J Internet Serv Appl (2011) 2:47–65
DOI 10.1007/s13174-011-0020-4

ORIGINAL PAPER

Mitigating the linkability problem in anonymous reputation management

M. Hussain · D.B. Skillicorn

Received: 6 January 2011 / Accepted: 20 April 2011 / Published online: 15 May 2011

Abstract Trust plays a key-role in enhancing user experience at service providers. Reputation management systems are used to quantify trust, based on some reputation metrics. Anonymity is an important requirement in these systems, since most individuals expect that they will not be profiled by participating in the feedback process. Anonymous Reputation management (ARM) systems allow individuals to submit their feedback anonymously. However, this solves part of the problem. Anonymous ratings by one individual can be linked to each other. This enables the system to easily build a profile of that individual. Data mining techniques can use the profile to re-identify that individual. We call this the linkability problem. This paper presents an anonymous reputation management system that avoids the linkability problem. This is achieved by constructing a system that empowers individuals to interact and rate service providers, securely and anonymously.

Keywords Trust · Reputation management · Security · Privacy · Anonymity

1 Introduction

Trust is a vital aspect in our daily life, especially for the web. The higher trust a user has in a service provider, the more comfortable that user is in interacting with that service. One way to quantify trust is based on reputation. The reputation-based approach for establishing trust among entities is a well-researched field [9, 20, 24, 27, 28]. The feedback of individuals regarding their interactions with others is an important criterion when calculating reputations.

There is, however, a problem in reputation-based trust, from an anonymity perspective. To accurately compute the reputation of an entity, individuals should provide feedback regarding their interactions with that entity, possibly, accompanied with transcripts. Alternatively, the individuals must permit the entity to provide the interactions' transcripts to the reputation system. This is problematic, since in both cases, it helps the system to profile the individuals. There is another problem that limits the practicality of current work on reputation management. The work fails to reward products and services that do not receive ratings, due to the low participation of the individuals using them. For example, the rating process may not by easy to use. In some cases, individuals may not even participate at all. Individuals may get discouraged from rating certain products for sensitivity, religious, and political reasons. Anonymous reputation management (ARM) systems [10, 24, 28] permit the anonymous rating of products and services; yet they fail to address the following.

1.1 The linkability problem in ARM

While ARM systems provide anonymity, there is a problem that remains unresolved. All ratings by an individual are linkable to each other, which may lead to profiling and re-identifying that individual. This is easily done by the servers managing the ratings, since ratings of an individual share the same pseudonym/identifier. Ratings that share a pseudonym form a profile of an anonymous individual. This profile may

M. Hussain (✉)
College of Computer and Information Technology,
Taif University, Taif, Saudi Arabia
e-mail: m.hussein@tu.edu.sa

D.B. Skillicorn
School of Computing, Queen's University, Kingston, Canada,
K7L 1T2
e-mail: skill@cs.queensu.ca

be linked to a real individual using another source of information, for example, the individual behavior at a public forum.

Data-mining techniques enable one to link partial identities together. For example, Narayanan [25] show that anonymous ratings in movie-rating datasets can be traced back to the actual individuals with the help of a very small amount of auxiliary information. Malin [23] has linked the genomic data of anonymized patients back to the actual patients. The re-identification algorithm presented by Narayanan [26] can link the accounts which belong to the same individual, at different social networking sites. Further, k-anonymity [30] suggests that the records of an individual must be indistinguishable from at least $k - 1$ records in a dataset, for that individual to be k-anonymous. These results suggest that anonymity is not enough. To protect an individual's privacy, the ratings submitted by that individual should be unlinkable to each other.

One may think of a system where an individual have a set of single-use identities that are replaced regularly. Such a system achieves unlinkability of an individual ratings. However, this system is impractical as each individual requires a new identity per transaction. We needed a system that allows an individual to have one identity, which can be used across different providers. The different showings of this identity are not linkable to each other. In this way, an individual may use and rate services, without service providers building a profile of that individual.

1.2 Contribution

In a previous work, we presented a secure protocol that enables individuals to interact with service providers, without enabling these providers to link different interactions to the same individual [16–18]. The protocol is based on empowering individuals to securely use web services, without enabling service providers to profile those individuals.

This paper extends our work to address the linkability problem. The contribution of this work is an anonymous reputation management system that satisfies the following properties.

– *Avoids the linkability problem*. The system prevents service providers, as well as the servers responsible for managing ratings, from profiling individuals.
– *Enhances the fairness of ARM*. Many individuals neglect to rate service providers after they interact with these providers. This causes the reputation system to be in favor of those providers who pressure their users to rate them. The presented system does not suffer from this issue. When individuals use services, the system generates proof of interaction. Service providers gain reputation by submitting the proofs of interaction to the system, which computes reputation values that reflect the fact that some individuals have used the services of these service providers. The system ensures that service providers cannot abuse this feature.
– *Minimizes the effects of Sybil attacks*. Service providers may specify constraints on the rate that an individual may rate a service, in a specific time interval. The system prevents both users and SPs from creating Sybil attacks. Though this does not fully address the issue of Sybil attacks, it minimizes these attacks.

The next section studies reputation-based trust, and explains why the related work on anonymous reputation management [10, 24, 28] fails to protect individuals' privacy. The underlying methodology of this paper is discussed in Sect. 3. Section 4 presents a reputation management system that mitigates the privacy problems present in the current work. The design and operations of the system are described in Sect. 5. Section 6 analyzes the system. The paper is concluded in Sect. 7.

2 Trust management

Grandison and Sloman [14] define the trust in an entity as the belief in the competence of that entity to perform a task in a dependable and reliable manner, in a specific context. Research on trust is usually categorized into policy-based [2, 19, 31] and reputation-based. In policy-based, an entity establishes trust in another by examining the credentials the former entity possesses. In reputation-based, the history of the interactions that an entity had with others is used to evaluate the entity's trustworthiness. One entity in a network may trust another entity based on the recommendation of other entities [1].

2.1 Reputation-based trust

There are many metrics for computing the reputation of an entity, for example, the EigenTrust algorithm [20] computes reputation scores of entities based on the PageRank [7] algorithm. In the web of trust [13] approach, each entity assigns reputation for its neighbors. To determine the level of trust an entity A should have in an entity B, A uses a trust metric which specifies how A should aggregate the reputation values on the paths to B.

Such reputation management systems do not allow individuals to participate in the system anonymously. This discourages individuals from participation, especially in the context of evaluating the reputation of services and products that are sensitive, religious, or political in nature. Anonymous reputation management (ARM) [10, 24, 28] tackles this problem by empowering individuals to participate anonymously.

In TrustMe [28], the reputation information of each peer in a peer-to-peer network is collected, stored, and updated by a randomly assigned set of peers, called the Trust-Holding Agent (THA) peers. If peer *i* wishes to submit a reputation value for peer *j*, peer *i* signs this value, encrypts it with the keys of the THA responsible for peer *j*, and broadcasts it over the network. The THA peers of peer *j* can read the value and update its reputation.

SuperTrust [10] is another ARM system. Just like TrustMe, SuperTrust assigns the management of the reputation of a peer to a set of super peers. A unique feature of SuperTrust is the use of a homomorphic encryption function. The function allows a super peer to aggregate the encrypted reputation values of its assigned peers, without decrypting the values. Thus, super peers do not learn the reputation values.

The work of Muler et al. [24] presents an ARM that provides anonymity for users, while protecting against Sybil attacks. In a Sybil attack [12], the attacker creates multiple accounts to be used to submit good/bad reputation values to gain some advantage.

3 Secure anonymous interactions with personas

A system that solves the linkability problem needs an underlying protocol that allows individuals to participate in unlinkable interactions. In this section, such a protocol is presented. The protocol is based on using *artificial identities*, called personas [17, 18, 29], that assure service providers that there are organizations guaranteeing the individuals.

3.1 Personas: definitions and features

A *persona* is a set of statements, where each statement asserts the status of an attribute of an individual. The statements of a persona can be self-issued, like preferences and tastes, or certified by an organization, like credit cards. A persona can be used to generate a set of unlinkable proofs of ownership, called *locked personas*. The following lists the properties of locked personas.

– A set of locked personas generated by a persona are not linkable to each other. Thus, the interactions of an individual cannot be profiled.
– A locked persona is a proof that an individual has interacted with an organization. Usually, the interaction is in the form of access request, from the individual's side, and access response, from the organization's side.
– A locked persona may encode the interaction details. In e-commerce settings, a locked persona may encode the purchased goods and time of purchase.
– A locked persona can be traced, in case of conflict resolution, to the persona which generated that locked persona. The tracing functionality is available only to trusted organizations responsible for law enforcement.

The properties help achieve privacy for individuals, while guarantee for organizations that they are protected against misuse. Note that personas are issued to individuals once, and are used by individuals at several service providers, on several occasions.

3.1.1 Encoding relations among individuals

Personas has the ability to encode relations among individuals, which allows access policies to be formulated based on relations [17]. Although existing identity management systems (IMS) [2, 8] may encode relations as extra attributes, this leads to a serious privacy violation. Consider the following scenario, in which a student and her supervisor want to prove, to a university web-service, the student/supervisor relation. Identity providers, in existing IMS, may provide the student and the supervisor with two certificates. Each certificate has an attribute stating part of the relation. But that creates the problem of preventing other students from claiming that they are supervised by that supervisor. The identity provider may assign the attributes as *supervisor_pseudonym* and *student_pseudonym*, where the pseudonym parts are equal. This, however, makes the actions of the student and supervisor linkable, since the pseudonyms remain fixed in all their actions. Personas encode relations without creating this form of linkability.

3.1.2 Symmetric interactions

There are many business-to-consumer (B2C) scenarios, in which businesses need to interact anonymously with consumers. For example, many hotel chains use arbitrageurs to sell their room surplus at a lower price, but without necessarily revealing their brand so as not to undercut their full-price sales. To prove the quality of the services, these hotels must prove their properties, for example, the star rating and the presence of amenities, such as swimming pools.

Since hotels do not reveal their brand names, they rely on arbitrageurs to prove hotels' properties to customers. Well-known arbitrageurs facilitating such transactions include Priceline, Hotwire, Travelocity, and Lastminute. Arbitrageurs present individuals with offers and their properties, and reveal the hotel identities only after transactions are complete. This leaves individuals, who wish to verify available offers, with no option other than to trust the arbitrageurs. Arbitrageurs charge the real suppliers not only for providing a service, but also for acting as guarantors of product attributes.

Personas generalize the notion of anonymity to include service providers. The goal is to make interactions between service providers and individuals to be symmetric, where each one proves to the other certain attributes, while both remaining anonymous.

3.1.3 Constrained interactions

Apart from individuals' identity attributes, service providers may need to enforce additional constrains before an individual may access a service. A provider may allow an individual to use a service n number of times per day. Personas permit the enforcement of such constraints using *tickets*. A ticket contains the number of requests an individual has made for a service, in a specific period of time. Such information helps organizations enforce constraints that control the rate by which individuals access services.

3.2 Illustration and assumptions

There are four entities that comprise the system. *Individuals* who use personas to protect their privacy. *Persona providers (PPs)* who provide individuals with personas and act as guarantors of these personas. *Service providers (SPs)* who offer services to individuals; *De-anonymization authorities (DAs)* who trace, with the help of PP, personas to their corresponding individuals.

Figure 1 shows an individual receiving a persona from a government agency (PP). The persona is then used to generate a set of locked personas on-demand that can be used at a phone store, a bank and an auction site (SPs). The locked persona used at the bank cannot be linked to the locked persona used at the phone store. Further, personas are collected once, and can be used on-demand.

3.2.1 Anonymous channel

We assume that individuals access service providers through anonymous channels, such as Tor [11]. This is needed since direct access lead to profiling through the IP addresses of individuals. This assumption is shared with other systems that enable individuals to have secure and private interactions, such as the systems presented by Hansen et al. [15] and Brands et al. [6].

3.2.2 Linking among personas

Linkability among personas of the same individual may happen if personas have a significant number of common attributes. This is discouraged. Yet, this does not mean that the individual need a lot of personas, since a small number of personas can produce a large set of possibilities. Five personas produce a space of 16 possibilities. Further, for small number of attributes, we may issue a persona per attribute. This solves the problem of linkability among personas.

3.3 Persona system architecture

Persona providers generate personas as follows. An individual contacts a PP and claim a set of attributes. The PP validates the individual claims. The process of validating the individual claim is highly dependent on several issues, for example, PP policies and the type of attributes claimed. Once the claims are validated by the PP, the PP encode the individual's claims. Then the encoded package is signed by the PP. Finally, the signed package represents the persona, which is sent back to the individual. Identity providers in our society are of a number of different types. Governments guarantee personal identities such as citizenship; schools and universities guarantee credentials such as degrees; supervisors guarantee character or performance by writing references; and financial institutions guarantee financial worth.

Individuals receive personas from PPs, and store these personas for later use at SPs. When an individual requests a service from the SP, the SP replies with a list of required attributes. The individual generates a locked persona from the personas that attest that the individual possesses the required attributes. The individual sends the locked personas to the SP. If the locked personas satisfy the SPs access policy for the requested service, the individual gets access. Figure 2 shows an individual receiving personas from the government

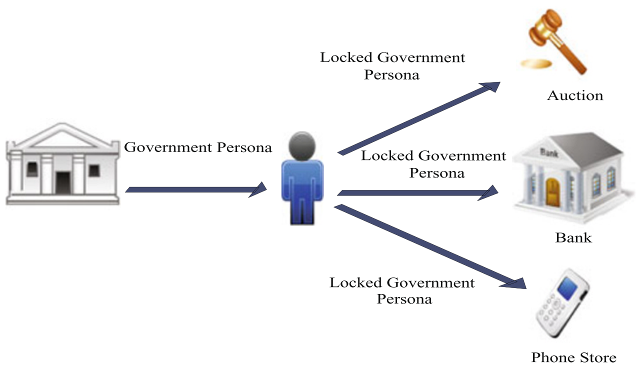
Government Persona
Locked Government Persona
Auction
Locked Government Persona
Bank
Locked Government Persona
Phone Store

Fig. 1 Generating a set of locked personas

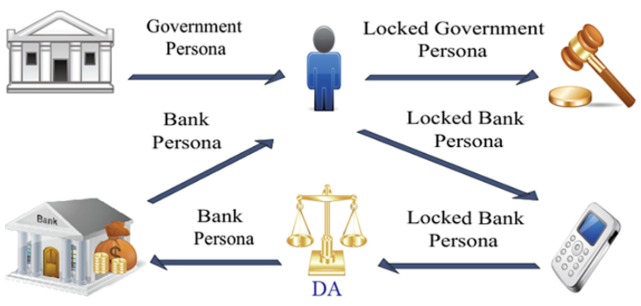

Fig. 2 The creation and usage of personas

and a bank. The individual then generates locked personas to be used at SPs. A locked government persona is used, at auction site, to prove that she is over 18. She pays her phone bills with a locked persona obtained from her bank.

Service providers can be any web service on the internet, or indeed any real-world service provider. Each SP keeps a list of PPs that it trusts. Individuals with personas from the trusted PPs are allowed to use the services at the SP. For example, Amazon accepts payments made using Visa and Master credit cards. To verify a locked persona of an individual, the SP determines which PP parameters to use. The SP verifies whether the locked personas are valid and the attributes satisfy the access policy of the requested service. This is done without contacting the PP.

De-anonymization authorities are invoked when there is a need to extract the persona used to generate a specific locked persona. SPs send locked personas to DAs, while DAs recover the personas. To prevent a DA from tracing a locked persona directly to the corresponding individual, a DA can only recover the persona. Only PPs can link a persona back to an individual. PPs and DAs can be implemented as separate components but might often be implemented within the same system. Figure 2 shows a DA, which receives a locked bank persona from a phone store. The DA extracts the bank persona from the locked persona, and sends it to the bank.

3.3.1 Enhanced persona distribution

Individuals request and receive personas from PPs. One issue that need to be resolved is how can individuals know which personas to request. A primitive approach is to let individuals contact SPs to be informed on which personas they need. This is limits the usability of personas. A better and more usable approach is that PPs suggest for individuals. For example, if an individual requested *persona A*, the PP may suggest, based on PP's experience, that the individual also applies for *persona B*. This is similar to how Amazon and eBay recommend their products to customers.

Individuals may supply PPs with the domain (leisure, education, government services, etc.) at which their personas will be used. PPs use this information to suggest relevant personas that individuals should request.

PPs may also provide online portals for persona distribution. For example, a university may create a web-service that allows their students to get personas to be used to authenticate at the online-library of that university.

4 Persona-based anonymous reputation management

A system that supports personas can be tweaked for reputation management. This section describes the usage of personas to construct an anonymous reputation management system.

4.1 The persona approach for reputation management

Recall that a locked persona is considered as a proof of a service request by an individual. Thus, an individuals' locked persona and an SP's signed response are a proof of interaction between that individual and that SP, in the form of requesting and providing a service. An individual can submit a locked persona, along with a reputation value measuring the satisfaction by the service, to a reputation management component. The component updates the reputation score of the SP's service and the SP itself.

Thanks to the unlinkability property of locked personas, individuals may submit all their locked personas to the reputation-management component, without the fear of being profiled. This has two significant implications. First, the reputation scores an individual submits are anonymous. Second, the reputation scores are unlinkable to each other. While current anonymous reputation management (ARM) systems achieve the first, they fail from achieving the second. Modeling reputation using personas are, therefore, more privacy-preserving than previous ARM systems.

The unlinkability of reputation scores may enable some individuals to launch Sybil attacks. That is, submitting large number of low or high reputation scores for an SP. The presented system prevents such attacks by disallowing an individual from submitting more than one reputation score for an SP, in a given time interval. This is achieved by the constrained-interactions property of personas. For each rating, a valid response from the SP is needed, restricting the ability of users to launch Sybil attacks. The system also prevents SPs from creating Sybil attacks. For each rating, a valid persona is needed, restricting the ability of the SP. Section 4.2 presents the approach in more detail.

Another advantage that our approach has is the following. An SP needs not to wait for individuals to submit their reputation scores for that SP. The SP may simply submit the locked personas it receives from individuals to the reputation-management component. The component can utilize the locked personas to give the SP partial reputation reflecting the fact that the SP is trustworthy enough to motivate individuals to request that SP's service. Recall that the well-known PageRank [7] algorithm works by counting the number of links to a page, rather than whether the links have good or bad connotation. EigenTrust [20] is a well-known reputation management system that utilizes PageRank.

Now we turn to the method of querying, submitting, and updating reputation scores of SPs. We introduce reputation-management components (RMC) to the persona system. These components respond to individuals when they query for an SP reputation, collect reputation scores from individuals, and update SPs reputation accordingly. Note that our approach focus on facilitating the functionalities of an ARM system, and thus the approach should work well with any reputation aggregation metrics. We also do not mandate a specific method for choosing RMC locations in the network, and the assignment of SPs to RMCs. Instead, this is left to administrators.

The operations provided by RMCs are the following.

- *Query for a reputation*. An individual queries RMCs for an SP's reputation.
- *Submit a reputation*. An individual submits a reputation message to an RMC.
- *Update a reputation*. An RMC updates the reputation score of an SP, based on a reputation algorithm, e.g., EigenTrust [20].

Let an individual D wishes to submit a reputation score for an SP to an RMC. D prepares a reputation message as

$$\mathit{Reputation} = \{\mathit{score}, \mathit{SP}, \mathit{locked\ persona}, \mathit{SP\ response}\} \quad (1)$$

where *SP* is the identity of the SP, and *SP response* is the signed message that D receives from *SP* as a response for a service request. D then generates a new locked persona to prove that D is the individual who used the service and reported the score. Note that D makes the new locked persona linkable to the original locked persona. D sends the tuple (reputation message and the new locked persona) to an RMC. We refer to the tuple as signed reputation. Finally, the RMC verifies that the locked persona in the reputation message and the locked persona generated based on the message are valid and linkable to each other, that is, generated by D.

Figure 3 shows an individual using a government persona to generate locked personas, to interact with an auction site and a phone store. There are two scenarios present in the figure. In the first, the individual submits a reputation for the auction site to the RMC. In the second, the phone store submits a locked persona to the RMC.

The second scenario is needed to allow SPs to gain reputation, even if their customers did not submit reputation scores for these SPs. The more locked personas are used at a service provider (SP), the more reputation that SP should have. The RMC assigns reputation values for the SP based on the rate of the transactions made by their individuals at that SP. Of course, the RMC can detect the case where both the individual and SP submit a reputation message and a locked persona for the same interaction. This is because the locked persona is equal in both cases.

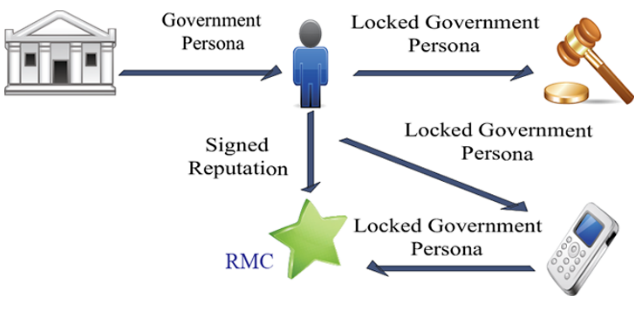
Government Persona
Locked Government Persona
Signed Reputation
Locked Government Persona
Locked Government Persona
RMC

Fig. 3 The persona-based reputation approach

Recall that locked personas encode the date of the interaction between individuals and SPs. This can be used by RMCs to check the frequency an SP's service or product is used by individuals. RMCs may use this frequency as a reputation metric, such that when the frequency consistently incline or decline, the reputation is also updated accordingly.

4.2 Minimizing Sybil attacks

To minimize Sybil attacks, RMCs do not accept reputation scores submitted by individuals unless they are accompanied with tickets generated by persona providers. Recall that a ticket contains information about the number of times an individual used a service, during a given time interval. An RMC can use this information to disallow an individual from submitting more than one reputation score for a specific service, in a given time interval.

If an individual wants to rate an SP, the RMC requests the individual to submit a reputation message, as described in Sect. 4.1. The individual also contacts her PP to receive a ticket to rate the SP. The PP does not know the SP that the individual wishes to rate. Then, the individual sends the ticket to an RMC. The RMC follows the steps of Appendix B.7 to verify the ticket. It should be noted that an individual may not submit the same ticket more than once, since RMCs check for ticket uniqueness. Similarly, SPs may not resend the same locked persona to RMC to gain more reputation.

Restricting ratings to be fixed per a given time interval minimizes Sybil attacks, but it does not solve it. Individuals may have multiple identities at the same PP, or at multiple PPs. Thus, some individuals may get multiple personas and use them to rate an SP more than their allowed quota. This is a weakness of this work, which constitutes an open problem for future work.

4.3 Preprocessing reputation messages

RMCs preprocess reputation scores before using them to update SPs' reputations. The preprocessing step is needed for practical issues, for example, avoiding malicious attacks and smoothing the effect of outliers. Such step exists in other reputation systems. For example, the trust values in EigenTrust [20] are normalized to the interval $[0, 1]$. This disallows individuals from providing arbitrary high or low scores. Another useful preprocessing step is truncation to get rid of outliers. That is, removing a fixed percentage of the reputation scores from both end of the spectrum. For example, one may truncate 5% of the scores from both sides.

4.4 Reputation for individuals

Personas may encode reputation for individuals. Assigning reputation values for individuals is important. It helps service providers to determine the trustworthiness of individuals; and it allows RMCs to give more weight for the reputation scores submitted by reputable individuals. Reputation should be assigned in a privacy-preserving manner. For example, a PP assigns a reputation level for an individual based on money spent, and time passed since registration.

A PP may encode reputation levels as attributes in personas. This is similar to information assurance in identity management systems, where an identity provider complements identity assertions with assurance values. These values represent the level of certainty that the identity provider has with respect to the assertions. Alternatively, the PP may use different sets of parameters to generate personas, whereas each set corresponds to a level of reputation. An individual generates a locked persona and claims a level of reputation. To verify the individual's locked persona, a service provider uses the PP's parameters corresponding to the reputation claimed.

5 System design and security

5.1 System operations through a sample scenario

This section presents the operations provided by the system. To facilitate the description, we use a sample scenario. In this scenario, Bob receives a persona from his university. Bob uses the persona to access the university library, and provide a rating.

Operations at PPs, University

$$\mathit{SetupAtPP} : \mathit{initializations} \rightarrow \mathit{PP\ pparam} \times \mathit{PP\ prkey}$$

PPs, like the university for Bob, use *SetupAtPP* to generate the public parameters (*PP pparam*) and their private keys.

$$\mathit{Wrap} : \mathit{attributes} \times \mathit{proof} \times \mathit{PP\ pparam} \times \mathit{PP\ prkey} \rightarrow \mathit{persona}$$

Wrap is executed by a PP to generate a persona for an individual. The PP receives a set of claimed attributes from the individual, along with the proof that the individual is entitled to the attributes. The proof may take the form of a locked persona or any other forms acceptable by the PP. The PP returns a persona to that individual. In Bob's scenario, the university executes wrap to generate a persona for Bob.

$$\mathit{Check\_Wrap} : \mathit{persona} \times \mathit{PP\ pparam} \rightarrow \mathit{boolean}$$

Check_Wrap is used by an individual or a PP to check if a persona is valid. Bob invokes *Check_Wrap* to check his university persona.

$$\mathit{Generate\_Ticket} : \mathit{tRequest} \times \mathit{locked\ persona} \times \mathit{PP\ pparam} \times \mathit{PP\ prkey} \rightarrow \mathit{ticket}$$

Generate_Ticket is executed by a PP to generate a ticket, to be used by an individual to rate an SP. In Bob's scenario, the university generates a ticket, based on Bob's ticket request. Bob needs a ticket whenever he wants to rate a service.

Operations by individuals, Bob

$$\mathit{Show} : \mathit{message} \times \mathit{persona} \times \mathit{PP\ pparam} \times \mathit{DA\ pparam} \rightarrow \mathit{locked\ persona}$$

Show is executed by an individual to generate a locked persona, proving the ownership of her persona. The individual then sends the locked persona to the SP. The show operation may also associate some meta-information with the locked persona, for example, an action, message, and timestamp. That is, we can treat *Show* as a signature on a message or an action. In Bob's scenario, Bob executes *Show* to generate a locked university persona.

$$\mathit{Selective\_Show} : \mathit{message} \times \mathit{persona} \times \mathit{PP\ pparam} \times \mathit{DA\ pparam} \rightarrow \mathit{locked\ persona}$$

An individual may use *Selective_Show* to show a subset of a persona to an SP. For example, the university may distribute Bob's attributes across several personas. Bob then can use each persona to prove a specific attribute.

$$\mathit{Submit\_Reputation} : \mathit{score} \times \mathit{locked\ persona} \times \mathit{SP} \times \mathit{SP\ response} \times \mathit{PP\ pparam} \times \mathit{DA\ pparam} \rightarrow \mathit{locked\ persona}$$

Submit_Reputation is used for sending reputation messages to RMCs. Note that reputation messages take the form of locked personas, so that RMC's verify them just like the SPs verify locked personas.

Operations at SPs, University Library

$$\mathit{Verify} : \mathit{message} \times \mathit{locked\ persona} \times \mathit{PP\ pparam} \times \mathit{DA\ pparam} \rightarrow \mathit{boolean}$$

A verifying entity *V*, receiving a locked persona, uses *Verify* to check the validity of the received locked persona. If *Verify* is passed successfully, the SP knows two facts. First, the individual is indeed certified by PP to use a persona. Second, the locked_persona is a proof that the individual has requested a service from the SP. In Bob's scenario, the university library uses *Verify* to check that the locked persona is valid with respect to the university.

$$\mathit{VerifyRelation} : \mathit{locked\ personas} \times \mathit{PP\ pparam} \times \mathit{DA\ pparam} \times \mathit{relations} \to \mathit{boolean}$$

A verifying entity V, receiving a set of locked personas, uses *VerifyRelation* to check the validity of not only the locked personas, but also the relations between them. If *VerifyRelation* is passed successfully, V knows three facts. First, the generating entities are indeed certified by a PP to use these personas. Second, the locked personas form a proof that these entities have participated in a transaction with V. Third, the claimed relations among these individuals are valid.

Operations at DAs

$$\mathit{SetupAtDA} : \mathit{initializations} \to \mathit{DA\ pparam} \times \mathit{DA\ prkey}$$

DAs, like the university in Bob's scenario, use *Setup At DA* to generate the public parameters *DA pparam* and their private keys.

$$\mathit{Trace} : \mathit{locked\ persona} \times \mathit{DA\ public\ parameters} \times \mathit{DA\ prkey} \to \mathit{persona}$$

Trace is used by a DA to trace a locked persona back to a persona. In Bob's scenario, the university uses *Trace* to trace Bob's locked personas back to him.

Operations at RMCs

$$\mathit{Update\_Reputation} : \mathit{reputation} \times \mathit{locked\ persona} \times \mathit{PP\ pparam} \to \mathit{score}$$

Update_Reputation is used by RMCs to update the reputation scores of SPs, based on a specific algorithm. The operation is invoked when a new reputation message arrives from an individual. The operation takes the old score of an SP and computes the new one.

$$\mathit{Verify\_Reputation} : \mathit{locked\ persona} \times \mathit{PP\ pparam} \times \mathit{DA\ pparam} \to \mathit{boolean}$$

Verify_Reputation is used by RMCs to test locked personas, which encode reputation scores, submitted by individuals to rate SPs.

$$\mathit{Verify\_Ticket} : \mathit{ticket} \times \mathit{tRequest} \to \mathit{boolean}$$

Verify_Ticket is used by RMCs to validate the tickets submitted by individuals to RMCs.

5.2 Cryptographic constructs

Personas can be supported by cryptographic systems that are capable of the following functionalities.

1. Unlinkability of interaction transcripts. When an individual uses a certificate repeatedly at SPs, possibly at the same SP, the SPs cannot link the different usages of that certificate. This functionality is needed to prevent SPs from profiling individuals.
2. Encoding and verifying relations. Two or more individuals with arbitrary relations may use their certificates at SPs, possibly at the same SP, to prove these relations.
3. Supporting constrained interactions. Service providers should be able to specify constraints on how often an individual may use a service, in a given time interval.

Idemix [8] achieves unlinkability of interaction transcripts; however, it needs to be modified to accommodate for relationship verification and for constrained interactions.

To implement personas, we begin with the hidden ID-based signature scheme [21], and extend it to support personas and the functionalities described above.

5.2.1 Hidden ID-based signatures (HIDS)

HIDS is an identity-based signature scheme from pairing. The scheme has the following property: signed messages are verifiable without the public key (identity) of the signer. Only the public key of the identity provider is needed. The scheme splits the role of the identity provider into: an identity provider and a de-anonymizing authority. The identity provider issues certificates to individuals, while the de-anonymizing authority may open the signatures generated from these certificates. Opening a signature refers to the process of extracting the public key of the signer. The scheme provides these four operations:

`Setup`. Initializes the public/private key pair of both Identity Provider (IDP) and De-anonymizing Authority (DA).
`Register`. The IDP registers an individual by issuing a certificate, which is a signature on that individual identity, produced by the IDP private key.
`Check Reg`. The individual checks whether the identity and certificate pair are valid with respect to each other.
`Sign`. Signatures are generated as follows. The individual commits the identity and the certificate. Then a Σ-protocol is used to prove the knowledge of the value of the committed identity and certificate, and that the certificate is a signature on that identity. The output of Σ-protocol is hashed along with the message to be signed. The committed identity and certificate, the protocol output, the produced hash, and the message comprise the signature.
`Verify`. The verifier uses the IDP public key to check the

signature is valid and that the signer certificate and identity are encrypted with the DA public key.
Open. The DA uses its private key to decrypt and extract the signer's committed identity from a valid signature.

The scheme provides the basic cryptographic support for generating, verifying, and tracing signatures. We extend the scheme to provide the needed persona features, such as showing attributes, proving relations, and enabling constrained interactions. An anonymous reputation-management system can be constructed once the features are implemented. Appendix A studies the HIDS operations, while Appendix B presents our extension of HIDS. The correctness and security of the extension is provided in Appendix C.

6 Performance analysis

We have implemented and tested the cryptographic constructs with the help of the Pairing-based Cryptography (PBC) library [22]. The machine used to test the operations is a 2.6 GHz Pentium 4 machine. The PBC is a free library written in C and it provides the necessary functions to write programs that handle elliptic curve generation, elliptic curve arithmetic, and pairing computation.

There are four factors that the analysis is based on: features, messaging cost, management of cryptographic credentials, and response time. The following addresses each factor in detail and compares it to two Anonymous Reputation Management (ARM) systems, TrustMe [28] and SuperTrust [10].

6.1 Features

Current ARM systems offer anonymity and prevent an individual from submitting more than one reputation message per interaction. However, none of the current systems offer unlinkability of reputation messages. Our system allows individuals to submit unlinkable reputation messages. Even reputation management components (RMCs) cannot link the reputation messages of one individual to each other or to that individual. It also prevents an individual from submitting more than one reputation message per one interaction. Our system, therefore, has an advantage in terms of features.

6.2 Messaging cost

In a centralized approach, individuals directly communicate with RMCs to receive and submit reputation values. TrustMe and SuperTrust use a decentralized approach for managing reputation values. A peer broadcasts queries for the reputation of other peers; then it broadcasts a feedback after interacting with those peers. The messaging cost grows with the number of peers and it is greater than the cost in a centralized approach. The overhead in TrustMe is 10^8 messages per 18,000 transactions, while in SuperTrust it is 10^6 messages per 18,000 transactions, as shown in [10, 28]. The smaller overhead in SuperTrust is due to the Super-Peer model, which organizes a peer-to-peer network into clusters managed by special entities, called Super Peers.

The presented ARM in this paper neither mandates a specific protocol for communicating with RMCs, nor it requires a specific arrangement for RMCs. The ARM can be implemented using any approach. Thus, the messaging cost is similar to current ARM systems.

6.3 Management of cryptographic credentials

ARM systems generate new public key credentials for individuals to be used for submitting their feedback. Individuals have two sets of credentials, one to prove their attributes to access services and interact with other individuals, and one for reputation management. In the presented ARM, individuals do not need new credentials for reputation management. Individuals use their credentials to interact with other individuals, as well as to submit their feedback. This represents one advantage of our system over other systems.

6.4 Response time

Generating a reputation message takes 200 milliseconds, while verifying that reputation message takes 240 ms. The total is 440 ms. Other ARM systems are based on public cryptography like RSA. In these systems, generating and verifying a reputation message require approximately 80 ms. The reported times are based on the same machine used to test our system. The Unlinkability of reputation messages, however, justifies the increase in response time.

Table 1 summarizes the above analysis. The table compares our work with two ARM systems, TrustMe and SuperTrust.

7 Conclusion

Trust is an important factor in the decision making process. One way to quantify trust is through the calculation of reputation scores, based on some reputation metrics. Reputation management systems facilitate such calculations, but do not offer anonymity for individuals. Current work on anonymous reputation management (ARM) allows for anonymity, but they fail to provide unlinkability of the anonymous ratings. Linking anonymous ratings of an individual to each other allows the system to build a profile of the individual, which may result in re-identification of that individual by data-mining techniques.

Table 1 Comparison between the presented work and two ARM systems

Factors	Personas	TrustMe [28]	SuperTrust [10]
Anonymity	✓	✓	✓
Duplicate prevention	✓	✓	✓
Unlinkability	✓		
Messaging cost	Depends on the model used	10^8 per 18,000 transactions	10^6 per 18,000 transactions
Additional credentials		✓	✓
Response time	440 ms	Around 80 ms	Around 80 ms

In this paper, an ARM system for web services is presented. The system is based on a protocol for secure and anonymous interactions. The transcripts of the interactions an individual has with a service provider are unlinkable. These transcripts can be used to enable individuals to submit feedback, without the fear of being profiled by the system. The presented ARM allow services to gain reputation, even if their customers neglected rating them. The system prevents Sybil attacks from degrading the quality of the reputation scores.

Moreover, the current work on privacy does not address trust management. Addressing privacy and trust by a single framework is more efficient, since the two subjects are related. In the Semantic Web [3] and the Semantic Social Web, trust is an integral part for automatic service discovery and invocation. Therefore, such a framework has a profound application in the Semantic Web setting.

Appendix A: The HIDS operations

`Setup`: $\{\textit{public parameters}, \textit{IDP keys}, \textit{DA keys}\} \leftarrow \textit{Setup}$.

Setup is used to generate the required public-private keys for the IDP and DA. The keys are generated based on Boneh and Boyen [4] signature scheme. Setup generates $(p, g, \mathbb{G}, \mathbb{G}_2, e)$, where $\mathbb{G}$ is a cyclic group of prime order p, with a generater g, and e is a bilinear map, $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_2$. IDP public key is the pair $(X = g^x, Y = g^y)$, where x and y are random elements in $\mathbb{Z}_p$. IDP private key is the pair (x, y). DA public key is given by (u, v, w), where w is a random element in $\mathbb{G}$ and $w = u^b = v^d$, b and d are random elements in $\mathbb{Z}_p$. DA private key is (b, d). The public parameters of the system are $(p, g, \mathbb{G}, \mathbb{G}_2, e, X, Y, u, v, w, h, H)$, where h is a random element in $\mathbb{G}$, and H is a hash function.

`Register`: $\textit{certificate} \leftarrow \textit{Register}(\textit{identifier})$

When an individual requests a certificate from the IDP, the IDP encodes that individual's identity as an identifier, say I. The mapping between the identifiers and the real identity of the individual is securely stored at the IDP. The IDP issues a certificate C to that individual by signing I with its private key. Note that r is a random element in $\mathbb{Z}_p$.

$$C = \{s = g^{x+I+y\,r^{-1}}, r\} \tag{2}$$

`Check Reg`: $\textit{boolean} \leftarrow \textit{Check Reg}(\textit{identifier}, \textit{certificate})$

An individual may check the validity of her certificate by checking if the following condition holds: $e(s, Xg^I Y^r) = e(g, g)$.

`Sign`: $\textit{signature} \leftarrow \textit{Sign}(\textit{identifier}, \textit{certificate}, \textit{message})$

This operation is used to generate signatures on messages. Signing a message M with a certificate C requires these steps:

– The individual uses the encryption scheme of [5] to commit I in (U, V, W).

$$U = u^l, \qquad V = v^k, \qquad W = w^{l+k} g^I \tag{3}$$

where l and k are random numbers in $\mathbb{Z}_p$.
– The individual commits C in (S, R), where $S = g^{r_1} s$, $R = g^{r_2} h^{r_1} Y^r$, and r_1 and r_2 are random elements in $\mathbb{Z}_p$.
– The individual uses a Σ-protocol (described below) to prove the knowledge of the committed values of C and I, and that C is a valid signature on I.
– The variables of the Σ-protocol are hashed along with the message to be signed.
– The committed values, hash, and M represents a HIDS signature.

The Σ-protocol

The individual uses the protocol to prove the knowledge of C and I committed in S and R and that C is a valid signature on I. Compute $\alpha_1 = r_1 k$, $\alpha_2 = r_1 l$, $\alpha_3 = r_1 r_2$, $\alpha_4 = r_1^2$, $\alpha_5 = r_1 r$. Choose η_I, η_k, η_l, η_r, η_{r_1}, η_{r_2}, η_{α_1}, η_{α_2}, η_{α_3}, η_{α_4}, and η_{α_5} randomly from $\mathbb{Z}_p$.

$$B_1 = u^{-\eta_k}, \qquad B_2 = v^{-\eta_l}, \qquad B_3 = w^{-(\eta_k+\eta_l)} g^{-\eta_I},$$
$$B_4 = g^{-\eta_{r_2}} h^{-\eta_{r_1}} Y^{-\eta_r}, \qquad B_5 = U^{-\eta_{r_1}} u^{\eta_{\alpha_1}},$$

$$\begin{aligned}
&B_6 = V^{-\eta_{r_1}} v^{\eta_{\alpha_2}}, \qquad B_7 = R^{-\eta_{r_1}} g^{\eta_{\alpha_3}} h^{\eta_{\alpha_4}} Y^{\eta_{\alpha_5}},\\
&B_8 = e(g, X\,W\,R)^{\eta_{r_1}} e(S, w)^{\eta_k+\eta_l} e(g, w)^{-(\eta_{\alpha_1}+\eta_{\alpha_2})}\\
&\qquad \times e(S, g)^{\eta_{r_2}} e(g, g)^{-\eta_{\alpha_3}} e(S, h)^{\eta_{r_1}} e(g, h)^{-\eta_{\alpha_4}}\\
&c = H(M \parallel S \parallel R \parallel U \parallel V \parallel W \parallel B_1 \parallel \cdots \parallel B_8),\\
&\lambda_I = \eta_I + cI, \qquad \lambda_r = \eta_r + cr,\\
&\lambda_{r_1} = \eta_{r_1} + cr_1, \qquad \lambda_{r_2} = \eta_{r_2} + cr_2,\\
&\lambda_k = \eta_k + ck, \qquad \lambda_l = \eta_l + cl,\\
&\lambda_{\alpha_1} = \eta_{\alpha_1} + c\alpha_1, \qquad \lambda_{\alpha_2} = \eta_{\alpha_2} + c\alpha_2,\\
&\lambda_{\alpha_3} = \eta_{\alpha_3} + c\alpha_3, \qquad \lambda_{\alpha_4} = \eta_{\alpha_4} + c\alpha_4,\\
&\lambda_{\alpha_5} = \eta_{\alpha_5} + c\alpha_5
\end{aligned} \tag{4}$$

The tuple $\sigma = \{S, R, U, V, W, c, \lambda_r, \lambda_{r_1}, \lambda_{r_2}, \lambda_k, \lambda_l, \lambda_I, \lambda_{\alpha_1}, \lambda_{\alpha_2}, \lambda_{\alpha_3}, \lambda_{\alpha_4}, \lambda_{\alpha_5}\}$ is the HIDS signature on M.

`Verify`: *boolean* ← *Verify*(*signature*, *message*)

The operation checks whether a received (signature, message) pair represents a valid HIDS signature. The verifier uses the following condition to check that σ is a valid signature on M.

$$\begin{aligned}
c = H\big(&M \parallel S \parallel R \parallel U \parallel V \parallel W \parallel U^c u^{-\lambda_k} \parallel V^c v^{-\lambda_l}\\
&\times \parallel W^c w^{-(\lambda_k+\lambda_l)} g^{\lambda_I} \parallel\\
&\times R^c g^{-\lambda_{r_2}} h^{-\lambda_{r_1}} Y^{-\lambda_r} \parallel U^{-\lambda_{r_1}} u^{\lambda_{\alpha_1}} \parallel V^{-\lambda_{r_1}} v^{\lambda_{\alpha_2}}\\
&\times \parallel R^{-\lambda_{r_1}} g^{\lambda_{\alpha_3}} h^{\lambda_{\alpha_4}} Y^{\lambda_{\alpha_5}} \parallel\\
&\times e(g, XWR)^{\lambda_{r_1}} e(S, w)^{(\lambda_k+\lambda_l)} e(g, w)^{-(\lambda_{\alpha_1}+\lambda_{\alpha_2})}\\
&\times e(S, g)^{\lambda_{r_2}} e(g, g)^{-\lambda_{\alpha_3}}\\
&\times e(S, h)^{\lambda_{r_1}} e(g, h)^{-\lambda_{\alpha_4}} \big(e(g, g)/e(S, XWR)\big)^c\big)
\end{aligned} \tag{5}$$

`Open`: *identifier* ← *Open*(*signature*)

The private key of the DA (b, d) is used to extract g^I from the commitment (U, V, W) and send it to the IDP ($g^I = U^{-b} V^{-d} W$). The IDP maps the identifier back to the real identity by looking up the values of g^I from a table.

Appendix B: The extended HIDS scheme

The HIDS scheme cannot be used as is to implement the needed system operations. The signer can prove the possession of a certificate from the IDP, but nothing beyond that. The scheme, therefore, needs modification to allow individuals to show attributes, prove relations, or rate service providers. We extend the scheme by modifying each operation. The modified system is called the extended HIDS for abbreviation.

B.1 Extended HIDS operations

A subscript e is appended to the names of the extended HIDS operations to distinguish them from the HIDS ones.

$\texttt{Setup}_e$: {*public parameters*, *PP keys*, *DA keys*} ← Setup_e.

Setup is not changed. The PP plays the role of the IDP.

$\texttt{Register}_e$: {*persona*, *secret*} ← $\mathit{Register}_e$(*identifier*)

The PP uses HIDS's Register operation to issue certificates to individuals. When an individual requests a certificate, the PP generates two identifiers (I_{base}, and I_{full}). The difference to the HIDS identifiers is that these identifiers are composed of two parts: a *pseudonym* part, which is a random number to distinguish between individuals, and an *attributes* part, which encodes the attributes of the individual. The pseudonym part of I_{base} and I_{full} are equal. The attributes part of I_{base} is assigned to 0, i.e., no attributes, while the attributes part of I_{full} is assigned to the encoding of the attributes that the individual is entitled to. For simplicity, assume that i bits of the identifier encodes the pseudonym, while the remaining j bits encodes the attributes.

The PP invokes the HIDS's Register operation twice, once per identifier. The PP sends the two identifiers and the certificate on them to the individual. The identifiers (I_{base}, I_{full}) constitute the persona, while the certificates (C_{base}, C_{full}) are the secret.

$$C_{\text{base}} \leftarrow \mathit{Register}(I_{\text{base}}), \qquad C_{\text{full}} \leftarrow \mathit{Register}(I_{\text{full}}) \tag{6}$$

$$\mathit{persona} = \{I_{\text{base}}, I_{\text{full}}\}, \qquad \mathit{secret} = \{C_{\text{base}}, C_{\text{full}}\} \tag{7}$$

$\texttt{Check Reg}_e$: *boolean* ← *Check Reg*(*persona*, *secret*)

An individual may check the validity of her (persona, secret) pair by checking if the following condition holds:

$$\begin{aligned}
\mathit{true} &= \mathit{Check\,Reg}(I_{\text{base}}, C_{\text{base}})\ \&\&\ \mathit{true}\\
&= \mathit{Check\,Reg}(I_{\text{full}}, C_{\text{full}})
\end{aligned} \tag{8}$$

$\texttt{Sign}_e$: *locked persona* ← Sign_e(*persona*, *secret*, *message*, *attributes*)

This operation is used to generate locked personas. This is achieved by generating HIDS signatures on messages. Signatures are the implementation of locked personas, where each (signature, message) pair represents a locked persona. $\texttt{Sign}_e$ can be used to sign messages, with or without showing attributes. For example, an individual affiliated with a university authenticates to the ACM Digital

Library, which requires nothing more than that the individual is affiliated with that university. In this case, the individual uses the I_{base} to sign a message M, which produces σ_{base} (9). The pair $(M, \sigma_{\text{base}})$ is a locked persona.

$$\sigma_{\text{base}} \leftarrow \mathit{Sign}(M, I_{\text{base}}, C_{\text{base}}) \tag{9}$$

The individual sends the signature and the message to the verifying entity.

If the individual needs to show attributes to the verifying entity, both identifiers are needed. $\texttt{Sign}_e$ invokes two instances of the HIDS's Sign operation, once per each (identifier, certificate) pair. Locked personas generated by this type of $\texttt{Sign}_e$ contains attributes as well (11). A locked persona is the tuple (signature, message, attribute).

$$\begin{aligned}
&\sigma_{\text{base}} \leftarrow \mathit{Sign}(M, I_{\text{base}}, C_{\text{base}}),\\
&\sigma_{\text{full}} \leftarrow \mathit{Sign}(M, I_{\text{full}}, C_{\text{full}})\\
&\sigma_{\text{base}} = \{S_{\text{base}}, R_{\text{base}}, U_{\text{base}}, V_{\text{base}}, W_{\text{base}}, \ldots\},\\
&\sigma_{\text{full}} = \{S_{\text{full}}, R_{\text{full}}, U_{\text{full}}, V_{\text{full}}, W_{\text{full}}, \ldots\}
\end{aligned} \tag{10}$$

Note that the same message M is used in both instances. Further, both instances of Sign should use the same values for the random variables l and k in (3). The individual sends the signatures, the message, and the attributes to the verifying entity.

$$\texttt{Verify}_e: \quad \mathit{boolean} \leftarrow \mathit{Verify}_e(\mathit{locked\ persona}, \mathit{message}, \mathit{attribute})$$

The $\texttt{Verify}_e$ operation has two flavors: one to deal with signatures generated by (9), and another to deal with signatures of (11). If (9) is used to generate a signature, then the HIDS's Verify operation is supplied with the (message, signature) pair. If Verify returns true, then σ_{base} is a valid signature on M.

$$\mathit{true} = \mathit{Verify}(M, \sigma_{\text{base}}) \tag{11}$$

If (11) is used to generate signatures, the verification proceeds as follows (this is the case where an individual needs to prove attributes to the verifying entity). First, the HIDS's Verify operation is invoked twice to check the validity of both signatures (12). If (12) holds, then both signatures are valid.

$$\mathit{true} = \mathit{Verify}(M, \sigma_{\text{base}})\ \&\&\ \mathit{true} = \mathit{Verify}(M, \sigma_{\text{full}}) \tag{12}$$

Second, the attributes that the individual is claiming are checked. Recall that the difference between I_{base} and I_{full} is that I_{full} encodes the individual's attributes, while I_{base} does not. That is, $\mathit{attributes} = I_{\text{full}} - I_{\text{base}}$. Recall also that σ_{base} contains the commitment of I_{base}, while σ_{full} contains the commitment of I_{full} (13).

$$W_{\text{base}} = w^{l+k} g^{I_{\text{base}}}, \qquad W_{\text{full}} = w^{l+k} g^{I_{\text{full}}} \tag{13}$$

To check whether the attributes that the individual is claiming are the same attributes encoded in the identifiers supplied to her by the PP, the verifying entity checks whether (14) holds:

$$W_{\text{base}} g^{\text{attributes}} = W_{\text{full}} \tag{14}$$

$$\texttt{Open}: \quad \mathit{persona} \leftarrow \mathit{Open}(\mathit{locked\ persona})$$

The HIDS Open operation is used to extract $g^{I_{\text{base}}}$ and $g^{I_{\text{full}}}$ from the commitments (15). The identifiers are then sent to the PP. The PP maps the identifiers back to the real identity by looking up the values of the identifiers from a table.

$$I_{\text{base}} = \mathit{Open}(\sigma_{\text{base}}), \qquad I_{\text{full}} = \mathit{Open}(\sigma_{\text{full}}) \tag{15}$$

B.2 Linkable signatures

The sign operation of the extended HIDS generates two HIDS signatures, σ_{base} and σ_{full}, that are linkable to each other. The signatures serve two purposes: proving that the individual has a persona from a PP, and proving that the individual is entitled to the attributes inferred from the two signatures. Each time $\texttt{Sign}_e$ is invoked, it generates a new pair of two signatures that are linkable to each other, but are unlinkable to other pairs. In some cases, however, an individual may need to produce a pair of signatures that is linkable to a previous one. To generate a new pair that is linkable to a previous pair, the individual must use the same message and the same values for the random variables l and k, when generating the new one.

B.3 Selective release of attributes

Instead of having two identifiers: I_{base} and I_{full}, the PP can provide an individual with many identifiers. Selective release of attributes is achieved by providing an individual with an identifier per attribute or a set of attributes. Let I_{age} be of the same bit-length as I_{full}, and the *pseudonym* bits be equal. However, all the *attributes* bits are 0s except for the bits encoding the age. An individual with I_{age} certified by PP can use the Sign operation of the extended HIDS to show the age only as follows:

$$\begin{aligned}
&\sigma_{\text{base}} \leftarrow \mathit{Sign}(M, I_{\text{base}}, C_{\text{base}}),\\
&\sigma_{\text{age}} \leftarrow \mathit{Sign}(M, I_{\text{age}}, C_{\text{age}})\\
&\sigma_{\text{base}} = \{S_{\text{base}}, R_{\text{base}}, U_{\text{base}}, V_{\text{base}}, W_{\text{base}}, \ldots\},\\
&\sigma_{\text{age}} = \{S_{\text{age}}, R_{\text{age}}, U_{\text{age}}, V_{\text{age}}, W_{\text{age}}, \ldots\}
\end{aligned} \tag{16}$$

Recall that $W_{\text{age}} = w^{l+k}\, g^{I_{\text{age}}}$. If (17) holds, then the individual is certified by the PP to have that *age* attribute.

$$\begin{aligned} \mathit{true} &= \mathit{Verify}(M, \sigma_{\text{base}})\ \&\&\ \mathit{true} \\ &= \mathit{Verify}(M, \sigma_{\text{age}})\ \&\&\ W_{\text{base}}\, g^{I_{\text{age}}} = W_{\text{age}} \end{aligned} \tag{17}$$

B.4 Encoding and verifying relations

Similar to the way personas prove attributes, personas may prove the existence of relations among individuals. An identifier is composed of a pseudonym part and an attribute part. To encode relations, a third part is added, called a *relation*. An identifier becomes the composition of a pseudonym part, a relation part, and an attribute part.

$$\mathit{identifier} = \mathit{pseudonym} \parallel \mathit{relation} \parallel \mathit{attributes}$$

Let I_1 and I_2 be two base identifiers issued by a PP for individuals D_1 and D_2, respectively. Let the *pseudonym* bits of both identifiers be equal, and there be a relation between D_1 and D_2, for example, D_1 is the boss of D_2. PP encodes that relation by giving D_1 and D_2 different values for the *relation* bits as follows. The relation bits of D_1 and D_2 are set to re_1 and re_2, respectively, such that $\mathit{relation} = re_2 - re_1$. Note that the *attribute* bits are assigned to 0 in both identifiers. The identifiers for D_1 and D_2 become

$$\begin{aligned} I_1 &= \mathit{pseudonym} \parallel re_1 \parallel 0 \\ I_2 &= \mathit{pseudonym} \parallel re_2 \parallel 0 \end{aligned} \tag{18}$$

The PP then register the identifiers, as in (7), to generate a persona and a secret pair for each individual, (P_1, S_1) and (P_2, S_2).

$$\{P_1, S_1\} \leftarrow \mathit{Register}_e(I_1), \qquad \{P_2, S_2\} \leftarrow \mathit{Register}_e(I_2), \tag{19}$$

When D_1 and D_2 want to prove their relation to a verifying entity V, both use $\texttt{Sign}_e$ to sign a message and produces two locked personas (LP_1 and LP_2). Then they send the locked personas and *relation* to V. Note that *relation* is appended-to bits of 0s of length a (the attribute bit size).

$$\begin{aligned} LP_1 &\leftarrow \mathit{Sign}_e(M, P_1, S_1), \\ LP_2 &\leftarrow \mathit{Sign}_e(M, P_2, S_2) \\ LP_1 &= \{\sigma_{D_1} = \{S_1, R_1, U_1, V_1, W_1, \ldots\}, M\}, \\ LP_2 &= \{\sigma_{D_2} = \{S_2, R_2, U_2, V_2, W_2, \ldots\}, M\} \end{aligned} \tag{20}$$

The verifying entity uses `VerifyRelation` to verify the relation. The two locked personas must be linkable, Appendix B.2, for the verification to be possible.

$$\texttt{VerifyRelation}_e : \mathit{boolean} \leftarrow \mathit{VerifyRelation}_e(\mathit{locked\ persona}, \mathit{locked\ persona}, \mathit{relation})$$

This operation takes two locked personas, which represent signatures on a message, and a relation. The operation verifies each locked persona alone as in the $\texttt{Verify}_e$ operation; then (21) is used to verify the relation. W_1 and W_2 are computed as W is computed for the case of one individual; see (3).

$$W_1 g^{\text{relation}} = W_2 \tag{21}$$

In the same manner, we can specify relations that involve several individuals. In other words, personas can model graphs, where the nodes are individuals and the edges are their relations. We achieve this by computing an adjacency matrix for the required graph. Each cell encodes the relation between two individuals: the individual corresponding to the column of the cell, and the individual corresponding to the row. Thus, each row encodes the set of relations between an individual and the remaining individuals.

Now, we explain in details how a PP provides a set of individuals D with a set of personas P and the corresponding secrets S, allowing them to prove a set of relations R. Let D_i denotes the ith individual, R_i denotes the set of relations of the ith individual, R_i^j denotes the cell at row i and column j, and S_i denotes the ith persona.

Algorithm 1 takes R as input and produces P and S as output. From R we compute $\acute{R}$, which combines the set R_i into a single value. The *relation* part of the identifier of P_i is set to $\acute{R}_i$. Then all P_i and S_i are generated to get P and S.

```
Input: $R$
Output: $S$
$\acute{R}_1 = 0$
foreach $\acute{R}_i$ in $\acute{R}$, $i \neq 1$ do
    $temp = \sqrt{(R_i^1)^2 + (R_i^2)^2 + \cdots + (R_i^n)^2}$
    $\acute{R}_i = temp + \acute{R}_{i-1}$
end
$pseudonym = random$
$attribute = 0$
foreach $P_i$ in $P$ do
    $relation = \acute{R}_i$
    $identifier = pseudonym \parallel relation \parallel attribute$
    $secret = Register_e(identifier)$
    $P_i = identifier$, $S_i = secret$
end
```
Algorithm 1: Generating personas based on an adjacency matrix

```
Input: $R$, $LP$
Output: $accept$, $reject$
foreach $LP_i$ in $LP$ do
    $relation = \sqrt{(R_i^1)^2 + (R_i^2)^2 + \cdots + (R_i^n)^2}$
    if $VerifyRelation_e(LP_{i+1}, LP_i, relation) = reject$
    then
        output reject
        hault
    end
end
output accept
```

Algorithm 2: Verifying locked personas against an adjacency matrix

Note that the *attribute* part is 0 for all P_i, and that all P_i have the same value for *pseudonym*. The $\|$ symbol refers to concatenation. The algorithm above encodes the relations of D_i, $1 \le i \le n-1$. The relations of D_n can be inferred from other relations (undirected graphs). For directed graphs, a new persona P_{n+1} is needed to compensate. P_{i+1} is computed as other P_i in the algorithm. The PP finally sends P and S to D, where each D_i receives a pair (P_i, S_i). In case of directed graphs, D_n receives two pairs (P_n, S_n) and (P_{n+1}, S_{n+1}).

Now we turn to how the individuals prove R to an entity V, using P and S. Each D_i signs the same message M using P_i and S_i and send the resulted locked persona LP_i to V, $LP_i \leftarrow Sign_e(M, P_i, S_i)$ Let LP denotes the set of the locked personas. The individuals also send R to V. The entity V runs Algorithm 2. It is clear from Algorithm 1 that the relations of the ith individual can be recovered from the $\acute{R}_i$ and $\acute{R}_{i+1}$. The algorithm uses LP_{i+1}, LP_i, and A_i as input to $VerifyRelation_e$. If any instance of $VerifyRelation_e$ does not pass, the algorithm outputs reject. Otherwise, it outputs accept.

B.5 Reputation management

Reputation management requires five operations: submit reputation, verify reputation, update reputation, generate ticket, and verify ticket. Individuals submit reputation scores, along with tickets generated by PPs, whereas RMCs update reputation scores, after verifying the tickets. Tickets generation and verification are described in Appendix B.6.

`SubmitReputation`:

$$locked\,persona \leftarrow SubmitReputation(peronsa, secret, reputation\,message)$$

Let an individual D wish to rate an SP. The operation takes from D a reputation score sc and a proof of interaction with SP. The proof consists of a locked persona lp from D's side, and an SP's signed response sr from SP's side. A reputation message m is constructed as $m = \{sc, SP, lp, sr\}$. `SubmitReputation` then executes $\texttt{Sign}_e(m)$ to generate a locked persona $\bar{lp}$, where $\bar{lp} = \{\sigma, m\}$ Further, $\bar{lp}$ and lp must be linkable. The individual sends L to the RMC as a reputation score for SP, by D. The individual should also submit a ticket to the RMC, see Appendix B.6.

`VerifyReputation`:

$$boolean \leftarrow VerifyReputation(locked\,persona)$$

`VerifyReputation` receives a locked persona $\bar{lp}$, which includes a reputation message m. The RMC extracts SP, sr, lp, sc from m and uses $\texttt{Verify}_e$ to check whether lp and $\bar{lp}$ are valid locked personas and linkable to each other. The RMC also checks whether sr is a valid SP response. If all tests passed, the RMC executes `UpdateReputation`.

`UpdateReputation`. The RMC updates the reputation score of SP, based on sc. Choosing which score aggregation algorithm to use and updating SP's reputation is left for system administrators.

B.6 Ticket management

The following describes the generation and verification of tickets.

`GenerateTicket`:

$$ticket \leftarrow GenerateTicket(ticket\,request)$$

An individual contacts her PP to generate a ticket to be used at an SP. The individual prepares a ticket request m_I and sends it to her PP.

$$m_I = \{h_1, h_2\}, \qquad h_1 = \tilde{\mathcal{H}}(lPersona), \qquad h_2 = \tilde{\mathcal{H}}(i, PP, SP) \tag{22}$$

where $\tilde{\mathcal{H}}$ is collision-resistant hash function, i is the time interval from the SP perspective, and $lPersona$ is the locked persona used to interact with SP. Since h_1 and h_2 are hash values, the PP cannot determine SP's identity. That is, the PP does not know for which SP the ticket is generated.

The PP records the total number of times h_2 has been submitted by the same individual, in the current interval i_{PP}, which may or may not be the same as the interval i. The total n is incremented and appended to m_I to get m_{PP}. The PP generates a ticket t by signing m_{PP} with PP's key PP_key (any public cryptography algorithm, for example, RSA, should work fine).

$$t = \{m_{PP}, s\}, \qquad m_{PP} = \{h_1, h_2, n, i_{PP}\},$$
$$s = \mathcal{S}_{PP\_key}(m_{PP}) \tag{23}$$

where $\mathcal{S}$ generates signatures based on the key *PP_key*. The PP sends t to the individual as a ticket.

`VerifyTicket:` $\quad boolean \leftarrow VerifyTicket(ticket)$

The individual forwards h_1, h_2, i, *lPersona*, *SP*, *PP*, and t to the SP. The SP verifies and evaluates t against the constraint attached to the service requested by the individual. If the following equality holds, the SP is ensured that t is indeed generated by PP.

$$\begin{aligned} h_1 &= \tilde{\mathcal{H}}(lPersona) \ \&\& \ h_2 \\ &= \tilde{\mathcal{H}}(i, PP, SP) \ \&\& \ true \\ &= \mathcal{V}_{PP\_key}(s, m_{PP}) \end{aligned} \tag{24}$$

where $\mathcal{V}$ verifies signatures based on the key *PP_key*.

Finally, the SP checks whether $n \leq threshold$, where *threshold* is the limit set by the SP per individual, in the interval i.

The described algorithm enables SPs to put additional constraints on the rate or way individuals access services, while it prevents PP from knowing which SPs are being used. Appendix B.7 applies the notion of constrained interactions in the area of anonymous reputation management. The application limits the effects of Sybil attacks.

PPs may compute h_2 for commonly used services and products; and use such values to query individuals' requests to determine which individuals have used these products. We assume that PPs do not carry out such attacks. Note that one may use a non-collision resistant hash function, but this leads to collisions. In this case, SPs may deny individuals from receiving legitimate access to services, but this should not be a serious problem, since it happens rarely and at random.

B.7 Ticket application: prevention of Sybil attacks

The individual prepares a ticket request m_I, as in (23), where *SP* is the SP the individual wishes to rate. The individual sends m_I to her PP. The PP generates a ticket t, as in (24), and sends it back to the individual. The individual sends m_I and t to the RMC. The RMC uses (24) to verify t. Recall that t includes n, which represents the number of times that the individual has requested a ticket for the same SP, in a given time interval. Finally, the RMC allows the individual to rate the SP only if $n = 1$, that is, this the first time a ticket is generated on behalf of the individual for the specified SP.

Note that the RMC should make sure that the locked persona in the ticket is the same locked persona that the individual submits in the reputation message.

The described algorithm limits the ability of an individual to rate an SP to once per time interval, while it prevents PP from knowing which SPs are being rated.

B.8 System operations: mapping the constructs

The building blocks of the system are now ready and given by Table 2. The table shows each operation/concept in our system and its equivalent construction that uses and extends the hidden ID-based signatures. `Show` and `Verify` allows for anonymity and unlinkability of interactions. Selective release of attributes is achieved by `SelectiveShow`, whereas encoding and verifying relations is achieved by `VerifyRelation`. `Trace` implements persona traceability. `SubmitReputation` and `UpdateReputation` allow for anonymous reputation management. `VerifyTicket` and `GenerateTicket` permit ticket management.

The HIDS Sign operation can be optimized to generate a HIDS signature with two pairing operations and 14 exponentiations. The HIDS Verify operation can be optimized to verify a HIDS signature with two pairing operations and 10 exponentiations [21]. Therefore, generating a locked persona that does not show any attribute requires the same number of parings and exponentiations to generate one HIDS signature. Verifying that locked persona requires the same number of parings and exponentiations to verify one HIDS signature. To generate a locked persona that shows attributes, one need double the operations for generating a

Table 2 Mapping cryptographic constructs to the system operations

Concepts and operations		The extended HIDS
`PP`	→	`IDP`
`DA`	→	`DA`
`persona`	→	Identifier
`secret`	→	Certificate
`attribute`	→	Attributes part of an identifier
`locked persona`	→	(HIDS signature, message)
`ticket`	→	Tickets as in Appendix B.6
`Wrap`	→	$\texttt{Register}_e$
`Check_Wrap`	→	$\texttt{Check Reg}_e$
`Show`	→	$\texttt{Sign}_e$
`Verify`	→	$\texttt{Verify}_e$
`Trace`	→	$\texttt{Open}_e$
`SelectiveShow`	→	Appendix B.3
`VerifyRelation`	→	Appendix B.4
`SubmitReputation`	→	Appendix B.5
`VerifyReputation`	→	Appendix B.5
`UpdateReputation`	→	Appendix B.5
`GenerateTicket`	→	Appendix B.6
`VerifyTicket`	→	Appendix B.6

HIDS signature, that is, four pairings and 28 exponentiations. Verifying that locked persona requires four pairings and 20 exponentiations.

Appendix C: Correctness and security analysis

C.1 Threat model

The threat model is illustrated in Fig. 4. The figure shows the attacks that can be launched against the system. The adversaries are represented as red circles and labeled A_1 to A_4. The following describes each attack:

– *Forging personas*. A_1 issues a persona that is valid with respect to a persona provider.
– *Forging locked personas*. A_2 generates a verifiable locked persona that corresponds to a valid persona, which A_2 does not posses.
– *Forging attributes*. A_2 has a valid persona and uses it to generate verifiable locked personas that contain attributes that the persona provider did not include in the corresponding persona.
– *Linking locked personas*. A_4 determines whether two locked personas have been generated by the same persona or not.
– *De-anonymizing locked personas*. A_3 traces a locked persona to the persona used to generate that locked persona.

The described threat model suggests several attacks: forging a persona, forging a locked persona, forging an attribute, linking locked personas, and de-anonymizing a locked persona. Below is the list of the attacks and their prerequisites.

– *Forging personas*. Forging personas can be achieved if the *Wrap* operation is insecure. *Rational*. If *Wrap* is not secure, an attacker may issue personas valid with respect to a PP, without having the private key of the PP.
– *Forging locked personas or attributes*. Forging locked personas or attributes can be achieved if the *Wrap* or the *Show* operations are insecure. *Rational*. If *Wrap* is not secure, an attacker may issue personas, and thus, may generate valid locked personas. If *Show* is not secure, an attacker may use a valid persona to generate locked personas with attributes that the attacker is not entitled to.
– *Linking or de-anonymizing locked personas*. Linking or de-anonymizing locked personas can be achieved if the *Trace* operation is insecure. *Rational*. If *Trace* is not secure, an attacker may de-anonymize a set of locked personas and link those which belongs to the same persona to each other.

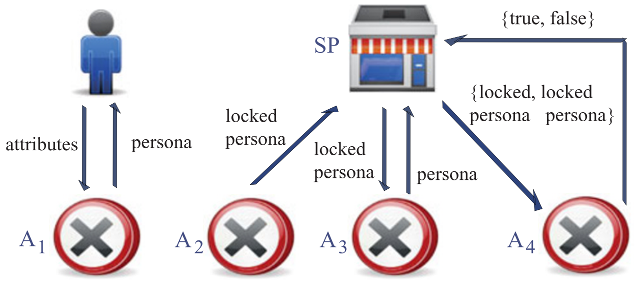

Fig. 4 The threat model

To protect the system against the attacks described in the threat model, we need to make sure that the operations that manage personas are correct, as well as, secure. The following sections prove the correctness and security of the operations.

The operations provided by the system can be categorized into: secure interaction operations and reputation management operations. The first category includes: *Wrap*, *Show*, *Verify*, *Trace*, *SelectiveShow,* and *VerifyRelation*. The second includes: *SubmitReputation*, *VerifyReputation*, *UpdateReputation*, *GenerateTicket*, and *VerifyTicket*. The correctness and security analysis is structured according to the categories.

C.2 Correctness of secure interaction operations

The correctness and security of the secure interaction operations is mainly drawn from the correctness and security of the underlying hidden ID-based signatures. *Wrap*, *Show*, *Verify*, and *Trace* are the operations in which other secure interaction operations are built on-top. Proving the correctness and security of *Wrap*, *Show*, *Verify,* and *Trace* operations implies the security and correctness of the remaining ones. The operations are correct if the following three conditions hold. *Check_Wrap* always succeed when executed on a valid (persona, secret) pair.

$$\begin{aligned}&\mathit{Probability}\big[(\mathit{persona}, \mathit{secret})\\&\quad \leftarrow \mathit{Wrap}(\mathit{attributes}, \mathit{proof}, \mathit{PP\,pparam}, \mathit{PP\,prkey})\\&\quad \mathit{true} \leftarrow \mathit{Check\_Wrap}(\mathit{persona}, \mathit{secret}, \mathit{PP\,pparam})\big] = 1\end{aligned}$$

Verify always succeed when executed on a valid locked persona *lpersona*.

$$\begin{aligned}&\mathit{Probability}\big[\\&\quad (\mathit{persona}, \mathit{secret}) \leftarrow \mathit{Wrap}(\mathit{attributes}, \mathit{proof},\\&\quad \mathit{PP\,pparam}, \mathit{PP\,prkey})\\&\quad \mathit{true} \leftarrow \mathit{Check\_Wrap}(\mathit{persona}, \mathit{secret}, \mathit{PP\,pparam})\\&\quad \mathit{lpersona} \leftarrow \mathit{Show}(\mathit{persona}, \mathit{secret}, \mathit{PP\,pparam},\\&\quad \mathit{DA\,pparam})\\&\quad \mathit{true} \leftarrow \mathit{Verify}(\mathit{lpersona}, \mathit{PP\,pparam}, \mathit{DA\,pparam})\big] = 1\end{aligned}$$

Trace always extracts the persona used by the show operation to generate a locked persona verifiable by the verify operation.

$$\begin{aligned}&\mathit{Probability}\Big[(\mathit{persona}, \mathit{secret})\\ &\quad \leftarrow \mathit{Wrap}(\mathit{attributes}, \mathit{proof}, \mathit{PP\ pparam}, \mathit{PP\ prkey})\\ &\quad \mathit{true} \leftarrow \mathit{Check\_Wrap}(\mathit{persona}, \mathit{secret}, \mathit{PP\ pparam})\\ &\quad \mathit{lpersona} \leftarrow \mathit{Show}(\mathit{persona}, \mathit{secret}, \mathit{PP\ pparam},\\ &\quad \mathit{DA\ pparam})\\ &\quad \mathit{true} \leftarrow \mathit{Verify}(\mathit{lpersona}, \mathit{PP\ pparam}, \mathit{DA\ pparam})\\ &\quad \mathit{persona} \leftarrow \mathit{Trace}(\mathit{lpersona}, \mathit{DA\ pparam}, \mathit{DA\ prkey})\Big] = 1\end{aligned}$$

The proof of correctness of the hidden ID-based signature scheme is presented in [21]. *Wrap* is implemented by *Register$_e$*, which is a composition of two instances of *Register* in HIDS. *Show* is implemented by *Show$_e$*, which is a composition of two instances of *Sign* in HIDS. *Verify* is implemented by *Verify$_e$*, which is a composition of two instances of *Verify* in HIDS. *Trace* is implemented by *Open$_e$*, which is a composition of two instances of *Open* in HIDS. Therefore, the three conditions described above hold. *Wrap*, *Show*, *Verify*, and *Trace* are correct, based on the correctness of the operations in the HIDS scheme.

SelectiveShow is a composition of n instances of *Sign$_e$*, where n is the number of different sets of attributes the individual wishes to prove. *VerifyRelation* is a composition of two instances of *Verify$_e$*, plus an additional check for the relation ($W_1\, g^{\text{relation}} = W_2$); where $W_1 = w^{l+k}\, g^{I_1}$, and $W_2 = w^{l+k}\, g^{I_2}$. Clearly, the check holds only if $\mathit{relation} = I_2 - I_1$.

Recall that when a PP validates the relation between two individuals D_1 and D_2, the PP makes the pseudonym part of their identifiers to be equal. The PP also makes the difference between relation part of their identifiers to be *relation*. Thus, *relation* is equal to $I_2 - I_1$, only if the PP did certify the relation between D_1 and D_2.

Therefore, *SelectiveShow* and *VerifyRelation* are correct based on the correction of *Wrap*, *Show*, *Verify*, and *Trace*.

C.3 Correctness of reputation management operations

SubmitReputation and *VerifyReputation* are correct if the following condition holds. *Verify_Reputation* always succeed when executed on a valid reputation message. Since the implementation of *UpdateReputation* varies from one domain into another, and is based on system administrators, the correctness of the operation not discussed.

$$\begin{aligned}&\mathit{Probability}\Big[\mathit{lpersona}_i\\ &\quad \leftarrow \mathit{Show}(\mathit{persona}, \mathit{secret}, \mathit{PP\ pparam}, \mathit{DA\ pparam})\\ &\quad \mathit{true} \leftarrow \mathit{Verify}(\mathit{lpersona}_i, \mathit{PP\ pparam}, \mathit{DA\ pparam})\\ &\quad \mathit{reputation}, \mathit{lperonsa}_r \leftarrow \mathit{SubmitReputation}(\mathit{score},\\ &\quad \mathit{lperonsa}_i, \mathit{persona}, \mathit{secret}, \mathit{DA\ pparam},\\ &\mathit{PP\ pparam}, \mathit{SP}, \mathit{response})\\ &\quad \mathit{true} \leftarrow \mathit{Verify\_Reputation}(\mathit{reputation}, \mathit{lpersona}_r,\\ &\quad \mathit{DA\ pparam}, \mathit{PP\ pparam})\Big] = 1\end{aligned}$$

SubmitReputation generates a locked persona on a reputation message. *SubmitReputation* uses *Show* to generate the locked persona. *VerifyReputation* verifies the locked persona contained in the reputation message *lpersona$_i$*, and the locked persona generated on the reputation message *lpersona$_r$*. *VerifyReputation* uses *Verify* to verify the locked personas. *VerifyReputation* uses the RSA public key of SP to validate the SP's response contained in the reputation message, and to make sure that the individual had an interaction with the SP being rated.

Since *Show* and *Verify* are proven to be correct, and the RSA public cryptography is correct, then the above condition holds. *SubmitReputation* and *VerifyReputation* are correct, based on the correctness of *Show*, *Verify* and RSA.

GenerateTicket and *VerifyTicket* are correct if the following condition holds.

$$\begin{aligned}&\mathit{Probability}\Big[\\ &\quad \mathit{ticket} \leftarrow \mathit{GenerateTicket}(\mathit{tRequest}, \mathit{PP\ prkey},\\ &\quad \mathit{PP\ pparam}, \mathit{response})\\ &\quad \mathit{true} \leftarrow \mathit{Verify\_Ticket}(\mathit{ticket}, \mathit{tRequest}, \mathit{PP\ pparam})\Big] = 1\end{aligned}$$

Recall that a ticket request *tRequest* consists of two hashed strings, generated by a hash function. *GenerateTicket* sign ticket requests to generate tickets. Tickets are RSA signatures on a ticket request. *VerifyTicket* uses signature verification of RSA to verify tickets against ticket requests. Since RSA signatures generated by RSA signing keys are verified by the corresponding RSA verification keys, the above condition holds. Therefore, *GenerateTicket* and *VerifyTicket* are correct, based on the correctness of RSA public cryptography.

C.4 Security of secure interaction operations

Wrap, *Show*, *Verify*, and *Trace* are secure against misidentification attacks, if the probability of an adversary succeeding in the following game is negligible. In this game, the adversary has access to *Wrap_Oracle*, which executes *Wrap* and returns the resultant (persona, secret) pair. The adversary has access to *Show_Oracle*, which executes *Show* and returns the resultant locked persona. The adversary wins the game if it produces a valid locked persona that is either not traceable to a persona, or is traceable but the adversary has not used *Wrap_Oracle* to receive that persona and she did not use *Show_Oracle* to produce that locked persona.

```
**Wrap_Oracle**$(attributes)$
    $(persona, secret) \leftarrow Wrap(attributes, proof, PP\,pparam, PP\,prkey)$
    $Personas \leftarrow \{persona\} \cup Personas$
    $return \quad (persona, secret)$
**Show_Oracle**$(persona)$
    $lpersona \leftarrow Show(persona, secret, PP\,pparam, DA\,pparam)$
    $Lpersona \leftarrow \{lpersona\} \cup Lpersonas$
    $return \quad lpersona$
**Misidentification_Game**$()$
    $lpersona \leftarrow Adversary(Wrap\_Oracle, Show\_Oracle)$
    **if**$(true \leftarrow Verify(lpersona, PP\,pparam, DA\,pparam)$
    **AND** $\Phi \leftarrow Trace(lpersona, DA\,pparam, DA\,prkey)$
        $Adversary \quad wins$
    **elseif**$(true \leftarrow Verify(lpersona, PP\,pparam, DA\,pparam)$
        **AND**
    $persona \leftarrow Trace(lpersona, DA\,pparam, DA\,prkey))$
    **AND**
    $persona \notin Personas$ **AND** $lpersona \notin Lpersonas)$
        $Adversary \quad wins$
    **else** $Adversary \quad loses$
```

Wrap, *Show*, *Verify*, and *Trace* is secure against adaptive chosen-cyphertext attacks (CCA2), if the probability of an adversary succeeding in following two game is $0.5 + \epsilon$, where ϵ is negligible. The adversary has access to *Trace_Oracle*, which reveals the persona used to generate a locked persona. The adversary is presented with a locked persona and two personas, in which one of personas was used to generate the locked persona. The adversary wins the game if it guesses the right persona. Of course, the adversary is constrained from using *Trace_Oracle* on the presented locked persona. $Trace\_Oracle_\alpha$ refers to that constraint.

```
**Trace_Oracle**$(lpersona)$
    $persona \leftarrow Trace(lpersona, DA\,pparam, DA\,prkey)$
    $return \quad persona$
```

```
**CCA2_Game**$()$
    $(persona_1, secret_1) \leftarrow Wrap(attributes_1, proof_1, PP\,pparam, PP\,prkey)$
    $(persona_2, secret_2) \leftarrow Wrap(attributes_2, proof_2, PP\,pparam, PP\,prkey)$
    $r \leftarrow random \quad from \; \{1, 2\}$
    $lpersona \leftarrow Show(persona_r, secret_r, PP\,pparam, DA\,pparam)$
    $challenge \leftarrow \{lpersona, persona_1, persona_2\}$
    $guess \leftarrow Adversary(trace\_oracle_\alpha, challenge)$
    **if**$(guess = persona_r)$
        $Adversary \quad wins$
    **else** $Adversary \quad loses$
```

Let an adversary A has the ability to launch successful misidentification and or CCA2 attacks on *Wrap*, *Show*, *Verify*, and *Trace*. Those operations are implemented by $Register_e$, $Sign_e$, $Verify_e$, and $Open_e$ in extended HIDS, respectively, whereas $Register_e$, $Sign_e$, $Verify_e$, and $Open_e$ are instances of HIDS operations. Therefore, A can launch successful misidentification and or CCA2 attacks on the HIDS operations.

The HIDS scheme is proven to be secure against misidentification and CCA2 attacks under the Strong Deffie–Helman (SDH) [4] and Decisional Linear Deffie–Helman (DLDH) [5] assumptions in the random oracle model. The security proof is presented in [21]. Since HIDS operations are proven to be secure, then *Wrap*, *Show*, *Verify*, and *Trace* are also secure against misidentification and or CCA2 under the SDH and DLDH assumptions in the random oracle model. The security of *SelectiveShow VerifyRelation* follows from the security of *Wrap*, *Show*, *Verify*, and *Trace*.

C.5 Security of reputation management operations

Since *Show* and *Verify* are secure, and RSA public cryptography is secure, then *SubmitReputation* and *VerifyReputation* are secure. The security of *GenerateTicket* and *VerifyTicket* follows from the security of the RSA public cryptography.

References

1. Artz D, Gil Y (2007) A survey of trust in computer science and the semantic web. J Web Semant 5(2):58–71
2. Becker M, Sewell P (2004) Cassandra: distributed access control policies with tunable expressiveness. In: Proceedings of the fifth IEEE international workshop on policies for distributed systems

and networks. IEEE Computer Society, Los Alamitos, pp 159–168
3. Berners-Lee T, Hendler J, Lassila O (2001) The semantic web, May 2001. Scientific American Magazine. Retrieved from http://www.sciam.com/article.cfm?id=the-semantic-web, on Jan 2011
4. Boneh D, Boyen X (2004) Short signatures without random oracles. In: Proceedings of the 24th international conference on the theory and applications of cryptographic techniques. Springer, Berlin, pp 56–73
5. Boneh D, Boyen X, Shacham H (2004) Short group signatures. In: Proceedings of the 24th international conference on the theory and applications of cryptographic techniques. Springer, Berlin, pp 41–55
6. Brands S (2000) Rethinking public key infrastructures and digital certificates: building in privacy. MIT Press, Cambridge
7. Brin S, Page L (1998) The anatomy of a large-scale hypertextual web search engine. Comput Netw ISDN Syst 30(1–7):107–117
8. Camenisch J, Herreweghen EV (2002) Design and implementation of the idemix anonymous credential system. In: Proceedings of the ACM conference on computer and communications security. ACM Press, New York, pp 21–30
9. Damiani E, Vimercati DCD, Paraboschi S, Samarati P, Violante F (2002) A reputation-based approach for choosing reliable resources in peer-to-peer networks. In: Proceedings of the ACM conference on computer and communications security. ACM Press, New York, pp 207–216
10. Dimitriou T, Karame G, Christou I (2007) SuperTrust: a secure and efficient framework for handling trust in super-peer networks. In: Proceedings of the twenty-sixth annual ACM symposium on principles of distributed computing. ACM Press, New York, pp 374–375
11. Dingledine R, Mathewson N, Syverson P (2004) Tor: the second-generation onion router. In: Proceedings of the 13th USENIX security symposium. USENIX Association, Berkeley, p 21
12. Douceur J (2002) The sybil attack. In: Proceedings of the first international workshop on peer-to-peer systems, IPTPS, Cambridge, MA, USA, pp 251–260
13. Golbeck J, Hendler J (2004) Accuracy of metrics for inferring trust and reputation in semantic web-based social networks. In: Proceedings of the international conference on knowledge engineering and knowledge management, pp 116–131
14. Grandison T, Sloman M (2000) A survey of trust in internet applications. IEEE Commun Surv Tutor 3(4):2–16
15. Hansen M, Berlich P, Camenisch J, Clauss S, Pfitzmann A, Waidner M (2004) Privacy-enhancing identity management. Inf Secur Tech Rep 9(1):35–44
16. Hussain M, Skillicorn DB (2008) Persona-based identity management: a novel approach to privacy protection. In: Proceedings of the 13th Nordic workshop on secure it systems. Technical University of Denmark, pp 201–212
17. Hussain M, Skillicorn DB (2009) Guarantee-based access control. In: Proceedings of the IEEE international conference on computational science and engineering. IEEE Comput Soc, Los Alamitos, pp 201–206
18. Hussain M, Skillicorn DB (2010) The case for service provider anonymity. In: Proceedings of the IEEE international symposium on signal processing and information technology. IEEE Comput Soc, Los Alamitos, pp 114–119
19. Kagal L, Finin T, Joshi A (2002) Developing secure agent systems using delegation based trust management. In: Proceedings of security of mobile multiagent systems held at autonomous agents and multiagent systems, pp 27–34
20. Kamvar S, Schlosser M, Garcia-Molina H (2003) The eigentrust algorithm for reputation management in p2p networks. In: Proceedings of the 12th international conference on World Wide Web. ACM Press, New York, pp 640–651
21. Kiayias A, Zhou H-S (2008) Hidden identity-based signatures. In: Proceedings of the 11th international conference on financial cryptography and data security. Springer, Berlin, pp 134–147
22. Lynn B (2011) Pairing-based cryptography library. Retrieved from http://crypto.stanford.edu/pbc, on Jan 2011
23. Malin B, Sweeney L (2004) How (not) to protect genomic data privacy in a distributed network: using trail re-identification to evaluate and design anonymity protection systems. J Biomed Inform 37(3):179–192
24. Müller W, Plötz H, Redlich J-P, Shiraki T (2008) Sybil proof anonymous reputation management. In: Proceedings of the 4th international conference on security and privacy in communication networks. ACM Press, New York, pp 1–10
25. Narayanan A, Shmatikov V (2008) Robust de-anonymization of large sparse datasets. In: Proceedings of the IEEE symposium on security and privacy. IEEE Comput Soc, Los Alamitos, pp 111–125
26. Narayanan A, Shmatikov V (2009) De-anonymizing social networks. In: Proceedings of the IEEE symposium on security and privacy. IEEE Computer Society, Los Alamitos (in press)
27. Rezgui A, Bouguettaya A, Malik Z (2003) A reputation-based approach to preserving privacy in web services. In: Lecture notes in computer science, vol 2819, pp 91–103
28. Singh A, Liu L (2003) TrustMe: anonymous management of trust relationships in decentralized P2P systems. In: Proceedings of the third conference on peer-to-peer computing, pp 142–149
29. Skillicorn DB, Hussain M (2009) Personas: beyond identity protection by information control. Technical report, School of Computing, Queen's University, Kingston, ON, Canada, March 2009. Retrieved from http://research.cs.queensu.ca/home/skill/opccreport.pdf, on Jan 2011
30. Sweeney L (2002) k-Anonymity: a model for protecting privacy. Int J Uncertain Fuzziness Knowl-Based Syst 10(5):557–570
31. Yu T, Winslett M, Seamons K (2003) Supporting structured credentials and sensitive policies through interoperable strategies for automated trust negotiation. ACM Trans Inf Syst Secur 6(1):1–42