Privacy-Preserving Management of Transactions' Receipts for Mobile Environments

Federica Paci
CS Department
Purdue University
West Lafayette, Indiana
paci@cs.purdue.edu

Ning Shang
CS Department
Purdue University
West Lafayette, Indiana
nshang@cs.purdue.edu

Sam Kerr
CS Department
Purdue University
West Lafayette, Indiana
skerr@cs.purdue.edu

Kevin Steuer Jr
CS Department
Purdue University
West Lafayette, Indiana
ksteuer@cs.purdue.edu

Jungha Woo
CS Department
Purdue University
West Lafayette, Indiana
wooj@cs.purdue.edu

Elisa Bertino
CS Department
Purdue University
West Lafayette, Indiana
bertino@cs.purdue.edu

ABSTRACT

Users increasingly use their mobile devices for electronic transactions to store related information, such as digital receipts. However, such information can be target of several attacks. There are some security issues related to M-commerce: the loss or theft of mobile devices results in a exposure of transaction information; transaction receipts that are send over WI-FI or 3G networks can be easily intercepted; transaction receipts can also be captured via Bluetooth connections without the user's consent; and mobile viruses, worms and Trojan horses can access the transaction information stored on mobile devices if this information is not protected by passwords or PIN numbers. Therefore, assuring privacy and security of transactions' information, as well as of any sensitive information stored on mobile devices is crucial. In this paper, we propose a privacy-preserving approach to manage electronic transaction receipts on mobile devices. The approach is based on the notion of *transaction receipts* issued by service providers upon a successful transaction and combines Pedersen commitment and Zero Knowledge Proof of Knowledge (ZKPK) techniques and Oblivious Commitment-Based Envelope (OCBE) protocols. We have developed a version of such protocol for Near Field Communication (NFC) enabled cellular phones.

Categories and Subject Descriptors

K.6.5 [**Management of Computing and Information Systems**]: [Security and protection]

General Terms

Security

Keywords

privacy, transaction record, registrar

1. INTRODUCTION

The combined use of the Internet and mobile technologies (e.g. mobile devices, mobile and wireless communication) is leading to major changes in how individuals communicate, conduct business transactions and access resources and services. People are able to communicate anytime, anywhere with anyone. Technological advances as well as the increased number of mobile applications have resulted in new additions in end-user equipment. Smart mobile devices are equipped with various communication technologies, such as GSM/GPRS, 802.11-WLAN, Bluetooth, NFC and RFID chips as well as GPS for location awareness. Mobile devices today offer a broad spectrum of functions, including web browsers, operating systems (e.g Symbian), environments (e.g., Java virtual machine) for running mobile applications, and e-mail clients.

In such context, establishing mutual trust between users and service providers is critical. A possible approach to establish trust is to view the transactions users have carried out in the past. The history of former transactions informs about users behavior, their ability and dispositions and thus helps to decide whom to trust. Yahoo! Auction, Amazon, eBay are examples of systems that rate both users and service providers based on their past interactions history. Maintaining the history of users' transactions and establishing trust based on these transactions and other factors is a complex task. An important component of any such solution is represented by systems managing *receipts* of transactions. By receipts we refer to information that characterizes a transaction, like the amount paid and the service provider with which the transaction was carried out.

Managing transaction receipts on mobile devices is very challenging. On one hand, the sharing of information about transactions should be facilitated among service providers. A customer should be able to disclose to a service provider a view of his/her past transactions with other service providers in order to get discounts or to prove good behavior over the past. On the other hand, transaction receipts need to be protected as they may convey sensitive information about

a user and can be the target of attacks. Moreover, users should be able to control which service provider has access to information about their past interactions. Assuring privacy and security of transactions' receipts, as well as of any sensitive information, in the context of mobile environments is further complicated by the fact that mobile devices are not secure. Recent statistics [4] show that millions of lost or stolen mobile devices which store users' sensitive data have been reported. In addition to loss or theft, there are an increasing number of viruses, worms and Trojan horses target mobile devices. Moreover, current attacks against Bluetooth and well-known WLAN and GPRS vulnerabilities show that it is very easy for attackers to compromise mobile devices [14]. Another issue is related to how service providers determine whether users are trusted based on their past transactions. Trust establishment should be a policy-driven process. Service providers should specify policies stating the conditions users' transaction receipts must satisfy for a user to be trusted and/or to get a service with favorable conditions. An example such a policy is that a user can receive a discount if he/she has spent $50 or more. Thus an important requirement is the introduction of a policy language that allows service providers to express conditions against transaction receipts.

To address such issues, we propose a policy-based approach for the management of users transaction history on mobile devices that provides:

1. integrity, confidentiality and privacy of users transaction information;

2. selective and minimal disclosure of transaction information;

3. trust establishment based on transaction history.

Our approach allows a user to prove to a service provider that he/she has performed a transaction satisfying a set of conditions by such service provider without revealing any information about the transaction. The approach is based on the notion of *transaction receipts* issued by service providers upon a successful transaction. Our approach combines Pedersen commitment and Zero Knowledge Proof of Knowledge (ZKPK) techniques and Oblivious Commitment-Based Envelope (OCBE) protocols [6] to assure privacy of information recorded in the receipts. We have developed a version of such an approach for Near Field Communication (NFC) [9] enabled cellular phones. A NFC device embedded in the cellular phone is able to communicate not only with Internet via wireless connections but also with smart card readers. In addition, the cellular phone applications, referred to as MIDlets, can access the phone's tag for reading and writing data.

The rest of the paper is organized as follows. Section 2 introduces the basic notions on which our approach is based. Section 3 presents our privacy-preserving approach to manage transaction receipts; it introduces all key notions of our approach, including the notion of verification policy, and describes our protocols. Section 4 analyzes the properties of our approach. Section 5 introduces the system architecture whereas Section 6 discusses the implementation and reports experimental results. Section 7 overviews related work. Finally, Section 8 concludes the paper and outlines some future work.

2. BASIC NOTIONS

In this section, we introduce the basic cryptographic notions on which our transaction receipts management approach is based.

2.1 Pedersen commitment

The Pedersen Commitment scheme, first introduced in [10], is an unconditionally hiding and computationally binding commitment scheme that is based on the intractability of the discrete logarithm problem.[1] The scheme is originally described with a specific implementation that uses a subgroup of the multiplicative group of a finite field. We remark that this choice of implementation is not intrinsic to the Pedersen commitment scheme itself – it can be implemented with any suitable abelian groups, e.g., elliptic curves over finite fields. Therefore, we rewrite the Pedersen commitment scheme in a more general language as follows.

Pedersen Commitment
Setup
A trusted third party T chooses a finite cyclic group G of large prime order p so that the *computational Diffie-Hellman problem*[2] is hard in G. Write the group operation in G as multiplication. T chooses an element $g \in G$ as a generator, and another element $h \in G$ such that it is hard to find the discrete logarithm of h with respect to g, i.e., an integer α such that $h = g^\alpha$. T may or may not know the number α. T publishes G, p, g and h as the system's parameters.
Commit
The domain of committed values is the finite field $\mathbb{F}_p$ of p elements, which can be represented as the set of integers $\mathbb{F}_p = \{0, 1, \ldots, p-1\}$. For a party U to commit a value $x \in \mathbb{F}_p$, it randomly chooses $r \in \mathbb{F}_p$, and computes the commitment $c = g^x h^r \in G$.
Open
U shows the values x and r to open a commitment c. The verifier checks whether $c = g^x h^r$.

2.2 Zero-knowledge proof of knowledge (ZKPK) protocol

It turns out that in the Pedersen commitment scheme described above, a party U referred to as the prover, can convince the verifier, V, that U can open a commitment $c = g^x h^r$, without showing the values x and r in clear. Indeed, by following the zero-knowledge proof of knowledge (ZKPK) protocol below, V will learn nothing about the actual values of x and r. This ZKPK protocol, which works for Pedersen commitments, is an adapted version of the zero-knowledge proof protocol proposed by Schnorr [12].

Zero-knowledge proof of knowledge (Schnorr protocol)
As in the case of Pedersen commitment scheme, a trusted party T generates public parameters G, p, g, h. A prover

---

[1]Let G be a (multiplicatively written) cyclic group of order q and let g be a generator of G. The map $\varphi : \mathbb{Z} \to G, \varphi(n) = g^n$ is a group homomorphism with kernel $\mathbb{Z}_q$. The problem of computing the inverse map of φ is called the *discrete logarithm problem (DLP) to the base of g*.

[2]For a cyclic group G (written multiplicatively) of order q, with a generator $g \in G$, the *Computational Diffie-Hellman Problem* is the following problem: Given g^a and g^b for randomly-chosen secret $a, b \in \{0, \ldots, q-1\}$, compute g^{ab}.

U who holds private knowledge of values x and r can convince a verifier V that U can open the Pedersen commitment $c = g^x h^r$ as follows.

1. U randomly chooses $y, s \in \mathbb{F}_p^*$, and sends V the element $d = g^y h^s \in G$.

2. V picks a random value $e \in \mathbb{F}_p^*$, and sends e as a challenge to U.

3. U sends $u = y + ex, v = s + er$, both in $\mathbb{F}_p$, to V.

4. V accepts the proof if and only if $g^u h^v = d \cdot c^e$ in G.

We use this protocol in Section 3.3.3 for proof of receipt ownership.

2.3 OCBE protocols

The Oblivious Commitment-Based Envelope (OCBE) protocols, proposed in [6], provide the capability of enforcing access control policies in an oblivious way. Three communications parties are involved in OCBE protocols: a receiver Re, a sender Se, and a trusted third party T. More precisely, the OCBE protocols ensure that the receiver Re can decrypt a message sent by Se if and only if its committed value satisfies a condition given by a predicate in Se's access control policy, while Se learns nothing about the committed value. The possible predicates are comparison predicates $=, \neq, >, \geq, <$ and $\leq$.

The OCBE protocols are built with several cryptographic components:

1. The Pedersen commitment scheme.

2. A semantically secure symmetric-key encryption algorithm $\mathcal{E}$, for example, AES, with key length k-bits. Let $\mathcal{E}_{\mathsf{Key}}[M]$ denote the encrypted message M under the encryption algorithm $\mathcal{E}$ with symmetric encryption key Key.

3. A cryptographic hash function $H(\cdot) : \{0,1\}^* \to \{0,1\}^k$. When we write $H(\alpha)$ for an input α in a certain set, we adopt the convention that there is a canonical encoding which encodes α as a bit string, i.e., an element in $\{0,1\}^*$, without explicitly specifying the encoding.

Given the notation as above, we summarize the EQ-OCBE and GE-OCBE protocols, i.e., the OCBE protocols for $=$ and $\geq$ predicates, respectively, in what follows. The OCBE protocols for other predicates can be derived and described in a similar fashion. The protocols are stated in a slightly different way than in [6], to better suit the presentation in this paper.

EQ-OCBE Protocol
Parameter generation
T runs a Pedersen commitment setup protocol to generate system parameters $\mathsf{Param} = \langle G, g, h \rangle$. T also outputs the order of G, p, and $\mathcal{P} = \{\mathsf{EQ}_{x_0} : x_0 \in \mathbb{F}_p\}$, where

$$\mathsf{EQ}_{a_0} : \mathbb{F}_p \to \{\mathsf{true}, \mathsf{false}\}$$

is an equality predicate such that $\mathsf{EQ}_{x_0}(x)$ is true if and only if $x = x_0$.
Commitment
T first chooses an element $x \in \mathbb{F}_p$ for Re to commit. T then randomly chooses $r \in \mathbb{F}_p$, and computes the Pedersen commitment $c = g^x h^r$. T sends x, r, c to Re, and sends c to Se.[3]
Interaction
- Re makes a data service request to Se.
- Based on this request, Se sends an equality predicate $\mathsf{EQ}_{x_0} \in \mathcal{P}$.
- Upon receiving this predicate, Re sends a Pedersen commitment $c = g^x h^r$ to Se.
- Se randomly picks $y \in \mathbb{F}_p^*$, computes $\sigma = (cg^{-x_0})^y$, and sends to Re a pair $\langle \eta = h^y, C = \mathcal{E}_{H(\sigma)}[M] \rangle$, where M is the message containing the requested data.

Open
Upon receiving $\langle \eta, C \rangle$ from Se, Re computes $\sigma' = \eta^r$, and decrypts C using $H(\sigma')$.

GE-OCBE Protocol
Parameter generation
As in EQ-OCBE, T runs a Pedersen commitment setup protocol to generate system parameters $\mathsf{Param} = \langle G, g, h \rangle$, and outputs the order of G, p. In addition, T chooses another parameter ℓ, which specifies an upper bound for the length of attribute values, such that $2^\ell < p/2$. T also outputs $\mathcal{V} = \{0, 1, \ldots, 2^\ell - 1\} \subset \mathbb{F}_p$, and $\mathcal{P} = \{\mathsf{GE}_{x_0} : x_0 \in \mathcal{V}\}$, where

$$\mathsf{GE}_{x_0} : \mathcal{V} \to \{\mathsf{true}, \mathsf{false}\}$$

is a predicate such that $\mathsf{GE}_{x_0}(x)$ is true if and only if $x \geq x_0$.
Commitment
This step is the same as EQ-OCBE. T chooses an integer $x \in \mathcal{V}$ for Re to commit. T then randomly chooses $r \in \mathbb{F}_p$, and computes the Pedersen commitment $c = g^x h^r$. T sends x, r, c to Re, and sends c to Se.[4]
Interaction
- Re makes a data service request to Se.
- Based on the request, Se sends to Re a predicate $\mathsf{GE}_{x_0} \in \mathcal{P}$.
- Upon receiving this predicate, Re sends to Se a Pedersen commitment $c = g^x h^r$.
- Let $d = (x - x_0) \pmod{p}$. Re picks $r_1, \ldots, r_{\ell-1} \in \mathbb{F}_p$, and sets $r_0 = r - \sum_{i=1}^{\ell-1} 2^i r_i$. If $\mathsf{GE}_{x_0}(x)$ is true, let $d_{\ell-1} \ldots d_1 d_0$ be d's binary representation, with d_0 the lowest bit. Otherwise if GE_{x_0} is false, Re randomly chooses $d_{\ell-1}, \ldots, d_1 \in \{0, 1\}$, and sets $d_0 = d - \sum_{i=1}^{\ell-1} 2^i d_i \pmod{p}$. Re computes ℓ commitments $c_i = g^{d_i} h^{r_i}$ for $0 \leq i \leq \ell - 1$, and sends all of them to Se.
- Se checks that $cg^{-x_0} = \prod_{i=0}^{\ell-1} (c_i)^{2^i}$. Se randomly chooses ℓ bit strings $k_0, \ldots, k_{\ell-1}$, and sets $k = H(k_0 \| \ldots \| k_{\ell-1})$. Se picks $y \in \mathbb{F}_p^*$, and computes $\eta = h^y, C = \mathcal{E}_k[M]$, where M is the message containing requested data. For each $0 \leq i \leq \ell - 1$ and $j = 0, 1$, Se computes $\sigma_i^j = (c_i g^{-j})^y, C_i^j = H(\sigma_i^j) \oplus k_i$. Se sends to Re the tuple

$$\langle \eta, C_0^0, C_0^1, \ldots, C_{\ell-1}^0, C_{\ell-1}^1, C \rangle.$$

---

[3]In an offline alternative, T can digitally sign c and sends x, r, c and the signature of c to Re. Then the validity of the commitment c can be ensured by verifying T's signature. In this way, after Se obtains T's public key for signature verification, no communication is needed between T and Se.
[4]Similarly, an offline alternative also works here.

Open

After Re receives the tuple $\langle \eta, C_0^0, C_0^1, \ldots, C_{\ell-1}^0, C_{\ell-1}^1, C \rangle$ from Se as above, Re computes $\sigma_i' = \eta^{r_i}$, and $k_i' = H(\sigma_i') \oplus C_i^{d_i}$, for $0 \leq i \leq \ell-1$. Re then computes $k' = H(k_0' \parallel \ldots \parallel k_{\ell-1}')$, and decrypts C using key k'.

LE-OCBE, the OCBE protocol for the $\leq$ predicates, can be constructed in a similar way as GE-OCBE. Other OCBE protocols (for $\neq, <, >$ predicates) can be built on EQ-OCBE, GE-OCBE and LE-OCBE.

All these OCBE protocols guarantee that the receiver Re can decrypt the message sent by Se if and only if the corresponding predicate is evaluated as true at Re's committed value, and that Se does not learn anything about this committed value.

We remark that for certain applications, we can let Se know whether Re's committed value satisfies the specified predicate, by extending the OCBE protocols with one more step: Re shows to Se the decrypted message. We discuss this in more details in Section 3.3.4.

2.4 Shamir's secret sharing scheme

Shamir's (k, n) threshold scheme [13] is a method that divides a secret into n shares and allows the secret to be reconstructed if and only if any k shares are present. Here k and n are both positive integers and $k \leq n$. It is also called Shamir's secret sharing scheme.

The scheme works as follows. A trusted party, T, chooses a finite field $\mathbb{F}_p$ of p elements, with p large enough. Let the secret message S be encoded as an element $a_0 \in \mathbb{F}_p$. T randomly chooses $k-1$ elements $a_1, \ldots, a_{k-1} \in \mathbb{F}_p$, and constructs a degree $k-1$ polynomial $f(x) = a_0 + a_1 x + \ldots + a_{k-1} x^{k-1} \in \mathbb{F}_p[x]$. T chooses n elements $\alpha_1, \alpha_2, \ldots, \alpha_n \in \mathbb{F}_p$, and creates the secret shares S_i as pairs

$$S_i = (\alpha_i, f(\alpha_i)), 1 \leq i \leq n,$$

where $f(\alpha_i)$ is the polynomial evaluation of f at α_i. Given any subset of k such shares, the polynomial $f(x)$, of degree $k-1$, can be efficiently reconstructed via interpolation (see, e.g., [5], Section 2.2). The secret S, encoded as the constant coefficient a_0, is thus recovered.

Shamir's (k, n) threshold scheme has many good properties. Most prominently, it is information theoretically secure, in the sense that the knowledge of less than k shares gives no information about the secret S better than guessing; and it is minimal, in that the size of each share does not exceed the size of the secret. Interested readers can refer to [13] for more details.

3. PROTOCOLS FOR THE RECEIPTS MANAGEMENT

Our approach is based on the notion of *transaction receipts* that are issued by service providers to users upon a successful transaction. In the following sections, we first introduce the notion of transaction receipts, the policy language used by service providers to specify conditions against transaction receipts, and then the privacy-preserving protocol that allow a user to prove the possession of transaction receipts verifying the service provider policies.

3.1 Transaction Receipts

A service provider, upon the completion of a transaction, usually sends the user a receipt that specify a set of in-

TRAN-ID	ATTR		COM		SIG
1234	BUYER	John Smith	BUYER	7645353 6366363	1124457 6590873 3647688
	SELLER	BookStore.com	SELLER	1312425 54546	
	CATEGORY	Books	CATEGORY	2224223 525	
	PRICE	30	PRICE	1341515	
	DATE	11-04-2008	DATE	1315657	

Figure 1: A transaction receipt example

formation about the transaction such as the user identifier, the identifier of the service provider, the item(s) bought, the price paid for the item(s), the quantity, the date of the transaction, and shipment and billing information. We denote this type of information as *transaction attributes*. We consider only a subset of the possible attributes that can be associated with a transaction. The subset includes the user identifier, the service provider identifier, the category to which the item bought belongs to, the item price and the date of the transaction because they are the more relevant attributes to establish trust in the user.

We assume that service providers have a PKI infrastructure that allows them to issue users signed transaction receipts. In particular, we assume that each service provider is associated with a pair of keys (K_{Priv}, K_{Pub}) where K_{Priv} is the private key used to sign the transaction receipts and K_{Pub} is the public key used by other service providers to verify authenticity and integrity of receipts. In order to support a privacy-preserving proof of the possession of such receipts, the transaction receipts released under our protocol include the transactions' attributes in clear and their corresponding Pedersen commitment. The Pedersen commitments of a transaction attributes are used by a user to prove the possession of the receipt of this transaction to other service providers. To compute the Pedersen commitments of the transaction attributes, the service provider runs the Pedersen commitment setup protocol described in Section 2.2 to generate the parameters $\mathsf{Param} = \langle G, g, h \rangle$. Then, the service provider publishes G, p, g and h and its public key K_{Pub}.

The structure of transaction receipts is defined as follows.

DEFINITION 3.1 (TRANSACTION RECEIPT). *Let* SP *be a service provider and* B *be a user with which* SP *has successfully carried out a transaction* $\mathcal{T}r$. *Let* (G, p, g, h, K_{Pub}) *be the public parameters of* SP. *The receipt for transaction* $\mathcal{T}r$ *carried out by* B *and* SP *is a tuple* $\langle$TRAN-ID, ATTR, COM, SIG$\rangle$, *where* TRAN-ID *is the transaction identifier;* ATTR *is the set of transaction attributes* {BUYER, SELLER, CATEGORY, PRICE, DATE} *where 1)* BUYER *is the user identifier, 2)* SELLER *is the service provider's identifier, 3)* CATEGORY *is the selling category of the item being bought, 4)* PRICE *is the price of the item and* DATE *is the date of the transaction, respectively;* COM *is the set of the Pedersen commitments of the attributes in* ATTR. *Each element in* COM *is a tuple of the form* $\langle$A, COMMIT$\rangle$ *where* A *is the value of an attribute in* ATTR *and* COMMIT *is the Pedersen commitment* $g^A h^r$ *of* A *and* r *is a secret known only to* B. SIG *is the signature of service provider* SP *on* COM[5]. ∎

─────────

[5]In what follows, we will use the dot notation to denote the different components of transaction receipt.

EXAMPLE 3.1. *Suppose that John Smith has bought for $ 30 a book from "BookStore.Com" on the 4th of November 2008. A receipt for this transaction, issued according to our protocol, is ⟨ "1234", ("John Smith", BookStore.Com", "Books", "$ 30", "11-04-2008"), (⟨BUYER, 45785687994674⟩, ⟨CATEGORY, 76553940894⟩, ⟨PRICE, 2223422262⟩, ⟨DATE, 58300242341⟩), 1375350748530-50356376037⟩ (see Figure 1).*

3.2 Verification Policy Language

Service providers usually evaluate users based on previous transaction interactions with service providers. Based on users' historical transactions, service providers are able to determine whether a user can be trusted and whether he/she can be qualified to gain some benefits such as a discount or rebate. Service providers define policies, referred to as *verification policies*, to specify the conditions against attributes which are recorded in transaction receipts.

Verification policies are formally defined as follows.

DEFINITION 3.2 (TERM). *A Term is is an expression of the form Name(attribute_list) where: Name is the name of a service or discount or an item, whereas attribute_list is a possible empty set of attribute names characterizing the service.*

DEFINITION 3.3 (ATTRIBUTE CONDITION). *An attribute condition $Cond$ is an expression of the form: "$name_A$ op l", where $name_A$ is the name of a transaction attribute A, op is a comparison operator such as $=, <, >, \leq, \geq, \neq$, and l is a value that can be assumed by attribute A.* ∎

EXAMPLE 3.2. *Examples of policies conditions are the following:*

- SELLER = "BookStore.Com"

- DATE < "11-04-2008"

- PRICE > $ 80

DEFINITION 3.4 (VERIFICATION POLICY). *A verification policy Pol is an expression of the form "$\mathcal{R} \leftarrow Cond_1, Cond_2, \dots Cond_n$", $n \geq 1$, where $\mathcal{R}$ is a Term and $Cond_1, Cond_2, \dots Cond_n$ are attribute conditions.* ∎

Given a transaction receipt Tr and a verification policy $Pol : \mathcal{R} \leftarrow Cond_1, Cond_2, \dots Cond_n$, $n \geq 1$, if for each $Cond_i \in Pol$, (ii) $\exists \bar{A} \in Tr.ATTR$ such that $name_{\bar{A}} = Cond.name_A$ and $value_{\bar{A}}$ satisfies $Cond.(name_A$ op $l)$, we say that Tr satisfies Pol, denoted as $Tr \triangleright Pol$.

EXAMPLE 3.3. *An example of verification policy is the following: $Pol : Discount(OnItem ="Glamour", Amount="$ 15") \leftarrow$ SELLER = "BookStore.Com", PRICE > "$ 80 ", DATE < "11-04-2008". The policy states that a user is qualified for a $ 15 discount on an yearly subscription to Glamour magazine, if the user has spent more than $ 80 at "BookStore.Com" before "11-04-2008'.*

3.3 Protocol to Manage Transaction Receipts

The privacy-preserving protocol proves the possession of a transaction receipt and is carried out between a user and a service provider. The protocol consists of four main phases (see Figure 2): [6]

---

[6]In what follows we use the term 'user'; however in practice the steps are carried out by the client software transparently to the actual end user.

1. **Integrity verification of Receipts Attributes**. The user sends a transaction receipt to a service provider to satisfy the service provider verification policy. The service provider verifies the signature on the transaction receipt sent by the user to prove the satisfiability of service provider's verification policy.

2. **Secret Sharing on the Mobile Phone**. The user reconstructs the secret r that has been used to compute the transaction attribute commitments. Remember that r has been split for better protection from unauthorized accesses.

3. **Proof of Receipt Ownership**. The user proves he/she is the owner of the transaction receipt by carrying out a zero-knowledge proof of knowledge protocol with the service provider.

4. **Verification of Conditions on Receipts**. The service provider verifies that the transaction receipt attributes satisfy its verification policy by carrying out an OCBE protocol with the user.

In the following sections, we describe the details of each phase of the protocol.

3.3.1 Integrity Verification of Receipts Attributes

This phase starts when a user makes a request to a service provider and the service provider sends the user the corresponding verification policy $\mathcal{R} \leftarrow Cond_1, Cond_2, \dots Cond_n$, $n \geq 1$. First the user selects a transaction receipt Tr that satisfies such policy. Then, the user sends the service provider Tr.COM, Tr.SIG, Tr.ATTR.SELLER, and the identifier of the service provider which has issued Tr. The service provider retrieves the public key K_{Pub} of the service provider that has issued Tr to be able to verify the signature Tr.SIG.

3.3.2 Secret Sharing on the Mobile Phones

In order for a user to be able to carry out ZKPK and OCBE protocols with the service provider, the user needs the random secret r, used to compute the Pedersen commitments of a transaction receipt's attributes. The security of the protocols strongly depends on r so it is necessary to protect it from unauthorized access that can occur on mobile devices. Mobile device security can be compromised if the device is lost or stolen, or due to the vulnerabilities of the communication network and/or the device software. To prevent these security threats, we adopt Shamir's secret sharing scheme that allows one to split a secret in n shares and then to reconstruct it if and only if k shares are present. The storage of the shares depends on the specific architecture of the mobile devices. Next we will focus on the Nokia NFC mobile phones that we have used in our implementation.

In our implementation the shares are stored on different mobile phone components and (possibly) on external devices such as a PC or a an external storage unit. We split each random secret into four shares s_1, s_2, s_3 and s_4. The first share s_1 is stored in the internal memory of the mobile phone. The second share s_2 is further split into two secrets. A user chosen PIN number P and a number P' are selected such that $P \oplus P' = s_2 \bullet P'$ is stored in the phone external memory. The third share s_3 is stored in the smart card integrated in the phone. Finally the fourth secret share s_4 is stored in the user's PC which has to be accessed remotely by the
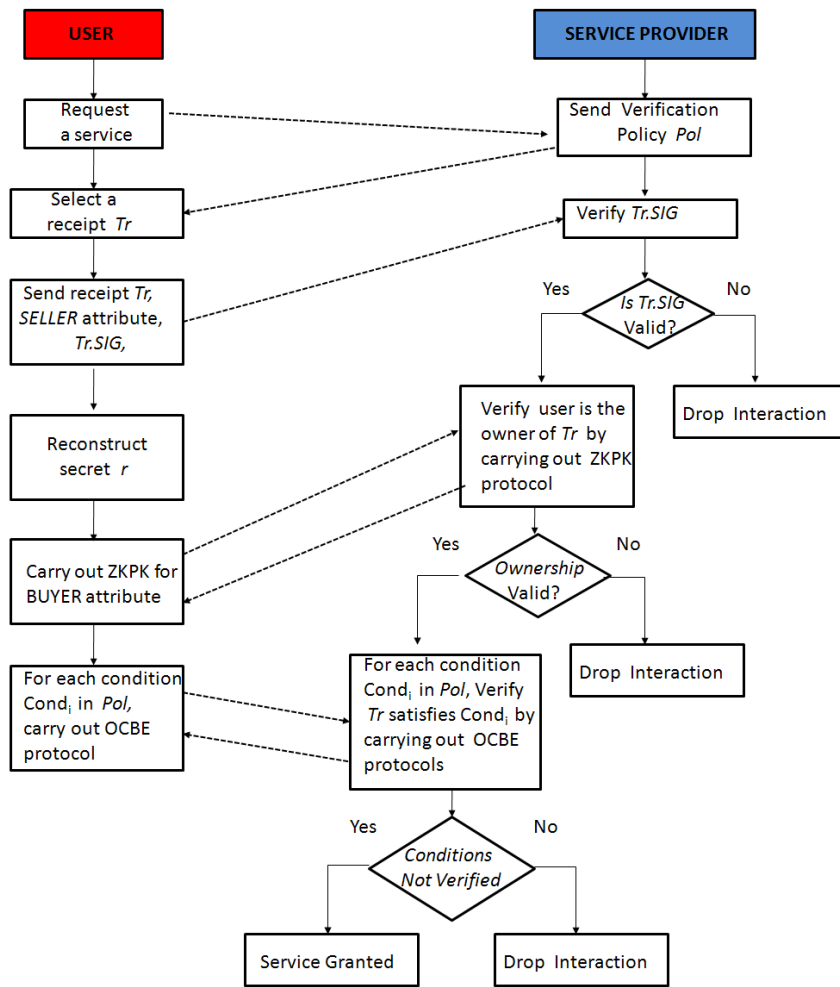
Figure 2: Approach schema

phone. We consider four levels of protection for the secret r that correspond to the number k of shares that are needed to reconstruct r. The possible levels of protection are *low*, *medium*, *medium-high* and *high*. The level of protection *low* requires no splitting of the secret r. In this case, r is stored in the phone smart card. The *medium* level corresponds to a value of k equal to 2. In this case the user has to retrieve two of the four shares s_1, s_2, s_3 and s_4 to obtain the secret r. If the *medium-high* level is chosen, three shares are needed while with level of protection *high*, all the four shares are needed to reconstruct the secret. The level of protection is set by the user[7] once the issuer of a transaction receipt sends the user the random secret r along with the transaction receipt containing the Pedersen commitments computed using r. Once set, the level of protection cannot be changed by the user.

When the user has to prove the ownership of the transaction receipt sent to the service provider, the r needs to be reconstructed. In order to do that, a number of shares

according to the level of protection set up by the user needs to be retrieved and then combined to obtain r.

EXAMPLE 3.4. *Suppose that John Smith has to prove the possession of receipt ⟨ "1234", ("John Smith", BookStore.Com", "Books", "\$ 30", "11-04-2008"), (⟨BUYER, 45785687994674⟩, ⟨CATEGORY, 76553940894⟩,⟨PRICE,2223422262⟩, ⟨DATE, 58300242341⟩)),13753507485305035637603⟩ to service provider "Borders". In order to accomplish that, John needs to reconstruct the secret r used to compute the Pedersen commitments contained in the receipt. John sets the security level for r to* high *and to retrieve each secret share he has to perform the following steps:*

1. *John retrieves s_1 from the phone internal memory.*

2. *To retrieve s_2, John inputs the secret PIN number P using the phone keypad. P' is retrieved from the phone external memory and it is used to compute the second secret share $s_2 = P \oplus P'$.*

3. *John retrieves the secret s_3 from the phone smart card.*

4. *To retrieve the secret share s_4 stored at the user's PC, John connects to its PC by using the phone*

---

[7]The specification of the security level and the entering of the PIN are the only steps that need to be carried by the actual end-user. The security level can however be set as a default and the end-user does not need to enter it each time it receives a new receipt.
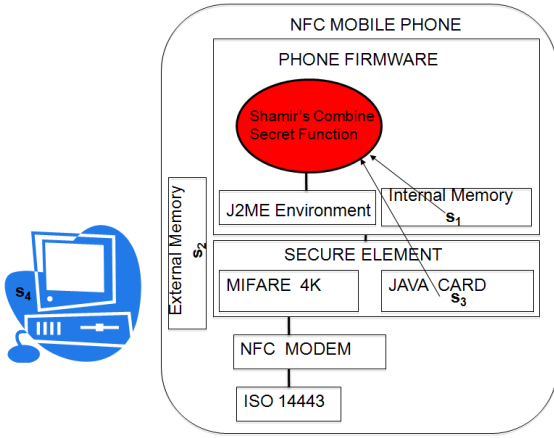
Figure 3: Random Secret Reconstruction

By contrast if John sets up a medium security level, he has to retrieve only two shares to obtain the secret r. For example, John can decide to get the shares s_1 and s_3 from the phone's internal memory and the phone smart card respectively without having to insert any PIN number (see Figure 3).

3.3.3 Proof of Receipt Ownership

Once the user has reconstructed the random secret r, the proof of the ownership of the transaction receipt can be achieved by engaging a ZPK protocol for the BUYER transaction attribute with the service provider. According to the ZPK protocol, the user randomly picks y, s in $\{1,..,p\}$, computes $d = g^y h^s$, where g and h are the public parameters of the service provider. The user then sends d to the service provider. Once received d, the service provider sends back a random challenge $e \in \{1,..,p-1\}$ to the client. Then the user computes $u = y + em$ and $v = s + er$ where m is the value of the BUYER transaction attribute and r is the random secret, and sends u and v to the service provider. The service provider accepts the aggregated zero knowledge proof if $g^u h^v = dc^e$. Otherwise, the interaction with the user is dropped.

3.3.4 Verification of Conditions on Receipts

We consider two scenarios that require the verification of conditions on transaction receipts. In the first scenario, a service provider provides a general service to all qualified users, and does not require to know the outcome of the transaction. For example, a book store may provide a transferable 10%-off coupon code to any user who presents a receipt showing a purchase of a product in the "Books" category. However, the book store does not care whether this coupon code is successfully received by the user; it only cares that a coupon code is valid when being used. The book store simply rejects a receipt if it is shown twice, to prevent a user from taking advantage of this offer for multiple times. In such a scenario, the OCBE protocols, (cfr. Section 2.3) can be used directly. Let the user be the receiver Re, and the service provider be the sender Se. Re sends a service request to Se, and Se responds with its verification policy. Based on the policy, Re selects a receipt $\mathcal{T}r$ which satisfies Se's policy, and sends $\mathcal{T}r$.COM, $\mathcal{T}r$.SIG, and the value of

SELLER attribute to Se. Se chooses the message M, as described in Section 2.3, to be the content of service (e.g., a coupon code). Then, it composes the envelope using the corresponding attribute value in the received receipt for M, and sends it to Re. Re can open the envelope if and only if the involved attribute value on the receipt satisfies the condition specified in the policy, but Se will not know if Re can open the envelope.

In the second scenario, the service provider needs to know the result of the condition verification, i.e., it should be informed if the attributes on the user's receipt satisfies the specified policy. There are many instances of such a scenario. For example, the service provider may require its policy be satisfied by a user's receipt in order to continue the transactions. In this case, for user privacy protection, the OCBE protocol for equality predicates, EQ-OCBE, should not be employed, because the service provider will be able to infer the attribute value if the verification is successful. However, other OCBE protocols which are for inequality predicates can still be used, with one more step appended to the protocol, described next.

In this additional step, the service provider acts as the sender Se, and the user acts as the receiver Re. The service provider chooses the message M to be a random bit string, which will be used as a secret of Se. The OCBE protocol for inequality predicates is executed between Se and Re, based on Se's policy and the involved attribute value recorded in Re's receipt, for this secret M. At the end of the protocol, after opening the envelope, Re shows Se the decrypted message M'. The attribute on the receipt passes Se's verification if $M = M'$, or fails if otherwise. The service provider continues with the transactions in the former case, or aborts the transaction in the latter case. Such additional step has been added to the OCBE protocols, to allow the service provider to learn the result of the verification, at the user's will. Since the random bit string M contains no useful information about the service content itself, a qualified user must choose to show the correctly decrypted secret message M, in order to continue the transactions with the service provider. In this sense, the extended OCBE protocols (for inequality predicates) works as a zero-knowledge proof scheme for our application.

In both scenarios, if the user's receipt's attributes need to satisfy multiple conditions in the service provider's policy, a run of the OCBE protocol must be performed for each condition. A receipt's attributes satisfy the conditions in the policy if and only if the user can open all related envelopes.

4. PROTOCOL ANALYSIS

In this section we analyze the security properties of our transaction receipts management protocol.

Our protocol is built on provably secure cryptographic protocols: digital signature scheme, Shamir's secret sharing scheme, Pedersen commitment, Schnorr's zero-knowledge proof protocol, and OCBE protocols.

After a user sends a service request to a service provider and receives a policy, he/she selects a transaction receipt $\mathcal{T}r$, and sends back $\mathcal{T}r$.COM, $\mathcal{T}r$.SIG and $\mathcal{T}r$.ATTR.SELLER, i.e., the receipt's parts containing Pedersen commitments, receipt issuer (seller)'s signature on these commitments and the identity of the issuer, respectively. On one hand, since the service provider verifies the issuer's signature on the Pedersen commitments, it is guaranteed that the Pedersen com-
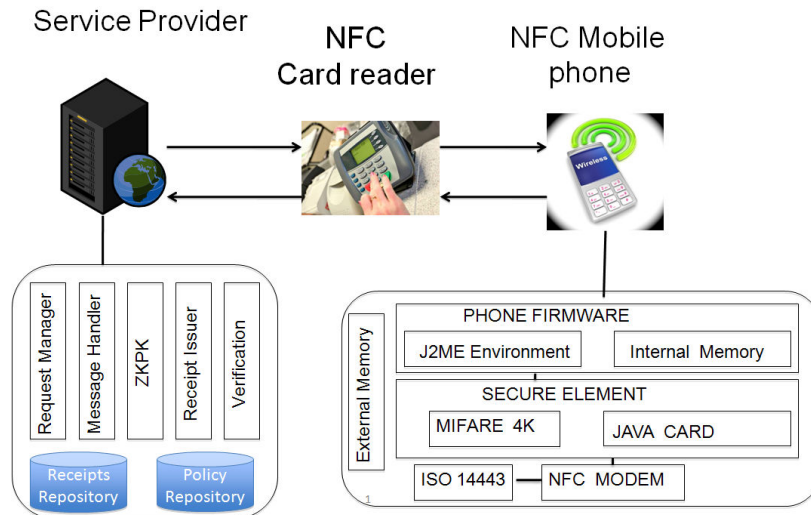
Figure 4: System architecture

mitments have not be modified. Thus, the integrity of the Pedersen commitments is assured. On the other hand, the service provider does not learn anything about the actual values of the transaction attributes. This is due to the unconditionally hiding property of the Pedersen commitment.

If the user passes the first step above, he/she starts to reconstruct the secret exponent r, which is used to prove the ownership of a receipt and the verification of conditions on receipts, from some of the shares s_1, s_2, s_3 and s_4 using Shamir's (k, n) threshold scheme. The number of shares needed for the reconstruction depends on the pre-defined level of protection. Since the shares are distributed at different locations, and protected by a PIN number, this makes it hard for a party other than the receipt owner to obtain all needed shares to recover r. Furthermore, since the Shamir's threshold scheme is information theoretically secure, unless enough shares are collected, any attempt to recover the secret r is not easier than guessing.

Once the secret r is reconstructed, the user carries out a zero-knowledge proof protocol for the BUYER attribute, in a manner like Schnorr's as described in Section 3.3.3, with the service provider. The user is able to convince the service provider that he/she knows how to open the commitment, only if he/she knows the values of both x and r such that the corresponding commitment is computed as $g^x h^r$. It prevents an entity who steals a valid receipt but does not know how to open the asked commitment in the receipt from authenticating with the service provider. Due to the zero-knowledge property of the protocol, the service provider does not learn the attribute value x for BUYER.

The last step of our protocol is the execution of the OCBE protocols for the verification of the conditions on the receipt attribute values. The OCBE protocols guarantee that a user can correctly retrieve a message, randomly chosen by the service provider, if and only if the user knows how to open the commitments whose committed values satisfy the conditions (equality or inequality) in the service provider's policy, while the service provider learns nothing about the actual values of the transaction attributes.

Based on the above considerations, our protocol guaran-

tees the integrity and the privacy of the information included in a transaction receipt and it also protects users against identity theft.

5. SYSTEM ARCHITECTURE

We have implemented our protocol on Nokia 6131 NFC [3] mobile phones. NFC enabled devices are gaining popularity because they provide easy-to-use mechanisms for ubiquitous accesses to systems and services. Based on a short-range wireless connectivity, the communication is activated by bringing two NFC compatible devices or tags within a few centimeters from one another.

The system architecture is shown in Figure 4. It consists of three main components: a service provider application, an external NFC reader and the Nokia 6131 NFC [3] mobile phone. The core architectural component is the NFC mobile phone. It consists of an `Antenna`, for detecting external targets such as tags, external readers, or other Nokia 6131 NFC mobile phones; an `NFC modem`, for providing the capability to send and receive commands between antenna, secure element and phone firmware including J2ME environment; a `Secure element`, for enabling third-party application development using tag/card emulation; `Phone firmware`, for providing mobile phone functions with NFC features; a `SIM card`, for GSM subscription identification and service management; `J2ME environment` included in phone firmware, for enabling third-party application development using Nokia 6131 NFC features; and an `External memory`.

The `Secure element` within Nokia 6131 NFC can store information securely, which can be used for payment and ticketing applications or for access control and electronic identifications. `Secure element` is divided into two sub-components, `Java Card` area (also referred to as smart card) and `Mifare 4K` area. `Mifare 4K` area can be considered as a memory with access control, and typically it is simpler to implement than a smart card application. `Mifare 4K` contains data, whereas smart card application contains an executable program. `Java Card` provides high security environment and executes code, which means it can be used for more complex applications. Therefore, we store in the

Java Card some of the shares in which the random secret r is split because of the high security provided by Java Card. Secure element is accessible through NFC modem internally from MIDLets and externally by external readers. MIDLets are Java applications running in the J2ME environment. In the next section we describe in details, how we have implement our protocol to manage receipts by using MIDLets.

The NFC reader enables the communication between the service provider application and the mobile phone. It transmits and receives messages from the NFC cellular phone. The service provider application consists of five main modules: Request Manager, Message Handler, ZKPK, Receipt Issuance and Verification. The Request Manager module parses users requests and selects from a local repository the verification policy that applies to the request. The Message Handler module provides all functions supporting the communications between the service provider application and the external NFC reader. The ZKPK module supports the verification of receipts' integrity and the ZKPK protocol to verify the BUYER attribute. The Receipt Issuance module provides the functions for creating a transaction receipt, such as the generation of the Pedersen commitments and the signature of the commitments. Once created, the transaction receipts are stored in a local repository. The Verification module supports the steps for the verification of conditions on receipts described in Section 3.3.4.

6. IMPLEMENTATION AND EXPERIMENTAL EVALUATION

To evaluate the performance of our protocol, we have developed a prototype version of the system. We have implemented a MIDLet that supports the integrity verification of receipts attributes, the proof of receipt ownership and the verification of conditions against receipts. The implementation of the secret sharing phase is under development.

We store users' transaction receipts in the external phone memory, whereas the secret r used to compute the secure commitments included in the receipts is saved in the Java Card component. The execution of the MIDLet is triggered when the Mifare 4K captures the verification policy sent by the service provider's external NFC reader and the Mifare 4K transfers such policy to the phone main memory. The MIDLet retrieves from the external memory a transaction receipt that satisfies the service provider policy and sends the part of the receipt containing the transaction attributes commitments, the signature affixed on the commitments, and the value of SELLER attribute to Mifare 4K so that can be read by the service provider's external NFC reader. If the service provider application successfully verifies the signature on the receipts commitments, the MidLet retrieves the secret r from the Java Card, and performs the other steps of the receipts management protocol.

The MIDLet runs on Java 2 Micro Edition (J2ME). Since J2ME is aimed at hardware with limited resources, it contains a minimum set of class libraries for specific types of hardware. In our implementation on conventional non-mobile platforms, we used the BigInteger and SecureRandom class, defined in J2SE java.math and java.security packages respectively, to implement secure commitments, but both packages are not supported in J2ME. Therefore, we have used the third-party cryptography provider BouncyCastle [2], a lightweight cryptography APIs for Java and C# that pro-

vide implementation of the BigInteger and SecureRandom classes. In addition, because of the limited memory size of mobile phones, we reduced the MIDLet's code size by using code obfuscation techniques provided by Sun's NetBeans IDE. Code obfuscation allows one to reduce a file size by replacing all Java packages and class names with meaningless characters. For example, a file of a size of 844KB can be reduced to a size of 17KB.

We have also implemented the service provider component as a web application using Java and the Apache Tomcat Application Server. The current implementation of the Verification module only supports the EQ-OCBE and GE-OCBE protocols for the verification of equality conditions and inequality conditions expressed by using the $\geq$ comparison operator. We are extending the implementation with support for other comparison operators.

We have performed several experiments to evaluate the execution time of the MIDLet and the service provider (SP for short) application for the proof of the receipt ownership and the verification of conditions (equality and inequality) against receipts. We have collected data about the execution times for verifying the equality conditions on receipts and the time for verifying the inequality conditions by using respectively EQ-OCBE and GE-OCBE protocols by varying the value of parameter ℓ from 5 to 20. ℓ determines the number of commitments $c_i = g^{d_i} h^{r_i}$, $0 \leq i \leq \ell - 1$ that the user has to send to the service provider to prove he/she satisfies an inequality condition in service provider policy.

The experiment compares the envelope creation time at the service provider's side, and the envelope opening time at the MIDLet's side, which are the most computationally expensive part for both protocols. We also record the time required for generating the additional Pedersen commitments c_i in GE-OCBE at the MIDLet's side. No additional commitment needs to be generated by the user in EQ-OCBE. We do not include the communication time and the symmetric encryption time in the comparisons, which vary with different network settings and plaintext lengths, in order to focus on the main operations of the protocols. We also do not include the signature verification time in the comparison, for the same reason.

In the experiment, we have executed the verification protocol both at the service provider's side and at the MIDLet size, for 10 times, and we have computed the average of the obtained values.

	Verification of Receipt Ownership
MIDLet	0.042
SP's Application	0.0311

Table 1: Average time (in seconds) to verify the ownership of a receipt at MIDLet's side and at SP's side

Table 1 shows the execution times taken by the verification of receipt ownership phase at MIDLet side and at the SP application side.

	Commitments Creation	Opening Envelope	Total Execution Time
Equality Condition	0	1.126	1.126
Inequality Condition ($\geq$)	5.875	6.088	11.963

Table 2: Verification of conditions' execution time (in seconds) at MIDLet's side ($\ell = 5$)

	Envelope Creation
Equality Condition	0.0409
Inequality Condition ($\geq$)	0.165

Table 3: SP's application's average execution time (in seconds) for verifying one condition ($\ell = 5$)

Table 2 and Table 3 report the average verification of conditions' execution time taken, respectively, by the MIDLet and by the SP application for a value of parameter ℓ equal to 5. When multiple conditions are to be verified, the execution time increases accordingly, as the protocol is repeated for multiple rounds. As expected, the execution time to verify inequality conditions takes more time than the verification of equality conditions. In fact, the GE-OCBE used to verify inequality condition with comparison predicate $\geq$, requires the MIDLet and the SP application to perform more interactions steps. Figure 5 shows the SP application's execution time to create the envelope according to GE-OCBE protocol while Figure 6 shows the time taken by the MIDLet to open the envelope. In both cases, we have varied the value of ℓ parameter from 5 to 20. The graphs show how the value of parameter ℓ dramatically impacts on verification of conditions' execution time. With the increasing of the ℓ parameter values, the execution time linearly increases. The verification time increases because when ℓ parameter increases, the SP application has to compute an higher number of $\sigma_i^j = (c_i g^{-j})^y$, $C_i^j = H(\sigma_i^j) \oplus k_i$ to be sent to the MIDLet running on user's mobile phone and the MIDLet to decrypt the envelope has to compute an higher number of $\sigma_i' = \eta^{r_i}$, and $k_i' = H(\sigma_i') \oplus C_i^{d_i}$, for $0 \leq i \leq \ell - 1$.

Therefore, in the implementation of our protocol, the parameter ℓ must be kept as small as possible in order to reduce the computational cost.

We expect other OCBE protocols for inequality predicates to give performance results similar to those of GE-OCBE, because the design and operations are similar.
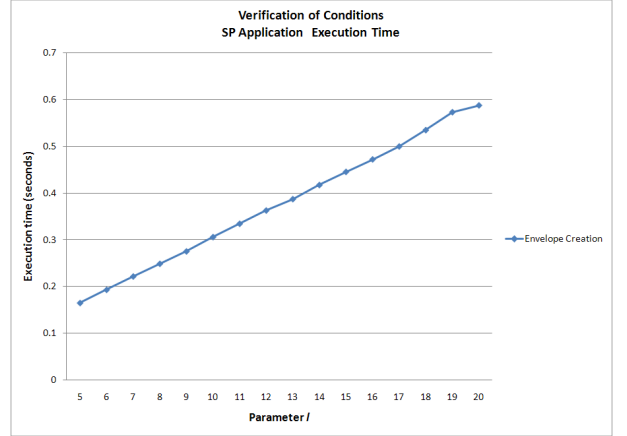

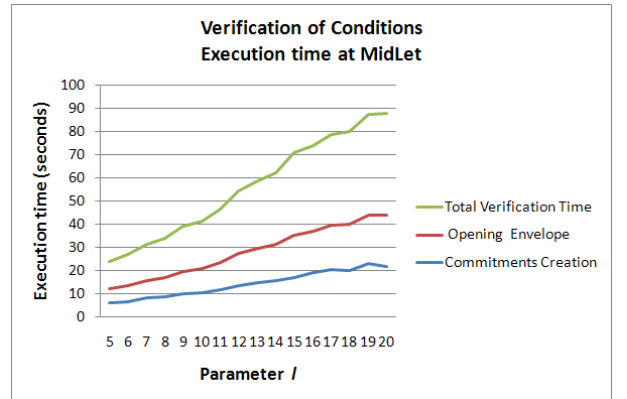Figure 5: SP Application's Envelope Creation Time varying the value of parameter ℓ

Figure 6: MIDLet's Envelope Opening Time varying the value of parameter ℓ

7. RELATED WORK

In this section, we compare our approach with other approaches for mobile transactions managements.

With the advent of high-speed data networks and feature-rich mobile, the concept of *mobile wallet* [8, 1] has gained importance. The ESPRIT project CAFE [1] has introduced the notion of electronic wallet, that is a small portable device which manages off-line digital payments to be used in commercial transactions. The electronic wallet transacts via a short range infrared channel either directly with compliant cash registers and wallets held by other individuals, or over the Internet, to merchants' tills or service points provided by banks and other organizations. The electronic wallet relies on a blind signature scheme to guarantee privacy and unlinkability for the electronic payment information while our approach preserves only the privacy of transactions information.

Mjolsnes et al. [8] have proposed a version of the electronic wallet for online payments. The authors exploits a credential server, denoted as *credential keeper* that securely stores the credentials issued to a user by different issuers. The credentials represents the wallet of the user. The user to access his/her credentials at the credential keeper and provide them to a service provider, has to present an access credential, e.g. a symmetric key, to the keeper server. To increase even more security, the access credential is encrypted and protected within a mobile device, and it can only be activated by using a PIN code or some other authentication method. In our approach we do not need a third component to guarantee a secure storage and management of the information included in a transaction receipt. The receipts can be securely stored on the phone external memory because the values of the transaction attributes are not stored in clear but they are substituted by their Pedersen commitments.

The Secure Electronic Transaction (SET) [11] protocol was developed to allow credit card holders to make transactions without revealing their credit card numbers to merchants and also to assure authenticity of the parties. SET deploys dual signature for merchant and payment gateway. Each party can only read a message designated for itself since each message is encrypted for a different target. To enable this feature, card holders and merchants must register with a Certificate Authority before they exchanging a SET message. SET assures both confidentiality and integrity of the messages among card holders, merchants and payment gateway whereas our protocol is designed to assure integrity and privacy of transactions information. SET authenticates the identity of the cardholder and the merchant to each other because both are registered with the same certificate authority. However, our protocols do not mandate this requirement. SET is considered to have failed because of its complexity. It requires cardholders and merchants to register in advance and get X.509 certificates to make transactions whereas the users need not to have such PKI certificate in our protocol. In our approach only service providers need to have a PKI certificate.

More recently, Veijalainen et al. [15], propose an approach to manage transaction on mobile devices. Their solution is based on the use of an application running on the phone denoted as *Mobile Commerce Transaction Manager* that provides the functionalities to start, terminate and resume a transaction with a service provider. With respect to security and privacy of transactions information, the *Mobile Commerce Transaction Manager* only guarantees confidentiality by encrypting the messages exchanged between the service provider application and the application running on the phone. In our approach by using digital signatures, Pedersen commitment, ZKPK techniques and OCBE protocols, we are able to guarantee both privacy and integrity of transactions information.

Finally, MeT initiative [7] has the goal to develop secure and easy methods and platforms for conducting e-commerce transactions on mobile phones. The strategy for MeT is to base the framework on existing standards such as WAP, Wireless Transport Layer Security (WTLS), Wireless Identification Module (WIM), Public Key Infrastructure (PKI) and Bluetooth. Privacy and security are ensured with digital signatures and cryptography services for transaction verification, confidentiality, authentication, and non-repudiation.

8. CONCLUSIONS

We have proposed a privacy preserving approach to manage electronic transaction receipts on mobile devices. We have focused on such type of device because we believe that in the near future users will conduct business transactions and access resources and services mostly using their mobile phones and PDAs. However, we have also implemented a web-based version of our receipt management system.

Our approach is based on the notion of *transaction receipt*, that records the information characterizing a transaction, and combines Pedersen commitment, ZKPK techniques and OCBE protocols. We have implemented our approach on Nokia 6131 NFC mobile phones and have evaluated its performance of on these devices. The experimental results show that our protocol is quite efficient in verifying equality conditions on receipts; however we need to improve the performance of the inequality conditions' verification. We believe that the reasons for the high execution times when verifying inequality conditions are the limited computational capability of Nokia 6131 NFC mobile phones and the use of the BouncyCastle API that are not natively supported by these phones. We plan to test our protocol on the Nokia 6212 NFC mobile phones that support JSR-177 Security and Trust APIs. These APIs provide security services to J2ME enabled devices without the need of using BouncyCastle API's. Since these API's are natively supported by these kind of phones, we believe that our protocols should perform better on such phones. We are currently completing the implementation of our prototype system by developing a MIDLet supporting the secret sharing phase of our protocol. We plan to complete the implementation of service provider application's `Verification` module in order to support the verification of inequality conditions containing the comparison predicates $<, >$ and $\neq$.

9. ACKNOWLEDGMENTS

of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the U.S. Department of Homeland Security, the I3P, or Dartmouth College.

10. REFERENCES

[1] J-P Boly, A. Bosselaers, R. Cramer, R. Michelsen, S. Fr. Mjolsnes, F. Muller, T.P. Pedersen, B. Pfitzmann, P. de Rooij, B. Schoenmakers, M. Schunter, L. Vallee, and M. Waidner. The ESPRIT project CAFE - high security digital payment systems. In *ESORICS*, pages 217–230, 1994.

[2] Bouncy Castle Crypto APIs. http://www.bouncycastle.org/.

[3] Nokia Forum. Nokia 6131 NFC Technical Description. http://www.forum.nokia.com.

[4] Help for lost and stolen phones. http://news.bbc.co.uk/1/hi/technology/4033461.stm.

[5] W. Gautschi. *Numerical Analysis: An Introduction.* Birkhauser Boston Inc., Cambridge, MA, USA, 1997.

[6] J. Li and N. Li. OACerts: Oblivious attribute certificates. *IEEE Transactions on Dependable and Secure Computing*, 3(4):340–352, 2006.

[7] Met initiative. http://www.mobiletransaction.org.

[8] S.F. Mjolsnes and C. Rong. Localized credentials for server assisted mobile wallet. *ICCNMC'01: International Conference on Computer Networks and Mobile Computing*, 00:203, 2001.

[9] Near Field Communication Forum. http://www.nfc-forum.org.

[10] T.P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In *CRYPTO '91: Proceedings of the 11th Annual International Cryptology Conference on Advances in Cryptology*, pages 129–140, London, UK,

[11] SET- Secure Electronic Transaction specification book 1: Business description, 1997. 1992. Springer-Verlag.

[12] C-P Schnorr. Efficient identification and signatures for smart cards. In *CRYPTO '89: Proceedings of the 9th Annual International Cryptology Conference on Advances in Cryptology*, pages 239–252, London, UK, 1990. Springer-Verlag.

[13] A. Shamir. How to share a secret. *Commun. ACM*, 22(11):612–613, 1979.

[14] TechRepublic. Identify and reduce mobile device security risks. http://articles.techrepublic.com.com/5100-22_11-5274902.html.

[15] J. Veijalainen, V. Y. Terziyan, and H. Tirri. Transaction management for m-commerce at a mobile terminal. *Electronic Commerce Research and Applications*, 5(3):229–245, 2006.