# Integrating Single-Sign-On Within the WebAuthn Framework

by

Asir Adnan
23341026
Nafisha Tabassum Anika
22101554
Saima Sobahan
22101109
A.J.M Istiaque
23341065

Final 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

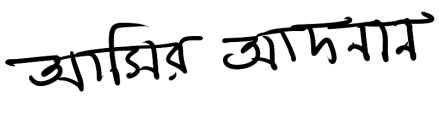
Department of Computer Science and Engineering
Brac University
January 2026

# Declaration

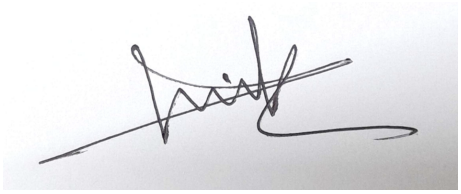
It is hereby declared that

1. The thesis submitted is 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.

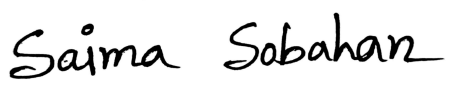
**Student's Full Name & Signature:**



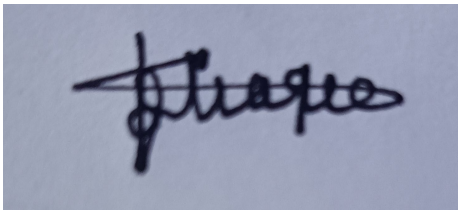
Asir Adnan
23341026



Nafisha Tabassum Anika
22101554



Saima Sobahan
22101109



A.J.M Istiaque
23341065

# Approval

The thesis titled "Integrating Single-Sign-On Within the WebAuthn Framework" submitted by

1. Asir Adnan(23341026)
2. Nafisha Tabassum Anika(22101554)
3. Saima Sobahan(22101109)
4. A.J.M Istiaque(23341065)

Of Fall, 2025 has been accepted as satisfactory in partial fulfillment of the requirement for the degree of B.Sc. in Computer Science on January 28, 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 Engineering
Brac University

Head of Department:
(Chair)

Sadia Hamid Kazi, PhD
Chairperson and Associate Professor
Department of Computer Science and Engineering
Brac University

# Abstract

In the world of digital identity, preserving user privacy while maintaining seamless access across platforms has become a challenge. WebAuthn, developed by World Wide Web Consortium (W3C) is mainly a web-based authentication standard. This system enhances security by enabling passwordless login through hardware-based and biometric authentication mechanisms. Another popular approach, to simplify authentication for users across the internet is Single-Sign-On (SSO) which allows a single credential to access multiple services or applications. This way users can get rid of the liability to manage multiple credentials, rather they can rely on only one credential to authenticate in a trusted manner and use that to authenticate in many other websites. Despite the potential of the SSO system, it has not been integrated with the WebAuthn framework till date. Through our research work, we have introduced a system that ensures passwordless authentication via WebAuthn and supports seamless access to service providers through SSO eliminating the requirements of repeated login. Moreover, this system empowers users with the full control over sharing their personal information by selective disclosure mechanism. Security Assertion Markup language (SAML) is used as the federated identity to exchange the authentication assertion securely between identity providers and service providers to enable seamless SSO. Thereby, introducing a new horizon of research on WebAuthn and SSO.

**Keywords:** SSO, Webauthn, SAML, Selective Disclosure, FIDO2, CTAP, W3C, Authentication

# Table of Contents

# List of Figures

# List of Tables

# Nomenclature

The next list describes several symbols & abbreviation that will be later used within the body of the document

2FA two-factor authentication

CTAP Client to Authenticator Protocol

FIDO2 Fast Identity Online 2

JSON JavaScript Object Notation

JWT JSON Web Token

SAML Security Assertion Markup Language

SSO Single Sign-On

W3C World Wide Web Consortium

WebAuthn Web Authentication

# Chapter 1

# Introduction

In the modern digitalized world, users use numerous online services based on which there should be a secure and suitable authentication. Conventional password systems still remain a problem because of the reuse of passwords, phishing and credential disclosure. The introduction of modern systems like WebAuthn has provided powerful passwordless authentication with public-key cryptography and hardware-bound credentials, which has given great protection against phishing and theft of personal identities [29].

In the meantime, it has been shown that Single Sign-On (SSO) standards[14], in particular, SAML 2.0 [22], enable users to authenticate only once and access numerous applications, enhancing usability and decreasing credential fatigue. Nevertheless, passwords continue to play a significant role in SAML-based SSO, which exposes the Identity Provider (IdP) to vulnerability. Also, existing SSO systems provide minimal support in selective disclosure and users are often expected to disclose entire groups of identity attributes rather than just the required ones- an issue of privacy concern. The convergence of WebAuthn phishing-resistant authentication with the SAML's federated identity model and attribute-level selective disclosure is an open area of exploration of the implementation of secure, privacy-enhancing, and seamless access to trusted domains.

The current chapter outlines the background study, the significance and relevance of the research gap and its impact, the specific issue the research addresses, the focus of the research, the boundaries of the study and any constraints. It also gives a overview of how the research was conducted.

## 1.1 Background

Authentication stands as a fundamental front-line defense to secure digital service access while safeguarding sensitive web information. Traditional password authorization systems continue to struggle because users face continuous issues with security breaches and repeated passwords and the necessity of managing multiple accounts' logins [31]. Complex solutions that combine Web Authentication (WebAuthn) and Single-Sign-On (SSO) emerge as revolutionary implementations for resolving existing security challenges. Both WebAuthn and SSO work to protect users but their combination represents an underutilized approach that shows great promise [31].
WebAuthn which came from the World Wide Web Consortium (W3C) enables

modern authentication through passwordless connections [20]. The framework uses hardware-based authenticators in combination with biometrics and public-key cryptography to deliver both security and usability beyond password authentication [20]. WebAuthn removes the requirement of password secrets therefore safeguarding against phishing scams and credential theft. At present WebAuthn primarily targets authentication of individual domains but struggles to scale across various services [20].
The authentication system Single-Sign-On (SSO) provides users access to several applications and services by using only one set of login details. When authentication functions from different applications integrate into a single trusted framework SSO creates better user experiences while simplifying the management of multiple credentials. Although SSO systems deliver helpful benefits they depend on conventional password authentication which makes them vulnerable to security threats. Security assertion markup language (SAML) facilitates the concept of single sign-on (SSO) that enables a user to authenticate just once with an identity provider and subsequently access numerous service providers without authentication again [2]. The unification of WebAuthn authentication technology with single sign-on platforms establishes new innovative possibilities for authentication systems evolution. The combination of WebAuthn's passwordless security features with SSO convenience enables us to build a robust authentication network that provides simple accurate authentication across trusted application domains [20] [31]. The proposed system resolves both technologies' current limitations by creating one unified authentication methodology. Another improvement to this intregated WebAuthn-SSO system is the selective disclosure, where the user may disclose only certain data of their choice to various services rather than shoaring their all attributes [11]. The inclusion of this tool in the system under consideration will allow minimizing the likelihood of exposure to data breaches and identity abuse, and give users more control over their personal data on a variety of levels without undermining the convenience and security advantages of passwordless SSO. This research studies the possible implementation of single sign-on functionality in WebAuthn to determine its architectural potential. The research examines core approaches to achieve safe and seamless identity verification throughout various domains combined with reliable passwordless authentication capability. This research delivers essential findings for creating an authentication framework that balances user comfort with maximum security [31].

## 1.2 Problem Statement

Traditional password authentication models are not sufficient because users face security threats from repeated password use and phishing attacks [31]. WebAuthn implements passwordless security that authenticates users through hardware components. However, its implementation is limited to a single platform, restricting cross-platform usability. This is inconvenient and will guide many users back to the old passwords, and the whole idea of using WebAuthn will be nullified.
On the other hand, SSO simplifies access across multiple services but remains vulnerable due to its reliance on traditional passwords, which can be compromised through attacks like credential stuffing and brute-force attempts [25][30]. This system in mostly unsafe at organizational environments where users use SSO to access tools like Slack, Google Workspace and so on. If an attacker somehow manages to

phish the SSO login, will get access into all linked systems with a single breach. Moreover, even the modern authentication systems, such as the ones based on high-tech applications like WebAuthn or Single Sign-On (SSO), usually demand that users disclose all the personal information regarding their identity to the service providers. This practice puts users at a risk of privacy such as unwarranted data disclosure, identity theft and possible abuse of personal information. The existing models do not have mechanisms through which users can manage the level of the information they reveal to various services. In the absence of selective disclosure, users will trade off authentication convenience with privacy protection. There is a distinct necessity of an authentication methodology that can allow verification to be conducted in a secure and passwordless manner, along with being capable of allowing users to only selectively disclose the required identity attributes to improve the privacy without impacting the usability or security.
As an example, most institutions have SSO login system for various services like email, course portals, library systems and so on. In that case, there is a high chance to be a victim of phishing emails which might bypass the credentials and attacker may get access into all other platforms. Although institutions may implement WebAuthn as an additional security measure, often such implementation is restricted to the devices managed by the institution, not to the personal devices owned by users which again reducing the efficiency of WebAuthn and sending users back to using conventional password system.

## 1.3 Research Motivation

Currently, there exists no framework which utilizes both the passwordless security features of WebAuthn and the convenience of cross-platform authentication with single credential of SSO while also supporting selective disclosure of user attributes. The failure to combine these approaches blocks the development of a unified solution that would provide both scalability, security and privacy in authentication systems.

Considering integrating SSO with WebAuthn, authentication system will be significantly better. This integrated system will enable user accounts to log into multiple websites and applications without any password, rather relying on proven public-key cryptography and long-term identity verification. Moreover, the integration will prevent on the password reuse, make phishing and credential stuffing attacks less successful, and enable cross-platform usability.

With the growth of digital services, user authentication is becoming more and more an essential need with security and seamlessness as paramount requirements. Although SAML (Single Sign-On)-based systems facilitate access control in various areas, they tend to be anchored on the conventional use of passwords, which means that user identities can be easily compromised, exposing such individuals to breach of identities and more overexposure to personal information. At the same time, new technologies such as WebAuthn are passwordless authenticators but do not have any policies to control the identity disclosure between different services. By adding selective disclosure to a SAML-based SSO system, these obstacles can be overcome to allow users to transmit only the minimal required identity attributes to each service, improving the privacy of the system without losing the convenience and security of

federated authentication. This drives the need to explore an integrated framework where SAML, WebAuthn, and selective disclosure are integrated together to provide an effective, privacy-aware, and user-focused authentication process.

To illustrate, in an organization where users must use SSO to work with the tools like Google Workspace, internal dashboard, and other enterprise software, existing systems tend to continue using traditional methods that use passwords to authenticate users. Combining WebAuthn and SSO system does not require any passwords, instead, users can use the biometric keys or hardware tokens to authenticate and, in fact, phishing is impossible as there is no secret to crack. Also, the system can tolerate selective disclosure to guarantee that the employees only provide the bare minimum of identity information to every service in order to preserve confidential personal or organizational data and still have access to services without any problems. Such a solution also enhances privacy, compliance, and user trust in various applications in the organization besides ensuring enhanced security and minimizing the chances of a credential theft.

## 1.4 Objectives

The objectives of this research are presented below:

RO1. **State of Art:** Explore the existing SSO technologies along with establishing approaches to unite WebAuthn features with SSO.

RO2. **Threat Modelling:** A thorough examination of threats will produce this new infrastructure's security risk assessment which includes both threat identification and risk analysis.

RO3. **Requirement Analysis:** Defining functional, security and privacy requirements in order to help design and implement an efficient and secure system.

RO4. **System architecture and protocols:** The proposal outlines a new architecture design for integrating WebAuthn with SSO, enabling passwordless user identity verification. This integration will be implemented through framework development, ensuring seamless authentication within the SSO system. The protocol design will define secure interactions between users, authenticators, and identity providers.

RO5. **Testing and Evaluation:** The proposed infrastructure performance assessment includes comprehensive examination to prove its operational feasibility together with practicality and effectiveness.

## 1.5 Methodology in Brief

The study presents a safe and scalable authentication system based on adopting Single Sign-On (SSO) and SAML. It aims to support passwordless authentication, hardware-based and biometric authentication, and, in this way, reduce vulnerabilities, including phishing and credential theft. The research methodology will start with a literature review to determine the existing technologies and gaps. Threat

modeling was carried out to measure the possible security threats and the functional and security requirements analysis was done to determine the requirements of the system. These findings were used to build a data model and system architecture, as well as particular algorithms and protocol flows which are safe to execute. The framework utilizes the power of public-key cryptography to authenticate user identities between domains of trust, which offers a user-friendly and cross-device seamless solution.
Also included in the framework is the federated identity management over SAML to support the use of secure Single Sign-On in several applications and service providers, as well as selective disclosure to allow users to disclose only the required attributes of identity to each service. The use cases were used to find out the weak points and make sure that the system achieves its goals of high security, privacy, and convenience.
The study presents a safe and scalable authentication system based on adopting Single Sign-On (SSO) and SAML. It aims to support passwordless authentication, hardware-based and biometric authentication, and, in this way, reduce vulnerabilities, including phishing and credential theft. The research methodology will start with a literature review to determine the existing technologies and gaps. Threat modeling was carried out to measure the possible security threats and the functional and security requirements analysis was done to determine the requirements of the system. These findings were used to build a data model and system architecture, as well as particular algorithms and protocol flows which are safe to execute. The framework utilizes the power of public-key cryptography to authenticate user identities between domains of trust, which offers a user-friendly and cross-device seamless solution.
Also included in the framework is the federated identity management over SAML to support the use of secure Single Sign-On in several applications and service providers, as well as selective disclosure to allow users to disclose only the required attributes of identity to each service. The use cases were used to find out the weak points and make sure that the system achieves its goals of high security, privacy, and convenience.

## 1.6 Scopes and Challenges

The scope of this research is focused mostly on user authentication. The proposed system will offer a passwordless authentication system where the user can identify himself or herself through biometric data or hardware authenticators with high security. Although authentication is completely discussed, the part of authorization a process that defines what kind of access a user may get, i.e., whether administrative access or access to certain resources, is not considered in the frame of the current study. The host applications or service providers are left to do these access decisions and grant or deny access depending on the authenticated identity and role. That is to say, the system will authenticate the user and give them the permission to do whatever they want, but does not determine what a user may do after authentication.
There are a number of issues that affect the construction of the system. It is automatically challenging to deliver an SSO experience across several subdomains or service providers due to the fact that WebAuthn does not acquire cross-origin au-

thentication by default to achieve a higher level of security. To obtain interoperability, SAML-based federated identity management has to be incorporated, but the adjustments of the protocol have to be done cautiously, as well as secure trust-sharing processes between service providers. The other critical issue is accessibility because WebAuthn requires hardware-based authenticators (e.g., biometric devices or security keys e.g. YubiKey), which may be expensive and not accessible to everyone, especially in developing countries or non-technical users. This brings about a usability gap which should be taken into account during the system design.
Furthermore, selective disclosure to support privacy is also introduced which further complicates the matter. Whereas selective disclosure should enable users to reveal only the most basic identities attributes to each service, it has to be carefully considered when applied to multiple domains based on SAML to ensure both security and usability. All these issues influenced the design choice of the architecture, data model, and protocol flows that are to be presented in the following design and implementation section.

## 1.7 Report structure

The structure of this report is as follows:

- **Chapter 2** gives preliminaries of the study which includes the following major concepts: WebAuthn, Single Sign-On (SSO), SAML and selective disclosure. It also features an extensive and critical review of the existing literature that places the proposed work in the context of the current research on federated identity, privacy protection, and authentication security.

- **Chapter 3** presents the design requirements and specifications of the system. This consists of functional, security, and privacy requirements that are required to provide passwordless SSO with attribute-level disclosure.

- **Chapter 4** provides the description of the proposed system. It explains the architecture, protocol flow, data model, threats, and algorithm. The chapter also provides the implementation process.

- **Chapter 5** aims to describe performance evaluation. It describes the development environment and discusses the behavior of the system to different loads, and also practical limitations.

- **Chapter 6** provides the conclusion of the research, which summarizes the main findings, contributions and provides future research directions to enhance usability and scalability.

# Chapter 2

# Literature Review

The current chapter provides essential information to portray theories, definitions, concepts and models of WebAuthn, SSO, SAML and Selective disclosure. It is followed by a literature survey. The analysis of earlier studies helps frame the challenges explored in this work and offers the theoretical grounding needed to support the proposed approach.

## 2.1 Preliminaries

### 2.1.1 Authentication

Authentication refers to a process that validates a user [13]. It verifies user's devices and systems to determine their genuine identity. Traditionally, people use the basic username-password authentication process. There are different types of traditional password-based authentication. Commonly used authentications are two-factor authentication (2FA), token hardware authentication, OTP authentication, knowledge-based authentication (security key), etc [13]. These authentications are low-cost and easy to use. Users need to remember those passwords for multiple applications and devices. But the problem arises when the passwords get leaked or hacked and there is no way to verify if the given password is coming from the real user or hacker. This is why modern authentication technologies adopt a passwordless policy, which removes the requirement for traditional password authentication. The technology implements authentication through biometrics systems and one-time code authentication protocols and public-key cryptographic procedures. Users embrace passwordless authentication because the technique offers both security benefits and convenience by preventing standard password vulnerabilities. The passwordless authentication model includes WebAuthn systems for secure authentication and one-time access codes delivered by email and SMS.

### 2.1.2 WebAuthn

WebAuthn (Web Authentication) is an API standard that provides a secured passwordless authentication through hardware and software authenticators including security keys and integrated device authenticator [9]. Asymmetric cryptography enables a system where the user has to store a private key on their device and link a public key to the server database. When users log in, the server authenticates users

identity through their public key. Through WebAuthn, users enjoy a secure authentication method that surpasses passwords in protection and matches user's needs better than traditional password systems.WebAuthn was launched by the FIDO Alliance and W3C in 2013 and received its official W3C standard in 2019. WebAuthn serves big tech platforms Google, Facebook and Microsoft through numerous browser options such as Chrome, Firefox, Edge and Safari to enhance security features and usability. The FIDO Alliance started the journey in around 2013 to lead the development of open standards that eliminate password dependence [10]. FIDO2 represents the newest FIDO standard that merges WebAuthn and CTAP2(Client to Authenticator Protocol) for delivering second-factor and passwordless and multi-factor authentication that can work across multiple hardware platforms. WebAuthn involves three key components: Relying party, Client-Platform and Authenticators(figure 2.1). A relying party refers to a website, service, or application which will authenticate users. A client platform indicates the software and hardware which are used by a user to authenticate. An authenticator takes the user's input and authenticates the user's credential for the relying party. Authenticators can be an internal authenticator like built-in biometric systems or an external roaming authenticator like hardware security keys. The authentication process starts with user registration where the user will enter only an username, not a password and create an account. Relying party will send a cryptographic challenge to the user's platform but the user won't be able to see it. The authenticator will create a pair of key, public and private key according to the cryptographic challenge. Then the private key will be stored to the user and the public key will be sent to the relying party. Once the registration is done, a user can easily authenticate with that authenticator. User will enter his/her registered username to the relying party. Relying party will create a new challenge using the user's public key. User's authenticator then will send a signed attestation using the private key. Then the relying party will decide if the user's authentication is valid or not [9].

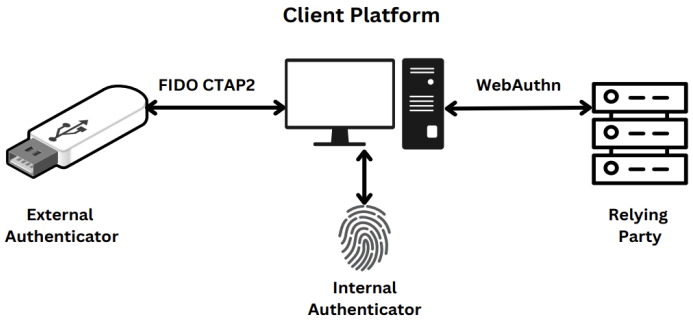



Figure 2.1: WebAuthn working mechanism

### 2.1.3 SSO(Single Sign On)

Single Sign-On (SSO) operates as an authentication framework that permits users to establish one login connection which enables entry to multiple software systems or services eliminating repeated credentials input. This system offers safety advance-

ments through decreased password fatigue, thus decreasing both password-related dangers and the consequences of credential repetition.
Figure 2.2 shows how SSO works, where a user authenticates once by personal information and then can access to different website or application easily without repetitive input. A central authentication server is used for SSO, which can verify a user's identity. The server provides a secure token to authorized applications when a user logs in. The trust of these applications is in the token which they use to access without another login process. SSO's common technologies are SAML (Security Assertion Markup Language), OAuth and OpenID Connect. At the beginning, Lightweight Directory Access Protocol (LDAP) was one of the earliest solutions which was introduced in the 1990s. LDAP made it easy for organizations to store and manage the user credentials for their organization in a centralized directory and therefore reduced the overhead when authenticating the user. Microsoft Active Directory (AD) was another key advancement on LDAP, basing the authentication and authorization services on enterprise networks. Nevertheless, these solutions still required the user to log in separately to individual applications.The challenge is therefore addressed by the Kerberos authentication protocol, developed in the 1980s at MIT [33]. With Kerberos, a password was used with a ticket based system, the user logs in once and receives a ticket granting ticket (TGT). Bazaz et al. [6] mentioned this Kerberos protocol makes use of Key Distribution Centre (KDC) as the server and by using secret key cryptography for client/server applications it provides provides strong token based authentication [34].The purpose of this ticket was to gain access to multiple services without the need to input credentials each time. Microsoft later adopted Kerberos for Windows authentication.
Nowadays, SSO is used in many enterprises and cloud-based platforms as well as social logins (i.e. Google, Microsoft and Facebook authentication). Moreover, companies integrate SSO with MFA so that it adds a second level of security (additional verification – OTP or biometric authentication). Indeed, SSO security will be enhanced in future terms by passwordless authentication based on biometrics, security keys, and AI backed risk analysis.

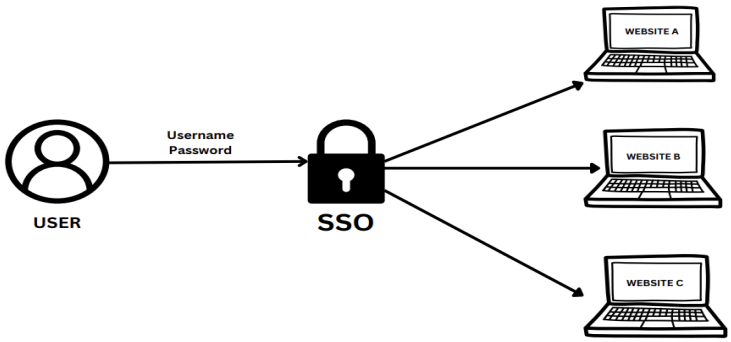



Figure 2.2: SSO Working mechanism

### 2.1.4 SAML

SAML (Security assertion Markup language) is an open standard providing security data (authentication and authorization data) exchange between parties in a network.

In particular, it allows a secure communication between an identity provider, which controls user credentials and user authentication, and a service provider, which serves as a host to the applications or services that a user desires to use. SAML is basically an XML-driven model which enables organizations to adopt Single Sign-On (SSO) systems, thus enabling users to access a variety of applications using a single set of credentials.
The mechanism of SAML is fairly simple as a user, but the underlying technology is rather advanced [3]. Once a user tries to access any service or application, he or she is redirected to an identity provider belonging to his or her organization to authenticate. After credentials of the user are verified by the identity provider, a SAML assertion is generated, which is simply a digitally signed XML document that holds information about the user identity and access permissions. This statement is then returned to the service provider which confirms it and gives the user access to the requested resource[5]. The user will not be required to type their credentials into the system of the service provider and this goes a long way in increasing the security.
SAML finds application in the enterprise and most educational institutions. Examples of these are logging into Google Workspace [1] or any other system where access to Gmail, Drive, Calendar, and other functions is automatically provided once one is initially authenticated, or university systems where students use their school credentials to access library databases, learning management systems, email, and other campus-provided services. SAML is an authentication method employed by many corporate settings where employees can now use dozens of various applications and services with a single log-in and do not need to remember dozens of passwords and the IT support departments no longer have to deal with the volume of calls.
The major advantages of SAML are that the system enhances the convenience of users, increases security, and provides centralized access control. The user only has to memorize a single set of credentials and authenticate after every session, and password fatigue is decreased, as well as the potential of using a weak or reused password. Security wise, SAML implies that service providers do not access or store user passwords, which minimizes the attacker surface and the possibility of stealing the credential. Also, the IT administrators will be able to control user access centrally and it is easy to revoke or grant access to a multiple systems at once. Although SAML continues to be popular, and may be the most popular in the enterprise world, newer standards including OAuth 2.0 and OpenID Connect have appeared, particularly in web and mobile software.

### 2.1.5 Selective Disclosure

Selective disclosure is a privacy enhancing approach to the digital identity system that lets the user demonstrate certain attributes of a credential without exposing the entire credential or unneeded personal information. This reduces the exposure of data and is still verifiable, frequently via cryptographic techniques such as zero knowledge inspiration or special signings such as BBS+ or BLS [12].
In selective disclosure, an issuer places on a piece of credential multiple claims, however, the holder can selectively reveal only pertinent claims to a verifier, which produces a proof that the disclosed portions are legitimate and have not been altered. As an example, to demonstrate that one is over 18, a user sends only an age predicate

without revealing the exact birthdate, and this is done by methods such as claim redaction or zero-knowledge proofs. This eliminates excessive sharing and minimizes risks.

## 2.2 Review of Existing Research

Louis Jannett et al. [31] present *SoK: SSO-MONITOR—The Current State and Future Research Directions in Single Sign-On Security Measurements*, which investigates how WebAuthn and blockchain, together with Single Sign-On (SSO) technologies, boost secure authentication systems. The research develops next-generation SSO systems to provide users with one-way access to multiple services through their single authentication details. Every authentication method based on centralized commands has associated dangers, including credential theft and token leaks. WebAuthn uses public key cryptography to deliver passwordless authentication protocols that improve user security against phishing attacks and protect their identity. WebAuthn establishes domain-specific matching keys that anchor authentication procedures to particular website origins to minimize potential attack paths. Through blockchain technology, organizations can distribute identity management across multiple nodes in a decentralized manner. Secure credential storage on blockchain delivers self-sovereign identity systems because blockchain ensures essential transparency and unchangeable data. The decentralized system prevents issues from single points of failure in SSO systems and follows modern WebAuthn trends prioritizing user privacy. The combination of these technologies forms an interconnected security framework for authentication management. WebAuthn protects user authentication by adding phishing-resistant components, while blockchain secures identity data via decentralized and tamper-resistant networks, which SSO additionally provides convenient access features. The complementary nature of these security mechanisms circumspect conventional SSO system vulnerabilities to provide scalable authentication systems suited for multiple industries. The work underscores the necessity of ongoing investigations and technical fusion to overcome new security risks and develop authentication solutions within transforming authentication systems.

Umaima Siddiqua et al. [25] explore how WebAuthn, blockchain, and Single Sign-On (SSO) can be utilized to create a secure cloud Identity and Access Management (IAM) system. WebAuthn, based on FIDO2 standards, allows users to authenticate securely using digital key pairs, making logins safer and more convenient. The authors demonstrate that implementing WebAuthn can reduce reliance on passwords, which are often a source of common security vulnerabilities. The proposed system, referred to as D2-IAM, relies on blockchain technology as its core foundation due to its decentralized and tamper-proof data structure. This system combines the functionality of smart contracts with blockchain storage to uphold precise access control rules that protect against system failures. By implementing this approach, authentication data remains secure and consistently available. As a primary access solution, Single Sign-On (SSO) enables users to verify their identity once and gain access to all authorized cloud applications simultaneously. The SSO-based D2-IAM enhances security by reducing dependency on third-party providers and mitigating risks associated with centralized failures through the incorporation of blockchain and

WebAuthn access control. This integrated solution offers faster security checks, improved scalability, and enhanced protection against security intrusions. The research indicates that the system provides three times better performance speed compared to other solutions, especially under high load conditions, while also eliminating the risks of replay attacks. This study illustrates how current technologies can enhance user access security for cloud-based ID management systems.

Kondakrindi et al. [20] demonstrated how blockchain technology interacts with SSO systems and WebAuthn systems to build secure digital authentication methods.
Users experience simplified login when SSO allows them to reach different services using just one single password. SSO systems in current use experience weak points at single control locations while user credentials remain susceptible to theft. To tackle current challenges the researchers examine how a blockchain network can establish trustworthy records that stay secure from outside tampering. Blockchain technology replaces single-point identity management systems by recording secure and permanent authentication data for better organization of access permissions.
WebAuthn helps FIDO2 make passwordless sign-ins possible through public key cryptography. WebAuthn builds unique cryptographic keys for each service rather than simple passwords so users remain secure from phishing attacks and credential misutilization. The blockchain can keep personal cryptographic data protected while building a secure system for digital authentication across networks.
The union of these methods makes an authentication system that protects users while operating efficiently on a large scale. Blockchain provides decentralized spaces to keep credentials safe with SSO methods that let people easily access their accounts and WebAuthn guarantees top-level defense against phishing attacks. These solutions combine to fix existing security issues and give users easy access to secure online management systems for all organizations.

Valeriia Balatska et al. [30] discuss how WebAuthn combines with blockchain and Single Sign-On to deliver better user security and decentralization in our evolving digital world. The authors demonstrate how WebAuthn improves the login experience and protects users from hackers through passwordless authentication that resists phishing attacks. By storing data on a blockchain network SSO systems gain safer decentralized protection against unauthorized access or manipulation. Through decentralized ledgers, blockchain safeguards digital data while giving users control of their digital identities. Through unalterable records and cryptographic protection of user credentials, blockchain enhances the trustworthiness and openness of SSO processes. Through its efficient and safe methods for sharing authentication data, blockchain resolves identity mismatches and data transfer delays among multiple SSO systems. By working together WebAuthn and this system build a trusted system for hassle-free security during user authentication. The combined solution protects against system weaknesses to build better security and ease of use for digital access controls in organizations and government departments.

Parmar et al. [13] evaluated the evolving domain of passwordless authentication. The paper compared traditional authentication methods with new systems including WebAuthn, to find how FIDO2 and WebAuthn technology benefits secure access technology. Experts evaluated Single Sign-On as an effective usability solution but

explained possible security problems that could happen when hackers attack the system. This research project identified three key problems with present authentication technologies and examined how these systems interact with each other while exploring how they prevent and handle cyberattacks from phishing emails and brute force attempts. They explained a security structure dependent on digital keys for encryption to replace knowledge-based verification methods. The research showed that systems like WebAuthn added better security and user convenience because they use public-key cryptography to replace password-based authentication. Despite these challenges user adoption proved difficult and the setup costs were too high to implement. The study team recognizes that future work needs to integrate WebAuthn, blockchain, and AI systems to solve passwordless systems' growth and usability problems. WebAuthn and blockchain technologies show why this research remains important today. The study needs more testing in real situations because it shows problems with practical use.

Klieme et al. [8] tried to enhance FIDO2/WebAuthn by introducing continuous authentication for web services. The research project added new functions to FIDO2/WebAuth to make up for the weakness in one-off authentication which stops working after users successfully log in once. Through public/private key cryptography research the team developed ways to monitor user identity constantly making sure sessions stay protected from start to end. The researchers employed FIDO2/WebAuthn to protect user access by using public/private key encryption methods. The researchers updated authenticatorGetAssertion to let the relying party and the authenticator seamlessly exchange data. The researchers solved standard compatibility problems while keeping usability high and managing expensive processing loads. The development merged WebAuth authentication technology with Bluetooth Low Energy (BLE) connections to enable user moveable devices. The analysis outlined why active user authorization helps to stop web session hijacking which poses large security risks in these settings. The team created a framework using FIDO2/WebAuthn features to maintain continuous authentication. They built a system by letting parties negotiate fresh authentication times while getting user permission and enabling continuous login support. They built an Android roaming authenticator for testing with various browser setups. Our research showed that this new method fits within FIDO2/WebAuthn standard rules and lets users stay authenticated continuously. The analysis showed multiple problems in web authentication, especially a mixed acceptance of features across browser platforms and a need to test by hand. The research found that users experience problems when several authentication checks overlap and the notification window appears too often. The researchers see additional study needed to enhance browser experiences and develop better control methods for continuous authentication systems. The researchers recommend enhancing their proposal because it works effectively in design but experiences technical problems during practical use. The researchers show how FIDO2/WebAuthn can begin authenticating web users consistently while building groundwork for better web security solutions in the future.

Bindel et al. [15] provided a formal security analysis of FIDO2 with its CTAP 2.1 and WebAuthn 2 sub-protocols, focusing on their classical and post-quantum security. The analysis focused on making the system resist attacks during crypto-

graphic algorithm exchanges while verifying user identity and preparing for post-quantum security standards. WebAuthn helps devices authenticate users with public/private keys and CTAP protects secured client-to-authenticator connections. To improve security models the researchers built on existing frameworks by introducing finer details about token bindings and verification types. They suggested network upgrades to resist downgrade attacks and recommended switching FIDO2 to Key Encapsulation Mechanisms (KEMs) and hybrid encryption methods for future quantum-security protection. The security framework protected each element of the system by establishing strong connections between relying parties, clients and authenticators. Research data shows FIDO2 with WebAuthn and CTAP protocols defend against traditional cryptanalysis and post-quantum encryption attacks when specific requirements are met. The system demonstrated improved protection from compromised cryptography and suggested methods to spot downgrading attacks. The researchers pointed out key obstacles including the need for common post-quantum standards along with issues involving registration authentication and making algorithm choices user-friendly. The system has both practical constraints due to its tamper-proof authenticator assumption and operational barriers in updating legacy platforms. The research team wants us to learn more about official post-quantum security algorithms plus advanced hybrid scheme structure plus create easy-to-understand security protocols. Although this research handles security on multiple levels with passwordless authentication it faces deployment hurdles since these advanced cryptographic tools are tough to integrate. Research moves the field of secure authentication closer to practical use and future protection while also showing its importance to experts.

Fugkeaw et al. [16] developed a blockchain-based access control system called D2-IAM for secure and efficient Single Sign-On (SSO) authentication, dynamic authorization, and preventive access control in cloud environments. This research responded to typical SSO issues by discussing security problems at centralized locations and excessive communication and operational waste in multi-application contexts. The work relied on blockchain technology and smart contracts to create decentralized access control methods which cannot be altered or damaged. Research showed how WebAuthn combines with blockchain and single sign-on solutions through its focus on password-free decentralized security but focuses more on blockchain implementations than WebAuthn itself. The researchers built out a system design that combined smart contracts with blockchain encryption tools to authenticate users while their access tokens and privilege settings were safely stored on the blockchain network database. They created a simple authentication process that cut communication delays and matched user permission levels to specific data you wanted to access. The researchers tested their system on Google Cloud through Java and Solidity programming. Data authentication on D2-IAM proved superior to other blockchain systems by processing security tasks with high speed and large capacity. The system processed IDs faster than other setups while accepting more transactions at once during growing workloads. D2-IAM proved strong against attacks by handling multiple applications while defending against replay and man-in-the-middle exploitation. Despite these benefits this research had two major issues: it did not offer an auditing system for policy security and stored sensitive data in regular cloud servers. The researchers proposed further research on

Internet Peer Distribution P2P (IPFS) storage framework combined with machine learning methods to improve security against network attacks. This work explores an effective method to merge blockchain technology with SSO systems and solve cloud-based access control problems.

Hanzlik et al.[17] conducted research on FIDO2 authentication standard by designing solutions to privacy issues and key revocation difficulties. Their main focus involved enhancing FIDO2 unlinkability implementations and creating practical worldwide key revocation functionality.The research examined WebAuthn as an essential FIDO2 component while studying key derivation functions and key wrapping approaches. The implementation introduced BIP32 as a key derivation standard from blockchain which enhanced control over key management systems. FIDO2 needs better protection against privacy breaches by making authentication sessions difficult to link while also requiring a reliable worldwide key revocation system. By showing how cryptographic key management affects authentication security this study integrated WebAuthn and blockchain (BIP32 for deterministic key derivation) together with Single Sign-On (SSO).The authors built upon past security models through the mathematical definition of unlinkability specifics at three different protection levels. The analysis of actual FIDO2 tokens revealed that KW maintains strong unlinkability but KDF provides only medium-linkability levels. The authors addressed revocation through BIP32-based procedures that allow users to cope with all credentials at once in contrast to individual service-based revocation approaches. Adding a revocation system to this model leads to decreased unlinkability benefits.The BIP32 mechanism lets users revoke keys throughout their network but this method exposes revoked cryptographic materials making their users detectable. The new implementation also fails to work with every FIDO2 service currently operational. Additional research needs to establish enhanced unlinkability protocols that support efficient key revocation functions.Formal privacy models together with a new key revocation technique represent major security enhancements described in this paper. Achieving ideal balancing between unlinkability features with usable security is still a pending research objective.

Westers et al.[18] studied Single Sign-On (SSO) security and privacy features in existing deployment environments. The research team pursued three main tasks which involved vulnerability discovery and privacy risk analysis alongside automated large-scale assessment tool development.The research team created SSO-MONITOR which operates as an automated system for analyzing SSO implementations. The system integrated OAuth 2.0 alongside OpenID Connect (OIDC) and SMART-PARAMS and Selenium to discover security flaws. The analysis addressed security misconfiguration issues as well as unprotected CSRF attacks and open redirect vulnerabilities and automatic unauthorized login behavior. WebAuthn received limited attention in the research while the team evaluated privacy protection approaches inspired by blockchain technology. The research evaluated advanced authentication methods for password replacement which connected SSO to WebAuthn.During its 10,000 website scan SSO-MONITOR discovered 1,632 sites which implemented SSO through Google, Facebook and Apple platforms. The analysis revealed 447 CSRF vulnerabilities together with 342 security issues in deprecated flows as well as 9 open redirects and 5 secret exposure points. The analysis discovered 200 websites

logging users into their systems without receiving explicit consent thus violating their privacy rights.SSO-MONITOR exclusively analyzed security at the front end while blind to token exchange operations in the backend. The implementation of CAPTCHAs presented difficulties while newer defense systems including DPoP and mTLS need more investigation.This research paper delivers high value to the field of SSO security scholarship. By revealing actual system weaknesses this project delivers both an accessible open-source framework that works at large scales. The development of secure authentication streams presents ongoing difficulties because they must support usability requirements.

Ghosh et al.[19] developed a resilient FIDO2-compatible smart card system for decentralized financial transactions. This study focused on improving security in fintech by solving authentication method weaknesses.The study used FIDO2 for passwordless authentication alongside blockchain for decentralized data security and WebAuthn for seamless platform integration. The research team focused on creating solutions which blocked phishing attacks together with man-in-the-middle attacks along with session hijacking. The system protected transit and storage data through advanced cryptographic standards RSA and AES256.Researchers deployed the PP2PP framework through physical smart cards that operated with NFC and Bluetooth and USB connections. Security through the system depended on HTTPS protocols along with DDoS protection and secure virtual cloud networks. Remote attacks became impossible through a security system which enabled multi-factor authentication using locally stored physical security keys and biometric data.The Azure Virtual Machine-based experimental setup processed transactions within 500 ms. The implemented framework effectively protected against standard cyber threats which incorporated both phishing attacks and SQL injection vulnerabilities. The study found success but the authors identified device theft risks that necessitate enhanced user education and additional security measures.The publication showed convincing evidence that PP2PP functions as a game-changing e-banking solution. This framework integrated WebAuthn with FIDO2 and blockchain to solve security flaws in current systems by providing both extensive protection and implementation flexibility. The future investigation of device protection methods together with user information campaigns will strengthen the solution's overall impact. These findings contribute substantially to future achievements in secure financial technology system development.

Lassak et al.[32] studied the sluggish pace of FIDO2 passwordless authentication adoption despite its proven security advantages. The study worked to discover why organizations hesitate to switch from passwords and suggest ways they could boost FIDO2 adoption rates.The research evaluated FIDO2 and WebAuthn and passkeys while investigating their compatibility with enterprise authentication frameworks. The research investigated fallback solutions alongside recovery protocols and security risks as well as regulations compliance before examining the user interface problems. Security worries alongside Single Sign-On (SSO) integration formed essential components of the discussions involving enterprise authentication even though blockchain remained outside the main focus.Through 28 structured interviews with CISOs, authentication managers and FIDO2 experts the researchers identified challenges in deployment. Security benefits from FIDO2 received support from companies but

adoption remained limited because of user friction and browser support issues as well as account recovery challenges and technical limitations and compliance norms including eIDAS and PSD2 regulations. Security experts rated different barriers as important obstacles but organizations actively chose usability and backup solutions instead of focusing on security improvements.The research examined enterprise applications while neglecting both small business scenarios and consumer integration obstacles. A large-scale statistical evaluation of FIDO2 adoption across different platforms was missing from this research study. The next phase of work must examine improved passkey implementation methods and regulatory standardization and backup security protocols.The research offers essential information about the practical barriers that FIDO2 encounters when organizations attempt to adopt it. FIDO2 security stands strong but the present limitations of usability and compliance standards continue to block its widespread acceptance. The research findings demonstrate the necessity for better industry coordination with improved fallback systems towards achieving broad FIDO2 adoption.

Lamba et al. [21] did studies to test two-factor authentication's effectiveness as a security enhancement for accounts. The researchers worked to study 2FA both positively and negatively while examining its Defense-against evolving cyber dangers and proposing new measures to manage present-day cybersecurity risks. The research aimed to develop improved authentication methods which protected against user security threats such as phishing, credential breaches and unauthorized access attempts. The team tested different 2FA solutions starting with time-based one-time password (TOTP) systems alongside cryptographic extensions like WebAuthn which deliver codes by SMS or email. WebRelation proved its best feature through FIDO2 because it uses public-private key cryptography to protect accounts from phishing incidents and ensures login credentials stay domain-specific. According to the research WebAuthn enables user authentication through single-sign-on (SSO) systems while maintaining smooth security functions. Blockchain principles seem good for authentication security goals because they build decentralized trust which removes single point failure and offer tamper resistance functionality. The authors did an extensive analysis of authentication systems by discovering vulnerability and then to find solutions which improve Two-factor authentication reliability. Analysis of failed 2FA implementations showed users face challenges because of session state confusion alongside insufficient lockout protection and SMS authentication issues. WebAuthn adoption gained support from researchers because it allows better security combined with usability while requiring users to set up multiple devices to avoid lockouts. Traditional 2FA methods through SMS or email prove vulnerable to interception attacks and social engineering methods, according to study findings. The analysis concluded that WebAuthn represents the most secure and easiest to use approach for two-factor authentication setup. The researchers also stated WebAuthn's adoption depends on hardware support but noted that this requirement hampers deployment in restricted resource settings. The researchers insisted on the value of maintaining improvements in 2FA solutions while raising awareness for user education about potential security risks to achieve a secure space in online platform. The research examines both the advantage and weakness of 2FA technology through its findings. WebAuthn emerges as a good security solution for user account protection although implementation barriers and user education programs remain

essential for fulfilling its potential.

In this paper, Pietilä et al. [24] conducted a research on passwordless identity and access management (IAM) system implementation through FIDO2 standards for web application authentication. This study examined the potential of trading conventional password authentication systems for a more secure cryptographic approach that offered improved user friendliness. The initiative focused on solving the main weaknesses of password authentication which includes - the susceptibility to repeated use and security incidents and phishing exploitation. Several authentication components were used during the research: FIDO2, WebAuthn, OpenID Connect (OIDC), alongside Keycloak functioning as the identity access management solution. Security evaluations along with user-friendliness and secure connection standards received attention to resolve current traditional authentication system challenges. The implementation noticed component-to-component interoperability while strengthening attack resilience and modernized authentication standards. While blockchain technology was absent from the system's design its approach to authentication security through cryptography functions in a manner similar to blockchain principles. The implementation of WebAuthn technology inside Keycloak proved how passwordless authentication can work together with single-sign-on to create a good secured system. The research team implemented Keycloak as their authentication management solution while WebAuthn handled both registration without passwords and login functions. The system architecture implemented two data partitions where PostgreSQL stored databases alongside Nginx managing routing and load-balancing operations. Seamless integration became possible through this framework while guaranteeing both security and scalability. Manual tests verified essential features of the system but the team will introduce future automation for their integration testing procedures. Testing resulted in an effective deployment of a passwordless IAM solution which met every essential condition. Through this solution consumers got easier access while stability from security vulnerabilities remains maintained and the system scales efficiently for authentication authentication. Though, user-quality security with managed email accounts worked as a vulnerability because the research relied on this method for recovery access. Future research directions should focus on adding more secure authentication techniques for recovery while enhancing education about passwordless systems for users according to the research team. The research introduces a durable and usable framework for passwordless identity and access management systems. WebAuthn integration with Keycloak SSO protocols presents organizations with a proven authentication solution that addresses current authentication requirements. But, additional research needs to be done to build recovery functions and establish widespread implementation of the solution.

Yan et al. [26] talks about web3 authentication, which is used for verifying the security of the users on the internet. But in this paper, the author talks about a major weakness of this system by introducing a new type of attack which is named- "Blind Message Attacks". In this attack, the users can be easily tricked into signing messages from malicious websites without their permissions, as users or Web3 do not often verify the source of the message. Then, the solution of this attack was discussed in this paper. In this research, the author first tries to find the vulnerability of Web3 by developing a tool named- "Web3AuthChecker". Then the author

tries to show the severe risk of the attack by showing a simulation of those attacks as a demo and lastly, developed a side tool for users named "Web3AuthGuard" to identify malicious activities. The result of this safety tool is remarkable. Initially, almost 76% of the tested applications were proven as vulnerable of Blind Message Attacks, which was very concerning. Then Web3AuthGuard was proven to be very useful by detecting 80% of the attacks in tested applications. So, enhancing or updating the Web3AuthGuard more can go a long way to identify protocol level flaws in future. This paper also proves how users also should gain basic knowledge about their security online, as it can be easily tampered.

To impose a solution on the traditional security method, Ou et al. [23] integrate FIDO2 with the blockchain architecture. Traditional security methods such as passwords are easy to crack. Even sometimes, biometric authentication also can not provide security wholly. FIDO2 provides a stronger security, but even then security can be a concern in case of a lost device. Sometimes the sensitive information can also be tampered or modified. The decentralization nature of blockchain prevents the risk of single point failure, while at the same time FIDO2 provides good security. Blockchain also provides immutability, meaning once an information is recorded in Blockchain, it is very much difficult to change them, so an attacker can not tamper with information easily. So after registration, the system generates a new key pair, and stores it in the blockchain, which is very secure. This stored key will be checked each time while logging in by using various challenges. ( like solving puzzles). ECDSA (Elliptic Curve Digital Signature Algorithm) has been used for generating the key and challenges.The proposed method shows better results over the traditional FIDO2 method as it has significantly reduced various types of attacks because of its immutability, decentralized nature and traceability. So this proposed system provides a more secure and less costly authentication. Furthermore, this author suggests to integrate the passkey in this system which could make it more flexible and convenient for users.

Ferdous et al. [7] introduced, a privacy-friendly attribute aggregation system, called Social Anchor, and used in social networks, overcame the main problem with identity management related to the impossibility of acquiring attributes of multiple Identity Providers (IdPs) during the same session and privacy-invasiveness of attribute release procedures. The study suggested drawbacks to current standards such as SAML, OpenID, and OAuth in which social networks (e.g. Facebook, Google and LinkedIn) compelled users to disclose all-or-nothing attributes without selective disclosure, explicit authorization of disclosed attributes or aggregation support, contravening data minimization principles and subjecting users to privacy risks. Based on an earlier Hybrid model that integrated Identity Proxying and Relay models, Social Anchor suggested two architectures, Type 1 used a Hybrid IdP (SimpleSAMLphp) to proxy attributes across seven SN IdPs with user consent interfaces; Type 2 used a Portable Personal IdP (Android app) to provide relay mode where attributes were encrypted to conceal their presence in the Hybrid IdP, a characteristic of different privacy levels through the SAML federation interoperability with non-SAML OpenID/OAuth providers. The structure entailed the threat modelling on the basis of the Dolev-Yao hypotheses, requirements on functions, privacy as well as security, aggregation/release algorithms, formal trust examination and protocols verified by

AVISPA, which established the secrecy and authentication objectives. Through proof-of-concept implementations and use cases (e.g., job applications using LinkedIn/Twitter/Facebook data to aggregate information) experimental results showed viability with an acceptable latency (Type 1 < Relay < full aggregation), successful integration exceeding baseline systems in privacy controls and multi-IDP support although no significant failures were noted. The limitations that were recognized were high user interactions that lowered usability and the dependency of Type 2 to mobile-browsers; future research proposed blockchain innovations (e.g., Ethereum to PPIdPs). Compared to the existing aggregation models, Social Anchor is more successful in integrating SNs privately.

Kommareddy [28] reviewed the SAML implementations in modern web services to gain insight into the design architecture, security attributes and performance limits and compare SAML to other identity protocols (OAuth and OpenID Connect). The main idea of the paper was to discover the issues related to SAML-based security implementations and to give some tips on how to enhance identity federation and secure web development. The study employed the use of SAML 2.0 as the foundation protocol, which was coupled with the XACML (eXtensible Access Control Markup Language) used to enforce policies and WS-Trust as used to facilitate the hybrid federation models. The author explored applications of SimpleSAMLphp and Shibboleth. Particular challenges that were addressed were XML signature wrapping attacks, scale issues with microservice architectures, cross-domain interoperability, and complexity of configuration. The paper covered the authentication latency at a large number of users and the experiments proved that the average time of authentication was between 220ms when only 100 users were present and 790ms when the number of users was 2000 parallel users. Although the paper did not directly apply the WebAuthn or blockchain, it recognized their applicability in the future research directions. The SAML use case that was used as the basis was Single Sign-On (SSO) which showed that it was able to make cross-domain authentication that was supported by centralized identity providers. The researcher has worked on the problem by considering literature review and face-to-face analysis of the available deployments. The framework that was proposed combined an Identity Broker to support multi-domain routing, Policy Engine to support dynamic access control with the help of XACML and Audit Logger to support compliance monitoring in accordance with the GDPR and HIPAA regulations. The metrics used in the study to compare SAML and OAuth 2.0 and OpenID connect comprised of latency (SAML: 510ms, OAuth: 310ms), token size (SAML: 2.4KB, OAuth: 1.1KB), and security ratings. It was found that SAML was found to offer better enterprise security but with the drawback of increased computational load. The cross-domain SAML implementations have a lower success rate at 97.3 percent with clock drift and mismatched trust than the single domain implementation of 99.8 percent. The author also recognized constraints such as linear scaling of CPU performance, fixed size assertion that introduces memory bottlenecks and complexity barriers to implementation. Additional areas that need to be explored further involved lighter SAML-based IoT, AI-based anomaly detection in federated authentication, blockchain-based decentralized identity models, and cross-protocol federation models.In the current paper, a comprehensive synthesis of SAML studies conducted during 2008-2020 is provided, and it successfully puts technical issues into perspective in real-world deployment.

Nevertheless, insufficient original experimentation and use of secondary data sources diminish the novelty of the contribution and the value of the contribution is rather a useful survey.

Hartl and Derek [27] conducted the initial formal security analysis of the SP-initiated POST/Artifact Bindings use case of SAML V2.0 Web Browser SSO Profile, aimed to test the security properties by automated symbolic analysis of the protocols as opposed to testing particular deployments. The main idea behind the paper was to provide a gap in the formal verification of the SAML implementations by modeling the technical specification itself and revealing vulnerabilities in various variants of the protocols. The study used Tamarin prover as the state of the art analysis tool of symbolic protocols, and it models communication over a one-way authenticated TLS. The authors investigated SimpleSAMLphp, Shibboleth, opensaml, LemonLDAP, and OneLogin implementations to learn how under-specified mechanisms were implemented in practice. Certain issues addressed were XML signature wrapping attacks, configuration, cross domain trust building, and scalability limitations in federated architecture. The research discussed the signing of authentication requests ( signed and unsigned versions), the implementation of the RelayState mechanism (with and without state storage), the creation of identifiers by SAML (cryptographically strong and weak random identifiers). Although it was not directly applied in the paper, the use of WebAuthn and blockchain technologies was considered in future models of decentralized identity. Single Sign-On was used as the architecture foundation where the POST/Artifact binding was used as the alternative of the more widely used Redirect/POST binding by transmitting the artifact as an indirect response. The researchers came up with a new methodology to determine the existence of sensitive optional features and under-specified mechanisms developing a meta-model which is instantiated in eight protocol variants. They modeled their framework by unlimited protocol participants having Dolev-Yao adversaries that can compromise arbitrary agents. The strategy involved formal modeling of the establishment of trust, new TLS communication primitives to do request-response patterns, and detailed system set up policies to permit arbitrary trusting. The researchers however found that the artifact-session confusion attack violates IdP-SP non injective agreement in all the variants, but standards-compliant SPs would not accept manipulated assertions. Cross domain setups had a low authentication success rate of 97.3 percent because of clock drift and metadata mismatch. Limitations mentioned by the authors were including idealized client modeling which abstracted browser-specific behavior and user error, simple client authentication protocol, and partial coverage of fields and error handling paths of the SAML messages. Topics that still needed to be developed were to analyze all Web Browser SSO Profile use cases, encrypted assertion variants, modeling the denial-of-service attacks, and integration with concrete SAML implementations in order to test them against practice deployments. The paper has introduced a break-through formal verification research that shows the security soundness of SAML POST/Artifact in addition to exposing hidden implementation traps.

## 2.3 Summary of Key Findings

The reviewed literature shows a significant movement towards the use of cryptographic and privacy-safe identity systems via modern and cryptographic authentication as opposed to password-based authentication. One of the themes that can be identified in a range of studies is the high level of security provided by WebAuthn and FIDO2 that do not require passwords but utilize asymmetric cryptography and offer a high level of phishing and replay protection along with domain-specific credentials. Jannett et al.[31], Siddiqua et al.[25], Parmar et al.[13] and Bindel et al[15]. highlight that WebAuthn promotes the level of security and usability by eliminating shared secrets and grounding authentication to particular origins. Simultaneously, researchers state some real-world issues, such as hardware reliance, compatibility with browsers, recovery processes, and complexities in user adoption. Other papers also expand the abilities of WebAuthn: as an example, Klieme et al. introduce a proposal of continuous authentication to mitigate the threat of the session beyond the initial authentication, whereas formal analysis by Bindel et al. investigates the security posture of WebAuthn/CTAP2 and shows what is needed to withstand downgrade attacks and the upcoming post-quantum threat.

The second key learning relates to the constraints and weaknesses of the current SSO systems. Real-world tests like large-scale SSO-MONITOR by Westers et al. indicate high-rates of configuration mistakes in OAuth/OIDC and SAML implementations such as CSRF risks, unsafe redirect usage and unintentional auto-log ins. These results emphasize that the traditional SSO implementations are not as secure as they need to be and that they should be subjected to much stricter validation and standard compliance and automated test frameworks. Although SSO leads to improved usability through easy access in a variety of applications, it is always indicated in the literature that configuration errors and poor deployment methods continue to be a leading source of vulnerability.

A third theme is connected to privacy, in particular, attribute release and selective disclosure. Surveys like Social Anchor (Ferdous et al.[7]) point to the limitations of existing SAML, OAuth, and OpenID-based platforms which compel users to all or nothing attribute disclosure without finely-grained consent. This goes against the principles of data minimization and results in the unwarranted exposure of information of user identity. These papers indicate that privacy-friendly attribute management and explicit user consent upon disclosure are necessary as well as greater control over intermediate processing of identity attributes which directly infer the applicability of the selective disclosure mechanisms in the contemporary SSO ecosystems.

Lastly, some of the works disclose adoption and usability issues with passwordless systems. Researchers like Lassak et al. have shown that companies are aware of the security advantages of WebAuthn/FIDO2 and have challenges such as fallback authentication, managing devices, regulatory challenges, and users to learn curves. Other studies emphasize the need to come up with systems that would be both very strong cryptography and sensible recovery functions and platform consistency in user experiences. Taken together, the insights presented above indicate that whereas passwordless authentication offers important security benefits, several practical usability, recovery, and interoperability issues have to be resolved so that large-scale implementation can be achieved.

# Chapter 3

# Requirements, Impacts and Constraints

This chapter explains the main benefits of the proposed system. It first introduces the requirements used to design the system architecture and then describes the development process. These requirements are based on the foundation and challenges discussed in Chapter 2. The chapter also outlines the steps followed during implementation, showing how the selected approach solves the identified issues and ensures secure and private identity management.

## 3.1 Final Specifications and Requirements

### 3.1.1 Threat Model

Here we are using STRIDE model which is a threat modeling framework stands for 6 categories of threats developed by Microsoft [4]. This model mainly helps to identify threats in software systems and classify them. Those 6 categories are discussed below:

T1. **Spoofing:** This threat indicates to ensure user identity verification and credential during WebAuthn usage. In our case, man-in-the-middle attack may intercept authentication request.

T2. **Tampering:** This threat refers to altering data like SSO tokens or WebAuthn responses to manipulate the system.

T3. **Repudiation:** This threat refers to the ability of a user or system to deny an action without proof of the contrary. Users may claim they never logged in via SSO, which make it difficult to verify the authentication system.

T4. **Information Disclosure:** This threat hello indicates unintended exposure or leak of sensitive information. Sensitive tokens or session data can be exposed if user's device is left unlocked and unauthorized individuals have access to the tokens.

T5. **Denial of Service:** This threat refers to overwhelm systems, making them unavailable. For example: Scammers can attack the system with fake requests.

T6. **Elevation of Privilege:** This threat indicates when a user gains unauthorized access beyond their authorized level, generally to higher levels of system. In our case, SSO misconfigurations may allow users to unauthorized high-level access.

### 3.1.2 Requirements

#### Functional Requirements

Functional requirements indicates the core functionalities that the system should have. The functional requirements are presented below:

F1. The system should integrate SSO within the WebAuthn Framework.

F2. Users should experience instant access to multiple applications from a single authentication point which removes the need for repeated authorization procedures.

F3. The system should create, maintain and terminate sessions in a secured way between multiple applications/browsers.

F4. Users should be able to register and maintain their WebAuthn credential including biometric or hardware authenticator.

F5. The system should not require any kind of passwords and should only require public/private keys.

F6. A reliable backup authentication solution should exist whenever WebAuthn credentials become inaccessible because of device loss or missing hardware authenticator.

F7. The system must ensure compatibility with one of the standard identity federation protocols, SAML.

#### Security Requirements

Security Requirements to reduce the identified security threats, the following requirements are needed for the proposed system:

S1. The proposed system should ensure a secure authentication mechanism so that only the verified real users can access the system. This will mitigate T1.

S2. The proposed system should ensure strong monitoring security so that important data can not be modified easily. By ensuring this, the risk of T2 can be reduced.

S3. By using a tracking setup in the proposed login system, the logs of users should be kept. By this way, attackers cannot deny any of their wrongdoings which reduce T3.

S4. Encryption can be a good solution against the unintentional threat T4. By this way, even if the information would somehow get exposed, it would be encrypted which seems meaningless to others that significantly reduces the risk of T4.

S5. The proposed system is expected to introduce request rate limiting, CAPTCHAS and load balancing to avoid congestion of regular or fake traffic that overloads the system. This will mitigate T5.

S6. The proposed system would require a good level of authentication in SSO actions and validating identity, which will prevent unauthorized access. This will counter T6.

**Privacy Requirements**

P1. The system must support attribute level selective disclosure.It must ensure that the Identity Provider (IP) or any Service provider (SP) do not receive any plaintext attributes. IP and SP will receive hashed attributes which users will disclose according to their choice.

## 3.2 Societal Impact

The proposed authentication system presents a user-centric, privacy-saving, and password-free identity model with important implications on society. The system significantly lowers the chances of phishing, identity theft, and unauthorized access of account information by eliminating passwords and replacing them with cryptographic credentials provided by the WebAuthn system, risks that present threats to millions of users all over the world. Selective disclosure integration will make people have full control over their personal attributes they want to disclose to service providers, which will benefit building a more healthy digital ecosystem in which privacy will be the norm, not the exception. This enables the users to be provided with a better degree of security in engaging in internet-based services like banking, health, education and e-government online. Culturally regarding the safety of the people, high-level authentication will minimize the chances of fraud in sensitive industries. Minimization of data also means that in case there is a breach, minimal value can be of the information leaked and this minimizes the damage in the long term. The system is legally compliant with the data protection rules in the world and adheres to the principles of limiting the purpose, minimizing the amount of data and the consent of the user. The system in question is cultural in that it has diverse expectations as far as privacy and trust are concerned since users get to decide what they want to share with others and this is accommodative to societies with different privacy norms. In general, the system helps to make the digital society safer, more transparent, and ethically responsible.

## 3.3 Environmental Impact

The environmental effects of the suggested system are not very high, since most of the elements are based on the digital infrastructure available, including identity providers, service providers, and common authentication protocols. Indirectly, the system reduces the computational cost of helpdesk operations, brute force attack defense and repetitive credential lifecycle management by eliminating password-based authentication and by minimizing password resets. Smaller resources of the backend are needed to store and process plaintext secrets, lowering long-term storage expenses and the energy consumptions. Also, selective disclosure eliminates data transmission that is not necessary within networks. Attributes that users do not set to be known are only processed and stored and this reduces the number of data that is not needed. Although cryptographic functions like WebAuthn and hashing do have a computational cost, they are lightweight in terms of computational cost relative to the traditional multifactor authentication systems. In the long term, distributed efficiency and minimizing the data storage volume will have a positive effect on digital sustainability. The system is shown to be environmentally responsible in that it will maximize the use of resources and be based on decentralized and minimalistic identity processing.

## 3.4 Ethical Issues

There are multiple ethical issues in the design and implementation of a privacy preserving authentication system. To start with, user identity attributes are handled in such a way that it requires strict compliance to the ethical data practices. Identity Provider does not get to know the actual attributes of the user since client-side hashing is used to make sure that the Identity Provider does not have a chance to profile, abuse, or access the user with unauthorised access. This is quite compatible with ethical aspects of privacy and autonomy of users. Second, the system should be transparent to the users. Consent mechanisms must be clear, particularly when it comes to selective disclosure, where the user should be aware of the consequences of releasing or not releasing certain attributes. The other ethical issue is connected to digital inclusivity: the system must be available to those with little access to hardware authenticators or technical skills. The system designers have the duty not to create digital divides. On the whole, the ethical system model is based on user control, fairness, transparency, and responsiveness in stewardship of personal data.

## 3.5 Project Management Plan

The project was managed in a systematic plan which was carried out in an iterative and incremental manner. This was broken down into four large stages: Research/ Requirement Analysis, System Design, Implementation and Performance Evaluation. Tasks were scheduled within the semester to ensure that there were clear milestones in how the architecture design, data model development, WebAuthn implementation, SAML SSO setup, selective disclosure module and performance testing were conducted. Resource management involved assignment of computing resources which included development servers, local testing environments and JMe-

ter load-testing tools. The prioritization of the tasks was based on the critical path: SSO integration, WebAuthn registration, hashing and selective disclosure, performance evaluation, documentation. This organized management has made sure that the use of resources is efficient, that the deliverables are completed in good time and the ability to adapt to technical issues.

## 3.6 Risk Management

The proposed authentication system needed to be developed with proper risk management, as it involves a variety of interacting elements, including WebAuthn, SAML SSO, cryptographic hashing, and disclosure that is controlled by users. A major technical risk was the failure of integration, as the integration of client-side hashing, SAML assertions, and WebAuthn flows needed to coordinate very closely among Service Providers and the Identity Provider. A mitigation of this risk was in the form of a modular architecture, incremental testing, and thorough API logging. Other risks were linked to security; improperly configured SAML signature validation, insecure hash functions, or the storage of temporary data might have potentially exposed sensitive data. In order to reduce such risks, strong salting, client side hashing and rigid implementation of XML signature validation and HTTPS had been embraced. In the performance risk, load testing and caching strategies were used because cryptographic operation may be latency-inducing when the load is heavy. The risks related to the users (e.g. the inaccuracy of the attributes being entered based on the selective disclosure, or the non-availability of hardware authenticators) were prevented since the user interface was designed to be very clear and the mechanism of fallback authentication was supported. The project was able to reduce operational, security, and usability risks through constant monitoring of the project, early detection of failure points and implementation of security practices that are acceptable by the industry.

## 3.7 Economic Analysis

The system proposed involves use of hardware authenticators that are WebAuthn-compatible like Yubikeys, which makes overall implementation and maintenance of the system more expensive than conventional password-based systems. Every user needs a hardware authenticator at least and an emergency technology preferably, which would cost more than an initial investment and continued maintenance costs in the event of replacement or a malfunction in the device. Also, the active server resources, updates, and security control are included in maintaining the Keycloak IdP, the SAML integration, and the selective disclosure elements. Even with such costs, the high passwordless security of the system can greatly mitigate risks and operational costs in the long term of password reestablishment, phishing, and identity fraud unlike in the conditions of low-security settings, and the investment is economically justified in high-security settings.

# Chapter 4

# Proposed Methodology

In this chapter, the goal is to present the system through architecture design, data model and algorithm, protocol flow and real-life implementation.

## 4.1 Methodology Overview

The proposal aims to integrate Single Sign-On (SSO) with WebAuthn to enhance authentication security and scalability. By leveraging hardware-based authentication and biometrics alongside SSO, the solution reduces vulnerabilities like phishing and credential theft. The research workflow for this integration is illustrated in Figure 4.1, which outlines the steps involved in the methodology of this research. The workflow begins with the development of a threat model to analyze the integration of WebAuthn and SSO. This is followed by requirement analysis to review existing authentication systems, architecture design, protocol flow to define system requirements, and implementation to design secure protocols. Finally, the process concludes with use-case formulation and evaluation, where the system is tested to ensure it meets the goals of secure and scalable authentication. The proposed framework will use public-key cryptography to validate users across trusted security domains while maintaining operational compatibility. The final system will be accessible on any device, offering a user-friendly authentication framework that balances practicality with robust security measures.

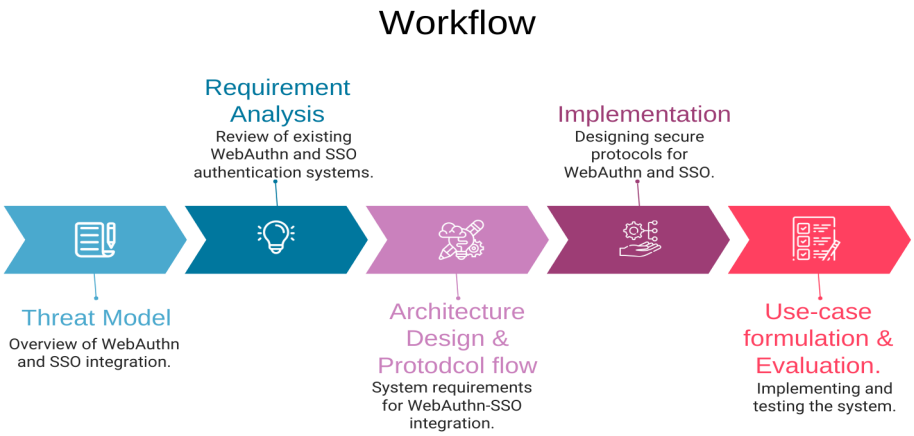



Figure 4.1: Our research workflow

## 4.2 Architecture

In Figure 4.2 , shows the design of our system we have implemented, integration of Single Sign-On (SSO) within the WebAuthn framework to provide a secure, convenient and passwordless authentication system. The system has four fundamental components: User, Service Provider, Identity Provider and WebAuthn Authenticator.

The authentication process starts when the user tries to get access to Service Provider I. If the user is not registered and authenticated previously, Service Provider I transfers the request to Identity Provider. Identity Provider provides a registration form where users submit their information. Upon submission, all information are hashed and secured and thus no plain text is transmitted to the identity provider which ensures strong security of the system. Then the user is redirected to WebAuthn Authenticator.

Webauthn authentication initiated by identity provider through the browser. The browser interacts with the authenticator by biometric or hardware-based authentication method. The user identifies themselves with their device-located WebAuthn credentials (e.g., fingerprint, face recognition or security key) which are cryptographically checked.

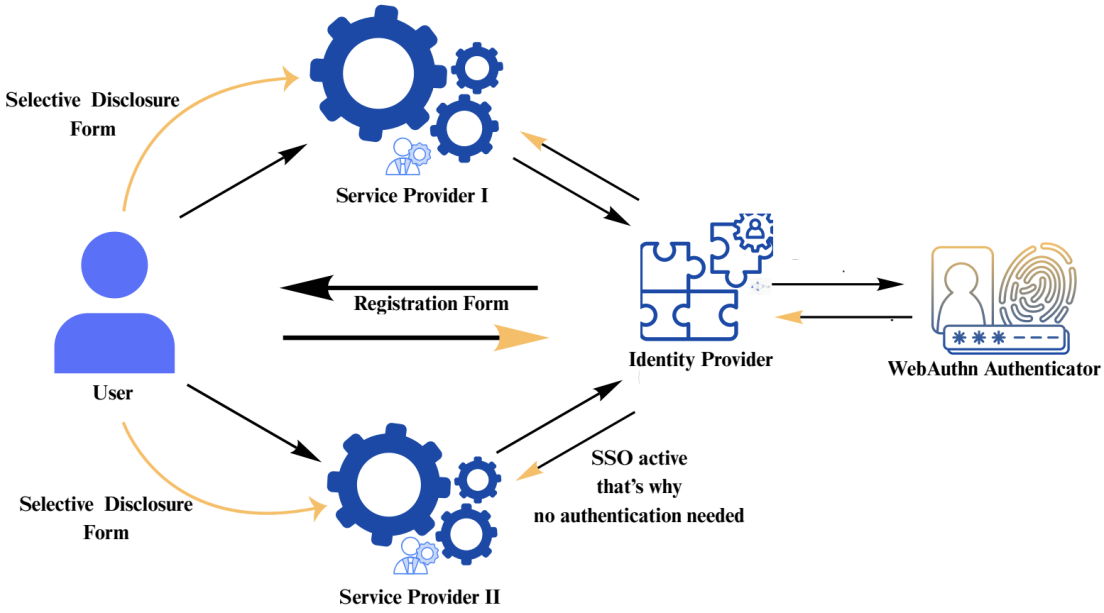



Figure 4.2: Architecture Design

After successful authentication, the Identity Provider sends the SAML assertion to the service provider containing user hash attributes and the user gets a selective disclosure form. In this form , users can clearly see which personal information is requested to share by service provider and can choose which attributes and informa-

tion they want to share with. The provided selected attributes are compared with the hashed information which are stored previously at the time of registration by the identity provider. This process ensures that users have full control over their data and thus the system ensures user's explicit consent.

The maintenance of the session is safely deleted on logging out to ensure against hijacked sessions. Service Provider II is added to exemplify the functionality of Single Sign-On (SSO). The same authenticated user who wants to access at Service Provider II, does not need to be authenticated again. This architecture ensures strong security by phishing-resistant WebAuthn credentials and usability by making background SSO among trusted domains.

## 4.3 Design (Model) Specification

In this section, we explore the protocol flow, illustrating the working mechanism of the proposed system along with the underlying use-cases. To achieve this aim, we present the data model and algorithm (Section 4.3.1) and protocol flow (Section 4.3.2) below.

### 4.3.1 Data Model and Algorithms

We present the notations used throughout the data model, algorithms, and protocol flow in Table 4.1.

**Data Model:** The data model in Table 4.2 defines a structured system for hash-based selective disclosure within SAML SSO.All message formats and stored items used in registration, WebAuthn verification, SAML SSO authentication and selective disclosure are formalized in the data model. Each protocol message can be expressed as a requestresponse pair, with Req = {reqtype,reqdata} and Resp = {resp type,respdata} such that all the operations have the common format.

The model has defined a few types of requests that are corresponding to the stages of system: WebAuthn registration (registrationReq, regChallengeReq, regVerificationReq), WebAuthn credential generation and assertion (credentialCreationReq, credentialAssertionReq), SAML login (samlLoginReq), and privacy-preserving selective disclosure (selectiveDisclosureReq). All request types have a structured data payload, which could be hashed attribute ($A_h$), salts ($\sigma$), challenges (CHL), relying-party identifiers (RPID), and cryptographic artifacts, like the credential ID (CID), public key (PK), signed challenge (SCHL) and attestation object (AttObj).

On the response side, the model identifies challenge responses, verification responses, SAML login responses, credential creation responses and selective disclosure outcomes. Selective disclosure generates verified, failed and withheld attributes ($A_{verified}$, $A_{failed}$, $A_{private}$), which guarantee attributes are released at the discretion of the user.

There are storage side that stores credential data (CredentialStore), ongoing challenges (ChallengeStore), user attribute verification states (AttributeStore) and SSO session data (SSOSession). Combining these architectures allows passwordless authentication, federation via SAML and privacy preserving attribute sharing.

Table 4.1: Notations and Descriptions

| **Notation** | **Description** |
|---|---|
| $U$ | User |
| $SP$ | Service Provider (Django Application) |
| $IdP$ | Identity Provider (Keycloak) |
| $BR$ | Browser / Client |
| $DB$ | Database |
| $H(\cdot)$ | SHA-256 Hash Function |
| $\sigma$ | Fixed Salt Value |
| $\|$ | Concatenation Operator |
| $A$ | Set of User Attributes |
| $A_h$ | Set of Hashed Attributes |
| $A_p$ | Set of Plaintext Attributes |
| $A_s$ | Set of Selected Attributes for Disclosure |
| $NameID$ | SAML Name Identifier |
| $Assertion$ | SAML Assertion Object |
| $SessionID$ | Session Identifier |
| $EntityID$ | SP/IdP Entity Identifier |
| $ACS$ | Assertion Consumer Service URL |
| $SLS$ | Single Logout Service URL |
| $\mathcal{V}$ | Verification Result $\in \{true, false\}$ |
| $B64(\cdot)$ | Base64 Encoding Function |
| $SAMLResp$ | SAML Response (Base64 encoded) |
| $XML$ | Decoded SAML Assertion XML |
| $CHL$ | WebAuthn Challenge issued by IdP |
| $RPID$ | WebAuthn Relying Party Identifier |
| $CID$ | Credential ID generated by authenticator |
| $PK$ | Public Key generated by authenticator |
| $SCHL$ | Signed Challenge returned by authenticator |
| $origin$ | Origin string used in WebAuthn client data |
| $clientDataJSON$ | WebAuthn client metadata from browser |
| $AttObj$ | Attestation Object |
| $authData$ | Authenticator Data |
| $sign$ | WebAuthn signature over challenge + metadata |
| $samlAssertion$ | SAML Assertion containing hashed attributes |
| $samlAttributes$ | Extracted hashed attributes from assertion |
| $UserAttribute$ | Stored attribute tuple for each SP |
| $UserSession$ | Stored session state for each SP |
| $CredentialStore$ | Stored credential tuple $\langle CID, PK, AttObj, counter \rangle$ |
| $ChallengeStore$ | Stored challenge $\langle CHL, RPID, createdAt \rangle$ |
| $SsoSession$ | SSO session state $\langle SessionID, NameID, SP, expiry \rangle$ |

Table 4.2: Data Model

| | |
|---|---|
| $Req$ | $= \{reqType, reqData\}$ |
| $Resp$ | $= \{respType, respData\}$ |
| $REQTYPE$ | $= \{registrationReq, regChallengeReq, regVerificationReq,$ $samlLoginReq, credentialCreationReq,$ $credentialAssertionReq, selectiveDisclosureReq\}$ |
| $RESPTYPE$ | $= \{regChallengeResp, regVerificationResp,$ $samlLoginResp, credentialCreationResp,$ $credentialAssertionResp, selectiveDisclosureResp\}$ |
| $registrationReqData$ | $= \langle username, A_h, \sigma \rangle$ |
| $A_h$ | $= \{H(\sigma \parallel attr_i) \mid attr_i \in A\}$ |
| $A$ | $= \{email, firstName, lastName, age, mobile.....\}$ |
| $regChallengeReqData$ | $= \langle A_h, \sigma, username, RPID_{req} \rangle$ |
| $regChallengeRespData$ | $= \langle CHL, RPID \rangle$ |
| $credentialCreationReqData$ | $= \langle CHL, RPID, origin \rangle$ |
| $credentialCreationRespData$ | $= \langle CID, PK, SCHL, AttObj \rangle$ |
| $AttObj$ | $= \langle authData, attStmt \rangle$ |
| $VerificationReqData$ | $= \langle CID, PK, SCHL, clientDataJSON, AttObj \rangle$ |
| $VerificationRespData$ | $= \langle Status, A_h \rangle$ |
| $verify(Data)$ | $= \mathcal{V} \in \{true, false\}$ |
| $samlLoginReqData$ | $= \langle EntityID_{SP}, ACS, RPID_{req} \rangle$ |
| $samlLoginRespData$ | $= \langle CHL, RPID \rangle$ |
| $samlAssertion$ | $= \langle AssertionID, Issuer, NameID, A_h \rangle$ |
| $selectiveDisclosureReqData$ | $= \langle A_s, userInputs \rangle$ |
| $userInputs$ | $= \{(attr_i, value_i) \mid attr_i \in A_s\}$ |
| $selectivedisclosureRespData$ | $= \langle A_{verified}, A_{failed}, A_{private} \rangle$ |
| $CredentialStore$ | $= \langle CID, PK, AttObj \rangle$ |
| $ChallengeStore$ | $= \langle CHL, RPID, createdAt \rangle$ |
| $AttributeStore$ | $= \langle A_{verified}, A_{failed}, A_{private} \rangle$ |
| $SSOSession$ | $= \langle SessionID, NameID, SP, expiry \rangle$ |

**Algorithms:** The following three algorithms define the core operations for registration, SAML authentication, and selective disclosure verification.

Algorithm 1 executes in the user's browser during registration. Each sensitive attribute is concatenated with a fixed salt and hashed using SHA-256 via the Web Crypto API. Only the username and hashed attributes are transmitted to the Identity Provider—plaintext values never leave the browser. A random password is auto-generated since authentication relies on the IdP, not user-managed passwords.

Algorithm 2 implements the SAML SSO authentication flow. The Service Provider redirects the user to the Identity Provider for authentication. Upon successful login, the IdP constructs a SAML assertion containing the user's NameID and all hashed attributes, then posts it to the SP's Assertion Consumer Service. The SP extracts and stores the hashed attributes, creates a session, and redirects the user to the selective disclosure interface where they can choose which attributes to verify and share.

Algorithm 3 implements the core privacy-preserving verification mechanism. For each attribute the user selects to share, the system computes the SHA-256 hash of the user-provided plaintext value concatenated with the same salt used during registration. This computed hash is compared against the stored hash received in

**Algorithm 1** User Registration with Client-Side Hashing

**Require:** $username$, $A = \{email, firstName, lastName, age, mobile, address, profession\}$
**Require:** $\sigma$ = fixed salt string
**Ensure:** User registered with hashed attributes at $IdP$

**function** REGISTRATION($username$, $A$, $\sigma$)
  // **Client-Side (Browser)**
  $A_h \leftarrow \emptyset$
  **for** each $attr_i \in A$ **do**
    **if** $attr_i \neq \emptyset$ **then**
      $combined \leftarrow \sigma \| attr_i$
      $digest \leftarrow H(combined)$ ▷ SHA-256
      $hash_i \leftarrow B64(digest)$
      $A_h \leftarrow A_h \cup \{(\text{“}hashed\_\text{”} + attrName_i,\ hash_i)\}$
    **end if**
  **end for**
  $password \leftarrow$ GENERATERANDOM(20) ▷ Auto-generated
  // **Submit to IdP**
  $formData \leftarrow \langle username,\ password,\ A_h \rangle$
  SUBMITTOIDP($formData$)
  // **IdP Processing**
  $IdP$ stores $\langle username,\ H(password),\ A_h \rangle$ in $DB$
  **return** *success*
**end function**

**Algorithm 2** SAML Login Authentication

**Require:** $U$ initiates login at $SP$
**Ensure:** $U$ authenticated with hashed attributes received

**function** SAMLLOGIN($request$)
  // **Step 1: SP initiates SAML Request**
  $SP \rightarrow BR$: Redirect to $IdP$ with $\langle EntityID_{SP}, ACS \rangle$
  // **Step 2: User authenticates at IdP**
  $U$ enters $username$ at $IdP$ login form
  $IdP$ validates credentials
  // **Step 3: IdP sends SAML Assertion**
  $A_h \leftarrow$ GETUSERHASHEDATTRIBUTES($username$)
  $Assertion \leftarrow \langle AssertionID,\ Issuer,\ NameID,\ A_h \rangle$
  $SAMLResp \leftarrow B64(Assertion)$
  $IdP \rightarrow BR \rightarrow SP$: POST $SAMLResp$ to $ACS$
  // **Step 4: SP processes SAML Response**
  $XML \leftarrow$ BASE64DECODE($SAMLResp$)
  $NameID \leftarrow$ EXTRACTNAMEID($XML$)
  $A_h \leftarrow$ EXTRACTHASHEDATTRIBUTES($XML$)
  // **Step 5: Store and create session**
  **for** each $(attrName,\ hashValue) \in A_h$ **do**
    STOREATTRIBUTE($NameID$, $SP$, $attrName$, $hashValue$)
  **end for**
  $SessionID \leftarrow$ CREATESESSION($NameID$, $SP$)
  **return** REDIRECT(“/selective-disclosure/”)
**end function**

the SAML assertion. If the hashes match, the attribute is verified and the plaintext value is stored, making it visible to the Service Provider. Attributes not selected by the user remain as hashes only (private), ensuring the SP cannot access their values. This mechanism gives users granular control over their data disclosure on a per-SP basis.

**Algorithm 3** Selective Disclosure with Hash Verification

**Require:** $A_h$ = hashed attributes from SAML assertion (stored in session)
**Require:** $A_s$ = user-selected attributes to share
**Require:** $userInputs$ = plaintext values entered by user
**Require:** $\sigma$ = fixed salt (same as registration)
**Ensure:** Verification results with matched/unmatched/private attributes

**function** SELECTIVEDISCLOSURE($A_h$, $A_s$, $userInputs$, $\sigma$)
  $verifiedAttrs \leftarrow \emptyset$
  $failedAttrs \leftarrow \emptyset$
  $privateAttrs \leftarrow \emptyset$
  **for** each $attr_i \in A_h.keys$ **do**
    $attrName \leftarrow attr_i[7:]$ ▷ Remove "hashed_" prefix
    **if** $attrName \notin A_s$ **then**
      **// User chose not to share this attribute**
      $privateAttrs \leftarrow privateAttrs \cup \{attrName\}$
      **continue**
    **end if**
    **if** $attrName \in userInputs$ **then**
      **// Compute hash of user-provided value**
      $inputValue \leftarrow userInputs[attrName]$
      $combined \leftarrow \sigma \| inputValue$
      $computedHash \leftarrow B64(H(combined))$
      **// Compare with stored hash**
      $storedHash \leftarrow A_h[attr_i]$
      **if** $computedHash = storedHash$ **then**
        **// Hash matches — attribute verified**
        $verifiedAttrs \leftarrow verifiedAttrs \cup \{(attrName,\ inputValue)\}$
        UPDATEDB($attrName,\ inputValue,\ isVerified \leftarrow true$)
      **else**
        **// Hash mismatch — verification failed**
        $failedAttrs \leftarrow failedAttrs \cup \{attrName\}$
        UPDATEDB($attrName,\ \emptyset,\ isVerified \leftarrow false$)
      **end if**
    **end if**
  **end for**
  **return** $\langle verifiedAttrs,\ failedAttrs,\ privateAttrs \rangle$
**end function**

### 4.3.2 Protocol Flow and Use Cases

In this section, we explore a number of use-cases to illustrate how a user would interact with the system using a number of protocol flows. We have used sequence diagrams and message tables to represent the protocol flow. We consider three different use-cases: Registration, Login by authenticating user when there is no sso cookies and Login with SSO. We discuss each of the use-cases in the following.

**Registration:** First, Each user must register their credential with the help of WebAuthn. In this use-case we present the respective registration process for a user. There are four entities in this use-case: User's Browser, Service Provider, Hardware Authenticator, and Identity Provider. The protocol for this use-case is represented as a sequence diagram in Figure 4.3 and as step by step messages in Table 4.3. We have also added the detailed explanation.

M1. The browser initiates the registration process by submitting the user information(username,email,phone, address etc.) to the Service Provider. But

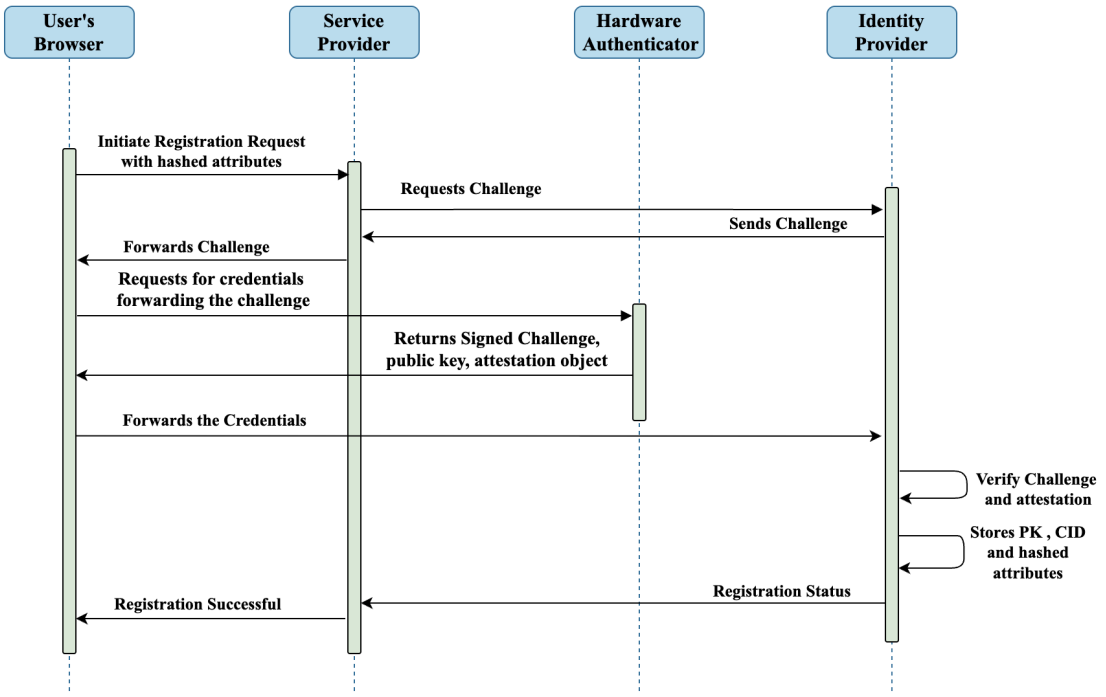



Figure 4.3: Registration Protocol Diagram

the information is submitted after client-side hashing so that Plaintext never leaves the user's browser.

M2. The Service Provider requests a challenge (CHL) and RP ID (Relying Party Identifier) from the Identity Provider. The challenge is a random value which is used to ensure the response is fresh and to prevent replay attacks. The RP ID identifies the web origin (like example.com).

M3. The Identity Provider generates and returns the challenge and RP ID to the Service Provider.

M4. The Service Provider forwards the challenge and RP ID to the browser.

M5. The browser calls createCredential(CHL, RP ID) using the WebAuthn API and sends this info to the authenticator.

M6. The Authenticator generates CID (Credential ID - a unique ID for this credential), SCHL (Signed Challenge), AttObj (Attestation Object-used to verify the device's identity, Public(PK)-Secret key(SK) pair and returns all this to the browser except private/secret key.

M7. The browser forwards the response from the Authenticator, including PK, CID, SCHL, AttObj, clientDataJSON (Encoded client-side information-challenge, origin, etc.) to the Identity Provider for verification.

M8. Identity provider verifies challenge and origin which ensures the challenge matches and that it came from a trusted origin. Identity Provider also verifies Attestation which confirms that the device (Authenticator) is authentic.

M9. Identity Provider stores CID, SCHL and the hashed attributes.

Table 4.3: Registration Protocol Messages

| | |
|---|---|
| $M1$ | $BR \rightarrow SP$: $\{regChallengeReq, registrationData\}$ |
| $M2$ | $SP \rightarrow IP$: $\{regChallengeReq, regChallengeReqData\}$ |
| $M3$ | $IP \rightarrow SP$: $\{regChallengeResp, regChallengeRespData\}$ |
| $M4$ | $SP \rightarrow BR$: $\{regChallengeResp, regChallengeRespData\}$ |
| $M5$ | $BR \rightarrow HA$: $\{credentialCreationReq, credentialCreationReqData\}$ |
| $M6$ | $HA \rightarrow BR$: $\{credentialCreationResp, credentialCreationRespData\}$ |
| $M7$ | $BR \rightarrow IP$: $\{VerificationReq, VerificationReqData\}$ |
| $M8$ | $IP \rightarrow IP$: $verify(VerificationReqData)$ |
| $M9$ | $IP \rightarrow IP$: $CredentialStore$ |
| $M10$ | $IP \rightarrow SP$: $\{VerificationResp, VerificationRespData\}$ |
| $M11$ | $SP \rightarrow BR$: $\{VerificationResp, VerificationRespData\}$ |

M10. Identity Provider sends the registration status back to the Service Provider.

M11. The browser receives a confirmation that registration was successful and informs the user.

**Login:** Our second use-case represents how an user can authenticate himself with the help of WebAuthn. In this use case we present the respective login process for a user. There are four entities in this use case: User's Browser, Service Provider, Hardware Authenticator and Identity Provider. The protocol for this use-case is represented as a sequence diagram in Figure 4.4 and as step by step messages in Table 4.4. We have also added the detailed explanation.

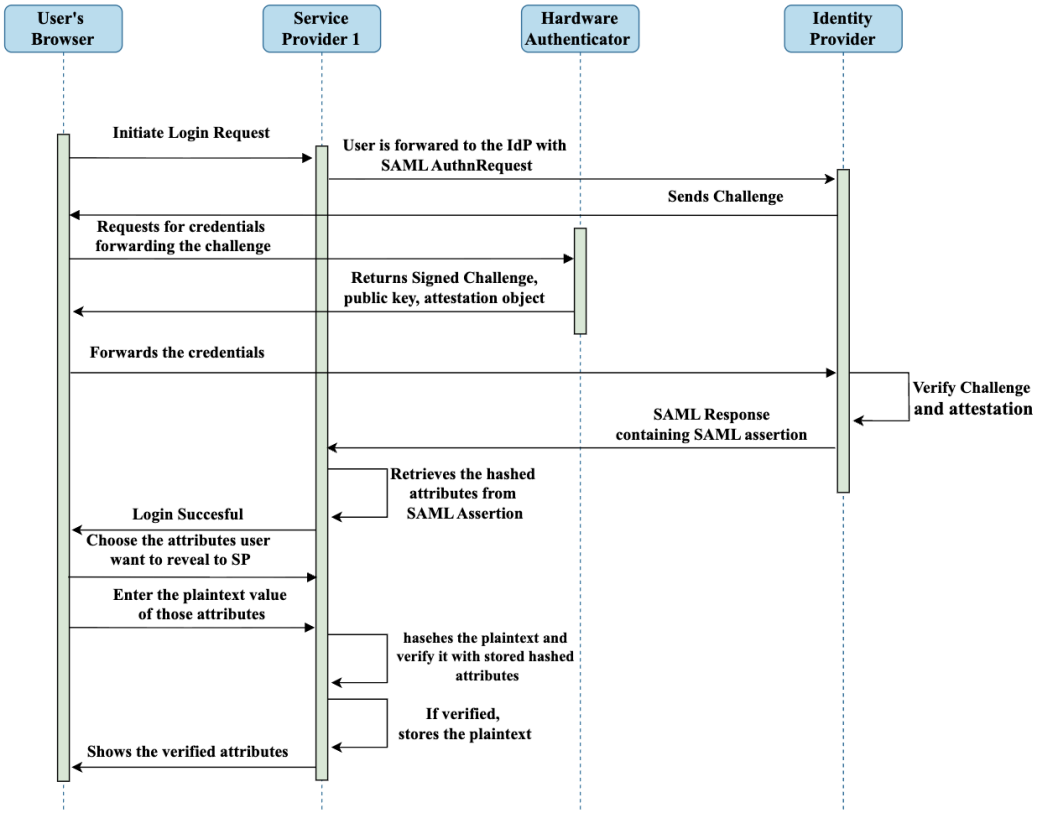



Figure 4.4: Login to SP1 Protocol Diagram

M1. The user accesses SP1 and requests authentication. Since no local session exists, SP1 generates a SAML Authentication Request (AuthnRequest) and redirects the user's browser to the Identity Provider (IdP).

M2. Since no local session exists, SP1 generates a SAML Authentication Request (AuthnRequest) and redirects the user's browser to the Identity Provider (IdP). The Authnrequest includes the SP1 EntityID, the Assertion Consumer Service (ACS) URL, requested attributes, timestamps, and relevant SAML metadata.

M3. The IdP receives the request and determines that no SSO session cookie (SSO-Cookie) is present. Because no active SSO session exists, the IdP initiates WebAuthn authentication. The IdP generates a challenge (CHL) and the Relying Party Identifier (RPID). These parameters are returned to the browser.

M4. The browser calls createCredential(CHL, RP ID) using the WebAuthn API and sends this info to the authenticator.

M5. The Authenticator generates CID (Credential ID - a unique ID for this credential), SCHL (Signed Challenge), AttObj (Attestation Object-used to verify the device's identity) and returns all this to the browser.

M6. The browser forwards the response from the Authenticator, including CID, SCHL, AttObj, clientDataJSON (Encoded client-side information-challenge, origin, etc.) to the Identity Provider for verification.

M7. Identity Provider fetches the CID, SCHL of that user from that database. Identity Provider verifies attestation which confirms that the device (Authenticator) is authentic. Identity provider verifies challenge and origin which ensures the challenge matches and that it came from a trusted origin.

M8. Once authenticated, the IdP creates a persistent SSO session and issues a SAML Response containing a signed SAML Assertion. The assertion includes the user's authenticated status, the session index for SSO, and hashed user attributes that were stored during registration.

M9. SP1 validates the XML signature, audience restrictions, and timestamp constraints. SP1 then extracts the hashed attribute values from the SAML assertion and stores them in the user's temporary session state.

M10. The browser receives a confirmation that login was successful and informs the user. SP1 prompts the user to choose which attributes to disclose (e.g., email or phone number).

M11. The user chooses the attributes that they want to disclose.

M12. The user provides the plaintext values corresponding to the selected attributes.

M13. SP1 computes the hash of each user-supplied attribute and verifies: hash(user input) == saml hash value.Verification success confirms the attribute belongs to the user.

M14. Once verified, SP1 stores the plaintext attributes in the user's profile.

Table 4.4: Login to SP1 Protocol Messages

| | |
|---|---|
| $M1$ | $BR \rightarrow SP$: $\{samlLoginReq, samlLoginReqData\}$ |
| $M2$ | $SP \rightarrow IP$: $\{samlLoginReq, samlLoginReqData\}$ |
| $M3$ | $IP \rightarrow BR$: $\{samlLoginResp, samlLoginRespData\}$ |
| $M4$ | $BR \rightarrow HA$: $\{credentialCreationReq, credentialCreationReqData\}$ |
| $M5$ | $HA \rightarrow BR$: $\{CredentialCreationResp, credentialCreationRespData\}$ |
| $M6$ | $BR \rightarrow IP$: $\{VerificationReq, VerificationReqData\}$ |
| $M7$ | $IP \rightarrow IP$: $verify(VerificationReqData)$ |
| $M8$ | $IP \rightarrow SP$: $\{samlLoginResp, samlAssertion\})$ |
| $M9$ | $SP \rightarrow SP$: $\langle AssertionID, Issuer, NameID, A_h \rangle$ |
| $M10$ | $SP \rightarrow BR$: $\{VerificationResp, VerificationRespData\}$ |
| $M11$ | $BR \rightarrow SP$: $\{selectiveDisclosureReq, selectiveDisclosureReqData\}$ |
| $M12$ | $BR \rightarrow SP$: $userInputs$ |
| $M13$ | $SP \rightarrow SP$: $verify(userInputs)$ |
| $M14$ | $SP \rightarrow SP$: $AttributeStore$ |
| $M15$ | $SP \rightarrow BR$: $\{selectivedisclosureResp, selectivedisclosureRespData\}$ |

M15. SP1 displays the verified attributes and completes the login flow.

**Login with SSO:** Once the user is authenticated at SP1, the IdP maintains an SSO session. When the user later attempts to access SP2, authentication proceeds without WebAuthn. The protocol for this use-case is represented as a sequence diagram in Figure 4.5 and as step by step messages in Table 4.5. We have also added the detailed explanation.

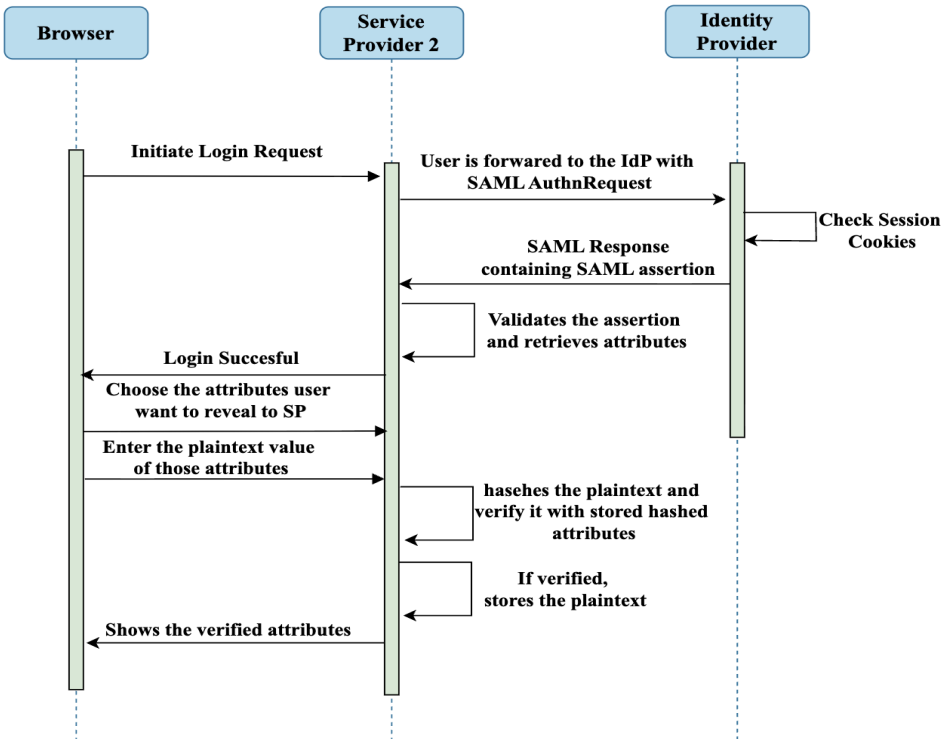



Figure 4.5: Login with SSO Protocol Diagram

M1. The user visits SP2 and activates the login procedure.

M2. SP2 constructs a SAML AuthnRequest and redirects the browser to the IdP.

M3. The IdP detects an active SSO session via its SSO cookie. Therefore, no WebAuthn challenge is required.

M4. The IdP constructs a new SAML Response containing: the existing SSO session index, authentication context from the prior WebAuthn login and hashed user attributes identical to those released to SP1. Then send the response to SP2.

M5. SP2 verifies the SAML signature, timestamps, and audience conditions, then extracts the hashed attribute values from the assertion.

M6. The browser receives a confirmation that login was successful and informs the user. SP2 prompts the user to choose which attributes to disclose (e.g., email or phone number).

M7. The user chooses the attributes that they want to disclose.

M8. The user provides the plaintext values corresponding to the selected attributes.

Table 4.5: Login with SSO Protocol Messages

| | |
|---|---|
| $M1$ | $BR \rightarrow SP$: $\{samlLoginReq, samlLoginReqData\}$ |
| $M2$ | $SP \rightarrow IP$: $\{samlLoginReq, samlLoginReqData\}$ |
| $M3$ | $IP \rightarrow IP$: $SSOsession$ |
| $M4$ | $IP \rightarrow SP$: $\{samlLoginResp, samlAssertion\})$ |
| $M5$ | $SP \rightarrow SP$: $\langle AssertionID, Issuer, NameID, A_h \rangle$ |
| $M6$ | $SP \rightarrow BR$: $\{VerificationResp, VerificationRespData\}$ |
| $M7$ | $BR \rightarrow SP$: $\{selectiveDisclosureReq, selectiveDisclosureReqData\}$ |
| $M8$ | $BR \rightarrow SP$: $userInputs$ |
| $M9$ | $SP \rightarrow SP$: $verify(userInputs)$ |
| $M10$ | $SP \rightarrow SP$: $AttributeStore$ |
| $M11$ | $SP \rightarrow BR$: $\{selectivedisclosureResp, selectivedisclosureRespData\}$ |

M9. SP2 computes the hash of each user-supplied attribute and verifies: hash(user input) == saml hash value.Verification success confirms the attribute belongs to the user.

M10. Once verified, SP2 stores the plaintext attributes in the user's profile.

M11. SP2 displays the verified attributes and completes the login flow.

## 4.4 Implementation of Selected Design

The proposed system is a web-based Single Sign-On (SSO) solution that implements hash-based selective attribute disclosure combined with WebAuthn-based passwordless authentication. It comprises a Django-based backend for Service Providers, Keycloak as the Identity Provider (IdP), and a custom registration theme with client-side hashing and YubiKey/NFC security key integration. The system enables users to authenticate using SAML 2.0 with hardware security keys while maintaining control over which attributes are shared with each service provider.

### 4.4.1 System Architecture

The implementation follows a federated identity model with four main components:

- **Identity Provider (IdP):** Keycloak server managing user identities, WebAuthn credentials, and SAML assertions
- **Service Providers (SP):** Two Django applications (SP1 on port 8001, SP2 on port 8002) consuming SAML assertions
- **Client:** Web browser handling client-side hashing, WebAuthn API interactions, and user interface
- **Hardware Authenticator:** YubiKey or NFC-supported security device storing private keys for passwordless authentication

### 4.4.2 Backend Implementation

The backend is built using Python-based Django framework with Django REST Framework extensions. The core packages used include:

- `python3-saml` (OneLogin): For SAML 2.0 protocol handling, assertion parsing, and metadata generation
- `cryptography`: For SHA-256 hashing and attribute verification
- `django.contrib.sessions`: For session management and SAML state persistence
- `sqlite3`: For storing user attributes, service provider configurations, and verification records

#### SAML Configuration

The Service Provider configuration defines essential SAML parameters including the SP entity ID, Assertion Consumer Service (ACS) URL with HTTP-POST binding, Single Logout Service (SLS) URL with HTTP-Redirect binding, and persistent NameID format. The IdP configuration specifies the Keycloak realm entity ID and Single Sign-On Service URL.

#### Hash Verification Module

The hash verification module handles attribute verification using SHA-256 with a fixed salt to ensure consistency between client-side and server-side hashing. The module provides two primary functions: `hash_attribute()` which concatenates the salt with the attribute value and returns a Base64-encoded SHA-256 hash, and `verify_attribute()` which computes the hash of a provided value and compares it against the stored hash.

#### Database Models

The system uses three primary database models shown in 4.6:

Table 4.6: Database Models

| Model | Fields |
|---|---|
| ServiceProvider | sp_entity_id, sp_name, sp_url |
| UserAttribute | saml_name_id, username, service_provider (FK), attribute_name, hash_value, plain_value, is_verified, is_shared, verified_at |
| UserSession | saml_name_id, service_provider (FK), session_key, is_active |

#### SAML Callback Handler

The SAML callback endpoint processes incoming SAML assertions by Base64 decoding the response, parsing the XML using ElementTree with SAML namespaces, extracting all attributes including NameID and WebAuthn credential references, identifying hashed attributes by their prefix, storing them in the database, and redirecting to the selective disclosure interface.

### 4.4.3 Frontend Implementation

#### Keycloak Custom Theme

A custom Keycloak registration theme was developed using FreeMarker Template Language (FTL) and JavaScript. The registration form contains visible input fields for plaintext entry (processed locally but not submitted in plaintext), hidden fields for hashed and encrypted values, WebAuthn credential registration interface, and a custom button that triggers client-side hashing and YubiKey enrollment.

#### WebAuthn Integration

The system implements WebAuthn (Web Authentication) API for passwordless authentication using hardware security keys. The WebAuthn module performs the following operations:

1. **Credential Creation:** During registration, the browser calls `navigator.credentials.create` with PublicKeyCredentialCreationOptions specifying the relying party (Keycloak), user information, supported algorithms (ES256, RS256), and authenticator requirements (cross-platform for external security keys).

2. **Key Pair Generation:** The YubiKey or NFC device generates an asymmetric key pair. The private key remains securely stored on the hardware authenticator and never leaves the device.

3. **Public Key Storage:** The public key and credential ID are transmitted to Keycloak and stored as user attributes for future authentication verification.

4. **Authentication:** During login, the user's security key signs a server-generated challenge using the stored private key. The IdP verifies the signature using the registered public key.

#### Client-Side Hashing and Encryption

The client-side module implements a two-layer security approach:
**Layer 1 - Attribute Hashing:** All sensitive attributes (email, firstName, lastName, age, mobile, address, profession) are hashed using SHA-256 with a fixed salt via the Web Crypto API. The hash function concatenates the salt with each attribute value, computes the digest using `crypto.subtle.digest()`, and encodes the result in Base64.
**Layer 2 - YubiKey Encryption:** For additional security, attributes can be encrypted using AES-256-GCM with a key derived from the WebAuthn credential:

1. Generate a random AES-256-GCM symmetric key
2. Encrypt each sensitive attribute with this symmetric key
3. Register WebAuthn credential on YubiKey (user touches the key)
4. Derive a wrapping key from: Credential ID + Public Key + Random Salt using PBKDF2
5. Wrap (encrypt) the symmetric key with the derived wrapping key
6. Store: encrypted attributes + wrapped key + credential ID + public key + salt

### 4.4.4 Identity Provider Implementation

Keycloak is configured as the SAML 2.0 Identity Provider with WebAuthn support and custom attribute mappers.

#### Realm Configuration

The `demo` realm is configured with SAML 2.0 protocol enabled, WebAuthn passwordless authentication flow, two SAML clients (django-sp1, django-sp2) for multi-SP demonstration, custom user attributes for storing hashes and WebAuthn credentials, and attribute mappers for including hashes in assertions.

#### WebAuthn Authenticator Configuration

Keycloak is configured to support WebAuthn with the settings in table 4.7.

Table 4.7: WebAuthn Configuration in Keycloak

| **Parameter** | **Value** |
|---|---|
| Relying Party Name | Keycloak SSO |
| Signature Algorithms | ES256, RS256 |
| Authenticator Attachment | Cross-platform (external keys) |
| User Verification | Preferred |
| Attestation Conveyance | Direct |
| Timeout | 60 seconds |

#### SAML Client Configuration

Each Django Service Provider is registered as a SAML client in Keycloak shown in 4.8:

Table 4.8: Keycloak SAML Client Configuration

| Parameter | Value |
|---|---|
| Client ID | django-sp1 / django-sp2 |
| Client Protocol | SAML |
| Assertion Consumer Service URL | http://localhost:800X/api/saml/callback/ |
| Single Logout Service URL | http://localhost:800X/api/saml/sls/ |
| Name ID Format | persistent |

#### Hashed Attribute Mappers

Custom protocol mappers in table 4.9 include hashed user attributes in SAML assertions:

Table 4.9: SAML Attribute Mappers

| User Attribute | SAML Attribute Name |
|---|---|
| hashed_email | hashed_email |
| hashed_firstName | hashed_firstName |
| hashed_lastName | hashed_lastName |
| hashed_age | hashed_age |
| hashed_mobile | hashed_mobile |
| hashed_address | hashed_address |
| hashed_profession | hashed_profession |
| webauthn_credential_id | webauthn_credential_id |

### 4.4.5 System Behavior

The complete authentication and selective disclosure flow operates as follows:

1. **Registration:** User enters attributes in Keycloak registration form. Client-side JavaScript hashes all sensitive attributes using SHA-256. User touches YubiKey/NFC device to create WebAuthn credential. The public key is stored on the IdP while the private key remains on the hardware authenticator. Only username, hashed values, and WebAuthn credential metadata are stored on the IdP.

2. **WebAuthn Authentication:** When accessing a Service Provider, user is redirected to Keycloak for authentication. User enters username and touches their security key. The YubiKey signs the server challenge using the private key. Keycloak verifies the signature using the stored public key.

3. **SAML Assertion Generation:** Upon successful WebAuthn authentication, Keycloak generates a SAML assertion containing hashed attributes and sends it to the SP.

4. **Attribute Extraction:** The SP extracts hashed attributes from the SAML assertion XML and stores them in the database associated with the user's NameID.

5. **Selective Verification:** User is presented with a verification interface where they can select which attributes to share, re-enter plaintext values for selected attributes, or skip verification to keep all attributes private.

6. **Hash Comparison:** For each selected attribute, the SP hashes the user-provided value and compares it with the stored hash. Matching hashes indicate successful verification.

7. **Attribute Storage:** Verified attributes are stored in plaintext in the SP database (marked as verified and shared). Unverified attributes remain as hashes only.

8. **Single Logout:** Logout initiates SAML SLO, terminating sessions across all Service Providers.

### 4.4.6 Deployment Configuration

The system uses Docker Compose for containerized deployment. Keycloak runs on port 8080 with admin credentials, custom theme mounted from the local directory, WebAuthn authentication flow enabled, and realm configuration auto-imported on startup. The Django Service Providers run on ports 8001 and 8002 respectively using Django's built-in development server.

### 4.4.7 Use Case

As discussed in the protocol flow and architecture, the use case of the actual implementation is presented below. The sequence of User Interfaces that an end user experiences during registration, authentication, and selective disclosure is demonstrated in this section.

**Used Technologies:** In the development, we have used Python-based Django framework for the backend Service Providers, Keycloak as the Identity Provider for SAML 2.0 authentication, JavaScript with Web Crypto API for client-side SHA-256 hashing, FreeMarker Template Language for custom Keycloak themes, SQLite for database storage, and Docker for containerized deployment. Below, we are presenting and explaining the implemented user interfaces.

This is our SP1 Home Page (shown in Figure 4.6) where users will be initially directed to. Users can view their authentication status and click "Login with SAML" to initiate the SSO flow.

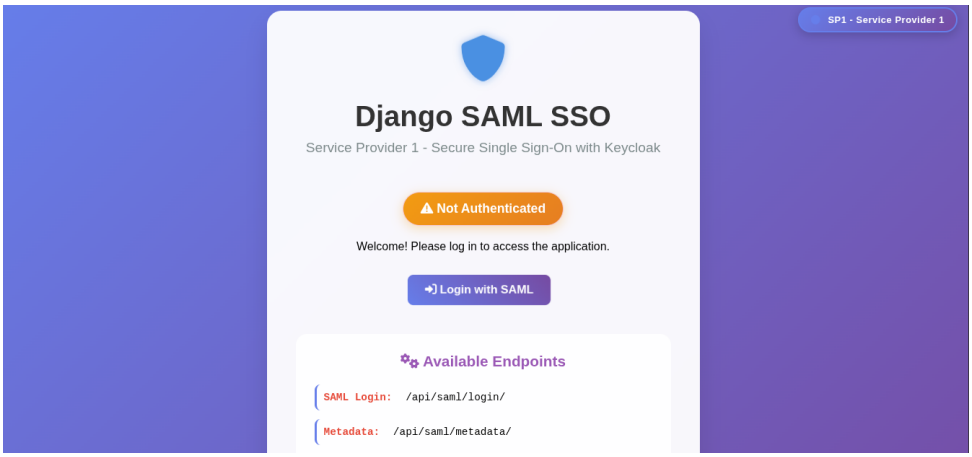



Figure 4.6: SP1 Home Page

In Figure 4.7 and Figure 4.8, we are showcasing our Registration and Client-Side Hashing interface.


Figure 4.7: Registration Page

Here, users enter their username and sensitive attributes (email, first name, last name, age, mobile, address, profession). All attributes are hashed client-side using SHA-256 before submission—the Identity Provider never sees plaintext values.

Next, in Figure 4.9, returning users authenticate via the Keycloak login page using their username.

After successful SAML authentication, users are redirected to the Selective Disclosure page (Figure 4.10). Here, users can choose which attributes to share with the Service Provider by selecting checkboxes and re-entering their plaintext values for hash verification.

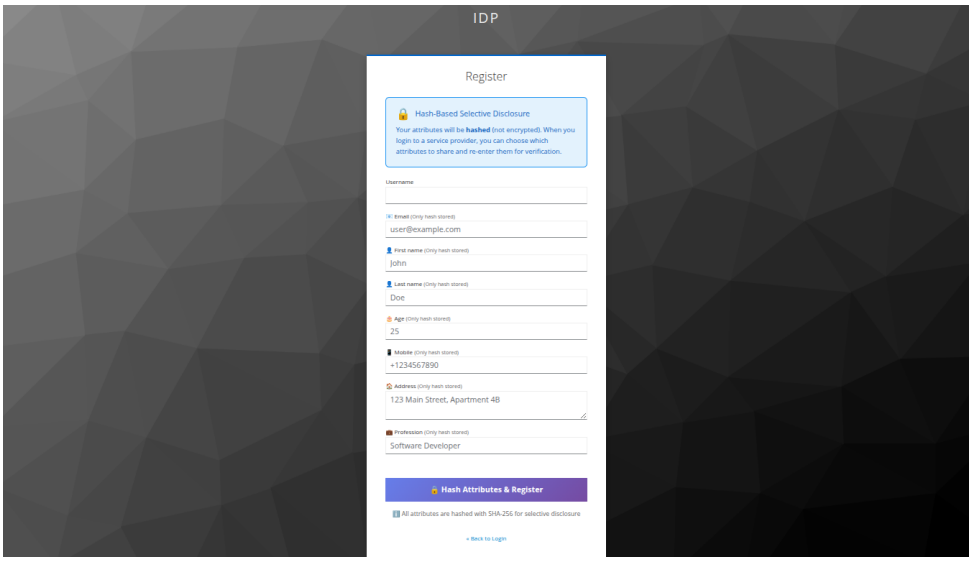



Figure 4.8: Client-Side Hashing Progress

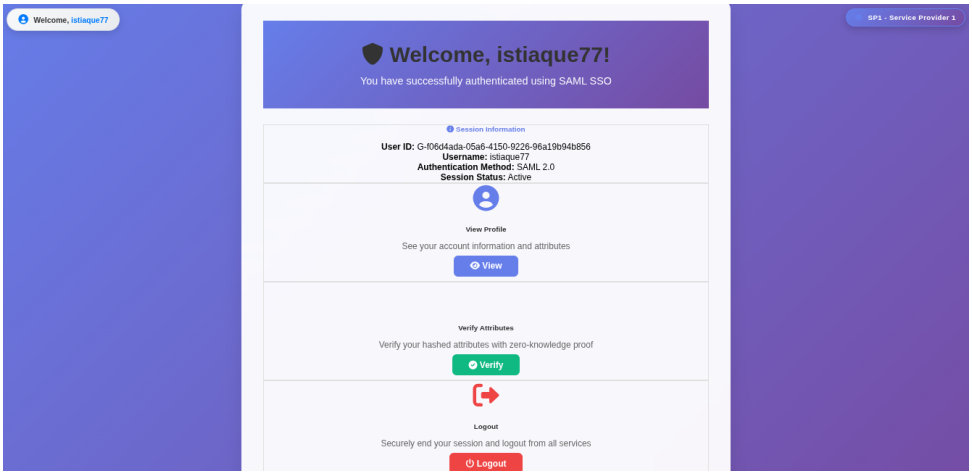



Figure 4.9: Login Page

In Figure 4.11, we showcase the Verification Results interface. The system displays which attributes were successfully verified (hash matched), which failed verification, and which attributes the user chose to keep private.

Next, in Figure 4.12, the User Profile page displays verified attributes in plaintext and unverified attributes as truncated hashes. This clearly distinguishes between shared and private information.

Finally, in Figure 4.13, we demonstrate the multi-SP capability where users can authenticate to Service Provider 2 independently. Each SP requires separate selective disclosure decisions, ensuring per-SP privacy control.

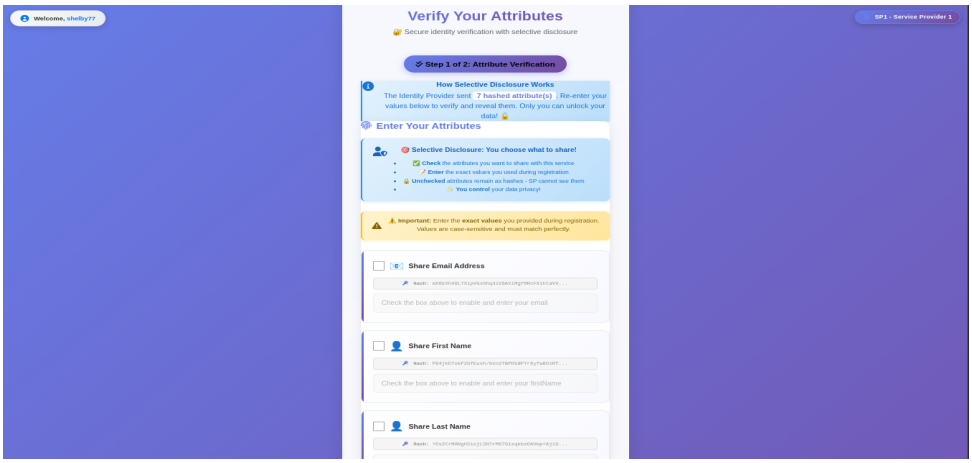



Figure 4.10: Selective Disclosure Page

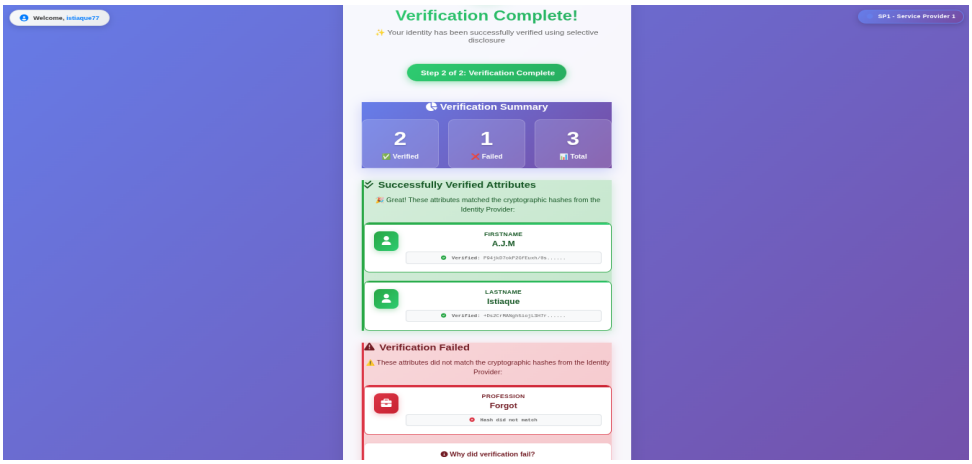



Figure 4.11: Verification Results Page

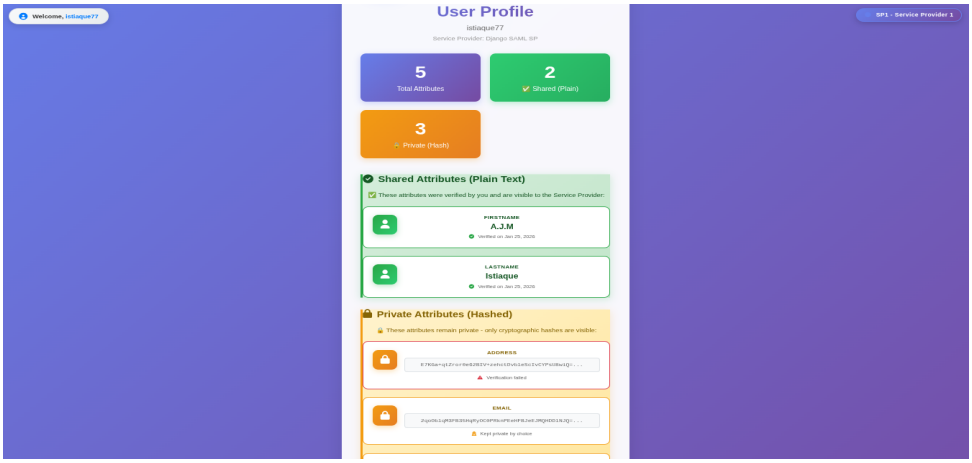



Figure 4.12: User Profile Page

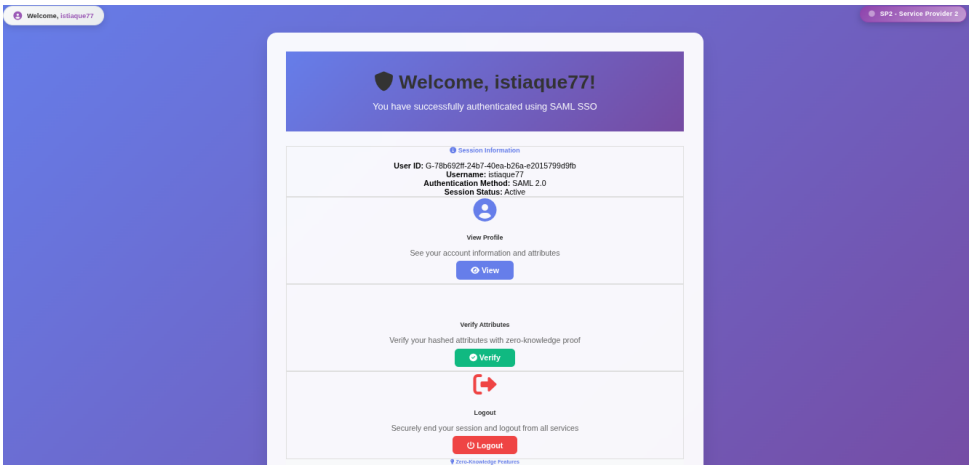



Figure 4.13: SP2 Authenticated Page

### 4.4.8 Security Features

The implemented system provides the following security guarantees:

- **Zero-Knowledge Storage:** The Identity Provider stores only cryptographic hashes, never plaintext attribute values.
- **Passwordless Authentication:** Users authenticate using hardware security keys (YubiKey/NFC), eliminating password-related vulnerabilities.
- **Private Key Protection:** The private key never leaves the hardware authenticator, providing strong protection against key theft.
- **Client-Side Hashing:** Sensitive data is hashed in the user's browser using Web Crypto API before any network transmission.
- **Selective Disclosure:** Users explicitly choose which attributes to reveal to each Service Provider.
- **Per-SP Isolation:** Each Service Provider maintains independent attribute verification records; sharing with one SP does not affect others.

### 4.4.9 Technology Stack Summary

Table 4.10 summarizes the technologies and frameworks used in the implementation.

Table 4.10: Technology Stack

| Component | Technology | Version |
|---|---|---|
| Backend Framework | Django | 4.x |
| SAML Library | python3-saml (OneLogin) | Latest |
| Identity Provider | Keycloak | 23.0.7 |
| Authentication | WebAuthn API | Level 2 |
| Hardware Authenticator | YubiKey / NFC Device | – |
| Database | SQLite | 3.x |
| Hashing Algorithm | SHA-256 | – |
| Client-Side Crypto | Web Crypto API | – |
| Containerization | Docker | Latest |
| Programming Languages | Python, JavaScript | 3.8+, ES6 |

# Chapter 5

# Result Analysis

The current chapter is based on the results of the performance evaluation of our implemented system. It also presents the analysis of requirements given in Chapter 3. The main goal to evaluating the performance is to find out the limitations which can be used to make specific enhancements in future work.

## 5.1 Performance Evaluation

This section focuses on the performance evaluation of our implemented system. It has a detailed experimental setup along with the results of Registration API, Service Provider 1 Log in API, Verify Attributes API. To represent the results after conducting a performance evaluation of the mentioned APIs line graph and tables are shown. This section therefore shows the statistical and mathematical insights on how our developed system is going to perform under varying numbers of concurrent users.

### 5.1.1 Experimental Setup

In order to evaluate our integrated system, A controlled experimental environment was set up and established. The environment was set up as close as possible to real world operation by placing varying levels of concurrent users on the system APIs. Performance metrics such as response latency, throughput and error rate were gathered in each cycle of the test.

#### Hardware Environment

The tests were conducted on a MacBook Air with Apple Silicon (M2, 2022) with the following specifications:

- **Processor and GPU:** Apple M2 chip with 8-core CPU (4 performance cores + 4 efficiency cores) and 10-core GPU
- **Memory (RAM):** 16 GB unified memory
- **Storage:** 512 GB SSD
- **Networking:** Wi-Fi 6 (802.11ax) and Bluetooth 5.3
- **Operating System:** macOS Ventura 13.x

### 5.1.2 Experimental Scenario

The performance evaluation was performed in a controlled single machine environment in order to ensure the repeatability of the experiments and to eliminate the influence of external network variations. Apache JMeter tool was used as a load testing tool to generate and manage the concurrent user requests.
A single load scenario was set up to observe the system behaviour in continuous and growing user demand. Virtual users are introduced gradually using a fixed ramp up period of 10 seconds, which enables the load to be increased smoothly and without sudden spikes in the traffic.
The experiments were conducted with different numbers of concurrent users, namely 200, 400, 600, 800, 1000, and 1200 users, in order to analyze the system scalability under the increasing workload conditions. For each configuration of user load, three separate independent test executions were performed in the same conditions to guarantee reliability and statistical consistency of the results. Each time an execution is called one test cycle.
Performance evaluations were performed on several API endpoints of the system, such as the Registration, Login and Verify Attributes, using the same workload patterns in order to fairly compare the results. During each cycle of testing, important performance characteristics, including average response latency and average throughput, as well as the error rate were recorded.
The results shown in this work are the mean values obtained during the three test cycles of each load level. These averaged results offer a stable and reliable foundation for the performance analysis of the implemented APIs in terms of their performance, scalability for varying levels of concurrency.

### 5.1.3 Test Results and Analysis for Registration API

Table 5.1 shows that the average latency of the Registration API continues to rise gradually with increase in the number of simultaneous users. Average latency increases gradually between about 45 ms to about 78 ms. After it begins to increase rapidly to about 435 ms at 1200 users, indicating the system is nearing its performance peak at high concurrency. Simultaneously, with a steady rise in the throughput, with a figure of approximately 20 requests/sec at 200 users to approximately 125 requests/sec at 1200 users indicating that the system remains capable of handling more requests, even when loaded to the maximum. Although the latency has increased dramatically when reaching peak concurrency, the error rate is 0 in all the 3 test cycles, which means that the API is still functional and not requesting failures.

Table 5.1: Performance Metrics for Registration API

| **Users** | **Avg Latency (ms)** | **Avg Throughput (req/s)** | **Error (%)** |
|---|---|---|---|
| 200 | 44.57 | 20.10 | 0.0 |
| 400 | 48.81 | 40.11 | 0.0 |
| 600 | 53.55 | 58.94 | 0.0 |
| 800 | 65.85 | 77.04 | 0.0 |
| 1000 | 77.80 | 100.11 | 0.0 |
| 1200 | 434.96 | 125.04 | 0.0 |

The API performance in the throughput graph 5.1, depicts the definite increasing trend in throughput with the rise of the number of concurrent users. There is an average improvement in throughput with a rise in user load between 200 to 1200 simultaneous users and this shows that the system can efficiently utilize the available resources under increased load conditions.

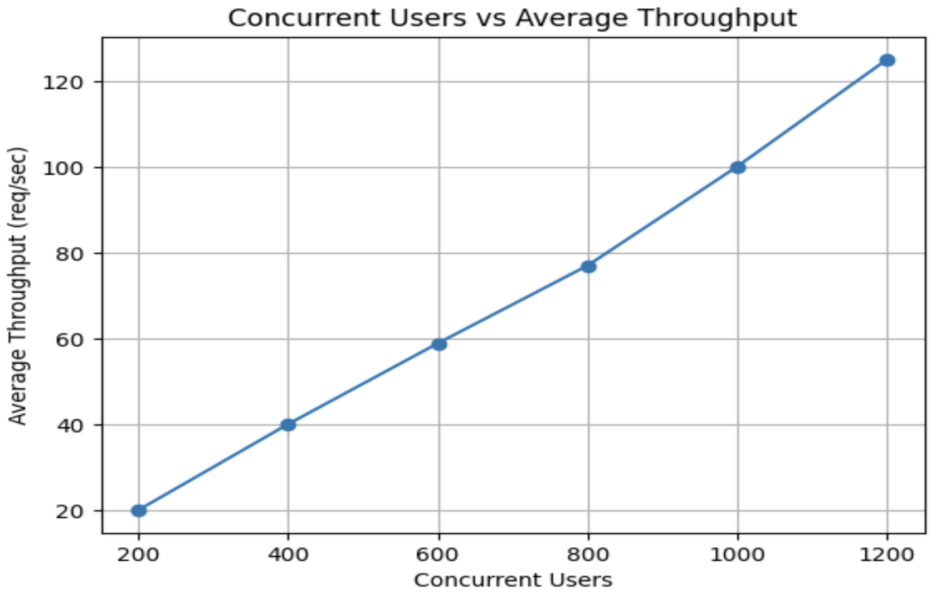



Figure 5.1: Throughput graph of Registration API

In the latency graph 5.2, average latency increases gradually between approximately 45 ms and approximately 78 ms with increased number of simultaneous users between 200 and approximately 1000 users and at the time of 1200 users, the latency skyrockets to more than 400 ms which means the system is exceeding its capacity, leading to queuing delays.

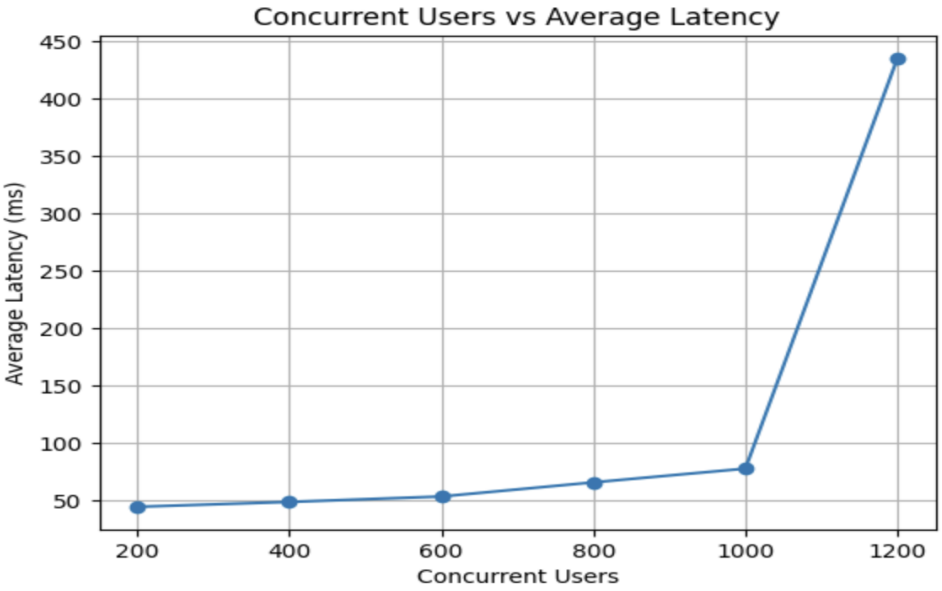



Figure 5.2: Latency graph of Registration API

### 5.1.4 Test Results and Analysis for Service Provider 1 Login API

The table 5.2 indicates the change in system performance with concurrent users growing to 200 to 1200. Throughput is increasing steadily at a rate of approximately 60 up to 370 requests per second, indicating that the system is able to process additional total work at a higher load, whereas average latency remains fairly stable (approximately in 14 ms) up to 1000 users, at 1200 users average latency increases drastically to around 67 ms, which is an indication that the system is approaching or slightly surpassing its capacity.

Table 5.2: Performance Metrics for Service Provider 1 Login API

| **Users** | **Avg Latency (ms)** | **Avg Throughput (req/s)** | **Error (%)** |
|---|---|---|---|
| 200 | 19.22 | 60.12 | 0.0 |
| 400 | 16.46 | 119.96 | 0.0 |
| 600 | 19.58 | 176.37 | 0.0 |
| 800 | 15.72 | 230.64 | 0.0 |
| 1000 | 14.45 | 299.80 | 0.0 |
| 1200 | 66.95 | 370.81 | 0.0 |

The throughput graph 5.3 reveals a constant and near linear rise in the average throughput with the number of concurrent users increasing between 200 and 1200, with a corresponding rise in requests per second, which were approximately 20 requests per second but increased to nearly 125 requests/second. It means that the scaling of the system in terms of the request handling capacity is successful and that it is possible to introduce more users and it can process more requests without indication of an initial saturation or plateau within the chosen test range.

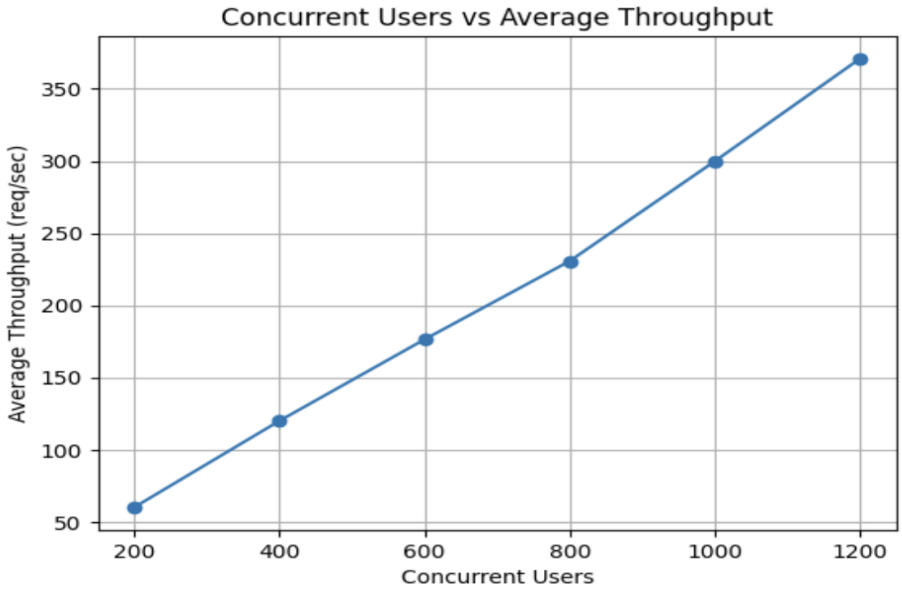



Figure 5.3: Throughput graph of Verify Login API

The graph 5.4 indicates the variation in the average latency with the number of used users. The mean latency does not change significantly within this range, going

between 14 ms and 20 ms as the user base increases to 200-1000 users, indicating that the system is managing the increasing load effectively in this region. The steep increase in latency to approximately 67 ms at 1200 simultaneous users and thus its performance is heavily impaired at this level. All in all, the graph indicates a good scalability on the way to approximately 1000 users, and after that the performance will be bottlenecked.

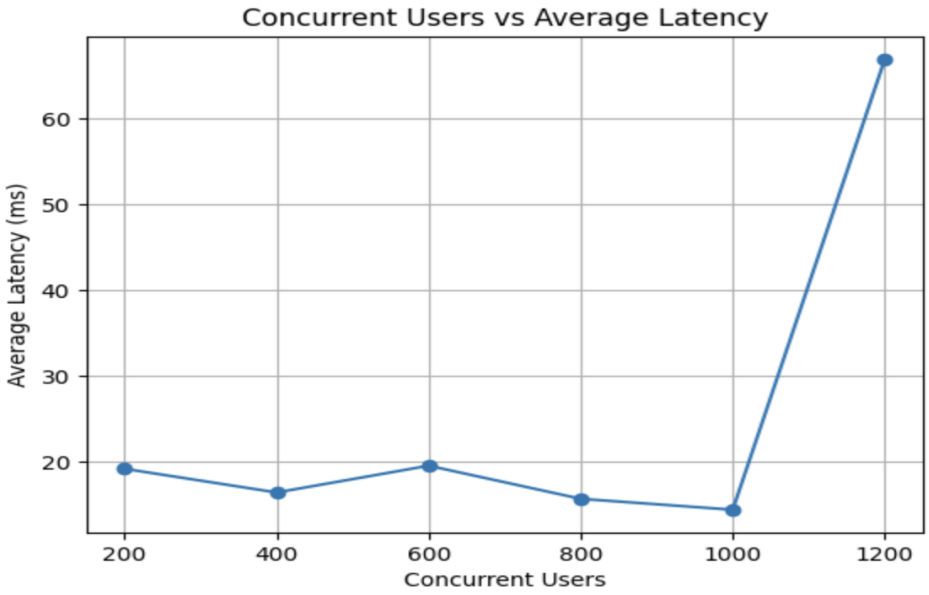



Figure 5.4: Latency graph of Login API

### 5.1.5 Test Results and Analysis for Verify Attributes API

In table 5.3, with the rise of concurrent users from 200 to 1200, it is observed that the average throughput increases steadily between 20 to 125 requests per second, thus the system will sustain increased requests with increase in load. Latency first drops to 22 ms at 400 users, and then begins to rise, to approximately 93 ms at 1000 users, and 2500 ms at 1200 users. This steep rise indicates an overcapacity of the system to accommodate more than 1000 users leading to massive queues and delays.

Table 5.3: Performance Metrics for Verify Attributes API

| **Users** | **Avg Latency (ms)** | **Avg Throughput (req/s)** | **Error (%)** |
|---|---|---|---|
| 200 | 31.96 | 20.11 | 0.0 |
| 400 | 21.55 | 40.11 | 0.0 |
| 600 | 28.35 | 58.94 | 0.0 |
| 800 | 45.69 | 77.05 | 0.0 |
| 1000 | 93.47 | 100.13 | 0.0 |
| 1200 | 2580.25 | 125.13 | 0.0 |

The throughput graph 5.5 that illustrates the multiple cycles and the average shows the steady and almost linear growth of throughput as concurrency increases in all test cycles. This consistency implies that the behavior of the system is predictable

and stable under load, and that the number of requests being served per second will remain constant.

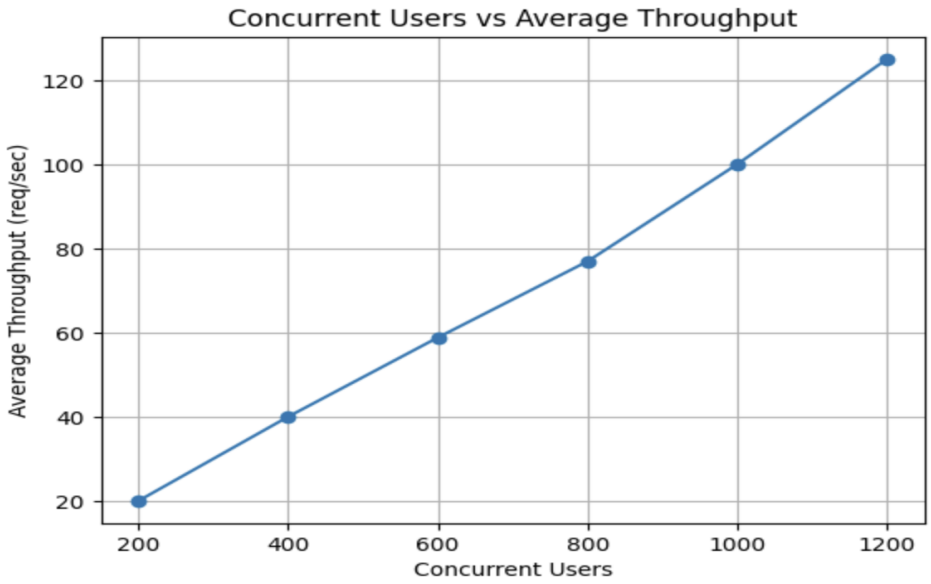



Figure 5.5: Throughput graph of Verify Attributes API

The latency graph 5.6 indicates fairly small and steady response times to a given user load up to approximately 800 to 1000 simultaneous users, with a drastic increase to 1200 users. This is an indication of a definite level of performance, anything below that is within acceptable limits of the system to handle the oncoming requests but when the number of concurrency surpasses the capacity of the system. Then requests get piled up at an alarming rate and response time grows exponentially.

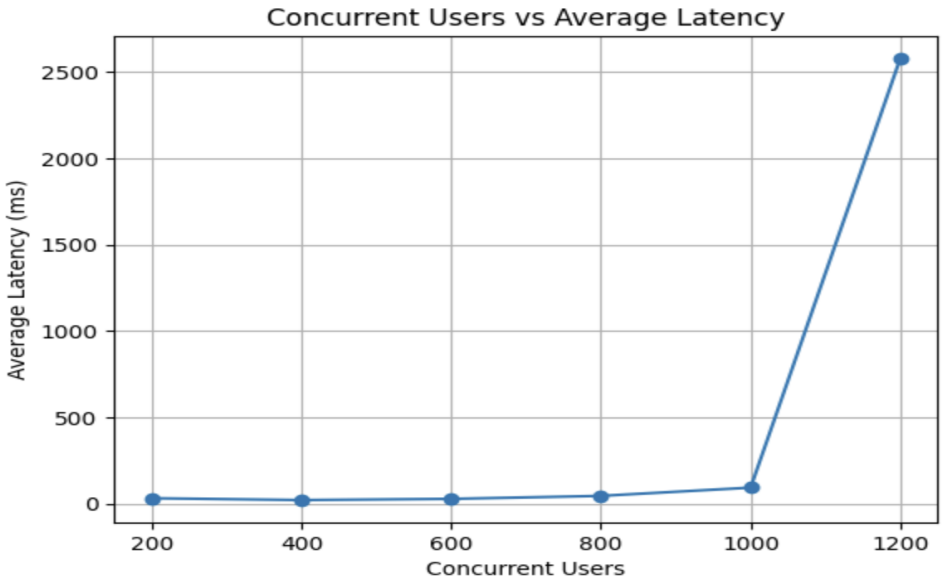



Figure 5.6: Latency graph of Verify Attributes API

## 5.2 Analysis of Design Solutions

The proposed system design was evaluated based on their ability to meet functional , security and privacy requirements of the proposed architecture.

### 5.2.1 Functional Requirement Analysis

The implementation was progressed keeping the seven functional requirements (F1-F7) in mind mentioned in 3.1.2.1 . The first one (F1) is that the system should implement a Single Sign-On (SSO) system into the WebAuthn system, where authentication will be passwordless, and cryptographically secure. This is completely met in the applied system. WebAuthn is used as the initial authenticator at the Identity Provider (IdP) and when identification is done, the IdP issues a SAML assertion granting authenticated access to the involved Service Providers (SPs). The SSO module is closely integrated with WebAuthn flows which is strongly authenticated prior to the creation of any SSO session.

The second requirement (F2) is that the users must have immediate access to a variety of applications using a single point of authentication without recurring authentication requests. This requirement is achieved by the system relying on SSO sessions controlled by IdP. Once the user has validated themselves with WebAuthn, any further access by the user on an SP will cause a lightweight redirect to the IdP, which will at once make a new SAML response by reason of the already valid SSO session. This does away with the re-authentication and has a seamless access across applications and this is a total fulfilment of F2.

The third requirement (F3) is secure creation and maintenance, and termination of multiple application and browser sessions. This is also fully satisfied. IdP controls the lifetime of SSO sessions such as issue of session identifiers, timeouts and authentication of SP-initiated logins. More to the point, the system manages to conduct SAML Single Logout (SLO) which allows the coordinated end of a session in all the SPs. A user who logs out of any single SP will have the logout request forwarded to all other SPs providing a single logout experience. This coordinated SLO mechanism is the confirmation that the system complies with F3.

The fourth requirement (F4) underlines the support of the registration and management of WebAuthn credentials, which are based on the hardware authenticator. This is an essential requirement. The system enables the users to add new authenticators with the purpose of creating WebAuthn credential creation challenges and attestation statements in addition to storing public keys at the IdP. The subsequent logins are carried out with correct WebAuthn authentication ceremony carried out using signed assertions of the authenticator. This provides a high-quality phishing-resistant authentication that is in line with the WebAuthn standard.

The fifth requirement (F5) states that the system should be able to work fully without the usage of passwords and rely only on the public/private key cryptography. The system that has been implemented completely fulfills this requirement. The user is never requested to create or save passwords at any point. The authentication is done through all challenge response signatures created by trusted hardware authenticators. The IdP has only stored the public keys and the SPs purely on

SAML assertions. Therefore, any security threat involving passwords like credential leakage, brute force or phishing attack are automatically neutralized.

The sixth requirement (F6) will require a strong backup authentication strategy to users who fail to gain access to their WebAuthn credentials. Although short-term redundancy (with a variety of WebAuthn credentials) is supported by the system, it does not have a full recovery process, including recovery codes, recovery through email, or trusted device verification. Thus, the system offers very little backup protection, and it is not in full compliance with the requirement.

Lastly, the seventh requirement (F7) is that it should be compatible with a standard identity federation protocol. This is fully achieved through this system by incorporating SAML 2.0 using Keycloak. Every significant SAML feature is covered such as SP initiated flow, IdP initiated flow, SAML assertion signing and Single Logout. Interoperability across many SPs and maintenance of the enterprise-level identity federation practices are guaranteed by the use of a standardized protocol.

### 5.2.2 Security Requirement Analysis

Our system is designed keeping in mind the main focus on security and privacy by integrating Single Sign-On (SSO) in WebAuthn based authentication and also by implementing a selective disclosure mechanism. These design decisions are direct responses to the identified security requirements (S1-S6) defined in 3.1.2.2 and counter the identified threats.

S1 requires a secure authentication mechanism to provide for only verified and legitimate users to access the system. This requirement is met with the integration of WebAuthn which provides passwordless authentication with hardware security keys such as YubiKey. Authentication is done through some biometric verification to ensure that only actual users who have valid credentials can access the system. This is an effective way to mitigate impersonation and unauthorized access threats (T1).

S2 addresses protection of important data from unauthorized modification. In the proposed system sensitive attributes of the user are never stored or transmitted in plaintext. Instead, user data is then hashing and securely processed by the Identity Provider. Since no raw data is exposed during authentication or verification, the risk of data manipulation is greatly reduced and therefore threat T2 is addressed.

S3 focuses on accountability in terms of logging and tracking mechanisms. The system keeps authentication and access records in the process of SSO. Since the users authenticate through a centralized Identity Provider, all the login and access events are traceable. This mechanism of logging provides for non-repudiation, which means that malicious users cannot deny their actions, and this directly reduces threat T3.

S4 points to encryption as one such defense against unintentional data exposure. The proposed system employs the technique of cryptographic hashing and the process of verifying attributes. Even if the data were to be intercepted, it would be encrypted or hashed, and would be of no use to the attackers. This greatly reduces the risk posed by accidental data leaks (T4).

S6 demands strong authentication during SSO operations to maintain unauthorized access to a number of services. This requirement is fully satisfied as SSO sessions are only created after successful WebAuthn-based authentication. Additionally, selective disclosure ensures that the service providers only receive verified and user-approved attributes, so that their identity is not misused and threat T6 is mitigated.

Overall, except S5 through the integration of SSO, WebAuthn, biometric authentication, hashing and selective disclosure, the proposed system has achieved the defined most of the security requirements, combined with strong privacy protection and secure access control.

### 5.2.3 Privacy Requirement Analysis

The mentioned privacy requirement (P1) in section 3.1.2.3 of selective disclosure is on the attribute-level that requires that plaintext user attributes are not disclosed to the IdP and SPs without explicitly stating it by the user. This is the key requirement of the privacy architecture of the system and it is entirely satisfied. All the attributes are hashed in the client with a salted hash function and then sent. The IdP only contains hashed values and only hashed attributes are contained in the SAML assertions to SPs. the Service Provider is not able to derive plaintext attributes out of these hashes, verification will only be done when the user provides a plaintext value during selective disclosure intentionally. This will ensure that privacy is maintained and the disclosure is under the control of the user in any SP.

### 5.2.4 Research Objective Analysis

Our research objectives are defined earlier in Chapter 1.4 those we have aimed to achieve through our work. By literature review and threat modelling, we have addressed RO1 and RO2 in our research. RO1 refers to explore the existing SSO technologies and establishing approaches to unite with webAuthn, and we have achieved this by literature review. RO2 refers to explore the vulnerabilities and possible threats of our system and we have identified the possible threats by threat modelling. By defining security requirements, functional requirements and privacy requirements in 3.1.2, RO3 have been achieved in our research. Additionally, we have fulfilled RO4 by designing the architecture and identifying the protocol flow of our system. Lastly, in chapter 5, by evaluating performance of our system, we have addressed RO5. And this way, by addressing all of our research objectives, our research has been completed.

## 5.3 Final Design Adjustments

Evaluation of the systems showed that there are particular points in which design modifications enhanced performance. A significant improvement required was the reorganization of the selective disclosure pipeline, so that only hashed attributes get out of the IdP. In the first design, attributes were delivered in mixed forms; the modifications were done such that the receiver of the SAML assertion, the Service Provider is given the hashed attributes only, and it is re-authenticated against the

plain-text provided by the user. This enhanced privacy protection in a greater way, decreasing the area of data exposure.

Also, the challenge management mechanism was enhanced. First, issues that were stored within the database were left until overwritten thereby posing a threat of replay attack. The new design adds hard expiration dates, so that the bad challenges are removed. This is also minimizing database size in the long run.

A complete Single Logout (SLO) cycle was added to the session management process involving IdP and SP. Previously the versions depended on independent session monitoring that introduced inconsistencies in which a user who had logged out of the IdP was active on some of the SPs. The refined SLO provides coordinated closure of all the sessions.

## 5.4 Statistical Analysis

The proposed WebAuthn-SSO-Selective Disclosure system statistical analysis is aimed at quantifying the behaviour of its performance, the latency and reliability of the system in various operational conditions. Because the system is a combination of several independent components, the WebAuthn operations, SAML-based SSO, and a hashed selective disclosure, the analysis is not based on hypotheses testing but on the empirical performance findings.

The system was tested by measuring three important operations to understand the behaviour of the system with regards to authentication: Registration latency of WebAuthn, SAML authentication and attribute process time. Basic descriptive statistics such as average time, minimum and maximum delay and observed variance were used to summarise the collected data in cases where repeating the same trials were done. The findings showed registration flows were the quickest and the variance was always low since most of the computation is performed in the authenticator. On the contrary, SAML operations showed increased variability because of overhead of XML processing and delays related to network round trips.

Also, measurements of throughput based were done to see the changes in system latency with increase in request load. Interestingly, a few endpoints, especially, the SAML callback handlers, indicated a reduction in latency with a moderate throughput, as a result of batching and internal caching. At higher loads, however, latency went up once more with queues at the IdP and SP. These findings show the non-linear nature of federated authentication system and indicate that load-balanced deployment is required in practical settings.

Failure rates were taken to test reliability since any operation in cryptography verification was registered. WebAuthn signature verification and hash-based selective disclosure checks had a zero false positive across several runs confirming the accuracy of the underlying cryptographic design. It was only seen to fail occasionally when test parameters were deliberately modified (e.g. wrong RPID, wrong salt, or modified SAML assertion).

On the whole, the statistical analysis confirms that the system is efficient when operating under normal conditions, predicts the latency behavior and provides the

highly reliable authentication results. The identified behaviour is consistent with the expectations of a hybrid WebAuthnSAML system, and it proves the practicality of the implementation of the system in the context of real multi-service systems.

## 5.5 Comparisons and Relationships

In this section, we present a comparative analysis between existing related research works and our approach. The result of the comparative analysis is shown in the Table 5.4. We have employed various properties for the comparison: Threat Model, Security Requirements, Functional Requirements, Security Requirements, Protocol validation, Implementation and Performance Analysis. Inside the table, the '● ' symbol is used to denote that a certain property is satisfied by the respective research, the '◐' denotes that a certain property is partially satisfied by the respective work and the '○' symbol denotes that the research does not satisfy the property.

Table 5.4: Comparison of related works

| | **Threat Model** | **Security Requirements** | **Functional Requirements** | **Security Analysis** | **Protocol Validation** | **Implementation** | **Performance Analysis** |
|---|---|---|---|---|---|---|---|
| Ou et al. (2024) [23] | ◐ | ◐ | ◐ | ◐ | ◐ | ○ | ○ |
| Hartl et al. (2025) [27] | ◐ | ◐ | ● | ◐ | ◐ | ○ | ○ |
| Kommareddy (2025) [28] | ● | ◐ | ◐ | ◐ | ◐ | ◐ | ○ |
| Ferdous et al. (2020) [7] | ○ | ◐ | ● | ◐ | ◐ | ◐ | ○ |
| Lassak et al. (2024) [32] | ○ | ◐ | ○ | ◐ | ○ | ○ | ○ |
| S.Fugkeaw (2023) [16] | ○ | ○ | ◐ | ○ | ◐ | ○ | ○ |
| J.Pietilä (2024) [24] | ○ | ○ | ◐ | ○ | ○ | ◐ | ○ |
| Klieme et al. (2020) [8] | ● | ○ | ◐ | ○ | ◐ | ◐ | ● |
| Our Approach | ● | ● | ● | ● | ● | ● | ● |

## 5.6 Discussions

### 5.6.1 Advantages

In proposed system, we have implemented an integrated system with the aim to protect the privacy of the user and to guarantee high security regarding the data. The system is a combination of Single Sign-On (SSO) and WebAuthn technologies to build a secure, efficient, and user-friendly authentication process.

Single Sign-On lets the users access multiple service providers with one session of authentication. This means that once a user has successfully logged in, he does not have to authenticate himself for each and every service. As a result, the system is able to reduce the repetition of login requests, enhance the convenience of users, and reduce the burden of managing multiple credentials.

To further increase privacy, the system has a selective disclosure mechanism. Through this mechanism, it is possible for users to clearly understand what attributes a service provider is asking for and what personal information they want to share. This ensures that only the data that is necessary to be disclosed is disclosed and explicit consentis ensured.

In addition, WebAuthn is built in conjunction with a YubiKey for passwordless authentication. Instead of using traditional passwords, users are authenticated with a biometric check with the YubiKey. This approach greatly improves security by mitigating risk that can come from using weak or reused passwords as well as reducing the chance of having credentials stolen.

Overall, by combining SSO, WebAuthn, and selective disclosure, the system not only offers robust security for user information, but also provides a seamless user authentication experience across multiple service providers.

### 5.6.2 Disadvantages

There are few disadvantages in our implemented system. Latency increases dramatically beyond 1000 concurrent users for all of the APIs tested - for example, the Registration API increases from about 78ms to 435ms at 1200 users and Verify Attributes increases to 2500ms, indicating queuing time and capacity overload. Though no complete failures occur, dramatic latency spikes (e.g. 14-20 ms stable to 67 ms in Login API) hurt user experience at 1200 users. In spite of this, throughput is improved linearly which indicates good handling of concurrency but not so efficient at large scale for the high rise of latency.

In our system, users must have an interaction with a YubiKey and attribute verification occurs with their information being provided twice, once to the identity provider when registering and a second time by the service provider at the time of selective disclosure. As a result, the user experience is a little more complicated than that of traditional identity provider systems. Moreover, the working of our system requires a YubiKey or an NFC supported device. These factors form the significant downsides of our system.

### 5.6.3 Limitations

Although our implemented system has a number of benefits, it has a number of limitations too. Firstly, the system can not be claimed completely user friendly since selective disclosure will need the users to type in every attribute individually (e.g., email, phone, name) when verifying. The attributes are all stored in hashed form hence users must provide the exact plaintext input to match the hash which reduces usability. Besides, the system does not have an effective recovery mechanism of accounts. Since sensitive information like email and phone numbers are also hashed, they cannot be handled with standard recovery procedures like sending recovery codes and one can hardly recover them in case of a lost WebAuthn gadget or a hardware authenticator (e.g., a YubiKey). This forms a single point of failure with users. Moreover, test analysis of multiple APIs shows that when the number of concurrent users exceeds around 1,000, latency increases significantly which indicates scalability constraints. Lastly, users may require guidance to use biometric keys, hardware tokens and selective disclosure mechanism properly. The recognition of these limitations can be used to make specific enhancements in future work.

# Chapter 6

# Conclusion

The research aims to determine how WebAuthn and Single Sign-On (SSO) can be integrated to allow secure, passwordless, and privacy-conscious authentication between domains. The proposed system fixes one of the most significant shortcomings of current SSO structures, namely, the absence of fine-grained and privacy-controlled sharing of attributes, by means of an introduction of a hash-based selective disclosure mechanism. This study through the design of a structured data model, protocol workflow, and a working prototype implementation has demonstrated that passwordless SSO can be established without much exposure of identity and minimal risk of exposing the identity to threats in case of password-based authentication.

The system assessment ascertains that selective attribute validation can be achieved through salted hashes and SAML can be augmented to conduct privacy preserving identity assertions. Despite good performance of system with average user loads, there are a few shortcomings such as absence of backup authentication feature, low usability because of features of manually entering attributes and scalability issues. In spite of these issues, the study makes its contribution to the integrative research on the topic of WebAuthn in the federated system of identity and the prospects of further development of privacy-conscious authentication methods.

All in all, this work shows a possible way forward to secure, interoperative and user-centered digital identity management. It preconditions the general shift towards passwordless SSO systems and creates a possibility to further research the field of usability improvement, recovery tools and actual application of this method in various systems.

## 6.1 Summary of Findings

The study manages to show that passwordless authentication could be easily deployed together with federated identity systems without compromising the privacy of the users using selective attribute disclosure. The solution implemented demonstrates that WebAuthn will have strong authentication that is resistant to phishing, whereas SAML will allow cross-application SSO without users having to verify their identity multiple times. The use of hash-based selective disclosure is introduced which guarantees that plaintext user attributes are not shared with any provider unless explicitly authenticated by the user, is a significant privacy shortcoming of traditional SSO systems. Continuous performance testing by WebAuthn operations

has ensured that the latency is low until 1000 users and SAML operations are more variable because of XML parsing and other network-related overheads. Notwithstanding this, the compound system will offer a steady throughput and will be fully operational with a load. In sum, the prototype confirms the existence of a single WebAuthn-SSO architecture which can provide secure, privacy-preserving and user-friendly authentication in various service domains.

## 6.2 Contributions to the Field

There are a number of meaningful contributions to the current identity and access-management research. To begin with, it introduces a hybrid authentication system that combines WebAuthn and SAML SSO without requiring the use of passwords and therefore removes long-standing security vulnerabilities like credential leaks and phishing attacks. Second, it presents a practical implementation of disclosure that principles based on hash functions and selective disclosure that enables Identity Providers and Service Providers to work without initially knowing the plaintext attributes of users at all, only to find out about them when the user explicitly reveals them. This fills in a fatal privacy vulnerability in existing federated identity systems where attributes are conventionally revealed wholesale. Third, data model and protocol flows presented in this study provide a formalized blueprint of integrating WebAuthn security property and SSO ease. Lastly, the experimental assessment of the system provides empirical data about the performance behaviour of federated passwordless authentication and has significant implications to developers and researchers dealing with infrastructures.

## 6.3 Future Work

We have plan to improve our system in certain aspects. To start with, the selective disclosure process should be made easier to use. Secure auto-completion, QR-based attribute transfer, or local encrypted storage may be available in future versions, so that users do not need to manually type each attribute when verifying them. This would prove to be a great way of reducing the level of friction in the interaction and would enhance the user experience.
Secondly, there is a lack of an efficient account recovery mechanism, which is a very serious problem. Email and phone numbers are hashed. Hence, the conventional recovery procedures cannot be used. Future research ought to involve privacy preserving recovery methods, e.g., trusted-device based recovery, multi-device WebAuthn registration, social recovery, or encrypted recovery agent that does not violate sensitive properties. The creation of such mechanisms without reducing the privacy guarantees will be one of the major contributions.
Thirdly, the level of scalability needs to be enhanced because the performance degrades when the number of simultaneous users reaches about 1,000. Adding database indexing schemes, caching layers, message queues, load-balanced authentication schemes, or parallelized verification modules may reduce the problem of latency. Assessment of the system based on distributed or cloud-based deployment models might also help in gaining an insight on higher throughput.
Lastly, the system might need improved user instructions and onboarding. In future

versions, there can also be interactive tutorials, with the right-hand side prompting the user and providing real-time feedback, to help users with WebAuthn hardware tokens, biometric authenticators and selective disclosure flows.
In general, these areas will be tackled and result in a more usable, resilient, and scalable authentication framework that can be used in wider real-world use.

# Bibliography


[1] A. Armando, R. Carbone, L. Compagna, J. Cuellar, and L. Tobarra, "Formal analysis of saml 2.0 web browser single sign-on: Breaking the saml-based single sign-on for google apps," in *Proceedings of the 6th ACM Workshop on Formal Methods in Security Engineering*, ser. FMSE '08, ACM, 2008, pp. 1–10. DOI: 10.1145/1456396.1456397.

[2] M. Al-Sarem and A. Qaoud, "Web single sign-on authentication using saml," *International Journal of Computer Science Issues*, vol. 2, no. 1, pp. 1–9, 2009. [Online]. Available: https://arxiv.org/pdf/0909.2368.pdf.

[3] M. S. Ferdous and R. Poet, "Dynamic identity federation using security assertion markup language (saml)," in *3rd Policies and Research in Identity Management (IDMAN)*, hal-01470512, London, United Kingdom, Apr. 2013, pp. 131–146. DOI: 10.1007/978-3-642-37282-7_13.

[4] A. Shostack, *Threat Modeling: Designing for Security*. Wiley, 2014, p. 45.

[5] M. S. Ferdous, "User-controlled identity management systems using mobile devices," Accessed: 2026-01-23, Ph.D. dissertation, University of Glasgow, 2015. [Online]. Available: https://theses.gla.ac.uk/6621/1/2015sadekphd.pdf.

[6] T. Bazaz and A. Khalique, "A review on single sign-on enabling technologies and protocols," *International Journal of Computer Applications*, vol. 151, no. 11, 2016. DOI: 10.5120/ijca2016911938.

[7] M. S. Ferdous, F. Chowdhury, M. O. Alassafi, A. A. Alshdadi, and V. Chang, "Social anchor: Privacy-friendly attribute aggregation from social networks," *IEEE Access*, vol. 8, pp. 61 844–61 871, 2020. DOI: 10.1109/ACCESS.2020.2981553.

[8] E. Klieme, J. Wilke, N. van Dornick, and C. Meinel, "Fidonuous: A fido2/webauthn extension to support continuous web authentication," in *2020 IEEE 19th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*, Guangzhou, China, 2020, pp. 1857–1867. DOI: 10.1109/TrustCom50675.2020.00254.

[9] Yubico, *The webauthn standard: Why it should matter to the public sector and how it works*, WEBAUTHN WHITE PAPER SERIES: MAY 2020, 2020. [Online]. Available: https://www.nass.org/sites/default/files/2020-05/Yubico%20White%20Paper%20How%20WebAuthn%20Works.pdf.

[10] F. Alliance, *How fido addresses a full range of use cases*, Online, 2022. [Online]. Available: https://fidoalliance.org/wp-content/uploads/2022/03/How-FIDO-Addresses-a-Full-Range-of-Use-Cases-March24.pdf.

[11] A. De Salve, A. Lisi, P. Mori, and L. Ricci, "Selective disclosure in self-sovereign identity based on hashed values," in *2022 IEEE Symposium on Computers and Communications (ISCC)*, IEEE, 2022. DOI: 10.1109/ISCC55528.2022.9913052.

[12] N. Fotiou, V. Kalos, Y. Thomas, G. Xylomenos, V. A. Siris, and G. C. Polyzos, "Selective content disclosure using zero-knowledge proofs," in *2022 Global Information Infrastructure and Networking Symposium (GIIS)*, 2022, pp. 52–56. DOI: 10.1109/GIIS56506.2022.9936933.

[13] V. Parmar, H. A. Sanghvi, R. H. Patel, and A. S. Pandya, "A comprehensive study on passwordless authentication," in *2022 International Conference on Sustainable Computing and Data Communication Systems (ICSCDS)*, Erode, India, 2022, pp. 1266–1275. DOI: 10.1109/ICSCDS53736.2022.9760934.

[14] C. Ardi and M. Calder, "The prevalence of single sign-on on the web: Towards the next generation of web content measurement," in *Proceedings of the 2023 ACM on Internet Measurement Conference*, ser. IMC '23, Open access, ACM, 2023, pp. 124–130. DOI: 10.1145/3618257.362484.

[15] N. Bindel, C. Cremers, and M. Zhao, "Fido2, ctap 2.1, and webauthn 2: Provable security and post-quantum instantiation," in *2023 IEEE Symposium on Security and Privacy (SP)*, San Francisco, CA, USA, 2023, pp. 1471–1490. DOI: 10.1109/SP46215.2023.10179454.

[16] S. Fugkeaw, "Achieving decentralized and dynamic sso-identity access management system for multi-application outsourced in cloud," *IEEE Access*, vol. 11, pp. 25 480–25 491, 2023. DOI: 10.1109/ACCESS.2023.3255885.

[17] L. Hanzlik, J. Loss, and B. Wagner, "Token meets wallet: Formalizing privacy and revocation for fido2," in *2023 IEEE Symposium on Security and Privacy (SP)*, 2023, pp. 1491–1508. DOI: 10.1109/sp46215.2023.10179373.

[18] M. Westers, T. Wich, L. Jannett, V. Mladenov, C. Mainka, and A. Mayer, "Sso-monitor: Fully-automatic large-scale landscape, security, and privacy analyses of single sign-on in the wild," *arXiv preprint arXiv:2302.01024*, 2023. DOI: 10.48550/arXiv.2302.01024.

[19] A. Ghosh, A. Mitra, S. C. Sethuraman, and A. K. Cherukuri, "Conceptual design and implementation of fido2 compatible smart card for decentralized financial transaction system," *arXiv preprint arXiv:2408.04977*, 2024. [Online]. Available: https://arxiv.org/abs/2408.04977.

[20] R. Kondakrindi, "Fido2: A new era in secure web authentication," *International Journal of Computer Engineering and Technology (IJCET)*, vol. 15, no. 4, pp. 841–858, 2024. DOI: 10.5281/zenodo.13479154.

[21] M. Lamba, R. K. Jaiswal, U. Kashyap, and S. K. Lakhan, "Is two factor authentication enough to secure your system: A study," *Journal of Management & Entrepreneurship*, vol. 13, no. 01, 2024.

[22] J. Müller and J. Oupický, "Post-quantum xml and saml single sign-on," *Proceedings on Privacy Enhancing Technologies*, vol. 2024, no. 4, pp. 525–543, 2024, hal-04845637. DOI: 10.56553/popets-2024-0128.

[23] H. Ou, C. Pan, Y. Tseng, and I. Lin, "Decentralized identity authentication mechanism: Integrating fido and blockchain for enhanced security," *Applied Sciences*, vol. 14, no. 9, p. 3551, 2024. DOI: 10.3390/app14093551. [Online]. Available: https://doi.org/10.3390/app14093551.

[24] J. Pietilä, *Implementing passwordless identity and access management system for a web application*, Bachelor's thesis, 2024.

[25] U. Siddiqua, M. Hijab, and F. Taranum, "Sso-identity access management system for cloud applications," *International Journal of Engineering Science and Advanced Technology (IJESAT)*, vol. 24, 2024. [Online]. Available: https://www.ijesat.com/ijesat/files/V24I1211_1734883630.pdf.

[26] K. Yan, X. Zhang, and W. Diao, "Stealing trust: Unraveling blind message attacks in web3 authentication," in *Proceedings of the 2024 ACM SIGSAC Conference on Computer and Communications Security (CCS '24)*, Salt Lake City, UT, USA: ACM, Oct. 2024. DOI: 10.1145/3658644.3670323.

[27] Z. Hartl and A. Derek, "Toward automated formal security analysis of saml v2.0 web browser sso standard—the post/artifact use case," *IEEE Access*, vol. 13, pp. 180 126–180 144, 2025. DOI: 10.1109/ACCESS.2025.3622379.

[28] R. R. Kommareddy, "Security and access control implementations using saml in modern web services," *International Journal for Research Trends and Innovation*, vol. 10, no. 8, a455–a463, 2025. [Online]. Available: https://www.ijrti.org.

[29] W.-W. Li, K.-H. Yeh, Y.-S. Ji, and S.-C. Cha, "Fido-enabled universal authenticator with web usability and privacy preservation," *Computers & Electrical Engineering*, 2025, Received 20 December 2024. DOI: 10.1016/j.compeleceng.2025.110201.

[30] V. Balatska, V. Poberezhnyk, P. Petriv, and I. Opirskyy, *Blockchain application concept in sso technology context*, Retrieved January 28, 2025.

[31] L. Jannett, M. Westers, T. Wich, C. Mainka, A. Mayer, and V. Mladenov, *SoK: SSO-MONITOR - The Current State and Future Research Directions in Single Sign-On Security Measurements*, https://sso-monitor.me/paper.pdf, Retrieved January 28, 2025.

[32] L. Lassak, E. Pan, B. Ur, and M. Golla, "Why aren't we using passkeys? obstacles companies face deploying fido2 passwordless authentication," *arXiv preprint arXiv:2408.04977*, Retrieved from https://arxiv.org/pdf/2408.04977.

[33] Massachusetts Institute of Technology, *Kerberos: The network authentication protocol – history*, https://kerborg-dc.mit.edu/about/history, Retrieved January 31, 2025.

[34] Microsoft, *How to configure sso with kerberos constrained delegation*, https://learn.microsoft.com/en-us/entra/identity/app-proxy/how-to-configure-sso-with-kcd, Retrieved January 31, 2025.