2016 IEEE 2nd International Conference on Collaboration and Internet Computing

RahasNym: Pseudonymous Identity Management System for Protecting against Linkability.

Hasini Gunasinghe
Department of Computer Science
Purdue University
West Lafayette, Indiana
Email: huralali@purdue.edu

Elisa Bertino
Department of Computer Science
Purdue University
West Lafayette, Indiana
Email: bertino@purdue.edu

***Abstract*—Unlinkability and accountability are conflicting yet critical requirements that need to be addressed in order to preserve users' privacy as well as to protect service providers in today identity ecosystems. In this paper we present a pseudonymous identity management system in which users can carry out unlinkable on-line transactions without having to disclose their actual identity information in plain text to the service providers. At the same time, the service providers have strong assurance about the authenticity of the transactions. In our approach, users' identity is cryptographically encoded in pseudonymous identity tokens. Our system includes a lightweight policy language which enables users and service providers to express their requirements pertaining to pseudonymous identity verification and a suite of protocols based on zero-knowledge-proofs which enables the fulfillment of these requirements. We have implemented a prototype of the proposed system and carried out a security and performance analysis.**

I. Introduction

Repositories storing users' identity information, such as credit card numbers, email addresses and user names managed at various on-line service providers (SPs) have become the target of attackers as evident from the recent history of cyber crimes. Some attacks exploit user identity information linked across multiple accounts of the same individual held at different SPs. An example is the epic attack [1] on the multiple accounts of the Wired writer Mat Honan, which were linked by his real name. In this attack, the last four digits of his credit card number stolen from his Amazon account were used by the attackers to gain access to his iCloud account through which they deleted the data in all his Apple devices. Apart from external attackers, SPs themselves can link user identities associated with different accounts and transactions for various purposes which may undermine user privacy.

As a basic solution, people can use pseudonyms when registering and interacting with various on-line systems, such as on-line merchants and discussion forums, in order to prevent their accounts held at different systems from being linked through their real name. A pseudonym is a digital identifier used to identify an individual which is different from the individual's real name. However if the other unique identity information, such as user's email address, credit card number (CCN), social security number (SSN), is provided to multiple SPs in plain text, accounts and transactions of the same user at different SPs can be easily linked through such information. Therefore, pseudonym based identity management, which aims to prevent linkability, should be accompanied by mechanisms allowing the users to prove the ownership of valid identities without having to reveal them in plain text, such as Zero Knowledge Proof of Knowledge (ZKPK) schemes [2]. In addition to help preventing linkability, such solutions also avoid security breaches caused by the use of identity information in clear text at third party SPs, such as the recent attacks on eBay [3] and Target [4].

On the other hand, it is also critical to protect SPs from abuses or frauds that could be perpetrated by individuals hidden by pseudonyms. For example, users may submit stolen credit card details or overuse a certain privilege granted to a particular individual by associating it with multiple different pseudonyms. Therefore, SPs need assurance about the ownership of the users' identities and accountability of the transactions. The level of assurance required w.r.t different properties of pseudonymous identity verification varies depending on the nature of the transaction as well as the preferences of the users and the SPs involved in the transaction. Hence, we need flexible identity management systems which accommodate identity verification according to such varying requirements.

In order to prevent privacy breaches resulting from identity linkability, the use of anonymous credentials has been proposed [5], [6], [7], [8], [9] based on different cryptographic techniques. However, each of such previous approaches addresses only a subset of the critical requirements related to pseudonymous identity verification. For examples, the approach by Chaum [7] supports unconditional untraceability at the cost of accountability assurance. His group signature scheme [6] facilitates anonymous identity verification of an individual belonging to a group, at the cost of scalability. Idemix [5] supports unconditional unlinkability at the cost of involving additional parties in the revocation and inspection and making the identity tokens and the associated proof protocols complex and bulky [10]. Therefore, widely used on-line transaction systems such as online merchants have not yet adopted such schemes to enable users to carry out online transactions in a privacy preserving manner, without concerns for linkability. Furthermore, to date there is no flexible identity management system that enables both the parties involved

978-1-5090-4607-2/16 $31.00 © 2016 IEEE
DOI 10.1109/CIC.2016.21

in a transaction to specify, negotiate and fulfill their varying requirements related to identity verification.

The aim of this work is to develop a practical pseudonymous identity management system which protects user's privacy against linkability in online-transactions while safeguarding SPs against frauds, and which can be adopted by existing transaction systems without much overhead in terms of architectural changes and performance. The pseudonymous identity management system called RahasNym, that we propose in this paper, includes a policy framework with a simple and expressive policy language and a suite of cryptographic protocols in order to achieve aforementioned goals. Summarized below are the main contributions of the paper:

1) Identification of key requirements in pseudonyms based digital identity management.
2) Definition of a policy framework supporting the specification of such requirements.
3) Definition of a suite of protocols implementing the specified identity verification policies.
4) An evaluation of a prototype implementation of RahasNym in terms of security and performance.

The rest of the paper is organized as follows. In Section II we present an overview of how a real world scenario is implemented with RahasNym. In Section III, we present our approach followed by the implementation details and performance analysis on RahasNym in Section IV. Section V includes the security analysis, Section VI discusses related work and Section VII concludes the paper. The paper also includes an appendix to present further details about our system.

II. Overview of the Solution

In this section, we provide a bird's-eye view of our solution by illustrating how users would carry out transactions in an on-line shopping system that adopts RahasNym.

An on-line merchant, for example Amazon, exposes different operations to users such as signing-up, registering for free-shipping membership and purchasing items, each of which requires the user to present different identity attributes with varying privacy-sensitivity. For example, the operation of purchasing items requires the user's approval to charge the payment amount on her credit/debit card, the shipping address for the item to be shipped and the email address to send delivery notifications.

RahasNym involves three main parties: user, SP and Identity Provider (IDP). During the initialization phase of RahasNym, both the user and the SP define their identity verification policies with fine-grained rules. When the user wants to perform a certain transaction, the user agent (i.e: the software in the user's device) obtains from the SP the identity verification policy related to that particular operation and checks if this policy matches the user policy. If they match, the user agent acquires the Identity Token(s) (IDT) required to perform the transaction from the corresponding IDP(s). For example, if CCN is required, the CCN-IDT is obtained from the user's credit card issuer and if the email address is required, the email-IDT is obtained from the user's email provider. The IDP cryptographically encodes the user's identity information using the Pedersen Commitment scheme [11] and sends back to the user agent the digitally signed IDT. Then the user agent and the SP execute the agreed identity verification protocol through which the user proves his/her identity to the SP in zero-knowledge. If there is a conflict in combining the two policies, the user agent prompts the user to change her policy or aborts the operation.

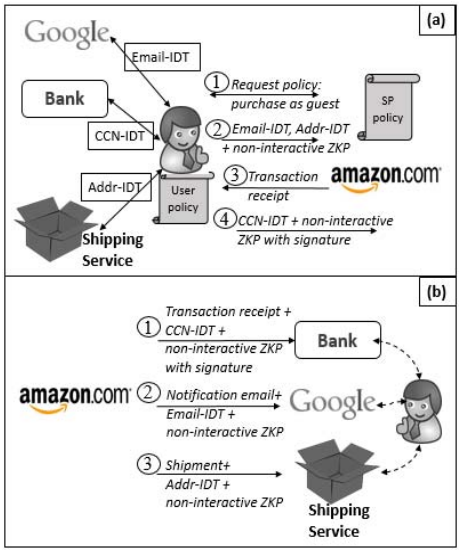

Fig. 1. (a) Identity verification at on-line shopping scenario (b) The merchant performing activities related to the purchase with the IDPs.

In what follows we analyze the identity verification carried out with regard to the operation, 'purchasing an item as a guest'. Assume that the user has active on-line accounts at the bank, shipping service, and email provider which act as IDPs issuing the respective identities to the user in the form of cryptographic IDTs. As shown in the step 2 of Figure 1(a), the first two attributes requested by the on-line merchant are email address and shipping address. However, since they are cryptographically encoded in the IDTs, there should be a way for the on-line merchant to verify the ownership of the IDTs and to use these IDTs for sending delivery notifications and shipping the items to the user. On the other hand, when the on-line merchant requests the email provider and the shipping service provider to perform the aforementioned operations by forwarding the user-provided IDTs, they should be able to link those IDTs to the user's accounts held at them, and verify that it is the actual owner who has given the IDTs to the particular merchant. We have defined an identity verification protocol based on non-interactive ZKPK schemes to address requirements of these kinds of scenarios. After successful identity verification, the on-line merchant saves the IDTs and the associated non-interactive zero knowledge proofs provided by the user, in order to use them later to send email notifications and ship item(s) through the respective IDPs[1] as

[1]Note that these identity providers also become SPs in different contexts.

shown in steps 2 and 3 of Figure 1(b). Those IDPs, after verifying that the proofs of ownership have been created by the genuine owners of the IDTs, complete the operations requested by the on-line merchant. The 'return labels' feature offered by on-line shipping services such as FedEx, can easily adopt the proposed scheme to allow users to obtain the IDTs which cryptographically encode their shipping address, instead of the plain-text shipping address included in the return labels today. Here, FedEx acts as the IDP for the user's identity represented by his/her shipping address and once the packages with such IDTs are received, the shipping service can link them to the actual user's account via the auxiliary information contianed in the IDTs.

We have extended the basic non-interactive ZKPK protocol to provide more authenticity for payment related identity verifications. For an example, the purpose of requesting credit card information is to claim the payment from the user's bank account. If the bank approves the claimed payment based only on the IDT and associated proofs as in the previous case, a malicious on-line merchant could claim an arbitrary payment (with a higher amount) that is not approved by the user, simply by forwarding the IDTs and proofs provided by the user for a different transaction (associated with a lower amount). Therefore, it is important that the bank obtains a proof that the owner of the CCN-IDT has approved the current transaction, before performing the credit transfer to the merchant's account (step 1 of Figure 1(b)). In order to address such requirements, we have defined another protocol which requires to include the transaction receipt also in the signature created during the non-interactive ZKPK protocol carried out in order to prove the ownership of the CCN-IDT (step 4 of Figure 1(a)). In this way, both the on-line merchant and the bank can verify the proof and have assurance about the authenticity of the transaction while neither on-line merchant nor any intermediate entity in the credit card network can see the actual details of user's credit/debit card. Different transactions are unlinkable since each time a new IDT is used.

For scenarios in which the identity verification is performed only by the party whom the user directly interacts with, RahasNym provides an interactive ZKPK protocol. For example, if the on-line merchant requires the user to prove that she is a student in order to be eligible to subscribe for free-shipping membership, the user can prove the ownership of a college-issued identity token through this protocol.

III. Our Approach

In what follows, we first identify the set of key requirements of a pseudonymous identity management system which is able to protect both users and SPs. Based on such requirements, we then introduce the policy framework that enables users and SPs to enforce such requirements during identity verification of pseudonymous transactions. Next we present the suite of protocols which facilitates acquiring IDTs and performing identity verification adhering to the specified policies.

A. Key Requirements

(1) Ownership Assurance: Each pseudonymous IDT should be used only by its genuine owner. In other words, neither an attacker with a stolen IDT nor a malicious SP with an IDT provided by its owner during an identity verification operation should be able to use this IDT and impersonate the user. This property is important for the user in order to protect their IDTs from being misused and for the SPs to verify that they are dealing with a legitimate owner of an IDT.

(2) Unlinkability: Different transactions and accounts of the same user managed under different pseudonyms should not be linkable. Unlinkability can be assured at three different levels:

1) Level 1:across multiple transactions with the same SP.
2) Level 2:across transactions with different SPs.
3) Level 3:across SPs and IDPs.

This property is important mainly for the user. However, the SPs can also specify which level of unlinkability they would offer for the users. For example, certain SPs, such as banks, might require the users to have only one pseudonym for all the transactions carried out with them, in order to maintain the transaction history of the user.

(3) Confidentiality: Protecting confidentiality of user identities without relying only on the transport level security is a general requirement of any digital identity management system. In addition, pseudonymous identity management should also protect confidentiality of different pseudonyms bound to the IDTs so that even the IDPs which issue the IDTs do not learn the pseudonyms.

(4) Accountability: Users should be held accountable for transactions performed with pseudonymous IDTs issued to them. In other words, if a user whose identity is represented by a pseudonymous IDT, has committed a fraud in a particular transaction, there should be means to de-anonymize the user, in order to take actions against him/her. This requirement is for protecting the SPs.

(5) Non-shareability: This requirement refers to preventing the legitimate owner of an IDT from deliberately sharing it with other users. Preventing the sharing protects the SPs from the users who might exploit certain privileges granted to them by sharing their IDTs with others. Non-shareability can be assured at two levels:

1) Level 1: discouraging the sharing of IDTs by enforcing negative consequences of sharing such as having to share the password accociated with the credential or having to share the user's device itself whose TPM (Trusted Platform Module) contains the master secret of the anonymous credentials [12].
2) Level 2: preventing the sharing of IDTs by including the user's biometric identity in the IDT and requesting to verify it at the identity verification.

(6) Authenticity: This requirement is related to guarantee that the transaction has been approved by the legitimate owner of the IDT (e.g: CCN-IDT) which is used during the transaction. In many of today online transaction systems,

any transaction performed through the logged-in session is considered valid, without any verification that the legitimate user has approved each transaction. If a malicious party steals the logged-in session, the authenticity of the transaction is at risk. Furthermore, as in the scenario discussed in Section II, the user does not get to directly interact with the bank during the payment process. Hence there should be a way for all the parties involved in the transaction to verify that the legitimate user has actually approved the current transaction.

(7) Invalidation of IDTs: The user should be able to invalidate an IDT in case of a compromise of the secrets involved with the cryptographic commitments of the IDT. On the other hand, the IDP should be able to invalidate all IDTs issued to a user if the user is detected to have committed a fraud. Therefore, the SP should verify that the IDT used for authentication is valid. This verification should be carried out while at the same time assuring unlinkability.

(8) Flexibility: Since the aforementioned required properties of pseudonymous identity verification vary based on different factors as mentioned in Section I, a pseudonymous identity management framework should enable the participating parties to express their requirements and accommodate such requirements in a flexible manner.

Aforementioned requirements can be satisfied at different levels of assurance. It is important to note that not all of the above requirements can be supported unconditionally in one transaction. In RahasNym, users and SPs can specify the levels at which different requirements are needed to be satisfied, through the policy framework which is described in the following section.

B. The Policy Framework

The policy framework provided by RahasNym consists of the policy language and the policy combining algorithm.

1) Policy Language: The policy language defines the policy elements used to enforce the aforementioned requirements w.r.t pseudonymous identity verification and the policy schema .

Policy Elements: In our model we have three main policy elements, each of which has different options (see Table I). How such options affect the protocol execution is explained in Section III-C.

(1) Subject Verification: This element specifies how a pseudonymous IDT should be bound to the subject who presents the IDT and to the subject who verifies this IDT. This information is recorded in the IDT (in 'From:' and 'To:' fields respectively) and should be verified by the SP at authentication. For a particular transaction, this information can be included either in clear text or in hidden format (i.e: as a commitment). Based on the required level of assurance, this policy element can take different values as shown in Table I. With the options available to specify how the 'From:' field of the IDT should be formed, one can achieve different levels of pseudonymity from full identification (real-name bound) to pseudonimity (pseudonym bound) to full anonymity (hidden pseudonym bound). Optionally, user's biometric identity is bound to require stronger level of non-shareability assurance. This policy element enforces the following subset of requirements: *ownership assurance, non-shareability, and confidentiality of pseudonyms*.

TABLE I
OPTIONS PROVIDED FOR THE POLICY ELEMENTS.

Policy Element	Options
Subject Verification	-Options for the 'From:' field: real-name-bound, pseudonym-bound, hidden-pseudonym-bound, biometric-bound -Options for the 'To:' field: SP-bound, hidden-SP-bound
Proof of Identity	Interactive Zero Knowledge Proof (ZKP-I) Non-Interactive Zero Knowledge Proof (ZKP-NI) ZKP-NI with Signature (ZKP-NI-S)
Pseudonyms Cardinality	Single Multiple

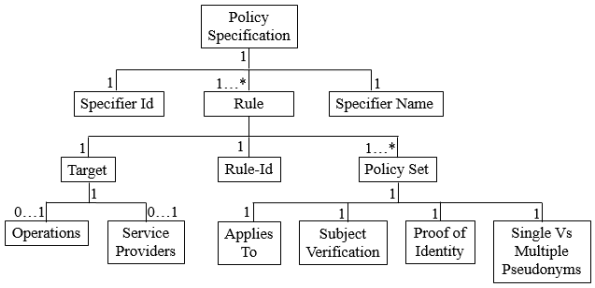

Fig. 2. Structure of the Policy Language.

(2) Proof of Identity: This element specifies how the proof of identity should take place. RahasNym uses cryptographic commitments for encoding identity information and different variations of ZKPK-based protocols to prove the ownership of the identity. Therefore, the different values that this policy element takes correspond to the names of the three identity proof protocols in RahasNym, as listed in Table I. This policy element allows one to enforce the following subset of requirements: *confidentiality of identity information, ownership assurance, and authenticity of the transaction*.

(3) Pseudonyms Cardinality: This element specifies whether the user has to use a single pseudonym or can use multiple pseudonyms when interacting with a particular SP. The two values it takes, shown in Table I, indicate the level of unlinkability assured for the user. If it is *single*, the user has unlinkability assurance only at level 2 and level 3 but not at level 1. If the value is *multiple*, the user has unlinkability assurance at all three levels (see Section III-A for different levels of unlinkability).

All three policy elements can be specified by both users and SPs since they address the requirements of interest of both parties.

Policy Schema: Identity verification policies in RahasNym are expressed in a lightweight policy language built on top of the JSON data format. Figure 2 shows the hierarchical structure of the policy objects that constitute the policy lan-

guage along with their relationship cardinality which defines the schema of a policy specification. 'Policy Specification' is the root object which consists of 'Specifier Id', 'Specifier Name' and one or more 'Rule' objects. One such 'Rule' object may define a fine-grained rule applied to verification of the identity attributes related to a particular transaction. Hence the 'Rule' object consists of a 'Rule Id', a 'Target' and one or more 'Policy Set' objects. The 'Target' object in a SP's policy specification defines the names of the operation(s) to which the 'Rule' is applied and that in a user's policy specification defines the names of the SP(s) to which the 'Rule' is applied. A 'Policy Set' object in a 'Rule' defines the policy elements (see Table I) applied to verification of a particular identity attribute or set of identity attributes. We have provided sample policy specifications by SP and user for further illustration of the use of the policy language.

2) Policy Combining Algorithm: In order to match the user's policy with the SP's policy and to obtain the agreed policy by both the parties, we provide a policy combining mechanism. Basically it goes through both the policies and compares the 'Policy Set' objects defined for the identity attributes required to perform the intended transaction. In case of a conflict, the user can configure the conflict resolution mechanism as *Client Overrides* or *SP Overrides*, which is specified in the 'Target' object of a user's policy specification. If it is *SP Overrides*, the execution proceeds by overriding the user's policy with that of the SP's and if it is *Client Overrides*, the execution aborts giving user the option to change her policy to match that of the SP's if she prefers. An example of a policy combining scenario is also provided in the appendix.

C. The Protocol Suite

The suite consists of one protocol for acquiring IDTs from IDPs and three protocols for *proof of identity*. The three identity proof protocols differ from each other depending on how the prover and the verifier interact during the protocol execution and how the proof of identity is created. Before defining the protocols, we provide an overview of some of the cryptographic primitives used in our protocol definitions.

Pedersen Commitment: We use this for cryptographically encoding the identity information. It is a secure commitment scheme [11] whose security is based on the hardness of solving discrete logarithms. The setup and the commitment creation can be described by the following steps.

Setup: Let p and q be large primes, such that q divides $p-1$. Typically p is of 1024 bits and q is of 160 bits. G_q is a unique, order-q sub group of Z_p. A trusted party (IDP in our case) chooses g -a generator of G_q and h (= g^amod p where 'a' is secret) -an element of G_q such that it is computationally hard to find $log_g h$, and publishes (p, q, g, h).

Commit: Commitment Y is created by choosing $r \in Z_q$ at random and computing: $Y(x, r)$ = $g^x h^r$ mod p $\in G_q$.

Hash Functions (H): Cryptographic hash functions are used for three main purposes in our protocols:

(1) converting the human readable identity information into $x \in Z_q$ for creating the Pedersen Commitment,

TABLE II
NOTATION USED IN THE PROTOCOL SUITE.

Symbol	Meaning
U	User/ User Agent
ID_{sp}	Identity of SP
T	Timestamp at IDT creation
T'	Timestamp at proof creation
D	Expiration time of the IDT
C_{IDP}	Public certificate of IDP
$E_{IDP}(.)$	Encryption using public key of IDP
S_{IDP}	Digital signature of IDP
$E_{SP}(.)$	Encryption using public key of SP
—	Concatenation operation
n	Identity attribute name
a	Identity attribute value
r, s, s'	Secrets derived from user's password
PPC_Params	Public pedersen commitment parameters
f	User's pseudonym at SP
f'	User's pseudonym at IDP
W	Auxiliary element in the IDT.

(2) creating commitments of the user's pseudonyms and SP's identity in order to embed them in the IDT in hidden format, (3) creating non-interactive zero knowledge proofs: as shown in Protocol 3 and 4.

We used SHA-256 for (1) and (2) and SHA-512 for (3) since we need to convert input of (1) and (2) into $x \in Z_q$ and need to extract three $e \in Z_q$ terms, from the hash output of (3).

PBKDF2: We utilize a Password Based Key Derivation algorithm to derive the random secrets required in creating the identity commitments for multiple IDTs and the commitments on user's pseudonyms and SP's identity from a single password provided by the user. This improves usability since the user doesn't have to provide different passwords for creation/verification of each of those commitments associated with different IDTs. Secrets of different bit lengths can be obtained using this algorithm. Inputs to this algorithm are as follows:

W = PBKDF2 (PKCS#5, Password, Salt, derived key length). Salt is 8 bytes long and randomly generated during each run of PBKDF2 so that no two secrets generated from the same password will be the same for a sufficiently long time, after which user can be forced to change the password.

In what follows we define the four protocols in the suite. Table II lists the notation used in the protocol definitions. First we explain the execution of the core protocols that correspond to the default options of the policy elements. In the appendix, we discuss the variations of the core protocols based on the other options of the values taken by the three policy elements introduced in Section III-B.

1) Protocol 1: IDT Request Protocol: The core IDT request protocol corresponds to the basic policy options of: *Subject Verification = pseudonym-bound AND SP-bound*, *Pseudonym Cardinality = Single*. As shown in step 1 of Protocol 1, the user agent sends the IDT request along with the specified input. The user agent sends the secret r (derived from the user's password) encrypted with IDP's public key to be used in creating the identity commitment. The IDP creates the pseudonymous IDT by enclosing the user's identity 'a' corresponding to the

requested identity attribute n, in the commitment Y (step 4 of protocol 1). IDP also creates the auxiliary element W to be included in the IDT which helps the IDP to link an issued IDT with the user's account afterwards, if required. While the IDP can decide which value is included in W, it should not allow multiple IDTs of the same user being linked through W. RahasNym defines W as the user's pseudonym held at the IDP concatenated with the timestamp T and encrypted using IDP's public key. RSA-OAEP is used for encryption which provides both randomized encryption and security against chosen cipher text attacks (CCA-secure). In step 6 of protocol 1, IDP vouches for the validity of the IDT by digitally signing it.

In order to prevent a malicious party from obtaining an IDT with fake identity, the IDP should enforce certain legal procedures such as requiring the user to prove his/her identity using legal proofs during the identity enrollment process at the IDP, which is out of the scope of this work.

Protocol 1 IDT Request Protocol

1: *U→IDP*: authenticates to IDP and sends IDT request along with: p, n, $E_{IDP}(r)$, ID_{SP}.
2: *IDP*: verifies the identity of *U*.
3: encodes a as $x \in Z_q$ using a cryptographic hash function.
4: creates identity commitment: $Y(x, r) = g^x h^r$ mod p.
5: creates auxiliary element: W = $E_{IDP}(f', T)$.
6: creates IDT: IDT = f $|ID_{SP}$ $|n$ $|Y$ $|D$ $|T$ $|W$ |*PPC_Params* $|S_{IDP}(f$ $|ID_{SP}$ $|n$ $|Y$ $|D$ $|T$ $|W$ |*PPC_Params*) $|C_{IDP}$.
7: *IDP→U*: sends the IDT.

2) Protocol 2: Interactive ZKPK Protocol: Protocol 2 lists the core Interactive ZKP Protocol which is used to verify IDTs issued through the core Protocol 1, when *Proof of Identity = ZKP-I* in the agreed polcy specification. In protocol 2, the user agent and the SP engage in an interactive protocol in which the user proves in zero knowledge, her knowledge about the identity information hidden in the commitment of the IDT by responding to the SP's challenges in step 5. This protocol is to be used when the prover and the verifier directly interact with each other.

Protocol 2 Proof in Interactive Zero Knowledge

1: *U→SP*: sends IDT and d, where d = $g^y h^k$ and $y, k \in Z_q$ are randomly selected.
2: *SP*: verifies S_{IDP}, D and T on the IDT.
3: **if** initial verification is successful: **then**
4: *SP→U*: sends challenge $e \in Z_q$.
5: *U→SP*: sends: $u = y + ex$ mod q, $v = k + er$ mod q.
6: SP verifies if Y^ed = $g^u h^v$.
7: if the verification is passed, SP accepts U as the owner of IDT.
8: **end if**

3) Protocol 3: Non-Interactive ZKPK Protocol: The core protocol 3 matches the policy element option: *Proof of Identity = ZKP-NI*. The difference in protocol 3 compared to protocol 2 is that instead of the challenge e sent by the SP in step 4 of protocol 2, the user agent generates three challenges in step 3 of protocol 3, based on which the non-interactive zero-knowledge proofs are created. One-wayness of the cryptographic hash function prevents the user agent from being able to predict the challenges beforehand and thereby prevents any cheating by the user during proof creation. Although we have set the number of challenges to three in the definition of protocol 3, that number can be varied based on the level of difficulty that needs to be maintained against the user's ability to cheat (by using a cryptographic hash function with larger output length (e.g: SHA-1024) or by using PBKDF to obtain an output of arbitrary length). Replay attacks are avoided since the timestamp at the proof creation (T') is included in the input to the hash function. By knowing the inputs given to the hash function and the proofs created based on the output of the hash function, any party can verify the user's ownership of the identity by following the steps from 7-13 in protocol 3.

Protocol 3 Proof in Non-Interactive Zero Knowledge

1: *U*: selects three pairs of $y_j, k_j \in Z_q$, and creates three commitments: $d_j = g^{y_j} h^{k_j}$ mod p where $j \in \{1, 2, 3\}$.
2: Hash Output (h) = H(Y, T', d_1, d_2, d_3).
3: obtains three challenge values: $e_j \in Z_q : j \in \{1, 2, 3\}$, from h above.
4: creates proofs based on e_j values: $u_j = y_j + e_j x$ mod q and $v_j = k_j + e_j r$ mod q, where $j \in \{1, 2, 3\}$.
5: *U→SP*: sends M that takes the form: M = IDT, T', d_1, d_2, d_3, u_1, v_1, u_2, v_2, u_3, v_3
6: *SP*: verifies S_{IDP}, D and T on the IDT and T' in M.
7: **if** initial verification is successful: **then**
8: obtains h' = H(Y, T', d_1, d_2, d_3).
9: obtains challenge values $e'_j : j \in \{1, 2, 3\}$ from h'.
10: **for** each e'_j **do**
11: verifies if $Y^{e'_j} d_j = g^{u_j} h^{v_j}$.
12: **end for**
13: if all the challenge-response verifications are passed, SP accepts U as the owner of IDT.
14: **end if**

4) Protocol 4: Non-Interactive ZKPK with Signature Protocol: The protocol 4 is used for identity verification when the agreed policy specifies *Proof of Identity = ZKP-NI-S*. The protocol 4 binds the transaction with the non-interactive proof of identity by requiring to include the transaction receipt as one of the inputs to the hash function, when creating and verifying the identity proof (step 3 and 9 of protocol 4). Protocol 4 lists all the steps from the transaction receipt transmission by the SP to the user, up to linking the transaction with the user's account by the IDP. This protocol provides a higher level of assurance of the authenticity of the transaction (i.e: assurance that the legitimate owner of the identity has approved the transaction.),

compared to the other two identity proof protocols in which the identity verification and the actual transaction happen in two different steps. This type of protocol for proof of identity is especially useful in scenarios where the IDP has to perform some critical operation on the user's account without having direct interaction with the user, such as debiting money from the user's account for a transaction claimed by an on-line merchant through the credit card network, as discussed under the Section II.

Protocol 4 Binding transactions with Proof of Identity

1: SP→U: Sends the receipt of the transaction: I.
2: U: selects $y_j, k_j \in Z_q$, and creates $d_j = g^{y_j} h^{k_j}$ mod p where $j \in \{1, 2, 3\}$.
3: checks the receipt I and creates Hash Output (h) = H(I, Y, T', d_1, d_2, d_3).
4: obtains three challenge values: $e_j \in Z_q : j \in \{1, 2, 3\}$, from h above.
5: creates proofs based on e_j values: $u_j = y_j + e_j x$ mod q and $v_j = k_j + e_j r$ mod q, where $j \in \{1, 2, 3\}$.
6: U→SP: sends M takes the form:
M = IDT, h, T', d_1, d_2, d_3, u_1, v_1, u_2, v_2, u_3, v_3
7: SP: verifies S_{IDP}, D and T on the IDT and T' in M.
8: **if** initial verification is successful: **then**
9: runs steps 8-13 of protocol 3 to verify the ZKPK made against the challenges obtained from the hash (h).
10: **if** ZKPK verifications passed **then**
11: SP→IDP: forwards I, M and W.
12: IDP: runs the same steps: 7-9 above.
13: **if** ZKPK verifications passed **then**
14: - decrypts W and verifies the name/pseudonym associated with the user's account at IDP.
15: - links the transaction with the user's account.
16: **end if**
17: **end if**
18: **end if**

To conclude this section we would like emphasize that the policy framework and the suite of protocols provided by RahasNym coherently provide means for both users and SPs to carry out pseudonymous identity verification by negotiating their preferred assurance level w.r.t the key requirements of a pseudonymous identity management system, which were listed at the beginning of this section.

IV. Implementation and Performance

In what follows we discuss the architecture of RahasNym and the performance based on its prototype implementation. The prototype implements the complete pseudonymous identity management system that simulates the on-line purchasing scenario discussed in Section II. We also implemented a cryptographic library which underlies the key components of RahasNym. It implements the three versions of ZKPK protocols based on Pedersen Commitment, with Java 7 v1.7.0_25 SDK and with security parameters p = 1024 and q = 160 bits.

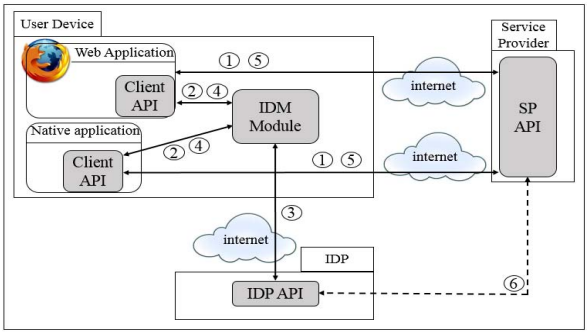

Fig. 3. Deployment architecture of RahasNym.

A. Architecture Overview

Figure 3 shows an high level overview of the architecture of RahasNym. The shaded areas represent the main components of RahasNym that are distributed across three main parties involved in the identity management system. The user agent (i.e: the software in the user's device) consists of two different RahasNym components: 1) The client application - that interacts with the SP and 2) The identity management module (IDMM) - that obtains IDTs from IDPs and creates proofs of identity on those IDTs. The client application that interacts with the SP can either be a web based application running in the user's web browser or a native client application installed in the user's device. The IDMM is a trusted application that runs as a background service in the user's device. The IDP and SP are server side applications.

The sequence of interactions between different entities (see Figure 3) is as follows: (1) The client application obtains the identity verification policy from the corresponding SP. (2) The client application hands over the SP's policy to the IDMM. (3) The IDMM compiles the policy agreed by both parties, by combining the SP's and the user's policies. It then contacts the corresponding IDP(s) to obtain the required IDT(s) which is(are) then handed over to the client application. (4),(5) The client application acts as a mediator between the SP and the IDMM during identity verification, by forwarding the challenges sent by the SP to the IDMM and forwarding the identity proofs provided by the IDMM to the SP. The intended transaction is completed after successful verification of identity. (6) The SP uses the proof(s) of identity if there are any subsequent activities to be performed with the corresponding IDP(s) related to the transaction.

B. Performance Analysis

The IDT request protocol and one of the identity proof protocols are executed during the identity verification phase of a transaction. Since the protocols involve expensive cryptographic operations, such as modular exponentiations in creating commitments, we carried out experiments to measure the average execution time of all the major steps of the protocols executed at different entities of the identity management system as well as the average time taken for end-to-end execution

TABLE III
EXECUTION TIME AND COMMUNICATION SIZE OF IDENTITY PROOF PROTOCOLS.

Steps \Protocol	Protocol 2	Protocol 3	Protocol 4
Creating helper commitment(s): $d(y,k)$=$g^y h^k modp$	1.24 (ms)	3.683 (ms)	3.679 (ms)
Challenge creation	0.018 (ms)	0.051 (ms)	0.696 (ms)
Proof creation	1.548 (ms)	1.532 (ms)	1.579 (ms)
Proof Verification	1.786 (ms)	5.231 (ms)	5.873 (ms)
Total execution	84.257(ms)	82.148(ms)	148.48(ms)
Comm. size	3.5 KB	3.5 KB	3.6 KB

of each protocol. The experiments were conducted in three machines: the client and the IDMM were deployed in a laptop machine with the Ubuntu 13.4 OS, Intel Core i7-3537U CPU, and 5 GB memory, and the RESTful web applications of IDP and SP were deployed in apache-tomcat-7.0.42 servers running in two other machines with Linux 3.15.10 based OS, Intel(R) Core(TM) i7-2600 CPU and 8 GB memory. They were connected through a wired network connection with download speed of 94.97 Mbps and upload speed of 21.41 Mbps and latency of 32ms.

1) IDT Request Protocol: The average execution time is: 2.945317 milli seconds (ms) which accounts for both IDT creation time and the network delay. Two significant steps in IDT creation are: 1) derivation of the secrets x and $r \in Z_q$ from the identity information and from the user's password respectively, 2) creation of the Pedersen commitment from those secrets. Average time taken by these two steps are: 1.760572 ms and 1.184745 ms respectively. Size of a basic IDT acquired from the core Protocol 1 is: 3.5 KB. In addition to the user identity information, the IDT includes PPC_Params which are required for proof creation by the user and proof verification by the SP, and C_{IDP} which is required for the signature verification by the SP.

2) Identity Proof Protocols: As shown in Table III, we first measured the execution times w.r.t the set of common steps involved in all three identity proof protocols. The execution time of the helper commitment creation in protocols 3 and 4 is three times that of protocol 2, which is expected because the non-interactive zero knowledge proof creation in protocols 3 and 4 requires three helper commitments where as protocol 2 needs only one such commitment to create the proof for the challenge sent by the verifier. Challenge (e) creation takes the longest execution time in protocol 4 because it includes the transaction receipt also in the input to the hash function. For the the experiments on protocol 4, we used transaction receipts of 100 KB in size. Proof creation does not show a significant difference across the three different protocols as it only involves computing values: u_j (= y_j + $e_j x$ mod q) and v_j (= k_j + $e_j r$ mod q). The execution times reported w.r.t proof verification reflect its increasing complexity in the three identity proof protocols. In addition to the time taken for the aforementioned key steps, the totoal execution times (i.e: end-to-end protocol execution times) also account for the network delay, policy request time, JSON message encoding and decoding time, policy combining time and IDT request time. Total execution times of protocol 2 and 3 are roughly equal because although protocol 2 is less computationally expensive, it involves more communication steps than protocol 3, since protocol 2 involves an interactive proof. Total execution time of protocol 4 is higher than that of other two protocols as it involves binding of the transaction receipt with the ZKPK identity verification. Communication sizes involved with the three protocols are roughly equal since protocol 2 involves more communication steps than other two protocols although its identity proof includes less information than other two protocols.

In summary, since the execution times of all three protocols are in the orders of milli-seconds, they will not add a significant overhead to the overall latency of the transaction. According to the performance report at [13], the average total execution times of on-line transactions range from 5 seconds to 20 seconds at different on-line merchants.

V. SECURITY ANALYSIS

In this section we discuss how RahasNym addresses the identified key requirements of a pseudonymous identity management system which aim to protect user's privacy and security and SP security, in online transaction systems.

Confidentiality: Identity information is cryptographically encoded using the Pedersen commitment [11] scheme whose security is based on the hardness of computing discrete logarithm. The Pedersen commitment ($Y(x,r) = g^x h^r modp$) has two properties: unconditionally hiding - every possible value of x is equally likely to be committed in Y, and computationally binding - one cannot open the commitment with any $x' \neq x$, unless he can compute $log_g h$. Therefore, any external party, including the SP, who gets hold of an IDT or inspects a transcript of any protocol execution learns nothing about the identity values of its genuine owner. This mitigates the threat of identity theft present in online transaction systems today, in which users have to reveal the identity information in plain text to the third party SPs, even though the communication happens through the channels protected with transport level security. Using standard public key encryption schemes (such as RSA) to cryptographically encode the identity information in the IDT does not solve the problem because they do not provide means for proving the knowledge of identity information in zero knowledge during identity verification, which is essential in addressing ownership assurance requirement, which is discussed next.

Ownership Assurance: All three protocols defined for proof of identity, guarantee that any party without the knowledge of the secrets hidden in the IDT cannot successfully execute the identity proof protocols. In the interactive ZKPK protocol, the random challenge sent by the verifier prevents an attacker from crafting seemingly valid fake proofs. In the non-interactive ZKPK protocols, one-wayness of the cryptographic hash functions prevents an attacker from crafting challenges and seemingly valid fake proofs. Identity proof protocols mitigate the threat of impersonation attacks that is present

in the online transaction systems today, in which the users' identity information is stored in identity repositories of various third party SPs which can be stolen by attackers as well as by malicious SPs themselves, to impersonate the user. RahasNym provides protection against the man-in-the-middle impersonation attacks, that could be carried out by malicious or compromised SPs (known as mafia attacks [14]) during the execution of interactive ZKPK protocols, by explicitly binding the pseudonym of the user and the identity of the SP to the IDT. The timestamps (T') used in the non-interactive ZKPK protocols protect against replay attacks by external parties.

Authenticity: RahasNym provides assurance on the authenticity of the transaction by binding the transaction receipt with the zero-knowledge based identity verification in protocol 4. In current transaction systems, identity verification and the actual transaction happen as two separate steps allowing room for session hijacking attacks. In today's credit card transactions, there is no strong form of verification of the user's approval of the current transaction, except for requiring the expiration date, the CCV code etc, which are readily available on a stolen credit card. One might argue that it is undesirable for the IDP to see the transaction receipt in the perspective of level 3 unlinkability. However, this anyway happens in current systems and the advantage of protocol 4 over these systems is that the transaction can be carried out without revealing the identity information such as credit card information to the third party SP while providing a stronger assurance on the authenticity of the transaction to all the parties. Furthermore, only the information required for the IDP to complete the transaction needs to be shown in the receipt sent to the IDP. This is an instance of a *trade-off* between authenticity and unlinkability.

Unlinkability: RahasNym supports unlinkability at level 1 and level 2 by using IDTs which cryptographically encode user's identity and which are bound to different pseudonyms (see Section III-A for different levels of unlinkability). We assume IDPs to be honest but curious parties and thereby RahasNym supports unlinkability at level 3 with the trust assumption placed on IDPs of not colluding with SPs to help them link different transactions of the same user. This is a reasonable assumption because an IDP is already trusted to vouch for the validity of the user's credential based on which the SPs authenticate the users and to protect the user's assets held at the IDPs such as user's money held at the bank, user's confidential emails held at the email providers, etc. It is important to note that we address unlinkability only at the application level. Concerns about linkability at the network level, such as through traffic analysis attacks etc., must be addressed by the use of network anonymizers such as Tor [15].

Accountability: The IDPs keep track of the issued IDTs in order to provide accountability. The same underlying mechanism used to address accountability requirement also enables SPs to complete certain user generated transactions by forwarding the IDTs and associated proofs to the IDPs, such as claiming the payment from the user's credit card, forwarding the notification emails and delivering the packages, as discussed in Section II. The IDPs link the issued IDTs with the user's account using the W element included in the IDTs (see protocol 1). This does not compromise unlinkability as the encryption mechanism used to create W is randomized and it does not compromise confidentiality of the user's pseudonym(s) held at the IDP(s) as the encryption mechanism is CCA-secure. Legal concerns related to the de-anonymization process is out of the scope of our work. Examples of such concerns are: under what circumstances a pseudonumous user is revealed and what the result of such disclosure is, etc.

Non-Shareability: RahasNym provides assurance that legitimate users are discouraged (by default) or prevented (optionally) from deliberately sharing their pseudonymous credentials with others, to exploit certain services provided by SPs (see Section III-A for different levels of non-shareability). By default, a secret (r) derived from user's password is bound to the identity commitment in the IDT (see protocol 1) which forces the user to share such secrets if he/she attempts to share the IDTs. Optionally, if the policy element *Subject-Verification* includes *biometrics bound*, user's biometric identity is encoded in an identity commitment similar to Y in protocol 1, and it is attached to the pseudonymous IDT. During the identity verification carried out using such IDT, the ownership of the biometrics identity should also be proved. Since the biometrics is tightly coupled with each individual, the non-legitimate holder of a shared IDT can not use it for successful identity verification. RahasNym employs the privacy preserving biometrics based authentication protocol proposed in [16], to achieve non-shareability at level 2, as further described in the appendix.

Table IV summarizes the mechanisms used in RahasNym to address the key requirements identified in Section III-A which are mainly related to user privacy and security and SP security.

VI. Related Work

The very first approach for anonymous credentials, proposed by D. Chaum [7], used blind signatures to achieve unlinkability. It does not provide accountability due to providing unconditional unlinkability, and also does not provide non-shareability due to providing transferability. Chaum's approach uses a separate signature key for each credential value, which limits scalability in supporting different credential types and values. Payment transactions are carried out in [7] by first obtaining a signature on a blind slip from the bank indicating its credit value and then presenting it to the merchant. This approach does not easily extend to support credit card based transactions.

Idemix [5], [17] is one of the current state-of-the-art anonymous credential systems. It provides unconditional (i.e: level 3) unlinkability. It uses Camenisch-Lysyankaya signature scheme [18] for attesting anonymous credentials which allows one to prove the knowledge of a signature in zero-knowledge fashion. It has the advantage that a single credential can be shown an unlimited number of times to different

TABLE IV
SUMMARY OF THE MECHANISMS ADDRESSING THE KEY REQUIREMENTS IN RAHASNYM

Requirements	Enabling Mechanisms
Ownership assurance	-Pseudonym bound IDTs. -ZKP based identity verification. -Biometric identity attached to IDTs. -SP identity bound to IDTs.
Unlinkability	-Multiple pseudonyms for unlinkability at level 1, 2. -Unlinkability at level 3 based on limited trust on IDPs.
Confidentiality	-Encoding identity in Pedersen commitments. -Encoding pseudonyms in cryptographic hash based commitments.
Accountability	-IDPs keep track of all the issued IDTs. -In case of a fraud, de-anonymizing the user with the participation of the IDP(s) who have issued the IDTs.
Non-shareability	-Discouraging shared IDTs by causing the user to share the secrets associated with the IDTs. -Preventing shared IDTs by requiring to attach biometric identity and verifying it.
Authenticity	-Non-interactive ZKPK with signature binds the transaction with the identity verification.

organizations while preserving unlinkability. However, such advantages come at the cost of increased complexity in the IDT, the proof protocols and the architecture of the entire system [10]. Since even the IDP can not link the identity proofs with the original IDT, based on unconditional unlinkability condition, the IDT should carry information and proofs about de-anonymization and revocation, which the SP(s) should verify. This makes IDTs to be complex and bulky. In addition to the proof/verification of the knowledge of the signature (which itself is more computationally expensive than the identity proof/verification protocols used in RahasNym), such approach requires additional proof/verification to be performed in order to prove that the parameters in the issuer's (IDP) public key were correctly generated [12]. Such approach also requires additional parties to be involved in the identity management system such as revocation authority and inspection authority which makes it less adoptable in the existing transaction systems which do not involve such entities by default.

In Idemix [5], [17] all the anonymous credentials are bound to a single master secret. The Direct Anonymous Attestation scheme [12], which involves similar cryptographic primitives as in Idemix, provides a stronger assuarance on non-shareability by binding the anonymous credential to the secrets held in the TPM of the user's device. Such schemes, however, do not have concrete mechanisms to attach the user's biometrics identity to the anonymous credential to provide non-shareability at level 2. In terms of flexibility, Idemix [10] enables only the SPs to express requirements related to the IDTs, which is called presentation policy. In contrast, the policy framework of RahasNym enables both users and SPs to express their requirements and to agree upon them.

UProve [9] is another annonymous credential scheme in which ownership assurance is supported with two-factor authentication. UProve also requires an additional entity to facilitate de-anonymization and revocation. With regard to token revocation, UProve requires that either the SP or both the user and the SP contact the revocation authority to create/verify non-revocation proofs of identity which involves complex cryptographic proofs. In contrast, RahasNym takes advantage of the on-line IDPs to incorporate a simple yet efficient strategy for token invalidation. The IDP includes the field: 'expiration time stamp' (D) in the IDT, that is validated by the SP at the IDT presentation time. The expiration time is limited to the span of a typical transaction and a new IDT is issued for every new transaction. This is similar to the concept used in the current identity management standards such as SAML and OpenID which also involve on-line IDPs.

VII. CONCLUSION

In this paper we have presented a comprehensive and flexible pseudonymous identity management system which we believe is a more practical solution than the previously proposed schemes. Here we highlight the advantages of using RahasNym from the perspective of each of the three parties involved. Users get to carry out secure and unlinkable transactions without fears of identity theft and privacy breaches reported in the previously mentioned attacks. Except for the initial overhead of defining the preferences for the identity verification policy elements and providing the password from which the secrets are derived, the whole process of obtaining IDTs and performing IDT verification is transparent to the user, as the user agent in the IDMM takes care of the complexities of the cryptographic protocols.

Despite the fact that the information that SPs obtain through linkability is directly linked to their revenue, it is advantageous for SPs to adopt Rahasnym due to these reasons: 1) SPs do not have to maintain and protect identity repositories of the users which are targets of attacks; 2) the guarantee of properties that are of interest to SPs, such as accountability of the transactions and non-shareability of the credentials; 3) the ability to seamlessly switch between pseudonymous and conventional (non-pseudonymous) modes of transactions by using the appropriate options for the policy elements. Using such ability the SPs can introduce a new level of membership for privacy concerned users, if they are not willing to offer pseudonymous transactions to all the users by default. Despite the cost of maintaining the issued IDTs, it is advantageous for the IDPs to adopt RahasNym as it preserves ownership assurance of the IDTs and authenticity of the transactions which will lower their cost on detecting and investigating fraudulent transactions. Furthermore, supporting secure and unlinkable identity management will attract more privacy concerned users. Since RahasNym has low performance overhead and required almost no architectural changes to existing transaction systems, both SPs and IDPs could easily adopt RahasNym in their existing systems.

REFERENCES

[1] M. Honan, "How Apple and Amazon Security Flaws Led to My Epic Hacking," http://www.wired.com/gadgetlab/2012/08/apple-amazon-mat-honan-hacking/all/.

[2] C. Schnorr, "Efficient signature generation for smart cards." in *Journal of Cryptography*, 1991.

[3] S. Kelly, "EBay's Massive Security Breach: What It Means for You," http://mashable.com/2014/05/21/ebay-breach-ramifications/.

[4] M. Riley *et al.*, "Missed alarms and 40 million stolen credit card numbers: How target blew it," http://www.businessweek.com/articles/2014-03-13/target-missed-alarms-in-epic-hack-of-credit-card-data.

[5] J. Camenisch and A. Lysyanskaya, "An Efficient System for Non-Transferable Anonymous Credentials with Optional Anonymity Revocation," in *Proceedings of EUROCRYPT '01*, pp. 93–118.

[6] D. Chaum, "Group signatures," in *Proceedings of EUROCRYPT '91*, pp. 257–265.

[7] ——, "Security without Identification: Transaction Systems to Make Big Brother Obsolete," in *Communications of the ACM*, vol. 28. ACM, October 1985, pp. 1030–1044.

[8] A. Lysyanskaya *et al.*, "Pseudonym Systems," in *Proceedings of Sixth Workshop Selected Areas in Cryptography*, 1999, pp. 184–199.

[9] C. Paquin, "Privacy and accountability in identity systems: the best of both worlds," http://research.microsoft.com/pubs/200815/Privacy and accountability - best of both worlds.pdf.

[10] J. Camenisch *et al.*, "Concepts and languages for privacy-preserving attribute-based authentication." in *Journal of Information Security and Applications*, 2014, pp. 25–44.

[11] T. P. Pedersen, "Non-interactive and information-theoretic secure verifiable secret sharing," in *Proceedings of CRYPTO'91*, 1992.

[12] E. Brickell, J. Camenisch, and L. Chen, "Direct anonymous attestation," in *ACM Conference on Computer and Communications Security*, 2004.

[13] E. Commerce Times, "Online retail transaction performance indices." http://www.ecommercetimes.com/web-performance/.

[14] T. Beth and Y. Desmedt, "Identification Tokens - or: Solving The Chess Grandmaster Problem," in *Proceedings of CRYPTO'90*, pp. 169–176.

[15] "Tor project: Overview," https://www.torproject.org/about/overview.

[16] H. Gunasinghe and E. Bertino, "Privacy Preserving Biometrics-Based and User Centric Authentication Protocol," in *Network and System Security - 8th International Conference, NSS 2014*, 2014, pp. 15–17.

[17] J. Camenisch and E. Herreweghen, "Design and Implementation of the Idemix Anonymous Credential System," in *Proc. Ninth ACM Conf. Computer and Comm. Security*, 2002, pp. 21–30.

[18] J. Camenisch and A. Lysyanskaya, "A signature scheme with efficient protocols." in *Security in Communication Networks*, 2002.

APPENDIX

The policy specification of the on-line merchant (see Figure 4) has two rules: rule-1 applies when the user purchases an item as a guest, whereas rule-2 applies when the user signs up with the on-line merchant by creating an account and later logs into that account. Rule-1 has two policy sets each of which applies to different identity attributes. The first policy set of rule 1 applies to email and shipping addresses; it enforces non-interactive ZKPK protocol for proof of identity and accepts the IDT bound to the pseudonym of the user in the 'From' field of the token and to the on-line merchant's identity in the 'To' filed of the IDT. The second policy set of rule-1 applies to the IDTs associated with the user's credit card number. It specifies that the proof of identity should be carried out through non-interactive ZKPK with signature protocol, the IDT should be bound to the user's pseudonym in hidden format and to the biometric identity of the user, and the pseudonym cardinality can be either multiple or single. Rule-2 applies to sign-up and login operations which requires an email address IDT under a single pseudonym since the on-line merchant needs


```
{
    "policySpec": {
        "specifierName": "amazon.com",
        "specifierId": "c225b4e0-eb57-11e3-ac10-0800200c9a66",
        "rule": [
            {
                "id": "rule-1",
                "target": {
                    "operations": ["purchase_as_guest"]
                },
                "policySet": [
                    {
                        "appliesTo": ["email","shipping_address"],
                        "subjectVerification": ["PSEUDONYM_BOUND_&_SP_BOUND"],
                        "proofOfIdentity": ["ZKP_NI"],
                        "pseudonymCardinality": ["MULTIPLE"]
                    },
                    {
                        "appliesTo": ["CCN"],
                        "proofOfIdentity": ["ZKP_NI_S"],
                        "subjectVerification": ["HIDDEN_PSEUDONYM_BOUND_&_BIOMETRIC_BOUND"],
                        "pseudonymCardinality": ["MULTIPLE","SINGLE"]
                    }
                ]
            },
            {
                "id": "rule-2",
                "target": {
                    "operations": ["sign_up","login"]
                },
                "policySet": [{
                    "appliesTo": ["email"],
                    "proofOfIdentity": ["ZKP_NI"],
                    "pseudonymCardinality": ["SINGLE"]
                }]
            }
        ]
    }
}
```


Fig. 4. Example Policy Specification of On-line Merchant.


```
{
    "policySpec": {
        "specifierName": "HTG",
        "rule": [{
            "id": "client_rule",
            "target":
            { "serviceProviders": ["all"],
              "overridingAlgorithm":"SP_OVERRIDES"
            },
            "policySet": [{
                "appliesTo": ["email"],
                "subjectVerification": ["HIDDEN_PSEUDONYM_BOUND_&_SP_BOUND"],
                "proofOfIdentity": ["ZKP_I"],
                "pseudonymCardinality": ["MULTIPLE"]
            }]
        }]
    }
}
```


Fig. 5. Example Policy Specification of User.

to associate the transactions that the user performs, with the account created by the user.

The policy specification of the user (see Figure 5) illustrates the default policy set that is applied to one of the user's identity attributes which is email address and it is applied when carrying out transactions with all the SPs. The user can be prompted by the software in the client device to customize this default policy specification for different SPs and for different identity attributes as and when the user performs transactions. In this way, user's policy will get refined over time, based on the user's preferences and trust with different SPs.

In what follows we list the extensions of the core protocols (defined in Section III-C) which correspond to the different options taken by the policy elements other than the basic options.

(1) If *Subject Verification* = *hidden-pseudonym-bound* AND

hidden-SP-bound, *Pseudonym Cardinality* = *Multiple*, the user agent sends f and ID_{SP} in hidden format by creating commitments on them using a cryptographic hash function: e.g: $H(f|s)$ and $H(ID_{SP}|s')$, in step 1 of Protocol 1. This protects the confidentiality of such information from the IDP and enables the user to prove such commitments only to the SP by revealing the associated secrets (s and s'). In this case, the user agent sends the secrets encrypted with the SP's public key along with other information at the beginning of each identity proof protocol (i.e: protocols 2, 3 and 4).

(2) If *Pseudonym Cardinality* = *single*, the user has to reveal ID_{SP} to the IDP, in order for the IDP to certify that the user has obtained IDTs only with a single pseudonym w.r.t the particular SP. This is an instance of a *trade-off* between unlinkability and accountability.

(3) If *Subject Verification* includes *biometric-bound*, the user agent requests the IDP to include in the IDT, the user's biometric identity commitment (B) by sending the optional argument: *biometric_id_required*, in step 1 of protocol 1. In RahasNym, we adopt the privacy preserving biometrics-based identity enrollment and verification mechanism proposed in [16]. User's biometric identifier (bid) is encoded in the biometric identity commitment using the Pedersen Commitment scheme: $B(bid, r')$ = $g^{bid}h^r modp$. Please refer [16] for details on how bid is derived from a user's biometric features. During the identity enrollment time at the IDP, user's biometrics is captured, features are extracted, bid is derived and B is created. After B is created, other sensitive information about the user's biometric identity is discarded and B is recorded by the IDP. When B is attached to the IDT, the user agent carries out the proof of ownership of the user's biometric identity as well during the verification of the IDT. This is done through the zero-knowledge biometrics-based identity verification protocol that has been proposed in [16] which is compatible with the suite of identity proof protocols in RahasNym.

In what follows we present two examples of how the policy combining algorithm defined in the policy framework of RahasNym resolves conflicts that are found during policy combining and helps negotiating preferences when multiple options are given as the policy element values.

A. Conflict Resolution

Assume that the *Pseudonym Cardinality* policy element in the policy set of the SP's policy, which is defined for the verification of a particular identity attribute, takes the value *Single*, and that the same policy element in the corresponding policy set of the user's policy takes the value *Multiple*. This is a conflict which will be resolved by the policy combining algorithm based on the conflict resolution mechanism specified by the user.

If the user has specified *SP Overrides*, pertaining to the particular SP that the user is interacting with (i.e: in the *target* element defined for the particular SP), the policy combining algorithm will include the value *Single* for the policy element *Pseudonym Cardinality* in the corresponding policy set of the combined policy. Then the IDMM will request an IDT from the corresponding IDP, by sending the optional parameter: *single_pseudonym_certification_required* in the first step of protocol 1, along with other details, including ID_{SP} in plain text. Then the IDP will check if the user has previously obtained any IDT to be presented for that particular SP, in which case the IDP will include in the issuing IDT, the same pseudonym used in such previous IDTs. Otherwise, the IDP will attach a new pseudonym (possibly sent by the user) to the issuing IDT and records it. The IDP will also include the certification: *single_pseudonym_certified* in the IDT created in the step 6 of protocol 1. Once the IDT is received at the SP, it will check for this certification made by the IDP to make sure that the IDT adheres the SP's policy.

On the other hand, if the user has specified *Client Overrides* as the conflict resolution mechanism pertaining to the particular SP that the user is interacting with, the execution will be paused and the user will be informed about the conflict (if it has not been done before) and will be given the option to edit the policy if needed. Accordingly, the transaction will proceed or abort based on the user's response.

Although we have used technical terms for the definition of the policy language, it can be made more user friendly when it is implemented in the real world.

B. Negotiation of Policy Options

When the user and/or the SP have specified more than one option for the three policy elements defined in Section III-B, the combining algorithm will pick the option that enforces the highest level of assuarance w.r.t the corresponding requirement(s) of pseudonymous identity management. For example, if the *Proof of Identity* policy element in the user's policy specifies the options: *ZKP-NI, ZKP-NI-S* and the SP's policy specifies the options: *ZKP-I, ZKP-NI, ZKP-NI-S*, the policy combining algorithm will select: *ZKP-NI-S* as the option to be included in the combined policy, as it is included in both the policies and provides stronger authenticity assurance, as discussed in Section III-C.