce the simulation trapdoor ST:

$$\mathcal{ST} = (\alpha, \beta, \gamma, \delta, \tau)$$

The CRS is composed of elements from G_1 and G_2, and is shared by both the prover and the verifier. Specifically, the CRS consists of two tuples:

$$\mathcal{CRS}_{\text{QAP}} = (\text{CRS}_{G_1}, \text{CRS}_{G_2}),$$

where:

$$\begin{aligned}
\text{CRS}_{G_1} = \Bigg\{ & g_1^{\alpha}, g_1^{\beta}, g_1^{\delta}, \left(g_1^{\tau^j}\right)_{j=0}^{\deg(T)-1}, \left(g_1^{\frac{\beta A_j(\tau)+\alpha B_j(\tau)+C_j(\tau)}{\gamma}}\right)_{j=0}^{n}, \\
& \left(g_1^{\frac{\beta A_{j+n}(\tau)+\alpha B_{j+n}(\tau)+C_{j+n}(\tau)}{\delta}}\right)_{j=1}^{m}, \left(g_1^{\frac{\tau^j T(\tau)}{\delta}}\right)_{j=0}^{\deg(T)-2} \Bigg\} \\
\text{CRS}_{G_2} = \Big\{ & g_2^{\beta}, g_2^{\gamma}, g_2^{\delta}, \left(g_2^{\tau^j}\right)_{j=0}^{\deg(T)-1} \Big\}
\end{aligned}$$

The elements $g_1^{\alpha}, g_1^{\beta}, g_1^{\delta}, \ldots$ are computed using the setup parameters $\alpha, \beta, \gamma, \delta \in \mathbb{F}_r$. The CRS components are shared by both the prover and the verifier.

Prover Phase: The prover, given the CRS, an instance I, and a witness W, computes a proof π as follows:

$$\pi = \mathcal{P}(R, \mathcal{CRS}, I, W).$$

The proof π consists of three group elements: two from G_1 and one from G_2, which are derived based on the witness and the instance.
To compute the proof, the prover evaluates the polynomial $P(I; W)$, derived from the Quadratic Arithmetic Program (QAP). The prover then divides this polynomial by the target polynomial T of the QAP. As $P(I; W)$ is derived from a valid solution to the R1CS, the division yields a polynomial $H := P(I; W)/T$, where $\deg(H) < \deg(T)$.
The prover then evaluates the polynomial $(H \cdot T)/\delta$ in the exponent of the generator g_1 at the secret point τ. Let $H(x)$ be the quotient polynomial P/T:

$$H(x) = H_0 \cdot x^0 + H_1 \cdot x^1 + \ldots + H_k \cdot x^k$$

To evaluate $(H \cdot T)/\delta$ at τ in the exponent of g_1, the prover computes using CRS:

$$\frac{H(\tau) \cdot T(\tau)}{\delta} = \left(\frac{\tau^0 \cdot T(\tau)}{\delta}\right)^{H_0} \cdot \left(\frac{\tau^1 \cdot T(\tau)}{\delta}\right)^{H_1} \cdots \left(\frac{\tau^k \cdot T(\tau)}{\delta}\right)^{H_k}$$

Then the prover takes a sample of two random field elements $r, t \in \mathbb{F}_r$. Using CRS, instance variables $I_1, \ldots, I_n$, witness variables $W_1, \ldots, W_m$, the following is computed:

$$g_1^W = \left(\frac{\beta A_{1+n}(\tau) + \alpha B_{1+n}(\tau) + C_{1+n}(\tau)}{\delta}\right)^{W_1} \cdots \left(\frac{\beta A_{m+n}(\tau) + \alpha B_{m+n}(\tau) + C_{m+n}(\tau)}{\delta}\right)^{W_m}$$

$$g_1^A = g_1^{\alpha} \cdot \frac{A_0(\tau)}{\delta_1} \cdot \left(\frac{A_1(\tau)}{\delta_1}\right)^{I_1} \cdots \left(\frac{A_n(\tau)}{\delta_1}\right)^{I_n} \cdot \left(\frac{A_{n+1}(\tau)}{\delta_1}\right)^{W_1} \cdots \left(\frac{A_{n+m}(\tau)}{\delta_1}\right)^{W_m} \cdot \left(\frac{\delta}{\delta_1}\right)^{r}$$

$$g_1^B = g_1^{\beta} \cdot \frac{B_0(\tau)}{\delta_1} \cdot \left(\frac{B_1(\tau)}{\delta_1}\right)^{I_1} \cdots \left(\frac{B_n(\tau)}{\delta_1}\right)^{I_n} \cdot \left(\frac{B_{n+1}(\tau)}{\delta_1}\right)^{W_1} \cdots \left(\frac{B_{n+m}(\tau)}{\delta_1}\right)^{W_m} \cdot \left(\frac{\delta}{\delta_1}\right)^{t}$$

$$g_2^B = g_2^{\beta} \cdot \frac{B_0(\tau)}{\delta_2} \cdot \left(\frac{B_1(\tau)}{\delta_2}\right)^{I_1} \cdots \left(\frac{B_n(\tau)}{\delta_2}\right)^{I_n} \cdot \left(\frac{B_{n+1}(\tau)}{\delta_2}\right)^{W_1} \cdots \left(\frac{B_{n+m}(\tau)}{\delta_2}\right)^{W_m} \cdot \left(\frac{\delta}{\delta_2}\right)^{t}$$

$$g_1^C = g_1^W \cdot \frac{H(\tau) \cdot T(\tau)}{\delta} \cdot \left(\frac{\delta}{\delta_1}\right)^{r} \cdot \left(\frac{\delta}{\delta_1}\right)^{-r-t}$$

Finally, the generated ZK-SNARK proof is

$$\pi = (g_1^A, g_1^C, g_2^B)$$

Verification Phase: Given the CRS, the instance $I =< I_1, \ldots, I_n >$, and the proof π, To verify the zk-SNARK,the verifier computes the following curve point:

$$\begin{aligned} g_1^I = &\left(\frac{\beta A_0(\tau) + \alpha B_0(\tau) + C_0(\tau)}{\gamma}\right) \\ &\cdot \left(\frac{\beta A_1(\tau) + \alpha B_1(\tau) + C_1(\tau)}{\gamma}\right)^{I_1} \\ &\cdots \\ &\cdot \left(\frac{\beta A_n(\tau) + \alpha B_n(\tau) + C_n(\tau)}{\gamma}\right)^{I_n} \end{aligned} \tag{2.4}$$

Then, $\pi = (g_1^A, g_1^C, g_2^B)$ is verified using the group element based on the equation:

$$e(g_1^A, g_2^B) = e(g_1^{\alpha}, g_2^{\beta}) \cdot e(g_1^I, g_2^{\gamma}) \cdot e(g_1^C, g_2^{\delta})$$

which if satisfied, the verifier outputs `accept`, else the verifier outputs `reject`.

Proof Simulation The ST computed during setup is be discarded at the end of the phase, as it can be used to forge proofs. For instance I of R1CS language L_R, a zk-SNARK for L_R is forged or simulated if it is not generated using witness W but it still passes verification (where $(I; W)$ is a word in L_R). If an attacker accesses the required Groth_16 parameters, a QAP of the problem, a CRS and its ST, they can generate a proof for the instance without access of others ZK-SNARKS or knowledge of witness by computing g_1^c for instance I:

$$g_1^C = g_1^{\frac{A \cdot B}{\delta}} \cdot g_1^{-\frac{\alpha \cdot \beta}{\delta}} \cdot g_1^{-\frac{\beta A_0(\tau) + \alpha B_0(\tau) + C_0(\tau)}{\delta}} \cdot \left(g_1^{-\frac{\beta A_1(\tau) + \alpha B_1(\tau) + C_1(\tau)}{\delta}} \right)^{I_1} \cdots \left(g_1^{-\frac{\beta A_n(\tau) + \alpha B_n(\tau) + C_n(\tau)}{\delta}} \right)^{I_n} \tag{8.10}$$

and hence produces $\pi_{\text{forged}} = (g_1^A, g_1^C, g_2^B)$, which passes verification. Note that the forger can compute $g_1^{A \cdot B}$ and all factors involved in finding g_1^C, as generators g_1, g_2 and the ST are known, making the ST both necessary and sufficient to compute π_{forged}. If the ST is unknown, deriving $g_1^{\alpha \cdot \beta}$ from g_1^{α} and g_1^{β} is infeasible due to the computational Diffie-Hellman assumption [27].

PLONK/PLOOKUP: PLONK [24] is a universal variant of zk-SNARK machine, such that it supports a universal trusted setup, which is more flexible to systems with multiple types of credentials or those that might need more frequent updates. PLOOKUP improves PLONK by increasing the speed of proof generation in systems with large datasets (e.g,. multi-credential systems)

Lightweight Clients Besides enhancingserver-sidee performance, the use of lightweight clients must be optimized when using HALP, particularly in a scenario such as that of mobile devices or web browsers since computing resources can be a limitation in such cases [53].

WebAssembly (WASM): WebAssembly (WASM) is an instruction format that is binary and permits the execution of ZKP circuits in web browsers. With WASM-SNARK, a non-computational client would be able to create and check zk-SNARK proofs originated and verified in the web browser without involving any server-side computation. This decreases latency and allows users to authenticate would within a short period of time, and in a manner that is personal, and does not require the full back-end infrastructure.

Mobile Optimization: Ristretto curves can be used (as in the case of Dalek cryptographytoto ensure the optimization of the performance on mobile devices. Such curves would be efficient with limited resources, so that ZKP-based authentication systems would be usable even on smartphones [53].

2.2 Review of Existing Research

A related system named CL-signatures was presented in 2001 by Camenisch and Lysyanskaya [5], under which users may demonstrate that they possess some credential without necessarily proving their identity. The primary drive behind this work was to solve the issue that traditional digital credentials, such as digital IDs

or certificate link the actions of a user with their real world, which can be a breach of privacy. The aimTheir activity aimedelop the means through which users could demonstrate that they possess a credential (e.g., to prove they are older enough) without disclosing additional personal data. Nevertheless, the system also has certain limitations. Creation and verification of these credentials is extremely complex, resource intensive and slow to compute. It also requires a centralized issuer, an authoritative figure deciding who should receive credentials. This is a weakness for the system because security depends on the issuer being trustable. In the case of HALP, this difficulty is addressed by substituting CL signatures with faster, less computationally complex mechanisms such as BBS+ and ZK-SNARKs, alongside a blind signature mechanism so that the issuer does not have any information on the holder.

Brands (2000) [4] proposed blind signatures, where the issuer signs a credential without learning its contents. This was motivated by the desire to guard against the privacy of the users such that the issuer of the credentials cannot track or trace the credential to identify the user. This system is particularly desirable in the cases where one party tries to prove to the other party a certain property (such as the age) without releasing any other information. The aim of the paper has been to demonstrate how users may be authenticated without the loss of privacy by using blind signatures. The system has several problems though: there is no method of revocation (or cancellation) of credentials in case this is necessary, and it requires the issuer to be trustful, which poses a risk in case the system in which the issuer resides is compromised.

Ben-Sasson [3] and others (2014) created a new cryptographic methodology known as ZK-SNARKs. The rationale was that this was done to make Zero-Knowledge Proofs (ZKPs) more workable in real systems. ZKPs enable a user to demonstrate that they know something (e.g., a valid credential) without also showing any information about it. However, earlier incarnations of ZKPs were far too slow to be efficient and verification of proofs were too cumbersome. They wanted to develop ZK-SNARKs, which are less significant, quicker, and able to be used in real time. These proofs measure approximately 200 bytes and can be verified in 300ms, which is an application to high response time systems (online authentication). But ZK-SNARKs involve an expensive trusted setup stage that is potentially dangerous to mishandle, as it may reveal weaknesses.

A new kind of zero-knowledge proof was presented by Bunz et al. (2018) [22], named Bulletproofs, intending to enhance ZK-SNARKs that performed a trusted setup by developing a transparent ZKP system. This openness implies that Bulletproofs do not require a trusted setup hence decreasing the integrity risk of disclosing sensitive data. Their goal in the work was to generate smaller proofs (in comparison to earlier schemes), and to offer quicker verification periods and to evade the complexity and threat of trusted preparations. Bulletproofs however are not completely perfect: the proofs are bigger (roughly 12KB), which requires more bandwidth, and it takes longer to verify them than ZK-SNARKs.(Bunz et al, 2018). HALP favors the ZK-SNARKs due to their efficiency on verification and can also utilize Bulletproofs when minimizing trusted setup is the key priority over the size of the proof.

Kosba et al. [20] (2016) developed a system known as Hawk manufactured with blockchain technology to deal with credential revocation. This was motivated by the fact that conventional revocation mechanisms such as Certificate Revocation Lists (CRLs) frequently undermine the privacy of users by making users publicly identifiable by having their credentials revoked. Hawk resolves this by making use of the blockchain to deposit nullifiers, which can cancel out credentials without displaying the identity of the user. It is also decentralized, which implies the lack of ability of its controlling central authority. This was aimed at creating a method through which credentials can be revoked, at the same time keeping privacy and being transparent. Revocation with blockchain however introduces latency (delays) and the existence of smart contracts which are not always available in any environment. HALP has a hybrid revocation approach, in that either on-chain or off-chain approaches can be used to implement revocation, and this leads to it being more flexible and to the minimization of latency.

Roio et al. [39] (2024) propose SD-BLS, a novel system for selective disclosure and revocation using BLS signatures. Their aim is to decentralize and strengthen credential revocation, which is often a central point of failure in current systems. SD-BLS distributes revocation governance among multiple issuers through publicly verifiable secret sharing (PVSS), requiring configurable consensus to activate revocation. In this way, it protects holders from corrupt actions by a sole authority. Although their multi-party threshold scheme improves security, it introduces complexity and coordination overhead. HALP implements revocation through a nullifier registry which is much more lightweight.

Rajasekaran et al. [38] (2024) introduce TPAAS (a novel scheme to support privacy-preserving anonymous authentication in online trading platforms), as part of their 2024 paper. The drive to undertake this research is based on the privacy issue and inefficiencies of existing authentication systems when used in such environments and may lead to delays in account creation processes, authentication-related bottlenecks, and communication expenses. TPAAS is intended to offer a combination of conditional privacy and confidential interaction without anyone leaking out personal information, and at the same time, the user can be removed by the system in case they are needed in case of a dispute or malicious actions. The main goal of this scheme is to provide a powerful privacy preserving mechanism that would adapt to an environment of online trading where the user is allowed to act with anonymity and yet be accountable. The scheme is however not devoid of drawbacks. Rajasekaran et al. admit that even though TPAAS can guarantee safe user interfaces and low-cost computations, the system might encounter scale-ability issues i.e. in managing numerous users, as additional cost of establishing and maintaining trust relationships and resolution of disputes will be experienced. This paper describes an increasing demand in privacy-preserving authentication systems and gives the significance of whether it satisfies performance at high traffic level, which may be crucial in large-scale systems.

Zhao et al. [41] (2024) discuss limitations of centralized trust models in cross domain authentication schemes, with the specific emphasis on the inability of the approach to support privacy preservation and revocation techniques. In their proposed

scheme they have used ZK-SNARKs with the objective of making the cross-domain system anonymous so that when authentication is done, the user is guaranteed of being capable of proving to various entities that he is a valid user of the system without explicitly providing sensitive personal data. The most important goal of the authors is to bring the balance between user privacy and authorization of the credential revocation, where they use the authorization- then-proof structure that needs neither the credential nor the combination of the hash and the timestamp to reveal user identity, just has the efficient way to revoke them in case of a need. Although the scheme is a significant extension to anonymous cross-domain authentication, it has some limitations. The low on-chain storage cost (64 bytes) to revoke is among the focal points, but even though this utility is compact it is not necessarily enough to accommodate more sophistically structured or extensive revocations. Also, the sheer use of ZK- SNARKs poses the risk of trusted setup and the risk of the vulnerability in the decentralized context. These shortcomings notwithstanding, the authors' contributions are enormous as they present an anonymous authentication arrangement that can evolve into several cross-application development procedures.

Sanders and Traore [40] (2024) discuss this recent work and concentrate on the security advantages of eliminating the need to have secret keys held by issuers within the anonymous credential system. It is a significant issue in a classical anonymous credential system, where junking of the issuer may result in violation of user privacy. They have presented a smaller issuer-hiding version of authentication, making sure that even in the case where the issuer is compromised, user credentials still stay safe. This work is driven by the fact that there is a growing demand to have more secure ways of anonymous systems more so where one is dealing with sensitive applications like voting or financial transactions whereby the information provided by the users should be secure against malicious users. The proposed work aims at delivering an authentication system capable of issuing privileged credentials in a secure manner without centralized trust. Sanders and Traor's solution have a few limitations despite the innovative solutions it offers. Though the proposed mechanism greatly increases the level of security of an issuer, it provides a new complexity to the process of issuing and verifying credentials, which may result in slowing down the system with the decrease in its effectiveness. The authors address the trade-off between security and the complexity of the protocol, a factor about which future anonymous credential systems such as HALP will have to be cautious.

2.3 Summary of Key Findings

A critical analysis of the literature available shows that although many privacy-friendly authentication systems prove to protect the anonymity of the user, most fail to meet expectations in the areas of scalability, revocation, or decentralization. Previous schemes like Idemix and U-Prove have massive computational overhead and require entirely centralized trust, modern systems like ZK-SNARKs and Bulletproofs are more efficient, although they continue to have issues with revocations. Newer paradigms, such as TPAAS and the one by Zhao et al., make visible partial steps towards the problem of anonymous and revocable schemes, but at the costs of significant performance or complexity. In addressing these concerns, HALP attempts

to unify unlinkability, selective revocation and scalability via hybrid cryptography. We can see a comparison on three parameters below in Table 2.1.

Paper	**R**	**S**	**D**
Camenisch & Lysyanskaya (2001) [5]	×	×	×
Brands (2000) [4]	×	✓	×
Ben-Sasson et al. (2014) [15]	×	✓	×
Bünz et al. (2018) [22]	×	✓	×
Roio et al. (2024) [39]	✓	×	✓
Kosba et al. (2016) [20]	✓	×	✓
Rajasekaran et al. (2024) [38]	✓	×	×
Zhao et al. (2024) [41]	✓	✓	✓
Sanders & Traoré (2024) [40]	×	×	×

Table 2.1: Comparison of credential system properties across selected papers. R: Revocation, S: Scalability, D: Decentralization.

Chapter 3

Requirements, Impacts and Constraints

The HALP system is a privacy-oriented authentication system that responds to an increasing need to have secure, anonymous, and user-centered solutions of digital identities. It implements sophisticated cryptography systems, such as zk -SNARKs to demonstrate zero-knowledge, BBS+ signatures to do selective disclosure, and Poseidon hash algorithms to make users authenticate without helping hardware to reconstitute their sensitive information. It is designed in a modular architecture; core cryptographic functionality is segregated with service logic, and this enhances maintainability and scalability. The most important features of HALP include anonymous authentication, unlinkable sessions, selective revocation, scalability and low overhead to clients; they are all well balanced to ensure privacy, security and usability. The system has great advantages: it allows the users to have complete control of their online identities, reduce the possibility of data loss, and adhere to privacy rules. These benefits are however constrained by factors which include proper cryptographic implementation, that it relies on circuit files and that all services must be properly initialized and synchronized. Altogether, HALP can be considered a future-oriented authentication solution that introduces a new level of privacy and security in the decentralized identity systems.

3.1 System Overview

Below are detailed the threat modeling, functional requirements, and security requirements of the Hybrid Anonymous Login Protocol (HALP). The STRIDE threat model [33], [45] is discussed as a means of dealing with security threats of spoofing, tampering, and denial of service. The functional requirements are presented and consider anonymous authentication, unlinkable session, and selective revocation. Scalability and minimized overhead is also required so that both privacy and performance are ensured.

3.1.1 Threat Modeling

T1- Spoofing: An impersonation involves an enemy party impersonating a genuine user, hoping to gain entry to a system without having the appropriate credentials. In anonymous authentication systems, this type of impersonation takes place

where malicious users pretend to be genuine users to access restricted systems. The cryptographic mechanisms used in mitigation strategies normally allow users to be verified without revealing their identity; sound proofs and computationally binding commitments and signatures are examples of such strategies. These tools promote the objective of hindering impersonation whilst allowing authentication to be carried out anonymously.

T2- Tampering: The second threat vector entails alteration of data or system components in transit. Most anonymous authentication schemes, especially those with ZK-SNARKs, use signature schemes supporting witness hiding and computationally binding properties to prevent data forgery in an authentication. Furthermore, soundness property of zero knowledge proofs ensures that tampered proofs cannot be accepted.

T3- Repudiation: The third type of attack is known as repudiation, where users deny an action they have committed. An effective mitigation system must include mechanisms that aid auditability and traceability but do not jeopardize anonymity. HALP generates a nullifier for each session based on the private master secret, which is stored in a registry and is also used to generate proof of knowledge. If a user has indeed committed an action, then their nullifier for that session will be present in the registry and also be used for generating proofs required for authentication, preventing them from denying their actions.

T4- Information Disclosure: Illegal access to sensitive information is a looming threat, to mitigate which anonymous authentication systems implement selective disclosure so that only the required information is shared, and the information is presented in the form of zero knowledge proofs of knowledge. For example, U-Prove or Idemix systems use selective disclosure and share as little data as needed to verify legitimacy.

T5- Denial of Service: The denial-of-service attacks that hinder legitimate users can be done with supports of scalable, high-load-capable architectures, with lightweight cryptographic proofs, especially zero-knowledge succinct non-interactive arguments of knowledge (ZK-SNARKs), which in turn protect the performance of the systems.

T6- Elevation of Privilege: The security of privileged resources requires combining the role-based access controls (RBAC) and anonymous credentials that limit actions of anonymous users based on verified attributes without making any identity disclosures.

3.1.2 Functional Requirements

F1- Anonymous Authentication: The anonymity authentication is described as the process that allows users to authenticate themselves without giving their identity-related information. It works with Zero-Knowledge Proofs (ZKPs), which allows showing that a holder owns a credential without expressing any personal data.

F2- Unlinkable Sessions: Unlinkable sessions check that every login be cryptographically unrelated to other logins so that behavior of the user over many logins cannot be assigned to a single identity. This is fulfilled by using individually assigned single-use pseudonyms to each session.

F3- Selective Revocation: Selective revocation implies the ability to pull credentials or pseudonyms in order that nobody is revoked in such a way that privacy of other users is lost. A combination of nullifiers and on-chain and off-chain revocation registries is an efficient way to do this.

F4- Scalability: Scalability is defined as the capability of the system to manage so many log in sessions at the same time; this means that the system can manage a user with a large application load. ZK-SNARKs enables efficient verification, minimising computational costs, thus enables high scalability.

F5- Minimal Client Overhead: Lastly, we require low client overhead so that the system is efficient for low-end devices, such as mobile platforms. Using the tool WebAssembly (WASM), HALP also allows for efficient proofs to be checked in web browsers.

3.1.3 Security Requirements

S1- Integrity and Authentication: There are two concerns at large during the design of a secure decentralised data storage system. First, it is necessary to put measures to ensure integrity and authentication, namely, to ensure that uncontrolled changes cannot be made and that the access of legitimate users with legitimately authenticated access can only be made. As described in the literature, the fulfilment of these requirements is promoted through cryptographic tools and measures like zero-knowledge Succinct Non-Interactive Argentum (ZK-SNARKs) and blind signatures.

S2- Anonymity and Privacy: The architecture should maintain anonymity and privacy at the same time, so that nobody would know the identities of individual users and the activities performed throughout different sessions could not be correlated. A solution to this problem is created through the use of session-specific pseudonymous and nullifiers.

S3- Revocation and Accountability: To accomplish these two goals, the system should also deal with the problem of revocation and accountability. Design should strike a balance between the necessity to cancel the compromised credentials and necessity to ensure the privacy of the user, such that the system should not leave the actors with malicious interest too much exposed to them, and legitimate users should not be affected in any way as well.

3.2 Societal Impact

With the introduction of the Hybrid Anonymous Login Protocol (HALP), the privacy and user anonymity on online systems will be greatly improved, especially in the areas where privacy is extra crucial, namely, in the healthcare and voting. Offers privacy-friendly interaction without the information leak by providing unlinkability and selective revocation, trust relationships among users can also be built. This has a chance of prompting the increased use of privacy protection technologies, which can help communities who need privacy to feel safe and empowered.

3.3 Environmental Impact

The HALP system environmental impact is minimal since the system operates more on cryptographic protocols and decentralized technologies that do not use much energy. Nevertheless, scalability and effective utilization of resources makes it possible to decrease the computational load of networks, so the undesirable impact on energy consumption is low when put into use at large scale.

3.4 Ethical Issues

HALP infers the moral issues regarding the conflict between privacy and accountability. Although it increases user privacy, it can also be misused by malignant users. The selective revocation option has to be implemented with caution so as to cause fairness and non-oppression. Also, the use of decentralized systems must imply the transparency, because users do not always know the risks. To make sure that HALP is consistent with the right to privacy and is ethical, clear protections must be installed.

3.5 Project Management Plan

The management plan of the development and implementation of HALP consists of systematic phases of research, design, testing coordination and deployment. The major achievements should be mentioned: the design of cryptographic protocols, the creation of a working prototype, the test of scalability, and deployment. The plan gives the distribution of resources, schedule and the budget part to make all the goals completed effectively and within the stipulated time.

3.6 Risk Management

Risk management in creating the development of HALP entail coming up with possible threats including system vulnerabilities, violation of privacy and implementation matters. Examples of mitigation strategies are the use of strong cryptographical solutions such as Zero-Knowledge Proofs (ZKPs) to secure the verification process with a selective revocation mechanism as part of their inputs to avoid abuse. Test and update will be done periodically to cover emerging risks and come up with a system that is secured and reliable.

3.7 Economic Analysis

The economic examination of HALP assesses the expenses and the advantages of its usage within the privacy-sensitive sectors. Although the development might be expensive in terms of cost, owing to the incorporation of the complex cryptographic protocols, the benefits of long-term savings and income-generation potential of the industry, such as decentralized finance (DeFi), healthcare, and e-voting make it worthwhile. The cost savings associated with the use of HALP are enormous as it will result in the decreased requirements of centralized authentication systems thereby cutting down both cost of operation as well as cost of maintaining such systems by service providers.

Chapter 4

System Design and Protocol Flow

The chapter System Design and Protocol Flow provides the architecture of HALP and logic of operations. HALP is an authentication platform that is privacy-preserving and is based on advanced cryptographic primitives. It uses ZK-SNARK proofs, BBS + signatures, and Poseidon hashing to provide secure, anonymous and unlinkable user authentication. The system has a structure in the form of modular services, such as wallet, issuer, verifier and registry, with concise responsibilities and interfaces. The interaction of these components to produce verifiable credentials and generate proofs of zero-knowledge as well as revoke credentials through an indexed Merkle tree is described in this chapter. Through tracking the end-to-end protocol flow, i.e. credential issuance, authentication up to revocation, it demonstrates how HALP achieves its objectives, namely privacy, scalability, minimal client overhead, strong security, and decentralized identity requirements.

4.1 Research Methodology

An overview of the process of conducting research is provided below, and it has been visualized it in Figure 4.1.

Recognition of the problem: Find problematic areas in existing privacy preserving authentication schemes, including scalability, unlinkability and revocation, particularly in schemes like Idemix and ZKLogin

Literature Review: Identify the existing systems with the desired privacy preserving properties such as anonymity, selective disclosure, unlinkability and revocation, and use of tools such as Zero-Knowledge Proofs (ZKPs), blind signatures, and revocation registries. Identify the strengths, weaknesses and gaps in the existing studies.

Research Objectives: The main objectives will be to design the HALP, obtain unlinkability and revocation, optimize the performance of the mechanism, security analysis, and designing a prototype, interoperability with currently existing standards, such as W3C Verifiable Credentials.

Threat Modeling: Determine the possible security issues like Sybil attacks, forgery of credentials, and linking of the sessions. A detailed threat modelling has been provided in chapter 3.

Requirement Analysis: Determine functional and non-functional requirements such as secure credential issuance, unlinkable authentication, scalability, low latency, and decentralized trust. A detailed requirement analysis has been provided in chapter 3.

System Design: Fix the architectural design of HALP and combine ZKPs, pseudonyms, and nullifiers. Make it scalable, modular, and configurable towards on-chain and off-chain revocation. Revocation and Privacy Mechanisms: Introduce unlinkability via pseudonyms and selective revocation by cryptographic nullifiers, and ensure this is privacy without compromising security.

Prototype Development: Build an open-source credential issuance, ZKP-based log-in, and revocation management prototype in Rust/TypeScript. Compare with existing systems using benchmarks.

Performance Optimization: Compare the latency results and throughputs of HALP to those of current systems, especially about real-time authentication using ZK-SNARKs.

Security and Privacy Study: Carry out a formal security analysis stating how the HALP fulfills the properties of unlinkability and selective revocation, along with cryptographic proofs.

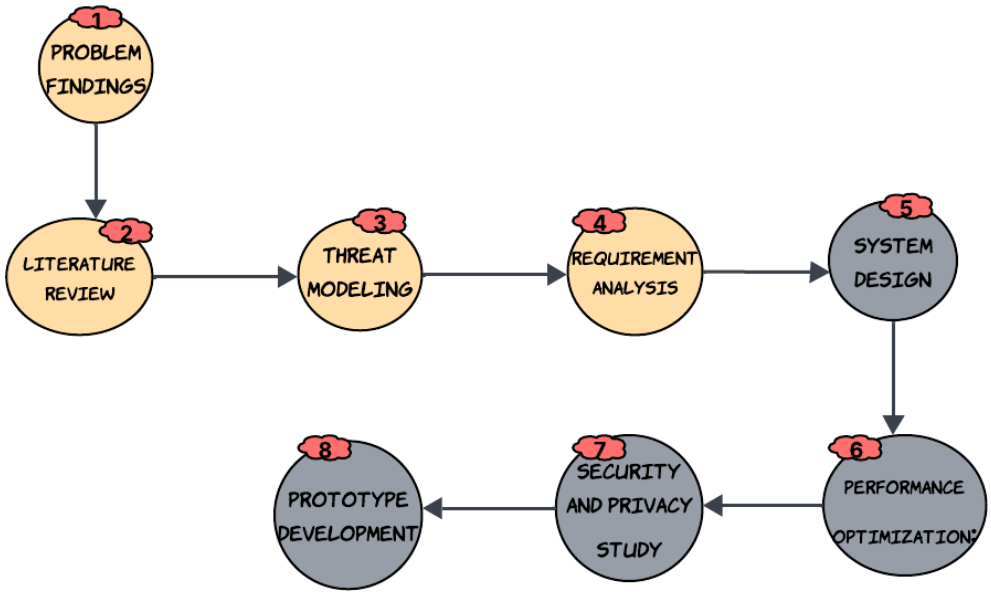


Figure 4.1: Research Methodology

4.2 Mathematical Foundations

We establish the mathematical notation and cryptographic primitives underlying HALP.

Table 4.1: HALP Notation Table

Symbol	Definition
λ	Security parameter (e.g., 128 bits)
$\mathbb{F}_r$	Scalar field of prime order r for BLS12-381
$\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$	Bilinear groups of order r
G_1, G_2	Generators of $\mathbb{G}_1$ and $\mathbb{G}_2$
$e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$	Bilinear pairing function
$ms \in \mathbb{F}_r$	Master secret (holder's private anchor)
$P \in \mathbb{F}_r$	Session-specific pseudonym
$N_f \in \mathbb{F}_r$	Session-specific nullifier
$\mathcal{R}$	Nullifier registry (indexed Merkle tree)
$\text{root}_{\mathcal{R}}$	Current registry Merkle root
(A, e, s)	BBS+ signature tuple: $A \in \mathbb{G}_1$, $e, s \in \mathbb{F}_r$
π_{ZK}	Zero-knowledge SNARK proof
w	Zero-knowledge SNARK witness
π_C	Zero-knowledge proof of commitment opening
π_{blind}	Zero-knowledge proof of blind signature
π_M	Merkle non-membership proof
H	Cryptographic hash function (Poseidon)
$\text{Com}(\cdot;\cdot)$	Pedersen commitment function
domain	Verifier domain identifier
ch	Authentication challenge (256-bit nonce)
n	Session nonce
r	Blinding factor for commitments
t	timestamp
$\mathcal{H}$	Holder
$\mathcal{I}$	Issuer
$\mathcal{V}$	Verifier
$\mathcal{X}$	challenge hash
$pk_{\mathcal{I}}, sk_{\mathcal{I}}$	Issuer's BBS+ public/private key pair
attrs	Vector of credential attributes
credID	Unique credential identifier

4.2.1 Definitions

Definition 9 (**BLS12-381 Pairing Groups**)**.** Let p is 381 bits base field prime and r is 255 bits large prime order of the prime-order subgroups $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$. $r \mid (p^{12}-1)$ and $r \nmid (p^k-1)$ for $1 \leq k < 12$ [23]. HALP operates over bilinear groups $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$ with embedding degree $k = 12$ where:

- $\mathbb{G}_1 \subset E(\mathbb{F}_p)$ of prime order r where $E : y^2 = x^3 + 4$
- $\mathbb{G}_2 \subset E'(\mathbb{F}_{p^2})$ of prime order r where $E' : y^2 = x^3 + 4(1 + i)$

- $\mathbb{G}_T \subset \mathbb{F}_{p^{12}}^*$ of prime order r
- Bilinear pairing $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$

satisfying the bilinearity property: $e([a]P, [b]Q) = e(P, Q)^{ab}$ for all $P \in \mathbb{G}_1, Q \in \mathbb{G}_2, a, b \in \mathbb{F}_r$, and non-degeneracy: $e(G_1, G_2) \neq 1_{\mathbb{G}_T}$ for generators G_1, G_2 [29]. The embedding degree $k = 12$ provides approximately 128 bits of security [23] under the discrete logarithm assumption in both $\mathbb{G}_1$ and $\mathbb{G}_2$, while maintaining efficient pairing computation [21].

Definition 10 (**BBS+ Signature Scheme**)**.** A BBS+ signature scheme $\Pi_{BBS+} = (\text{KeyGen}, \text{Sign}, \text{Verify})$ is a digital signature scheme supporting multi-message signing and selective disclosure [50]:

- $\text{KeyGen}(1^\lambda, L) \to (sk, pk)$: On input security parameter λ and maximum message count L:
  - Choose $sk \leftarrow \mathbb{F}_r^*$ uniformly at random
  - Generate message generators $H_0, H_1, \ldots, H_{L+1} \in \mathbb{G}_1$ using hash-to-curve
  - Compute $pk = [sk]G_2$
  - Return (sk, pk) and public parameters $(H_0, \ldots, H_{L+1})$
- $\text{Sign}(sk, (m_1, \ldots, m_L)) \to (A, e, s)$: Sign message vector $(m_1, \ldots, m_L) \in \mathbb{F}_r^L$:
  - Choose $e, s \leftarrow \mathbb{F}_r^*$ uniformly at random
  - Compute $B = H_0 + \sum_{i=1}^{L} [m_i]H_i + [s]H_{L+1}$
  - Compute $A = [1/(sk + e)]B$
  - Return signature $(A, e, s) \in \mathbb{G}_1 \times \mathbb{F}_r^2$
- $\text{Verify}(pk, (m_1, \ldots, m_L), (A, e, s)) \to \{0, 1\}$: Verify signature validity:
  - Check $A \neq \mathcal{O}$ (identity element of $\mathbb{G}_1$)
  - Compute $B = H_0 + \sum_{i=1}^{L} [m_i]H_i + [s]H_{L+1}$
  - Return 1 iff $e(A, pk + [e]G_2) = e(B, G_2)$

The scheme provides selective disclosure through zero-knowledge proofs of knowledge and unlinkable presentations, where each proof is computationally indistinguishable from random even when generated from the same signature [50].

Definition 11 (Pedersen Commitment Scheme)**.** Let $\mathbb{G}$ be a cyclic group of prime order q with generators $g, h_0, h_1, \ldots, h_k \in \mathbb{G}$ where the discrete logarithm relationships between generators are unknown. The Pedersen commitment scheme $\Pi_{Ped} = (\text{Setup}, \text{Commit}, \text{Open})$ is defined as [11]:

- $\text{Setup}(1^\lambda, k) \to pp$: Generate public parameters including group $\mathbb{G}$ and generators
- $\text{Commit}(pp, (m_0, \ldots, m_k)) \to (C, r)$: For messages $m_0, \ldots, m_k \in \mathbb{F}_q$:.
  - Choose blinding factor $r \leftarrow \mathbb{F}_q$ uniformly at random

  - Compute $C = [r]g + [m_0]h_0 + \sum_{i=1}^{k}[m_i]h_i$
  - Return commitment C and opening information r

- $\text{Open}(pp, C, (m_0, \ldots, m_k), r) \to \{0, 1\}$: Verify $C = [r]g + [m_0]h_0 + \sum_{i=1}^{k}[m_i]h_i$

The scheme satisfies:

- **Perfect Hiding**: For any two message vectors $(m_0, \ldots, m_k)$ and $(m'_0, \ldots, m'_k)$, the commitments C and C' are statistically indistinguishable [3].
- **Computational Binding**: Under the discrete logarithm assumption, no PPT adversary can find $(m_0, \ldots, m_k), r$ and $(m'_0, \ldots, m'_k), r'$ such that both open the same commitment C with non-negligible probability [9].

Definition 12 (**Poseidon Hash Function**)**.** Let $\text{Poseidon} : \mathbb{F}_p^t \to \mathbb{F}_p$ be a cryptographic hash function family optimized for algebraic constructions over large prime fields, particularly zero-knowledge proof systems. Poseidon employs a substitution-permutation network with:

- S-box function $S(x) = x^\alpha$ where α is the smallest integer such that $\gcd(\alpha, p - 1) = 1$
- MDS (Maximum Distance Separable) matrix for optimal diffusion
- Round constants derived from secure pseudo-random generation
- Variable input capacity t field elements

For HALP, we instantiate Poseidon over $\mathbb{F}_r$ for BLS12-381 with capacity $t = 3$ for:

- Pseudonym derivation: $P = \text{Poseidon}(ms, n, H(\text{domain}))$
- Nullifier derivation: $N_f = \text{Poseidon}(\text{credID}, n, H(\text{domain}))$

Poseidon provides collision resistance, preimage resistance, and second preimage resistance while maintaining low multiplicative complexity in arithmetic circuits compared to SHA-256 or other hash functions [30].

Definition 13 (Indexed Merkle Tree Registry)**.** An indexed Merkle tree $\mathcal{R}$ for nullifier management is a data structure supporting efficient insertions and non-membership proofs, consisting of [26]:

- **Ordered Leaf Set**: $\mathcal{N} = \{N_f^{(1)}, N_f^{(2)}, \ldots, N_f^{(k)}\}$ where $N_f^{(i)} < N_f^{(i+1)}$ for all i
- **Tree Structure**: Complete binary tree of depth d with 2^d leaves
- **Linking Structure**: Each leaf $N_f^{(i)}$ contains pointer to next leaf $N_f^{(i+1)}$
- **Root Commitment**: $\text{root}_\mathcal{R} = \text{MerkleRoot}(\mathcal{N})$ binding the entire set

Operations include:

- $\text{Insert}(N_f, \text{metadata}) \to \text{root}'_\mathcal{R}$: Insert nullifier maintaining order
- $\text{NonMembershipProof}(N_f, \text{root}_\mathcal{R}) \to \pi_M$: Generate proof that $N_f \notin \mathcal{N}$

- VerifyNonMembership$(\pi_M, N_f, \text{root}_{\mathcal{R}}) \to \{0,1\}$: Verify non-membership proof

The indexed structure enables $O(\log |\mathcal{N}|)$ insertion time compared to $O(|\mathbb{F}_r|)$ for sparse Merkle trees, crucial for HALP's scalability requirements.

Definition 14 (**Zero-Knowledge SNARK**)**.** A zero-knowledge Succinct Non-interactive Argument of Knowledge (ZK-SNARK) for relation $\mathcal{R} \subseteq \mathcal{X} \times \mathcal{W}$ consists of algorithms $(\text{Setup}, \text{Prove}, \text{Verify})$ where:

- $\text{Setup}(1^\lambda, \mathcal{R}) \to (pk, vk)$: Generate proving key pk and verification key vk
- $\text{Prove}(pk, x, w) \to \pi$: Generate proof π for statement x with witness w where $(x, w) \in \mathcal{R}$
- $\text{Verify}(vk, x, \pi) \to \{0,1\}$: Verify proof π for public statement x

The system satisfies:

- **Completeness**: Honest proofs always verify
- **Knowledge Soundness**: Extractability of witness from accepting proofs
- **Zero-Knowledge**: Proofs reveal no information beyond statement validity
- **Succinctness**: Proof size and verification time are $O(\log |\mathcal{C}|)$ in circuit size

HALP employs Groth16 or PLONK for the relation $\mathcal{R}_{HALP}$ encoding BBS+ signature knowledge, commitment opening, pseudonym/nullifier derivation, and Merkle non-membership.

4.2.2 Security Properties

Definition 15 (**Computational Anonymity/Unlinkability**)**.** HALP provides computational unlinkability if for any probabilistic polynomial-time (PPT) adversary $\mathcal{A}$, the advantage in the unlinkability game is negligible:

$$\text{Adv}^{\text{Unlink}}_{\text{HALP},\mathcal{A}}(\lambda) = \left| \Pr[\text{UnlinkGame}^{\mathcal{A}}(\lambda) = 1] - \frac{1}{2} \right| \leq \text{negl}(\lambda)$$

The $\text{UnlinkGame}^{\mathcal{A}}(\lambda)$ is defined as follows:

1. **Setup**: Challenger generates system parameters and issuer key pair $(sk_{\mathcal{I}}, pk_{\mathcal{I}})$
2. **Query Phase**: $\mathcal{A}$ adaptively queries:
   - $\mathcal{O}_{\text{Issue}}(\text{attrs})$: Issues credential for attribute vector
   - $\mathcal{O}_{\text{Auth}}(\text{credID}, \text{domain})$: Performs authentication with credential
3. **Challenge Phase**: $\mathcal{A}$ submits two credential-domain pairs (cred_0, d_0) and (cred_1, d_1)
4. **Challenge Response**: Challenger chooses $b \leftarrow \{0,1\}$ and returns authentication proof for (cred_b, d_b)

5. **Guess Phase**: $\mathcal{A}$ outputs guess $b' \in \{0, 1\}$
6. **Output**: Return 1 if $b' = b$, else 0

This captures that authentication proofs from the same credential across different domains (or different credentials) are computationally indistinguishable.

Definition 16 (**Unforgeability under q-SDH Assumption**)**.** HALP satisfies existential unforgeability under adaptive chosen-message attacks if for any PPT adversary $\mathcal{A}$ making at most q credential issuance queries:

$$\mathrm{Adv}^{\mathrm{Forge}}_{\mathrm{HALP},\mathcal{A}}(\lambda) = \Pr[\mathrm{ForgeGame}^{\mathcal{A}}(\lambda) = 1] \leq \mathrm{negl}(\lambda)$$

The $\mathrm{ForgeGame}^{\mathcal{A}}(\lambda)$ proceeds as:

1. **Setup**: Generate system parameters and issuer keys $(sk_{\mathcal{I}}, pk_{\mathcal{I}})$
2. **Query Phase**: $\mathcal{A}$ makes up to q queries to $\mathcal{O}_{\mathrm{Issue}}(\mathrm{attrs}_i)$ for $i = 1, \ldots, q$
3. **Forgery Attempt**: $\mathcal{A}$ outputs authentication proof $(\pi_{ZK}, P, N_f, \mathrm{challenge})$
4. **Verification**: Check that:
   - The proof verifies: $\mathrm{Verify}(\pi_{ZK}, \mathrm{public\ inputs}) = 1$
   - The proof corresponds to attributes not queried in any $\mathcal{O}_{\mathrm{Issue}}$ call
5. **Output**: Return 1 if verification succeeds, else 0

Security relies on the q-Strong Diffie-Hellman assumption: given $(G_1, G_2, [x]G_2, [x^2]G_2, \ldots, [x^q]G_2)$, no PPT algorithm can compute $([1/(x+c)]G_1, c)$ for any $c \in \mathbb{F}_r$ with non-negligible probability.

Definition 17 (Perfect Zero-Knowledge)**.** HALP authentication proofs satisfy perfect zero-knowledge if there exists a PPT simulator $\mathcal{S}$ such that for any verifier $\mathcal{V}^*$ (potentially malicious) and all auxiliary inputs z:

$$\left\{\mathrm{View}^{\mathrm{Real}}_{\mathcal{V}^*}(\mathrm{stmt}, \mathrm{wit}, z)\right\} \equiv \{\mathcal{S}(\mathrm{stmt}, z)\}$$

where:

- $\mathrm{View}^{\mathrm{Real}}_{\mathcal{V}^*}(\mathrm{stmt}, \mathrm{wit}, z)$ represents the view of $\mathcal{V}^*$ in real interaction with honest prover $\mathcal{P}$ on statement stmt with witness wit
- $\mathcal{S}(\mathrm{stmt}, z)$ is the output of simulator $\mathcal{S}$ given only statement stmt and auxiliary input z
- $\equiv$ denotes statistical indistinguishability (identical distributions)

The statement stmt includes public inputs $(P, N_f, \mathrm{root}_{\mathcal{R}}, ch, pk_{\mathcal{I}})$ while witness wit contains $(ms, n, A, e, s, \mathrm{attrs}, \pi_M)$. This ensures that authentication proofs reveal no information about the holder's credential, master secret, or any private attributes beyond what is explicitly disclosed.

Definition 18 (**Computational Soundness**)**.** HALP authentication proofs satisfy computational soundness (knowledge soundness) if there exists a PPT knowledge extractor $\mathcal{E}$ such that for any PPT cheating prover $\mathcal{P}^*$:

$$\Pr\left[\begin{array}{c}(\text{stmt}, \pi) \leftarrow \mathcal{P}^*(pp) \\ \text{wit} \leftarrow \mathcal{E}^{\mathcal{P}^*}(pp, \text{stmt}) \\ \text{Verify}(\text{stmt}, \pi) = 1 \wedge (\text{stmt}, \text{wit}) \notin \mathcal{R}_{HALP}\end{array}\right] \leq \text{negl}(\lambda)$$

where $\mathcal{R}_{HALP}$ is the HALP relation encoding:

- Valid BBS+ signature: $e(A, pk_{\mathcal{I}} + [e]G_2) = e(H_0 + \sum[m_i]H_i + [s]H_{L+1}, G_2)$
- Correct pseudonym: $P = \text{Poseidon}(ms, n, H(\text{domain}))$
- Correct nullifier: $N_f = \text{Poseidon}(\text{credID}, n, H(\text{domain}))$
- Valid non-membership proof: $\text{VerifyNonMembership}(\pi_M, N_f, \text{root}_{\mathcal{R}}) = 1$

This ensures that only holders with legitimate credentials can generate accepting proofs.

Definition 19 (Perfect Session Unlinkability)**.** For any two authentication sessions using the same credential but different domains $d_i \neq d_j$ or different session nonces $n_i \neq n_j$, the pseudonyms are information-theoretically unlinkable:

$$\Pr[\text{Link}(P_i, P_j) = 1] = \text{negl}(\lambda)$$

where Link is any efficient algorithm attempting to determine if pseudonyms $P_i = \text{Poseidon}(ms, n_i, H(d_i))$ and $P_j = \text{Poseidon}(ms, n_j, H(d_j))$ are derived from the same master secret ms.
This property holds information-theoretically due to the cryptographic properties of Poseidon hash function and the requirement that session nonces are chosen uniformly at random for each authentication.

Definition 20 (Replay Resistance)**.** HALP provides replay resistance if no PPT adversary $\mathcal{A}$ can successfully reuse a valid authentication proof:s

$$\Pr[\text{ReplayGame}^{\mathcal{A}}(\lambda) = 1] \leq \text{negl}(\lambda)$$

where $\text{ReplayGame}^{\mathcal{A}}(\lambda)$ allows $\mathcal{A}$ to:

1. Observe valid authentication sessions $(P_i, N_{f,i}, \pi_i)$
2. Attempt to authenticate using any observed nullifier $N_{f,i}$
3. Win if authentication succeeds with a previously used nullifier

Protection is guaranteed by the nullifier registry $\mathcal{R}$ which maintains cryptographic commitments to all used nullifiers and rejects any authentication attempt with N_f where $N_f \in \mathcal{R}$.

4.3 HALP System Architecture

HALP consists of four main components (Figure 4.2) interacting through cryptographic protocols, following design principles from anonymous credential systems [6], [26], [50].

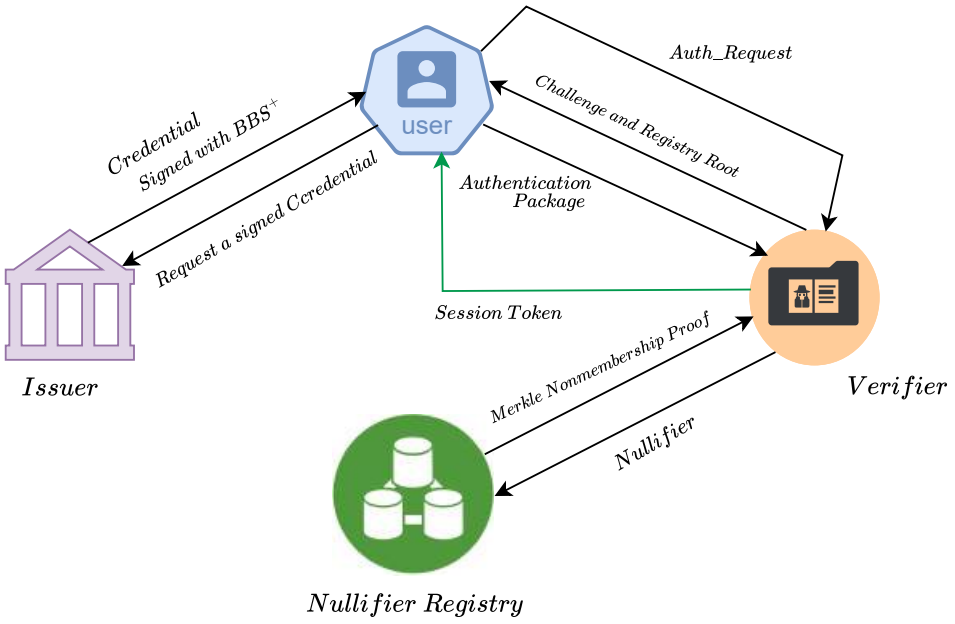


Figure 4.2: HALP Architecture

4.3.1 Credential Issuer $\mathcal{I}$

Credential Issuer is the universal trust entry-point in the HALP protocol. It signs blind BBS+ with the help of committed attributes [50]. Being a respected authority it authenticates the eligibility of holders and cryptographically attests but maintains holder privacy by use of blind signature protocols.

The first stage workflow used by the issuer involves the creation and administration of cryptography key material. It generates a BBS+ key pair $(sk_{\mathcal{I}}, pk_{\mathcal{I}})$ of the elliptic curve BLS12-381, providing the security requirements of pairing-based cryptography [23]. The issuer also publishes its public key pk_I in a DID Document, as specified in the DID Core specification [32] in order to be verifiable by the public and meet its standards. This enables the verifiers to cryptographically establish credential authenticity by standardized resolution mechanisms.

The issuer takes additional basic verification measures before signing a credential. It verifies the holder making the request by first using protocols to ensure there is a verification of identity. It then implements zero-commitment regulations, which are stipulated by its holder. These protocols ensure that the holder is able to know the attributes performed and possesses the secrets of the definite attribute values. After success in verification, the issuer executes the blind signing protocol on Pedersen commitments [2], generating valid BBS+ signatures without being aware of the values of the particular attribute undergoing signing.

In order to maintain system integrity and compliance, the issuer keeps detailed cryptographically secure audit records of every issuance. These logs are useful in

assisting transparency in issuance decisions and privacy of holders in the blind signature model.

The HALP protocol imposes three important security properties on the Credential Issuer. To begin with, the blindness property guarantees that the issuer obtains no information about the attributes committed, and this is ensuring privacy of holders against malicious issuers. Second, the unforgeability property ensures that the adversary is not able to generate valid signatures when the issuer does not have the secret key, and a security guarantee cannot be reduced to the q-Strong Diffie-Hellman (q-SDH) assumption in the random oracle model [8]. Third, unlinkability of a session makes the issuer unable to associate certain issuance sessions with subsequential authentication events, due to the infertility of the deterministic nature of the blind signature construction that removes dependencies between issuance transcripts and authentication artifacts.

4.3.2 Credential Holder ($\mathcal{H}$)

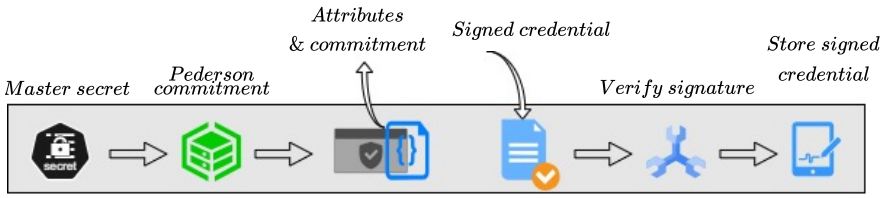


Figure 4.3: Sequence of Holder's action to get the signed credential

The Credential Holder represents the privacy-seeking entity within the HALP protocol, functioning as the primary user who manages credentials and generates anonymous authentication proofs following principles established in anonymous credential systems like Idemix [5]. The holder operates as an autonomous agent responsible for maintaining cryptographic material while preserving anonymity across multiple authentication sessions and verifier domains.

The holder's operational lifecycle begins with the secure generation and storage of a cryptographic master secret ms, which serves as the foundational identity anchor throughout the system. This master secret, randomly sampled from the scalar field $\mathbb{F}_r$ of the BLS12-381 curve, must be protected with the highest security measures as it represents the holder's core cryptographic identity. The holder utilizes this master secret alongside other attributes to create Pedersen commitments, which provide computational hiding and binding properties essential for the blind signature protocol during credential issuance (Figure 4.3).

Following successful credential issuance, the holder performs critical verification operations to ensure the integrity of received credentials. This process involves unblinding the BBS+ signature received from the issuer (Figure 4.3) and conducting comprehensive signature verification against the issuer's public key. The holder must validate that the signature correctly corresponds to their committed attributes while

confirming the cryptographic binding between the signature and their master secret.

During authentication phases, the holder demonstrates sophisticated cryptographic capabilities through the derivation of session-specific identifiers. Using the cryptographically secure Poseidon hash function [30], the holder generates unique pseudonyms and nullifiers for each authentication session. These derivations follow deterministic but unpredictable patterns that enable consistent identification within sessions while preventing cross-session linkability. The pseudonym P serves as a session-specific identifier, while the nullifier Nf provides replay protection through registry-based double-spending prevention.

The core technical contribution of the holder involves constructing zero-knowledge SNARK proofs for authentication using efficient proof systems such as Groth16 [19]. These proofs demonstrate possession of valid credentials and knowledge of the master secret without revealing any sensitive information to verifiers. The holder must carefully balance proof efficiency with privacy guarantees, ensuring that proof generation remains computationally feasible while maintaining zero-knowledge properties.

Throughout the credential lifecycle, the holder maintains active responsibility for managing revocation state and credential validity. This involves monitoring nullifier registries to ensure authentication attempts comply with anti-replay mechanisms and tracking credential expiration or revocation status across multiple verifier domains.

The HALP protocol guarantees three critical security properties for credential holders that distinguish it from traditional authentication systems. The confidentiality property ensures that the master secret ms never leaves the holder's secure environment, even during credential issuance and authentication procedures, providing strong protection against credential theft or compromise. The unlinkability property represents a fundamental privacy guarantee, ensuring that pseudonyms generated across different authentication sessions and verifier domains remain cryptographically unlinkable, even to coalition attacks involving multiple verifiers. Finally, the replay resistance property provides robust protection against authentication replay attacks through the nullifier mechanism, which prevents holders from reusing previously consumed nullifiers for fraudulent authentication attempts, thereby maintaining the integrity of one-time authentication protocols.

4.3.3 Credential Verifier ($\mathcal{V}$)

The part of the HALP protocol referred to as the service provider is the Credential Verifier. It authenticates the authentication proofs which are anonymous, and allocates authorized session tokens to the holders who have established themselves. The verifier provides the linkage between anonymous credential holder users and the secure services or resource they require. Its design ensures high security checks with holder identities and sensitive attributes being totally anonymous.

The cryptographically secure challenges are the foundations of the authentication process by the verifier. These problems are used to prove that the credential is pos-

sessed and also as freshness tokens that are used to prevent replay attacks. These challenges are normally 256-bit cryptography nonces that bind each authentication request in the creation of the evidence. This makes sure that the authentication artifacts are not used during subsequent requests and sessions.

The verifier will frequently update the global nullifier registry to prevent spending the same money twice. It retrieves the current Merkle tree root(rootR). This root is a cryptographic commitment to all the earlier used nullifiers in all domains of verifiers. It allows the verifier to verify that the nullifier is fresh, and it is not necessary to download the complete registry.

The center verification operation runs advanced cryptographic verification of SNARK proofs (zero-knowledge) by a holder. Efficient algorithms, like Groth16, allow the verifier to verify that the proofs are correct without revealing any of the secret keys: the authentication problem, the registry root, and the domain identifier. This verification most often takes only a matter of milliseconds, which is an ideal match with application of high throughput web applications and cryptographic rigor[52].

One of the security mechanisms is the nullifier verification. The verifier checks the registry in order to be sure that each nullifier is not used. The protocol checks Merkle non-membership proof along with the present registry state, which provides cryptography assurance that an authentication request is driven by some fresh non-replayed credential. As a result, in a case where a nullifier has already been generated, the verifier denies the authentication process and integrity of the system is maintained.

Once the verifier successfully does validation, JWT session tokens are issued by the verifier (Figure 4.3). These are cryptographically bound to a session specific pseudonym (P) that is produced by the holder. With tokens, privacy is not compromised because the authorization data is stored in the tokens, while the standard web-application session management can be used. The verifier stores session state and implements policies of access controls depending on the authenticated pseudonym and any characteristics revealed.

The HALP protocol has three fundamental privacy and security properties of anonymous authentication systems that credential verifiers must fulfill. To start with, the privacy property means that even after a sophisticated statistical analysis or engaging in corroboration with other verifiers, a verifier cannot gather any information regarding the real identity of the holder or any other undisclosed property. This property applies to prevent not only unlinkability but also it prevents breach of attribute privacy and enables those holding it to disclose only that needed to fulfill a specific authentication setting.

Second, unlinkability property does not allow the verifiers to correlate cross-domain or cross-time authentication sessions. The reason why pseudonyms prevent cross-domain tracking and cross-domain profiling is that they are domain independent, and used at a per-session basis.

And, lastly, the anti-replay property provides to the verifier the integrity required to

reject any authentication attempt where the nullifier is reused. The unknown nebular plates have a cryptographically verifiable registered issues of nullifer, thus making the verifier activate genuine one-time authentication, thus deterring credential-reuse attacks.

4.3.4 Nullifier Registry ($\mathcal{R}$)

Nullifier Registry is an important component of the HALP protocol. It keeps a cryptographically secure, tamper evident register of all the nullifiers that have already been ingested within the whole authentication ecosystem. This element uses indexed Merkle tree, along the lines of systems like Aztec indexed Merkle tree, to enable fast operations of the insertion of the elements and the generation of cryptographic proofs to validate nullifier freshness[26].

Its major innovation is indexed Merkle tree implementation, which addresses the complexity issues of computation of traditional sparse Merkle tree constructions. Traditional methods need O (Fr) space to store the entire nullifier space. Conversely, the indexed design has a complexity of O(log N) when it comes to the complexity of the insertion operation, where N is the number of consumed nullifiers actually. This optimization allows practical deployment even in the high throughput authentication setting and does not violate the cryptographic guarantees that zero-knowledge proof systems are based on.

The workflow of the registry is on giving effective non-membership proofs of fresh nullifiers on authentication verification (Figure 4.6). A verifier that obtains an authentication request with a new nullifier will consult the registry to obtain cryptographic evidence that the nullifier has not been recorded. The registry yields a non-membership certificate depending on the hash with Merkel, thus the verifier can verify the cryptographic freshness of the nullifier without the full information about the registry, and hence, privacy and integrity is not broken.

Upon authentication, the registry logs the nullifiers that have been used against metadata that includes timestamps and domain identifiers. This metadata will enable domain specific high-level management functions e.g. analytics and time based security monitoring. The videos are cryptographically attached to the registry by recomputing the Merkle tree and thus, any unauthorized change is obvious by checking the root hash.

The registry maintains its entire lifecycle by automatically collecting garbage of lapsed nullifiers. This is automatically done to eliminate the nullifiers which are no longer valid which will limit the growth of the registry indefinitely and the security during the active authentication periods. Real time verification performance is not affected by garbage collection; authentication flows are not interrupted.

Since global registries of nullifiers may have a problem with scalability, more sophisticated sharding protocols are presented in the HALP protocol. These protocols spread the registry functions and activities within several domains without jeopardising the security. Domain-specific sharding is enabled to provide horizontal scale

to the registry infrastructure and read-only registry instances. Privacy properties are further highly boosted when nullifiers are shared across domain, where necessary.

The HALP protocol provides that the Nullifier Registry meets three basic security and privacy properties that are essential to anonymous authentication systems. The tamper-evidence property offers cryptographic guarantees that unauthorized changes to the registry state are computationally infeasible and can be instantly identified by the use of Merkle root verification. It is a property that depends on collision-resistant hash functions, and guarantees that even an adversarial operator of the registry cannot insertively modify nullifier records with impunity. The privacy property ensures that the nullifiers though publicly recorded in the registry do not disclose any information on the identity of holders or any sensitive property. The nullifiers are computed with a secure hash based on master secrets and session specific parameters which therefore cannot be decrypted even by analysing the registry under strong attacks on traffic analysis. Lastly, the non-replay property gives the conclusive guarantees of authentication proofs to be consumed at most once as the registry cryptographically ensures that the non-replay property holds even in the face of a non-replaying nullifier, since the neutrality of a single-time credential consumption semantics is fundamental to secure anonymous authentication.

4.4 Protocol Flow of HALP

The HALP credential issuance protocol enables privacy-preserving credential distribution through a four-phase blind signature mechanism that preserves holder anonymity while ensuring cryptographic integrity (Figure 4.5). This protocol builds upon the BBS+ signature scheme and Pedersen commitment constructions to achieve perfect blindness during the signing process.

From	To	Phase	Information Exchanged
$\mathcal{H}$	$\mathcal{I}$	Issuance	DID auth, Pedersen commitment, ZK proof
$\mathcal{I}$	$\mathcal{H}$	Issuance	Blind BBS+ signature, public key
$\mathcal{H}$	$\mathcal{V}$	Auth Start	Authentication request
$\mathcal{V}$	$\mathcal{H}$	Auth Start	Challenge ch, registry root
$\mathcal{V}$	$\mathcal{R}$	Auth	Registry root query
$\mathcal{H}$	$\mathcal{R}$	Auth	Non-membership proof request
$\mathcal{R}$	$\mathcal{H}$	Auth	Merkle non-membership proof
$\mathcal{H}$	$\mathcal{V}$	Auth Complete	ZK-SNARK proof, nullifier, pseudonym
$\mathcal{V}$	$\mathcal{R}$	Auth Complete	Nullifier recording
$\mathcal{V}$	$\mathcal{H}$	Auth Complete	JWT session token

Table 4.2: HALP Component Interaction Matrix

4.4.1 Credential Issuence

Phase 1: Commitment Generation
The credential issuance process begins with the holder $\mathcal{H}$ establishing a cryptographic commitment to their private identity (Figure 4.3). The holder first generates

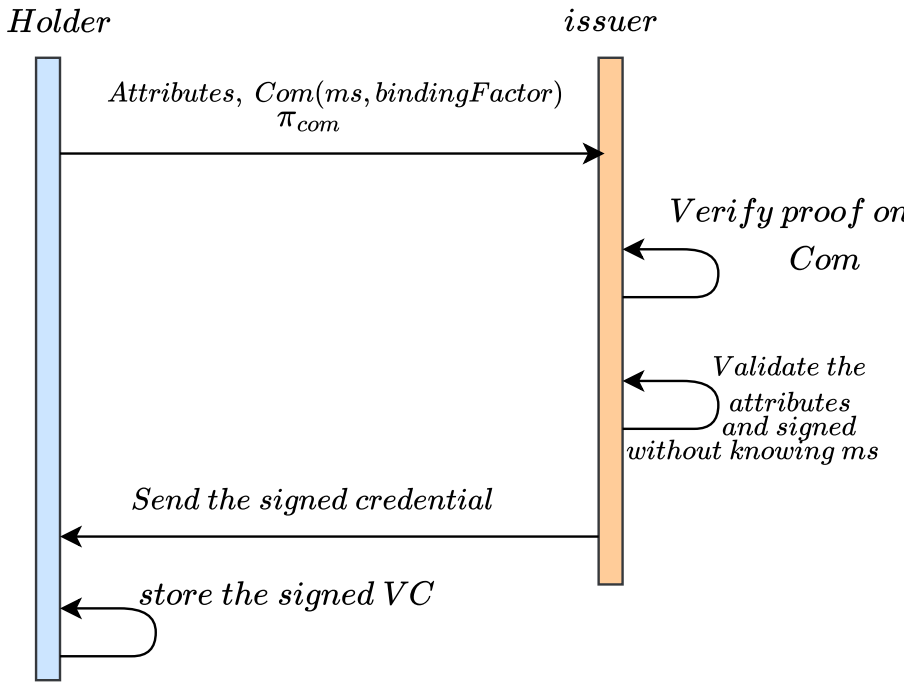


Figure 4.4: HALP Issuence Flow

a cryptographically secure master secret ms by sampling uniformly from the scalar field $\mathbb{F}_r$ of the BLS12-381 elliptic curve. This master secret serves as the fundamental identity anchor that enables pseudonym derivation and nullifier generation in subsequent authentication protocols (Figure 4.5).

Following master secret generation, the holder selects a random blinding factor $r \leftarrow \mathbb{F}_r$ to ensure the commitment remains computationally hiding. The holder then constructs a Poseidon-based commitment $C = \text{Poseidon}(ms, r)$ that cryptographically binds the master secret to a unique identifier while maintaining circuit compatibility for subsequent ZK-SNARK proof generation. Notably, credential attributes are *not* included in this commitment; instead, they are bound to the holder's identity through BBS+ signature binding in Phase 2, enabling selective disclosure capabilities.

For anonymous credential requests, the holder generates a zero-knowledge proof π_C demonstrating knowledge of the commitment opening without revealing the master secret. This proof follows the Schnorr-based Σ-protocol construction:

$$\pi_C = \text{PoK}\{(ms, r) : C = \text{Poseidon}(ms, r)\} \tag{4.1}$$

The proof proceeds as follows: the holder samples random $k_{ms}, k_r \leftarrow \mathbb{F}_r$, computes the commitment to randomness $T = \text{Poseidon}(k_{ms}, k_r)$, derives the Fiat-Shamir challenge $c = \text{Hash}(T \| C \| \text{nonce})$, and computes responses $s_{ms} = k_{ms} + c \cdot ms$ and $s_r = k_r + c \cdot r$. This provides computational soundness while ensuring the issuer

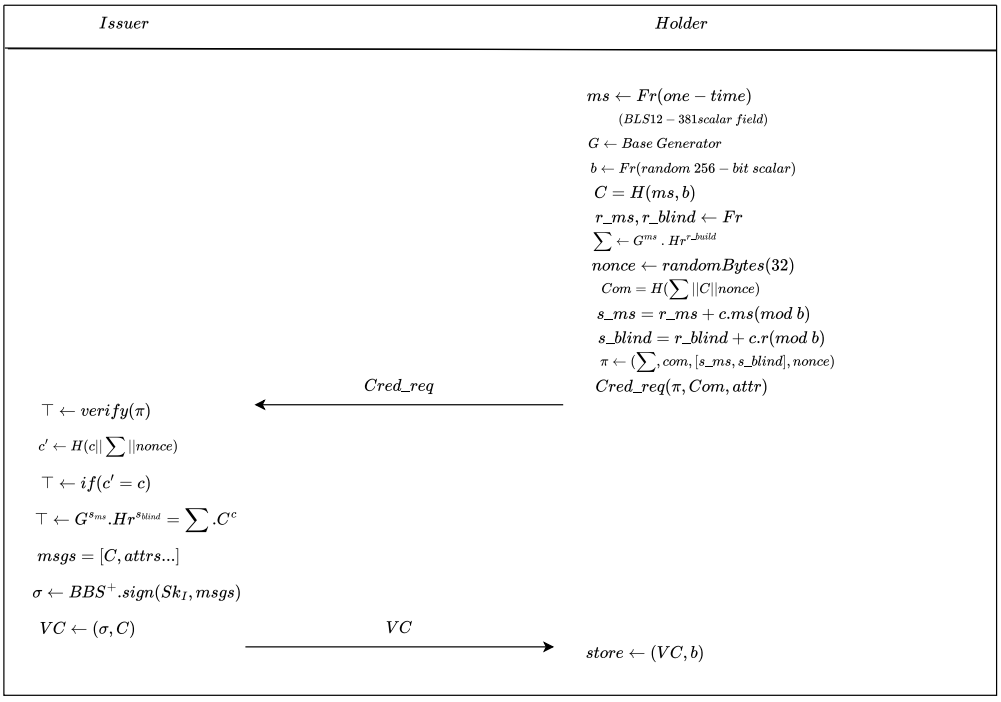


Figure 4.5: Cryptographic Protocol Flow of Issuence

learns nothing about the master secret beyond the validity of the commitment.

Phase 2: Request Submission and Verification
The holder initiates credential issuance by submitting a request to the issuer $\mathcal{I}$ (Table 4.2) through one of two modes: *standard issuance* with holder DID disclosure, or *anonymous issuance* preserving holder privacy (Figure 4.5).
In standard issuance mode, the holder transmits a credential request containing their identity commitment, the credential type, and the plaintext claim attributes $\{attr_1, \ldots, attr_k\}$ required for the credential. This mode is appropriate when the holder's identity must be known to the issuer for eligibility verification. In anonymous issuance mode, the holder constructs a privacy-preserving request containing: (1) a context-specific pseudonym $P = G_{\text{ctx}}^{H(ms\|\text{credentialType})}$ derived from the master secret, ensuring unlinkability across different credential types; (2) the Poseidon commitment C binding the master secret; (3) the zero-knowledge proof π_C demonstrating knowledge of the commitment opening; and (4) the credential claims encrypted under the issuer's public key $\text{Enc}_{pk_{\mathcal{I}}}(\{attr_1, \ldots, attr_k\})$ along with a claims hash $h = \text{SHA256}(\{attr_j\})$ for integrity verification. A freshness nonce prevents replay attacks on the request.

Upon receiving the credential request, the issuer performs the following verification steps (Figure 4.5):

1. **Proof Verification**: For anonymous requests, the issuer verifies the zero-knowledge proof π_C by recomputing the Fiat-Shamir challenge $c' = H(T\|C\|\text{context})$ and checking that $\text{Poseidon}(s_{ms}, s_r) = T \cdot C^c$ holds, confirming the holder pos-

sesses valid openings to the commitment.

2. **Claims Decryption**: The issuer decrypts the encrypted claims(Figure 4.5)using their private key and verifies integrity by checking SHA256(decrypted claims) = h.

3. **Policy Validation**: The issuer validates that decrypted attributes satisfy credential issuance policy requirements, such as minimum age thresholds or organizational membership criteria.

Upon successful verification, the request is queued in a pending state awaiting issuer approval, enabling administrative review before credential signing. This two-phase approach (request submission followed by explicit approval) provides an additional authorization checkpoint for credential issuance governance.

Phase 3: BBS+ Credential Signing with Commitment Binding
Following successful verification and administrative approval (Figure 4.5), the issuer proceeds with BBS+ signature generation over the W3C Verifiable Credential structure [50]. Importantly, the HALP system employs *commitment-bound signatures* rather than blind signatures the issuer has full visibility of credential attributes but cryptographically binds the credential to the holder's master secret, preventing credential transfer.

The issuer first constructs a W3C Verifiable Credential conforming to the VC Data Model v2.0, containing the credential identifier, type, issuer DID, validity period, and credential subject with all claim attributes. Each credential field is then encoded as a separate message m_i to enable selective disclosure during presentation.

For anonymous issuance with commitment binding, the issuer constructs the message vector by *prepending* the holder's Poseidon commitment as the first message:

$$\vec{m} = (C, m_{\text{@context}}, m_{\text{id}}, m_{\text{type}}, m_{\text{issuer}}, m_{\text{validFrom}}, m_{\text{subject.id}}, \{m_{attr_j}\}_{j=1}^{k}) \tag{4.2}$$

where $C = \text{Poseidon}(ms, r)$ occupies index 0, ensuring the signature is cryptographically bound to the holder's master secret. For standard issuance, the commitment is omitted and messages begin with credential metadata.

The issuer generates the BBS+ signature $\sigma = (A, e, s) = \text{Sign}(sk_{\mathcal{I}}, \vec{m})$ using the BLS12-381 G2 private key $sk_{\mathcal{I}}$. The signature computation selects random exponents $e, s \leftarrow \mathbb{F}_r$ and computes:

$$A = \left(g_1 \cdot h_0^s \cdot \prod_{i=0}^{|\vec{m}|-1} h_{i+1}^{m_i} \right)^{1/(sk_{\mathcal{I}}+e)} \tag{4.3}$$

where h_i denotes message-specific generators derived from the BLS12-381 curve.

The issuer returns a complete credential package containing: the signed W3C Verifiable Credential with embedded `BbsBlsSignature2020` proof, the base64-encoded BBS+ signature, the issuer's public key $pk_{\mathcal{I}}$ for verification, and a message labels

array mapping indices to credential fields for selective disclosure reference. For anonymous credentials, metadata flags indicate commitment binding is active, enabling verifiers to enforce master secret proof requirements during authentication.

Security Property: The commitment binding ensures credential non-transferability—only the holder possessing the master secret ms that opens commitment C can generate valid authentication proofs, as the authentication circuit (Phase 4) requires demonstrating $C = \text{Poseidon}(ms, r)$ for the same ms used in pseudonym and nullifier derivation.

Phase 4: Credential Reception and Secure Storage
Upon receiving the signed credential package from the issuer, the holder processes and stores the credential for subsequent authentication use (Figure 4.3). The credential package contains the complete W3C Verifiable Credential with embedded BBS+ proof, the issuer's public key, and message label mappings for selective disclosure reference.

The holder constructs a secure credential storage record containing the following components:

- **Credential Identifier**: The unique credential URN (e.g., `urn:uuid:...`) serving as the primary storage key
- **Signed Credential**: The complete W3C Verifiable Credential object including the `BbsBlsSignature2020` proof
- **BBS+ Signature**: The base64-encoded signature $\sigma = (A, e, s)$ for proof generation
- **Commitment Metadata**: The commitment hash C and blinding factor r required for authentication proofs
- **Issuer Public Key**: The BLS12-381 G2 public key $pk_{\mathcal{I}}$ for signature verification during presentation
- **Message Labels**: An ordered array mapping message indices to credential field names, enabling selective disclosure by field name rather than index

For credentials without issuer-provided commitment metadata, the holder generates local binding values: a random blinding factor $r \leftarrow \{0,1\}^{256}$ and a commitment hash $h = \text{SHA256}(\text{credentialId} \| r)$ for local credential identification.

The master secret ms is stored separately in the operating system's secure credential storage (e.g., Windows Credential Manager, macOS Keychain) encrypted with AES-256-GCM, ensuring isolation from credential data and providing hardware-backed protection where available. This separation ensures that compromise of credential storage does not expose the master secret required for authentication proof generation.

Privacy Properties: The stored credential structure enables the holder to: (1) generate unlinkable authentication proofs across different verifier domains through

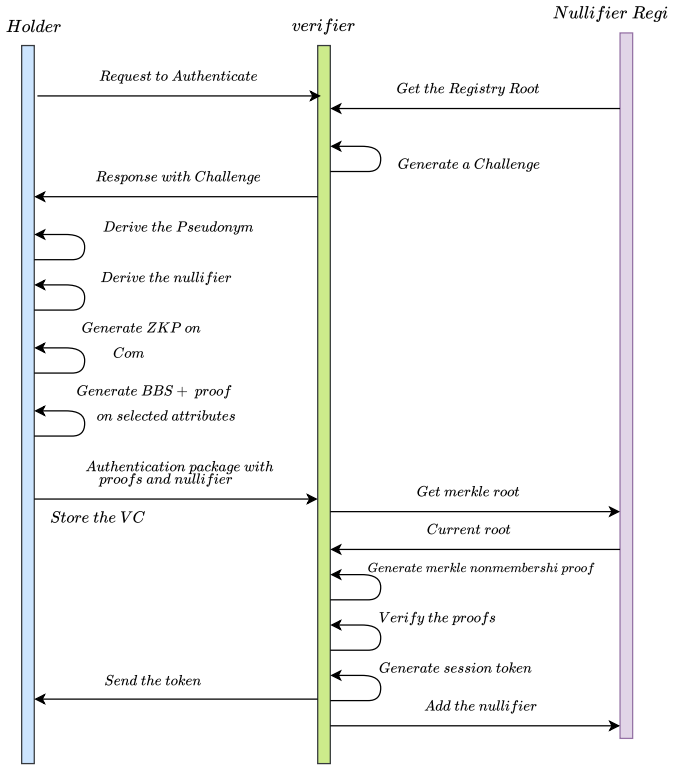


Figure 4.6: HALP Authentication Flow

domain-specific pseudonym derivation $P_d = G_d^{H(ms\|d)}$; (2) selectively disclose credential attributes using BBS+ proof-of-knowledge protocols; and (3) prevent credential correlation through the cryptographic binding between the master secret commitment and the BBS+ signature, ensuring only the legitimate holder can utilize the credential.

4.4.2 Anonymous Authentication Protocol

The HALP anonymous authentication protocol allows a holder $\mathcal{H}$ to prove possession of a valid BBS + credential and obtain a session token from a verifier $\mathcal{V}$ under domain d without revealing any secret attributes. Let *credID* denote the unique identifier of the stored credential, ms the holder's master secret, and $pk_{\mathcal{I}}$ the issuer's public key. The protocol proceeds in four phases, leveraging a Merkle-tree nullifier registry $\mathcal{R}$ for replay prevention and a SNARK circuit *circuitID* for proof verification.

Challenge Generation The Verifier $\mathcal{V}$ (Table 4.2) uses 256 bits of randomness (generated using crypto.randomBytes(32)) to prepare a 256-bit random challenge ch (Figure 4.7)in order to ensure each season is unique and prevent replay attacks. The challenge is hex-encode and accompanied with a unique 16 bits challengeId $chId$ to track the exact session. As part of operational security the Verifier maintains the challenge, fetch the $\text{root}_{\mathcal{R}}$ and store them in a activeChallenges Map. Entries do go out-of-date after fixed time with this map. The implementation meets formal cryptographic needs and at the same time enables the freshness of a new session.

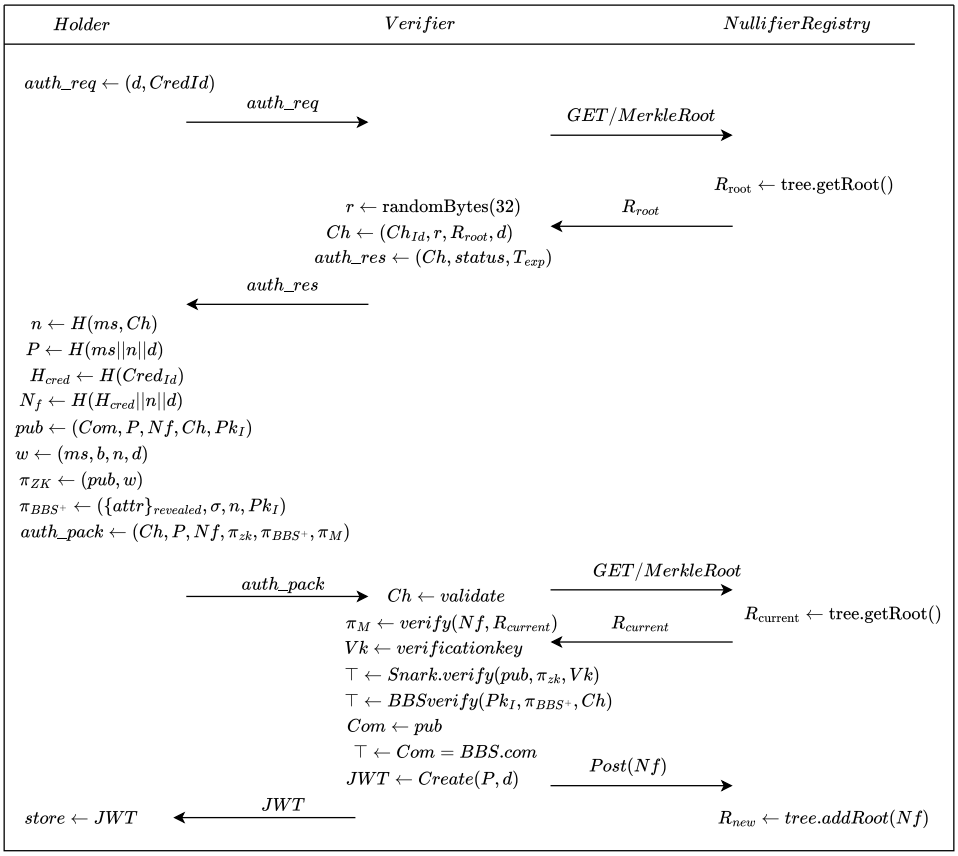


Figure 4.7: Cyptographic Protocol Flow of Verification

Credential Preparation and Identity Anchoring The credential preparation begins with the retrieval of the stored credential from a wallet storage. The Wallet Service read the W3C Verifiable Credential (VC), the BBS+ signature and original Pedersen commitment data. These components contain all the material needed in cryptography verification of identity. The system will then authenticate the master secret ms. It does verify the presence of a 256-bit BLS Fr scalar master secret (ms) in the OSkeychain. As mentioned previously this secret is the cryptography basis of the holder. Then to complete the preparation and counter the replay attack, the wallet send a authentication request to the Verifier. The verifier will response with ch, ch_{ID}, $\text{root}_{\mathcal{R}}$, $circuit_{Id}$. This is because the challenge-response mechanism ensures that all authentication requests are unique, and time-limited, to make it hard for any attacker to re-use a captured protocol message (Figure 4.7).

The Hybrid Proof Generation Engine: The process of generating proofs is controlled by a HybridProofGenerator function which is a combination of BBS+ selective disclosure and Groth16 SNARK proofs(Figure 4.7). This dual-proof method retains cryptographic integrity and unlinkability between sessions while allowing users to authenticate without disclosing any extraneous information.

Session-Specific Identifiers: It begins with the generation of session specific identifiers that make it impossible to identify between domains. The system produces a different contextual pseudonym by generating the poseidon hash value from ms, nonce n, and domainName d. The outcome is a hex-coded pseudonym (P) which is unique to each domain and binds the identity to it. A holder cannot be correlated across a set of services even though he/she might be logging in to several services at the same time.

$$P = H\big(ms \,\|\, n \,\|\, d\big),$$

Simultaneously, the system uses Poseidon to generate a cryptographic nullifier. The nullifier is used to create a single token (Nf) out of $credentialId$, $domainName$ and $nonce$. This burn token prevents the use of the same credential on the same domain twice, allowing it to be spent only once in the domain.

$$N_f = H\big(Cred_{Id} \,\|\, n \,\|\, d\big).$$

BBS+ Selective Disclosure: The wallet relies on the library, which is named, 0xmattrglobal/bbs-signatures to transform the signature of the issuer into a 0-knowledge presentation. This allows the issuer to verify that the signature is legitimate on the basis of the issuer public key without the credential data being disclosed. The holder will only reveal the essential characteristics such as name or the university they are with, but conceal sensitive elements. The Bbs+ proof is also bound to the challenge set by the verifier to prevent replay attacks and therefore, this proof is only valid under that particular session.

SNARK Circuit Constraints: The last step uses snarkjs and the halp -auth.circom circuit to generate a Groth16 SNARK proof ($\pi_z k$). Four significant restrictions on the inputs(Figure 4.7)of the privates are proven by the evidence:

1. The credential is appropriately recognized in the Merkle Registry, which is validated through a membership or non-membership path of proof.

$$\pi_M = \mathcal{R}.\text{NonMemberProof}(N_f,\ root_{\mathcal{R}}),$$

2. The nullifier was obtained correctly by using the holder masterSecret, which was consistent.
3. The evidence is tied to the challenge presented by the verifier, and this does not allow using a single evidence for multiple sessions. These limitations render the evidence genuine and new.

Public Inputs: $P, N_f, C, R, \mathcal{X}$
Private Witness : ms, n, b, d, Ch

The Authentication Package: All the results of the hybrid engine are represented in one Authentication Package, which is a structured JSON object. The contextual pseudonym, the cryptographic nullifier, and the hybrid proofs (SNARK and BBS+) are also included in the package, as well as explicitly revealed attributes. The conversion of cryptographic points and byte arrays into Base64 or hexadecimal strings is secured to transmit them. The package provides complete authentication testimony, which allows the verifier to identify the statements of the holder and not

StoredCredential

```
Credential{
  id: "urn:uuid:...",
  credential: W3C_VC_Object,
  bbsSignature: "base64-encoded-
signature",
  commitmentHash: "hex-string",
  blindingFactor: "hex-string",
  issuerPublicKey: "base64-public-
key",
  messageLabels: ["@context", "id",
"type", ...],
  storedAt: "2024-01-
01T00:00:00Z"
}
```

ChallengeResponse

```
ChallengeResponse{
  success: true,
  challenge: {
  challengeId: "7f3a2b1c...",
  challenge: "d4e6f8a1b2c3...",
  domain: "example.com",
  registryRoot: "a3f5b2c1...",
  circuitId: "halp-auth-v1",
  expiresAt: 1737705900000
 }
}
```

AuthenticationPackage

```
AuthPackage{
 challengeId: "7f3a2b1c...",
 challenge: "d4e6f8a1b2c3...",
 domain: "example.com",
 pseudonym: "8f3a2b1c...",
 nullifier: "2a4b6c8d...",
 hybridProof: {
   snarkProof: {
     proof: { pi_a, pi_b, pi_c },
     publicSignals: [root, nullifier,
challenge]
   },
   bbsProof: {
     proof: Uint8Array,
     revealedMessages: [...]
   }
 },
 revealedAttributes: {
   name: "Alice",
   university: "BracU"
 },
 timestamp: Date.now()
}
```

Figure 4.8: Json messages

the private information.

Phase 3: Proof Submission When the hybrid zero-knowledge proofs have been generated successfully, the Holder triggers the proof submission phase by using a standardized HTTP interface. The wallet service is a client in this critical step and in this step, the wallet service sends a package of consolidated authentication to the Verifier at a specified verification endpoint. This submission step will deliver the Verifier all the public inputs required to verify the SNARK and BBS+ components in a dual way and at the same time keeping a small, interoperable message format.

$$\text{AuthPkg} = \langle ch_{ID}, ch, d, P, N_f, \pi_{ZK}, \pi_{BBS}, \{attr_j\}_{j \in revealed}, t \rangle$$

Information Transmission and Structure of Packages: The submission of proof is enclosed in one POST request in a json format (Figure 4.8). The design is used to guarantee the heterogeneity of decentralized heterogeneous platforms and offer a standard machine-readable interface to the cryptographic verification. The architecture of the transmitted authentication package is such that it provides the Verifier with all the public inputs needed to execute the dual-verification mechanism in five critical functional components.

Session Context The authentication package includes the session identification $challenge_{Id}$ and the initial challenge value, allowing the verifier to find the correct session state in its internal active challenges mapping. This feature ensures that the presentation of proof is tied to a specific authentication session, preventing cross-session attacks and allowing the Verifier to maintain temporal integrity throughout the authentication lifecycle.

Anonymized Identifiers Two anonymized identifiers that are very critical are sent as hex-encoded strings. It uses a Pseudonym (P) which is the identity of the holder

to use in a particular session, and is created in a unique manner to ensure that the identity is not linked across a different session. The Nullifier (Nf) represents a replay prevention token, which allows the system to identify and reject multiple authentication activities, but keep the privacy of the user intact. Collectively, these identifiers detach the authentication transaction with the real world identity of Holder, but still retain the accountability mechanisms that are required in Revocation.

Verification & Token Issuance When the Verifier receives a hybrid authentication package, the logic of the Verifier will lead to a multi-tiered validation strategy. This is a series of countermeasures arranged in such a way that all the cryptographic constraints, temporal bindings, and protocol invariants are met, and then session authorization is granted. The process of verification is structured around four consecutive steps that are state validation, replay prevention, cryptographic proof verification and session binding, each of which has to pass on its own in order to prove a valid authenticated session.

Registry State validation and session validation. Before the actual computationally expensive cryptographic operations are performed, the Verifier conducts a sequence of initial state checks which are efficiently aimed at selectively weeding out stale, invalid or malformed requests. Such an optimization minimizes redundant cryptographic computation and ensures strong security.

Verification of Freshness of Challenges: The first step in state validation is the initiation by the Verifier searching the given challengeId in the activeChallenges map. The lookup should meet four criteria: the challenge should be present in the map, which should verify that the challenge has been issued earlier by the Verifier, the challenge must not be more than five minutes old, which constrains the authentication window and deters such token-fixation attacks; the challenge should not have been marked as used, which prevents such attacks; the domain field of the challenge should be identical to the claimed domain in the current request and prevents such cross-domain authentication attacks. This multi-factor freshness check acts as an effective gatekeeper and it disapproves a large number of invalid requests and moves on to more intensive verification operations.

Registry state synchronization: Once the challenge has been checked to be fresh the Verifier does a key consistency check. It retrieves the registryRoot value of the SNARK public signals and compares it with the one that was stored at the time when the challenge was generated.
This analogy will assure two facts:
(1) The evidence was generated on an up-to-date copy of the Nullifier Registry, and therefore Registry states that are old or obsolete cannot be employed;
(2) Evidences based on a revoked state of registry are not accepted. In this way, all authentication proofs have to indicate the current state in the registry.

The check of the registry synchronization maintains the semantics of revocation. In case a credential is revoked and its nullifier is included into the registry after the challenge has been generated but before submitting the proof, the inconsistency will

be detected and the authentication request will be rejected.

Prevention of Replay through Nullifier Checking: The Verifier then compares the received nullifier to the used Nullifiers set, which is simply an in-memory set of all the effectively processed nullifiers history of the previous sessions. In case the nullifier is in place, it will reject the request and avoid the replay attack.

Security Property and Uniqueness Guarantee: The replay prevention is based on one basic cryptography property: the nullifier is a deterministic hash of (Master, Secret, credential, ID, Domain). Consequently, it specially recognizes the use of a specific credential in a specific field. Thus, a domain can only use a given credential once in a session nonce. This is actually guaranteed by the deterministic nature of the hash function and the Master Secret that the Verifier does not see. Without knowing the Master Secret, or having some other valid credential, which are beyond the threat model of the protocol, this is impossible of an adversary to create a fresh nullifier.

Verification of cryptographic Proofs: The base checking step determines whether there are two distinct cryptographic verification systems that operate on two points of view of the HALP protocol. Having both proofs to authenticate gives defense-in-depth against attacks on either of the two schemes: the ZK-SNARK or the BBS+.

Zk-SNARK Proof Verification The snarkjs library is used by the Verifier to authenticate the Groth16 proof (π_{ZK}) with its own verification key that is located locally. This two-way check of the BLS12-381 curve proves that the circuit limitations were met. Effective verification ensures that the Holder has a valid credential, obtained the Nullifier in a sound manner and has included the challenge nonce. The check spills no information of the credential attributes or Master Secret.

$$e(\pi_A, vk_\alpha) \cdot e(\pi_B, vk_\beta) \cdot e(\pi_C, vk_\gamma) = e(vk_\delta, 1) \cdot \prod_j e(vk_{\gamma,j}, 1)$$

Verification of Signatures Proof: BBS+. Similar to SNARK verification, the Verifier checks the BBS+ selective disclosure evidence using the library, @mattrglobal/bbs-signatures. This checks three properties, namely that the signature is valid to the attributes revealed; that all the concealed attributes are still valid under the public key of the issuer; and that the proof is bound to the initial challenge nonce, so it cannot be reused across sessions or domains. The BBS+ verification does not rely on the SNARK verification, and it offers an independent guarantee that the attributes revealed are genuine and undistorted. This autonomy implies that in case the vulnerability is discovered in one of the proof systems, the other system still provides cryptographic security.

4.4.3 Session Management and Revocation

The successful authentication is followed by the implementation of a JWT-based session management protocol by the HALP system. Once the hybrid proof (SNARK + BBS+) is verified by the verifier, a session token is emitted which allows accessing the API again without the need to re-run zero-knowledge proofs.

Session Token Generation: When a successful authentication verification occurs the verifier service issues a JSON Web Token (JWT) with the following cryptographic payload: **Payload Components:**

$$
\begin{aligned}
P &\leftarrow \text{Session-specific unlinkable identifier (32 bytes hex)} \\
d &\leftarrow \text{Authentication context (e.g., “example.com”)} \\
\{attr_j\}_{j \in revealed} &\leftarrow \text{Disclosed credential fields (JSON object)} \\
\text{verifiedAt} &\leftarrow \text{Unix timestamp of authentication (milliseconds)} \\
\text{exp} &\leftarrow \text{Token expiration(verifiedAt} + 3600 \text{ seconds)} \\
\text{iat} &\leftarrow \text{Token issuance timestamp}
\end{aligned}
$$

The token signing uses the HS256 (HMAC-SHA256) with a shared secret .env file between all services, and provides cryptographic integrity. The default one hour expiration time trade-offs between security (minimizing the exposure period) and usability (minimizing the frequency of authentication).

Session Token Lifecycle: Issuance: the verifier then builds JWT payload with the successful verification of proof and signs it using the shared secret and sends the payload back to the wallet in addition to the verification status. This token is stored in the memory of the wallet to be used in further API calls.
To get access to a secured resource, the holder includes the session token in the HTTP `Authorization` header:
`Authorization:Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9`.

The verifier authenticates the signature on the token, ensures that it has not expired, and retrieves the pseudonym and the revealed attributes in order to make authorization decisions.

Expiration Sessions expire automatically after 3600 seconds (1 hour). The verifier rejects expired tokens and the holder must start a new authentication flow with new proof generation. This time constraint limits the effects of token compromise an attacker with a leaked token can only exploit such a token during the remaining validity period.

Revocation Mechanism in HALP: The HALP system uses a nullifier-based revocation mechanism, which is used to ensure that a credential is not replayed as well as maintaining privacy of a holder. In contrast to the conventional certificate revocation lists, which reveal the identifiers of the credential, HALP is unlinkable with the help of cryptographic nullifiers.

Derivation and Registration of Nullifiers: In the generation of authentication proofs, the wallet will compute a zero-knowledge circuit optimized deterministic nullifier Nf based on the Poseidon hash function: This formula ensures that: This credential produces various domain-specific nullifiers (domain-specific binding). The credential produces a variety of nullifiers on sessions within the same domain (nonce freshness). Legal identification of the nullifier is not possible without master secret knowledge. Upon successful verification, the verifier gets the nullifier out of the authentication package and enters it in the Indexed Merkle Tree of the registry

service.The registry contains a cryptographic accumulator of all used nullifiers, and membership may be checked efficiently with a logarithmic complexity of O(log n).

Replay Prevention Protocol: The nullifier mechanism prevents two critical attack vectors: Same-Credential Replay: The replay of an authentication proof can not be done by an attacker who has already obtained an authentication proof since the nullifier has been registered. Freshness of the nullifiers is verified by the verifier, that is, by querying the non-membership proof provided by the registry:

Nullifier Revocation Check Protocol:

1. Extract nullifier from authentication package
2. Query registry: `POST /merkle/non-membership-proof`
   Input: $\{\text{value} : N_f\}$
3. Verify response: $\{\text{exists} : \text{false}, \text{proof} : \pi_M\}$
4. (a) If exists = true $\rightarrow$ **REJECT** (credential already used)
   (b) If exists = false $\rightarrow$ **ACCEPT** and register nullifier

Cross-Session Linkability: Because nullifiers use session nonces, even the authorized user is not able to correlate his or her authentications across sessions. The authentication produces a cryptographically independent nullifier, which maintains long-term unlinkability.

Chapter 5

System Evaluation and Discussion

The chapter Performance Evaluation and System Analysis is a detailed evaluation of the efficiency, scalability and the security of the HALP system in real-life scenario. The chapter looks at the practical performance of the cryptographic protocols of HALP, including ZK-SNARK proof generation, BBS+ signature verification, and Merkle tree operations, both on client devices and on servers. The main performance measures such as the proof production time, the verification latency, and resource consumption are evaluated to assess the appropriateness of the system to apply in large scale deployment and usage on resource-constrained systems. Besides, the chapter talks about the resilience of the system, privacy assurance and the performance of the unlinkability and revocation mechanisms. This section will show through empirical findings and critical thinking how HALP has achieved its functional requirements whilst ensuring sound privacy and security provisions.

5.1 Performance Evaluation

Circuit Size Analysis

The time taken in each phase of HALP authentication in detailed in Figure 5.2. Pairing check constraints for BBS+ is provided in Table 5.2. Below we discuss the ZK-SNARK circuit constraint composition (Table 5.1) for HALP.

Merkle Proof Dominance (71.4%): Majority of the limitations in the focus on non-membership proof verification (Table 5.1). While this is necessary and unavoidable for replay protection, it is a decent cryptographic strategy that results in better security in general.

Efficient Hash Usage (17.6%): The circuit uses four Poseidon hashes, three Poseidon3 and two Poseidon2, with just approximately 1,500 additional constraints[30]. Poseidon is specifically developed for zero-knowledge applications, therefore it is significantly lighter than typical hashes (such as SHA-256), which would impose almost 20,000 limitations!

Minimal Overhead (0.1%): Control flow such as IsZero and OR gates contribute hardly nothing to the constraint count in a slim circuit.

Component	Constraints	Percentage
Poseidon3 (P)	∼400	4.7%
Poseidon3 (Nf)	∼400	4.7%
Poseidon2 (Com)	∼300	3.5%
LessThan (2×254-bit)	∼1,016	11.9%
IsZero + OR gate	∼5	0.1%
Poseidon3 (Leaf hash)	∼400	4.7%
MerkleProof (20 levels)	∼6,080	71.4%
Challenge binding	∼1	0.01%
Total	∼8,502	100%

Table 5.1: ZK-SNARK Circuit Constraint Composition for HALP Authentication Proof

Weaknesses and Possible Optimization Opportunities: Range Check Cost (11.9%): Two 254-bit LessThan comparisons (for enforcing boundaries on Merkle trees) result in 1016 constraints. This is a common constraint for comparison circuits in SNARKS. Replacing them with lookup tables in PLONK could reduce the cost.

Merkle Tree Depth A 20-depth tree, which can handle approximately a million nullifiers, introduces 6,080 restrictions. If the nullifier load is lower in the real world, the tree could be lowered to 16 levels, saving approximately 1,200 constraints.

Schnorr Based issuence: The issuance - commitment circuit is based on the Schnorr - based zero - knowledge proof (PoK) to deliver the privacy - preserving credential issuance in the HALP system. Rather than obliging the holder to reveal his identity to the issuer, the circuit allows the holder to prove possession of a valid master secret associated with a Pedersen commitment - without having to disclose the secret.This cryptographic approach is crucial for providing unlinkable credential issuance while also strongly attaching a credential to its authorized possessor.

Component	Constraints	Percentage
Scalar multiplication $G^{s \cdot ms}$	$\sim 2,500$	50%
Multi-exponentiation $\prod H_i^{s_i}$	$\sim 1,500$	30%
Point addition	~ 500	10%
Field operations	~ 500	10%
Total	$\sim 5,000$	100%

Table 5.2: Pairing Check Constraint Composition for BBS+ Signature Verification

SNARK Proof Generation (Groth16) Credential possession zero-knowledge proofs are generated using the HALP system with Groth16 ZK-SNARKs. The benchmarking of the proofs was done against the halp-auth.circom circuit that contains approximately 8,502 R1CS constraints. The snarkjs library was used with the Bengamena (BN128, or altbn128) elliptic curve, which needs to run on an implementation of Node.js 18 or later.

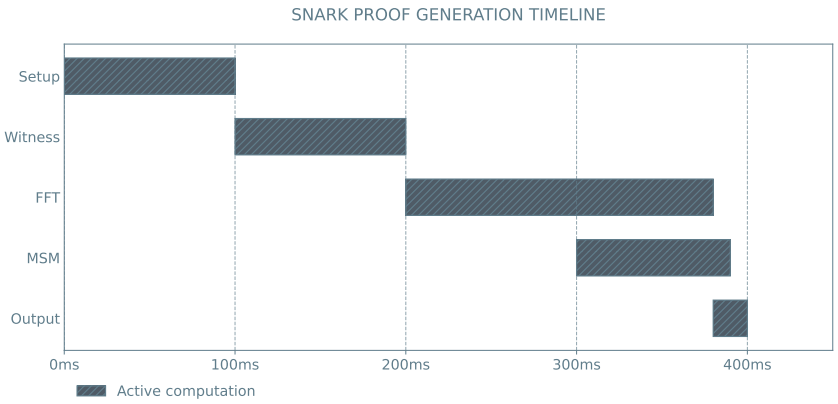


Figure 5.1: Snark Proof Generation Timeline

Generation Process and Performance: SNARK generation of proofs is signed in a system of pipeline. It is initiated by a setup phase (0-100ms) loading the WASM module and the proving key (zkey) into memory (Figure 5.1). Module and key sizes are provided in Table 5.4. Subsequent proofs do not do this step of caching because they use these resources as the input. The second step is the witness computation step (100-200ms) where all the intermediate circuit values of the private inputs are calculated that include the master secret, session nonce and the Merkle proof siblings.

Fast fourier transform (FFT) operations (200-400 ms) and multi-scalar multiplication (MSM) (350-450 ms) are the most compute-intensive steps. The two of them take up approximately 70 percent of the proving time, transforming the witness in the form of the polynomial commitment that Groth16 uses. The last step compresses the elements of proof (π_A, π_B, π_C) to a more compact 128-byte format.

According to performance measurements performance of the first time creation is between 470 and 850 ms due to the cost of the initialisation. The next evidences are completed in 300-500ms and so the average generation time of 400ms is openly satisfactory in interactive authentication.

Metric	**Value**
Generation Time	40–60 ms
Public Key Size	96 bytes (G2 compressed)
Secret Key Size	32 bytes
Security Level	128 bits

Table 5.3: BBS+ Key Generation Metrics and Security

Proof Structure and Size: The Groth16 proof consists of three elliptic curve points, which are: π_A, π_B and π_C. Together they use 128 bytes. Including the 5 public parameters (Table 5.5), a typical SNARK package is 288-bytes. Such a small size has the advantage of storage and transmission.

Key / Artifact Type	Size
Proving Key ($zkey$)	∼15 MB
Verification Key ($vkey$)	∼2 KB
WASM Module($wasm$)	∼3 MB

Table 5.4: ZK-SNARK Circuit Key and Artifact Sizes

Input	Size	Total
Pseudonym	32 bytes	
Nullifier	32 bytes	
Commitment Hash	32 bytes	
Registry Root	32 bytes	
Challenge	32 bytes	
Total Public Inputs		**160 bytes**

Table 5.5: Public Inputs Size: ZK-SNARK Proof Package Composition

BBS+ Proof Generation: The BBS+ signature allows for selective disclosure of credential attributes, allowing the holder to disclose only the amount of claims required while demonstrating possession of the entire signed credential. BBS+ uses the BLS12-381 curve, which provides 128 bits security [49] in HALP. Critical Generation and Credential Signing. The minimum time to build a BLS12-381 key pair is 40-60ms, where public key is of size 96-byte and secret key 32-byte (Table 5.3). Credential signing takes O(n) [37] where n is the number of attributes. Its empirical formula (Equation 5.1) milliseconds provides a typical 5 -attribute credential that signs approximately 35 ms and a 20 -attributes credential that signs approximately 95 ms.

$$T_{\text{sign}} \approx 15 + 4n \tag{5.1}$$

Generation of Proofs of Disclosure Selectivity. Jurisdictional-disclosure evidence is a four stage process:
1. The encoding (2-5ms) converts the credential attributes into field values applicable to BBS+.
2. The zero -knowledge proof creation (50-100 ms) constructs the proof of knowing hidden messages without being able to see them.
3. Nonce binding (1 -2 ms) is the challenge the verifier adds to disallow replay-attacks.
4. This is followed by serialization (2 5 ms) of the proof in a transmittable form.

Time to generate complete proofs is between 55 and 112ms with different numbers of attributes revealed. The complexity is still O(n) of the number of messages. The size of a base proof is 80 bytes and is increased by an average of 36 bytes with each revealed attribute, and extensively practically, we achieve 150 to 300 attributes in common use.

Hybrid Proof Integration In the hybrid authentication between SNARK and BBS+ proofs, the system executes the two generation processes sequentially. The hybridized proof package, which combines the SNARK proof, the BBS+ selective-

Proof Type	Min (ms)	Avg (ms)	Max (ms)
SNARK only	300	400	500
BBS+ only	50	75	100
Hybrid	400	550	700

Table 5.6: Proof Generation Time Analysis

disclosure proof, identifiers and metadata, is approximately 826 bytes. The average and range of the generation times are (Figure 4.4) 550 and 400-700ms (Table 5.6) respectively. This is a design that provides the greatest flexibility of privacy: the SNARK is able to show the possession of credentials and master-secret binding without revealing any contents of those credentials, whereas the BBS+ proof can provide a fine-grained disclosure of attributes as needed by the verifier.
Experiments on the performance of the system prove that the HALP proof generation system can provide realistic, sub-second generation times and economy of proof sizes that can be deployed to the real-world and that adhere to both bandwidth and user experience limitations.

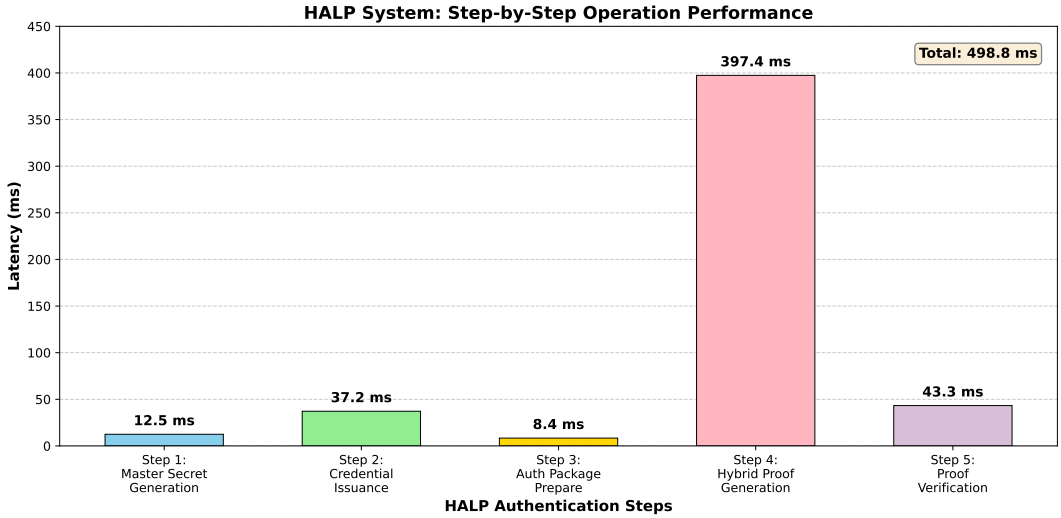


Figure 5.2: Halp Authentication Timeline

Pseudonym Derivation: The P=Poseidon(ms, n, d) pseudonym is computed using three inputs that include the master secret (ms) of a holder a fresh session nonce (n) and the domain identitfier(d). This three-input version of Poseidon hash [30] takes about 600 microseconds to calculate (Table 5.7). The pseudonym is the identifier that cannot be linked to a single authentication session of the holder, as different authentication sessions of the same holder generate a different pseudonym, whereas the same holder authenticating to different domains gets a domain-specific pseudonym that cannot be correlated to.

Nullifier Generation: The nullifier Nf=Poseidon$(cred_{Id})$,n,d) has the same computational pattern and it is input with the credential identifier ($cred_{Id}$), session nonce (n) and domain (d). The execution time of about 600 microseconds is a re-

flection of the derivation of the pseudonym because both use Poseidon [30] with three inputs. The nullifier has a very different purpose: it cannot be reused in the same domain context after the credentials have been published to the registry because it was successfully verified and the nullifier was used.

Session Nonce Generation: The nonce of the session is the production of a random element in the BN128 scalar field Fr and takes approximately 100 microseconds. It is based on the cryptographically secure random number generator built into the system to generate a 254-bit number that is spread evenly over the field. The freshness of the nonce is important–this guarantees that the pseudonym as well as the nullifier is unique in a session even when the same holder applies the same credential.

Domain Hash Computation: The domain string (e.g. example.com) should be turned into a field element via Poseidon before pseudonym and nullifier derivation. This has been demonstrated to require approximately 400 microseconds on Poseidon with variable length input encoding. Practically, this value can be stored in cache of often used domains and it would do away with overhead of repeated computation.

Poseidon is reported to take a slow 600 microseconds compared to traditional hash algorithms such as SHA-256 at about 1 microsecond. But this analogy is incorrect with the case of zero-knowledge. Application of SHA-256 in ZK-SNARK circuit would need about 25,000 R1CS constraints in a hash, as opposed to Poseidon's roughly 400 constraints. The native computation overhead is the intentional trade-off that makes circuit complexities and time to generate proofs, the primary cost in the authentication flow significantly lower.

Operation	**Formula**	**Time**
Pseudonym	$P = \text{Poseidon}(ms, n, d)$	~ 600 µs
Nullifier	$N_f = \text{Poseidon}(cid, n, d)$	~ 600 µs
Session nonce	Random $\mathbb{F}_r$ element	~ 100 µs
Domain hash	$\text{Poseidon}(\text{domain}_{\text{bytes}})$	~ 400 µs

Table 5.7: Prover-Side Operations: Timing Breakdown

5.1.1 Comparison With Existing System:

The privacy feature matrix compares five anonymous credential systems: HALP, Idemix [13], U-Prove [10], Semaphore [51][28] and AnonCreds [43][46] based on seven criteria (Table 5.8).

Core Privacy Features: The privacy capabilities are based on the attributes, Hiding and Selective Disclosure. Full attribute hiding is offered by HALP (signature scheme BBS+) and Idemix/AnonCreds (CL signatures) and U-Prove (Schnorr based commitments). The BBS+ selective disclosure of HALP has complexity of $O(m)$ [37] where m is the disclosed attribute count and this means that attribute granularity can be achieved without needing credentials that are specific to each

Feature	HALP	Idemix	U-Prove	Semaphore	AnonCreds
Attribute Hiding	✓	✓	✓	N/A	✓
Selective Disclosure	✓	✓	✓	×	✓
Predicate Proofs	✓	✓	Limited	×	✓
Multi-show Unlinkability	✓	✓	Limited	✓	✓
Replay Prevention	✓	Partial	✓	✓	Partial
Revocation	✓	✓	×	×	✓
W3C VC Compatible	✓	×	×	×	✓

Table 5.8: Privacy Feature Matrix: Anonymous Authentication Systems

disclosure combination.

Predicate Proofs allow proving statements concerning attributes without the knowledge of the values (revealing only $age \geq 18$).The Groth16 circuit of HALP employs arithmetic constraints to implement predicates, which can be quickly tested but must be changed to support additional predicate types. Idemix and AnonCreds provide native predicate support via sigma protocols, which are more flexible at runtime but require bigger proofs.

Multi-Show Unlinkability precludes the connection of numerous credential presentations. HALP can be completely unlinkable by utilizing pseudonym derivation: P, and new session nonce: n can be used to construct cryptographically independent pseudonyms each time authentication is conducted. U-Prove is not particularly unlinkable due to its token-based structure, in which each token can only be presented a few times.

Replay precautions prevent captured proofs from being replayed. HALP uses Indexed Merkle Trees in its nullifier Nf registry to guarantee full prevention—each authentication generates a deterministic nullifier that is published upon verification, cryptographically preventing replay with negligible collision probability ($\epsilon \leq 2^{-254}$). Semaphore has a similar technique, although Idemix and AnonCreds depend on external challenge-response mechanisms.
U-Prove uses accumulators for revocation [17], which is not efficient for size as it grows linearly, whereas merkle trees.

Standards and Architecture W3C VC Compatibility is what defines ecosystem interoperability. HALP is W3C VC Data Model v2.0 compliant so that it can interact with W3C wallets and verifiers. AnonCreds has passed W3C compatibility via AnonCreds-W3C specification. Idemix, U-Prove and Semaphore have proprietary formats that have poor interoperability.

Efficiency characteristics are basically determined by ZK-SNARK Foundation. Both HALP and Semaphore use Groth16 ZK-SNARKs, both of which verify in O(1) time irrespective of the complexity of the statement. Idemix, U-Prove, and AnonCreds use sigma protocols with scaling verification of O(n), with larger proofs and slower verification of complex credentials.

Comparison Analysis on Protocol level: The protocol-level comparison measures systems in five performance dimensions (Table 5.9) which are proof type, proof size, generation time, verification time, and setup requirements.

System	Proof Type	Proof Size	Gen Time	Verify Time	Setup
HALP	Groth16 + BBS+	∼826 B	550 ms	80 ms	Trusted
Idemix	CL + Σ-protocol	∼2–4 KB	800–1200 ms	150–300 ms	None
U-Prove	Schnorr-based	∼1–2 KB	100–200 ms	50–100 ms	None
Semaphore	Groth16	∼256 B	300 ms	30 ms	Trusted
IRMA	Idemix-based	∼2–4 KB	500–800 ms	100–200 ms	None
AnonCreds	CL Signatures	∼2–3 KB	600–900 ms	120–250 ms	None

Table 5.9: Protocol-Level Comparison of Anonymous Authentication Systems

Evidence Size directly affects the cost of transmission and storage needs. The SNARK and BBS+ with metadata version of the hybrid proof is 826 bytes on average (HALP), which is a 3-5 times smaller size than Idemix and AnonCreds proofs (2-4 KB). Semaphore makes the minimal proofs of 256 bytes by avoiding attribute support and only doing membership proofs.

User experience is determined by Generation Performance. The hybrid proof generation of HALP takes about 550ms (400ms SNARK and 150ms BBS+). Although it is slower than the 100-200ms of U-Prove, this is within a reasonable range of interactive authentication. Idemix and AnonCreds take 600-1200ms because of the overhead of the sigma protocol.

Scalability at the service side is dependent on Verification Efficiency. The full hybrid check of HALP can be done in a time of about 80ms–allowing about 12 checks per second per CPU core. Importantly, Groth16 has the O(1) property and thus the verification time of HALP is independent of the complexity of the credential. Idemix and AnonCreds verification (120-300ms) have disclosed attributes.

Setup Trade-offs The major deployment consideration of HALP is Trusted Setup. Groth16 needs a circuit-dependent ceremony, security based on the honesty of at least one of the participants. Idemix, U-Prove and AnonCreds all have transparent setup zero ceremony. HALP alleviates this by via existing contribution of Powers of Tau and possible future migration to universal-setup systems such as PLONK.

t**Comparative Positioning** It has been analyzed that HALP fits into a unique position in the design space: SNARK-succinct (826 bytes, 80ms verification) and with rich attribute semantics on par with traditional anonymous credential systems. The 3-5x factorization of the proof size and constant time verification is a major plus to bandwidth-limited or high-throughput applications, at the expense of a trusted setup and more or less expensive proof generation.

HALP is very attractive in deployments where verification throughput is a priority, proof compactness is important, and compatibility with W3C ecosystem. In deployments where trusted setup is inadmissible or the speed of generation of proof

is paramount, U-Prove or systems based on sigma-protocol can be found more appropriate. There is no single system that is superior in all aspects, architecture decisions are based on underlying trade-offs that need to be assessed with a particular deployment.

5.2 Analysis of Design Solution

5.2.1 Security Analysis

Spoofing: The HALP system prevents spoofing by associating credentials with a particular user through a multi-layer security approach. Every user has their own master secret ms, which is a random 254-bit scalar from the BLS12-381 field $\mathbb{F}_r$. This secret is never disclosed or transmitted over the network. Instead, the system uses a Pedersen commitment scheme, which is efficient for cryptographic proofs, to link the secret to user credentials through the Poseidon hash function [30]. The perfect $\mathcal{L} - HVZK$ property of BBS+ [35] and zero-knowledge property of the Schnorr proof [48] used for commitments ensures that an adversary cannot learn any witness information, hence cannot diguise their identity (Equation 5.2,Equation 5.5).

$$\pi_C = \pi_{\mathsf{ZK}} \{(ms, r) : C = \mathsf{H}(ms \| r)\} \tag{5.2}$$

$$m = (C, attrs_2, \ldots, attrs_n) \tag{5.3}$$

$$\sigma = \text{BBS}^+.Sign(sk_{\mathcal{I}}, m) \tag{5.4}$$

where [34]

$$(A, e, s) \leftarrow \text{BBS}^+.Sign(sk_{\mathcal{I}}, m). \tag{5.5}$$

Tampering The HALP system ensures data integrity using three total layers of cryptography. First, it uses BBS+ signatures which are robust against forgery [49]. Because these signatures are resistant to chosen message attacks it is not possible for an unauthorised party to create a valid signature even when they observe a large number of legitimate transactions. A signature is only identified as true if it was created using the appropriate secret key, and keeps every credential authentic and tamper proof. For n length message, $\sigma = (A, e, s)$ and $pk_{\mathcal{I}} \in \mathbb{G}_2$, $H \in \mathbb{G}_1^{n+1}$, the signature is verified when both sides are identical [34] in the pairing equation Equation 5.6:

$$\mathbf{e}(A, pk_{\mathcal{I}} + e \cdot G_2) = \mathbf{e}\left(G_1 + s \cdot H_1 + \sum_{i=1}^{n} m_i \cdot H_{i+1}, G_2\right). \tag{5.6}$$

Second, the Groth16 SNARK proofing system ensures that it is computationally sound. A proof just passes verification if it corresponds to a valid set of hidden data satisfying all constraints in the system. For an adversary to produce valid proof is unlikely, because the advantage when playing the strong unforgeability game (Figure 5.5) :

$$\text{Adv}^{\mathsf{suf}}_{SS}(A, \lambda) := \Pr[\mathsf{SUF}^A_{SS}(\lambda) = 1] \tag{5.7}$$

is equivalent to the probability of the adversary solving the $q - sdh$ problem for the same parameters [49].

Finally, the nullifier registry is protected from the integrity perspective by a structure consisting of a Merkle tree. Each proof is cryptographically tied to a particular registry root which makes reusing a proof or transferring them from registry to registry impossible. Any non-sanctioned registry modification entails a root mismatch and the system will immediately detect this and reject it, rendering the registry consistent and reliable. The verifier checks:

$$\text{root}_{\mathcal{R}'} \neq \text{root}_{\mathcal{R}} \implies \pi_{\mathsf{M}}(N_f, \text{root}_{\mathcal{R}}) \text{ fails} \tag{5.8}$$

$$\pi_{\mathsf{M}}(N_f, \text{root}_{\mathcal{R}}) \stackrel{?}{=} \text{valid} \wedge \text{root}_{\mathcal{R}} = \text{root}_{\text{current}} \tag{5.9}$$

Repudiation The HALP system eliminates the prospect of repudiation by means of the deterministic nullifier generation process. During each authentication session, the holder generates a one-time nullifier by hashing his or her credential ID and a session nonce and domain using Poseidon. This nullifier is a permanent marker of the session that is non-repudiable. It has three important properties: Uniqueness, determinism and irreversibility that keep the original private inputs hidden and unrecoverable. After authentication, the nullifier is kept in a public registry, making the authentication event immutable. Because only the user knows a master secret that is needed to generate a valid proof to that nullifier, that user cannot later deny that it participated. Furthermore, a challenge-response binding for temporal freshness is introduced by the system, which mandates that every proof contain a random verifier challenge. This keeps intercepted authentication data from being used again and guarantees that each proof is new. Nullifier Properties:
Uniqueness: Each $(credId, nonce, domain)$ tuple produces exactly one nullifier.
Determinism: Same inputs always produce same nullifier.
Irreversibility: Cannot derive inputs from nullifier.

Information Disclosure The HALP system guarantees high confidentiality by a combination of zero-knowledge proof (ZKP) and selective revealing. The Groth16 and BBS+ satisfy perfect zero-knowledge and perfect $\mathcal{L} - HVZK$ respectively [19],[35]. Hence, real proofs are indistinguishable from proofs generated by a simulator without knowledge of secret. The real and simulated witness hiding games entail an adversary querying a real and simulated oracle, respectively, for proofs based on data *pub*. In simple terms, the two zero-knowledge properties ensure that a scenario $game_0$ with real Groth16 and BBS+ proofs, followed by $game_1$ with Groth16 proof substituted with simulation, followed by $game_2$ with both proofs substituted with simulation are all computationally indistinguishable. If an adversary cannot tell apart real and simulated proofs, they cannot learn witness data from proofs; hence, we achieve witness-hiding. We express formally:

Definition 21 (Witness Hiding)**.** The HALP system Π_{HALP} achieves witness hiding if for all PPT adversaries A:

$$\text{Adv}^{WH}_{HALP}(\mathcal{A}, \lambda) = |\Pr[\text{WH-Real}_{\Pi_{HALP}}(\lambda) = 1] - \Pr[\text{WH-Sim}_{\Pi_{HALP}}(\lambda) = 1]|$$

where the witness includes $w = priv^* = (ms, n, A, e, s)$ for master secret ms, nonce n, BBS+ signature (A, e, s), and Groth16 and BBS+ proof systems are π_G, π_B, respectively (Figure 5.3).

Game WH-Real$_{\Pi_{HALP}}(\lambda)$	**Oracle** $\mathcal{O}_{Auth}(pub^*)$
$pub^* \leftarrow$ HALP.Setup(λ)	obtain $priv^*$s.t. $P(pub^*, priv^*) = 1$
$b \leftarrow A^{\mathcal{O}_{Auth}}(pub^*)$	$\pi^*_{ZK} \leftarrow \pi_G(priv^*, pub^*)$
return b	$\pi^*_{BBS} \leftarrow \pi_B(priv^*, pub^*)$
	return $(\pi^*_{ZK}, \pi^*_{BBS})$

Game WH-Sim$_{\Pi_{HALP}}(\lambda)$	**Oracle** $\mathcal{O}_{SimAuth}(pub^*, trap)$
$(pub^*, trap) \leftarrow$ Sim.Setup(λ)	$\pi^*_{ZK} \leftarrow \pi_{G_{Sim}}(trap, pub^*)$
$b \leftarrow A^{\mathcal{O}_{SimAuth}}(pub^*, trap)$	$\pi^*_{BBS} \leftarrow \pi_{B_{Sim}}(trap, pub^*)$
return b	**return** $(\pi^*_{ZK}, \pi^*_{BBS})$

Figure 5.3: HALP Witness Hiding Game and Oracles

Theorem 1 (Witness Hiding)**.** *If the Groth16 and BBS+ proof systems (π_G, π_B, respectively) are witness hiding, then the HALP system Π_{HALP} achieves witness hiding in the oracle model.*

Proof. We prove the theorem via a sequence of games. Let λ be the security parameter, and let $\mathcal{A}$ be any PPT adversary attacking the witness hiding property of Π_{HALP}. We define the advantage of $\mathcal{A}$ as:

$$\text{Adv}^{WH}_{HALP}(\mathcal{A}, \lambda) = |\Pr[\text{WH-Real}_{\Pi_{HALP}}(\lambda) = 1] - \Pr[\text{WH-Sim}_{\Pi_{HALP}}(\lambda) = 1]|$$

We construct a sequence of games Game_0 through Game_2 such that:

- Game_0 is identical to WH-Real$_{\Pi_{HALP}}(\lambda)$
- Game_2 is identical to WH-Sim$_{\Pi_{HALP}}(\lambda)$
- Successive games are computationally indistinguishable under the stated assumptions

By witness hiding, we mean that a probabilistic polynomial time adversary cannot distinguish a proof made from a real witness and a proof made from a simulated witness [36]. This is true for BBS+ as it satisfies perfect $\mathcal{L} - HVZK$, which states that the simulated and real distributions are identical given that verifier computations follow the exact proof protocols; hence, they are computationally indistinguishable [35]. On the other hand, proofs by Groth16 are perfect zero-knowledge[19], which is a stronger condition meaning that simulated and real proofs have identical distributions given any deviation in the protocol by verifier; hence, they are also computationally indistinguishable. However, for our purposes, it is enough for the exact protocol to be followed since our scheme is non-interactive, so there is no live verifier to deviate the protocol, and oracle queries in the game can only give fixed outputs.

Game Definitions: **Game_0:** This is the real witness hiding experiment where $\mathcal{A}$ has oracle access to $\mathcal{O}_{Auth}$ which returns:

$$(\pi^*_{ZK}, \pi^*_{BBS}) \leftarrow \pi_G(priv^*, pub^*)$$

where $priv^* = (ms, n, A, e, s)$ and pub^* includes all public parameters.
Game$_1$: Identical to Game$_0$, except we replace the main Groth16 proof π^*_{ZK} with a simulated proof. Specifically, for each oracle query:

$$\pi^{*'}_{ZK} \leftarrow \Pi_{G_{Sim}}(trap, pub^*)$$

.

Claim 1. Game$_0$ and Game$_1$ are computationally indistinguishable.

Proof of Claim 1. This follows directly from the perfect zero-knowledge property of the Groth16 proof system[19]. For any statement $x = pub^*$ and witness $w = priv^*$, the distributions:

$$\{\pi \leftarrow \Pi_G(w, x)\} \quad \text{and} \quad \{\pi' \leftarrow \Pi_{G_{Sim}}(\text{trap}, x)\}$$

are identical. Therefore, no adversary $\mathcal{A}$ can distinguish between Game$_0$ and Game$_1$, even with multiple oracle queries. □

Game$_2$ **(Simulate BBS+ Groth16 Proof):** Identical to Game$_1$, except we replace π^*_{BBS} with a simulated proof:

$$\pi^{*'}_{BBS} \leftarrow \Pi_{B_{Sim}}(\text{trap}, pub^*)$$

where Π_{BBS} is the Groth16 proof system instantiated for the BBS+ statement.

Claim 2. Game$_1$ and Game$_2$ are computationally indistinguishable.

Proof of Claim 2. This follows directly from the perfect $\mathcal{L} - HVZK$ of the BBS+ proof system. Again, for any statement $x = pub^*$ and witness $w = priv^*$, the distributions

$$\{\pi \leftarrow \Pi_B(w, x)\} \quad \text{and} \quad \{\pi' \leftarrow \Pi_{B_{Sim}}(\text{trap}, x)\}$$

□

are identical. Therefore, no adversary $\mathcal{A}$ can distinguish between Game$_0$ and Game$_1$, even with multiple oracle queries.

Final Calculation: From the sequence of games, we have:

$$\begin{aligned}
\text{Adv}^{WH}_{HALP}(\mathcal{A}, \lambda) &= |\Pr[\text{Game}_0 = 1] - \Pr[\text{Game}_2 = 1]| \\
&\leq \sum_{i=0}^{1} |\Pr[\text{Game}_i = 1] - \Pr[\text{Game}_{i+1} = 1]| \\
&\leq \text{negl}_1(\lambda) + \text{negl}_2(\lambda) \\
&\leq \text{negl}(\lambda)
\end{aligned}$$

where each $\text{negl}_i(\lambda)$ is negligible by the corresponding claim.
Therefore, for all PPT adversaries $\mathcal{A}$, $\text{Adv}^{WH}_{HALP}(\mathcal{A}, \lambda) \leq \text{negl}(\lambda)$, proving that Π_{HALP} achieves witness hiding in the oracle model. □

The BBS+ selective disclosure mechanism allows users to have granular control over personal data. When creating a proof, a holder can disclose only certain attributes of the proof it creates, while hiding the others. For example, a user can prove that he or she is over a certain age by revealing only an "eligibility" attribute without giving out his or her birthdate or other sensitizing markers. The BBS+ proof is that the user has a valid signature over the entire credential, despite the fact only a subset of it is shared. The most sensitive elements i.e. master secret, session nonce, blinding factor, and Merkle path are private inputs to the SNARK circuit and are never over the network. The circuit allows users to make statements about these values locally, thus keeping a large part of their identity away from exposure and with a high level of security.

Denial of Service: The HALP system makes a number of design choices that are protective against denial-of-service attacks. Even for extremely complicated proofs, the Groth16 technique only requires three pairing operations, giving verifiers an $O(1)$ effort. For both proof synthesis and proof verification, the indexed Merkle tree offers $O(\log n)$ complexity, where n is the number of stored nullifiers. With a tree height of 20 levels that are able to put in a capacity of nearly one million entries, proof operations are still efficient at maximum capacity. In addition, the BBS+ signature scheme supports batch verification, which allows many proofs to be checked together more efficiently (5.10) than individually, and therefore enabling high throughput scenarios while retaining security guarantees.

$$e(\pi_{\mathrm{ZK}}, G_2) \stackrel{?}{=} e(G_1, G_2) \cdot e\left(\sum_{i=1}^{n} m_i \cdot vk_i, G_2\right) \cdot e(\pi_c, \delta) \tag{5.10}$$

Elevation of privilege: The HALP system prevents privilege escalation using four mechanisms that are oblivious to each other. First, the nullifier registry is only used once. Once a credential is registered, it cannot be used for any further authentication attempts because the non-membership proof is no longer valid. Second, BBS+ message binding avoids attribute inflation; the signature of the message encompasses all the attribute of the credential and any change invalidates the signature. Third, domain separation is realized by deriving a pseudonym. This ensures that credentials that are being authenticated to one service cannot be replayed to another one, because each domain perceives a different unlinkable pseudonym. Finally, in combination with nullifier uniqueness, challenge freshness thwarts replay attacks, that is, captured proofs expire along with the challenge and the nullifier becomes registered and cannot be used again.

Unlinkability: The unlinkability for game HALP is simply the witness hiding game played twice, once for a set of data pub_1^*, then for a set of data pub_2^*, and trying to see if they match. However, the perfect hiding of pedersen commitments and poseidon hash ensures that after reaching $game_2$ in the previous proof, if the data is swapped from pub_1^* to pub_2^* to reach $game_3$, then the two games are indistinguishable, and finally we substitute the simulated proofs one after another to reach $game_5$, where $game_1$ is the real game for pub_1^* and $game_5$ is the real game for pub_2^*, which means that these are indistinguishable.

Definition 22 (Unlinkability)**.** The HALP system Π_{HALP} achieves unlinkability if for all PPT adversaries $\mathcal{A}$:

$$\mathrm{Adv}^{UL}_{HALP}(\mathcal{A},\lambda) = \left|\Pr[\text{UL-}\Pi_{HALP}(\lambda)=1] - \frac{1}{2}\right| =$$

$$\left|\Pr[\text{WH-Real}_{\Pi_{HALP1}}(\lambda)=1] - \Pr[\text{WH-Real}_{\Pi_{HALP2}}(\lambda)=1] - \frac{1}{2}\right| \leq \mathrm{negl}(\lambda)$$

where the witness includes $w = priv^* = (ms, n, A, e, s)$ for master secret ms, nonce n, BBS+ signature (A, e, s), and Groth16 and BBS+ proof systems are π_G, π_B, respectively (Figure 5.4).

Game UL-$\Pi_{HALP}(\lambda)$	**Oracle** $\mathcal{O}_{Auth}(pub^*)$
$pub_1^* \leftarrow$ HALP.Setup(λ)	obtain $priv^*$s.t. $P(pub^*, priv^*) = 1$
$pub_2^* \leftarrow$ HALP.Setup(λ)	$\pi^*_{ZK} \leftarrow \pi_G(priv^*, pub^*)$
$b_1 \leftarrow A^{\mathcal{O}_{Auth}}(pub_1^*)$	$\pi^*_{BBS} \leftarrow \pi_B(priv^*, pub^*)$
$b_2 \leftarrow A^{\mathcal{O}_{Auth}}(pub_2^*)$	**return** $(\pi^*_{ZK}, \pi^*_{BBS})$
return $b_1 == b_2$	

Figure 5.4: HALP Unlinkability Game and Oracle

Unforgeability HALP used BBS+ signature for signing credentials, which satisfies strong unforgeability. We note the following game based definition:

Definition 23 (Strong Unforgeability Game)**.** In the strong unforgeability game defined $\mathbf{SUF}^A_{BBS+}(\lambda)$, the advantage of any adversary A playing the game is given by

$$\mathbf{Adv}^{\mathsf{suf}}_{BBS+}(A,\lambda) := \Pr[\mathsf{SUF}^A_{BBS+}(\lambda)=1].$$

which has been proven [49] to be just as probable as the adversary solving an equivalent q-sdh problem for security parameter λ (Figure 5.5).

Game SUF$^A_{BBS+}(\lambda)$	**Oracle S(m)**
Sigs $\leftarrow \emptyset; par \leftarrow BBS+.Setup(1^\lambda)$	$\sigma \leftarrow BBS+.Sign(sk, m)$
(sk, vk) $\leftarrow BBS+.KeyGen()$	**if** $\sigma \neq \perp$ **then**
(m*, σ^***)** $\leftarrow A^{\mathbf{S}}(par, vk)$	**Sigs** $\leftarrow$ **Sigs** $\cup \{(m, \sigma)\}$
if $(m^*, \sigma^*) \notin Sigs$ **and**	**return** σ
$BBS+.Ver(vk, m^*, \sigma^*) = 1$ **then**	
return 1	
return 0	

Figure 5.5: Strong Unforgeability Game

Completeness, Soundness and Zero-Knowledge: The HALP system relies on the completeness, soundness and zero-knowledge properties of the Groth16 and BBS+ proofs. A detailed explanation of the zero-knowledge satisfaction of HALP

is achieved through the witness hiding game, which shows that witness data cannot be learned even if proof and public data is attained.

5.3 Research Objective Analysis

HALP successfully fulfills all six research objectives (ROs) outlined in Section 1.4. The design of a Hybrid Anonymous Login Protocol (RO1) was implemented through the integration of ZK-SNARKs, one-time pseudonyms, and nullifiers, supporting both on-chain and off-chain revocation. Session-specific pseudonyms and efficient proof verification using Groth16 allowed us to achieve unlinkability and scalability (RO2). Selective revocation (RO3) was fulfilled by implementing nullifier registries and Merkle tree-based non-membership proofs, which is also better scalable compared to accumulators. Benchmarking has ensured that performance has been optimization (RO4), as proof generation and verification require sub-second times. Formal security analysis (RO5) was ensured through STRIDE threat modeling, and our system satisfies unlinkability and revocation. Finally, an open-source prototype (RO6) was developed in Rust/TypeScript, confirming practical feasibility and interoperability with W3C Verifiable Credentials.

5.4 Requirement Fulfilment Analysis

The security analysis has been detailed in section 5.2.1, and shows fulfillment of all threat modeling requirements. Anonymous authentication (F1) and unlinkable sessions (F2) are ensured in HALP by use of zero-knowledge proofs and dynamically generated pseudonyms. HALP also satisfies Selective revocation (F3) through implementation of nullifier registries and Merkle proofs. Scalability (F4) and minimal client overhead (F5) were achieved through efficient ZK-SNARK verification and WebAssembly-based lightweight clients. Security requirements such as integrity (S1), anonymity (S2), and revocation with accountability (S3) are satisfied through BBS+ signatures, Poseidon hashing, and tamper-evident registry designs. The system also adheres to decentralized identity standards (W3C VCs, DIDs), ensuring interoperability and real-world applicability.

5.5 Limitations

Despite its strengths, the HALP authentication system has its limitations. The trusted setup requirement for Groth16 ZK-SNARKs introduces a potential security risk and usability hurdle. Although proof generation times are below one second, it may still be non-trivial for resource-constrained devices. Although the Merkle tree-based nullifier registry is scalable, it may face challenges in fully decentralized environments without blockchain integration. Furthermore, the system's dependence on advanced tools like BBS+, Poseidon and Merkle trees increases implementation complexity and audit burden. Future work should focus on these limitations by opting for transparent setup alternatives, proof aggregation, and enhanced registry decentralization.

Chapter 6

Conclusion

6.1 Conclusion

HALP provides a new hybrid architecture, combining ZK-SNARK, BBS+ signatures, poseidon hash pseudonyms and nullifiers and an indexed merkle tree based nullifier registry for selective disclosure, anonymity, unlinkability and revocation. It provides efficient identity binding and the flexibility of attribute disclosure in one authentication system. The witness hiding property ensures that adversaries cannot link different sessions, while the soundness of proofs and computational biding property of commitments ensures unforgeability.

It can be seen in the implementation that privacy preserving authentication can be realized without compromising the performance or interoperability. HALP conforms to current identity ecosystems via the use of W3C verifiable credentials 2.0 and decentralized identifiers. It is also more privacy assuring than traditional authentication systems.

6.2 Future Work

There are various areas in which improvements can be made in future work. Implementing the nullifier registry on a blockchain through smart contracts would for allow decentralized revocation without requiring operators. The master secret may also be further secured by including hardware security modules. Recurrent SNARK composition may decrease proof generation time and ease proof generation for resource bound devices. Lastly, the verification of the circom circuit using automated provers would enhance trust in the accuracy of the implementation.

Bibliography

[1] S. Goldwasser, S. Micali, and C. Rackoff, "The knowledge complexity of interactive proof-systems," in *Proc. 17th Annu. ACM Symp. Theory Comput. (STOC)*, Dec. 1985, pp. 291–304. DOI: 10.1145/22145.22178.

[2] T. P. Pedersen, "Non-interactive and information-theoretic secure verifiable secret sharing," in *Advances in Cryptology – CRYPTO '91*, ser. Lecture Notes in Computer Science, vol. 576, Springer, 1991, pp. 129–140. DOI: 10.1007/3-540-46766-1_9.

[3] S. Halevi and S. Micali, "Practical and provably-secure commitment schemes from collision-free hashing," in *Advances in Cryptology—CRYPTO '96*, ser. Lecture Notes in Computer Science, vol. 1109, Springer, 1996, pp. 201–215. DOI: 10.1007/3-540-68697-5_16.

[4] S. Brands, *Rethinking Public Key Infrastructures and Digital Certificates: Building in Privacy*. Cambridge, MA, USA: MIT Press, 2000.

[5] J. Camenisch and A. Lysyanskaya, "An efficient system for non-transferable anonymous credentials with optional anonymity revocation," in *Advances in Cryptology—EUROCRYPT*, vol. 2045, Springer, 2001, pp. 93–118. DOI: 10.1007/3-540-44987-6_7.

[6] J. Camenisch and A. Lysyanskaya, "Design and implementation of the idemix anonymous credential system," in *Proceedings of the 2003 RSA Conference*, RSA Conference, 2003, pp. 1–20.

[7] S. Steinbrecher and S. Köpsell, "Modelling unlinkability," in *Third International Workshop on Privacy Enhancing Technologies (PET 2003)*, Dresden, Germany, Mar. 2003.

[8] D. Boneh and X. Boyen, "Short signatures without random oracles and the q-strong diffie-hellman problem," in *Advances in Cryptology – EUROCRYPT 2004*, Springer, 2004, pp. 56–73. DOI: 10.1007/978-3-540-24676-3_5.

[9] G. Barthe, B. Grégoire, S. Heraud, and S. Z. Béguelin, "Computer-aided security proofs for the working cryptographer," in *Advances in Cryptology—CRYPTO 2011*, ser. Lecture Notes in Computer Science, vol. 6841, Springer, 2011, pp. 71–90. DOI: 10.1007/978-3-642-22792-9_5.

[10] C. Paquin, *U-Prove WS-Trust Profile V1.0*, Draft Revision 1, Microsoft Corporation, Feb. 2011.

[11] G. Barthe, G. Danezis, B. Grégoire, C. Kunz, and S. Zanella-Béguelin, "Verified computational differential privacy with applications to smart metering," in *2013 IEEE 26th Computer Security Foundations Symposium*, IEEE, 2013, pp. 287–301. DOI: 10.1109/CSF.2013.26.

[12] I. Miers, C. Garman, M. Green, and A. D. Rubin, "Zerocoin: Anonymous distributed e-cash from bitcoin," in *IEEE Symposium on Security and Privacy*, IEEE, 2013.

[13] P. Vullers and G. Alpár, "Efficient selective disclosure on smart cards using idemix," in *Policies and Research in Identity Management*, Springer Berlin Heidelberg, 2013, pp. 1–14. DOI: 10.1007/978-3-642-37282-7_5.

[14] E. Ben-Sasson *et al.*, "Zerocash: Decentralized anonymous payments from bitcoin," in *IEEE Symposium on Security and Privacy*, IEEE, 2014.

[15] E. Ben-Sasson, A. Chiesa, E. Tromer, and M. Virza, "Succinct Non-Interactive zero knowledge for a von neumann architecture," in *23rd USENIX Secur. Symp.*, USENIX Association, Aug. 2014, pp. 781–796. [Online]. Available: https://www.usenix.org/conference/usenixsecurity14/technical-sessions/presentation/ben-sasson.

[16] M. Chase and S. Meiklejohn, "Transparency overlays and applications," in *Proc. 2014 ACM SIGSAC Conf. on Computer and Communications Security (CCS '14)*, ACM, 2014.

[17] C. Paquin, "On the revocation of U-Prove tokens," Microsoft Research, Tech. Rep. MSR-TR-2014-122, Sep. 2014. [Online]. Available: https://www.microsoft.com/en-us/research/publication/on-the-revocation-of-u-prove-tokens/.

[18] A. Azaria, A. Ekblaw, T. Vieira, and A. Lippman, "Medrec: Using blockchain for medical data access and permission management," in *Proc. 2nd Int. Conf. Open Big Data (OBD)*, 2016, pp. 25–30. DOI: 10.1109/OBD.2016.11.

[19] J. Groth, "On the size of pairing-based non-interactive arguments," in *EUROCRYPT 2016*, Springer, 2016, pp. 305–326. DOI: 10.1007/978-3-662-49890-3_12.

[20] A. Kosba, A. Miller, E. Shi, Z. Wen, and C. Papamanthou, "Hawk: The blockchain model of cryptography and privacy-preserving smart contracts," in *Proc. IEEE Symp. Secur. Privacy (SP)*, 2016, pp. 839–858. DOI: 10.1109/SP.2016.55.

[21] S. Bowe, *BLS12-381: New zk-SNARK elliptic curve construction*, Electric Coin Company Blog, (accessed Feb. 4, 2026), Jan. 2017. [Online]. Available: https://electriccoin.co/blog/new-snark-curve/.

[22] B. Bunz, J. Bootle, D. Boneh, A. Poelstra, P. Wuille, and G. Maxwell, "Bulletproofs: Short proofs for confidential transactions and more," Cryptology ePrint Archive, Report 2017/1066, Tech. Rep., 2018. [Online]. Available: https://eprint.iacr.org/2017/1066.

[23] B. Edgington, *BLS12-381 for the rest of us*, HackMD, (accessed Feb. 4, 2026), 2019. [Online]. Available: https://hackmd.io/@benjaminion/bls12-381.

[24] A. Gabizon, Z. Williamson, and O. Ciobotaru, "PLONK: Permutations over lagrange-bases for oecumenical noninteractive arguments of knowledge," Cryptology ePrint Archive, Tech. Rep. 2019/953, 2019. [Online]. Available: https://eprint.iacr.org/2019/953.

[25] A. Sonnino, M. Al-Bassam, S. Bano, and G. Danezis, "Coconut: Threshold issuance selective disclosure credentials with applications to distributed ledgers," in *Proc. Netw. Distrib. Syst. Secur. Symp. (NDSS)*, 2019. DOI: 10.14722/ndss.2019.23272.

[26] V. Buterin, J. Woods, and A. Mavridou, "Aztec: Privacy-preserving smart contracts using indexed merkle trees," in *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security (CCS '20)*, ACM, 2020, pp. 1009–1023. DOI: 10.1145/3372297.3423353.

[27] Z. Foundation, *Moonmath manual*, Available at https://github.com/zcash/moonmath, 2020.

[28] K. Gurkan, B. Whitehat, and K. W. Jie, *Community proposal: Semaphore: Zero-knowledge signaling on ethereum*, ZKProof Community Workshop 3 Proposal, (accessed Feb. 4, 2026), Mar. 2020. [Online]. Available: https://docs.zkproof.org/pages/standards/accepted-workshop3/proposal-semaphore.pdf.

[29] IETF CFRG, *Pairing-friendly curves*, Internet Draft, IRTF Crypto Forum Research Group, Jun. 2020. [Online]. Available: https://datatracker.ietf.org/doc/html/draft-irtf-cfrg-pairing-friendly-curves-10.

[30] L. Grassi, D. Khovratovich, C. Rechberger, A. Roy, and M. Schofnegger, "Poseidon: A new hash function for zero-knowledge proof systems," in *30th USENIX Security Symposium (USENIX Security 21)*, USENIX Association, 2021, pp. 519–535. [Online]. Available: https://www.usenix.org/conference/usenixsecurity21/presentation/grassi.

[31] E. C. Company and Z. Foundation, *Zcash protocol specification, version 2022.3.8*, https://zips.z.cash/protocol/protocol.pdf, (accessed Feb. 4, 2026), 2022.

[32] W3C Credentials Community Group, *Decentralized identifiers (dids) v1.0: Core architecture, data model, and representations*, https://www.w3.org/TR/did-core/, May 2022.

[33] Z. Abuabed, A. Alsadeh, and A. Taweel, "STRIDE threat model-based framework for assessing the vulnerabilities of modern vehicles," *Computers & Security*, vol. 133, p. 103 391, 2023. DOI: 10.1016/j.cose.2023.103391.

[34] J. Doerner, Y. Kondi, E. Lee, A. Shelat, and L. Tyner, "Threshold bbs+ signatures for distributed anonymous credential issuance," in *2023 IEEE Symposium on Security and Privacy (SP)*, IEEE, 2023, pp. 773–789. DOI: 10.1109/SP46215.2023.10179470.

[35] S. Tessaro and C. Zhu, "Revisiting BBS signatures," in *Advances in Cryptology – CRYPTO 2023*, ser. Lecture Notes in Computer Science, vol. 14084, Cham: Springer, 2023, pp. 103–133. DOI: 10.1007/978-3-031-38551-3_4.

[36] F. Baldimtsi *et al.*, "Zklogin: Privacy-preserving blockchain authentication with existing credentials," in *Proceedings of the 2024 ACM SIGSAC Conference on Computer and Communications Security (CCS)*, 2024. DOI: 10.1145/3658644.3690356.

[37] Jaromil, *Benchmark of the bbs signature*, Dyne.org blog, (accessed Feb. 4, 2026), Aug. 2024. [Online]. Available: https://news.dyne.org/benchmark-of-the-bbs-signature-scheme-v06/.

[38] A. S. Rajasekaran, A. Maria, J. Lloret, and S. Dannana, "TPAAS: Trustworthy privacy-preserving anonymous authentication scheme for online trading environment," *PLOS ONE*, vol. 19, no. 11, e0307738, Nov. 2024. DOI: 10.1371/journal.pone.0307738.

[39] D. Roio, A. Aloisio, A. Bria, and E. Piana, *SD-BLS: Privacy preserving selective disclosure of verifiable credentials with unlinkable threshold revocation*, Cryptology ePrint Archive, Paper 2024/673, Aug. 2024. DOI: 10.48550/arXiv.2406.19035. [Online]. Available: https://eprint.iacr.org/2024/673.

[40] O. Sanders and J. Traoré, "Compact issuer-hiding authentication: Application to anonymous credentials," *Proceedings on Privacy Enhancing Technologies*, vol. 2024, no. 3, pp. 645–658, 2024. DOI: 10.2478/popets-2024-0097.

[41] X. Zhao, F. Xia, H. Xia, Y. Mao, and S. Chen, "A zero-knowledge-proof-based anonymous and revocable scheme for cross-domain authentication," *Electronics*, vol. 13, no. 14, p. 2730, 2024. DOI: 10.3390/electronics13142730.

[42] D. M. B. Mbimbi and M. Wilson, "Preserving whistleblower anonymity through zero-knowledge proofs and private blockchain: A secure digital evidence management framework," *Blockchains*, vol. 3, no. 2, p. 7, 2025. DOI: 10.3390/blockchains3020007.

[43] W. V. C. W. Group, "Verifiable credentials data model v2.0," World Wide Web Consortium (W3C), Tech. Rep., May 2025. [Online]. Available: https://www.w3.org/TR/2025/REC-vc-data-model-2.0-20250515/.

[44] F. Hoops, J. Gebele, and F. Matthes, "CRSet: Private non-interactive verifiable credential revocation," arXiv, Tech. Rep., Sep. 2025, arXiv:2501.17089v3 [cs.CR]. DOI: 10.48550/arXiv.2501.17089. [Online]. Available: https://arxiv.org/abs/2501.17089v3.

[45] R. Khan *et al.*, "STRIDE-based threat modeling and risk assessment framework for IoT-enabled smart healthcare systems," *Int. J. Online Biomed. Eng. (iJOE)*, vol. 21, no. 09, pp. 63–80, Jul. 2025. DOI: 10.3991/ijoe.v21i09.55517.

[46] The Hyperledger Foundation, *AnonCreds Specification*, GitHub repository, (accessed Feb. 4, 2026), 2025. [Online]. Available: https://anoncreds.github.io/anoncreds-spec/.

[47] C. Udokwu, "Zero knowledge proof solutions to linkability problems in blockchain-based collaboration systems," *Mathematics*, vol. 13, no. 15, p. 2387, 2025. DOI: 10.3390/math13152387.

[48] zkDocs, *Schnorr's identification protocol*, https://www.zkdocs.com/docs/zkdocs/zero-knowledge-protocols/schnorr/, (accessed Feb. 4, 2026), 2025.

[49] R. Chairattana-Apirom and S. Tessaro, "On the concrete security of bbs/bbs+ signatures," in *Advances in Cryptology – ASIACRYPT 2025*, G. Hanaoka and B.-Y. Yang, Eds., ser. Lecture Notes in Computer Science, vol. 16250, Singapore: Springer, 2026, p. 6298. DOI: 10.1007/978-981-95-5119-4_13.

[50] T. Looker, V. Kalos, A. Whitehead, and M. Lodder, "The BBS signature scheme," Internet Research Task Force (IRTF), Tech. Rep., Jan. 2026. [Online]. Available: https://datatracker.ietf.org/doc/draft-irtf-cfrg-bbs-signatures-10/.

[51] Semaphore Protocol Documentation, *What is semaphore?* https://docs.semaphore.pse.dev/, Official Semaphore documentation, 2026. [Online]. Available: https://docs.semaphore.pse.dev/.

[52] K. George, *The mathematical mechanics behind the Groth16 zero-knowledge proof system*, [Online]. Available: https://kayleegeorge.github.io/math110_WIM.pdf, (accessed Feb. 4, 2026).

[53] IDEN3, *Circom and SnarkJS documentation*, https://docs.circom.io, (accessed Feb. 4, 2026).