# VeryIDX - A Digital Identity Management System for Pervasive Computing Environments

Federica Paci[1], Elisa Bertino[1], Sam Kerr[1], Aaron Lint[1], Anna Squicciarini[2], and Jungha Woo[1]

[1] CERIAS and Computer Science Department, Purdue University
[2] Information Sciences and Technology, The Pennsylvania State University

**Abstract.** The problem of identity theft, that is, the act of impersonating others identities by presenting stolen identifiers or proofs of identities, has been receiving increasing attention because of its high financial and social costs. In this paper we address such problem by proposing an approach to manage user identity attributes by assuring their privacy-preserving usage. The approach is based on the concept of privacy preserving multi-factor authentication achieved by a new cryptographic primitive which uses aggregate signatures on commitments that are then used for aggregate zero-knowledge proof of knowledge (ZKPK) protocols. We present the implementation of such approach on Nokia NFC cellular phones and report performance evaluation results.

## 1 Introduction

Today a global information infrastructure connects remote parties worldwide through the use of large scale networks, relying on application level protocols and services, such as recent web service technology. Execution of activities in various domains, such as shopping, entertainment, business and scientific collaboration, and at various levels within those contexts, is increasingly based on the use of remote resources and services. The interaction between different remotely-located parties should be based on little knowledge about each other. In such a scenario, digital identity management (DIM) technology is fundamental in customizing user experience, protecting privacy, underpinning accountability in business transactions, and in complying with regulatory controls. Digital identity can be defined as the digital representation of the information known about a specific individual or organization. As such, it encompasses not only login names, but many additional information, referred to as *identity attributes*. The management of identity attributes raises a number of challenges. On one hand, identity attributes need to be shared to speed up and facilitate authentication of users and access control in a variety of contexts, including mobile environments. Users should be able to manage their identity attributes when carrying transactions or other interactions from portable devices such as cellular phones. On the other hand, the identity attributes must be protected as they may convey sensitive information about an individual and can be target of attacks.

The management of identity attributes on portable devices is however challenging. First, it is not trivial to ensure the security and privacy of the identity attributes. By using technologies such as Bluetooth or RFIDs [13], a party, for example a service provider, could retrieve information from the user portable devices without user consent. A second issue is related to the storage and computational constraints of most portable devices which require efficient protocols for managing identity attributes. To date there are no comprehensive solutions for handling identity attributes on mobile devices and even solutions for conventional non-mobile environments are still at a preliminary stage.

In this paper we make some steps towards such a solution and present a multi-factor identity attribute verification approach for mobile devices. By multi-factor verification we mean that whenever an individual presents an identity attribute for carrying on a transaction with a party, such party may verify the right of this individual to use such identity attribute by asking him/her to present other identity attributes. The specification of which identity attributes have to be presented is stated by *verification policies*. Different parties in a distributed system may specify different policies. To assure that such an approach does not undermine privacy, we have developed a cryptographic protocol, referred to as *aggregate zero knowledge proof* [4]. Such a protocol allows a user to prove the knowledge of multiple secrets to a party without having to reveal them to this party. We have developed a version of such protocol for Near Field Communication (NFC) [13] enabled cellular phones. NFC is a standard-based, short-range ($\sim$ 15 centimeters) wireless connectivity technology supporting two-way interactions among electronic devices [13]. 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 provides an overview of VeryIDX, our system for managing identity attributes. Section 3 introduces the basic notions on which the multi-factor identity verification is based. Section 4 presents the protocols for securing, managing and using identity attributes on the cellular phone. Section 5 describes the implementation of the multi-factor identity verification protocol on Nokia NFC mobile phones. Section 6 presents experimental performance results. Section 7 discusses related work. Finally, Section 8 concludes the paper and outline some future work.

## 2 VeryIDX Overview

Our approach is based on an extended notion of federation. A federation is composed of the following entities: identity providers (IdPs), service providers (SPs), registrars and users. SPs provide services to users as in conventional e-commerce and other federated environments. IdPs issue certified identity attributes to users and control the sharing of such information. The *registrars* store and manage information related to *strong identity attributes*, that is, identity attributes uniquely identifying an individual, as opposed to *weak identity attributes*

which do not have such property. The information recorded at the registrar is used to perform multi-factor identity attribute verification. Note that, unlike the IdPs, the information stored at the registrar does not iclude the values of the strong identity attributes in clear. Instead, such information only contains the cryptographic semantically secure *commitments* of the strong identity attributes which are then used by the clients, running on behalf of users, to construct zero knowledge proofs of knowledge (ZKPK) [10] of those attributes. The key elements of our solution can be summarized as follows:

1. Whenever a party $P$ presents a strong identity attribute to a SP in the federation, the SP requires additional proofs of identity according to its local verification policies. The submission of additional proofs of identity by $P$ and the corresponding verification by the SP is executed through the use of our aggregated ZKPK protocols. By using such protocol the party can prove knowledge of any strong identity attributes efficiently. Since the actual values of the identifiers are not revealed to the SP, this approach preserves the privacy of the parties.
2. Each strong identity attribute used by a party $P$ in a federation, either for direct use or just for identity proof, must be registered with a registrar that, upon registration, provides $P$ with a signature on the commitment of the identifier. The management of the registered strong identity attributes is based on a *identity record* (IdR) created for each registering party. The identity record collects the commitments corresponding to the strong identity attributes.
3. To prevent a malicious party from registering with a federation a strong identity attribute owned by another individual, a duplicate detection protocol is run upon registration to determine whether the same strong identity attribute has already been registered by a different party.

*Example 1.* Consider a user Bob who is part of the E-Mall federation, that offers a safe environment for online shopping. Bob enrolls at registrar $Reg_1$ and registers his strong identifiers: his credit card number (CCN) and his social security number (SSN). The commitments values of CCN and SSN signed by the registrar are maintained in Bob's IdR. Bob now can use his CCN and SSN to prove his identity. Suppose then that Bob wants to buy a book from $e-Follets$ SP. According to $e-Follets$'s policy, this store requires Bob's CCN along with a different form of identity verification for authentication. $e-Follets$ thus challenges Bob's SSN. As such, Bob, in order to prove the ownership of CNN, downloads his IdR from the registrar $Reg_1$ onto his NFC cellular phone. The device retrieves the identity tuples corresponding to CCN and SSN specified in the SP-'s $e-Follets$ policy and builds the aggregate proof of knowledge to be sent to $e-Follets$.

## 3 Preliminary Concepts

In this section we first introduce the cryptographic protocols that are used to implement our privacy preserving multi-factor identity verification approach. We

first introduce the Pedersen commitments used to generate strong identity attributes secure commitments and the ZKPK protocol. Then, we briefly describe the Boneh's protocol [6] to generate aggregate signatures based on bilinear mappings.

*Pedersen Commitment.* Let $g$ and $h$ be generators of a group $G$ of prime order $q$. A value $m$ is committed by choosing $r$ randomly from $\mathbb{Z}_q$ and giving commitment $C = g^m h^r$. 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 $r$.

*Zero-knowledge proof of knowledge.* In our approach we use the techniques by Camenisch and Stadler [7] for the various ZKPK of discrete logarithms and proofs of the validity of statements about discrete logarithms. We also conform to the same notation [7]. For instance to denote the ZKPK of values $\alpha$ and $\beta$ such that $y = g^\alpha h^\beta$ holds, and $u \leq \alpha \leq v$, we use the following notation:

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

*Bilinear maps.* For a security parameter $k$, let $q$ be a prime of length $k$, and $G_1$, $G_2$, $G_T$ be groups of order $q$. Let $g_1 \in G_1$, $g_2 \in G_2$ be generators. Function $e$: $G_1 \times G_2 \rightarrow G_T$ is a bilinear mapping if it satisfies the following properties:

1. For all $u \in G_1$, $v \in G_2$ and $a$, $b \in Z$, $e(u^a,v^b) = e(u,v)^{ab}$.
2. $e(g_1, g_2) \neq 1 \in G_T$.
3. There exists a computable isomorphism $\varphi$ from $G_2$ to $G_1$, such that $\varphi(g_2) = g_1$.

*Bilinear aggregate signatures.* The aggregate signature concept has been proposed by Boneh et. al [6]. We refer to such signature scheme as BGLS. Informally, an aggregate signature scheme allows multiple signatures to be aggregated into one signature with respect to the public keys of the signers and the signed messages. The BGLS scheme consists of five algorithms: *KeyGen*, *Sign*, *Verify*, *Aggregate* and *AggVer*. Any principal $P$ uses $KeyGen$ to generate the private and public key pair ($\chi$,$v$) such that $v = g_2^\chi$ where $g_2 \in G_2$, $\chi$ is the private key and $v$ is the public key. The $Sign$ algorithm computes the signature on input message $m_i$ in $G_1$ by a full-domain hash function $h : \{0,1\}^* \rightarrow G_1$. The output $\sigma_i = h(m_i)^\chi \in G_1$ is the signature for $m_i$. The *Aggregate* algorithm aggregates the signatures $\sigma_1, \sigma_2, \ldots, \sigma_t$ for $t$ different messages $m_1, m_2, \ldots, m_t$ into one signature $\sigma = \prod_{i=1}^{t} \sigma_i$. The $AggVer$ algorithm verifies a signature and works like the *Aggregate* signature algorithm. For a set $m_1, m_2, \ldots, m_t$ of different messages, and public keys $v_1, v_2, \ldots, v_t$ and a signature $\sigma$, the verifier checks if $e(\sigma, g_2) = \prod_{i=1}^{t} e(h_i,v)$, where $h_i = h(m_i)$ and $e$ is the bilinear mapping.

# 4 Protocols for the Multy-factor Verification of Strong Identity Attributes

In this section, we present the protocols for multi-factor strong identity attribute verification. We first introduce the notion of identity records (IdRs) that provide a representation of user identity attributes. Then, we introduce the protocol for strong identity attributes enrollment that consists of creating secure commitments and in signing them with the private key of the registrar. Finally, we present the protocol to create and verify the aggregate ZKPK of strong identity attributes' committed values.

## 4.1 Identity Records

As we mentioned, each principal $P$ in a federation has associated one or more IdRs, each recorded at some registrar in the federation. Each IdR in turn consists of several identity tuples, denoted as $\tau_i$. Each identity tuple is associated with one strong identity attribute and records all information related to the verification of this identifier at the time of use. In particular, a strong identity attribute $m$ is associated with a secure commitment denoted as $M$ that is signed by the registrar upon enrollment. The signature on $M$, denoted by $\sigma$ in the paper, is part of the identity tuple associated with $m$. $M$ is computed as $g^m h^r$, where $g$ and $h$ are generators in group $G$ of prime order $q$. $G$ and $q$ are public parameters of the registrar and $r$ is chosen randomly from $\mathbb{Z}_q$. $m$ is also tied to a set of *weak identity attributes*, denoted by $\{w_1, \ldots, w_k\}$. For example, assume 4040330043794877 to be a credit card number and Bob and Smith be the first and last name of an individual. Here, 4040330043794877 is the strong identity attribute value, while Bob and Smith are the associated weak identity attributes. All strong identity attributes' commitments and weak identity attributes are tagged with an attribute descriptor tag and two types of assurance, namely *validity assurance* and *ownership assurance*. Validity assurance corresponds to the confidence about the validity of the identity attribute based on the verification performed at the identity attributes original issuer. Ownership assurance corresponds to the confidence about the claim that the principal presenting a given identity attribute is its true owner. There are four levels of assurance: absolute assurance, tagged as A, corresponding to the absolute certainty about the claim; reasonable assurance, tagged as B, corresponding to case when one or more assertions from trusted parties exist regarding the certainty of the claim; unknown assurance, tagged as U, when there is no information to assert the certainty of the claim; and false assurance, tagged as F, denoting that the claim is incorrect. We assume that absolute validity of a given strong identity attribute can only be determined by authorities which have issued the strong identity attributes. This corresponds to value A of the validity-assure of the associated strong identity attribute. Instead, we mark as B the validity assurance of a strong identity attribute the validity of which has been asserted by a principal, whose identity record has a validity assurance set to A. If no entity other than the principal

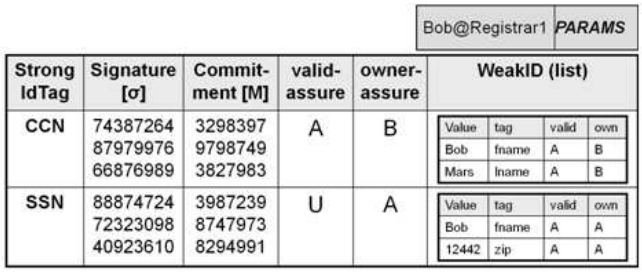



**Fig. 1.** Simplified graphical representation of an IdR

supports the validity of the strong identity attribute, this attribute is marked with unknown assurance U.

With reference to Example 1, Figure 1 shows an example of an IdR. Here the principal is known as Bob@Registrar1 and has enrolled two strong identity attributes, namely a CCN and SSN.

### 4.2 Enrollment of Strong identity attributes

1. *Registrar parameters.* The registrar runs parameter generation algorithm $GenKey$ that picks a prime $q$ and three multiplicative groups $G_1$, $G_2$, $G_T$ of prime order $q$. Also two generators $g_1$, $h_1$ in $G_1$ such that $log_{g_1} h_1$ and a $G_2$ group generator $g_2$ are returned by $GenKey$. Then, the registrar runs algorithm $KeyGen$ to generate the secret key $\chi$ that is a random number from $\mathbb{Z}_q$ and the public key $v = g_2^{\chi}$. The resulting set of parameters is ($G_1$, $G_2$, $G_T$,$g_1$,$h_1$,$g_2$,$v$).
2. *Commitment of a value* $m \in \mathbb{Z}_q$. The principal chooses a value $r \in \mathbb{Z}_q$, and computes $M = g_1^m h_1^r$.
3. *Zero-knowledge proof of the committed value.* The principal gives ZKPK of opening the commitment $M$ to the registrar:

$$PK\{(m, r) : y = g_1^m h_1^r, m, r \in \mathbb{Z}_q)\}$$

4. *Signing of the committed value.* After performing the security checks on the committed value (namely the local consistency and federation duplicate detection), the registrar executes the $Sign$ algorithm on the commitment $M$ to output $M^{\chi}$ as the signature where $\chi$ is the secret key of the registrar.

### 4.3 Aggregate zero-knowledge proof of knowledge (AgZKPK)

Suppose that a principal $P$ requests a service from a $SP$ which requires $P$ to first authenticate by proving that it knows how to open a specified set of commitments. To indicate this set of commitments a set of tags is given which is denoted by $\pi_{SP}$. The protocol that provides aggregate proof of knowledge of the commitments corresponding to $\pi_{SP}$ is composed of the following steps:

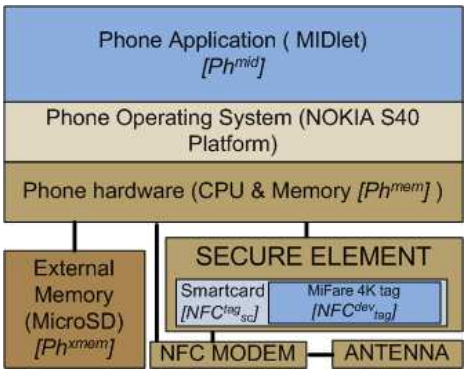



**Fig. 2.** Nokia NFC cellular phone components.

1. *Principal's aggregation.* Let $\sigma_1, \ldots, \sigma_t$ be the signatures corresponding to the strong identity attributes in $\pi_{SP}$. The principal aggregates the signatures into $\sigma = \prod_{i=1}^{t} \sigma_i$, where $\sigma_i$ is the signature of committed value $M_i = g_1^{m_i} h_1^{r_i}$. It also computes $M = \prod_{i=1}^{t} M_i = g_1^{m_1+m_2+\ldots+m_t} h_1^{r_1+r_2+\ldots+r_t}$. Finally, the principal sends $\sigma$,$M$, $M_i$, $1 \leq i \leq t$, to the verifier.
2. *Zero-knowledge proof of aggregate commitment.* The principal and the verifier SP carry out the following ZKP protocol:

$$PK\{(m, r) : y = g_1^m h_1^r, m, r \in \mathbb{Z}_q)\}$$

   where $m = m_1 + m_2 + \ldots + m_t$ and $r = r_1 + r_2 + \ldots + r_t$.
3. *Verification of aggregate signature.* After the verifier accepts the zero-knowledge proof of the commitments, it checks if the following verifications succeed: $M = \prod_{i=1}^{t} M_i$ and $e(\sigma, g_2) = e(M, v)$.

## 5 NFC implementation of the Multy-factor Identity Attribute Verification Protocol

In this section we first describe the main components of the Nokia 6131 NFC cell phone and then we present some details about the implementation of the multi-factor identity attribute verification protocol.

### 5.1 NFC Cellular Phone Architecture

We have developed our portable multi-factor identity attribute verification protocol on the Nokia 6131 NFC cell phone ($Ph^{NFC}$) [13]. We assume that the SPs have a NFC reader (denoted as $NFC^{SP}_{reader}$) which transmits and receives messages from the NFC cellular phone. The phone is integrated with a NFC device and thus contains both reader and writer for the embedded smart card and tags

that directly communicate with SP's reader. $Ph^{NFC}$ 's components are shown in Figure 2.

The main software component for managing strong identity attributes is the MIDlet suite. The MIDlet suite consists of a Java Application Descriptor (JAD) and a MIDlet. A MIDlet (denoted by $Ph^{mid}$) is a Java program that runs on the Java Virtual Machine(JVM) enabled mobile device. The JAD controls possible permissions that the MIDlet can have. A $Ph^{mid}$ is installed onto a phone and operates in a sandbox [16] so that different MIDlets are isolated from each other. The cellular phone has a secure element which can only be accessed by MIDlets signed by a trusted third party; these MIDlets should know the access key. The secure element consists of two main components, namely the Mifare tag ($NFC_{tag}^{dev}$) and Smartcard ($NFC_{sc}^{dev}$).

### 5.2 Implementation

In this section we describe how we have implemented the multi-factor identity attribute verification protocol on the Nokia 6131 NFC cell phone. We store the users' IdR in the external phone memory $Ph^{xmem}$, while the secret $r$ used to compute the secure commitments is saved in $NFC_{tag}^{dev}$. We have implemented a MIDlet that creates the AgZKPK. The MIDlet execution is triggered when the user's cell phone tag $NFC_{tag}^{dev}$ captures the verification policy sent by the SP's $NFC_{reader}^{SP}$ [1] and the $NFC_{tag}^{dev}$ transfers this policy to the cell phones main memory $Ph^{mem}$. The MIDlet retrieves from $Ph^{xmem}$the commitments corresponding to the strong identity attributes requested by the verification policy. Then, the MIDlet runs a new MIDlet which is executed in a protected domain with restricted permissions. This is necessary because the new MIDlet uses cryptographic secrets associated with the strong identity attributes to create the aggregate zero knowledge proof AgZKPK. Once the AgZKPK is computed, the MIDlet sends it to the main MIDlet. Upon receiving the AgZKPK, the main MIDlet transfers it to the $NFC_{tag}^{dev}$ so that it can be read by the $NFC_{reader}^{SP}$(see Figure 3).

The MIDLets developed to generate the AgZKPK run on Java 2 Micro Edition (J2ME), a subset of Java 2 Standard Edition (J2SE), which provides environments and APIs for mobile and embedded devices. Since J2ME is aimed at hardware with limited resources, it contains a minimum set of class libraries for specific types of hardware. In our AgZKPK implementation on conventional non-mobile platforms, we used the java.math.BigInteger and java.security.SecureRandom class defined in J2SE to implement secure commitments, but both java.math and java.security package are not supported in J2ME. Therefore, we have used the third-party cryptography provider BouncyCastle [1], a lightweight cryptography APIs for Java and C# that provide implementation of the BigInteger and SecureRandom classes. In addition, because of the limited memory size of mobile phone, we reduced the MIDlets' code size by using code obfuscation techniques

[1] The NFC reader is a device that can transmit as well as receive data using NFC technology.

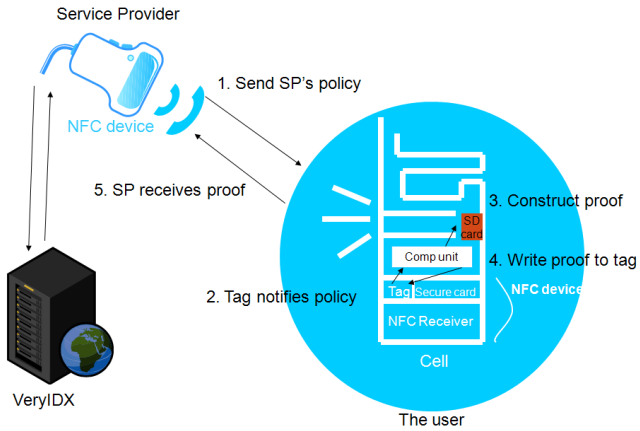



**Fig. 3.** Interactions between VeryIDX NFC module and SP card reader

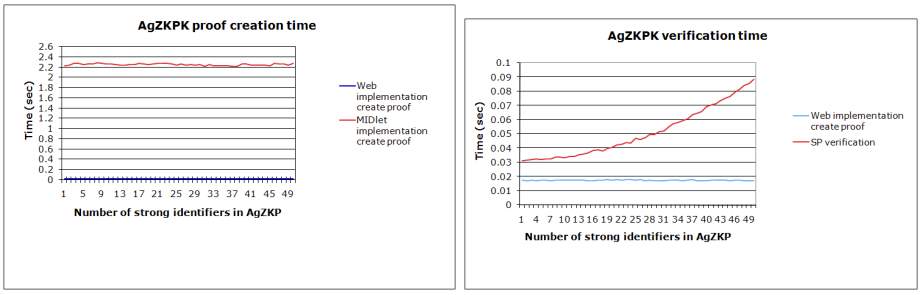
provided by Sun's NetBeans IDE. Code obfuscation allows one to reduce a file size of 98% by replacing all Java packages and class names with meaningless characters. For example, a file of size 844KB can be reduced to a size of 17KB.

Moreover, the MIDlets must have read and write privileges on the user's phone tag $NFC_{tag}^{dev}$ in order to enable the communication with the SP's NFC reader $NFC_{reader}^{SP}$. In fact, the SP's verification policy is saved in $NFC_{tag}^{dev}$ and then passed to the MIDlet to create the proof. Then, the created AgZKPK is stored in $NFC_{tag}^{dev}$ in order to be read by the SP $NFC_{reader}^{SP}$. In order to allow the MIDlets to access $NFC_{tag}^{dev}$, the MIDlets must be signed. To sign the MIDlets we used the Carbide.j tool [2] provided by Nokia that requires a code signing certificate released by a certification authority (CA) to generate the signature.

## 6 Experimental Results

In this section we present the results of the tests we have performed to evaluate the performance of the multi-factor identity attribute verification protocol implementation on the mobile phone. An aspect that might influence the performance of our protocols is the number of strong identity attributes that are aggregated and verified. Therefore, we have measured the time that the mobile client application takes to create the aggregate ZKPK and the time that SP's interface takes to perform the verification by varying the number of strong identity attributes that are verified from 1 to 50. We have compared the execution time to create the aggregate ZKPK on the mobile phone with the time to perform the same operation on the VeryIDX web-based implementation [3].

Figure 4 (a) reports the times required by the VeryIDX mobile phone implementation and by the web-based protocol implementation for generating the aggregate zero knowledge. In both cases, the AgZKPK protocol takes almost




(a) AgZKPK Creation on Midlet versus Web-based implementation (b) AgZKPK Verification versus Creation

**Fig. 4.** Experimental results

constant time for the ZKPK generation even if the number of identity attributes being proven increases. The reason is that the AgZKPK only requires a constant number of exponentiations [4]. Moreover, as expected, the time to create the proof on the mobile phone is higher than the time to perform the same operation on the web-based implementation due to the phone's limited computing power. The average time for the creation of an aggregate proof on the mobile phone is 2.257 seconds, while on the web-based application is around 0.02 seconds. Figure 4 (b) reports the time that the SP application takes to perform the strong identity attributes verification. Notice that the verification time linearly increases with the number of strong identity attributes to be verified. The reason is that during the verification the SP is required to multiply all the commitments to verify the resulting aggregate signature.

## 7 Related Work

In this section we discuss related work on the use of cellular phones for e- and m-commerce transactions involving identity attributes and other recent developments in mobile identity management initiatives.

With the advent of high-speed data networks and feature-rich mobile devices, the concept of *mobile wallet* [12, 5] has gained importance. A seminal work introduced the concept of wallets with observers [8] enabling off-line digital cash and credentials to be used in commercial settings. A major difference of our approach is that it does not require an observer, as the integrity of the strong identifiers is based on the signature of the registrar on the strong identifiers. The addition of the observer would, however, be beneficial if the usage of the strong identity attributes were constrained for example by the number of times of use.

Other mobile identity management initiatives have gained importance with the rapid adoption of second-generation mobile telecommunication systems, leading to the growth of m-commerce [14, 11]. Two critical specific factors in this domain are usability and trust. Several approaches to enhance usability of mobile

devices have been proposed [9]. Trust on the device comprises of several security and privacy properties such as confidentiality, integrity, user control and minimal disclosure of the identity data stored on such devices. One approach to mobile IdM is based on the GSM [14]. GSM based IdM uses the GSM infrastructure and the Subscriber Identity Module (SIM) as the underlying platform.

The Secure Electronic Transaction (SET) [15] 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 messages assure both confidentiality and integrity of the messages among card holders, merchants and payment gateway whereas our protocol is designed to assure integrity between service providers and registrar. SET authenticates the identity of the cardholder and the merchant to each other because both of them 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 [2].

## 8 Conclusion

This paper proposes protocols for managing identity attributes in cellular devices and supporting their secure and privacy preserving usage. The protocols are based on aggregate zero knowledge proof and aggregate signature on strong identity attributes' commitments. We have implemented the protocols on the Nokia NFC cellular phones and we have shown that the execution time to create the aggregate proof of knowledge is almost constant with respect to the number of strong identity attributes being aggregated. As future work we plan to extend our approach in several directions. A first direction is to adopt Shamir's secrete sharing scheme to protect the cryptographic secret $r$ used to compute Pedersen commitments associated with strong identity attributes. A second direction is the support of more sophisticated verification policies.

## 9 Acknowledgements

This material is based in part upon work supported by the National Science Foundation under the ITR Grant No. 0428554 "The Design and Use of Digital Identities" and upon work supported by the U.S. Department of Homeland Security under Grant Award Number 2006-CS-001-000001, under the auspices of the Institute for Information Infrastructure Protection (I3P) research program. The

[2] Only SPs and registrars must have certificates.

I3P is managed by Dartmouth College. The views and conclusions contained in this document are those 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.

## References

1. Bouncy Castle Crypto APIs, http://www.bouncycastle.org/
2. Development tools, http://www.forum.nokia.com/main/resources/tools_and_sdks/carbide/index.html
3. Bhargav-Spantzel, A., Woo, J., Bertino,E.: Receipt management- transaction history based trust establishment, In Proceedings of the 2007 ACM workshop on Digital identity management, pp. 82–91, New York, NY, USA.
4. Bhargav-Spantzel, A., Squicciarini, A.C, Xue, R., and Bertino, E.: Practical Identity Theft Prevention using Aggregated Proof of Knowledge, Technical report CERIAS TR 2006-26, 2006.
5. Boly, J., Bosselaers, A., Cramer, R., Michelsen, R., Mjolsnes, S., Muller, F., Pedersen, T.P, Pfitzmann, B., de Rooij, P., Schoenmakers, B., Schunter, M., Vallee, L., Waidner, M.: The ESPRIT Project CAFE - High Security Digital Payment Systems, ESORICS, pp. 217–230,1994.
6. Boneh, D., Gentry, C., Shacham, H., Lynn, B.: Aggregate and verifiably encrypted signatures from bilinear maps, In Proceedings of Advances is Cryptology , Eurocrypt'03, LNCS. Springer-Verlag, 2003.
7. Camenisch, J., Stadler, M.: Efficient Group Signature Schemes for Large Groups, Advances in Cryptology, CRYPTO '97, pp. 410–424, 1997.
8. Chaum, D.: Security without identification: transaction systems to make big brother obsolete, Communications of the ACM, 28(10), pp. 1030–1044, 1985.
9. Dix, A., Rodden, T., Davies, N., Trevor, J., Friday, A., Palfreyman, K. :Exploiting space and location as a design framework for interactive mobile systems, ACM Transactions on Computer Human Interaction, 7(3), pp. 285–321, 200, New York, NY, USA.
10. Fiege, U., Fiat, A., Shamir, A.: Zero knowledge proofs of identity, In Proceedings of the nineteenth annual ACM conference on Theory of computing, pp. 210–217, New York, NY, USA, 1987.
11. Jendricke, U., Kreutzer, M., Zugenmaier, A., Mobile Identity Management, UBICOMP 2002: Workshop on Security in Ubiquitous Computing, 2002.
12. Mjolsnes, S.F., Rong, C.: Localized Credentials for Server Assisted Mobile Wallet, In Proceedings of International Conference on Computer Networks and Mobile Computing, Los Alamitos, CA, USA, 2001.
13. Near Field Communication Forum, http://www.nfc-forum.org
14. Rannenberg, K. : Identity management in mobile cellular networks and related applications, Information Security Technical Report, Johann Wolfgang Goethe University Frankfurt, January 2004.
15. SET Secure Electronic Transaction Specification Book 1: Business Description, 1997.
16. Wolfe, A.: Toolkit: Java is Jumpin',Queue, 1(10), pp. 16–19, 2004