Received February 22, 2020, accepted March 11, 2020, date of publication March 17, 2020, date of current version April 14, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2981553

Social Anchor: Privacy-Friendly Attribute Aggregation From Social Networks

MD. SADEK FERDOUS[1,2], FARIDA CHOWDHURY[2], MADINI O. ALASSAFI[3], ABDULRAHMAN A. ALSHDADI[4], AND VICTOR CHANG[5]
[1]Imperial College Business School, Imperial College London, London SW7 2BU, U.K.
[2]Department of Computer Science and Engineering, Shahjalal University of Science and Technology, Sylhet 3114, Bangladesh
[3]Department of Information Technology, Faculty of Computing and Information Technology, King Abdulaziz University, Jeddah 21589, Saudi Arabia
[4]Department of Information Systems and Technology, College of Computer Science and Engineering, University of Jeddah, Jeddah 23890, Saudi Arabia
[5]School of Computing and Digital Technologies, Teesside University, Middlesbrough TS1 3BX, U.K.

Corresponding author: Md. Sadek Ferdous (s.ferdous@imperial.ac.uk)

ABSTRACT In the last decade or so, we have experienced a tremendous proliferation and popularity of different Social Networks (SNs), resulting more and more user attributes being stored in such SNs. These attributes represent a valuable asset and many innovative online services are offered in exchange of such attributes. This particular phenomenon has allured these social networks to act as Identity Providers (IdPs). However, the current setting unnecessarily imposes a restriction: a user can only release attributes from one single IdP in a single session, thereby, limiting the user to aggregate attributes from multiple IdPs within the same session. In addition, our analysis suggests that the manner by which attributes are released from these SNs is extremely privacy-invasive and a user has very limited control to exercise her privacy during this process. In this article, we present *Social Anchor*, a system for attribute aggregation from social networks in a privacy-friendly fashion. Our proposed Social Anchor system effectively addresses both of these serious issues. Apart from the proposal, we have implemented Social Anchor following a set of security and privacy requirements. We have also examined the associated trust issues using a formal trust analysis model. Besides, we have presented a formal analysis of its protocols using a state-of-the-art formal analysis tool called *AVISPA* to ensure the security of Social Anchor. Finally, we have provided a performance analysis of Social Anchor.

INDEX TERMS Attribute aggregation, privacy, social networks, SAML, OpenID, OAuth, AVISPA.

I. INTRODUCTION

A plethora of different types of Social Networks (SNs) have emerged in recent years providing a wide variety of online services in different domains with different use-cases. Many of these social networks have received wide-spread adoption and have been deeply integrated into our day to day lives. A recent study indicated that social networks users reached around 22.7 billions in 2017 [1]. With such deep integration in our lives, these social networks are increasingly being used in different application domains. The consequence of this is that an increasing amount of user attributes are stored in these social networks.

These attributes represent our identities in a wide variety of scenarios in different time periods. In such, they are very lucrative to other online service providers to understand our contexts and to offer more personalised services in exchange of such attributes. The demand for these attributes has allured social networking sites to extend their model so that they can act as Identity Providers (IdPs) and provide such attributes to third party online Service Providers (SPs). This has created a new business model for the social networks by which users can access such online services by leveraging the identities in their existing social networks. This is also beneficial to users as there is no need to register yet again to access a new online service and create a new set of attributes in a new SP.

However, the current setting of accessing online services by leveraging social networks has imposed an unnecessary restriction: a user can only release attributes from one single IdP in a single session. This restriction has this underlying unjustified assumption that a user needs to utilise only one IdP for service provisioning as if there was only one IdP to

The associate editor coordinating the review of this manuscript and approving it for publication was Longxiang Gao.

store all her attributes. In reality, users have different types of attributes scattered in multitude of different IdPs and social networking sites.

To highlight this restriction, let us consider a real-life scenario. Imagine a user, named *Alice*, wants to buy an age-restricted product from her favourite online shop. She might need to use her Governmental IdP (e.g. Passport or Driving authorities who issue her passport or driving licence) to prove her age, her bank details to pay for the product and her loyalty card information to collect loyalty points from the loyalty card provider during this purchase. All this information is stored at different IdPs and the current setting of online services does not allow Alice to provide all this information in a single session (e.g. during her purchase). In other words, Alice will need to retrieve all the required information from different providers and submit them manually at the online shop. Allowing users to aggregate attributes from different providers will enable them to release these attributes to a service provider in a single service session in a semi-automatic fashion. Thus, it would also enable service providers to offer more innovative services that have not been possible so far. Within the scope of this paper, a more relevant and specific example can be something as follows. Let us consider another user *Tom* who maintains three different social networking sites: LinkedIn (to maintain professional activities), Twitter (to follow well-known professionals and to get almost real-time updates on professional fields) and Facebook (to maintain social activities). Tom has recently completed a project quite successfully which has been updated on his LinkedIn profile. One of the most well-known technologists has shared the news of this project on Twitter praising the role Tom played on the successful completion of the project. Similarly, he has written a technical note on Facebook regarding his project which has received thousands of shares all over the worlds. Tom thinks that such a tweet and his popular technical note would add significant values to his resume and in his future career, for example when he applies for his next job. There is no way for Tom to aggregate such information from different social networks in the current setting of online services so that he could submit all his achievements to his future employer. One of the ways this could be achieved is attribute aggregation.

Another important aspect of releasing attributes using a social network to a third party service provider is the issue of privacy. This is because many of the attributes stored in such social networks are highly private in nature. Therefore, it is crucial to ensure that users have enough control to exercise her privacy while the attributes are being released to the service providers. Within this scope, we have investigated how privacy-friendly is the attribute release process using different popular social networking sites. Our findings suggest that the attribute release process is extremely privacy-invasive and users have very limited control during the process.

In this article, we aim to mitigate both of these serious issues. Towards this aim, we present *Social Anchor*, a system for attribute aggregation from social networks in a privacy-friendly fashion. Social Anchor is based on the work of Ferdous *et al.* [2] and allows a user aggregating attributes from different social networks and release them to a service provider in a secure and privacy-friendly mechanism with different levels of privacy control.

Contribution. This article has the following major contributions:

- At first, we highlight the shortcomings of the existing systems by presenting a detailed privacy analysis of how attributes are released from different social network sites.
- We present Social Anchor, the first ever system to allow users to collate attributes from several social network IdPs (SN IdPs, in short) in a privacy-friendly fashion, with a detailed discussion of how its proof of concept has been developed and implemented following a rigorous threat model and detailed requirement analysis to mitigate the identified threats.
- We analyse the underlying trust issues using a formal model of trust.
- We showcase its applicability by illustrating two different use-cases using the proof of concept.
- We present a formal analysis of Social Anchor protocols using a state-of-the-art protocol verification tool called AVISPA [3], [4] to ensure the security of its underlying protocols.
- Finally, We present a comparative performance analysis, in terms of latency, of Social Anchor to highlight its efficacy.

Structure. The structure of the paper is as follows. In Section II, we present a short background on Identity and Identity Management and briefly introduce a model of Identity and Attribute Aggregation. Section III presents our analysis on how attributes are released from different SN IdPs using different protocols and highlights the major limitations of the existing approach and the impact it has on the privacy of users. Social Anchor is based on the work of [2] in which a Hybrid model of Attribute Aggregation is showcased. We briefly describe the Hybrid Model in Section IV and underline the necessity of introducing Social Anchor. Section V presents two different architectures of Social Anchor to offer a user different levels of privacy controls during the attribute aggregation process along with their associated algorithms, underlying threat model, a thorough requirement analysis, implementation details and a formal analysis of its underlying trust issues. Section VI illustrates the applicability of Social Anchor by presenting two different use-cases. We present a formal security analysis of Social Anchor using AVISPA in Section VII with a discussion of its privacy and performance analysis, the advantages Social Anchor offers and the current limitations it has. We explore the existing related works and compare them with Social Anchor against a set of criteria in Section VIII. Finally, we conclude in Section IX with a hint of future work.

II. BACKGROUND

This section provides a brief background of Identity and the related concepts of Identity Management and Identity Management systems. This will help the readers to understand how different attributes in any social network make up the notion of identities for users in the corresponding social networks. A mathematical model helps to concretise the understanding and follow the associated protocols and formal analysis of Social Anchor in the subsequent sections. That is why a brief mathematical model of identities and attribute aggregation is also presented.

A. IDENTITY, IDENTITY MANAGEMENT AND FEDERATED IDENTITY MANAGEMENT

As per [5], an entity is a physical or logical object which can be uniquely identified within a certain context with its own *identity*. In this article, we only consider users as entities.

To provide a mathematical definition of the identity of a user, we utilise the Digital Identity Model (DIM) [6]. The model argues that the (whole) identity of a user actually consists of a set of partial identities, where each partial identity is valid within the domain (context) of an enterprise (organisation, e.g. an SN). Moreover, each partial identity encodes a number of attributes and their corresponding values, representing different characteristics of a user. Next, we elaborate this relation mathematically.

We assume that the set of domains is denoted with D whereas the domain of a single organisation is denoted with d. We use U_d, A_d and AV_d to denote the set of users, the set of attributes and the set of values for those attributes in d. The following partial function relates users and their attributes in a domain:

Definition 1: Let $atEntToVal_d : A_d \times U_d \rightarrow AV_d$ be the (partial) function that for an entity and attribute returns the corresponding value of the attribute in domain d.

The function is defined as a partial function to emulate the settings of practical systems. For example, such systems often have some compulsory attributes (e.g. email, telephone number, etc.) whose values must be provided by the users. At the same time, such systems can also have some optional attributes (e.g. postal addresses, age, etc.) for which users may not provide any values.

Next, the partial identity of a user u is defined using the following definition.

Definition 2: For a domain d, the partial identity *of a user $u \in U_d$ within d, denoted $parIdent^u_d$, is given by the set:*

$$\{(a, v) \mid a \in A_d,\ atEntToVal_d(a, u) \text{ is defined and equals } v\}.$$

Considering there are n valid attribute-value pairs for a user u, the partial identity of u in d can be defined as:

$$parIdent^u_d \triangleq \langle\, (a_1, v_1), (a_2, v_2), (a_3, v_3) \ldots (a_n, v_n)\, \rangle$$

Ultimately, we define the (total/whole) identity of a user u as the union of all her partial identities in all domains.

Definition 3: For an entity $u \in U$, the identity *of u is given by the set:*

$$ident^u \triangleq \bigcup \langle\, (d, parIdent^u_d) \mid d \in DOMAIN \text{ and } u \in U_d\, \rangle$$

Identity Management (IdM) is the concept to facilitate the management of digital identities [5]. Formally, Identity Management can be considered as the combinations of different technologies and policies that are used to represent and recognise any entity with its digital identities [7]. An Identity Management System (IMS) is a system that is used for facilitating identity management. Generally, each IMS interacts with the following parties:

- **Client/User.** A client/user is entity that receives services from a service provider (see below).
- **Service Provider.** A Service Provider (SP) is an entity, mostly an organisation, that is responsible for providing services to the clients or to other SPs. In other terminologies, it is also regarded as the Relying Party.
- **Identity Provider.** An Identity Provider (IdP) is an entity, also mostly an organisation, that is responsible for storing digital identities of clients and enabling clients to utilise their identities to receive services from an SP.

Among different models of identity management, Federated Identity Management (FIM) is one of the most widely used. It is based on the concept of Identity Federation where a federation is actually a legally binding business and technical contract among a group of two or more trusted organisations to allow their users to access restricted resources from different domains in a secure and seamless manner [8], [9]. Organisations in a federation thus form the so-called Circle of Trust (CoT) to signify their assumed trusted relationship. Within this relationship, FIM offers the Single Sign On (SSO) capability which enables any user to log in to the IdP of one organisation and then access services from other federated SPs without further logins. This alleviates the need for a user to authenticate each time she needs to access the federated services.

Sharing partial identities of users between different organisations (e.g. an IdP and SP) is a crucial functionality of any IMS. To preserve the privacy of the user, it is essential to ensure that the full partial identity (encoding the values of all attributes) of the user is never shared. Instead, a limited subset of the partial identity of the user is released during this process. We define this limited subset as the *profile* of a user in a single session. Mathematically, a profile is a subset of the partial identity of a user within a domain: $PROFILE^u_d \subseteq parIdent^u_d$. Hence, we can define the profile of a user $u \in U_d$ in domain d in the following way, where $j \leq n$:

$$PROFILE^u_d \triangleq \bigcup \langle\, (a_1, v_1), (a_2, v_2), (a_3, v_3) \ldots (a_j, v_j)\, \rangle$$

One of the attributes shared within a profile is an *identifier* which is used to uniquely identify a user in an IdP as well as in an SP. There are two types of identifier: persistent and transient/pseudonymous. A persistent identifier is permanent in nature and is used, accompanied by a special attribute

TABLE 1. **SAML/OpenID/OAuth notations.**

Name	Mathematical Representation
SAML Authentication Request	$AuthnReq \triangleq \langle\ id_{req}, id_{sp}\ \rangle$
SAML Assertion	$SAMLAssrtn \triangleq \langle\ (PROFILE^{u}_{idp}\ \rangle$
Encrypted SAML Assertion	$EncSAMLAssrtn \triangleq \langle\ \{SAMLAssrtn\}_{K_{sp}}\ \rangle$
SAML Response	$SAMLResp \triangleq \langle\ id_{req}, id_{sp}, id_{idp}, (\{SAMLAssrtn\}_{K^{-1}_{idp}} \| \{EncSAMLAssrtn\}_{K^{-1}_{idp}})\ \rangle$
OpenID Request	$OpenIDReq \triangleq \langle\ (id_{req}, id_{idp}, id_{rp}, url_{sp})\ \rangle$
OpenID Response	$OpenIDResp \triangleq \langle\ (id_{req}, id_{idp}, id_{rp}, \{PROFILE^{u}_{idp}\}_{K^{-1}_{idp}})\ \rangle$
OAuth Request	$OAuthReq \triangleq \langle\ (id_{req}, url_{sp})\ \rangle$
OAuth Response	$OAuthResp \triangleq \langle\ (id_{req}, id_{rs}, \{PROFILE^{u}_{rs}\}_{K^{-1}_{rs}})\ \rangle$

called *credential*, during the authentication process. A persistent identifier can be of different types such as a username, email and phone number while passwords are the most prominent credential nowadays. Conversely, a transient identifier is generated per session basis and is targeted for a specific SP. To preserve the privacy of a user, a profile generally contains a transient identifier to allow an SP maintaining a session for the user. This also undermines the threat where two malicious SPs collude to build a profile and ultimately track a user over their domains.

B. ATTRIBUTE AGGREGATION

As per [2], *Attribute Aggregation* is the process that allows a user to aggregate or combine her attributes from several identity providers and release them to an SP in a single session. Since a profile is utilised while attributes are released from an IdP to an SP, we can regard the attribute aggregation process as the aggregation of profiles of a user from different IdPs. Mathematically, we can define the set of aggregated profiles in the following way:

For a user u, the set of aggregated attributes of u, denoted by $Att{-}Agg^{u}$ is given by:

$$Att{-}Agg^{u} \triangleq \bigcup \langle\ PROFILE^{u}_{d} \mid d \in DOMAIN\ \rangle$$

There are several existing attribute aggregation models. A description and comparative analysis of these models can be found in [10].

III. ANALYSING EXISTING IDENTITY MANAGEMENT STANDARDS & SOCIAL NETWORKS

Currently, there are three popular Identity Management Standards, namely SAML (Security Assertion Markup Language) [11], OpenID [12], [13] and OAuth [14]. SAML is widely used in educational and Governmental settings within a trusted federation (explained below) while many popular social networks utilise OpenID and OAuth protocols to authenticate users and release their attributes to third party service providers. Even though there is an existing work which supports integrating such social networks with SAML federations [15] to offer federated services, it has never been investigated how attributes are released from different social network sites using these protocols. At first, we have examined these issues. In particular, we are interested to see if a user can leverage the chosen protocols to request a specific set of attributes (denoted as *Attribute Request*), to select and release specific attributes from the chosen social network sites (denoted as *Selective Disclosure*) and if the social networks sites have any support for attribute aggregation (denoted as *Attribute Aggregation*). Our findings are presented below with a brief discussion of each technology.

A. SAML

SAML is one of the most widely deployed federated identity management technologies [11]. It is based on XML (EXtensible Markup Language) and facilitates the exchange of authentication and authorisation information between different organisations in different domains. It utilises a request/response protocol that allows one party (an SP) to request identity information regarding a user which is then followed by response from the other party (an IdP) with the information by means of an assertion.

A SAML protocol flow between a user (denoted as u), an IdP (denoted as *idp*) and an SP (denoted as *sp*) is presented below using the notations presented in Table 1 [2].

While trying to access a service provided by *sp*, u is forwarded to a special service called the *Discovery Service* or *Where Are You From (WAYF)* Service. The WAYF shows a list pre-configured trusted IdPs (IDP_{wayf}) to u. After choosing her preferred IdP ($idp \in IDP_{wayf}$), u is forwarded to *idp* with a SAML authentication request consisting of an identifier of the request and an identifier (called *entity ID* in SAML) of *sp*. A SAML request is denoted using *AuthnReq* and is modelled as presented in Table 1, where id_{req} representing the identifier in each SAML request and id_{sp} denoting the entity ID of *sp*. At *idp*, u is authenticated at first and then *idp* prepares a SAML response with an embedded SAML assertion. The assertion contains the user profile as released by *idp*. Mathematically, the assertion is denoted with *SAMLAssrtn* and is modelled as per Table 1.

Then, *idp* digitally signs the (encrypted or unencrypted) SAML assertion and then embeds it inside a SAML response. The response also contains the request identifier (id_{req}), the entity ID (id_{idp}) of *idp* and the entity ID (id_{sp}) of *sp*. A SAML response is denoted with *SAMLResp* and modelled as per Table 1. Table 1 also introduces the following two notations:

- $\{SAMLAssrtn\}_{K_{idp}^{-1}}$ represents an assertion which is digitally signed with the private key of idp (K_{idp}^{-1})
- $\{EncSAMLAssrtn\}_{K_{idp}^{-1}}$ represents an encrypted assertion which is encrypted with the public key of sp (K_{sp}) and signed with the private key of idp (K_{idp}^{-1})

Finally, the response is sent back to sp by the idp. Upon receiving the response, the (encrypted/unencrypted) SAML assertion is extracted. If the response consists of an encrypted assertion, sp decrypts the assertion at first with the private key of sp (K_{sp}^{-1}) and then validates the signature with the public key of idp (K_{idp}). In case of an an unencrypted assertion, sp just validates its signature using the public key of idp. sp retrieves the embedded user attributes (the profile, $PROFILE_{idp}^{u}$) from the assertion, only if the signature is valid, otherwise the assertion is discarded.

This protocol flow requires establishing a notion of trust between an IdP and an SP within a federation. This notion of trust is established by exchanging the respective metadata of the IdP and SP and which are then stored at the appropriate repositories. In this way, each party builds up the Trust Anchor List (TAL). Upon completing the exchange of metadata, the IdP and SP are regarded belonging to the same federation (the CoT or Circle of Trust).

SAML has many implementations: Shibboleth [16], SimpleSAMLphp [17] and ZXID [18]. Among them, only Shibboleth and SimpleSAMLphp are widely used. In [2], it has been analysed how SAML and SimpleSAMLphp have been combined for attribute request, selective disclosure and attribute aggregation. For brevity, we are not repeating the analysis process here. The interested readers are requested to consult [2] for understanding how the analysis has been carried out. However, the result of our analysis is presented in Section III-D.

B. OpenID

OpenID is aimed to provide SSO (Single Sign On) solutions for web services using decentralised IMS [12], [13]. Like SAML, it consists of three parties : Users, OpenID Providers and Service Providers. It also utilises a request/response protocol which is founded upon the open trust paradigm meaning every party in OpenID trusts each other without formally establishing trust like SAML.

The protocol flow in OpenID starts when a user submits a request to access a service provided by an SP, also known as RP or Relying Party in OpenID terminology. Upon providing her OpenID identifier, the user is forwarded to the provider where she is authenticated. Then, the user returns back to the SP with an authentication response which is then validated. If the response contains the profile of the user as released by the provider, the user attributes embedded within the profile are extracted from the response. Based on these attributes, the SP takes an authorisation decision. Like SAML, an OpenID request and response are denoted with *OpenIDReq* and *OpenIDResp* respectively. Table 1 presents how they are modelled, where id_{req}, id_{idp} and id_{rp} denote the identifier for the request, the endpoint of the provider and the RP identifier respectively. Moreover, the URL of the SP at where the response needs to be returned is denoted with url_{sp}. Finally, $\{PROFILE_{idp}^{u}\}_{K_{idp}^{-1}}$ represents a user profile digitally signed with the private key of the provider.

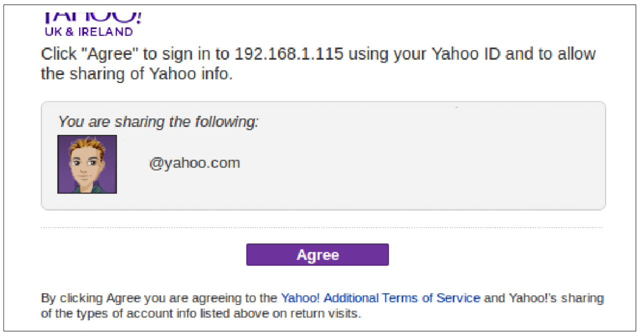
The original OpenID protocol was intended for authentication only, hence, the aspect of exchanging attributes was not initially considered. However, two extensions, the OpenID Simple Registration Extension (SREG) [19] and the OpenID Attribute Exchange (AX) [20], have been added to the original OpenID specification to allow an RP to request and fetch attributes from the provider. Such extensions allow an RP to request attributes from the provider and the provider might ask the user to release the requested attributes which are then sent back to the RP embedded inside the response.

Within this scope, our intention is to investigate how a few popular OpenID providers act when they receive a request to return attributes. We are particularly interested to see how each provider allows a user to selectively choose her attributes individually (the selective disclosure of attributes). With these in mind, we have selected four popular OpenID providers: *Yahoo (http://openid.yahoo.com/)*, *Google (www.google.com)*, *LiveJournal (http://www.livejournal.com/)* and *VeriSign (https://pip.verisignlabs.com/)*. We have deployed a SAML SP and IdP using the SimpleSAMLphp implementation. Even though the IdP and SP are based on SAML, SimpleSAMLphp allows different types of (e.g. OpenID or OAuth enabled) IdPs to be configured as authentication sources at the IdP which allows a user to get authenticated at another IdP and return attributes to the SimpleSAMLphp IdP. At this moment we are not interested what the SimpleSAMLphp IdP does with the attributes. We have used this approach because of two reasons:

- since the support of such IdPs are built into SimpleSAMLphp, we do not need to deploy a separate OpenID or OAuth SP and
- we plan to use a very similar setup for building our proof of concept of Social Anchor.

We have configured the SimpleSAMLphp OpenID module to request *openid, email, fullname* and *age* attributes among which the *age* attribute is optional and the rest are compulsory. A typical flow using this approach is as follows: a user visits the SP and then chooses the SAML IdP. When the user is forwarded to the SAML IdP, she chooses OpenID as the authentication source and then she provides her OpenID and the usual OpenID protocol flow takes place.

We have created dummy user accounts in these providers that we have used during our experiments. It has been noted how a provider allows a user to authorise it to release her attributes, once the user is authenticated or during the user authentication process and screenshots of the process have been captured. The screenshots for Yahoo, Google, LiveJournal and VeriSign are illustrated in Figure 1, Figure 2, Figure 3 and Figure 4 respectively. All personal information has been removed from the screenshots of the paper to preserve privacy and anonymity.



YAHOO!
UK & IRELAND

Click "Agree" to sign in to 192.168.1.115 using your Yahoo ID and to allow the sharing of Yahoo info.

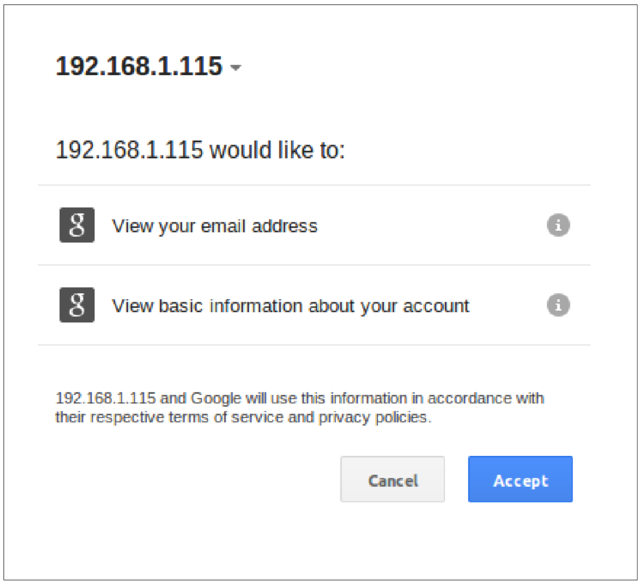
You are sharing the following:

@yahoo.com

Agree

By clicking Agree you are agreeing to the Yahoo! Additional Terms of Service and Yahoo!'s sharing of the types of account info listed above on return visits.

FIGURE 1. Yahoo login screen.



192.168.1.115

192.168.1.115 would like to:

View your email address

View basic information about your account

192.168.1.115 and Google will use this information in accordance with their respective terms of service and privacy policies.

Cancel Accept

FIGURE 2. Google login screen.

We have noted two problems: the login screen does not inform the user which individual attributes have been requested (e.g. in Figure 1, the user has no way of knowing which information will be released) and none of the providers except VeriSign has any support of selective disclosure. This practice not only violates the principle of data minimisation but also forces a user to release attributes based on obscure information. This may have serious implications on the privacy of the user.

We have also noted that none of the OpenID implementations has any attribute aggregation facility.

C. OAuth

OAuth, like OpenID, is based on the concept of open trust. It is one of the fastest growing systems widely used over the Internet for delegating the access right of a user to another user in a secure and more user-friendly fashion [21], [22]. It interacts with four different types of entities.

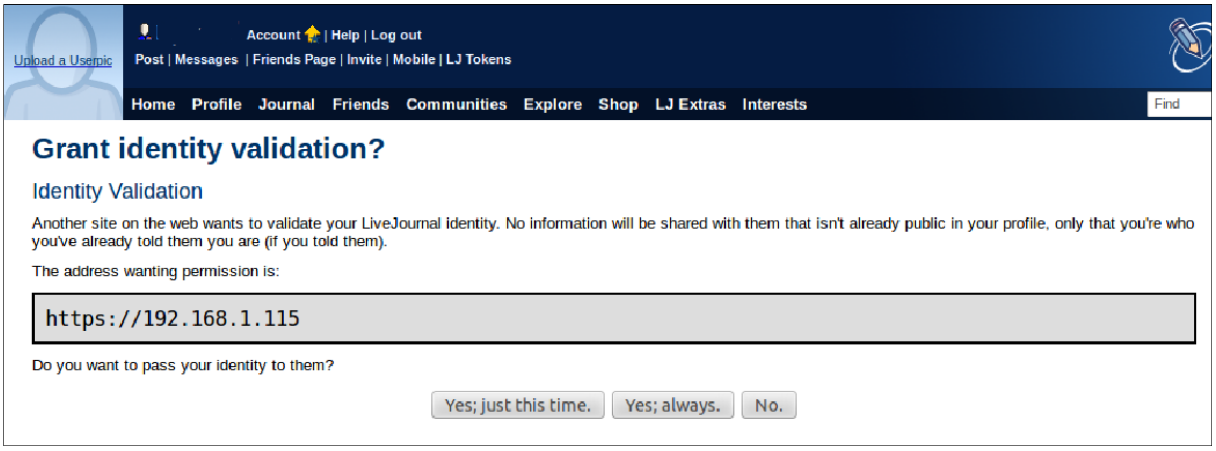
- **Resource Owners.** Resource owners emulate the role of users in SAML and OpenID. They own and control their protected resources for which they would like to grant access rights to third parties.
- **Clients.** Clients represent third party applications which submit requests to access protected resources on a user's behalf.
- **Authorisation Servers.** Authorisation servers grant access requests so that clients can access the requested resources.
- **Resource Servers.** Resource servers host protected resources. In many settings, both resource and authorisation servers are combined to act as the same entity.

Next, we present a simplified abstract OAuth protocol flow with the assumption that a resource owner wants to delegate access rights to a client so that it can access some protected resources of the resource owner, hosted at a resource server. At first, the client requests an authorisation for the corresponding resources from the owner. The owner authorises the client by returning a credential called the *authorisation grant*. The client is then authenticated at the authorisation server and provides the authorisation grant. Upon validating the authorisation grant, the authorisation server issues an *access token*. The client then presents the access token to the resource server in order to access the protected resources. Upon validating the access token, the resource server returns the requested resource to the client.

An OAuth request and response are denoted with *OAuthReq* and *OAuthResp* respectively and are modelled as presented in Table 1, where id_{req}, id_{rs} and url_{sp} denote the identifier for the request, the identifier of the resource server and the URL of the SP to receive any response respectively. Like before, $\{PROFILE^{u}_{rs}\}_{K^{-1}_{rs}}$ represents a user profile digitally signed with the private key of the resource server.

Unlike OpenID, OAuth does not require any separate mechanism to exchange attributes since all user attributes are considered as resources which can be accessed using OAuth upon approval from the user. Similar to OpenID, we are particularly interested to see how a user can grant authorisation to a client. We have chosen three popular IdPs based on OAuth: *Facebook (https://www.facebook.com)*, *LinkedIn (http://www.linkedin.com/)* and *Twitter (https://twitter.com/)*. We have used the same SimpleSAMLphp setup to initiate the OAuth protocol flow. A typical flow is as follows: the user visits the SP and then chooses the SAML IdP. When the user is forwarded to the SAML IdP, she chooses one of the OAuth providers (Facebook, Twitter and LinkedIn) as the authentication source and then the usual OAuth protocol flow takes place.

To initiate this protocol flow, a respective application needs to be registered at each provider. For Facebook, it is at [23], for LinkedIn it is at [24] and for Twitter it is at [25]. Once the application is registered, an application key and a corresponding secret are generated. These two pieces of information are then used at SimpleSAMLphp to initiate the protocol flow at the respective IdP. In addition, how user attributes can be requested using a registered application depends entirely on the respective provider. For Facebook, the application can be


Upload a Userpic
Account | Help | Log out
Post | Messages | Friends Page | Invite | Mobile | LJ Tokens
Home Profile Journal Friends Communities Explore Shop LJ Extras Interests
Find

Grant identity validation?

Identity Validation

Another site on the web wants to validate your LiveJournal identity. No information will be shared with them that isn't already public in your profile, only that you're who you've already told them you are (if you told them).

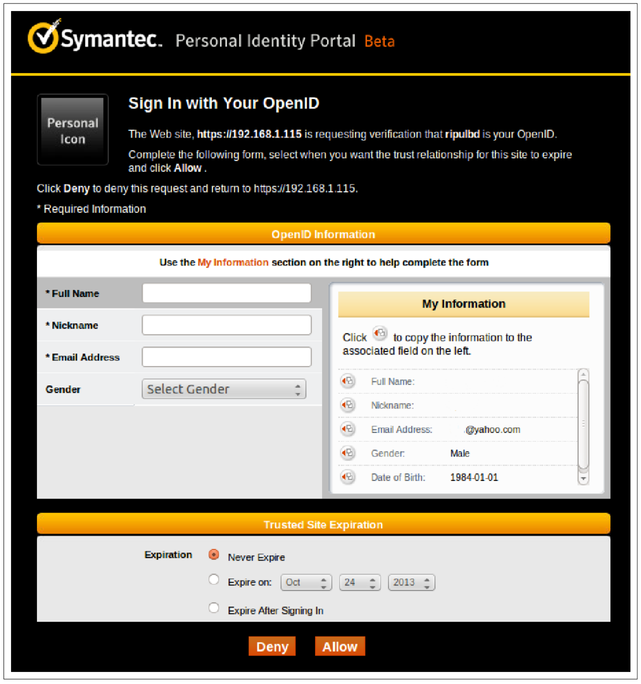
The address wanting permission is:

https://192.168.1.115

Do you want to pass your identity to them?

Yes; just this time.
Yes; always.
No.

FIGURE 3. **LiveJournal login screen.**



Sign In with Your OpenID

The Web site, **https://192.168.1.115** is requesting verification that **ripulbd** is your OpenID.

Complete the following form, select when you want the trust relationship for this site to expire and click **Allow** .

Click **Deny** to deny this request and return to https://192.168.1.115.

* Required Information

OpenID Information

Use the My Information section on the right to help complete the form

* Full Name	
* Nickname	
* Email Address	
Gender	Select Gender

My Information

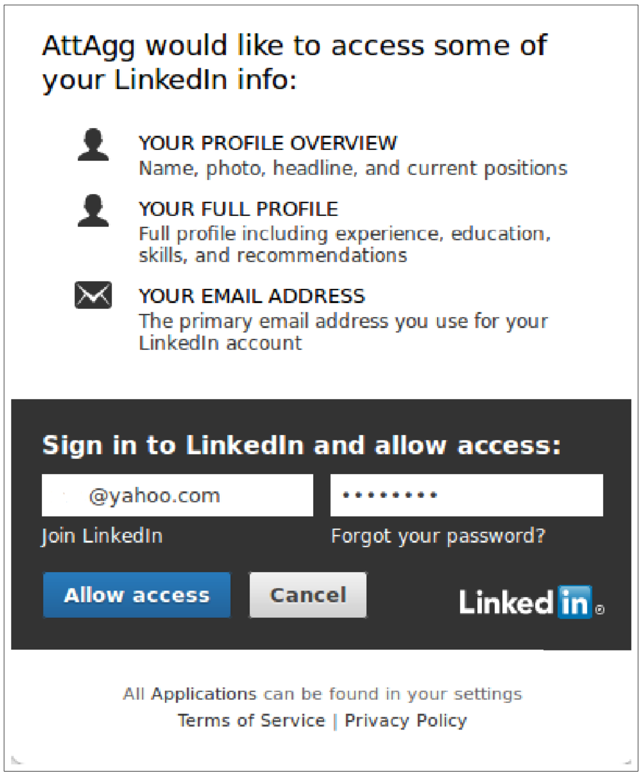
Click to copy the information to the associated field on the left.

Full Name:	
Nickname:	
Email Address:	@yahoo.com
Gender:	Male
Date of Birth:	1984-01-01

Trusted Site Expiration

Expiration [✓ Never Expire] Expire on: Oct 24 2013 Expire After Signing In

FIGURE 4. **VeriSign login screen.**



AttAgg would like to access some of your LinkedIn info:

YOUR PROFILE OVERVIEW
Name, photo, headline, and current positions

YOUR FULL PROFILE
Full profile including experience, education, skills, and recommendations

YOUR EMAIL ADDRESS
The primary email address you use for your LinkedIn account

Sign in to LinkedIn and allow access:

@yahoo.com	••••••••

All Applications can be found in your settings
Terms of Service | Privacy Policy

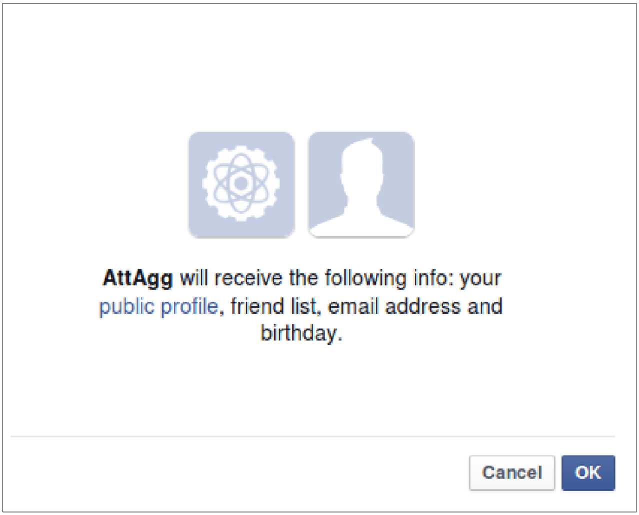
FIGURE 5. **LinkedIn authorisation screen.**

configured to request different user attributes or information regarding requested attributes can be passed as configuration parameters while initiating the protocol. We have adopted the second approach. For LinkedIn, the application needs to be configured to request attributes and it has been configured to request the full public profile and the email address attributes of a user. For Twitter, the public user attributes have been requested from SimpleSAMLphp. Like before, we have noted and captured screenshots of the process by which a user allows the application to access the requested attributes. Screenshots for LinkedIn, Facebook and Twitter are given in Figure 5, Figure 6 and Figure 7 respectively. We have found that, like OpenID providers, the same two problems exist: the login screen does not inform the user which individual attributes have been requested (for example, in Figure 6 it is not clear which information from the public profile will be released) and none of the providers has any support for selective disclosure. A user needs to release either all attributes or none. This approach also violates the principle of data minimisation and forces a user to release attributes based on obscure information and thus can have serious implications on the privacy of the user.

Similar to OpenID, we have noted that none of the OAuth provider has any such facility of attribute aggregation.

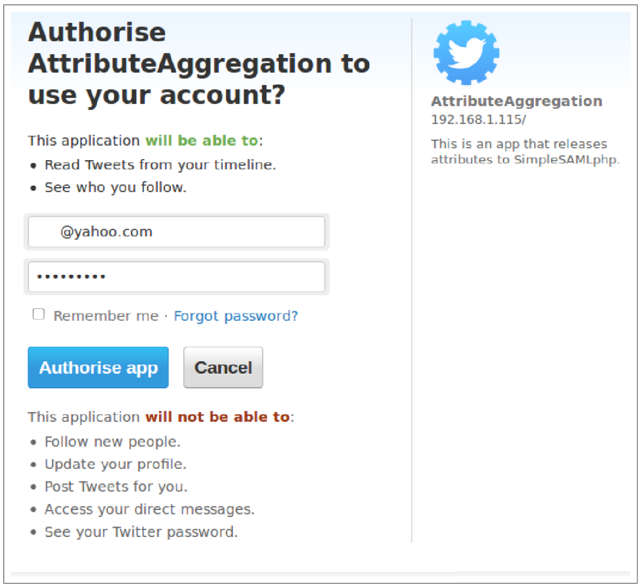
TABLE 2. Comparison of protocols and implementations.

Protocol	Implementation	Attribute Request	Attribute Release	Attribute Aggregation
SAML	Shibboleth	✓	✗	✗
	SSPHP	✗	✓	✗
OpenID	Yahoo	✓	✗	✗
	Google	✓	✗	✗
	LiveJournal	✓	✗	✗
	VeriSign	✓	✓	✗
OAuth	Facebook	✓	✗	✗
	LinkedIn	✓	✗	✗
	Twitter	✓	✗	✗



AttAgg will receive the following info: your public profile, friend list, email address and birthday.

FIGURE 6. Facebook authorisation screen.



Authorise AttributeAggregation to use your account?

This application **will be able to**:
- Read Tweets from your timeline.
- See who you follow.

@yahoo.com

•••••••••

Remember me · Forgot password?

This application **will not be able to**:
- Follow new people.
- Update your profile.
- Post Tweets for you.
- Access your direct messages.
- See your Twitter password.

AttributeAggregation
192.168.1.115/
This is an app that releases attributes to SimpleSAMLphp.

FIGURE 7. Twitter authorisation screen.

D. SUMMARY

The result of our investigation is presented in Table 2. In the table, the symbol ✓ indicates that a particular function is supported by the respective implementation and the symbol ✗ signifies the absence of support in a corresponding implementation.

From the table, Shibboleth allows specific attributes to be requested by an SP, however, does not support the selective disclosure function. On the other hand, SimpleSAMLphp does support the selective disclosure property using its *Consent* module, however, does not allow specific attributes to be requested. SAML has no mechanism to aggregate attributes, however, it has been used for the implementations of existing models of attribute aggregation.

For OpenID and OAuth, even though there is a mechanism to request specific attributes, a user can hardly perform selective disclosure of attributes in all the investigated providers except VeriSign. Also, no provider supporting OAuth and OpenID has any provision for attribute aggregation from multiple social networking accounts.

IV. HYBRID MODEL OF ATTRIBUTE AGGREGATION

In [2], a novel Hybrid model of attribute aggregation using SAML within a federated domain has been proposed. The proposed model is based on the Identity Proxying and Identity Relay models which are briefly described below.

A. IDENTITY PROXYING (IP) MODEL

A user in this model can aggregate attributes from different IdPs by leveraging a trusted IdP called the *Proxy IdP* [26], [27]. During the aggregation process, the SP forwards the user to the Proxy IdP; from where the user is forwarded to other IdPs one after another. After each successful authentication at the IdPs, the user returns back to the Proxy IdP with an assertion consisting of the profile. The assertion is then validated by the trusted IdP and the embedded attributes are retrieved. In this manner, attributes from different IdPs are aggregated into a combined set which is then released to the SP. Based on the combined attributes, the SP takes an authorisation decision.

B. IDENTITY RELAY (IR) MODEL

The IP model requires a strong trust assumption on the Proxy IdP. To relax this assumption, Identity Relay model, a generalised case of the IP model, has been proposed [26], [27]. Like the IP model, the IR model also relies on a central IdP, called the *Relay IdP*. However, the Relay IdP functions in a different way. In this model, upon a successful authentication at another IdP, the user returns back to the Relay IdP with an encrypted assertion which has been encrypted by the other IdP using the public key of the SP. Thus, the Relay IdP has no knowledge of the attributes released by other IdPs. The Relay IdP just combines these encrypted assertions into a single assertion and forwards it to the SP. The SP, then, extracts, decrypts and validates each single encrypted assertion and finally retrieves attributes from each of them. Based on the combined attributes, the SP takes an authorisation decision.

The proposed Hybrid model allows the aggregation and release of attributes from different federated SAML IdPs to an SP by leveraging a trusted third party called *Hybrid IdP*. The architecture of the Hybrid model is illustrated in Figure 8. The dotted area in the figure represents the Circle of Trust (i.e. the federation) between two entities.

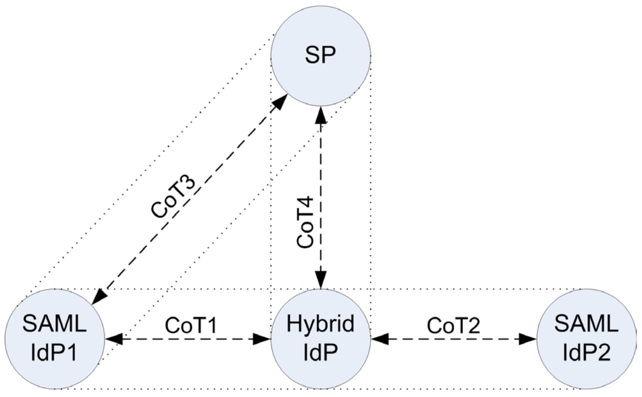


FIGURE 8. Architecture of hybrid model.

The Hybrid model essentially combines the capabilities of both the Proxy and Relay IdP in a single setting to offer different privacy levels while aggregating attributes from different SAML IdPs. Assuming the role of the Proxy IdP, the Hybrid IdP allows a user to aggregated attributes from other SAML IdPs. To facilitate this, other third-party SAML IdPs (*SAML IdP1* and *SAML IdP2* in Figure 8) must be federated with the Hybrid IdP. In addition, the Hybrid IdP needs to be federated with the SP (as illustrated in Figure 8).

On the other hand, when the Hybrid IdP assumes the role of a Relay IdP, it will only receive encrypted assertions from the other third-party SAML IdPs. In this way, Hybrid IdP will have no knowledge on which attributes are released to the SP by the other IdP and hence, provides a layer of privacy for the user. To facilitate this, other SAML IdPs must be federated with the SP so that they can create and release encrypted assertions to the SP without exposing them to the Hybrid IdP. This condition is visualised in Figure 8, encrypted assertions can be released by SAML IdP1 to the SP as they belong to the CoT whereas the SAML IdP2 cannot release such assertions to the SP.

In [2], the authors have presented how such an architecture has been designed and how a proof of concept has been developed following a set of requirements and design choices. In addition, they have illustrated two use-cases to explore the suitability of the proposed model and analysed different security and privacy issues. One crucial feature lacking in the Hybrid model is that it cannot be used to aggregate attributes from any other IdPs except SAML IdPs. With the emergence of social networks, a lot of user attributes are stored in the corresponding IdPs of different social networks. A mechanism to aggregate attributes from these IdPs, hence, would be beneficial and timely. This mechanism would provide an additional layer of advantage if it could address the serious privacy issues of attribute disclosure identified in Section III. In this article, we aim to achieve these goals. In particular, our motivation is two-fold:

- we seek to explore how we can extend the concept of the Hybrid model for aggregating attributes from different social networks and
- how it can be carried in a privacy-preserving fashion so that we can rectify the problems and gaps identified in Section III.

V. SOCIAL ANCHOR

In this section, we present Social Anchor, our proposed privacy-friendly attribute aggregation system for social networks. Social Anchor is based on the ideas of the Hybrid model of attribute aggregation with the sole aim to address the privacy issues identified in the previous section. In addition, to accommodate different privacy requirements, we present two different types of architectures. In the following, we present a threat modelling process to identify different security and privacy threats for Social Anchor (Section V-A), formulate different functional, security and privacy requirements (Section V-B), describe the two architectures along with their algorithms (Section V-C & Section V-D), discuss their implementation strategies (Section V-E) and finally, analyse the issues of trust involving Social Anchor (Section V-F).

A. THREAT MODELLING

Threat modelling is an integrated process of designing and developing a secure system. A well-defined threat model helps to identify threats on different assets of a secure system. In order to tackle such threats, different mitigation strategies need to be outlined by formulating different security and privacy requirements [28]. In essence, a threat modelling consists of the following steps [29], [30]:

(a) listing assets of the respective system,
(b) identifying possible threats for each of those assets and
(c) determining strategies to mitigate those threats.

An asset is the abstract or physical resource in a system that needs to be protected from an adversary (attacker) [28]. It is the resource for which a threat exists and represents the target of the adversary in the system. we have identified a few assets for an Identity Management System (IMS) in [2]. Since Social Anchor is also a part of an IMS, these assets are also applicable to Social Anchor. The assets are discussed below.

- **Partial Identity of a user**. In an Identity Management system, we consider the partial identity ($parIdent_d^u$) of a user u in a domain/system d as the core asset. This is because the partial identity consisting of different attributes represent a valuable commodity to any adversary. It is to be noted that such a partial identity consists of different attribute name-value pairs for that user in that application domain as presented in Section II.
- **Activities associated with a partial identity**. In addition to the partial identity, activities associated with a partial identity represents a valuable asset. The reason is that an adversary can abuse the knowledge of such activities to profile users across different domains. Within the scope of this article, if an attribute aggregation feature is added to a third party, e.g. a social network, it can build a profile of a user when the attribute aggregation is carried out at its end which then can be abused in many ways.

Identifying Threats. A threat represents the capability of an adversary to abuse an asset of a system so that she can invade the security of the system or invade the privacy of a user [28]. We have identified several threats within the scope of Social Anchor, which can be classified in two categories: security threats and privacy threats and are presented below.

1) SECURITY THREATS

Security threats are closely related to the capability of an attacker who intends to compromise the system's architecture. In order to better understand these threats, it is therefore beneficial to model the capabilities of an adversary. The most well-known and strongest adversary model used in analysing communication protocols is the Dolev-Yao (D-Y) model [31]. In this model, it is assumed that a D-Y adversary (attacker) can fully access all communication channels and has the capability to intercept all messages sent and received over the specified channels. The attacker can also modify such messages, only if the attacker has the correct cryptographic key. That is, the attacker is incapable to break cryptographic mechanisms.

In relation to the D-Y model, the identified threats are as follows.

- **Spoofing**. An unauthenticated and unauthorised user getting hold of an information regarding the partial identity of a valid user achieves the ability to impersonate as the valid user.
- **Tampering**. Data while being transmitted is intercepted and altered by an attacker with malicious intents.

2) PRIVACY THREATS

In the absence of any model that explores the capability of an attacker in relation to privacy, we assume that privacy threats mostly emerge from the lack of any privacy control for any user. That is, to the user, the Service Provider can be a potential attacker who tries to get as many attributes as possible from the users with the malicious intents to abuse them afterwards. Therefore, the goal of the user is to limit this abuse by releasing only a minimum number of attributes that are essential to access the requested service. Based on this assumption, the identified threats are as follows.

- **Selective disclosure**. Users cannot choose appropriate attributes that they want to release to a third party, e.g. a service provider.
- **Explicit consent**. Users have no way to provide any consent before their attributes are disclosed to service providers.
- **Lack of control**. Users have little control on the way attributes are released to the service providers.
- **Aggregated profiling**. Attributes aggregated by an untrusted entity would enable it to profile any user in a massive scale.

B. REQUIREMENT ANALYSIS

To minimise the identified threats, mitigation strategies must be carefully planned and transformed into actions. One way to achieve this is to transform them into requirements which the system must fulfil as proposed in [28]. In [2], we have already identified a set of different functional, security and privacy requirements for the Hybrid model. Since the current research is an extension of the Hybrid model, most of these requirements apply here as well. That is why we omit most of these requirements here for brevity and present only those requirements needed to minimise the threats for aggregating attributes within the setting of social networks.

1) FUNCTIONAL REQUIREMENTS (FR)

Functional requirements ensure the core functionalities of a system. The requirements are listed below.

- **F1**. The Hybrid IdP can interact with different SN IdPs using their corresponding protocols.
- **F2**. The Hybrid IdP can validate any response received from different SN IdPs and extract embedded attributes.

2) PRIVACY REQUIREMENTS (PR)

The privacy requirements presented below aim to tackle the specific privacy threats identified above. The requirements are listed below.

- **P1**. Users can select each attribute provided by the SN IdPs before they are released to the SP. This requirement is to undermine the *selective disclosure* threat.
- **P2**. Users can provide explicit consents before any attribute is released to an SP. This requirement is to undermine the *explicit consent* threat.
- **P3**. Users have the control to decide when attributes can be exposed to the Hybrid IdP and when they need to be masked from the Hybrid IdP. This is to deter the *lack of control* threat.
- **P4**. Attributes are aggregated only by a trusted entity. This requirement is leveraged to undermine the *aggregated profiling* threat.

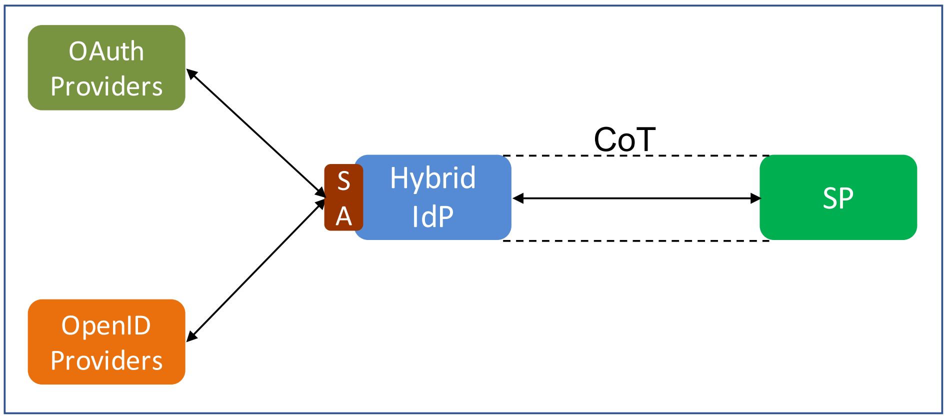


FIGURE 9. Social anchor: Type 1 architecture.

3) SECURITY REQUIREMENTS (SR)

Similarly, the security requirements presented below aim to tackle the security threats identified above. The requirements are listed below.

- **S1**. There must be a secure user registration and authentication mechanism in each SN IdP. Before accessing her attributes and releasing those attributes to the Hybrid IdP, a user must be authenticated.
- **S2**. The transmitted data between any two parties is not disclosed to any unauthorised entity. S1 and S2 are combinedly used to undermine the *spoofing* threat.
- **S3**. The transmitted data is not altered during transmission. This requirement is used to undermine the *tampering* threat.

C. ARCHITECTURE: TYPE 1

Social Anchor is a system for aggregating attributes from different SN IdPs. It is based on the Hybrid model allowing any user to release aggregated attributes to a SAML SP using the Hybrid IdP. Its ultimate focus is to facilitate this provision in a privacy-friendly way. Towards this aim, we present two different architectures of Social Anchor to offer different levels of privacy controls for any user. In this section we present the first architecture (denoted as *Type 1 architecture*) while the second architecture (denoted as *Type 2 architecture*) is presented in the following section (Section V-D).

The Type 1 architecture of Social Anchor is illustrated in Figure 9. This is analogous to the Hybrid model where a Hybrid IdP takes the central stage and the SP is under the same CoT as the Hybrid IdP, implying they belong to the same federation. Additionally, the Hybrid IdP has been extended with the Social Anchor capabilities to allow attribute aggregation from multiple social networking websites. In Figure 9, this is represented with *SA* attached to the Hybrid IdP. The added capability enables a user to aggregate attributes in a privacy-friendly manner from different social networks utilising the Hybrid IdP of Social Anchor in the Identity Proxying (IP) mode. As identified earlier, the privacy-friendly mechanism specifically aims to address the following gaps:

- the selective disclosure of attributes released from social networks to a SAML SP.
- explicit consents for attributes released from social networks.

To facilitate this, the Hybrid IdP in Social Anchor has been modified with the following capabilities using an attribute aggregation algorithm:

- Interactions with different SN IdPs in such a way that they can release attributes to the Hybrid IdP in a single session.
- An interface to aggregate attributes from different SN IdPs in a single session.

The attribute aggregation algorithm is presented in Algorithm 1 which can be divided in two phases: the setup phase and the aggregation phase. In the setup phase, different SN IdPs are added to a list in the Hybrid IdP (represented as *soruceList* in Algorithm 1) in such way that they can act as external IdPs and the authentication task can be delegated to them. In addition, the set of aggregated attributes ($Att - Agg^u$) is initialised as null. The core of the algorithm is the aggregation phase in which a user can aggregate attributes from the external IdPs. The steps in this process are captured in the *aggregate* method in the algorithm. The method is called with *soruceList* and $Att - Agg^u$ when the Hybrid IdP receives a SAML authentication request (*AuthnReq*) from the SP. Then, the Hybrid IdP displays the list of external IdPs (line 19 in Algorithm 1) from which a user can repeatedly select an IdP (not already been used) one after another. During each iteration, the Hybrid IdP interacts with the selected IdP using the respective protocol. Once a response from the respective IdP is received, the Hybrid IdP validates the response and extracts the set of attributes released from the IdP. This set is then aggregated to the already aggregated set(s) of attributes (line 34 in Algorithm 1). This process is repeated (line 35 in Algorithm 1) until the user decides to end the aggregation process or attributes from all external IdPs have been aggregated. The detailed protocol flow utilising this architecture is illustrated in Section VI.

Algorithm 1 Attribute Aggregation Algorithm at Hybrid IdP

```
1  Input: The request for attribute aggregation from an SP
2  Output: A SAML response to the requested SP
3  Start
4      Set $Att - Agg^u := null$
5      Set $OAuthSource := \{Facebook \cup LinkedIn \cup Twitter\}$
6      Set $OpenIDSource := \{Yahoo \cup Google \cup LiveJournal \cup VeriSign\}$
7      Set $soruceList := \{OAuthSource \cup OpenIDSource\}$
8      On receive ($AuthnReq$)
9          $id_{req} := AuthnReq.id_{req}$
10         $id_{sp} := AuthnReq.id_{sp}$
11         $Att - Agg^u = \text{aggregate}(soruceList, Att - Agg^u)$
12         $SAMLAssrtn := \{Att - Agg^u\}_{K^{-1}_{hyidp}}$
13         $SAMLResp := (id_{req}.id_{sp}.id_{idp}.SAMLAssrtn)$
14         send $SAMLResp$ to $SP$
15 function aggregate($soruceList, Att - Agg^u$)
16     if $isEmpty(soruceList)$ then
17         return $Att - Agg^u$
18     end
19     show ConsentPage($soruceList$)
20     On receive ($releaseAttribute$)
21         return $Att - Agg^u$
22     On receive ($userChoice$)
23         $x := userChoice$
24         if $x \in OAuthSource$ then
25             initiate OAuth protocol using the respective app
26         end
27         if $x \in OpenIDSource$ then
28             initiate OpenID protocol
29         end
30     On receive ($response$)
31         validate response using the respective protocol
32         $idp := response.idp$
33         $PROFILE^u_{idp} := response.PROFILE^u_{idp}$
34         $Att - Agg^u := Att - Agg^u \cup PROFILE^u_{idp}$
35         return aggregate($soruceList \setminus idp, Att - Agg^u$)
```

D. ARCHITECTURE: TYPE 2

One particular weakness of the Type 1 architecture of Social Anchor is that the Hybrid IdP has an overarching control over the attributes released from the external IdPs during the aggregation process. This enables it to be in a position to abuse the released attributes and hence, a user has to trust the Hybrid IdP that it will not undermine her trust. One way to tackle this is to introduce a corrective mechanism based on the Identity Relay (IR) mode so that the Hybrid IdP has no knowledge of what attributes are released from an IdP. The Type 2 architecture of Social Anchor aims to introduce this corrective mechanism. The architecture is illustrated in Figure 10.

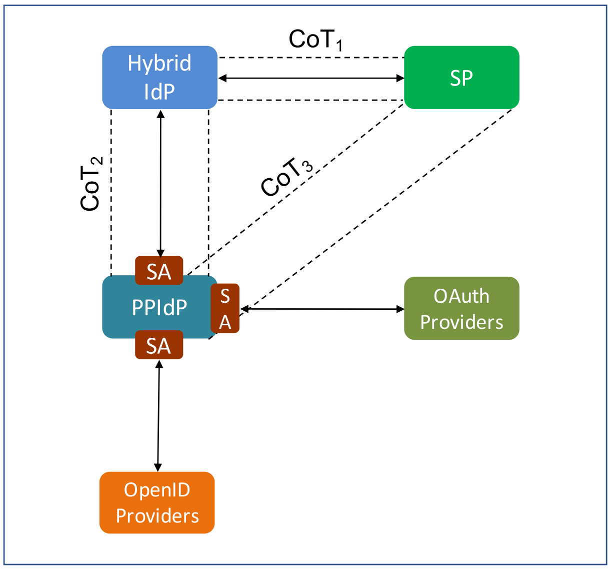


FIGURE 10. Social anchor: Type 2 architecture.

This architecture leverages two IdPs: Hybrid IdP and Portable Personal IdP (PPIdP). A PPIdP is a special type of IdP introduced in [32] which can be installed in any smart-device of a user. It consists of a Personal Attribute Store (PAS) with the required interfaces to add and store attributes securely within those devices and release them to an SP. In such, the PPIdP and any attributes stored within it remain under the full control of the user. We have extended the PPIdP with the functionalities of Social Anchor so that it can aggregate attributes from different SN IdPs as presented in Section V-C. Furthermore, the Hybrid IdP has been equipped with the IR mode of attribute aggregation. As per the architecture, the SP needs to be federated with the Hybrid IdP and PPIdP (no necessarily within the same federation), as illustrated with notations CoT_1 and CoT_3 in Figure 10. On the other hand, the Hybrid IdP and the PPIdP must belong to the same federation (CoT_2 in Figure 10).

The aggregation algorithms used in these two IdPs are presented in Algorithm 2 and Algorithm 3. As before, the algorithm in each IdP has two phases: setup and aggregation. During the setup phase in Algorithm 2, the PPIdP is added as an external IdP in Hybrid IdP and other SN IdPs are added as external IdPs in PPIdP. In addition, the sets of aggregated attributes in each IdP ($Att - Agg^u_{hyidp}$ and $Att - Agg^u_{ppidp}$) are initialised as null. The aggregation in this architecture has two phases. In the first phase, upon receiving a SAML authentication request, the Hybrid IdP delegates the authentication task to the PPIdP by creating a new SAML authentication request. Once the PPIdP receives the authentication request, it follows the same aggregation procedure described in Section V-C and illustrated in Algorithm 3. Once the aggregation is complete at PPIdP, an encrypted SAML assertion, containing the aggregated attributes, is created in such a way that only the target SP can decrypt it. The encrypted assertion is embedded into a regular SAML assertion as a special attribute.

Algorithm 2 Attribute Aggregation Algorithm at Hybrid IdP

```
Input: The request for attribute aggregation from an SP
Output: A SAML response to the requested SP
Start
    Set $Att - Agg^u_{hyidp} := null$
    Set $soruceList_{hyidp} := \{PPIdP\}$
    On receive ($AuthnReq$) at $HyIdP$
        $id_{req} := AuthnReq.id_{req}$
        $id_{sp} := AuthnReq.id_{sp}$
        $Att - Agg^u_{hyidp} =$ aggregateHyIDP($soruceList_{hyidp}$,$Att - Agg^u_{hyidp}$)
        $SAMLAssrtn := \{Att - Agg^u_{hyidp}\}_{K^{-1}_{hyidp}}$
        $SAMLResp := (id_{req}.id_{sp}.id_{idp}.SAMLAssrtn)$
        send $SAMLResp$ to $SP$
function aggregateHyIDP($soruceList$,$Att - Agg^u$)
    if $isEmpty(soruceList)$ then
        return $Att - Agg^u$
    end
    show ConsentPage($soruceList$)
    On receive ($releaseAttribute$)
        return $Att - Agg^u$
    On receive ($userChoice$)
        $x := userChoice$
        if $x = PPIdP$ then
            prepare $AuthnReq$ for $PPIdP$ and send it to $PPIdP$
        end
    On receive ($response$)
        validate response using the respective protocol
        $ppidp := response.idp$
        $PROFILE^u_{ppidp} := response.PROFILE^u_{ppidp}$
        $Att - Agg^u := Att - Agg^u \cup PROFILE^u_{ppidp}$
        return aggregateHyIdP($soruceList \setminus idp$,$Att - Agg^u$)
```

Algorithm 3 Attribute Aggregation Algorithm at PPIdP

```
Input: The request for attribute aggregation from the Hybrid IdP
Output: A SAML response to the Hybrid IdP
Start
    Set $Att - Agg^u_{ppidp} := null$
    Set $OAuthSource_{ppidp} := \{Facebook \cup LinkedIn \cup Twitter\}$
    Set $OpenIDSource_{ppidp} := \{Yahoo \cup Google \cup LiveJournal \cup VeriSign\}$
    Set $soruceList_{ppidp} := \{OAuthSource_{ppidp} \cup OpenIDSource_{ppidp}\}$
    On receive ($AuthnReq$) at $PPIdP$
        $id_{req} := AuthnReq.id_{req}$
        $id_{sp} := AuthnReq.id_{sp}$
        $Att - Agg^u_{ppidp} =$ aggregatePPIdP($soruceList$,$Att - Agg^u_{ppidp}$)
        $SAMLAssrtn := \{Att - Agg^u_{ppidp}\}_{K^{-1}_{ppidp}}$
        $EncSAMLAssrtn := \{SAMLAssrtn\}_{K_{sp}}$
        $SAMLAssrtn' := \{EncSAMLAssrtn\}_{K^{-1}_{ppidp}}$
        $SAMLResp := (id_{req}.id_{sp}.id_{idp}.SAMLAssrtn')$
        send $SAMLResp$ to $Hybrid\ IdP$
function aggregatePPIdP($soruceList$,$Att - Agg^u$)
    if $isEmpty(soruceList)$ then
        return $Att - Agg^u$
    end
    show ConsentPage($soruceList$)
    On receive ($releaseAttribute$)
        return $Att - Agg^u$
    On receive ($userChoice$)
        $x := userChoice$
        if $x \in OAuthSource$ then
            initiate OAuth protocol using the respective app
        end
        if $x \in OpenIDSource$ then
            initiate OpenID protocol
        end
    On receive ($response$)
        validate response using the respective protocol
        $idp := response.idp$
        $PROFILE^u_{idp} := response.PROFILE^u_{idp}$
        $Att - Agg^u := Att - Agg^u \cup PROFILE^u_{idp}$
        return aggregatePPIdP($soruceList \setminus idp$,$Att - Agg^u$)
```

Finally, a SAML response is created using the regular assertion and returned to the Hybrid IdP where it is validated. The Hybrid IdP then extracts the encrypted assertion and treats it as a special attribute. Since the assertion is encrypted, the Hybrid IdP has no knowledge of the sets of aggregated attributes. The Hybrid IdP then follows the same procedure to return the encrypted assertion as an attribute within a SAML response to the SP. The SP extracts the encrypted assertion, decrypts it, validates it and then finally extracts the sets of aggregated attributes from different IdPs. A detailed protocol flow utilising this architecture is presented in Section VI.

E. IMPLEMENTATION

We have developed a Proof of Concept implementing two architectures of Social Anchor by extending the code-base of the Hybrid model. In this section, we detail how the proof of concept has been developed and implemented.

As in [2], SimpleSAMLphp has been used for the implementation of Hybrid IdP both in Type 1 and Type 2 architectures. The SimpleSAMLphp provides the *Multiauth* and *Consent* modules to be used for the following two purposes:

(a) The *Multiauth* module allows other SN IdPs to be added as external IdPs so that it can delegate the authentication task to them.

(b) The *Consent* module enables the selective disclosure and explicit consent of attributes at the Hybrid IdP.

Specifically, we have amended the code-base of the Multiauth module so that it can allow a user to initiate protocol flows with different SN IdPs in a single session. Similarly, we have extended the code-base of the Consent module so that the aggregation process from different SN IdPs can be completed and the aggregated attributes can be displayed to the user within the same session.

For PPIdP, we have extended the PPIdP proof of concept, developed as an Android App as presented in [32] by adding the following capabilities for the Type 2 Social Anchor architecture.

- Adding the attribute aggregation mechanism from different OpenID and OAuth based SN IdPs in a single session, and
- Implementing a module, similar to the *Multiauth* module in SimpleSAMLphp, that will allow the PPIdP to interact with an OpenID and OAuth based IdP and then delegate any authentication task to such external IdPs.
- Implementing a consent module, similar to the *Consent* module in SimpleSAMLphp, to act as the interface for attribute aggregation and to enable the selective disclosure of attributes within PPIdP.

Also, in the previous proof of concept of the Hybrid model, the Hybrid IdP would provide a user with the option to toggle between IP and IR mode. However, in the current implementation we have made amendments in such a way that the IR mode is automatically selected when the Hybrid IdP delegates the authentication task to the PPIdP. To enable this, the Hybrid IdP just passes on the reference of the SP (its entity ID), via a hidden field, in the authentication request.

In addition, we have kept the following features of Hybrid model as presented in [2]:

- The same data structure, a list of lists, has been utilised to represent the aggregated data in both architectures.
- The modified simlpeSAMLphp codebase so that it can handle such data structure both in the Hybrid IdP and a SAML SP.

For this implementation, seven external IdPs have been added to the Hybrid IdP for the Type 1 architecture and to the PPIdP for the Type 2 architecture. The added IdPs are: Yahoo, Google, LiveJournal, VeriSign, Facebook, LinkedIn and Twitter. Among them, the former four represent OpenID providers and the latter three are OAuth IdPs.

Different deployment strategies have been sought for creating the federations among different entities in Social Anchor. The SAML SP has been federated with the Hybrid IdP in a traditional way by exchanging their corresponding metadata offline for both architectures. On the other hand, PPIdP has been added as an authentication source with the Hybrid IdP by creating a federation among them in a dynamic fashion as presented in [32], [33], [35].

F. TRUST ANALYSIS

The issue of trust is a fundamental concept in SAML as different entities within the SAML Circle of Trust (CoT) need to trust each other inside the federation. The integration of non-SAML social IdPs with the SAML IdP in Social Anchor influences this notion of trust. Hence, it is crucial to analyse the trust issues between different entities in Social Anchor.

Towards this aim, we utilise the mathematical model of trust for FIM introduced in [36]. According to this model, trust is of two types: Direct Trust (DT) and Indirect Trust (IT) [37]. A direct trust implies a direct trust relationship between the entities based on some experiences and evidences. On the other hand, the indirect trust, also known as Transitive Trust, represents an indirect trust relationship between two entities which have been established using a referral from one or more intermediate third parties. The notation T is used to denote the set of trust types. Therefore, $T = \{DT \bigcup UT\}$.

Every trust relationship is accompanied by a scope which signifies the specific purpose in which the relationship makes sense. The trust strength represents the degree (level) of trust a trustor has over a trustee. There might be different trust scopes in different scenarios, however, for Social Anchor, the set of trust scopes, denoted with S, consists of only one scope denoted with FED which implies that the scope is valid within a federated domain. That is, S is a singleton set with $S = \{FED\}$.

One important characteristic of trust is that it exhibits the transitivity property [38]: if an entity A trusts another entity B in a scope and B trust another entity C within the same scope, a trust relation can be derived between A and C.

Trust in a dynamic federation is modelled using three types [33]. These three types with their respective trust strength is discussed below.

- Entities (IdPs and SPs) in a traditional SAML federation are regarded as *Fully Trusted* as they trust each other under their contractual conditions. Here, the trust strength is denoted with FT.
- SPs added dynamically to an IdP inside the federation in a dynamic fashion under *some conditions* without any legal contract are regarded as *Semi-trusted* to the IdP. Here, the trust strength is denoted with ST.
- Non-SAML SN IdPs are regarded as *Untrusted* entities to a SAML IdP (e.g. Hybrid IdP and PPIdP). Here, the trust strength is denoted with UT.

Among these, UT has the lowest trust strength implying that a trustor does not trust a trustee at all. On the other hand, FT implies the highest strength when the trustor and trustee are part of a traditional federation. Thus, the ranking of trust strengths is:

$$UT < ST < FT.$$

To indicate an entity $e_1 \in E_f$ (the trustor) has $t \in T$ trust over an entity $e_2 \in E_f$ (the trustee) with a trust scope

$$\dfrac{\dfrac{\left[SP \xleftrightarrow[FT]{DT\,:\,FED} HyIdP\right] \qquad \left[HyIdP \xrightarrow[ST]{DT\,:\,FED} PPIdP\right]}{\left[SP \xrightarrow[ST]{DT\,:\,FED} PPIdP\right]} \qquad \left[PPIdP \xrightarrow[UT]{DT:FED} idp_{sn}\right]}{\left[SP \xrightarrow[UT]{IT:FED} idp_{sn}\right]}$$

FIGURE 11. Trust relationship between a non-SAML IdP and SP.

(signifies the specific purpose in which a trust relationship is valid) of $s \in S$ and the trust strength (signifies the level of trust a trustor has over a trustee) of v in a federation f, the following notation is used:

$$e_1 \xrightarrow[v]{t\,:\,s} e_2$$

Next, we can model the required trust relationships between different entities in Type 1 architecture in the following manner:

- Between Hybrid IdP (denoted with *HyIdP*) and SP:

$$HyIdP \xleftrightarrow[FT]{DT\,:\,FED} SP$$

- Between a non-SAML SN IdP (denoted as idp_{sn}) and Hybrid IdP:

$$HyIdP \xrightarrow[UT]{DT\,:\,FED} idp_{sn}$$

- Between a non-SAML SN IdP and the SP, equation as shown at the bottom of this page. This implies that the SP would have an indirect trust to any SN IdP which is based on the trust transitivity property. The trust strength is calculated based on the trust rules presented in [36] which essentially state that the calculated trust must assume the lowest value within the trust transitivity chain. In this case, between FT and UT, the trust rules will calculate UT as the trust strength as it has the lowest trust strength between these two.

Similarly, we can model the required trust relationships between different entities in Type 2 architecture in the following manner:

- Between Hybrid IdP, PPIdP and SP:
  - Between the SP and Hybrid IdP:

$$SP \xleftrightarrow[HT]{DT\,:\,FED} HyIdP$$

  - Between the Hybrid IdP and PPIdP:

$$HyIdP \xrightarrow[ST]{DT\,:\,FED} PPIdP$$

  - Between the SP and PPIdP:

$$SP \xrightarrow[ST]{DT\,:\,FED} PPIdP$$

- Between a non-SAML SN IdP and PPIdP:

$$PPIdP \xrightarrow[UT]{DT\,:\,FED} idp_{sn}$$

- The relationship between a non-SAML SN IdP and the SP is illustrated in Figure 11.

VI. PROTOCOL FLOW

In the following, we present the protocol flows utilising the two architectures of Social Anchor. The first flow is based on the Type 1 architecture illustrating the steps for attribute aggregation from different SN IdPs using the Hybrid IdP. On the other hand, the second flow is based on the Type 2 architecture showcasing the steps for the same attribute aggregation process using the Hybrid IdP and PPIdP which is more privacy-friendly as discussed above.

A. TYPE 1

In the first use-case, we illustrate the simplest form of attribute aggregation using Social Anchor where a user needs to access a service provided by a SAML SP after releasing attributes from several non-SAML IdPs. To illustrate this use-case, let us consider this situation where a staff of a university would like apply for another position in the same university via their career portal (the service provider). During her previous employment period, the user completed some certification courses in different social networks such as LinkedIn and Google (e.g. Google Certification) which she would like to submit during her application to the service provider. This scenario is motivated from the use-case presented in [15] where a user accesses services from an organisation upon releasing attributes from a single social IdP. We have extended their use-case to include attribute aggregation from multiple social IdPs in a single session. Within this setting, it is assumed that the university has integrated

$$\dfrac{\left[SP \xleftrightarrow[FT]{DT\,:\,FED} HyIdP\right]\left[HyIdP \xrightarrow[UT]{DT\,:\,FED} idp_{sn}\right]}{\left[SP \xrightarrow[UT]{IT\,:\,FED} idp_{sn}\right]}$$

Social Anchor with their central IdP which is acting as the Hybrid IdP. That is, the SP (the career portal) is deployed using the modified SimpleSAMLphp as presented in [2] and is federated with their IdP in a traditional way by exchanging their corresponding metadata. Other non-SAML IdPs also have been added to the Social Anchor as per the method described in Section V-E. This means that for OpenID IdPs, necessary endpoints have been set up at the Hybrid IdP and for OAuth IdPs, an app is installed at the Authorization Server (AS) to facilitate the interaction between the Hybrid IdP and OAuth IdP.

With this setup, the protocol flow for the first use-case is illustrated in Figure 12 and is described below:

i A user visits the SP to access one of its services. The user clicks the respective service and is forwarded to the WAYF (Where Are You From) Page of the SP to choose an IdP. The user chooses the Hybrid IdP.

ii The SP reads the required attribute(s) for the requested service from the configuration file. These attributes are embedded inside a SAML authentication request with which the user is redirected to the Hybrid IdP.

iii The consent form at the Hybrid IdP shows the list of attributes along with their respective IdPs to indicate to the user the attributes that need to be aggregated from different SN IdPs (line 19 in Algorithm 1).

iv The user initiates the attribute aggregation function by clicking the *Aggregate More Attributes* button. The Hybrid IdP creates a session to keep track of the previous attributes and then forwards the user to the IdP selection page of the Hybrid IdP. This page contains the list of those SN IdPs from where the user has not aggregated attributes in the current session.

v The user chooses one SN IdP (line 22 in Algorithm 1) and based on the user's selection, the respective protocol is initiated. For example, if the user chooses an OAuth-based IdP (line 23 in Algorithm 1), she is forwarded to the respective IdP following the OAuth protocol. Similarly, if the user chooses OpenID (line 27 in Algorithm 1), a HTML form is shown to the user to input the user's OpenID. Once the OpenID is provided and the Login button is clicked, the usual OpenID protocol starts and the user is forwarded to the OpenID provider.

vi The user is authenticated at the respective IdP. Then, she releases the attributes within a response to the Hybrid IdP using their respective protocol.

vii The Hybrid IdP validates the response using their corresponding mechanisms (line 31 in Algorithm 1). Then, the attributes are extracted from the response and the previously aggregated attributes are retrieved using the session. The two sets of attributes are then merged (line 34 in Algorithm 1) and are shown to the user by grouping them based on the IdP in the consent form.

viii The requested attributes from the SP are shown on the consent form for the user to determine if she needs to aggregate more attributes from other SN IdPs. If so, the steps iv-vii can be repeated.

ix Once the required attributes have been aggregated, she decides to select and release those attributes. Based on her selection, the attributes from each SN IdP are wrapped inside a SAML assertion which is then embedded inside a SAML response and is sent back to the SP (line 14 in Algorithm 1).

x Upon receiving the response, the SP validates the assertion and retrieves the set of attributes from each SN IdP. The SP then can take authorisation decisions based on those attributes.

B. TYPE 2

In the second use-case, we illustrate the applicability of the Type 2 Architecture of Social Anchor to address the privacy-invasive nature in the Type 1 Architecture: the Hybrid IdP has the ultimate control and knowledge of the user's attributes released from different SN IdPs. Following our university scenario as presented in Section VI-A, let us consider the scenario when the user does not wish to expose the aggregated attributes to the Hybrid IdP. Instead, the user wants the aggregated attributes to be released to the SP (the career portal), via the Hybrid IdP, in such a way that the Hybrid IdP would have no knowledge regarding the aggregated attributes. The Type 2 Architecture would enable the user to achieve this goal by allowing her to aggregate attributes using Social Anchor within a PPIdP so that users have the ultimate control over the aggregated attributes.

Towards this aim, during the setup phase, the PPIdP has been equipped with the Social Anchor feature by adding different non-SAML IdPs as authentication sources, similar to the Hybrid IdP in the first use-case. Then, the PPIdP has been installed within a smart mobile device of a user and then is federated with the Hybrid IdP and SP using the mechanism of Dynamic SAML as presented in [33]. In addition, as in the first use-case, the SP has been deployed using the modified SimpleSAMLphp and has been federated with the Hybrid IdP of Social Anchor in a traditional way by exchanging their corresponding metadata. Finally, the Hybrid IdP has been equipped with an extended IR (Identity Relay) capability as discussed previously.

With this setup, the protocol flow for the second use-case is illustrated in Figure 13 and is described below. For brevity, we have omitted the first three steps which are similar to the ones illustrated in the first use-case.

i The user wishes to aggregate attributes from non-SAML IdPs without revealing them to the Hybrid IdP.

ii The user is forwarded to the IdP selection page of the Hybrid IdP, as before (line 17 in Algorithm 2), after creating a session to keep track of the previously aggregated attributes. However, this time the page shows only the PPIdP.

iii When the user chooses the PPIdP, a SAML authentication request with a special attribute called

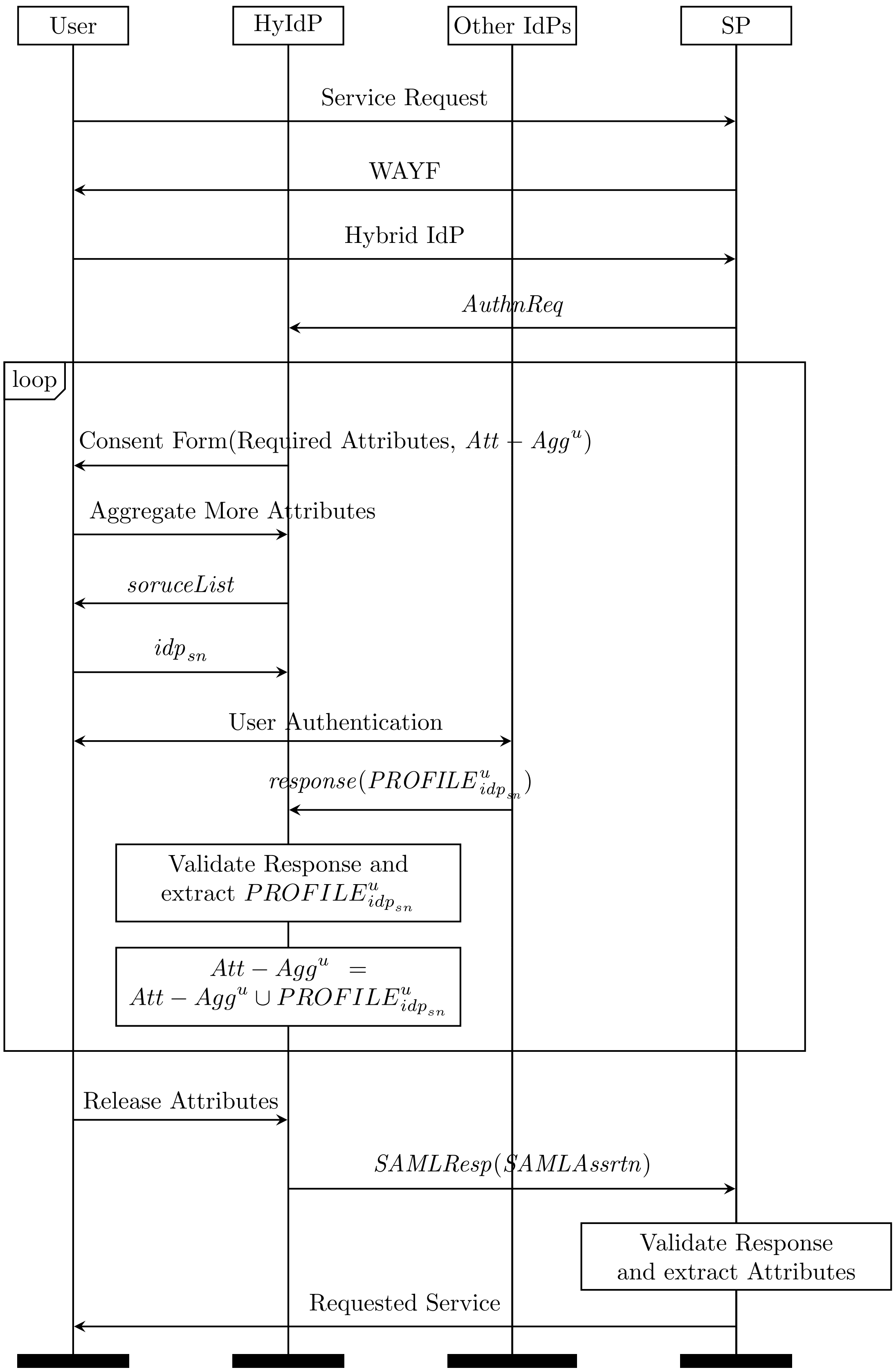

User
HyIdP
Other IdPs
SP
Service Request
WAYF
Hybrid IdP
$AuthnReq$
loop
Consent Form(Required Attributes, $Att-Agg^u$)
Aggregate More Attributes
$soruceList$
idp_{sn}
User Authentication
$response(PROFILE^u_{idp_{sn}})$
Validate Response and extract $PROFILE^u_{idp_{sn}}$
$Att-Agg^u = Att-Agg^u \cup PROFILE^u_{idp_{sn}}$
Release Attributes
$SAMLResp(SAMLAssrtn)$
Validate Response and extract Attributes
Requested Service

FIGURE 12. Protocol flow for Use-case 1.

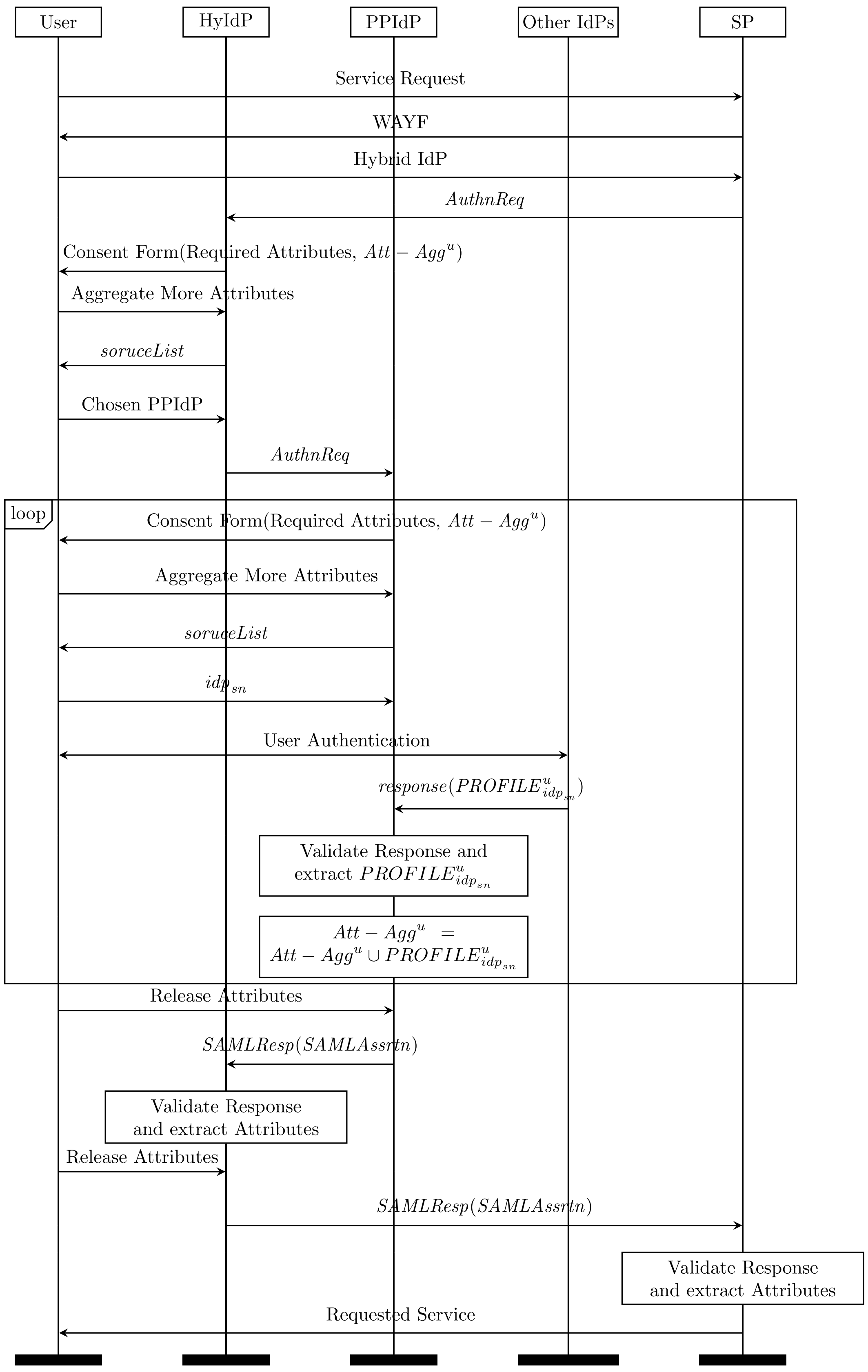
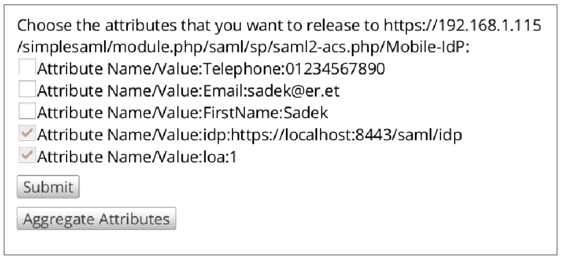

User
HyIdP
PPIdP
Other IdPs
SP
Service Request
WAYF
Hybrid IdP
AuthnReq
Consent Form(Required Attributes, $Att - Agg^u$)
Aggregate More Attributes
soruceList
Chosen PPIdP
AuthnReq
loop
Consent Form(Required Attributes, $Att - Agg^u$)
Aggregate More Attributes
soruceList
idp_{sn}
User Authentication
$response(PROFILE^u_{idp_{sn}})$
Validate Response and extract $PROFILE^u_{idp_{sn}}$
$Att - Agg^u = Att - Agg^u \cup PROFILE^u_{idp_{sn}}$
Release Attributes
$SAMLResp(SAMLAssrtn)$
Validate Response and extract Attributes
Release Attributes
$SAMLResp(SAMLAssrtn)$
Validate Response and extract Attributes
Requested Service

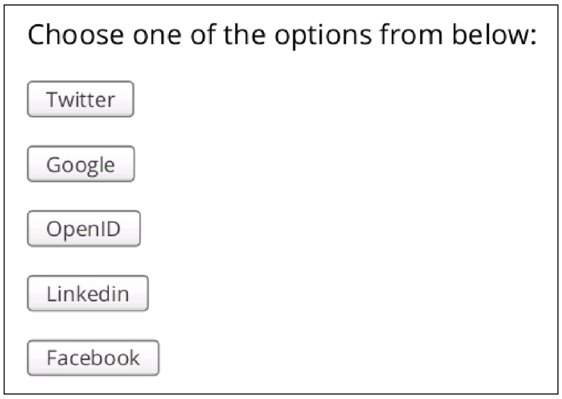
FIGURE 13. Protocol flow for Use-case 2.



Choose the attributes that you want to release to https://192.168.1.115 /simplesaml/module.php/saml/sp/saml2-acs.php/Mobile-IdP:

- [ ] Attribute Name/Value:Telephone:01234567890
- [ ] Attribute Name/Value:Email:sadek@er.et
- [ ] Attribute Name/Value:FirstName:Sadek
- [x] Attribute Name/Value:idp:https://localhost:8443/saml/idp
- [x] Attribute Name/Value:loa:1

FIGURE 14. **Consent form at the PPIdP.**



Choose one of the options from below:

FIGURE 15. **Available IdPs at the PPIdP.**

EmbedAssertion is forwarded to the PPIdP (line 23 in Algorithm 2). This special attribute is used to indicate that the receiving IdP is requesting an encrypted assertion to be returned.

iv The PPIdP notes the *EmbedAssertion* attribute in the authentication request and the entity ID of the SP is retrieved and stored in a session variable.

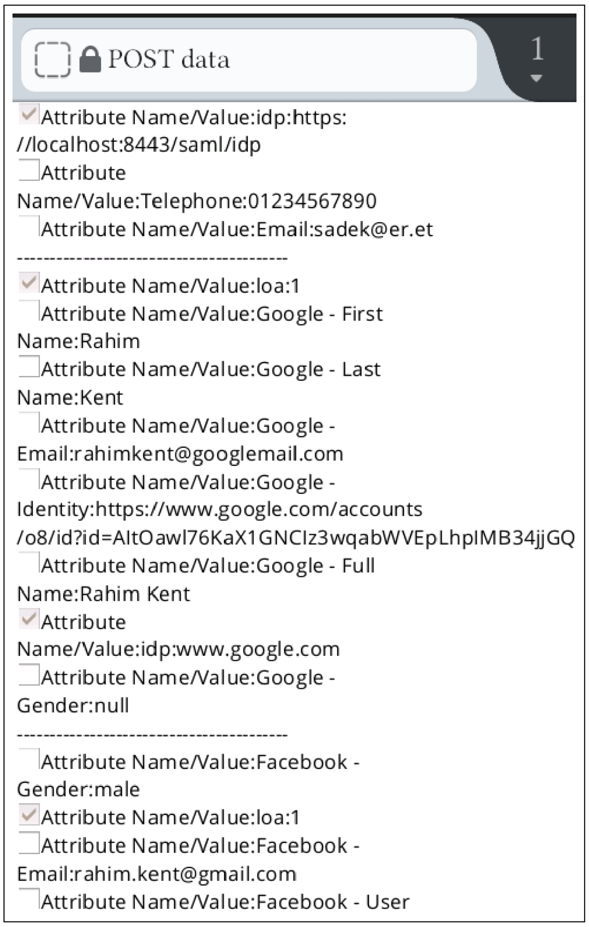
v The user is authenticated at the PPIdP and then the consent form containing attributes stored at this IdP are shown to the user (Figure 14). The form has a *Aggregate Attributes* button which the user can click to initiate the attribute aggregation process.

vi Assuming the user clicks the *Aggregate Attributes* button, a new page with a list of SN IdPs is shown to the user (Figure 15, line 21 in Algorithm 3). The current implementation of the PPIdP allows the user to aggregate attributes from Twitter, Google, OpenID, LinkedIn and Facebook.

vii The user chooses one of the IdPs (lines 24-31 in Algorithm 3) and she is forwarded to the corresponding IdP where she is authenticated. Then, she comes back to the PPIdP with attributes as discussed in the first use-case.

viii The user can repeat steps v-vii to aggregate more attributes from other IdPs. After aggregating attributes from multiple IdPs (line 36 in Algorithm 3), the screenshot of the consent form at the PPIdP is given in Figure 16.

ix The user chooses the attributes that she wants to release and clicks the *Submit* button. The PPIdP utilises the



- [x] Attribute Name/Value:idp:https: //localhost:8443/saml/idp
- [ ] Attribute Name/Value:Telephone:01234567890
- [ ] Attribute Name/Value:Email:sadek@er.et

- [x] Attribute Name/Value:loa:1
- [ ] Attribute Name/Value:Google - First Name:Rahim
- [ ] Attribute Name/Value:Google - Last Name:Kent
- [ ] Attribute Name/Value:Google - Email:rahimkent@googlemail.com
- [ ] Attribute Name/Value:Google - Identity:https://www.google.com/accounts /o8/id?id=AItOawl76KaX1GNCIz3wqabWVEpLhpIMB34jjGQ
- [ ] Attribute Name/Value:Google - Full Name:Rahim Kent
- [x] Attribute Name/Value:idp:www.google.com
- [ ] Attribute Name/Value:Google - Gender:null

- [ ] Attribute Name/Value:Facebook - Gender:male
- [x] Attribute Name/Value:loa:1
- [ ] Attribute Name/Value:Facebook - Email:rahim.kent@gmail.com
- [ ] Attribute Name/Value:Facebook - User

FIGURE 16. **Aggregated attributes at the PPIdP.**

SP entity ID retrieved from the SAML authentication request. This entity ID is used to retrieve the public key of SP (K_{SP}) from its metadata repository. The public key is then used to create an encrypted assertion (line 12-13 in Algorithm 3) for the SP using the selected attributes.

x The encrypted assertion is then added as a special attribute called *encryptedAssertion* and attached within a a regular unencrypted SAML assertion. The PPIdP then sends the regular assertion back to the Hybrid IdP via a SAML response (lines 16 in Algorithm 3).

xi Upon receiving the regular assertion, it is validated by the Hybrid IdP. Then, the *encryptedAssertion* attribute is retrieved and shown on the consent page in a specific way.

xii When the user chooses the encrypted assertion and clicks the *Release attributes to SP* button, a SAML assertion containing the *encryptedAssertion* attribute is created which is then sent back to the SP (lines 10-12 in Algorithm 2).

xiii The assertion is validated by the SP. Then, the embedded *encryptedAssertion* attribute is retrieved and is treated in a special way. The encrypted assertion is at first decrypted and then validated. Afterwards, all the attributes aggregated at the PPIdP are retrieved. Then, the SP can take an authorisation decision which we do not elaborate any further.

In this way, a user can aggregate attributes using Social Anchor at PPIdP in a privacy-friendly way. The user has the ultimate control on how the attributes are aggregated and the Type 2 architecture ensures that Hybrid IdP has no knowledge whatsoever regarding which attributes are released to the SP.

VII. DISCUSSION

In this section, we discuss different aspects of Social Anchor. Specifically, we analyse how Social Anchor has satisfied different requirements (Section VII-A). During the security analysis, we provide a formal analysis of Social Anchor protocols using AVISPA. Then, we present a comparative performance analysis between a traditional federation and Social Anchor (Section VII-B). Finally, we explore the advantages of Social Anchor and the current limitations it has (Section VII-C).

A. FUNCTIONAL, SECURITY & PRIVACY ANALYSIS

In the following, we explore how Social Anchor satisfies different functional, privacy and security requirements.

1) FUNCTIONAL ANALYSIS

The Social Anchor deployed in the Hybrid IdP and the PPIdP enables these IdPs to interact with different SN IdPs utilising the underlying OpenID and OAuth protocols, thereby satisfying the *F1* requirement. Similarly, Social Anchor also enables both the Hybrid IdP and PPIdP to validate any security token or response received from different SN IdPs and then extract attributes released by them. In this way, the F2 requirement is also satisfied.

2) PRIVACY ANALYSIS

Social Anchor deployed both in the Hybrid IdP and PPIdP enables a user to select each single attribute from any SN IdP before they are released to the SP. In addition, a user must provide her explicit consent, by selecting the attributes to be released and by clicking the Submit button within the consent form, to release any attribute to the SP. Both these features enable Social Anchor to satisfy requirements *P1* and *P2*. Furthermore, the user can choose between Architecture type 1 and Architecture type 2 to control when attributes can or cannot be exposed to the Hybrid IdP. This satisfies the requirement *P3*. Finally, the attributes are aggregated only at the Hybrid IdP and PPIdP, depending on the architecture being used. Both these IdPs are highly trusted from the perspective of a user and hence, the requirement *P4* is satisfied as well.

3) SECURITY ANALYSIS

Each SN IdP utilised in our implementation have strict secure procedures during the registration as well as authentication process. These procedures are rigorously implemented as the reliability and reputation of online services provided by such IdPs heavily rely on such procedures. This leads us to conclude that the requirement *S1* is satisfied.

Our implementation heavily relies on HTTPS communication channels to ensure that any transmitted data is not altered during transmission and is not disclosed to any unauthorised third party, thereby satisfying requirements *S2* and *S3*.

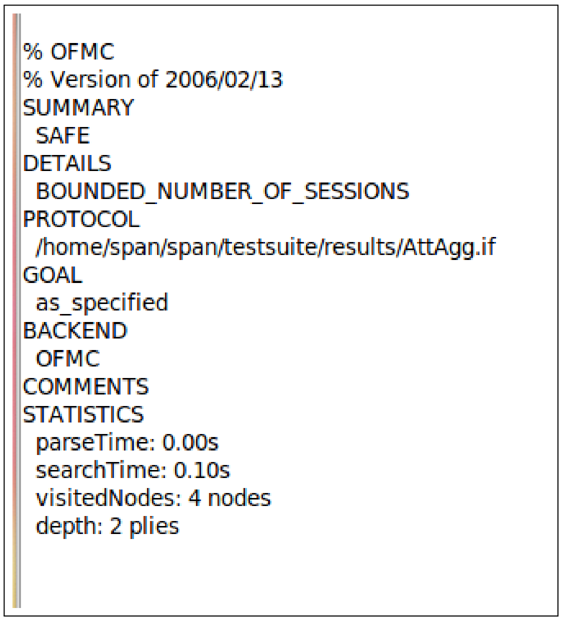
We have utilised standardised protocols in our implementation while engaging in interactions between different entities. This is to ensure that the security of the system is well preserved. Even so, Social Anchor employs complex interactions with many components from different systems. It is well understood that the security of any such complex systems is difficult to ensure. One prominent approach to guarantee the security of any such complex system is to formally analyse its security. Towards this aim, we have formally analysed the security of Social Anchor using a tool called AVISPA (Automated Validation of Internet Security Protocols and Applications).

AVISPA [3], [4] is utilised for formal modelling and verification of security protocols with the aim of determining if certain security properties are satisfied. The security protocol in AVISPA is modelled using the High-Level Protocol Specification Language (HLPSL) which is based on Lamport's Temporal Logics of Actions (TLA) [39]. In HLPSL, different entities of a security protocol are modelled using roles and their interactions using transitions along with a set of security goals under the knowledge assumption of a Dolev-Yao attacker model as described in Section V-A.

In practice, protocol formalisation using AVISPA is a two step process: specification and verification.

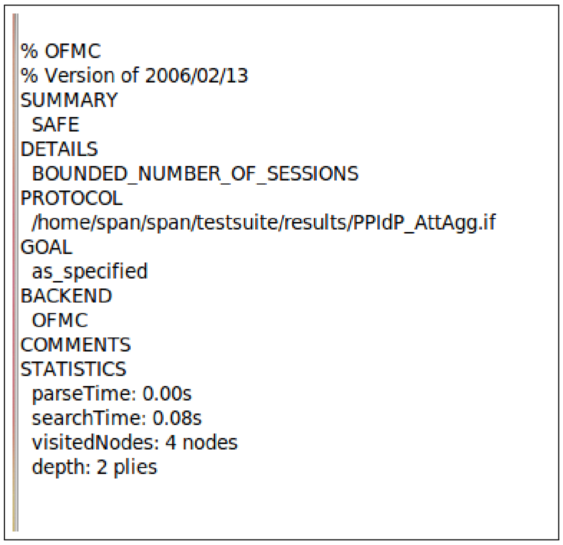
- The first step is the specification where the protocol is specified in a formal way. This also is a two step process:
  - Specifying the protocol in Alice-Bob (A-B) notation, which provides a clear illustration of which entities are involved and what messages are exchanged between them.
  - Modelling the protocol and its security goals in HLPSL and saving the formalisation within an HLPSL file.
- The last step is the verification of the protocol which is carried out by executing the AVISPA tool with the saved HLPSL file to determine if the security goals are satisfied. For this, the HLPSL file is passed into a translator which translates the HLPSL format into a low level language called the *Intermediate Format (IF)*. The translated IF is then passed into one of the four back-ends that AVISPA currently supports: the On-the-fly Model-Checker (OFMC), Constraint-Logic-based Attack Searcher (CL-AtSe), the SAT-based Model-Checker (SATMC) and the Tree Automata based on Automatic Approximations for the Analysis of Security Protocols (TA4SP). The back-end checks if the security goals are satisfied and outputs accordingly.

HLPSL currently supports two different types of security goals: *secrecy* and *authentication*. Here, *secrecy* refers to the



```
% OFMC
% Version of 2006/02/13
SUMMARY
  SAFE
DETAILS
  BOUNDED_NUMBER_OF_SESSIONS
PROTOCOL
  /home/span/span/testsuite/results/AttAgg.if
GOAL
  as_specified
BACKEND
  OFMC
COMMENTS
STATISTICS
  parseTime: 0.00s
  searchTime: 0.10s
  visitedNodes: 4 nodes
  depth: 2 plies
```

FIGURE 17. AVISPA result for type 1 architecture.



```
% OFMC
% Version of 2006/02/13
SUMMARY
  SAFE
DETAILS
  BOUNDED_NUMBER_OF_SESSIONS
PROTOCOL
  /home/span/span/testsuite/results/PPIdP_AttAgg.if
GOAL
  as_specified
BACKEND
  OFMC
COMMENTS
STATISTICS
  parseTime: 0.00s
  searchTime: 0.08s
  visitedNodes: 4 nodes
  depth: 2 plies
```

FIGURE 18. AVISPA result for type 2 architecture.

goal asserting if a certain value should be kept secret between only two entities and corresponds to our *S1* (spoofing) security requirement. Conversely, the *authentication* goal checks if two entities are properly authenticated to each other and agree on the required data during their interaction. In such, this corresponds to *S2* (tampering) requirement.

Our formal analysis of Social Anchor is based on the work presented in [40] where three well-known Identity Management protocols - SAML, OpenID and OAuth - have been formally analysed using AVISPA. We have adopted their analysis and extended it as required for Social Anchor.

As supplementary materials, we have attached the Alice-Bob notations for Social Anchor Type 1 and Type 2 architectures in two files, Listing 1.txt and Listing 2.txt respectively. In addition, the corresponding HLPSL files for Social Anchor Type 1 and Type 2 architectures are provided in AttAgg.hlpsl and PPIdPAttAgg.hlpsl files respectively. Each specified role in each HLPSL file essentially captures the interactions presented in the respective Alice-Bob notation. The security goals within each HLPSL file signify different security requirements for different interactions. The security of protocol can be verified by saving each of the presented HLPSL formalisation in an HLPSL file and executing the AVISPA tool with the file and the *ofmc* back-end. The verification results, presented in Figure 17 and Figure 18, will indicate that the modelled protocols are secure against the specified security goals. This essentially verifies that all security goals, encoding the security requirements, are satisfied by Social Anchor.

B. PERFORMANCE ANALYSIS

It is crucial that the performance of any newly proposed system is analysed using different experiments to show its suitability. There are different metrics such as latency (response time), throughput and so on, which can be used to measure the performance of any system. Among all these, we consider latency/response time to be the most crucial metric for Social Anchor. This is because throughput mostly measures the number of requests a system can handle per unit time. Since Social Anchor is developed and deployed using web technologies such as Apache and PHP and these technologies can provide substantial throughput, this metric has not been used to analyse the performance of Social Anchor. There are different interpretations for the latency, therefore, it is important to outline what they mean in the context of Social Anchor. In our experiment, latency defines the elapsed time between a user request is submitted to the system and a response is generated by the system.

As the first step, we have carried out some base test cases so as to collect base performance metrics and use them for subsequent analysis. Next, we explain the experiment setup and the procedure for this base case which has utilised a simple SAML setup in which a single SAML federation consisting of an IdP and an SP has been created using SimpleSAMLphp. The experiment has been carried out within a simulated environment where multiple users have been simulated to use the base test case in an automatic fashion and in the background, the required metrics have been collected. After that, the metrics have been analysed and different graphs are generated to investigate the performance of this base test case. The simulated environment consists of a desktop computer with a Core-i5 machine and 8GB RAM connected to a 100Mbps LAN.

User interactions have been simulated with Selenium [41], a widely-used load testing tools for web services, where each interaction involves submitting a request to access the SP. This request initiates a SAML authentication request to the IdP where the authentication is carried out automatically. After this, a few pre-defined attributes are automatically selected to create the corresponding SAML response, which is sent back to the SP. After validating the response, the requested page is returned to the requesting Selenium thread. Finally, a request to log the user out has been submitted. This base case experiment consists of ten rounds where in each round we have gradually increased the number of simulated users. For example, the first round consists of one simulated user, the second round consists of two simulated users and so on. Again, each simulated user has been configured to generate five consecutive requests, generating five interactions described previously. For each of these interactions, we have measured the elapsed time, between the request submitted and the response received, both for login and logout scenarios at the service provider. The measured elapsed time acts as the base performance metrics.

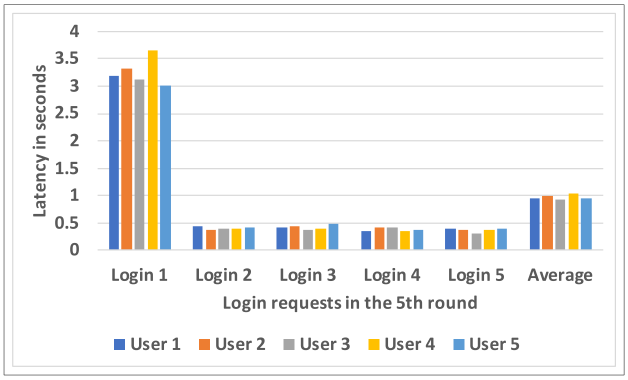


FIGURE 19. Login latency.

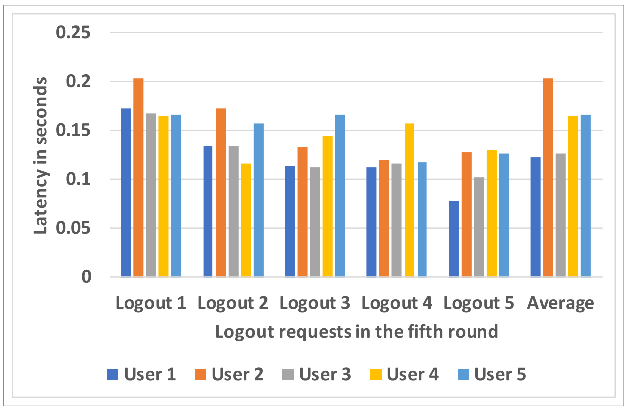


FIGURE 20. Logout latency.

Next, we present several graphs analysing the collected metrics. The first two graphs, Figure 19 and Figure 20, present the elapsed time for login and logout respectively for five simultaneous users in the fifth round. It is clear from the graphs that there is a specific pattern both for logins and logouts. For example, the first requests (denoted with *Login 1* in Figure 19) have incurred the maximum delay (around 3 sec) for all users where other subsequent requests (denoted with *Login 2*, *Login 3*, *Login 4* and *Login 5* in Figure 19) have taken less than half a second. This is because the first request is to initiate the SAML authentication flow where all the subsequent requests have taken advantage of the Single Sign On facility of SAML, thereby reducing the elapsed time significantly. However, there have been little differences in elapsed time among different logout requests (denoted with *Logout 1*, *Logout 2*, *Logout 3*, *Logout 4* and *Logout 5* in Figure 20) for all users. Login and logout graphs for other rounds have exhibited similar patterns and have been skipped for brevity.

In Figure 21 and Figure 22, we have plotted three different graphs with respect to different simultaneous accesses by users in different rounds. For example, *1 User* denotes the first round with only a single user submitting the (login/logout) request using Selenium, *2 Users* denotes the second round with two simultaneous users submitting the requests and so on. Three different graphs plotted in Figure 21 and Figure 22 represent the maximum, minimum and average latency for logins and logouts respectively for different simultaneous access. From Figure 21, it is clear that the maximum login time (time representing the elapsed period for the first request) increases with the number of simultaneous access in different rounds: 0.99s for a single user (in the first round), around 3s for five users (in the fifth round) and 4.85s for 10 users (in the tenth round). This is because of how Selenium simulates multiple users at the same time, has nothing to do with the performance of SimpleSAMLphp. Selenium uses threads to imitate simultaneous users and as the number of threads increases, Selenium itself starts to take longer time to switch between the generated threads which ultimately has caused the elapsed time to increase. However, the minimum login time remain mostly same for all simultaneous access in different rounds. Therefore, the maximum login time mostly controls the average login time as demonstrated in Figure 21.

On the other hand, Figure 22 of logout latency clearly illustrates a similar trend like login latency, where the latency increases as the number of simultaneous users increases. However, the deviations among the elapsed time in three different graphs here are much less in comparison to their login counterparts. For example, the maximum and minimum latencies in the first round are .27s and .12s respectively, in the fifth round are .32s and .22s respectively and in the tenth round are .38s and .29s respectively.

TABLE 3. Social anchor test cases for login.

Cases	Description	Latency
Case 1	Single SAML	T1
Case 2	SA Type 1 Google	T2
Case 3	SA Type 1 Facebook	T3
Case 4	SA Type 1 Aggr from Google and Facebook	T4
Case 5	SA Type 2 Aggr from Google and Facebook	T5

Next, we analyse the impact of latency of Social Anchor in an abstract fashion. For this we have considered five different test cases as presented in Table 3 where *SA* and *Aggr* imply Social Anchor and Aggregation respectively. Here, Case 1

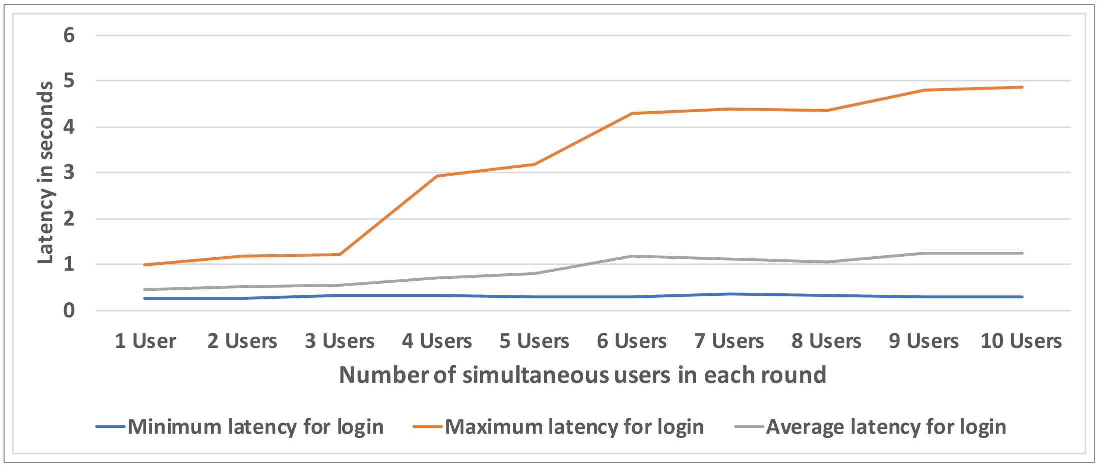


FIGURE 21. Login latency for different test cases.

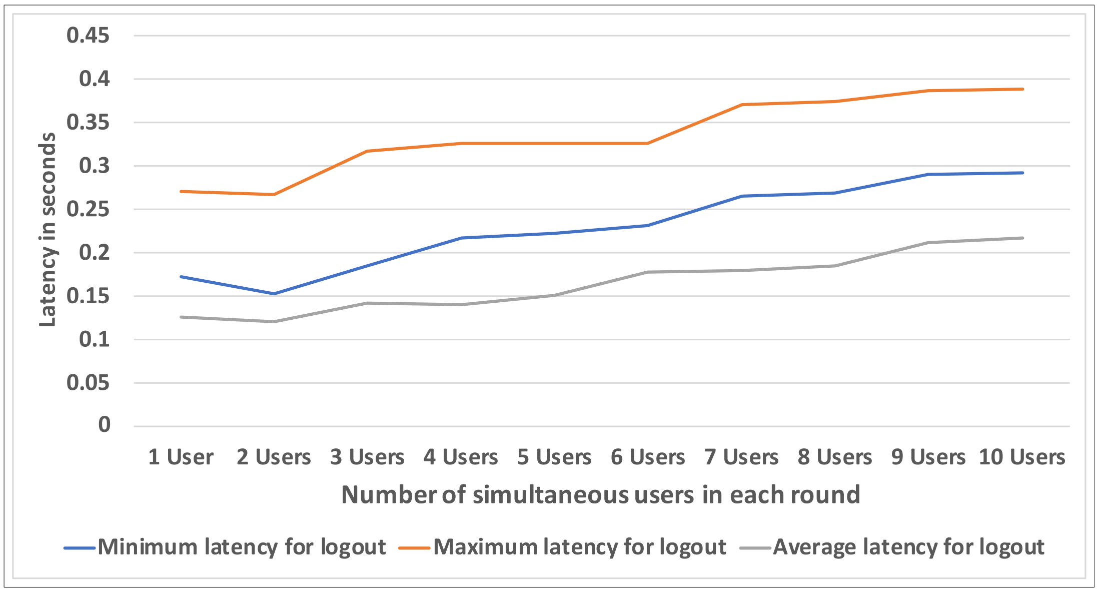


FIGURE 22. Logout latency for different test cases.

represents the base test case involving a traditional SAML federation as described previously. Case 2 and 3 represent the test cases for releasing attributes from Google and Facebook respectively using the Social Anchor Type 1 architecture, without aggregating attributes from multiple IdPs. On the other hand, Case 4 and 5 represent the attribute aggregation test cases involving Type 1 and Type 2 architectures of Social Anchor respectively. The third column in each test case is their corresponding representation of latency. In the following, we analyse the relation between T1 to T5.

AS, T1 represents the login latency for a simple federation setup as explained before and hence $T1 = 0.99s$ as per Figure 21 for User 1. However, unlike $T1$, the latencies denoted by $T2$ to $T3$ have not been collected empirically with an experiment. This is because of two reasons explained below:

- In such empirical experiments, the latency will be highly dependent on how quickly/slowly a user selects/aggregates attributes from other SN IdPs. Different users have different cognitive understanding and hence, might take different times to select/aggregate attributes. The more time a user takes, the higher the latency becomes and thus, it might have a detrimental effect on the latency of Social Anchor.
- The respective test cases (Case 2 to Case 4) would require to interact with SN IdPs such as Google and Facebook. Thus, the corresponding latencies (T2 to T4) will highly depend on the interaction time with these entities which itself will vary depending on the bandwidth of the access network from where a user accesses such services. An empirical experiment will thus be irrelevant.

Continuing with the analysis, $T2$ and $T3$ will be certainly higher than $T1$. This is because the simple federation was setup in the same local computer from where the experiment was carried out. On the other hand, $T2$ and $T3$ would require to interact with Google and Facebook respectively for releasing attributes. However, the comparison between $T2$ and $T3$ will be meaningful only after an empirical experiment. Therefore, we relate them in this manner: $T2 \simeq T3$. Similarly, $T4$ and $T5$ will be higher than $T1, T2$ and $T3$. This is because $T4$ and $T5$ will be dependant of the number and type of IdPs from where the attributes are aggregated. Like before, we model the relationship between $T4$ and $T5$ as $T4 \simeq T5$ because the same reasons explained before. Finally, the relation between all latencies can be represented in the following way:

$$T1 < (T2 \simeq T3) < (T4 \simeq T5)$$

It is to be noted that we have excluded the Logout analysis for Social Anchor. This is because the logout procedure in Social Anchor, like any traditional SAML federation, submits a SAML logout request to the Hybrid IdP and hence, the latency will be similar to that of any traditional SAML federation which has been already presented in Figure 22.

C. ADVANTAGES & LIMITATIONS

Social Anchor provides a number of advantages which are summarised below:

- It is the first system to enable a user to aggregate attributes from multiple SN IdPs in a privacy-friendly fashion and then release them to a SAML SP in a single session. This opens up the door of opportunities for providing innovative online service which have not been possible until now.
- It provides an overlay of privacy protection for a user while aggregating attributes from different SN IdPs. As our findings suggest, almost all of the chosen SN IdPs do not have any facility for the selective disclosure of attributes while attributes are released. To address this, Social Anchor allows a user, via its consent modules in the Hybrid IdP and PPIdP, to choose attributes and provide explicit consent before they are released to an SP.
- Social Anchor empowers a user by enabling her to exercise the ultimate control on how attributes from different SN IdPs are released. This is achieved by allowing the user to choose the IdP where the aggregation can take place: either at the Hybrid IdP or at the PPIdP. In particular, by allowing the user to aggregate attributes at the PPIdP and then release to the SP via the Hybrid IdP, Social Anchor ensures that the aggregation takes place at the most trusted entity which is under the full control of the user.
- Another important feature of Social Anchor is its support for security. The whole flow is carried out under HTTPS channels and is based on standardised security protocols. This ensures that the attribute aggregation takes place in a secure fashion. Our formal analysis using AVISPA testifies the security goals of Social Anchor.

Social Anchor, unfortunately, suffers from a few limitations which are summarised below:

- The number of interactions for aggregating attributes from multiple SN IdPs can be quite high. In a traditional setting where a user releases attributes only from one single IdP to an SP the interaction is quite straightforward: just between a user, the IdP and SP. However, an attribute aggregation process naturally requires much more interactions involving different entities. This might raise questions regarding the usability of Social Anchor. In future, we would like to investigate this issue.
- The Type 2 Architecture of Social Anchor is reliant on PPIdP. Since, in its current form, the PPIdP can only be accessed from a mobile device where it is installed, the aggregation process can be initiated only from a browser of the corresponding mobile device. In future, we would like to investigate how this limitation can be addressed by utilising a smart-contract supporting blockchain system where the smart-contract would act like a PPIdP.

VIII. RELATED WORK

In this section, we explore some of the previous related works on attribute aggregation within the domain of federated identity. We also analyse their scopes with respect to Social Anchor and compare them using six criteria: *federated setting*, *social network*, *attribute aggregation*, *selective disclosure*, *explicit consent* and *formal verification*. These six criteria, essentially, represent the core themes of the present work: *attribute aggregation in a federated setting from different social networks with privacy features of selective disclosure and explicit consent and a formal verification of its protocols*.

The initial concept of attribute aggregation from different identity providers within a federated identity domain was introduced by Klingenstein *et al.* in [26]. The authors also presented the theoretical conceptualisation of different (Type 1) attribute aggregation models within a federated identity domain by different entities, such as identity providers, service providers and so on. However, the concept of aggregating attributes from different social networks within a federated domain was not explored. The authors also did not explore how attributes could be selectively disclosed with explicit consent during the attribute aggregation process. There was no implementation of any of the proposed models and hence, the formal analysis of protocols was not considered.

The first implementation of attribute aggregation in a federated identity setting was presented in [42] where a novel user-centric model of (Type 1) attribute aggregation was introduced. The users, using that model, can initiate and control the attribute aggregation process from different

TABLE 4. Comparison of existing works with social anchor.

	Federation	Social Net.	Att. Agg.	Sel. Dis.	Exp. Con.	Formal V.
Klingenstein et al. [26]	✓	✗	Type 1	✗	✗	✗
Chadwick et al. [42]	✓	✗	Type 1	✗	✓	✗
Chadwick et al. [43]	✓	✗	Type 1	✓	✓	✗
Chadwick et al. [15]	✓	✓	✗	✗	✓	✗
Ferdous et al. [10]	✓	✗	Type 1	✗	✗	✗
Ferdous et al. [2]	✓	✗	Type 1 & 2	✓	✓	✗
Jin et al. [45]	✗	✗	✓(from devices)	✗	✗	✓
Jin et al. [46]	✗	✗	✓(from devices)	✗	✗	✓
Qian et al. [47]	✗	✗	✓(from devices)	✗	✗	✓
Social Anchor	✓	✓	✓	✓	✓	✓

federated IdPs. However, their implementation did not allow to aggregate attributes from different social networks. Besides, the concept of selective disclosure and formal verification of protocols were absent.

In [43], Chadwick *et al.* presented a visual way of (Type 1) attribute aggregation from different federated IdPs in a federated setting. For this, the authors utilised the concept of Windows CardSpace [44]. CardSpace provides a logical representation of the concept of plastic cards (e.g credit/debit cards, loyalty cards, ID cards and so on) that are widely used to store different attributes in the physical world. The core idea in CardSpace is to represent attributes in different identity providers using logical cards where each card contains the attributes from the corresponding identity provider. Even though the concept was no longer supported by Windows, the authors utilised this notion to create a visual representation of attribute aggregation process supporting selective disclosure and explicit consent. However, their proposal does not allow to aggregate attributes from social networks and there was no formal verification of their presented protocols.

The authors in [15] presented a novel mechanism to link and utilise social network attributes with organisational identities within a federated setting. Their method enables the users to access restricted resources of federated service providers using their social network attributes with explicit consents. However, they did not explore the concept of attribute aggregation with selective disclosure and did not formally verify their security protocols.

Ferdous *et al.* (in [10]) analysed different (Type 1) attribute aggregation models against a set of functional, security, privacy and trust requirements in a federated identity setting. Using these requirements, the authors compared each of these models to identify their relative strengths and weaknesses. This was a mere theoretical analysis with no implementation of any of the presented models and thus, does not meet the most of the selected criteria (see below).

In our previous work [2], we presented a hybrid model of (Type 1 and 2) attribute aggregation from SAML IdPs within a federation with explicit consent and selective disclosure. The current work, as mentioned earlier, is an extension of this work and has modified the hybrid model to enable Type 1 and Type 2 attribute aggregation from different social networks. Also, there was no formal verification the corresponding security protocols in [2].

In addition to the federated identity domain, the concept of attribute aggregation was also considered in other application domains. Next, we explore a few of such existing works. The works of Jin *et al.* [45], [46] aim to build a system which collects/aggregates sensory data from the mobile devices of different users. Their proposal relies on an incentive mechanism in order to encourage others to share their mobile sensor data. This is in contrast to Social Anchor as the users in Social Anchor are self-motivated to aggregate and share their personal data in order to access some online services and hence, do not rely on any incentive mechanism. The notion of privacy is handled in a different way in their works. For example, their system requires to ensure that the privacy of different users is guaranteed while aggregating sensor data from different devices of different users. On the other hand, Social Anchor tries to ensure the privacy of a single user by enabling the user to control what data she wants to release to a service provider.

In another work, Qian *et al.* [47] proposed a system to aggregate user behaviour data from multiple devices so that it could facilitate a privacy-preserving analysis over this data set by any third party. Their system uses Homomorphic Encryption and Differential Privacy to achieve this goal. Their notion of privacy is, however, different than ours. For example, the main focus of their work is to enable privacy-preserving analysis whereas we mainly focus on releasing data to a service provider in a privacy-friendly manner by giving users the required control before any data is released. That is why we consider their work not to be very relevant to our work.

A comparison of the analysed works with respect to Social Anchor is provided in Table 4 against the set of selected criteria. Here, Att. Agg. is the shortened expression for Attribute Aggregation while Sel. Dis., Exp. Con. and Formal V. represent Selective Disclosure, Explicit Consent and Formal Verification respectively. We have used the notation '✓' to indicate a work meets the corresponding criterion while the notation '✗' implies the corresponding criterion is not met by the respective work. It is clearly evident from Table 4, other than Social Anchor, none of the previous works meets the selected criteria.

IX. CONCLUSION

In this article, we have presented Social Anchor - a system to aggregate attributes from different SN IdPs in a privacy-friendly fashion. This is the first system to enable users to aggregate attributes from these IdPs and release them to a SAML SP in a semi-automatic way so that the users do not need to provide such information manually. The whole system has been designed and developed with the aim of empowering users in preserving their privacy. In particular, Social Anchor facilitates the selective disclosure of attributes, a feature absent in most SN IdPs. Even worse, users are often obscured from what attributes are released to an SP from these IdPs. Social Anchor in a way provides an overlay of privacy protection that allows a user to know and to select each single piece of attribute that will be released to a SAML SP. Furthermore, it releases such attributes only after the respective users provides her explicit consent.

We have presented two different architectures of Social Anchor to illustrate two different deployment strategies. The first architecture is simple to deploy and has no access limitation. On the other hand, the second architecture provides more empowerment to a user as it can be used to aggregate attributes at a personal and portable IdP which is under the full control of a user. However, it has limitations in the sense that attributes can be only be aggregated from a browser of a mobile device where such an IdP is deployed. Having based on the work of [2], Social Anchor satisfies a number of functional, security and privacy requirements which have been analysed in details in the article. Specifically, we have presented a formal analysis of Social Anchor using AVISPA to ensure its security during its complex protocol flow. Finally, the performance of Social Anchor has been analysed.

With the raising popularities of different SN IdPs, it is safe to assume that they will store an increasing number of our sensitive attributes. These attributes represent a lucrative asset and can be leveraged to access a wide range of online services. As the current setting dictates, such attributes are scattered across different IdPs and there is no mechanism to aggregate such attributes at a trusted entity so that they can be released in a single session to an SP. Social Anchor breaks this barrier and provides an opportunity to introduce a new era of online services. That is why we are confident that Social Anchor will be an important tool for providing next generation online services while protecting the privacy of users.

In future, we would like to explore how novel emerging technologies can be utilised to improve the architecture of Social Anchor. In recent years, smart-contract enabled blockchain systems such as Ethereum [48] and Hyperledger [49] have emerged as one of the most influential technologies offering numerous possibilities. We would like to investigate how such blockchain systems can be incorporated into the architecture of Social Anchor to improve its current limitations and to extend its current range of functionalities.

REFERENCES

[1] *Number of Social Media Users Worldwide From 2010 to 2021 (in Billions)*. Accessed: Aug. 10, 2019. [Online]. Available: https://www.statista.com/statistics/278414/number-of-worldwide-social-network-users/

[2] M. S. Ferdous, F. Chowdhury, and R. Poet, "A hybrid model of attribute aggregation in federated identity management," in *Enterprise Security*, vol. 10131. Basel, Switzerland: Springer, 2017, pp. 120–154.

[3] A. Armando, D. Basin, Y. Boichut, Y. Chevalier, L. Compagna, J. Cuellar, P. H. Drielsma, P. C. Heám, O. Kouchnarenko, J. Mantovani, S. Mödersheim, D. von Oheimb, M. Rusinowitch, J. Santiago, M. Turuani, L. Viganò, and L. Vigneron, "The AVISPA tool for the automated validation of Internet security protocols and applications," in *Proc. Int. Conf. Comput. Aided Verification*, 2005, pp. 281–285.

[4] *AVISPA Project*. Accessed: Jul. 5, 2019. [Online]. Available: http://www.avispa-project.org/

[5] M. S. Ferdous, A. Jøsang, K. Singh, and R. Borgaonkar, "Security usability of petname systems," in *Proc. Nordic Conf. Secure IT Syst.* in Lecture Notes in Computer Science, vol. 5838. Berlin, Germany: Springer, 2009, pp. 44–59.

[6] M. S. Ferdous, G. Norman, and R. Poet, "Mathematical modelling of identity, identity management and other related topics," in *Proc. 7th Int. Conf. Secur. Inf. Netw.*, 2014, pp. 9–16.

[7] A. Jøsang, M. Al, and Z. Suriadi, "Usability and privacy in identity management architectures," in *Proc. ACSW*, 2007, pp. 143–152.

[8] D. W. Chadwick, "Federated identity management," in *Foundations of Security Analysis and Design V* Lecture Notes in Computer Science, vol. 5705, A. Aldini, G. Barthe, and R. Gorrieri, Eds. Berlin, Germany: Springer-Verlag, Berlin, Jan. 2009, pp. 96–120.

[9] M. Sadek Ferdous, M. J. M. Chowdhury, M. Moniruzzaman, and F. Chowdhury, "Identity federations: A new perspective for bangladesh," in *Proc. Int. Conf. Informat., Electron. Vis. (ICIEV)*, May 2012, pp. 219–224.

[10] M. S. Ferdous and R. Poet, "Analysing attribute aggregation models in federated identity management," in *Proc. 6th Int. Conf. Secur. Inf. Netw. (SIN)*, 2013, pp. 181–188.

[11] OASIS Standard. (Mar. 15, 2005). *Assertions and Protocols for the OASIS Security Assertion Markup Language (SAML) V2.0*. Accessed: Jul. 23, 2019. [Online]. Available: http://docs.oasis-open.org/security/saml/v2.0/saml-core-2.0-os.pdf

[12] D. Recordon and D. Reed, "OpenID 2.0: A platform for user-centric identity management," *Proc. 2nd ACM Workshop Digit. Identity Manage.*, pp. 11–16, 2006.

[13] (Dec. 5, 2007). *OpenID Authentication 2.0 - Final*. Accessed: Jul. 23, 2019. [Online]. Available: http://openid.net/specs/openid-authentication-2_0.html

[14] D. Hardt. (2012). *The OAuth 2.0 Authorization Framework*. Accessed: Jul. 23, 2019. [Online]. Available: https://tools.ietf.org/html/rfc6749

[15] D. W. Chadwick, G. L. Inman, K. W. S. Siu, and M. S. Ferdous, "Leveraging social networks to gain access to organisational resources," in *Proc. 7th ACM workshop Digit. Identity Manage. (DIM)*, New York, NY, USA, 2011, pp. 43–52.

[16] *Shibboleth*. Accessed: May 1, 2019. [Online]. Available: https://www.internet2.edu/products-services/trust-identity/shibboleth/

[17] *SimpleSAMLphp*. Accessed: Jun. 12, 2019. [Online]. Available: http://simplesamlphp.org/

[18] *ZXID*. Accessed: Jun. 12, 2019. [Online]. Available: http://www.zxid.org/.

[19] (Dec. 6, 2016). *OpenID Simple Registration Extension 1.1–Draft 1*. [Online]. Available: http://openid.net/specs/openid-simple-registration-extension-1_1-01.html

[20] (Dec. 5, 2007). *OpenID Attribute Exchange 1.0-Final*. Accessed: Jul. 23, 2019. [Online]. Available: http://openid.net/specs/openid-attribute-exchange-1_0.html

[21] *OAuth 2.0*. Accessed: Jul. 23, 2019. [Online]. Available: http://oauth.net/2.

[22] (May 15, 2019). *Introducing OAuth 2.0*. Accessed: Jul. 23, 2019. [Online]. Available: http://hueniverse.com/2010/05/introducing-oauth-2-0/

[23] *Facebook Developer Centre*. Accessed: Nov. 1, 2098. [Online]. Available: https://developers.facebook.com/

[24] *LinkedIn Developer Network*. Accessed: Nov. 8, 2019. [Online]. Available: https://www.linkedin.com/secure/developer

[25] *Twitter Developer Centre*. Accessed: Nov. 8, 2019. [Online]. Available: https://dev.twitter.com/

[26] N. Klingenstein, "Attribute aggregation and federated identity," in *Proc. Int. Symp. Appl. Internet Workshops*, Jan. 2007, p. 26.

[27] B. Hulsebosch, M. Wegdam, B. Zoetekouw, N. van Dijk, and R. P. van Wijnen, *Virtual Collaboration Attribute Management*. vol. 1. Utrecht, The Netherlands: Surf Net, 2011.

[28] S. Myagmar, and A. J. Lee, and W. Yurcik, "Threat modeling as a basis for security requirements," in *Proc. Symp. Requirements Eng. Inf. Secur. (SREIS)*, 2005, pp. 1–8.

[29] L. Desmet, B. Jacobs, F. Piessens, and W. Joosen, "Threat modelling for Web services based Web applications," in *Communications and Multimedia Security*. Boston, MA, USA: Springer, 2005, pp. 131–144.

[30] D. De Cock, K. Wouters, D. Schellekens, D. Singelee, and B. Preneel, "Threat modelling for security tokens in Web applications," in *Communications and Multimedia Security*. Boston, MA, USA: Springer, 2005, pp. 183–193.

[31] D. Dolev and A. Yao, "On the security of public key protocols," *IEEE Trans. Inf. Theory*, vol. 29, no. 2, pp. 198–208, Mar. 1983.

[32] M. S. Ferdous and R. Poet, "Portable personal identity provider in mobile phones," in *Proc. 12th IEEE Int. Conf. Trust, Secur. Privacy Comput. Commun.*, Jul. 2013, pp. 736–745.

[33] Md. Sadek Ferdous and Ron Poet, "Dynamic identity federation using security assertion markup language (SAML)," in Policies and Research in Identity Management in IFIP Advances in Information and Communication Technology, vol. 396, S. Fischer-Hubner, E. Leeuw, and C. Mitchell, Eds. Berlin, Germany: Springer 2013, pp. 131–146.

[34] *UK Access Management Federation for Education and Research*. Accessed: Feb. 21, 2020. [Online]. Available: https://www.ukfederation.org.uk/

[35] S. Ferdous and R. Poet, "Managing dynamic identity federations using security assertion markup language," *J. Theor. Appl. Electron. commerce Res.*, vol. 10, no. 2, pp. 53–76, 2015.

[36] M. S. Ferdous, G. Norman, A. Jøsang, and R. Poet, "Mathematical modelling of trust issues in federated identity management," in *Proc. IFIP Int. Conf. Trust Manage.*, 2015, pp. 13–29.

[37] U. Kylau, I. Thomas, M. Menzel, and C. Meinel, "Trust requirements in identity federation topologies," in *Proc. Int. Conf. Adv. Inf. Netw. Appl.*, 2009, pp. 137–145.

[38] A. Jøsang, E. Gray, and M. Kinateder, "Simplification and analysis of transitive trust networks," *Web Intell. Agent Syst., Int. J.*, vol. 4, no. 2, pp. 139–161, 2006.

[39] L. Lamport, "The temporal logic of actions," *ACM Trans. Program. Lang. Syst.*, vol. 16, no. 3, pp. 872–923, 1994.

[40] M. S. Ferdous and R. Poet, "Formalising identity management protocols," in *Proc. 14th Annu. Conf. Privacy, Secur. Trust (PST)*, Dec. 2016, pp. 137–146.

[41] *Selenium*. Accessed: Sep. 1, 2019. [Online]. Available: https://www.seleniumhq.org/

[42] D. W. Chadwick and G. Inman, "Attribute aggregation in federated identity management," *Computer*, vol. 42, no. 5, pp. 33–40, May 2009.

[43] D. W. Chadwick and G. Inman, "The trusted attribute aggregation service (TAAS)–Providing an attribute aggregation layer for federated identity management," in *Proc. Int. Conf. Availability, Rel. Secur.*, Sep. 2013, pp. 285–290.

[44] D. Chappell. (Apr. 2006). *MSDN*. Accessed: Nov. 8, 2019. [Online]. Available: http://msdn.microsoft.com/enus/library/aa480189.aspx

[45] H. Jin, L. Su, H. Xiao, and K. Nahrstedt, "Inception: Incentivizing privacy-preserving data aggregation for mobile crowd sensing systems," in *Proc. 17th ACM Int. Symp. Mobile Ad Hoc Netw. Comput.*, 2016, pp. 341–350.

[46] H. Jin, L. Su, H. Xiao, and K. Nahrstedt, "Incentive mechanism for privacy-aware data aggregation in mobile crowd sensing systems," *IEEE/ACM Trans. Netw.*, vol. 26, no. 5, pp. 2019–2032, Oct. 2018.

[47] J. Qian, F. Qiu, F. Wu, N. Ruan, G. Chen, and S. Tang, "Privacy-preserving selective aggregation of online user behavior data," *IEEE Trans. Comput.*, vol. 66, no. 2, pp. 326–338, Feb. 2017.

[48] *Ethereum*. Accessed: Sep. 1, 2019. [Online]. Available: https://ethereum.org/

[49] *Hyperledger*. Accessed: Sep. 1, 2019. [Online]. Available: https://www.hyperledger.org/

MD. SADEK FERDOUS received the dual master's degree in security and mobile computing from the Norwegian University of Science and Technology, and the Norway and University of Tartu, Estonia, and the Ph.D. degree in identity management from the University of Glasgow. He is currently working as an Assistant Professor with the Department of Computer Science and Engineering, Shahjalal University of Science and Technology, Sylhet, Bangladesh. He is also a Research Associate with the Centre for Global Finance and Technology, Imperial College Business School. He has several years of experience of working as a Postdoctoral Researcher in different universities in different European and UK-funded research projects. He has published numerous research articles and book chapters in these domains in different books, journals, conferences, workshops, and symposiums. His current research interests include blockchain, identity management, trust management and security and privacy issues in cloud computing, and social networks.

FARIDA CHOWDHURY received the joint master's degree in networking and e-business centred computing from the University of Reading, U.K., the Universidad Carlos III de Madrid, Spain, and the Aristotle university of Thessaloniki, Greece, and the Ph.D. degree from the University of Stirling, U.K., where she investigated the effect of churn in NAT-ed Structured Peer-to-Peer Overlays. She is currently an Associate Professor with the Department of Computer Science and Engineering, Shahjalal University of Science and Technology, Bangladesh. She has published many articles in reputed journals and as book chapters as well as in different conferences and workshops. Her research interests include networking, blockchain, big data, cloud computing, HCI, and security and privacy issues in social networks.

MADINI O. ALASSAFI received the M.S. degree in computer science from California Lutheran University, United State of America, in 2013, and the Ph.D. degree in cloud computing security from the University of Southampton, U.K., in 2018. He is currently working as the Chairman and as an Assistant Professor with the Information Technology Department, Faculty of Computing and Information Technology, King Abdulaziz University, Saudi Arabia. He has published numerous research articles in different conferences, journals and book chapters. His current research interests mainly focus on cloud computing and security, distributed systems, blockchain, the Internet of Things (IoT) security issues, cloud security adoption, risks, cloud migration project management, cloud of things, and security threats.

ABDULRAHMAN A. ALSHDADI received the Ph.D. degree in cloud computing from the University of Southampton, Southampton, U.K., in February 2018. He is currently an Assistant Professor of computer science in computer science and engineering (CCSE) with the University of Jeddah. He is also acting as the Deputy of the Big Data Centre at Maakah Province and the Vice Dean of College of Computer Science and Engineering (CCSE), University of Jeddah, Jeddah, Saudi Arabia. He has published numerous conference papers, journal articles and book chapters and is being involved in several funded projects. His research interests span mainly around Industry 4.0 pertaining issues of cloud computing and fog computing security, the Internet of Things (IoT) and smart cities, intelligent systems, deep learning, and data science analytics and modelling.

VICTOR CHANG received the Ph.D. degree in computer science from the University of Southampton. He was a Senior Associate Professor, the Director of Ph.D. from June 2016 to May 2018, and the Director of MRes from September 2017 to February 2019 with the International Business School Suzhou (IBSS), Xi'an Jiaotong-Liverpool University (XJTLU), Suzhou, China, from June 2016 to August 2019. He was also a Very Active and a contributing key member at the Research Institute of Big Data Analytics (RIBDA), XJTLU. He was an Honorary Associate Professor with the University of Liverpool. Previously, he was a Senior Lecturer at Leeds Beckett University, U.K., from September 2012 to May 2016. He has been a Full Professor of data science and information systems with the School of Computing, Engineering and Digital Technologies, Teesside University, Middlesbrough, U.K., since September 2019. He published three books as sole authors and the editor of two books on *Cloud Computing* and related technologies. He gave 18 keynotes at international conferences. During his career, he received multiple awards, such as European Award on Cloud Migration, in 2011, the IEEE Outstanding Service Award, in 2015, and best paper awards, in 2012, 2015, and 2018. He has also been a Visiting Scholar/Ph.D. Examiner at several universities, the Editor-in-Chief of IJOCI & OJBD journals, the Editor of *Future Generation Computer Systems*, an Associate Editor of TII & Information Fusion, the Founding Chair of two international workshops, and the founding Conference Chair of IoTBDS and COMPLEXIS, since 2016. He has also been the founding Conference Chair for FEMIB, since 2019.