RECEIPT MANAGEMENT- TRANSACTION HISTORY BASED TRUST ESTABLISHMENT

by Abhilasha Bhargav-Spantzel, Jungha Woo, Elisa Bertino

Center for Education and Research in
Information Assurance and Security,
Purdue University, West Lafayette, IN 47907-2086

Receipt Management- Transaction History based Trust Establishment

Abhilasha Bhargav-Spantzel
Purdue University
West Lafayette, IN
bhargav@cs.purdue.edu

Jungha Woo
Purdue University
West Lafayette, IN
wooj@cs.purdue.edu

Elisa Bertino
Purdue University
West Lafayette, IN
bertino@cs.purdue.edu

ABSTRACT

In a history-based trust-management system, users and service providers use information about past transactions to make trust-based decisions concerning current transactions. One category of such systems is represented by the reputation systems. However, despite the growing body of experience in building reputation systems, there are several limitations on how they are typically implemented. They often rely on scores that are evaluated by service providers and are often not reliable or well understood. We believe that reputation has to be based on objective and reliable information. In such context, transaction histories play an important role. In this paper, we present the VeryIDX system that implements an electronic receipt infrastructure and supports protocols to build and manage online transaction history of users. The receipt protocols are shown to have several essential security and privacy properties. We present a basic yet reasonably expressive language which provides service providers with a new way to establish trust based on users' transaction history. We also describe the architecture and prototype implementation of VeryIDX, based on several important design considerations of a real-world e-commerce system infrastructure.

1. INTRODUCTION

With the advent of e-commerce applications, there are increasing requirements of establishing mutual trust between users and end service providers (SP's for brevity). SP's rely extensively on authentication to attain trust, such as Single Sign-On (SSO) services. More recently, transaction histories have gained importance for implementing advanced services such as *reputation systems* [13, 10]. Several e-commerce SP's have built reputation systems so as to give a better idea of how trustworthy both the buyers and the sellers are. This is because the sellers are typically SP's but could also be users in a peer to peer (P2P) environment. Sellers benefit from the use of such systems because good reputation score is likely to attract more customers. Similarly buyers may qualify for better deals and services if they have good reputation. However, most reputation systems have a major limitation in that the only information they maintain are scores and they do not typically provide information about the actual transactions a seller or buyer has made. We believe that it is important that trust be established also in the basis of the transaction history. Information about the history can be consulted to evaluate and manage the potential risks in a given transaction. Capturing and using transaction history for trust establishment entail addressing several challenges. In e-commerce applications, transaction history should include a customer's profile of transactions with several SP's and a SP's profile of transactions with several customers. Such transaction history needs to be accessed by various SP's, which may use heterogenous transaction history formats. In some existing real world scenarios the SP's store transaction history in such a way that makes impossible for other SP's to use it. Therefore the user cannot benefit from its past transactions. Moreover, there is a lack of user control on his/her transaction history. The transaction history is generally stored at the SP end, and the user may not be able to control who accesses this information. One solution is to introduce a third party receipt management server. To this extent, we propose the VeryIDX electronic receipt infrastructure and protocols to build and manage online transaction history of users. With such system, SP's can have access to the user's transaction history according to the users' permissions.

There are several desired properties for such a transaction history management system. First is the **sharing prevention** for receipts. If a receipt R_{U_A} is issued to a user U_A, then U_A should not be able to provide it to another user U_B who could then present R_{U_A} as its own receipt. Second is the **availability** of receipts. If the transaction history is saved as cookies locally at the client machine, portability and hence the availability of such receipts is hard to achieve. The VeryIDX infrastucture is therefore based on an identity management system which makes the receipt information available to the online users. Third is the **minimal disclosure** of the information stored in the receipts, to minimize the information revealed about the users transactions at the various SP's. Fourth is the **user choice**; the user should be able to select parts of a receipt based on the information needed to carry on the current transactions. Fifth is **integrity** of user's history data. Integrity should be maintained to enable high assurance trust establishment and reputation evaluation. From the architectural perspective, a sixth desired property is that the system should be **easy to deploy** in current e-commerce systems with minimal extensions to the existing systems. The management overhead imposed on human users should be as low as possible so to assure **usability**. The final property is that the system should support **interoperability**, in that it should be possible to use the transaction history from one SP at another SP.
The goal of our work is to develop protocols for managing transaction histories that verify the above properties.

Among our key innovations is a series of protocols for the establishment and management of users' transaction history. These receipt protocols satisfy strong security requirements namely 1) correctness, 2) integrity, 3) single submission, 4) fairness and 5) non-repudiation. To achieve such properties several cryptographic tools like zero-knowledge proofs, identity-based signatures, contract signing and certified email protocols are used in the receipt protocols. All receipt protocols are privacy-preserving with respect to user consent and minimal disclosure. We provide a standard yet extensible format of e-receipts which are used in these protocols.

We also define a simple policy language to allow SP's to specify what kind of information is needed from the receipts. We present the architecture and design of the VeryIDX system taking into account several important considerations of a real-world e-commerce system infrastructure. We also describe a prototype implementation of the VeryIDX system with detailed performance analysis.

The rest of the paper is organized as follows. Section 2 provides an overview of the approach and the key functionalities of the system with security and privacy criteria that the receipt protocols need to satisfy. Section 3 introduces the proposed protocols followed by a protocol analysis in Section 4. Section 5 describes the VeryIDX system implementation and presents the system analysis. Section 6 discusses related work. Section 7 concludes the paper and highlights future work directions. The paper includes some appendices giving additional details about the protocols and implementation.

2. OVERVIEW OF THE APPROACH

Our approach is based on an extended notion of federation [2]. The federation is composed of the following entities: identity providers (IdP's), service providers (SP's), registrars and users. SP's provide services to users as in conventional e-commerce and other federated environments. IdP's issue certified attributes to users and control the sharing of such information. The *registrars* store and manage information related to users' strong attributes. Strong identity attributes uniquely identify a user, as opposed to weak identity attributes which may be common to a group of users. The information recorded at the registrar is used to perform multi-factor identity verification of users. Note that, unlike the IdP's, the information stored at the registrar does not disclose the values of the strong user attributes in clear. Instead, this information contains the cryptographic semantically secure *commitments* [5] of the strong attributes which are then used by the users to construct zero knowledge proofs of knowledge (ZKPK) [9] of those attributes.

In our approach to transaction history management, the registrar manages user's receipts and provide them to SP's when needed. All receipts are stored in the user Receipt Record (RREC for brevity) which is created for each registered user. An e-receipt has 9 key elements, namely– TRANSACTION ID, SELLER, BUYER, ITEM, ITEM DESCRIPTION, PRICE, USER INFO, ASSURANCE LEVEL and TIME. The TRANSACTION ID and SELLER form a key to uniquely identify the receipt. Most of the items in the receipt correspond to those of traditional receipts except USER INFO and ASSURANCE. USER INFO captures only the weak attributes collected about the user during the e-transaction. This information is used to assess the ASSURANCE LEVEL that the given receipt belongs to the user claiming a given RREC. If the combination of the weak attributes uniquely identifies the user, then the assurance level of the receipt is set to 'A'. Depending on the amount of information available about the user, the assurance level could be set to 'B' or 'U' for unknown. Assurance level will be lower if conflicts are identified. For example, if the citizenship of the user in two different receipts is different, then there is the possibility that the two different e-transactions have been executed by different users. We ensure, by using digital signatures, that the receipts cannot be tampered with once they are issued, even by the registrar storing them.

Function	Purpose
Add Receipt.	Once a user has completed a transaction, it executes the 'add receipt' protocol to retrieve the receipt from the SP and store it at the registrar.
Extend Receipt.	For a receipt that is already stored at the registrar the user can create x-receipts by adding the cryptographic commitment to the original e-receipt.
Use Receipt.	When the user interact with a SP, it can use the receipts to prove properties about its past transactions. Properties required about past transactions are specified by the *trust establishment policies*.
Remove Receipt.	If a receipt is unusable, expired or revoked, then the user, registrar or SP can delete it. Once this receipt is removed from the registrar, no other copy of this receipt stored at any other can be successfully used.

Table 1: Functions Summary

EXAMPLE 1. *An example of a receipt R of user Alice is* ⟨ 401,E-BOOK STORE, ALICESSO, BOOK, *Quantum Mechanics*,$103.27, "AMERICAN, LAFAYETTE-IN, JUGGLER", 'A', 14:34 03/12/2007⟩ *where 401 is the transaction ID,* E-BOOK STORE *is the name of the SP and* ALICESSO *is the Alice's single sign-on (SSO) ID with the registrar where this e-receipt is stored. In this receipt the* ASSURANCE LEVEL *of the* USER INFO *is 'A'.*

To ensure minimal disclosure of the receipt attributes, in that the user can create ZKPK's [9] of its receipt attribute properties, we allow the user to extend the original receipts with Pedersen's commitments [5]. Once the commitments are enrolled at the registrar, they can then be used to create proofs regarding properties of those attributes as elaborated in Section 3. The receipt extension (*x-receipt* for brevity) has the 'Transaction ID' and 'Seller' to uniquely identify the receipt, followed by the element tag, such as 'Price', and the corresponding cryptographic commitment. In Example 1 if Alice enrolls a commitment corresponding to the price, then she can prove that the price is greater than $100[1] without having to reveal the exact price.

Our system provides the functions listed in Table 1 supporting the creation, use and deletion of the receipts. The protocols implementing the functions are described in detail in Section 3. It is important to mention that there are specific *security* and *privacy* requirements for all these protocols. We briefly discuss such requirements in what follows.

Security requirement. Security of the receipt protocols includes five main properties.

1. **Correctness.** It means that if two honest parties successfully complete an e-commerce transaction, then the final receipt is constructed with the correct receipt attributes and is included in the right users' RREC.

2. **Integrity.** It refers to the tamperproofness of the constructed receipt. If any receipt attribute is modified, then it should be possible to detect the change.

3. **Single Submission.** It requires that the same receipt be not submitted more than once as two different receipts.

4. **Fairness.** It requires that the proof-of-delivery from the buyer and the proof-of-origin from the seller are available to the seller and buyer, respectively. Moreover, the protocol must be fail-safe,

[1]There are ZKP's that allow to prove that a committed integer satisfies an inequality, such as a given committed value x is greater than a constant A. A possible approach to accomplish this is using interval proofs [3].

in that the incomplete execution of the protocol must not result in a situation in which the proof-of-delivery is available to the seller but the proof-of-origin is not available to the buyer, or vice versa.
5. **Non-repudiation.** For two-party protocols the non-repudiation property is two-fold [21]: a) non-repudiation of origin, that is, providing the buyer with irrefutable proof that the content received was the same as the one sent by the seller; b) non-repudiation of delivery, that is, providing the seller with irrevocable proof that the content of item or token received by the buyer was the same as the one sent by the seller.
Privacy requirement. The privacy requirement for the receipt protocols consists of two main properties.
1. **User Consent.** It requires that the users be able to consent or agree to terms or conditions that may be associated with the disclosure and use of its receipt attributes. It is important that the user has an opportunity to reject any disclosure of receipt information if required by the SP [14].
2. **Minimal Disclosure.** It requires that only the minimal piece of receipt information, as needed by the SP, is revealed.

3. PROTOCOLS

In this section we present receipt based protocols which enable users to enroll their receipts with registrars, and use them with SP's. More specifically, we provide detailed protocols based on two-party message exchange and cryptographic primitives like identity based signature (IBS) and zero knowledge proof of knowledge (ZKPK). The protocols are summarized in Table 2.

3.1 Preliminary Concepts

Following are the preliminary concepts regarding identity based signatures, commitments and ZKPK, and the corresponding protocol notation.

Pedersen commitments: Let g and h be generators of group G of prime order q. A value m is committed by choosing r randomly from $\mathbb{Z}_q$ and giving commitment $C = g^m h^r$ [5]. Commitment C is opened (or revealed) by disclosing m and r, and the opening is verified by checking that C is indeed equal to $g^m h^r$. A prover can prove by using a zero-knowledge proof that it knows how to open such commitment without revealing either m or r.

ZKPK: In our approach we use the techniques by Camenisch and Stadler [4] for the various ZKPK of discrete logarithms and proofs of the validity of statements about discrete logarithms. We also conform to the same notation [4]. For instance to denote the ZKPK of values α and β such that $y = g^\alpha h^\beta$ holds, and $u \leq \alpha \leq v$, we use the following notation:

$$PK\{(\alpha, \beta) \, : \, y = g^\alpha h^\beta \wedge (u \leq \alpha \leq v)\}$$

The convention is that Greek letters denote quantities the knowledge of which is being proved, whereas all the other parameters are sent to the verifier. Using this notation, we will simply describe each protocol's purpose without elaboration.

Identity Based Signature Scheme: We use the ID-based signature scheme derived from the Schnorr's signature scheme given in [18]. ID-based signature scheme consists of four main protocols, namely $Setup$, $Extract$, $Sign$ and $Verify$.

Public Key Encryption: We assume a public key infrastructure for the registrars and the SP's. Public key encryption (PKE) is used while encrypting the data for a particular SP or registrar, and also when data is signed by these entities.

3.2 Adding Receipts at the Registrar

We define two protocols for adding receipts to a registrar which varies according to the parties involved. The first protocol is applicable when a user has conducted an e-commerce transaction with a SP and wants a receipt. The second one applies when two users want to conduct peer to peer e-commerse transaction without involving an external SP.

Protocol 1: Adding Receipts generated by User-to-SP Transactions.

Steps 1–12 illustrated in Figure 1 are followed by the user to add a receipt, generated by a SP, to its RREC at the registrar. In steps 1–4 the user obtains a random session handle generated by the SP. In step 5 the user conducts strong authentication as follows.
Strong Authentication. The user first logs into the registrar using its user ID and password. To perform *strong authentication* during login, we require that users provide ZKPK of the secrets corresponding to *multiple*, say t, strong attribute commitments stored at the registrar. Only after a user successfully provides a proof, then it is allowed to perform management tasks. The registrar and user then perform the following steps to execute the strong authentication.
First the registrar retrieves this user's t strong identifier commitments of the form $M_i = g^{m_i} h^{r_i}$ where $m_i \in \mathbb{Z}_q$, $1 \leq i \leq t$, is a user's strong identifier and $r_i \in \mathbb{Z}_q$ is the secret random value the user had chosen at the time of enrollment [1]; (g, h, q) are the public parameters of the registrar[2].
Second the user aggregates its secrets. The registrar challenges the user to prove knowledge of t commitments $\{M_i\}$, $1 \leq i \leq t$. The user computes $M = \prod_{i=1}^{t} M_i = g^{m_1+\cdots+m_t} h^{r_1+\cdots+r_t}$ and sends M, and M_i, $1 \leq i \leq t$, back to the verifier. Finally a ZKPK of aggregate commitment is performed. The user and the registrar carry out the following aggregated ZKPK (AgZKP for brevity) protocol:

$$PK\left\{(\alpha, \beta) \colon M = g^\alpha h^\beta, \, \alpha, \beta \in \mathbb{Z}_q\right\}$$

where $\alpha = \{m_1+\cdots+m_t\}$ and $\beta = \{r_1+\cdots+r_t\}$. If the AgZKP is successful then the user is considered strongly authenticated.

Steps 6–11 in Figure 1 illustrate the messages exchanged among the registrar, user and SP to retrieve the receipt. In the final step, before storing the new receipt in the RREC, the registrar calculates the assurance using the procedure described below.
Assurance Assessment. To assess the assurance, the registrar verifies receipt R and compares the USER INFO (W for brevity) in the receipt, with the weak attributes (W_{user}) stored at the registrar which have a high assurance level. These weak attributes can be stored as a part of other receipts in RREC and user information available to the registrar. Based on the overlap of this information the registrar computes the assurance that the user who is registered is the same user who performed the e-commerce transaction. For example if $W \bigcap W_{user} = W$ then there is a complete overlap. If W uniquely identifies the individual, the assurance level would be as high as the lower bound of the assurances of all $w_{user} \in W_{user}$. Note that the higher the number of overlapped attributes, the higher is the assurance level. Once the assurance level is assessed the registrar adds the receipt to the user's RREC.

Protocol 2: Adding Receipts generated by User-to-User Transactions.

Consider a case of two users that carry out an e-commerce transaction directly with each other. A user U_A is selling item I for price P to user U_B. Both users are interested in submitting a receipt of this transaction to the registrar in order to extend their transaction

[2]In [1] it is assumed that the registrar runs generation algorithm GenKey on input 1^k to generate the public parameters: a prime q of length k, three groups G_1, G_2, G_T of order q. Two generators g, h in G_1 are specified such that $\log_g h$ is unknown.

#	Title	Parties	Protocol Goal	Key Challenges [*Techniques used*]
1	Adding receipt (SP-user)	U, SP, REG	User adds the receipt provided by a SP after an e-transaction to its RREC at the registrar.	(a) The user is authenticated correctly [*AgZKP*], (b) Integrity of the receipt [*PKE*], (c) Single submission of receipt [*Session handles*]
2	Adding receipt (user-user)	U, REG	Both buyer and seller are users who perform e-transaction and add their receipts to their RREC's.	(a),(b),(c), (d) Both parties should get their receipts simultaneously [*Contract Signing Protocol*], (e) Non-repudiation [*IBS*]
3	Extending receipt	U, REG	The user creates cryptographic commitments for selected receipt attributes.	(a), (f) The extension is done correctly and on the claimed attribute [*ZKPK*]
4	Providing receipt attributes	U, SP, REG	User provides selected receipt attributes to SP.	(a), (g) Availability of the users' receipts [*online Registrar*] (h) User consent on the released attributes [*Registrar portal UI*], (i) Integrity of the released attributes [*PKE*]
5	Providing receipt attribute proofs	U, SP, REG	User provides proof of knowledge of selected receipt attributes.	(a),(g), (j) minimal disclosure of attribute information [*ZKPK*], (k) Non-repudiation of proof [*IBS and ZKPK*]
6	Revoking a receipt	U, SP, REG	SP invalidates the users' receipt due to the refund of the e-transaction.	(a), (l) The refund of the item and receipt revocation happens simultaneously [*Contract Signing Protocol*], (e) Receipt is removed from RREC [*Semi-trusted registrar*]

Table 2: Receipt Protocols Summary

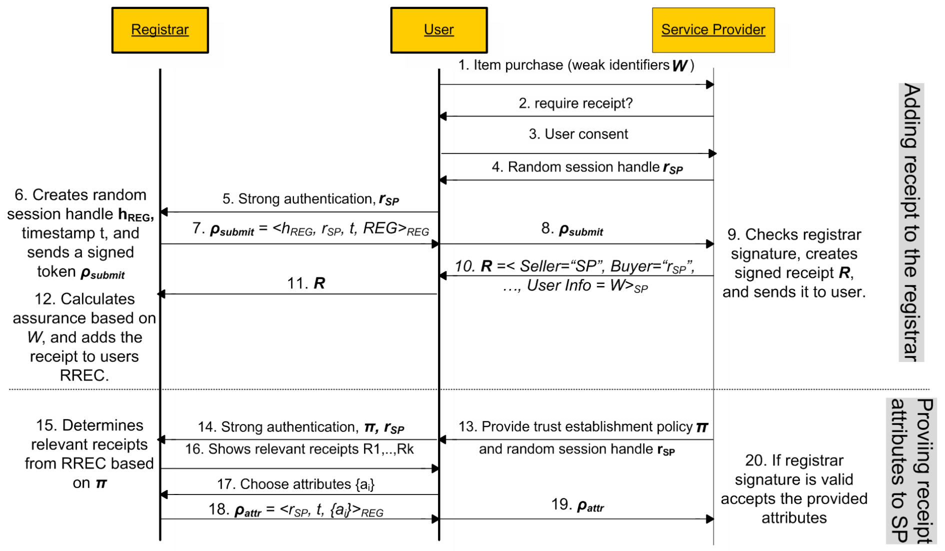

Registrar
User
Service Provider
1. Item purchase (weak identifiers W)
2. require receipt?
3. User consent
4. Random session handle r_{SP}
5. Strong authentication, r_{SP}
6. Creates random session handle h_{REG}, timestamp t, and sends a signed token ρ_{submit}
7. ρ_{submit} = <h_{REG}, r_{SP}, t, REG>$_{REG}$
8. ρ_{submit}
9. Checks registrar signature, creates signed receipt R, and sends it to user.
10. R =< Seller="SP", Buyer="r_{SP}", …, User Info = W>$_{SP}$
11. R
12. Calculates assurance based on W, and adds the receipt to users RREC.
Adding receipt to the registrar
13. Provide trust establishment policy π and random session handle r_{SP}
14. Strong authentication, π, r_{SP}
15. Determines relevant receipts from RREC based on π
16. Shows relevant receipts R1,..,Rk
17. Choose attributes {a_i}
18. ρ_{attr} = <r_{SP}, t, {a_i}>$_{REG}$
19. ρ_{attr}
20. If registrar signature is valid accepts the provided attributes
Proving receipt attributes to SP

Figure 1: Message flow of receipt Protocol 1 and Protocol 4

history as a seller and a buyer respectively. This receipt would have to be constructed with the consent and verification of both U_A and U_B. We assume that users have pre-established accounts at the registrar, in that they have a user name and password corresponding to a RREC. Since U_A and U_B do not trust each other, if such purchase/selling transaction were unsupervised, then it would be difficult to settle any dispute. Therefore the following protocol is carried out to make the purchase, followed by the generation and submission of the receipt.

1. *ID-Based Signature Setup.* Users U_A and U_B execute the IBS Scheme introduced in Section 3.1. Here the public ID's are the SSO ID's of U_A and U_B at the registrar. Note that the key used to sign is only known to the user owning that ID.

2. *Receipt and Context Agreement.* Users U_A and U_B first sign their user ID at the registrar with their private key, using the IBS $Sign$ protocol. Once each signature is verified, by using the $\mathtt{Verify}$ protocol, the seller and buyer names at the registrar are known. Then they need to agree on the details of the purchase involving details of price, item and other such information to construct the potential receipt. They also need to agree to provide a valid signed receipt when the transaction is complete. These terms of agreement are formalized in a *contract* C. To achieve fairness, such contract should be signed simultaneously. Therefore, a *contract signing protocol* [20] is used so that each party has a signed copy of this contract simultaneously.

3. *Purchase.* To make the purchase, user U_B needs to provide its strong attributes like Credit Card Number and weak attributes like Name with Address ($U_B Attr$ for brevity). To do this the following steps are taken [15]

   (a) U_B generates a random key K and sends U_A the encrypted message $E_K(U_B Attr||C||R_{U_B})$ where C is the contract they agreed upon and R_{U_B} is a receipt signed by U_B detailing the purchase.

   (b) U_A publishes a signed message requesting U_B to publish the key for E_K–encrypted message whose digest is $H(E_K(U_B Attr||C||R_{U_B}))$ by date T at location X.

   (c) U_B publishes the pair $H(E_K(U_B Attr||C||R_{U_B}))$, K in X on or before date T.

The above is a certified email protocol which prevents U_A from denying the fact that U_B provided required information for the purchase. At this point U_B's side of the purchase is made. In a similar fashion, U_A provides the resource, contract C, and a signed receipt of purchase R_{U_A} to U_B.

4. *Receipt Addition.* In this step both U_A and U_B have signed receipts. They both log onto the registrar using strong authentication to add their respective receipts as described in Protocol 1.

Protocol 3: Receipt Extension with Commitments. A user can extend receipts stored at its RREC by creating cryptographic commitments of receipt attributes. This is to allow the users to create ZKPK's, in future transactions, based on the commitments. The following protocol is carried out to make such extension.

1. *Login and receipt attribute identification.* The user logs at the registrar to access its RREC. The user identifies a particular receipt using a TRANSACTION ID and name of SELLER pair. Then the user chooses which attribute a_{RREC_i} in this receipt it wishes to commit.

2. *Commitment creation.* Referring to the public parameters described in the strong authentication step of Protocol 1, the user first uses a public function f such that $f(a_{RREC_i}) = m_i$ with $m_i \in \mathbb{Z}_q$. Then the user chooses a value $r_i \in \mathbb{Z}_q$, and computes the final commitment $C_i = g^{m_i} h^{r_i}$. The user also signs this commitment using the $Sign$ protocol of the ID-based signature scheme and provides this to the registrar.

3. *Commitment verification.* The registrar first verifies the signature using the $Verify$ protocol of the ID-based signature scheme [18]. Next the user gives a ZKPK of opening the commitment M to the registrar.

$$PK\{(\beta) \colon C = g^{m_i} h^{\beta},\ \beta \in \mathbb{Z}_q\}$$

where $m_i = f(a_{RREC_i})$ and β is a random secret chosen by the user. The above ZKPK proves to the registrar that the user has constructed a commitment corresponding to the receipt attribute a_{RREC_i}.

4. *Commitment addition.* If the verification was successful, the registrar adds the signed commitment for the specified attribute to the RREC.

Protocol 4: Trust establishment with a SP using e-receipt. Steps 13–20 in Figure 1 show how the user can provide its receipt attributes to the SP to establish trust or a reputation level based on the criteria specified by that SP. This criteria may be specified as policies at the SP.

Trust establishment policies on e-receipts. The policies are specified as conditions on the receipts. We use first order logic formula (FOLF) to reason about the policies. The vocabulary Ψ_{open} contains binary predicates corresponding to receipts and receipt attributes. A complete list of these predicates is provided in Table 3. The SP trust establishment policy Π is a FOLF expressed in terms of Ψ_{open}.

EXAMPLE 2. *A policy of the online-book store 'e-book' could be as follows – 'if a user has bought a book for more than* \$80 *from 'e-book', then it is a trusted customer.' The trust establishment policy can be encoded in the logic as:*
$TrustedCustomer(U) := \exists R_U(\mathbf{Receipt}(R_U) \wedge \mathbf{Buyer}(R_U, U) \wedge \mathbf{Seller}(R_U, 'e-book') \wedge \mathbf{Price}(R_U, R_U.Price, >, 80))$

The SP provides the user with the trust establishment policy Π as illustrated in step 13 of Figure 1, along with the random session handle which is needed to ensure freshness of the transaction. The user then logs on to the registrar and provides this information. The registrar evaluates the policy Π to identify a list of receipts $R_1, \ldots, R_k$ which would satisfy the trust establishment criteria[3]. Once the receipts are identified the registrar provides a way for the user to select the attributes $[a_{RREC_i}]_{R_t}$ from receipt R_t, where $1 \leq t \leq k$, $1 \leq i \leq n$, and n is the total number of attributes needed to satisfy Π. The user is also given an option to add more receipts from its RREC if it desires to do so. The user also provides the random handle r_{SP} to the registrar. Given this information, the registrar constructs the signed attribute token $\rho_{attr} = \langle \{[a_{RREC_i}]_{R_t}\}, r_{SP}, t \rangle_{REG}$ where t is the current timestamp. The registrar sends ρ_{attr} to the user. Finally the user provides ρ_{attr} to the SP. The SP verifies the attributes and provides the service accordingly.

Protocol 5: Trust establishment with a SP using x-receipt. If a user does not want to provide clear attributes from the receipts and instead wants to prove properties of the receipt attributes, it can use the enrolled cryptographic commitments of the x-receipts to create proofs of such properties. The policies for such kind of trust establishment can be expressed as follows.

Trust establishment policies on x-receipts. For the cases in which the trust establishment criteria are related to *cryptographic proofs* of receipt attributes belonging to the user, the SP uses an extension of policy vocabulary Ψ_{open} denoted as Ψ_{proof}. Ψ_{proof} also has binary predicates but unlike Ψ_{open} the attributes specified do not have to be revealed in clear. Instead ZKPK of those receipt attributes need to be provided by the user. For each of the predicates listed in Table 3, there is an equivalent predicate for the Ψ_{proof} vocabulary, pre-pended by the letter 'x'. For example **xSeller, xItem,** and **xPrice**. If in Example 2 the clear attributes are not required, instead the cryptographic proofs of those attributes are sufficient, and then the same trust establishment policy can be written as:
$TrustedCustomer(U) := \exists R_{xU}(\mathbf{xReceipt}(R_{xU}) \wedge \mathbf{xBuyer}(R_{xU}, U) \wedge \mathbf{xSeller}(R_{xU}, 'e-book') \wedge \mathbf{xPrice}(R_{xU}, R_{xU}.Price, >, 80))$
Using such policies, the precise steps of the protocol are described as follows.

Step 1. *SP Policy.* The SP provides the trust establishment policy requiring ZKPK Π_x together with a random handle r_{SP} and sends it to the user.

Step 2. *User retrieves receipt commitments.* It is required that the user has created a commitment for each of the receipt attributes for which it has to construct a ZKPK. Assuming that these commitments are created for each such attribute using Protocol 3, the next step is to retrieve these commitments.

The user logs at the registrar to access its RREC. The registrar then evaluates the policy Π_x to identify a list of receipts which would satisfy the trust establishment criteria based on the attribute information available in clear. Once the receipts are identified the registrar provides a user-interface for the user to add additional receipts if needed. Let the resulting list of selected receipts be $R_1, \ldots, R_k$. The user then selects the attributes $[a_{RREC_i}]_{R_t}$ with the corresponding commitments $[C_{RREC_i}]_{R_t}$ from receipt R_t where $1 \leq t \leq k$, $1 \leq i \leq w$ and w is the number of commitments needed. We simplify the notation of the commitment and represent it

[3]The evaluation is possible only if the RREC has the value of the attributes needed to satisfy Π in clear.

Predicate	Arity	Arguments	Meaning
Receipt	1	receipt R	If **Receipt**(R) is true, then R is a valid receipt belonging to the user who is claiming this receipt.
ReceiptKey	3	receipt R, transaction ID R.TID and seller R.SELLER	If **ReceiptKey** (R, R.T,R.S) is true, then R receipt can be uniquely identified using its transaction ID R.T and seller name R.S.
Seller	2	receipt R and seller information R.SELLER	If **Seller**(R,R.S) is true, then R has the seller name R.S.
Buyer	2	receipt R and buyer information R.BUYER	If **Buyer**(R,R.B) is true, then R has the buyer pseudonym R.B.
Item	2	receipt R and item tag R.ITEM	If **Item**(R,R.I) is true, then R has the item tag R.I.
Price	4	receipt R, price number value R.PRICE, numeric operator, number constant	If **Price**(R,R.P,o,C) is true, then R has the price value R.P which has a relation denoted by operator o (e.g. $=, >, <$) with numeric constant C.
Assurance	2	receipt R and assurance tag R.ASSURANCE	If **Assurance**(R,R.A) is true, then R has the assurance level R.A.
Time	2	receipt R and time R.TIME	If **Time**(R,R.T) is true, then R was issued at time R.T.

Table 3: Predicates for Service Providers Trust Establishment Policies

as $C_1, \ldots, C_w$. The user is also given an option to add more receipts from its RREC if the user desires to do so. The user also provides the random handle r_{SP} to the registrar.
Given this information the registrar constructs the signed commitment token $\rho_{commit} = \langle\{([C_{RREC_i}]_{R_t})\}, r_{SP}, t\rangle_{REG}$ where t is the current time stamp. The registrar sends ρ_{commit} to the user.

Step 3. *Proof submission of user's x-receipt attributes.* The user performs an AgZKP with the SP in order to provide proof of knowledge of the required attributes. Note that only the user knows the random secrets and the actual attribute values corresponding to each of the committed receipt attributes. The proof consists of the following two key steps.
(a) *Users' aggregation.* Consider that the SP has challenged the user to prove knowledge of commitments $\{C_i\}$ where $1 \leq i \leq w$. The user computes
$C = \prod_{i=1}^{w} C_i = g^{a_1+\cdots+a_t} h^{r_1+\cdots+r_t}$, where a_i and r_i are the attribute and secret random corresponding to the commitment c_i respectively. The user sends C, ρ_{commit} to the verifier.
(b) *Zero-knowledge proof of aggregate commitment.* The user and the registrar carry out the following AgZKP protocol:

$$PK\left\{(\alpha,\beta)\colon C = g_1^{\alpha} h_1^{\beta},\ \alpha,\beta \in \mathbb{Z}_q\right\}$$

where $\alpha = \{a_1 + \cdots + a_t\}$ and $\beta = \{r_1 + \cdots + r_t\}$. If the ρ_{commit} is constructed correctly to satisfy Π_x and AgZKP in step 3.2 is successful, then the user proof is considered correct and the trust is established.

Protocol 6: Revoking a receipt

By revocation of a receipt we mean the removal of the receipt from the users' RREC. We consider three cases for the revocation of a receipt depending on the party revoking the receipt, namely the user, the registrar and the SP.
The first two cases are trivial. For the user case, the user is required to log onto its account using strong authentication to access its RREC. Once logged in, our system provides a way for the user to remove any of the receipts from its RREC. For the registrar case, the registrar may define the criteria for removing receipts. For example, the registrar may revoke receipts that are more than 100 days old. The registrar periodically checks the RREC to see if the receipts are compliant to its criteria to retain the receipts. If not, the registrar asks the user to remove the specific receipts within a given time period, after which the receipt is removed by the registrar itself.

For the SP revocation case we consider an interesting scenario where the user needs to return the purchased item from an SP and the SP may provide a refund. More importantly this SP needs to ensure that the receipt stored from the previous transaction gets void and the user needs to ensure that it gets the refund. The following steps are needed to do this revocation.

1. *User retrieves receipt from its registrar.* The user logs on to access its RREC. It identifies the receipt R which is to be revoked once it returns the item relevant to the purchase identified in the receipt. The registrar then constructs the signed token $\rho_{revoke} = \langle R, U, t\rangle_{REG}$. It sends ρ_{revoke} to the user where U is the SSO ID of the user, and t is a timestamp to ensure freshness.

2. *User requests revocation from SP.* The user then signs the $\langle\rho_{revoke}\rangle_U$ using IBS $Sign$ protocol and provides this token to the SP. The SP verifies the signature using the public key of U and the IBS $Verify$ protocol. Only if the verification is successful, the revocation protocol proceeds.

3. *Users' refund and SP's revocation.* The user and SP agree on a contract C using simultaneous contract signing protocol, which would state that for the transaction identified by the receipt R in ρ_{revoke} the user will provide the SP the identifier i of the purchased item, and the SP will provide the refund f applicable to that purchase. Note that the item identifier should not be a sensitive value, but instead a public service number for the item purchased. For example i could be a pin. Once the pin is revoked, no other user can use it to access the same resource. Once this contract is agreed upon, the three steps as in Protocol 5 Step 3 are executed with message $E_K(i||C)$ where C is the refund contract they agreed upon. Using these steps the user can request revocation and the SP cannot deny the user did revoke its purchase. Then the SP sends the user a token $\alpha = [f, \rho_{revoke}]_{REG}$ encrypted with the registrar's public key. The user sends α to registrar. The registrar removes the receipt identified in ρ_{revoke} and sends the refund f to the user.

4. PROTOCOL ANALYSIS

In this section we present an analysis of the receipt protocols based on the security and privacy requirements introduced in Section 2. For ease of understanding, a summary of the cryptographic techniques used in the various protocols which provide the various security and privacy properties are given in Table 4, Appendix B.

THEOREM 4.1. *(Security of Receipt protocols) All receipt protocols satisfy the security criteria namely 1)Correctness, 2)Integrity, 3) Single Submission, 4) Fairness and 5) Non-repudiation properties*

For all the protocols, the strong authentication at the registrar using AgZKP prevents impersonation attacks [1, 2]. As such, if an adversary is able to impersonate a given user U and authenticate using k random commitments, then that would imply that this adversary was able to steal the corresponding $2k$ secrets of U to construct a valid proof. Such compromise can occur with a very low probability and hence the login at the registrar is reliable. In addition the evaluation of the ASSURANCE LEVEL based on the users' weak identifiers stored at the RREC's, also mitigates the risk of the adding and using incorrect receipts. For each of the detailed protocols, the security criteria are discussed below.

In Protocol 2, *correctness* of the buyer and seller information is achieved by strong authentication and use of IBS. The IBS scheme is provably secure based on the Schnorr's signature scheme [16] in a random oracle model. If the signatures are correct, it would imply that the buyer and seller information provided for the receipt is correct. For the correctness of the receipt attributes, the contract signing protocol [20] is used. The users agree on a set of attributes relevant to the e-transaction and use it to carry out the protocol. *Integrity* of the e-receipts is achieved by IBS signatures [18] on the final receipts R_{U_A} and R_{U_B} provided by each user. The *single submission* is ensured based on the session handles included in ρ_{submit} token used during the final addition of the receipt at the registrar. The *fairness* is proven and achieved because of the use of the simultaneous contract signing protocol [20]. Finally *non-repudiation* is achieved because of the use of the IBS. This is because the IBS scheme used achieves the Girault's trusted level 3 [17] which implies the that private key generator (i.e. the registrar) does not know, or cannot easily compute, the users' private keys. Moreover, the certified email protocol given at step 3 requires that the requests and keys shared in step 3, be published, therefore the parties cannot deny carrying out the transaction.

In Protocol 3, the *correctness* is ensured because of the use of the ZKPK at step 3. Here the attribute being committed m_i is fixed and the ZKPK would not be correct if the Pedersen commitment was not formed correctly. Moreover, for the same reason, the user cannot create two commitments for a certain attribute m_i , thus achieving *single submission* property. The *integrity* and *non-repudiation* properties are achieved because of the use of the IBS signature and AgZKP at the time of strong authentication.

In Protocol 5, the *correctness* is achieved using strong authentication, followed by the AgZKP on the commitments identified in token ρ_{commit}. For the *integrity* of ρ_{commit}, the public key signature of the registrar on this token is used. The tamperproofness of ρ_{commit} prevents adding any additional commitment of an attribute which may not belong to a valid receipt. Thus the *single submission* of the attribute commitment is achieved.

In Protocol 6, ρ_{revoke} is first signed by the registrar using PKE, and eventually by the user using IBS. These signatures ensure *integrity* of the receipt which needs to be revoked. The signed token $\langle\rho_{revoke}\rangle_U$ and the timestamp t prevent receipt from being resubmitted by an adversary. Using ρ_{revoke} also helps in the *single submission* of receipt revocation requests. Finally the *fairness* and *non-repudiation* properties are achieved as in Protocol 2.

THEOREM 4.2. *(Privacy of receipt protocols) All receipt protocols preserve the privacy criteria, namely 1) user consent and 2) minimal disclosure of user receipt attributes and other user information, as described in Section 2.*

In Protocol 2 the *user consent* is captured using the IBS signatures, and the contract signing protocol. This is because only the user is assumed to have the secret key for executing the $Sign$ protocol. Moreover the terms and conditions of the e-transaction are encoded in the contract that is signed by each participant user. The protocol also ensures *minimal disclosure* which is achieved by the use of random session handles. Even if both users are strongly authenticated using AgZKP at the registrar, the users do not learn any other information besides the SSO ID of each other, and the information required for the e-transaction to occur.

In Protocol 3 the *user consent* is ensured when the user creates the cryptographic commitment followed by the IBS signature and the ZKPK on the committed value. Subsequently in Protocol 5, *user consent* is captured based on the ZKPK which can only be performed if the user provides its secrets associated with the receipt attributes on which the proofs are formed. The ZKPK also helps in the *minimal disclosure* because of the security of Pedersen commitment [5] which relies on the hardness of the discrete log problem. Finally in Protocol 6, the IBS and the certified email protocol ensures that the user participates in the revocation procedure and that there is *user consent*. During this protocol no other information other than the users' SSO ID is revealed to the SP conducting the revocation, thus ensuring *minimal disclosure* of user attributes.

5. VERYIDX IMPLEMENTATION

In this section we provide the architectural design and prototype implementation details of VeryIDX. We also perform a performance, usability and storage analysis of the prototype system.

5.1 System Architecture

To implement the VeryIDX system we developed components for three main entities of this system, namely *registrar, SP* and *user*. Several main considerations were taken into account in the design of VeryIDX.

1. The requirement to *minimally extend* the existing components used for e-commerce transactions.
   First, since users and SP should have easy access to the registrar we made our system web-based. Thus, no client side software installation is needed. Second, requiring modification to the current payment process of SP would not be desirable because of backward compatibility and scalability issues. Therefore we provide add-on modules for SP to join the VeryIDX system. Furthermore our system does not affect the legacy interactions between SP and users.

2. Providing *de facto interoperation*. VeryIDX achieves interoperation using a few registrar components. Different SP can specify their requirements according to their service policies and subsequently use the registrar to obtain relevant and reliable information when they have to make decisions for trust establishment.

3. Providing *scalable and interchangeable building blocks*. A modular application is composed of smaller, separated modules that are well isolated. Thus, it makes easier to develop and manage than tightly coupled application. We adopt modularization so it is easy to update component and simple to add new functionality.

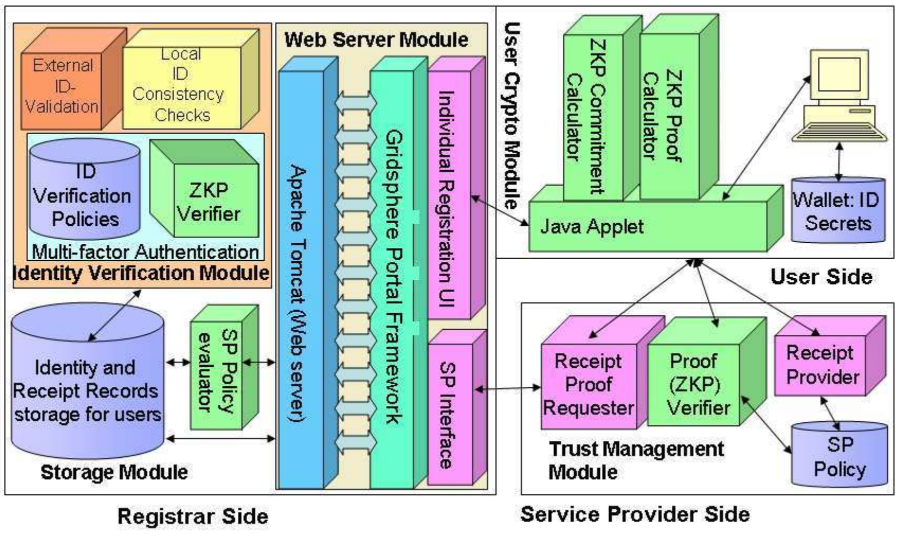


Figure 2: System architecture of VeryIDX

Figure 2 shows the general architecture of the current VeryIDX system. In the following we describe system architecture, implementation details and subcomponents of the three entities, namely registrar, user and SP.

5.1.1 Registrar Side

Registrar handles users' request to add or extend receipts and other user identity information. Registrar comprises four key modules which are described as follows.

1. *Webserver module.* This module comprises servlet container and the implementation of registrar. The servlet[4]container accepts users' connection and relay it to the registrar components, which is in charge of processing the users' request, e.g, showing dynamic receipts, allow users to add new receipt and so on. We used Gridsphere Portal Framework to create dynamic web pages because of its extensive built-in servlets, user management portlet capabilities and reliability. As a servlet container, Jakarta Tomcat has been selected for its popularity and robustness.

2. *Record storage.* To store the user records, namely RREC, we use Oracle 10g database. Identity related information like registrar's public key and some public parameters for example, p, q, g and h required for ZKPK protocols are also stored in the database.

3. *Identity verification module.* The identity verification module checks the correctness of the claimed identity. It performs ZKPK with the user to strongly authenticate this user to access its RREC. It is also used to assess the assurance level of a given receipt. This module checks whether a user logged in at the registrar is the same as the one indicated in the receipt obtained from SP. This is done using the assurance assessment steps defined in Protocol 1.
   We have deployed Secure Socket Layer (SSL) for the secure traffic using the Java Secure Socket Extension (JSSE) package.

4. *SP policy evaluator.* This module receives as input the trust establishment policies of the SP and a given RREC of the user and then uses this information to identify the potential receipts of the user that can satisfy the policy requirements. Thereafter the result is presented to the user, so it can construct the selective receipt based token like ρ_{attr} in Protocol 4 of Section 3.

[4]A servlet as traditional CGI but it's more efficient and convenient.

5.1.2 User Side

The key module used at the user end is the *User Crypto Module*. This module consists of two components namely *ZKP commitment Calculator* and *ZKP Proof Calculator*. The ZKP commitment calculator computes commitment of any given attribute related to either identity or receipt information. A critical requirement is that the secrets involved in creating the commitment should not leak outside the users' machine. Only the final commitment is revealed once the computation is complete and the user's secrets are stored at the user's local machine. The ZKP Proof Calculator creates a proof object which can be submitted to the SP or registrar. Using this object, the SP can carry out the ZKPK proof verification.

Java Applets are used to implement both components. Java applets can communicate not only with the servlet in the registrar but also with the users' local file system. In addition, some parameters about users can be passed to the applets from JSP. Applets have some access limitations to the user file system because they are not part of the local system. In our system, we use signed applets which can be allowed to access local filesystem. In addition, users are required to put a policy file to enable an applet to access local file having users' secret.

We devised a logical structure named 'wallet' where the user securely stores its personal information and secrets. When users create commitments, the secret information is stored at the wallet to be used later in ZKPK. The secret information in the wallet is never revealed to the Registrar nor the SP. We use Java Serialization technique[5] for applet to servlet communication. This approach makes implementation easy because it does not require the system to deploy additional protocols for the data transmission. The receiver simply needs to get serialized stream and recover the same objects Sender transmitted.

[5]Serialization saves the current state of objects to a stream and restores an equivalent object from the stream. Stream can hold data in a persistent container (disk) or transient container (RAM).

5.1.3 *Service Provider Side*

The *Trust Management Module* is the main extension required to the SP. Such module has four main components. The first is the SP *policy database*; such database is accessed by the *proof requester* component which is responsible for creating the conditional statements required to establish trust. Once the proof is provided, the *proof verifier* takes the proof object which may consist of clear attribute values or cryptographic proofs. The proofs are verified to get a boolean value, which determines the trust establishment decisions. The last component is the *receipt provider* which issues the receipts when the e-commerce transaction is completed.

5.2 Implementation Analysis

Our system is web oriented. Therefore the response time mainly depends from the network speed. The computation time and the storage requirements of our protocols are minimal as detailed below. We have carried out our experiments under following environments. The client computer had Intel Core duo 2GHz and 2G RAM, and server had 2.8GHz Pentium D CPU with 1G RAM.

Average time to execute ZKP using applets. From our experiments, the average time to log onto the registrar over 100 iterations takes less than 1 sec. Likewise, the time to download an applet takes around one second under a network whose average data receive rate is 928 Bytes per second (relatively slow connection). To extend a receipt, the applet running on client receives a tag and a value pair from the registrar which are then used to calculate the commitment. Excluding the user interaction time, to calculate a single commitment it takes an average of 0.011 sec.

Summing up the total time including commitment computation, transmission to the server and receipt of the reply, the average time to extend a receipt, takes 1.03 seconds. At the registrar side, one of the major functions is to store user's record into the database. Every time a user's commitment arrives to the server, the registrar makes a connection to the Oracle 10g database by issuing one INSERT statement. Finally the average insertion time was measured 0.5 sec.

Average storage needed at the user and the registrar. Our implementation requires less than 6M bytes of disk space for the portal codes under the tomcat directory at server side. At the client side, 5KB of user's local disk space is required for each x-receipt operation. Users' secret needed for extending receipts are recorded at the VeryIDX.wallet and its size increases around 5KB for each extension of a receipt. Storage efficiency can be improved by saving only the most essential data. If a user does online shopping 4 times a month, a local wallet requires only 240KB hard disk space per year.

The registrar stores users' record into Oracle database. For the other registrar components, the minimum space required is about 50MB for tomcat excluding disk space for the Operating System. The RREC of a user is on an average 67M bytes for 106 receipts with one cryptographic commitment. Each commitment value takes 31 digit characters on an average.

5.3 Usability

Usability is crucial to an e-commerce environment. As mentioned earlier, since most interactions and tasks can be carried out on the web, the users do not have to manage many applications in their local machine. The key component at the user side is the wallet where user's secret information is serialized into a local file. Therefore, if a user changes machines, it is not able to generate the required proof. One approach to address such issue is to use portable memory devices which can be secured by means of strong authentication mechanisms. Here we could use a simple utility that extracts the data from the wallet and generates an XML [8] document encoding all the data. We plan to explore this option as a part of our future work.

A key usability issue is the GUI of the applet itself. From the snapshots of the "adding identity" Applet (Figure 4) and "creating proofs" Applet (Figure 3), we observe that there are buttons like *create random*, *calculate commitment* which may not be intuitive to the user who is not aware of the protocol steps. These buttons are created at this stage of development to verify the correctness of the protocols, but can be automated in the actual deployment of the system. Potentially, we can merge all needed user interaction into just one button to provide a user-friendly interface.

6. RELATED WORK

Transaction history-based trust establishment has been explored from different perspectives. We elaborate on three different perspectives; the *reputation systems* which rely on the history of e-commerce activities of the users; the *transaction protocols* that ensure fair and safe transactions; the *cryptographic protocols* which ensure unforgeable receipts. Related work in each such area is detailed in the following.

Reputation systems have been investigated extensively. One approach to build a reputation system is to have a distributed trust management systems [13]. The basic idea of this work is to construct hierarchical reputation systems. Users who want to know reputation for a specific server (seller), query a local broker. Such reputation is calculated from users' evaluation after the completion of a transaction with the server. This score is merged throughout several brokers. Note that in this framework only servers' rating of the users is stored and not the attributes on which the score is calculated. In our system it is possible for SP to draw reputation score from users' transaction history given users' consent to view the receipts. Another key difference is with respect to the subject who uses reputation scores. In [13], users, as buyers, take advantage of SP's reputation score to choose trustworthy sellers whereas in our approach, the sellers utilize users' past transaction history to determine users' reputation.

Another approach to reputation systems has been developed in the context of P2P networks [10]. Such an approach does not depend on the customers' evaluation of the seller. Instead it suggests new credit computation schemes of a reputation system for decentralized unstructured P2P networks such as Gnutella [12]. The proposed system has two computation schemes, namely the debit-credit reputation computation (DCRC) and the credit-only reputation computation (CORC). In P2P system, every user shares files with other users and get files from them. Each user as a peer is both a client and a server in these networks. So when a user joins the system, its machine becomes a peer (server) to others. The reputation score of a user, as a server, is an important factor for decision-making- who to download content from. This score is raised as peers download more files from it. One novel contribution of this paper is to enable a peer to keep its reputation locally for the fast reputation retrieval. Reputation computation agent (RCA) prevents malicious reputation modification by use of a public key based mechanism. Unlike real world transactions, a sender (who shares files with others) is the one who gets receipts from receivers. Senders report those receipts to the RCA and receive the updated reputation score about themselves in return. In our approach we provide a way for both the buyer and the seller to create and submit receipts of their past transactions. In [10], the use of receipts is restricted to acquire credit from the server. By contrast, our receipts can be used as a proof of purchase for other types of transactions as well.

In the AttentionTrust [11] approach, users install Firefox extension to share their website access log with the SP's. This system supposes that users are willing to share their privacy without gaining any financial benefit. The Information sent to central server and SP's may be used for customized advertisement to the users. The extension sends web page URL, title, HTTP response code and so on to the server. This is similar to ours in that users are allowed to choose SP to share privacy information. However the users of AttentionTrust don't have much freedom about which information will be shared, though the users can specify the list of websites that should not receive its data. In our case, the users of Registrar are free to choose which information will be revealed to which SP, at what time and for what purpose.

Transaction protocols provide mechanisms to execute price negotiation, ordering and payment procedures. For example, a transaction server named NetBill for information goods was suggested in early 90s [7]. It takes part in payment procedure so as to allow a buyer to hide its identity from the seller and give certified receipts to the buyer. The main goal of this system is to assure a fair exchange between two parties i.e customers can read or use electronic goods only after they receive a decryption key from a merchant. The merchant sends decryption key to the buyer only if he got payment from the user and then reports an endorsed payment order to the server. Customers receive receipts consisting of transaction result, identity, price, product ID and so on. The server signs such receipt and then transmits it to the merchant. However, it is the responsibility of buyers to manage these receipts. We also investigate fair exchange in Protocol 2. In addition, we allow users to utilize this receipt for different purposes. Furthermore, our registrar helps the buyer to manage receipts systematically.

Finally, cryptography-oriented approaches have been proposed which deal with history-based trust establishment. For example, Simmons and Purdy proposed ZKP of identity attributes in transaction receipts [19]. They focus on the unforgeable transaction receipts using ZKP. They use a public authentication channel to create trusted credentials. These credentials can then be used for constructing proofs. Even though receipts in their scheme can be extended for use in two-way protocol between a seller and a buyer, using this receipt for other purposes does not seem trivial. This is because each user's credentials are highly specialized for the given scheme.

Another related research deals with non-repudiation in transactions is Coffey and Saidha [6]. They propose an approach to achieve mandatory mutual non-repudiation including both mandatory proof of origin and mandatory proof of receipt. As a result, their approach ensures that neither party can gain from the exchange not until the non-repudiation protocol is completed. This research is more focused on the transaction itself rather than on the receipt management. Since our system is not affected by the payment process, those techniques could be used together with ours.

7. CONCLUSION AND FUTURE WORK

In this paper we have presented an infrastructure and protocols to build and manage online transaction histories of users. Our architecture and prototype implementation is shown to be effective for history based trust establishment. As a part of future work, we plan to extend this work in several directions. The first is related to extend the protocols in order to use *cached receipts* so that the user does not have to construct new receipt tokens each time it accesses a service requiring similar receipt attributes. The second deals with extending the receipt addition protocols in order to acquire multiple receipts from various SP's in batches. The third deals with the extension of the trust establishment policies with respect to current e-commerce reputation establishment use cases. The fourth is related to an extension to the architecture and implementation of the system in order to support identity based signatures and contract signing. Currently the system performs the AgZKP as elaborated in Section 5. Finally we plan to investigate the human factor aspects of using and managing receipts in VeryIDX.

8. ACKNOWLEDGEMENT

The work reported in this paper has been sponsored by NSF under the ITR Project 0428554 "The Design and Use of Digital Identities".

9. REFERENCES

[1] A. Bhargav-Spantzel, A. Squicciarini, R. Xue, and E. Bertino. Practical identity theft prevention using aggregated proof of knowledge. Technical report, CS Department. CERIAS TR 2006-26.

[2] A. Bhargav-Spantzel, A. C. Squicciarini, and E. Bertino. Establishing and protecting digital identity in federation systems. *Journal of Computer Security*, 14(3):269–300, 2006.

[3] E. F. Brickell, D. Chaum, I. B. Damgård, and J. van de Graaf. Gradual and verifiable release of a secret. In C. Pomerance, editor, *Proc. CRYPTO 87*, pages 156–166. Springer-Verlag, 1988. Lecture Notes in Computer Science No. 293.

[4] J. Camenisch and M. Stadler. Efficient group signature schemes for large groups. In B. S. K. Jr., editor, *Advances in Cryptology – Crypto '97*, pages 410–424, Berlin, 1997. Springer. Lecture Notes in Computer Science 1294.

[5] D. Chaum and T. P. Pedersen. Wallet databases with observers. In E. F. Brickell, editor, *Proc. CRYPTO 92*, pages 89–105. Springer-Verlag, 1992. Lecture Notes in Computer Science No. 740.

[6] T. Coffey and P. Saidha. Non-repudiation with mandatory proof of receipt. *SIGCOMM Comput. Commun. Rev.*, 26(1):6–17, 1996.

[7] B. Cox, J. D. Tygar, and M. Sirbu. NetBill security and Transaction Protocol. pages 77–88, July 1995.

[8] v. . Extensible Markup Language (XML). W3c recommendation, 2006. http://www.w3.org/XML/.

[9] U. Feige, A. Fiat, and A. Shamir. Zero knowledge proofs of identity. In *Proc. 19th ACM Symp. on Theory of Computing*, pages 210–217, May 1987.

[10] M. Gupta, P. Judge, and M. Ammar. A reputation system for peer-to-peer networks. In *NOSSDAV '03: Proceedings of the 13th international workshop on Network and operating systems support for digital audio and video*, pages 144–152, New York, NY, USA, 2003. ACM Press.

[11] http://www.attentiontrust.org. Attentiontrust.org.

[12] http://www.gnutella.com. Gnutella.

[13] K.-J. Lin, H. Lu, T. Yu, and C. en Tai. A reputation and trust management broker framework for web applications. In *EEE '05: Proceedings of the 2005 IEEE International Conference on e-Technology, e-Commerce and e-Service*, pages 262–269, Washington, DC, USA, 2005. IEEE Computer Society.

[14] A. S. Patrick and S. Kenny. From privacy legislation to interface design: Implementing information privacy in human-computer interfaces. In *Proceedings of Privacy Enhancing Technologies Workshop (PET2003), LNCS 2760*, 2003.

[15] B. Schneier and J. Riordan. A certified e-mail protocol. In *ACSAC*, pages 347–352, 1998.

[16] C. P. Schnorr. Efficient signature generation by smart cards. *Journal of Cryptology*, 4(3):161–174, 1991.

[17] A. Shamir. Identity-based cryptosystems and signature schemes. In *Proceedings of CRYPTO 84 on Advances in cryptology*, pages 47–53, New York, NY, USA, 1985. Springer-Verlag New York, Inc.

[18] J. Shao, Z. Cao, and L. Wang. Efficient id-based threshold signature schemes without pairings. *Cryptology ePrint Archive*, 2006. http://eprint.iacr.org/2006/308.pdf.

[19] G. J. Simmons and G. B. Purdy". "zero-knowledge proofs of identity and veracity of transaction receipts". *EUROCRYPT '88: Workshop on the Theory and Application of Cryptographic Techniques, Davos, Switzerland, May 1988. Proceedings*, 330:35, 1998.

[20] C.-H. Wang and Y.-S. Kuo. An efficient contract signing protocol using the aggregate signature scheme to protect signers' privacy and promote reliability. *SIGOPS Oper. Syst. Rev.*, 39(4):66–79, 2005.

[21] G. Wang. Generic non-repudiation protocols supporting transparent off-line ttp. *J. Comput. Secur.*, 14(5):441–467, 2006.

APPENDIX

A. IDENTITY BASED SIGNATURE SCHEME

The Identity Based Signature Scheme introduced in [18] is described as follows. The ***Setup*** algorithm consists of the follows steps. Given security parameters $k_1, k_2 \in Z^+$

Step 1: Choose a k_1-bit prime p and a k_2-bit prime q, such that $q|p-1$.

Step 2: Choose generator g of order q in Z_p.

Step 3: Choose a random $x \in Z_q^*$, and compute $y = g^x \mod p$.

Step 4: Choose two cryptographic hash functions $H_1(\cdot)$ and $H_2(\cdot)$, such that $H_1 : \{0,1\}^* \to Z_q^*$, $H_2 : \{0,1\}^* \to Z_q^*$.

The ***Extract*** algorithm is an interactive protocol between the user and the Private Key Generator (PKG).

1. The user chooses a random $r_{ID} \in Z_q^*$, and computes $R_{ID} = g^{r_{ID}}$. It sends (ID, R_{ID}) to the PKG.
2. Upon receiving (ID, R_{ID}), the PKG does the following: 1) Chooses a random $r_{PKG} \in Z_q^*$, 2) computes $R_{PKG} = g^{r_{PKG}} \mod p$, and 3) computes $d_{ID} = r_{PKG} + xH_1(ID||R_{ID}||R_{PKG}) \mod q$. The PKG sends (R_{PKG}, d_{ID}) to user.
3. User checks $g^{d_{ID}} \stackrel{?}{=} R_{PKG}y^{H_1(ID||R_{ID}||R_{PKG})} \mod p$. If this check holds then the private key of the user is $sk_{ID} = r_{ID} + d_{ID} \mod q$.

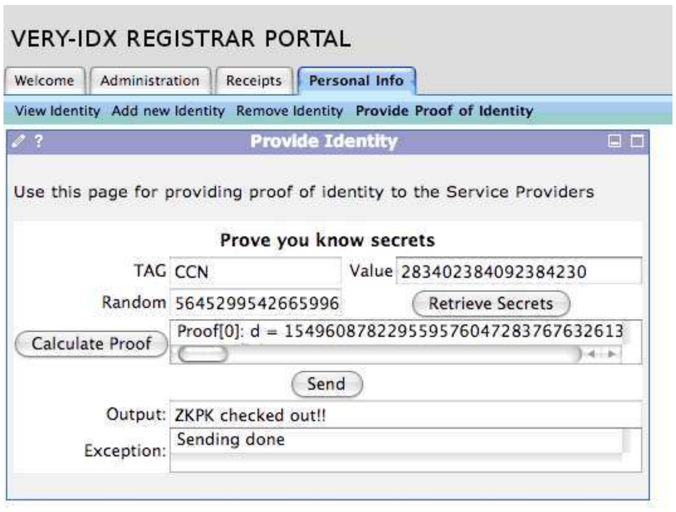
To ***Sign*** a message m, under the public key ID, the user 1) chooses a random $r \in Z_q^*$, 2) computes $R = g^r \mod p$ and $\beta = H_2(ID||R_{ID}||R_{PKG}||R||m)$, and 3) set the signature to be $R_{ID}, R_{PKG}, R, \sigma$ where $\sigma = r + (r_{ID} + d_{ID})\beta \mod q$.

To ***Verify*** a signature $R_{ID}, R_{PKG}, R, \sigma$ for message m, the verifier checks $g^{\sigma} \stackrel{?}{=} R(R_{ID}R_{PKG}y^{H_1(ID||R_{ID}||R_{PKG})})^{\beta} \mod p$

Note that in this scheme, *non-repudiation* is achieved by step 3 of the *Extract* protocol. This is because, the private key used to sign is never revealed even to the PKG involved.

B. PROTOCOL ANALYSIS SUMMARY

A summary of the cryptographic techniques used in the various protocols which provide the various security and privacy properties is provided in Table 4.



Use this page for providing proof of identity to the Service Providers

Prove you know secrets

TAG: CCN Value: 283402384092384230

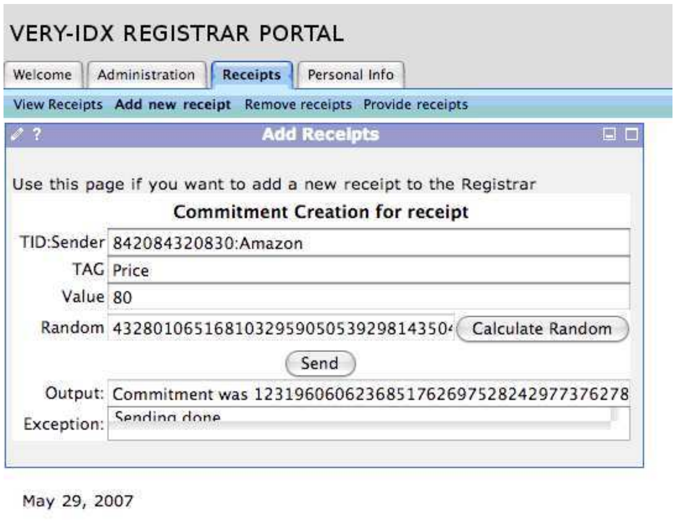
Random: 5645299542665996

Proof[0]: d = 154960878229559576047283767632613

Output: ZKPK checked out!!

Exception: Sending done

Figure 3: Applet for creating ZKPK for Identity Attributes



Use this page if you want to add a new receipt to the Registrar

Commitment Creation for receipt

TID:Sender: 842084320830:Amazon

TAG: Price

Value: 80

Random: 432801065168103295905053929814350

Output: Commitment was 1231960606236851762697528242977376278

Exception: Sending done

May 29, 2007

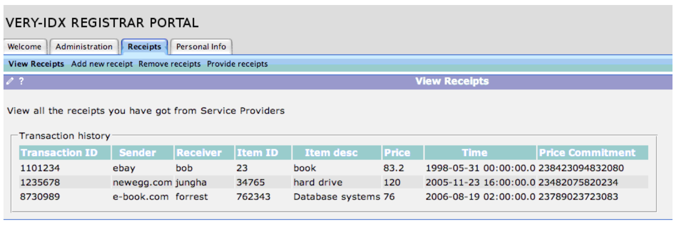
Figure 4: Applet for creating commitments for x-receipts

C. ILLUSTRATIVE EXAMPLE OF THE VERYIDX RECEIPT BASED SYSTEM

Consider a scenario when a user "Alice" has bought a book from VeryIDX enabled Amazon and now wants to opt-in to add the receipt of this transaction to her RREC at the registrar. She uses Protocol 1 to do so. Once she has sent her intention to get the receipt to SP (Step 1) then she logs on to the registrar using the user SSO ID and password. Her registrar requires strong authentication (Step 2) so it requires Alice to prove she knows the secrets corresponding to her Credit Card number commitment. She runs the proof calculator applet (See Figure 3) where she can automatically retrieve the required secrets by clicking the "Retrieve Secrets" button. Once the secrets are retrieved she clicks on "Calculate Proof" to calculate the proof object. Finally she sends the proof object to the registrar which is in turn used to verify its the correctness of the proof. If the proof is verified correctly, the reply of the registrar appears on the user applet. As a next step the registrar generates the ρ_{submit} used eventually by the SP to give Alice the correct receipt. Note

PROTO#	SECURITY					PRIVACY	
	Correctness	**Integrity**	**Single Submit**	**Fairness**	**Non-repudiation**	**User Consent**	**Min. Disclosure**
2	IBS, AgZKP, Contract Signing	IBS	Session handles	Contract Signing	IBS, Certified Email Protocol	IBS, Contract Signing	AgZKP, SSO ID
3	ZKPK	IBS	ZKPK	N/A	IBS	Commitment, ZKPK, IBS	N/A
5	AgZKP	PKE	Commitments	N/A	ZKPK transcript	ZKPK	ZKPK
6	PKE	PKE	PKE	Contract Signing	IBS, Certified Email Protocol	IBS, Certified Email	SSO ID

Table 4: Analysis of the security and privacy requirements of the receipt protocols based on cryptographic building blocks.



View all the receipts you have got from Service Providers

Transaction history

Transaction ID	Sender	Receiver	Item ID	Item desc	Price	Time	Price Commitment
1101234	ebay	bob	23	book	83.2	1998-05-31 00:00:00.0	238423094832080
1235678	newegg.com	jungha	34765	hard drive	120	2005-11-23 16:00:00.0	23482075820234
8730989	e-book.com	forrest	762343	Database systems	76	2006-08-19 02:00:00.0	23789023723083

Figure 5: Registrar portal view of Receipts in RREC

here that SP creates a TRANSACTION ID that is unique to this SP. Subsequently Alice can add this receipt using Step 5 of Protocol 1. At any point Alice can view her RREC at registrar by logging on to her registrar using step 2 of Protocol 1 (See Figure 5).

Once the e-receipt is submitted, Alice wants to extend this receipt using Protocol 3. More specifically she wants to create a cryptographic commitment corresponding to PRICE attribute of her receipt. She logs on to her account at the registrar and this time she runs the commitment creation applet (See Figure 4). Here the main requirement is that the user should have unique tag values corresponding to each commitment. The TAG is the combination of the TRANSACTION ID, SENDER and the type of attribute being committed (on this case the price). The random needed at Step 2 of Protocol 3 is computed when she clicks the "calculate Random" button. She can then send this commitment to the server.