PRIVACY ENHANCING TECHNIQUES FOR DIGITAL IDENTITY MANAGEMENT

by

Hasini T. Urala Liyanage Dona Gunasinghe

A Dissertation

Submitted to the Faculty of Purdue University

In Partial Fulfillment of the Requirements for the degree of

Doctor of Philosophy

Department of Computer Science

West Lafayette, Indiana

August 2021

ACKNOWLEDGMENTS

I would like to thank my advisor, Professor Elisa Bertino, for her advices, thoughtful guidance and support throughout. I would not have come this far in grad school if it were not for her continuous encouragement and help. I would like to thank Professor Mikhail Atallah and Hugo Krawczyk for their advice and time during my research collaboration with them. Their inputs were very valuable to enrich my research work. I am thankful to all my committee members for their valuable feedback and time. I am grateful to all the great faculty members at Purdue whom I learned from. I extend my sincere gratitude to all the teachers in my life.

I would like to thank Professor Dunsmore with whom I was privileged to work as a teaching assistant and whose kindness and care made my career as a TA a very pleasant and rewarding one during the grad school. I am thankful to my colleague Fang-Yu Rao and other researchers around the world, namely, Daniel Escudero, Ahmed Kosba and Fabrice Benhamouda, who generously shared their time and inputs during the fruitful research discussions. I am thankful to the Emil Stefanov fellowship, Bisland dissertation fellowship and IBM PhD fellowship for funding my research. I would like to thank Dr. Ashish Kundu for serving as the mentor during the IBM fellowship and sharing his valuable feedback and time.

I am thankful to everyone who helped me in numerous ways during the time of applying to grad school, including Dr. Sanjiva Weerawarana, Dr. Srinath Perera, Dr. Amal Shehan Perera and Prabath Siriwardena. I am thankful to everyone who helped me in numerous ways during the grad school time, including my doctors who took care of my health, Sri Lankan community at Purdue, all my friends and colleagues, specially Sandi, for their great emotional support and friendship, and the administrative staff of the CS department and ISS at Purdue for their various help. Reaching this milestone would not have been possible without all their support.

Words are not sufficient to express my sincere heartfelt gratitude and appreciation towards my family; my mother Pathma Pathirana, my father Chandrananda Gunasinghe, my mamma Sumana Pathirana, my husband Thilina and my sister Umesha for giving me unconditional love, support and encouragement during both ups and downs, although we were living thousand miles apart. I am forever grateful for them for everything they did for me. Last but not least I offer my deepest heartfelt respect and gratitude to my teacher of life, Ajahn Brahm whose talks on life teachings and guided meditations helped me a lot to pull through hard times.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

Proving and verifying remotely a user's identity information have become a critical and challenging problem in the online world, with the increased number of sensitive services offered online. The digital identity management ecosystem has been evolving over the years to address this problem. However, the limitations in existing identity management approaches in handling this problem in a privacy preserving and secure manner have caused disruptions to users' digital lives and damages to revenue and reputation of service providers.

In this dissertation, we analyze different areas of the identity management ecosystem in terms of privacy and security. In our analysis, we observe three critical aspects to take into account when identifying the privacy and security requirements to address in identity management scenarios, namely: i) protecting privacy and security of digital identity and online transactions of users; ii) providing other stakeholders with assurance about user identity information and accountability of transactions; iii) preserving utility (e.g. accuracy, efficiency and deployability). We show that existing authentication models and identity management protocols fail to address critical privacy and security requirements related to all these three aspects, mainly because of inherent conflicts among these requirements. For example, existing authentication protocols, which aim to protect service providers from imposters by involving strong authentication factors, such as biometrics, fail to protect privacy and security of users' biometrics. Protecting an identity management system against counterfeits of identity assets, while preserving unlinkability of the transactions carried out using the identity assets, is another example of conflicting yet critical privacy and security requirements. We demonstrate that careful combinations of cryptographic techniques and other technologies make it feasible to design privacy preserving identity management protocols which address critical and conflicting requirements related to the aforementioned three aspects. Certain techniques, that we have developed for these protocols, are independent contributions with applications beyond the domain of digital identity management. We validate our contributions by providing prototype implementations, experimental evaluations and security proofs.

1. INTRODUCTION

1.1 Problem Statement

Access to services offered by online service providers (SPs) is controlled by identity verification processes. Depending on the sensitivity of the service offered, the SPs may require verification and storage of various identity information of the user. Repositories storing user identity information, such as credit card numbers, social security numbers, email addresses, etc., have become targets of attacks as evident from the recent history of cyber crimes [1]–[3]. The cyber attack on the Wired writer Mat Honan [4] shows that the attackers can exploit multiple issues in the existing digital identity management ecosystem, such as linkability of different accounts of the same user held at different online SPs, and weaknesses of online authentication and identity verification mechanisms used for account recovery, in order to wipe out the entire digital life of an individual. Apart from external attackers, the SPs themselves track users' online behavior via digital identities for various purposes, such as targeted advertising, in the interest of the SPs' business model [5], [6]. Prominent privacy protection regulations, such as the General Data Protection Regulation (GDPR) [7], have emerged to protect users' online privacy by considering a user's transactional patterns as personal data and prohibit the tracking such personal data.

It is thus critical that privacy and security requirements should be addressed in the design of digital identity management systems and protocols. However, it is challenging to satisfy multiple such requirements at the same time due to their conflicting nature. For example, biometrics is preferred over passwords, as a strong authentication factor. However, since biometrics is permanently attached to a user's identity, biometric data should not be revealed to all the third party SPs that the user interacts with. On the other hand, SPs need to have confidence on the verification of the biometric identity. Although centralized storage and verification [8], where a trusted party stores and verifies a user's biometrics on behalf of third party SPs, address those concerns, such an approach undermines the user's transaction privacy, because such centralized party is involved in every transaction of the user. As another example, the exchange of users' identity information between different parties in the identity management ecosystem should incorporate mechanisms to provide anonymity, unlinkability and selective disclosure of identity information, in order protect the users' digital life from the aforementioned types of attack. On the other hand, it is also critical to protect SPs from abuses or frauds perpetrated by anonymous parties, by ensuring ownership of the exchanged identity information and accountability of transactions. The existing mechanisms, which provide anonymity by employing a centralized party to mediate the identity exchange transactions [9], [10], have introduced even more privacy and security issues (see [11]). Mechanisms, which provide unconditional unlinkability, either do not provide accountability for SPs (e.g. [12]) or introduce additional parties into the system which undermines deployability, and involve complex identity tokens which undermines efficiency of identity verification (e.g. [13], [14]).

1.2 Focus of the Dissertation

This dissertation focuses on demonstrating the following statement: *careful combination of advanced cryptographic techniques and other technologies makes it feasible to design identity management protocols which fulfill critical yet conflicting requirements related to three main aspects of the digital identity management ecosystem, namely: i) protecting privacy and security of digital identity and online transactions of the users; ii) providing other stakeholders with assurance about user identity information and accountability of transactions; iii) preserving utility (e.g. accuracy, efficiency and deployability).*

In this dissertation, we analyze two main areas of digital identity management: i) privacy preserving and biometrics based online authentication; and ii) privacy preserving exchange and verification of identity in online transactions. Based on our analysis, we identify the critical requirements and the shortcomings of the existing protocols in fulfilling such requirements. Next, we introduce our techniques and protocols, which combine in novel ways advanced cryptographic techniques, such as zero knowlege proofs, secure multiparty computation (SMPC) techniques and threshold cryptography, and other technologies, such as biometric matching, machine learning, blockchain and fine grained policy based identity verification, in order to provably preserve the users' privacy, while satisfying other critical yet conflicting requirements. Some of the techniques that we have developed have applications beyond the domain of digital identity management, as we highlight later on this chapter. We have also implemented prototypes, carried out experimental evaluations and developed security proofs for validating our approaches.

In what follows we provide an overview of our research and contributions to digital identity management.

1.3 Privacy Preserving and Biometrics based Online Authentication

1.3.1 Background

Authentication refers to verifying whether an individual is who she claims to be. The stronger the authentication factors used to prove such claim, the harder it is to impersonate. Existing online authentication approaches are based on either one of two main authentication models. The first model is the SP-specific authentication model where each SP implements its own authentication system. A major drawback of this model is that the users have to store sensitive identity information with multiple SPs, some of which may not be trusted to store such information securely. This model also imposes high costs to the SPs as they have to securely maintain identity repositories which are targets of attacks (e.g. [1]–[3]). These concerns amplify with the use of biometrics (e.g. [15]) as a stronger authentication factor, because unlike passwords biometrics is permanently associated with the user's identity. As today mobile phones include various biometric sensors, major SPs, such as banking institutions [16], credit card companies [17] and e-commerce organizations [18], are increasingly adopting biometrics based online authentication and such SPs usually implement their proprietary biometrics based authentication schemes in the SP-specific authentication model.

The second model is the identity provider (IDP)-centric authentication model. In this model, the user maintains credentials at a centralized party, e.g. the IDP, who is trusted by both the user and the SP(s). Whenever the user needs to authenticate to an SP, the SP redirects the user to the IDP with whom the user authenticates, and the IDP vouches for the successful authentication. This model has increasingly been popular as many SPs today provide options for the users to authenticate using their credentials held at major IDPs such as Google, Facebook, LinkedIn, etc., via standardized protocols like OpenID [19] and

OAuth [20]. This model does not have the drawbacks of the SP-specific model. Today there are commercial products [8] which specifically support biometrics based authentication in the IDP-centric model. However because the IDP is involved in every transaction, the IDP can learn about and link the user's transactions, in the interest of the IDP's business model, as shown by recent privacy breaches [5], [6], compromising the user's transactional privacy. We thus need a stronger online authentication protocol based on biometrics, which has the advantages of the IDP-centric model, but which does not involve the IDP during the authentication, so that it preserves the user's biometric privacy from the SPs and transactional privacy from the IDP.

1.3.2 Our Contributions

Unlike previous authentication models, we propose a user-centric authentication model, for privacy preserving and biometrics based online authentication. In our model, the user first enrolls her biometrics at a trustred biometric certification authority (BCA). The BCA cryptographically encodes the user's enrollment biometrics in an enrollment token signed by the BCA and hands it over to the user along with certain other authentication artifacts signed by the BCA. The user submits such enrollment token to different SPs for authentication verification, without involving the BCA in the authentication. The authentication verification phase enables the user to prove her ownership of the biometric identity encoded in the enrollment token, by using a fresh capture of her biometrics taken at the authentication, without revealing any sensitive biometric information to the SP. Since the BCA is not involved in the authentication, the user's transaction privacy is protected from the BCA. Note that the BCA does not become a single point of attack though, as the BCA does not store any sensitive information involved in the enrollment phase.

The two main challenges in designing a biometric based online authentication protocol in the user-centric model are: i) preserving accuracy and efficiency of authentication when biometric matching is performed using cryptographically encoded enrollment biometrics; and ii) protecting the authentication system from stolen and compromised user devices which store the authentication artifacts. We propose two protocols for biometrics based online authentication in the user-centric model: i) PEBASI [21]; and ii) PrivBioMTAuth [22]. Both protocols address the aforementioned challenges in completely different ways and, therefore, are suitable for deployment in different contexts. In what follows, we introduce the two protocols and then describe the different contexts in which each of them is preferred over the other.

PEBASI: A Privacy Preserving and Efficient Biometric Authentication Scheme based on Irises

The concrete instantiation of the authentication protocol in PEBASI is defined using iris as the underlying biometric trait, since iris is considered as a very reliable biometric trait which is used for on-site passenger identification in airports [23]. In PEBASI, we use pseudo one-time pad (POTP) [24] to encrypt the user's iris biometrics recorded in the enrollment token since it enables efficient secure biometric matching during authentication. In biometric matching, one first computes the distance between two biometric feature vectors and then compares the result against a pre-defined threshold. The computation of biometric matching is performed using SMPC techniques, in order to preserve biometric privacy as well as accuracy of the authentication result. The authentication protocol in PEBASI is thus organized into two phases:

1. *Secure computation of the distance between the encrypted biometrics recorded in the enrollment token and the biometrics captured at the authentication.* This phase is designed using arithmetic circuits as secure multiplication is known to be the fastest using arithmetic circuits. For this phase, we design novel techniques to enable the computation of normalized Hamming distance on encrypted iris biometrics, as certain required operations are not preserved by POTP encryption.

2. *Secure comparison of the distance against the threshold.* This phase is designed using Yao's garbled circuits (GC) as secure comparison is more efficient using Boolean circuits.

Both phases perform *computations over a ring.*

PEBASI is designed to be a three-factor authentication scheme s.t. a malicious user cannot succeed in impersonating by only submitting a stolen biometric template of the genuine user. Furthermore, we enforce a potentially malicious user to follow the prescribed protocol in two different ways, in two versions of the protocol. Version 1 (V1) assumes the ability to verify the integrity of the authentication software running in the user's device via attestation techniques enabled in the modern user devices. This approach allows us to use cryptographic primitives that provide only semi-honest security and thereby allows V1 to serve as an important first step towards an efficient privacy preserving online biometric authentication protocol. Version 2 (V2) enforces the user to follow the prescribed protocol purely based on cryptographic primitives which provide active security. In V2, we introduce techniques to securely bridge phase 1 and phase 2 against a malicious user.

Phase 1 of V1 is designed using a hybrid approach for secure multiplications (see Section 2.3.2), involving an efficient construction of Oblivious Linear Function Evaluation (OLFE) [25]. While several approaches have been proposed for privacy preserving biometric matching [26]–[31] mainly in the context of biometric identification, Demmler et al. [31] have proposed the most efficient approach which uses Beaver's multiplication triples (MTs) [32] for secure multiplication operations. Beaver's MTs is considered the state-of-the-art technique for secure multiplication in the online phase of secure multiparty computation (SMPC) protocols [33]–[35]. Compared to the approach by Demmler et al. [31], our hybrid approach used in phase 1 of V1 has an average improvement of 35% in the end-to-end execution time. This hybrid approach for secure multiplication is of general interest to SMPC protocols.

Our contributions from PEBASI:

1. Definition of a user-centric online authentication scheme based on iris biometrics which preserves users' biometric privacy from SPs and transactional privacy from the BCA and which is secure against a malicious user.

2. Design of techniques enabling the computation of normalized Hamming distance over binary data encrypted with pseudo one-time pad (Section 2.2.2).

3. Introduction of a hybrid approach for secure multiplication. As shown in benchmarks of protocol V1, secure computation of distance in iris biometric matching using such hybrid approach outperforms the state-of-the-art approaches.

4. Design of techniques to securely bridge arithmetic and Boolean computation phases of an actively secure two party computation protocol, while enabling the integrity verification of arithmetic computation, without having to reveal the output of such computation.

PrivBioMTAuth: A Privacy Preserving Generic Authentication Scheme based on Biometrics

Unlike PEBASI, in PrivBioMTAuth, a user's biometric privacy is protected by encoding the user's enrollment biometrics in a cryptographic commitment and involving a zero knowledge proof of knowledge (ZKPK) protocol in the authentication verification phase. Previous approaches [36]–[38] have been proposed that use ZKPK for verification of static identities, such as email addresses. Designing a biometric authentication protocol using ZKPK is particularly challenging due to two main reasons. The first reason is that one needs to be able to generate the exact same secrets at both enrollment time and authentication time in order for the ZKPK based authentication to succeed. However, unlike static identities, biometrics is non-repeatable, i.e. the same biometric trait of the same user captured at two different times is not identical. Because of this, the genuine owner of the biometric identity might fail the ZKPK based authentication, resulting in an increased false rejection rate for the authentication system. The second reason is that ZKPK based authentication protocols are inherently vulnerable to a special type of impersonation attacks called Mafia Fraud [39], which needs to be prevented for strong assurance on ownership of the user's biometric identity.

PrivBioMTAuth addresses these two challenges by: i) deriving a unique, repeatable and revocable biometric identifier (BID) from the users' biometrics at the enrollment and training a machine learning classifier to output the user's BID when the user's biometrics captured at the authentication is given as inputs to the trained classifier; and (ii) including a key agreement mechanism tied to the authentication protocol to mitigate Mafia attacks. The proposed generic solution does not depend on any particular biometric trait, feature extraction mechanism or machine learning algorithm.

Our contributions from PrivBioMTAuth:

1. Definition of a user-centric online authentication protocol based on biometrics which is not tied to any particular biometric trait.
2. Design of techniques to derive a unique, repeatable and revocable BID from the user's biometric template.
3. Design of techniques to train the machine learning classifier s.t. the authentication system is not compromised via a stolen user device which stores the trained classifier.
4. Design of techniques to mitigate Mafia attacks in ZKPK protocols, which is an independent contribution.

As mentioned before, each of PEBASI and PrivBioMTAuth is preferred over the other, based on the context in which they are to be deployed, due to their unique characteristics. The biometric matching function used in PEBASI is independent of the number of users enrolled in the authentication system, so are the accuracy of the authentication result and the efficiency of the authentication protocol. This characteristic is desirable for large scale identity management systems in which the number of enrolled users may increase arbitrarily. In contrast, since the biometric matching function used in PrivBioMTAuth involves a machine learning classifier, its robustness is affected by the number of users enrolled in the system. Therefore, it is not preferable for large scale deployments. On the other hand, in PrivBioMTAuth, the trusted IDP, i.e. the BCA, is not required to provide any service to the user after the enrollment phase. In contrast, the authentication verification phase of PEBASI, which involves SPMC techniques, requires that certain pre-processed artifacts are supplied prior to the online phase, in order to make the online phase more efficient. In PEBASI, these artifacts are periodically supplied to the user by the BCA in a separate pre-processing phase. Furthermore, both PEBASI and PrivBioMTAuth do support different biometric traits such as iris, face and fingerprint. In PrivBioMTAuth, one only needs

Table 1.1. Table summarizing the comparison between the two schemes PEBASI and PrivBioMTAuth.

	PEBASI	PrivBioMTAuth
Enrollment Token	pseudo one-time pad/ shared secrets	Pedersen commitment
Authentication	SMPC techniques	zero knowledge proofs, trained machine learning classifier
Accuracy	same as the non-secure counterpart	depends on the robustness of the classifier
Involvement of the BCA	enrollment phase, artifact re-fill phase	enrollment phase
Preferred deployment	in large scale IDPs	in small to medium scale IDPs
Flexibility to support other biometric traits	static support	dynamic support

to plug-in the applicable feature extraction algorithm and the machine learning algorithm in order to support authentication using a particular biometric trait, while all the protocol steps remain unchanged. In contrast, in PEBASI, one needs to change certain protocol steps in order to support authentication using different biometric traits. Table 1.1 summarizes the comparison between the two schemes w.r.t multiple different aspects.

Therefore, in summary, PEBASI is more suitable in deployment in large scale IDPs, which publicly provide authentication services (e.g. Google, Facebook, LinkedIn, etc.) and which are available to provide the periodic service of pre-processed artifact re-filling, and in which the biometric trait used for authentication is pre-defined. On the other hand, PrivBioMTAuth is more suitable for deployment in small to medium scale IDPs which provide authentication services locally (e.g. employing organization of the user) and which are not available to provide any continued service after the enrollment phase, and in which the biometric trait used for authentication might need to be dynamically changed.

1.4 Privacy Preserving Exchange and Verification of Identity in Online Transactions

Apart from authentication, which is performed each time a user accesses an online service, the SPs verify a user's identity information at two main stages of an online transaction: i) during the trust establishment phase, in order to confirm a user's eligibility to consume a particular service (e.g. due diligence processes such as Know Your Customer (KYC) compliance verification in banking/financial services [40]); and ii) during the execution of the transaction, in order to confirm accountability of the transaction (e.g. verifying credit card information when making a purchase from an e-commerce service). In what follows, we introduce the two privacy preserving solutions named PrivIdEx [41] and RahasNym [42], [43] respectively, which we propose to address the key privacy and security requirements of identity exchange and verification in the aforementioned two stages of an online transaction.

PrivIdEx: Privacy Preserving and Secure Exchange of Digital Identity Assets

Customer due diligence processes of financial services, such as KYC compliance verification, require the SPs to perform background checks on the user to verify the user's status of credit score, Anti-Money Laundering (AML), Counter Terrorist Financing (CTF), Politically Exposed Person (PEP) [40], etc. Similar examples of such rigorous due diligence processes include, but not limited to, joining a new employer, applying for temporary visa in a foreign country, etc. Once an SP has verified the identity of a user, the package of information associated with such *verified identity* becomes an asset of the SP, which we refer to as *identity asset* (the National Strategy for Trusted Identities in Cyberspace (NSTIC) labels such verified identity information as belonging to LOA (Level Of Assurance) 2+ category [44]). On the other hand, since an identity asset contains personal information about a user, the user also becomes an owner of the identity asset, leading to having two legitimate owners per *identity asset*.

Currently, when a user needs to establish trust with different SPs who offer similar services, the user is treated as an "alien" by each SP and is required to go through a similar identity verification performed by each SP. These repeated and lengthy processes not only are costly, but also are inherently error-prone, causing inconvenience to both parties [45]. If there were a standard protocol through which different SPs could share the same *identity assets* of a user, that would result in substantial cost savings and notable convenience to both parties [40].

Let us consider the following **use case**: *The user Ursula first consumes financial services from bank A where bank A performs identity verification steps for KYC compliance on Ursula. Later Ursula needs to consume financial services from bank B as well. Bank B wants to know if Ursula has already performed KYC compliance verification and if so, both Ursula and bank B would like to re-use the corresponding identity asset.* However, in this use case, Ursula would not like to reveal to bank B which bank(s) she has interacted before, and would not like to reveal to bank A, which other bank(s) she is planning to be a customer of. Bank A and bank B themselves would also not like to reveal their identity to each other during potential identity asset exchange, due to competition in business. On the other hand, Ursula would not like the transactions she carries out with different banks based on the same identity asset to be linkable. Therefore, *anonymity of the parties who exchange the identity asset* and *unlinkability of the transactions* are key privacy requirements to be addressed when designing a protocol to facilitate identity asset exchange, in addition to protecting *confidentiality* of identity assets from external parties.

The existing identity management protocols, which facilitate exchanging users' identity information, do not address all the key privacy requirements. For examples, OpenIDConnect [19], an industry standard widely used by SPs to obtain a user's identity information from an IDP, does not preserve anonymity and unlinkability. Two nation-scale brokered identification systems built by the USA and UK governments, namely, Federal Cloud Credential Exchange (FCCX) [9] and GOV.UK Verify [10], respectively, focus on protecting from each other the anonymity of the parties who exchange users' identity information, in order to preserve users' privacy. Those systems, however, use a government managed broker to mediate the identity exchange transactions, in which case, the identity of the two exchanging parties is revealed to the broker, although the two parties stay anonymous to each other. Brandao et. al [11] have raised certain other privacy concerns on the introduction of such a centralized broker.

One of our goals is to avoid introducing such a centralized broker. Therefore, the protocol that we propose is designed to be executed in a decentralized identity management ecosystem backed by a permissioned blockchain network. Distributed trust implemented on the basis of the consensus protocol through which blockchain peers validate protocol executions eliminates the requirement of a centralized broker. Participants of the decentralized identity management ecosystem invoke the identity exchange transactions with pseudonyms, in order to preserve anonymity and unlinkability. However, when anonymity and unlinkability are enforced in confidential identity asset exchanges, without a mediating centralized party, it is challenging to achieve the required security properties, such as *correctness*, *ownership assurance* and *counterfeits elimination* of the identity assets, and optionally, *financial fairness* of the identity asset exchange transactions. This is because the participants, appearing with a new pseudonym in each round of the protocol execution, can violate such security properties. This problem can be related to the challenge of preventing double spending in bitcoin [46] and ZeroCash [47], without a centralized financial institute managing the payment transactions. However, unlike in bitcoin and ZeroCash, whose goal is to prevent double spending of cryptocurrency, which has a single owner at a time, our goal is to enable multiple exchanges of the same identity asset which has two owners, which poses a different set of challenges. We have designed the dedicated phases of the protocol to address such challenges by leveraging the power of certain cryptographic primitives such as threshold homomorphic encryption and ZK-SNARKS **QSP**.

Our contributions from PrivIdEx:

1. Definition of a protocol realizing privacy preserving and secure exchanges of identity assets in a decentralized identity management ecosystem.

2. Design of novel techniques to eliminate counterfeits of identity assets while preserving unlinkability.

3. Design and implementation of the core cryptographic building blocks of the protocol, i.e. the circuits required for zero knowledge proofs using ZK-SANRKs.

RahasNym: A Pseudonymous and Flexible Identity Verification System for Protecting against Linkability.

The main privacy problem associated with identity verification carried out during the execution of online transactions is linkability of the users' online accounts and transactions via the same identity information revealed to multiple SPs. Elaborating on the cyber attack on the Wired writer Mat Honan [4], mentioned in Section 1.1, the attackers used the last four digits of his credit card number stolen from his Amazon account to gain access to his iCloud account and deleted the data in all his Apple devices. In this serial attack, the attackers also hacked his Gmail and Twitter accounts.

As a basic solution to protect against linkability, one could use pseudonyms when interacting with various on-line systems. A pseudonym is a digital identifier used to identify an individual which is different from the individual's real name. However, linkability is still possible via the other unique identity information provided to multiple SPs in plain text. Therefore, pseudonym based identity management, which aims to protect against linkability, should be accompanied by mechanisms allowing the users to prove the ownership of valid identities without having to reveal them in plain text, such as ZKPK schemes [48]. On the other hand, users may submit stolen credit card details or overuse a certain privilege granted to a particular individual by associating it with multiple different pseudonyms. Therefore, SPs need assurance about the ownership of the users' identities and accountability of the transactions. The level of assurance required w.r.t different properties of pseudonymous identity verification varies depending on the nature of the transaction as well as the preferences of the users and the SPs. Hence, we need flexible identity management systems which accommodate identity verification according to such varying requirements.

In order to prevent privacy breaches resulting from identity linkability, the use of anonymous credentials has been proposed [12], [13], [49]–[51] based on different cryptographic techniques. However, each of such previous approaches addresses only a subset of the critical requirements. For examples, the approach by Chaum [12] supports unconditional untraceability at the cost of accountability assurance. His group signature scheme [49] facilitates anonymous identity verification of an individual belonging to a group, at the cost of scalability. Idemix [13] supports unconditional unlinkability at the cost of involving additional parties in the revocation and inspection, and making the identity tokens bulky and the associated proof protocols complex [14]. Therefore, widely used on-line transaction systems, such as online merchants, have not yet adopted such schemes to enable users to carry out online transactions in a privacy preserving manner. Furthermore, to date there is no flexible identity management system that enables both the parties involved in a transaction to specify, negotiate and fulfill their varying requirements related to identity verification.

The aim of RahasNym is to develop a practical pseudonymous identity management system which protects user's privacy against linkability in online transactions while safeguarding SPs against frauds, and which can be adopted by existing transaction systems without much overhead in terms of architectural changes and performance. RahasNym provides a policy framework with a simple and expressive policy language to specify preferred identity verification requirements and a suite of cryptographic protocols to achieve them.

Our contributions from RahasNym:

1. Identification of key requirements in pseudonymous identity management.
2. Definition of a policy framework for specification of such requirements.
3. Definition of a suite of protocols for implementing the specified identity verification policies.

1.5 Organization of the Dissertation

The remainder of this dissertation is organized as follows. Chapters 2 and 3 present our proposed privacy preserving and biometrics based online authentication protocols, PEBASI and PrivBioMTAuth, respectively. Chapter 4 presents PrivIdEx, our solution for privacy preserving exchange of confidential digital identity assets during the trust establishment phase of online transactions. Chapter 5 presents our privacy preserving solution, named RahasNym, for pseudonymous and flexible identity verification during the execution of online transactions. Chapter 6 discusses related work. Chapter 7 outlines the future research directions and concludes the dissertation.

2. PEBASI: A PRIVACY PRESERVING, EFFICIENT BIOMETRIC AUTHENTICATION SCHEME BASED ON IRISES

In this chapter, we present our proposed privacy preserving and biometrics based online authentication protocol, named PEBASI, which we introduced in Section 1.3.2. This chapter is organized as follows. Section 2.1 provides a high level overview of the proposed protocol along with the threat model that we consider and the security and privacy goals that we aim to achieve. Section 2.2 presents our approach and the protocol details. We present the details of the prototype implementation and the experimental evaluation in Section 2.3. In Section 2.4, we prove that the protocol achieves our security goals. Section 2.5 describes the cryptographic building blocks used in our protocol design. Section 2.6 describes how PEBASI can be extended to support other biometric traits such as face and fingerprint.

2.1 System Overview

The authentication scheme involves three parties: i) user; ii) BCA; iii) SP. Figure 2.1 illustrates the interactions that take place between these parties during the identity management life cycle. 1) The user enrolls her biometrics with the BCA. 2) The BCA hands over to the user, an enrollment token (ET), which includes the encrypted biometric information of the user, and other authentication artifacts signed by the BCA, to be used during authentication. 3) The user signs up at one or more SP(s) and establishes the foundation to perform future authentications. 4) The user performs biometric authentication as many times as needed with different SP(s), using her biometrics captured at the authentication and the authentication artifacts obtained from the BCA, without involving the BCA in the authentication. We assume that all the communication in PEBASI takes place over secure channels.

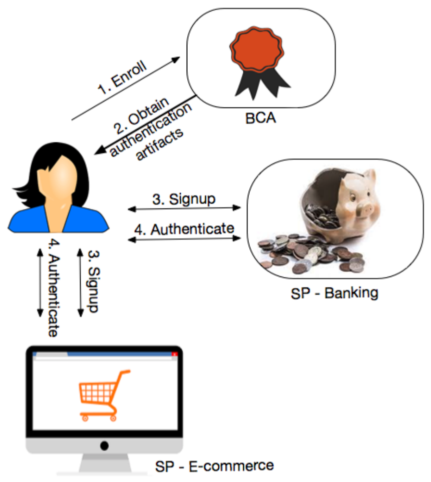


Figure 2.1. High level overview of the authentication scheme.

2.1.1 Threat Model and Security Goals

In the scope of our work, we consider the following security and privacy threats. 1) The BCA attempts to learn the user's transactional information with different SPs, via the protocol interactions. This is motivated by the business models of online IDPs, which may act as BCA(s), as revealed in recent privacy breaches [5], [6]. 2) The SP attempts to learn the user's sensitive biometric information via the protocol interactions. 3) A malicious user attempts to impersonate a legitimate user by: i) submitting a stolen biometric template by by-passing the biometric scanner, ii) deviating from the prescribed protocol to fake a successful authentication result.

We model the BCA and the SP as semi-honest adversaries who follow the prescribed protocol, but may attempt to infer sensitive information from the protocol execution. Note that the BCA and the SP want their business to work and they gain nothing by deviating from the protocol and making the output incorrect. Therefore, it is reasonable to model them as semi-honest adversaries. On the other hand, we model the user as a malicious adversary who may attempt to impersonate a legitimate user via either of the aforementioned attacks.

Accordingly, security and privacy goals of our protocol are as follows: 1) The authentication protocol of PEBASI achieves the same accuracy as its non-privacy preserving and non-secure counterpart. 2) The BCA does not learn any information about the user's transactions with different SPs via protocol interactions. 3) The SP does not learn any sensitive biometric information of the user via protocol interactions. 4) A malicious user does not succeed in impersonating a legitimate user, via either of the two attacks mentioned above.

2.2 Approach

2.2.1 Plain Iris Matching

Iris code [52] is the standard feature extraction algorithm used in commercial iris matching systems. Given an iris template F, the iris code algorithm outputs the extracted features and a binary mask of length n. The mask indicates which bits are valid (indicated by 1) and which bits are not valid in the feature vector. Two iris codes are considered matching if the normalized Hamming (NHD) distance between them is below a pre-defined threshold ϵ. Let X and Y denote the iris feature vectors extracted at the enrollment and the authentication respectively and M_I denote the mask associated with a given feature vector I, where $I \in \{X, Y\}$. The following equation represents the computation of NHD between X and Y.

$$NHD(X,Y) = \frac{\sum_{\mathrm{i}=1}^{n}(X[\mathrm{i}] \oplus Y[\mathrm{i}]) \wedge (M_X[\mathrm{i}] \wedge M_Y[\mathrm{i}])}{\sum_{\mathrm{i}=1}^{n} M_X[\mathrm{i}] \wedge M_Y[\mathrm{i}]}$$

The predicate: $NHD(X,Y) \leq \epsilon$ decides if X and Y are a match or not. Division is expensive to compute using SMPC techniques, therefore, the following representation of this predicate is used in secure iris matching protocols (e.g. Blanton et al. [28]).

$$\sum_{\mathrm{i}=1}^{n}((X[\mathrm{i}] \oplus Y[\mathrm{i}]) \wedge (M_X[\mathrm{i}] \wedge M_Y[\mathrm{i}])) \leq \epsilon . \sum_{\mathrm{i}=1}^{n}(M_X[\mathrm{i}] \wedge M_Y[\mathrm{i}]) \tag{2.1}$$

Let $m = \sum_{\mathrm{i}=1}^{n} M_X[\mathrm{i}] \wedge M_Y[\mathrm{i}]$ and $d = \sum_{\mathrm{i}=1}^{n}(X[\mathrm{i}] \oplus Y[\mathrm{i}]) \wedge (M_X[\mathrm{i}] \wedge M_Y[\mathrm{i}])$. Then the above predicate can be re-written as: $d \leq \epsilon.m$. Since we operate with integers in secure computation and since ϵ is a fraction, both sides of the above inequality are multiplied by 2^l with suitable l. This gives us the final representation of the predicate (eq. 2.2), where $d = 2^l.d$ and $m = 2^l.\epsilon.m$, which we use as the authentication function.

$$d \leq m \tag{2.2}$$

It is assumed that the enrollment and authentication iris images are properly aligned, given the on-screen feedback available to users in the devices in order to guide the angle and the distance when capturing the iris image [53]. If there are alignment issues, the extracted feature vectors need to be rotated (with a predefined angle) and the matching steps need to be repeated a predefined number of times [28].

2.2.2 Building Blocks of PEBASI

Structure of the Secure Iris Matching Protocol

In the secure version of iris matching executed during the authentication protocol, the SP inputs the encrypted iris code included in the registered user's enrollment token and the user inputs iris biometric features captured during authentication. Neither party should infer anything about the other party's inputs during protocol execution and the SP should learn nothing but the authentication decision at the end of the protocol.

The high level structure of the protocol is shown in Figure 2.2, in which secure computation of iris matching is executed in two phases. The goal of phase 1 is to securely compute additive shares of the two components of NHD: m and d, by the SP (m^s, d^s) and the user (m^u, d^u). Phase 1 is designed using arithmetic circuits because it involves secure multiplications which is known to be computed faster using arithmetic circuits [31]. The goal of phase 2 is to perform secure comparison given in the predicate in eq. (2.2). The SP

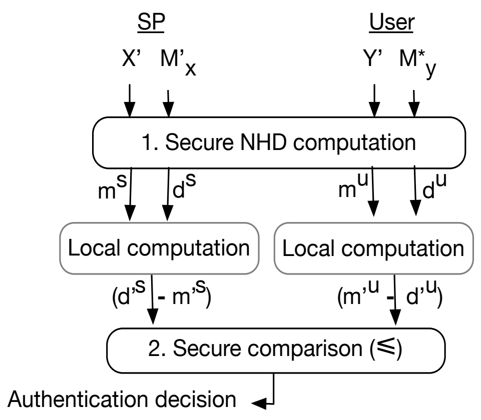


Figure 2.2. Structure of the secure iris matching protocol

and the user input into phase 2, $(d^s - m^s)$ and $(m^u - d^u)$ respectively, which preserve the output of the predicate in eq. (2.2). Phase 2 is designed using Boolean circuits, because secure comparison is faster in Yao's GC based protocols as we show in the benchmarks. We designed the structure of the protocol using a hybrid approach involving arithmetic circuits and Boolean circuits, in order to improve the performance of the authentication protocol, in contrast to designing the entire protocol using either Boolean circuits [30] or arithmetic circuits [35] alone.

Encrypting the Enrolled Biometrics

During the computation of the authentication function, X and M_X (i.e. enrolled iris biometrics) have to be with the SP while Y and M_Y (i.e. iris biometrics obtained at the authentication) are with the user. We use pseudo one-time pad (POTP) [24] to encrypt X and M_X in order to protect confidentiality of the enrolled biometrics from the SP(s). Let $X = X \oplus R_1$ and $M_X = M_X \oplus R_2$, where R_1 and R_2 are two pseudorandom binary vectors and $\oplus$ represents XOR. In order to preserve the output of the computation: $\sum_{\mathrm{i}=1}^{n}(X[\mathrm{i}] \oplus Y[\mathrm{i}])$ (see eq. (2.1)), when X is replaced with X, Y should also be replaced with Y s.t. $Y = Y \oplus R_1$. We also need to make sure that the values of d and m of the authentication function are preserved when M_X is replaced by M_X. That is challenging because unlike with XOR, POTP does not preserve the output of AND. Therefore, we designed the following techniques to preserve the output of d and m when M_X is encrypted with POTP.

Enabling NHD Computation with M_x

By analyzing a truth table containing all possible combinations of values that can be taken by M_X, M_Y and R_2 at any given index i, we observed that if M_Y is updated according to a certain pattern based on the values of M_X and R_2, in order to obtain M_Y^*, then the output of the computation $M_X \wedge M_Y$ is preserved by $M_X \oplus M_Y^*$. Out of all possible combinations of $M_X[\mathrm{i}]$, $M_Y[\mathrm{i}]$ and $R_2[\mathrm{i}]$, the output of $M_X[\mathrm{i}] \oplus M_Y[\mathrm{i}]$ differs from that of $M_X[\mathrm{i}] \wedge M_Y[\mathrm{i}]$, only in the rows: 2, 3, 5 and 6, which are marked with * in Table 2.1. For example, in row 2, if $M_Y[\mathrm{i}] = 1$ when $M_X[\mathrm{i}] = 0$ and $R_2[\mathrm{i}] = 0$, then flipping the value of $M_Y[\mathrm{i}]$ to obtain $M_Y^*[\mathrm{i}]$

Table 2.1. Table illustrating where $M_X \oplus M_Y$ differs from $M_X \wedge M_Y$, in order to identify the pattern for updating M_Y in to M_Y^*, such that $M_X \oplus M_Y^* = M_X \wedge M_Y$.

	M_X[i]	R_2[i]	M_Y[i]	M_X[i] = M_X[i] $\oplus$ R_2[i]	M_X[i] $\wedge$ M_Y[i]	M_X[i] $\oplus$ M_Y[i]
1	0	0	0	0	0	0
2	**0**	**0**	**1**	**0**	**0**	**1***
3	**0**	**1**	**0**	**1**	**0**	**1***
4	0	1	1	1	0	0
5	**1**	**0**	**0**	**1**	**0**	**1***
6	**1**	**0**	**1**	**1**	**1**	**0***
7	1	1	0	0	0	0
8	1	1	1	0	1	1

= 0 will make the outputs of M_X[i] $\oplus$ M_Y^*[i] and M_X[i] $\wedge$ M_Y[i] equal. Similar rules can be derived from rows 3, 5 and 6.

Therefore, an instruction vector W is generated during enrollment, which instructs how to update M_Y to obtain M_Y^* during authentication so that the output of the original authentication function is preserved by the updated authentication function. W is a two-dimensional array with two rows and the number of columns (k) being equal to the number of indices at which a potential flip might have to be performed in M_Y (note that M_Y is obtained during authentication). The first row of W indicates such indices and the second row indicates the values of a potential M_Y at such indices which should be flipped in order to obtain M_Y^*. The values in the second row of W can be either 11 or 00 or 10 which indicates that a flip should happen if 1 or 0 or either is present in M_Y respectively (see eq. (2.3)).

Function I takes as inputs M_X and R_2 and outputs W as shown in eq. (2.3). At an index i, if M_X[i] and R_2[i] values lead to a flip in potential M_Y[i], according to the patterns observed from Table 2.1, I outputs a tuple for W: (W[0][j], W[1][j]) for j = $\{0, 1, ..., k-1\}$.

$$I(M_X, R_2) = \begin{cases} \text{For i} = \{0, 1, ..., n\}; \\ (\text{i}, 11), \text{if } M_X[\text{i}] = 0 \text{ and } R_2[\text{i}] = 0 \\ (\text{i}, 00), \text{if } M_X[\text{i}] = 0 \text{ and } R_2[\text{i}] = 1 \\ (\text{i}, 10), \text{if } M_X[\text{i}] = 1 \text{ and } R_2[\text{i}] = 0 \\ \text{no entry, otherwise.} \end{cases} \tag{2.3}$$

During authentication, the user updates M_Y based on W, according to function U and obtains M_Y^*.

$$U(W, M_Y) = \begin{cases} \text{For i = \{0, 1, ..., n\};} \\ M_Y[\text{i}] \oplus 1, \text{if i = W[0][j] AND} \\ \{(W[1][\text{j}] = 10 \text{ OR} \\ (W[1][\text{j}] = 11 \text{ AND } M_Y[\text{i}] = 1) \text{ OR} \\ (W[1][\text{j}] = 00 \text{ AND } M_Y[\text{i}] = 0))\}. \\ M_Y[\text{i}], \text{ otherwise.} \end{cases} \tag{2.4}$$

Correctness of functions I and U is proved by demonstrating that $M_X \wedge M_Y = M_X \oplus M_Y^*$, where $M_Y^* = U(I(M_X, R_2), M_Y)$, using a truth table similar to Table 2.1 (see Section 2.2.3).

2.2.3 Proof of $M_X[\text{i}] \wedge M_Y[\text{i}] = M_X[\text{i}] \oplus M_Y^*[\text{i}]$

Updated Computation of NHD

In what follows we present the updated equations to compute the two components of NHD, m and d. Differences in these equations from those presented in Section 2.2.1 are due to two factors: i) updated computation of m using encrypted M_X (i.e. M_X) and the updated

Table 2.2. Table proving the correctness of applying functions I and U on M_Y[i], based on M_X[i] and $R2$[i], to obtain M_Y^*[i] such that M_X[i] $\wedge M_Y$[i] $=$ M_X[i] $\oplus M_Y^*$[i].

	M_X[i]	$R2$[i]	M_Y[i]	$W = I(M_X, R2)$	M_Y^*[i] $= U(W, M_Y)$	M_X[i] $=$ M_X[i] $\oplus$ $R2$[i]	M_X[i] $\wedge$ M_Y[i]	M_X[i] $\oplus$ M_Y^*[i]
1	0	0	0	(i, 11)	0	0	0	0
2	**0**	**0**	**1**	(i, 11)	**0**	**0**	**0**	**0***
3	**0**	**1**	**0**	(i, 00)	**1**	**1**	**0**	**0***
4	0	1	1	(i, 00)	1	1	0	0
5	**1**	**0**	**0**	(i, 10)	**1**	**1**	**0**	**0***
6	**1**	**0**	**1**	(i, 10)	**0**	**1**	**1**	**1***
7	1	1	0	-	0	0	0	0
8	1	1	1	-	1	0	1	1

M_Y (i.e. M_Y^*); ii) replacement of Boolean operators: AND and XOR with their arithmetic counterparts. All the arithmetic operations are performed in modulo 2^q.

$$\begin{aligned}\text{Let } T &= M_X \oplus M_Y^* \\ &= M_X(1 - M_Y^*) + (1 - M_X)M_Y^* \\ &= M_X + M_Y^* - 2M_X M_Y^* \\ \text{and then; } m &= \sum_{\text{i}=1}^{n} T[\text{i}]\end{aligned} \tag{2.5}$$

$$\begin{aligned}\text{Let } H &= X \oplus Y \\ &= X + Y - 2XY \\ \text{and then; } d &= \sum_{\text{i}=1}^{n} H[\text{i}]T[\text{i}]\end{aligned} \tag{2.6}$$

Secure Authentication Function

The user and the SP need to securely compute additive shares of m and d, given in equations (2.5) and (2.6) respectively. The two parties have to collaborate only to compute the multiplication operations as the remaining operations can be performed locally. In equations (2.5) and (2.6), there are three pairs of vectors that need to be element-wise multiplied via interactive secure multiplication protocols. In SMPC literature, Beaver's MTs based protocol (see Section 2.5.2) is the most commonly used technique for secure multiplications. Such protocol requires the inputs privately known to each party to be additively shared between the two parties. In contrast, OLFE protocol (see Section 2.5.1), which also enables secure multiplication, does not have such requirement.

We observed that in the computation of T and H, the two parties want to securely multiply secrets privately known to them (i.e. M_X, M_Y^* and X, Y), and in the computation of d, the parties want to securely multiply secrets shared between them, because having securely computed T and H, they already have additive shares of T and H. Accordingly, we design the secure NHD computation by using OLFE in the computation of T and H, and Beaver's MTs in the computation of d. This observation about the characteristics of secret pairs used in secure multiplications and using a hybrid approach for secure computation of NHD had a profound impact on improving the efficiency of phase 1 (see benchmarks in Section 2.3.2).

This hybrid approach for secure computation is an independent contribution which can be used in any secure two-party computation protocol, as appropriate. We have identified four different types of secret pairs and observed that except for a pair of secrets already shared between the two parties, secure multiplication of all the other types of secret pairs are computed faster using OLFE than using Beaver's MTs. See Section 2.3.2 for complexity analysis and benchmarks.

Let P^u and P^s be the vectors containing the additive shares of $P = M_X M_Y^*$ computed by the user and the SP respectively. According to the OLFE protocol given in Section 2.5.1, P^u - $P^s = M_X M_Y^*$. Substituting this in eq. (2.5), we obtain eq. (2.7) for securely computing

T and m. The additive shares of T and m computed by the user and the SP are: T^u, m^u and T^s, m^s respectively.

$$\begin{aligned} T &= (M_X + 2P^s) + (M_Y^* - 2P^u) \\ &= T^s + T^u \\ \text{and } m &= \sum_{\text{i}=1}^{n} T^s[\text{i}] + \sum_{\text{i}=1}^{n} T^u[\text{i}] \\ &= m^s + m^u \end{aligned} \tag{2.7}$$

Similarly, let Q^u and Q^s be the vectors containing the additive shares of $Q = XY$ computed by the user and the SP respectively. Then we obtain eq. (2.8) for securely computing H by the two parties.

$$\begin{aligned} H &= (X + 2Q^s) + (Y - 2Q^u) \\ &= H^s + H^u \end{aligned} \tag{2.8}$$

Substituting for H and T in the second equation in eq. (2.6), we obtain eq. (2.9) for computing d, which is computed using the Beaver's MTs.

$$\begin{aligned} d &= \sum_{\text{i}=1}^{n} (H^s[\text{i}] + H^u[\text{i}])(T^s[\text{i}] + T^u[\text{i}]) \\ &= d^s + d^u \end{aligned} \tag{2.9}$$

Finally, the two parties compute the additive shares of d and m, by using the computed additive shares of m, d, scaling factor l and threshold ϵ. Both ϵ and l are decided by the BCA and publicly known. With that, the authentication function is re-written as follows.

$$\begin{aligned} d^s + d^u &\leq m^s + m^u \\ (d^s - m^s) &\leq (m^u - d^u) \end{aligned} \tag{2.10}$$

Given the two targets of the above comparison predicate as inputs into phase 2, the eq. (2.10) is evaluated using a GC composed of greater than equal comparator.

2.2.4 Enforcing a Malicious User to Follow the Prescribed Protocol

As defined in the threat model (Section 2.1.1), the user can be malicious, who may deviate from the prescribed protocol to make the output of the protocol incorrect. The two main ways that a malicious user could achieve this are: i) submitting random values as inputs to phase 2, which are different from the actual output of phase 1. ii) cheating when playing the role of the GC creator by generating GCs that would always output 1 regardless of the inputs. In what follows, we present the two versions of the protocol which enforce the user to follow the prescribed protocol via two different approaches.

Protocol Version 1 (V1)

In V1, we assume the existence of hardware based security mechanisms such as attestation through which the SP can verify integrity of the authentication software running in the user's device [54]. Therefore, in V1, the authentication function is computed as it is presented in Section 2.2.3. Such approach can also be used for more efficient secure iris matching in settings which require only semi-honest security, such as biometric identification, with the only difference of not having to encrypt the inputs of the two parties.

Protocol Version 2 (V2)

In V2, we enforce the user to follow the protocol solely based on cryptographic primitives. For this, first we need to use an homomorphic MAC scheme to authenticate the user's inputs to phase 1 at the beginning of the protocol and to verify the integrity of all values output by the user, including the final output of phase 1, which is given as input to phase 2, in order to avoid the aforementioned attack 1. Next we need to get the BCA to generate signed GCs and supply them during the offline phase, in order to avoid aforementioned attack 2.

Definition of an homomorphic MAC scheme to be used for actively secure SMPC protocols with dishonest majority where *computation is performed over a ring* has been an open problem until SPDZ2k [55] was proposed. However, in SPDZ2k protocol, secure multiplication is performed using Beaver's MTs only. An homomorphic MAC scheme that enables actively secure multiplications with OLFE *over a ring* has not yet been proposed, to the best of our knowledge. Therefore, in V2, we use Beaver's MTs based protocol for all secure multiplications and adapt the techniques proposed in SPDZ2k (see Section 2.5.5) for verifying the integrity of the computations carried out by the user.

Let $[x]$ represent the authenticated version of x, where $[x] = (x, m_x, \alpha)$, α is the global MAC key and $m_x = \alpha x \bmod 2^p$ for any appropriately selected p. During the offline phase, the BCA generates α and provides its additive shares to the user and the SP as α^u and α^s respectively. This enables the two parties to compute authenticated values in the local computations performed during the online protocol. The BCA itself uses α to pre-compute the authenticated values provided to the two parties during the offline phase. Section 2.5.6 describes how the BCA pre-computes such artifacts.

The user and SP compute authenticated shares of their secret inputs, following the protocol given in Section 2.5.7. Let $[X^s]$, $[X^u]$, $[M_X^s]$, $[M_X^u]$ and $[Y^s]$, $[Y^u]$, $[M_Y^{*s}]$, $[M_Y^{*u}]$ be the authenticated shares of the secret inputs (i.e. $[X]$, $[M_X]$ and $[Y]$,$[M_Y^*]$) of the SP and the user respectively. Let $[P^u]$ and $[P^s]$ be the authenticated shares of $[P] = [M_X][M_Y^*]$ computed by the user and the SP respectively. According to the Beaver's MTs based protocol, $[P^u] + [P^s] = [M_X][M_Y^*]$. Then according to equation 2.5, we obtain:

$$\begin{aligned} [T] &= ([M_X^u] + [M_Y^{*u}] - 2[P^u])) + ([M_X^s] + [M_Y^{*s}] - 2[P^s]) \\ &= [T^u] + [T^s] \end{aligned} \tag{2.11}$$

Similarly, let $[Q^u]$ and $[Q^s]$ be the authenticated additive shares of $[Q] = [X][Y]$. Then according to equation 2.8, we obtain:

$$\begin{aligned} [T] &= ([X^u] + [Y^u] - 2[Q^u])) + ([X^s] + [Y^s] - 2[Q^s]) \\ &= [H^u] + [H^s] \end{aligned} \tag{2.12}$$

The rest of the protocol continues as shown in Section 2.2.3 for computing the additive shares of d and m, and the two parties' inputs to phase 2, except that the authenticated values are involved in the computation throughout. The 'AffineComb' procedure of SPDZ2k [55] shows how each party performs arithmetic operations locally, involving authenticated values.

In order to avoid the aforementioned attack 1, the SP needs to verify the integrity of the computation of phase 1 by the user. First, the SP verifies the MACs of all the intermediate values sent by the user via the 'BatchCheck' procedure of SPDZ2k. Next, the SP needs to verify integrity of the final output of phase 1 (i.e. input to phase 2) by the user. However, we do not want the user and the SP to open their inputs to phase 2 and check the MAC values, unlike the 'SingleCheck' procedure presented in SPDZ2k for this purpose. This is because our privacy goal is that the SP learns nothing but the authentication decision.

Instead, the GC used in phase 2 is designed to perform MAC verification of the user's input as well as secure comparison, as shown in Figure 2.3. The user inputs into phase 2; $m^u - d^u$, its MAC value, and α^u. The SP inputs $d^s - m^s$, MAC value of its negation, and α^s. The SP's second input is designed to be the MAC of the negation of the first input, in order to minimize the total number of inputs and computations handled by the GC. Subcircuit 1 performs greater than equal comparison, same as the GC used in protocol V1. Subcircuit 2 computes $m - d$. Subcircuit 3 computes MAC($m - d$), from the MAC values input by the two parties. Sub circuit 4 computes the global MAC key α from its shares input to the GC. Subcircuit 5 computes MAC($m - d$), by multiplying α and $m - d$, computed inside the GC. Subcircuit 6 verifies if the two MAC values output by the subcircuits 3 and 5 are equal, in which case the integrity verification of the user's computations is successful. Authentication is successful if and only if both sub circuits 1 and 6 output 1, which is computed by the output AND gate.

During the execution of the Yao's GC protocol, a malicious GC generator / OT (Oblivious Transfer) sender can perform two main attacks: i) Constructing incorrect garbled truth tables (GTTs) which always output the desired result. ii) Carrying out a selective input attack during OT, where the sender sends an incorrect label for a particular bit of the receiver's input and if the protocol fails early, the sender guesses the receiver's input for that particular bit. In general-purpose Yao's GC protocols, attack (i) is addressed using techniques such as cut-n-choose [56], [57], authenticated garbling [58], etc. We address it by getting the BCA to provide signed GTTs during the offline phase, which are trusted by the SP. Accordingly, we decouple the GC generator's and OT-sender's roles of the Yao's GC protocol and assign them to the BCA and to the user respectively. Attack (ii) is not relevant in our case because the SP's input to phase 2 is a random value during each execution of the protocol. Therefore, trying to guess one bit of the SP's input during each execution does not provide any advantage for a malicious user to be successful in a fake authentication attempt.

2.2.5 PEBASI

There are four sub protocols in PEBASI: i) enrollment; ii) sign-up; iii) authentication artifact re-fill; and iv) authentication. The enrollment phase is the same for both versions of PEBASI, whereas the other three phases differ in certain ways across the two versions. We describe such differences under each phase.

Enrollment Protocol

The enrollment protocol, (Protocol 2.1) is executed between the user and the BCA. At the end of the protocol, the user gets an enrollment token (ET) signed by the BCA, along with other artifacts to be used during authentication. The BCA deletes all the sensitive information at the end of the enrollment protocol. Therefore, an attack on the BCA's database does not compromise the user's biometric privacy.

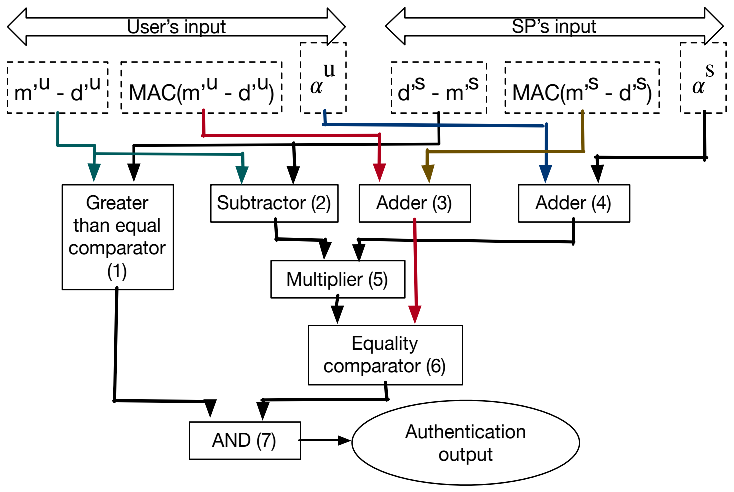


Figure 2.3. Garbled circuit used in phase 2 of protocol version 2

Protocol 2.1: Enrollment Protocol

Inputs: User inputs a capture of her iris biometric F_E, and three secret pseudo random binary vectors R_1, R_2 and R_3 (see Remark 1).

Output: The BCA outputs the signed ET and the other authentication artifacts.

Protocol Execution:

1. *User*$\rightarrow$*BCA*: F_E, R_1, R_2 and R_3.
2. *BCA*: i) extracts iris features X and mask M_X from F_E. ii) computes $X = X \oplus R_1$ and $M_X = M_X \oplus R_2$. iii) computes $W = I(M_X, R_2)$ and encrypts each element in the second row of W with the next two bits in R_3, i.e. for j $= 0, 1, .., k-1$, sets $W[1][\mathrm{j}] = W[1][\mathrm{j}] \oplus R_3[\mathrm{j} \times 2]R_3[\mathrm{j} \cdot 2 + 1]$.

BCA$\rightarrow$*User*: W and the signed enrollment token (ET) which includes X, M_X and any other user identifier(s).

BCA: deliberately discards all the data.

Remarks: 1) The three pseudo random binary vectors can be derived from a password or from the randomness introduced in the user's device during manufacturing via a physically unclonable function (PUF) [59]. In the scope of this thesis, we derive them from a strong password of the user as: $R_1|R_3 \leftarrow$ PRF(password, nonce1) and $R_2 \leftarrow$ PRF(password, nonce2). Each of R_1 and R_2 is of length n and R_3 is of length $2n$. R_1, R_3 and R_2 are generated separately based on the same password and using two different nonces because R_1 and R_3 need to be re-generated during authentication, whereas R_2 is required only during enrollment. Therefore, only nonce1 should be reproducible. Given that W is stored in the user's device, PEBASI is a three factor authentication scheme involving authentication factors of types: *'what you are'* (i.e. biometrics), *'what you have'* (i.e. the user's device) and *'what you know'* (i.e. password). 2) The BCA may require the user to visit the BCA in person to perform the enrollment or allow the user to perform the enrollment remotely, based on the required level of assurance of the biometric identity and the legal requirements of the context, which are out of the scope of our work. 3). Revocation is discussed in Section 2.2.6.

2.2.6 Revocation

Revocation of the enrollment tokens can be caused by three main factors. 1) *Expiration:*. During the enrollment, the BCA stamps the expiration time on the enrollment token. Once the enrollment token is expired, the SP should refuse to authenticate the user until the user obtains a new one from the BCA. 2) *Revocation initiated by the user* due to forgetting the password or losing access to the authentication artifacts (e.g. loss of the user's device). 3) *Revocation initiated by the BCA* due to termination of the user's affiliation with the BCA, e.g. the BCA is the user's employer organization and the user leaves the organization. In this case, the BCA can stop executing the re-fill protocol with the user. If the user re-uses the old authentication artifacts to authenticate with SPs even after the revocation, the user risks her biometric privacy. The SP can detect such authentication attempts via the serial number that the BCA includes in the authentication artifacts. The authentication artifacts have a shorter life time than the enrollment tokens.

Signup

The signup phase serves two main purposes: i) The user hands over to the SP, the ET signed by the BCA. The SP verifies the signature on the ET and associates the ET with the user's account created at the SP. ii) The user and the SP compute base-OTs to be used during authentication. In phase 2 of the authentication protocol, the garbled labels representing the inputs of the SP need to be transferred using oblivious transfer (OT) during authentication. The base-OTs computed by the user and the SP during this signup phase are used during as many instances of the authentication protocol as needed without apriori bound on the number of authentications to be performed. The number of input bits of the SP into phase 2 differ among the two versions of the protocol based on the GC used in each version.

Authentication Artifact Refill

In the authentication artifact refill phase, the BCA computes, signs and distributes the pre-computable artifacts required by the two parties for the online protocol. Although these authentication artifacts differ across V1 and V2 of the protocol, the following three steps are executed between the three parties in this phase of both versions: i) transferring the aforementioned artifacts generated for both parties and signed by the BCA, from the BCA to the user. ii) transferring the artifacts intended for the SP, from the user to the SP. iii) verifying the BCA's signature on the received artifacts by the SP. In those three steps, the

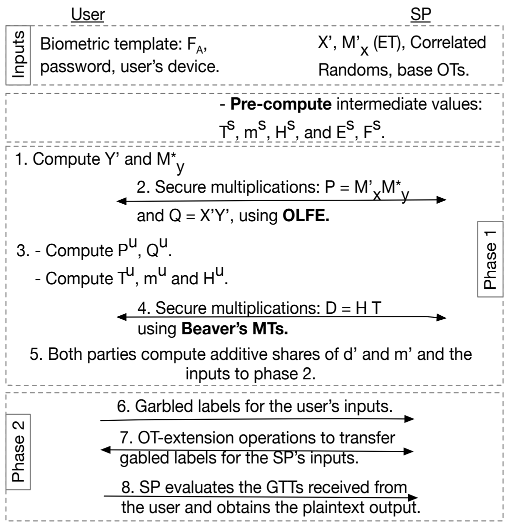

User
SP
Inputs
Biometric template: F_A, password, user's device.
X', M'_x (ET), Correlated Randoms, base OTs.
- Pre-compute intermediate values: T^s, m^s, H^s, and E^s, F^s.
1. Compute Y' and M*_y
2. Secure multiplications: P = M'_x M*_y and Q = X'Y', using OLFE.
3. - Compute P^u, Q^u.
- Compute T^u, m^u and H^u.
Phase 1
4. Secure multiplications: D = H T using Beaver's MTs.
5. Both parties compute additive shares of d' and m' and the inputs to phase 2.
Phase 2
6. Garbled labels for the user's inputs.
7. OT-extension operations to transfer gabled labels for the SP's inputs.
8. SP evaluates the GTTs received from the user and obtains the plaintext output.

Figure 2.4. Concise steps of the authentication protocol V1.

artifacts are transferred in batch mode such that each re-fill step supplies several sets of the artifacts, sufficient to execute several instances of the authentication protocol. We assume a secure communication channel to protect the artifacts at transit and encryption to protect the artifacts in rest.

V1: In phase 1, the two parties securely multiply three vector pairs (see Section 2.2.3). Therefore, for each instance of the authentication protocol, each party receives: i) two pairs of vectors, each pair containing the correlated randoms to perform the secure multiplications using OLFE (see Section 2.5.1); ii) three vectors containing the correlated randoms to perform the secure multiplications using Beaver's MTs (see Section 2.5.2). **V2:** Refill phase of V2 can be best described after describing the main ideas of the SPDZ2k protocol, that are relevant to PEBASI V2. Therefore, refill phase of V2 is described in Section 2.5.5.

Authentication

The authentication protocol executed between the user and the SP, consists of two phases as described in Section 2.2.4. **V1:** Figure 2.4 illustrates the concise steps of the protocol V1. In the secure computation of T and H, we deliberately assign Alice's role in the OLFE protocol to the SP and Bob's role to the user, in order to minimize the number of interactions

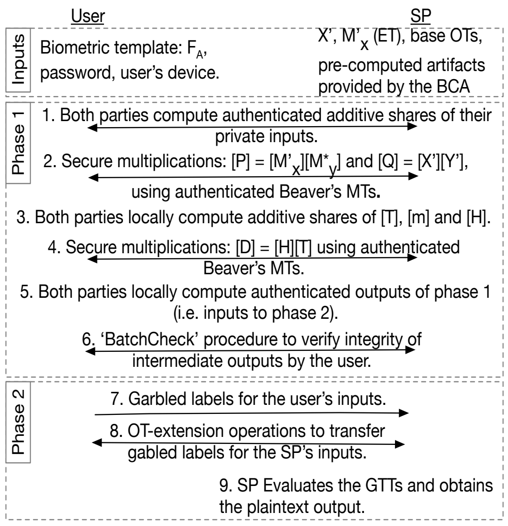

User
SP
Inputs
Biometric template: F_A, password, user's device.
X', M'_X (ET), base OTs, pre-computed artifacts provided by the BCA
Phase 1
1. Both parties compute authenticated additive shares of their private inputs.
2. Secure multiplications: [P] = [M'_X][M*_Y] and [Q] = [X'][Y'], using authenticated Beaver's MTs.
3. Both parties locally compute additive shares of [T], [m] and [H].
4. Secure multiplications: [D] = [H][T] using authenticated Beaver's MTs.
5. Both parties locally compute authenticated outputs of phase 1 (i.e. inputs to phase 2).
6. 'BatchCheck' procedure to verify integrity of intermediate outputs by the user.
Phase 2
7. Garbled labels for the user's inputs.
8. OT-extension operations to transfer gabled labels for the SP's inputs.
9. SP Evaluates the GTTs and obtains the plaintext output.

Figure 2.5. Concise steps of the authentication protocol V2.

and to maximize the number of operations that can be pre-computed. Accordingly, the SP pre-computes its shares of T, m and H, and those of E and F vectors for the secure multiplications required to compute d using Beaver's MTs. At the end of phase 1, both parties have shares of d and m. In addition to the outputs of phase 1, both parties use, as inputs to phase 2, the base-OTs computed during the signup phase. The user generates the garbled label pairs representing the inputs to the GC by the two parties. In phase 2, first, the user transfers the generated garbled labels corresponding to her inputs into phase 2. Then the two parties execute OT-extension for the SP to learn the garbled labels corresponding to the SP's inputs. At the end of phase 2, the user and the SP perform parallel execution of transferring the GTTs from the user to the SP and evaluation of the GTTs by the SP. Finally, the SP learns the output of the GC evaluation (i.e. authentication decision) in plain text.

V2: Figure 2.5 illustrates the concise steps of the protocol V2. The main differences compared to the protocol V1 are: i) the two parties have more pre-computed authentication artifacts as inputs; ii) all three sets of secure multiplications are performed using Beaver's MTs based protocol; iii) all computations in phase 1 take authenticated inputs and output authenticated results; iv) there is an additional step at the end of phase 1 to perform ‘BatchCheck’ procedure; v) phase 2 performs MAC verification in addition to secure comparison, using the signed GC artifacts provided by the BCA.

2.3 Empirical Evaluation

We implemented prototypes of both versions of PEBASI in Java and carried out experiments in order to: i) benchmark the two phases of the authentication protocol in V1 against the state-of-the-art approaches ii) compare the performance of the online authentication protocol of the two versions of PEBASI. Out of the four protocols of PEBASI, enrollment and signup protocols are one-time protocols and the re-fill steps are run as scheduled background tasks in the user’s device. Therefore, in the scope of this thesis, we focus on evaluating the performance of the authentication protocol which directly affects the online user experience.

2.3.1 Implementation details

We first implemented a framework which enables two party secure computation of arithmetic circuits using either OLFE primitive or Beaver’s MTs primitive or a combination of both, as appropriate, based on the type of the secrets present. Next we implemented the techniques adapted from the SPDZ2k in order to make the two party computation of arithmetic circuits based on Beaver’s MTs secure against a malicious party. To enable secure computation of Boolean circuits via Yao’s GC protocol, we integrated FastGC [60], [61] into our framework. We selected FastGC after evaluating multiple available implementations. FastGC comes as a complete tool box for implementing Yao’s GC protocols, unlike other implementations, which although contain more recent optimizations [62]–[64], provide only one or few of the required building blocks. FastGC also provides sufficient control over the circuit design, unlike the circuit compilers [65]. Both versions of PEBASI are implemented on top of the aforementioned framework. We used SecureRandom, which is a cryptographically strong random number generator provided by Java [62], in order to generate the correlated randoms used in those primitives. In order to generate the pseudorandom binary vectors used in POTP encryption, we use password based key derivation (PBKDF) algorithm in

PKCS#5 2.0 with HMACSHA256 as the pseudorandom function, nonce values of 64 bytes and 10000 iterations.

For phase 2 of the authentication protocol we introduced certain additions and modifications to the FastGC framework which are of general interest, including, but not limited to: i) Four new circuit building blocks: greater than equal comparator, subtractor, multiplier and equality comparator (see Figure 2.3). ii) Changing the computation and evaluation of GTTs corresponding to *output gates* s.t. they directly output 0 or 1. This enables the evaluator itself to learn the plaintext output at the end of the GC evaluation [66]. The original FastGC framework requires the evaluator to send the garbled output to the GC creator in order to learn the plaintext output, which does not fit in our protocol setting. iii) Decoupling the two roles of the Yao's GC protocol from the client and the server, because FastGC is designed assuming that the server is the circuit creator and the client is the circuit evaluator always, which is not the case in our protocol. iv) Decoupling the roles of GC creator and OT-sender (which were played by one single party in the original framework), s.t. they can now be played by two different parties (i.e. BCA and the user in our protocol V2). v) Enabling each party to give more than one input to the composite GC, because FastGC is designed assuming only one input from each party.

2.3.2 Experimental Setup and Benchmarks

We executed the authentication protocol in two Amazon EC2 instances of type t3a.large, with the following configurations: Ubuntu 18.04 LTS OS, 8 GB memory, two 2.5 GHz AMD EPYC 7000 CPU and up to 5 Gbps network performance. In the benchmarks, we compare the performance of the implementations of the two main phases of our authentication protocol V1 with the same two phases that we also implemented using the techniques which were known/claimed to be the most efficient as per previous works. We plot the performance numbers of the two phases of protocol V2 in the same graphs in order to compare the performance of the two phases across the two versions of our protocol. At the end of this section, we present benchmarks for the hybrid secure multiplication approach w.r.t the identified four types of secret pairs (Section 2.2.3).

We report benchmark results by varying the statistical security parameter q from 80 to 128. In phase 1 of V1, q is the bit length of the randoms which hide the actual values of d and m from the additive shares computed by the two parties. Modular arithmetic is performed in Z_{2^q}. We start from $q = 80$ and use it as the default value for q, because: i) it is good enough for the security requirement of phase 1 of V1. ii) it is the statistical security parameter used in FastGC [60], which we use to implement phase 2 of the two versions of the protocol. In V2, we set the parameter s introduced by the SPDZ2k techniques to be twice as large as q (i.e. $s = 2 * q$), which is good enough for the required security of s - $log(s)$. Iris codes were generated using the same parameters used in the iris identification scheme proposed by Blanton et al. [28], which are also the parameters used in commercial iris recognition software. The size of the iris code (n) is 2048, and 75% are reliable bits.

Benchmarks for Phase 1

We compare phase 1 of the authentication protocol V1 built using our hybrid approach for secure computation (referred to as approach **1-A**) with the phase 1 built using the approach proposed in ABY [31] (referred to as approach **1-B**). Approach 1-B is known to be the most efficient technique proposed for secure computation of distance in biometric matching in the semi-honest model. As shown in Figure 2.6, approach 1-A outperforms approach 1-B in terms of end-to-end execution time, with an average improvement of 35% at the default configuration of q. The reason is that the secure NHD computation involves a number 2*n of secure multiplications of secrets privately known to the parties (i.e. referred to as *Type 1* in Section 2.3.2) in the computation of T and H, and a number n of secure multiplications of secrets shared between the two parties (i.e. *Type 4*) in the computation of d, where n is the size of the iris code. And according to the benchmarks in Section 2.3.2 and Table 2.4, Beaver's MTs is more expensive than OLFE in secure multiplication of *Type 1* secrets, although OLFE is more expensive than Beaver's MTs in secure multiplication of *Type 4* secrets.

Approach 1-D (PEBASI V2) uses authenticated Beaver's MTs for all secure multiplications and the techniques adapted from SPDZ2k for integrity verification of the user's computations, which incur additional computation and communication costs. Therefore, approach 1-D is expected to be more expensive than the aforementioned two approaches. Approach 1-D has an average increase of 56.7% in the end-to-end execution time compared to approach 1-A at the default configurations (i.e. q =80, and s =160). However, according to Figure 2.6, approach 1-D has performance values very close to those of approach 1-B for the frist two configurations of q. The difference in the performance gets larger as q (hence s) increases.

Benchmarks for Phase 2

We benchmark phase 2 of V1 built using: i) a Boolean circuit based Yao's GC protocol (referred to as approach **2-A**) ii) an arithmetic circuit based secure comparison protocol (referred to as approach **2-B**). Approach 2-B was proposed in [67], as a more efficient version of the secure comparison protocol proposed in [68]. Demmler et al. [31] claimed that approach 2-A is the fastest for secure comparison and subsequently, Droandi et al. [35] claimed that approach 2-B is the fastest for the same. The benchmark results in Figure 2.7 confirms that approach 2-A outperforms approach 2-B. The reason is that in approach 2-A, there is only one round of interaction between the two parties for OT-extension and transfer of GTTs, whereas in approach 2-B, there are q rounds of interactions. Therefore, we use approach 2-A in phase 2 of our authentication protocol. The curve named approach 2-C represents phase

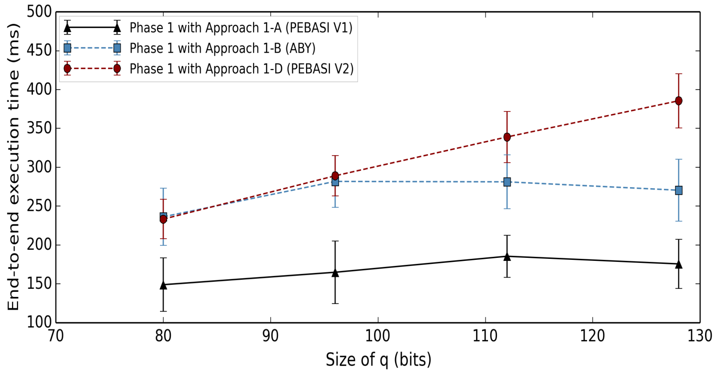


Figure 2.6. Benchmarks for end-to-end execution times of Phase 1. The error bars represent the standard deviation.

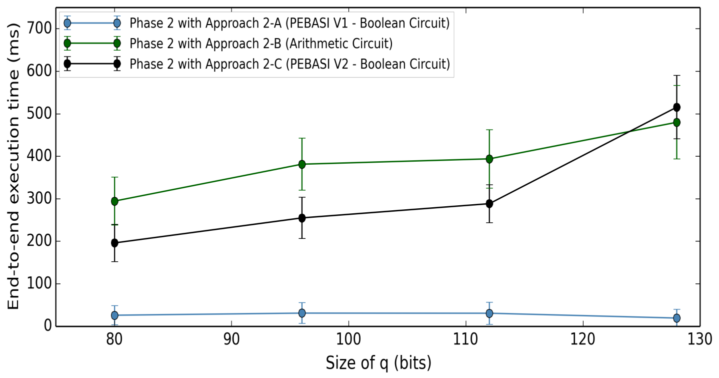


Figure 2.7. Benchmark for end-to-end execution times of Phase 2. The error bars represent the standard deviation.

2 of V2 implemented using the Yao's GC protocol with the GC given in Figure 2.3. Since phase 2 of V2 involves a more complex GC and more number of inputs compared to phase 2 of V1, approach 2-C has an average increase of 71.3% compared to approach 2-A, in terms of end-to-end execution time at the aforementioned default configuration.

Comparison of the Two Versions of PEBASI

Table 2.3. Average end-to-end execution times, percentage of CPU usage averaged over the two cores, average memory usage and communication size (with iris code size = 2048 bits) of the two phases of the authentication protocol in the two versions of PEBASI.

q=80 / s=160	V1				V2			
	Exec. time (ms)	CPU usage (%)	Memory usage (MB)	Comm. Size (MB)	Exec. time (ms)	CPU usage (%)	Memory usage (MB)	Comm. Size (MB)
Phase 1	148.85 (± 34.36)	user: 17.67 SP: 11.79	user: 400.7 SP: 376.24	0.1953	233.27 (± 25.36)	user: 30.78 SP: 31.53	user: 810.4 SP: 843.43	0.4687
Phase 2	26.04 (± 22.75)	user: 24.48 SP: 26.06	user: 205.7 SP: 248.52	0.0038	196.17 (± 43.87)	user: 2.73 SP: 10.05	user: 264.1 SP: 1095.2	0.0305

For each of the two phases of the authentication protocol of the two versions of PEBASI with the default configurations of $q = 80$ and $s = 160$, we report in Table 2.3, the average end-to-end execution time (which include both computation and communication overhead) in milliseconds (ms), average percentage of CPU usage over the two cores (which indicates how much CPU bound each process is), average memory usage in MegaBytes (MB) and communication size in MB. The average measurements are computed over 1000 runs. Accordingly, the average total executions times of the authentication protocol are less than one fifth of a second for V1 and less than half of a second for V2. Both user and SP are almost equally CPU-bound in all cases except for phase 2 of V2. This is because in both phases of V1 and phase 1 of V2, both parties perform similar amounts of computations and network/file IO, except in phase 2 of V2, where the SP performs more computations and file IO than the user since SP reads and evaluates the GTTs supplied during the offline phase, using the garbled labels obtained from the user. Memory usage is also symmetric for the two parties in all cases except for phase 2 and V2 due to the same reasons. Communication size is a fraction of a MB for all cases, with the highest and lowest communication sizes reported at phase 1 of V2 and phase 2 of V1 respectively, as expected.

Hybrid approach for two-party secure multiplication

We have identified four different types of secret pairs that two parties may perform secure multiplication with. We encounter all these types during our secure biometric matching protocols. In each equation for multiplication listed under the four types below, the components which require secure multiplication between two parties are highlighted in bold font. *Type 1:* One party knows secret A. The other party knows secret B. Multiplication = $\mathbf{AB}$. *Type 2:* One party knows secret A. Secret B is additively shared between the two parties. Multiplication = $A(B_1 + B_2) = AB_1 + \mathbf{AB_2}$. *Type 3:* Secret ($A$) is additively shared between the two parties. They want to compute the square of A. Multiplication = $(A_1 + A_2)^2 = A_1^2 + \mathbf{A_1A_2} + A_2^2$. *Type 4:* Both secrets (A, B) are additively shared between the two parties. Multiplication = $(A_1 + A_2)(B_1 + B_2) = A_1B_1 + \mathbf{A_1B_2} + \mathbf{A_2B_1} + A_2B_2$.

We compared the complexity of performing secure multiplication on the above four types using OLFE and Beaver's MTs, both theoretically (see Table 2.4) and empirically (see Figure 2.8). Figure 2.8 shows the average end-to-end execution times for 100,000 secure multiplications with the four types of secret pairs (shown in four different colors), performed using both OLFE (shown with a solid line of each color) and Beaver's MTs (shown with a dashed line of each color). The performance numbers were measured on Amazon T3.a instances running Ubuntu 18.04 LTS, by varying the size of the secrets as 8, 16 and 32 bits. Percentage differences in the end-to-end execution times between OLFE and Beaver's MTs on each type of secret pairs are listed on the top left corner of Figure 2.8 in the corresponding colors.

Table 2.4. Complexity analysis of a single secure multiplication of different types of secret pairs using OLFE and Beaver's MTs. Note that $*$ represents number of multiplications and $\pm$ represents number of additions/subtractions. q is the bit length of the elements of the arithmetic group Z_{2^q}. This analysis is obtained from the algorithms of the two primitives presented in Sections 2.5.1 and 2.5.2.

Type of the secrets	# Computations		Communication size		# Interactions		# Correlated randoms required	
	OLFE	Beaver	OLFE	Beaver	OLFE	Beaver	OLFE	Beaver
1. Two known secrets	2^*, $6^\pm$	5^*, $15^\pm$	$3q$	$6q$	2	4	4	6
2. A shared secret and a known secret	3^*, $7^\pm$	5^*, $14^\pm$	$3q$	$5q$	2	2	4	6
3. Two shared secrets known to be the same (i.e. squaring of the shared secret)	2^*, $8^\pm$	5^*, $13^\pm$	$3q$	$4q$	2	2	4	6
4. Two shared secrets	6^*, $14^\pm$	5^*, $13^\pm$	$6q$	$4q$	2	2	8	6

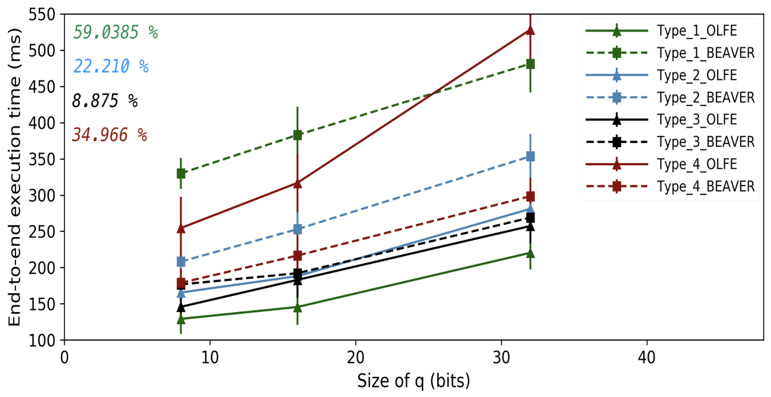


Figure 2.8. End-to-end execution times for secure multiplications (of vectors of size 100,000) averaged over 10,000 executions, of the four types of secret pairs using OLFE and Beaver's MTs, vs size of the secrets in bits.

According to Protocol 2.2 and 2.3 in Sections 2.5.1 and 2.5.2, and the analysis in Table 2.4, OLFE takes less number of communication rounds and computations compared to Beaver's MTs in the secure multiplication of secrets privately known to the two parties. This is mainly because Beaver's MTs protocol always require the two parties to first share the secrets privately known to them, which adds additional rounds of communication. Therefore, OLFE is faster than Beaver's MTs for secure multiplications of *Type 1*, *Type 2* and *Type 3*, because they require only one invocation of OLFE to compute the component(s) shown in bold font in each equation (other components, if any, can be computed locally). On the other hand, OLFE is more expensive than Beaver's MTs for secure multiplication of *Type 4*, because it requires two invocations of OLFE whereas it can be computed with one invocation of Beaver's MTs.

2.4 Security Proofs

In what follows, we prove that both PEBASI V1 and V2 achieve the security goals mentioned in Section 2.1.1.

Lemma 1 establishes that the authentication protocol is correct.

Lemma 1. *The execution of the authentication protocol preserves the output of the non-secure authentication function defined in eq.* (2.1).

Proof: The proof depends on correctness of the building blocks. The properties of the XOR function and the proof of correctness of functions I and U (given in Section 2.2.3) guarantee that m and d values in eq. (2.1) are preserved by those in eq. (2.5) and eq. (2.6) when X and Y are replaced with X and Y and when M_X and M_Y are replaced with M_X and M_Y^*, respectively. Correctness of the OLFE construction [25] and Beaver's MTs based protocol [32], [33] guarantees that m and d values in eq. (2.1) are continued to be preserved by those computed in both V1 and V2. Given the use of a suitable scaling factor l [28], eq. (2.2) preserves the output of eq. (2.1). The proof of correctness of the GC protocol given in [69] guarantees that the authentication decision given by eq. (2.1) is continued to be preserved by the output of the greater than equal comparator in the two GCs used in phase 2 of both versions of the protocol.

Lemma 2 establishes that the BCA does not learn anything about the user's transactions with SP(s) via the protocol interactions.

Lemma 2. *All four sub protocols of PEBASI protects the user's transactional privacy from the BCA.*

Proof: Out of the four sub protocols in PEBASI, the BCA is not involved in either authentication or signup protocols that the user carries out with the SP(s). Out of the two protocols that the BCA is involved in, the enrollment protocol does not involve any information about SP(s). In the re-fill protocol, the authentication artifacts are passed from the BCA to the SP, via the user. Therefore, the BCA neither has any interactions with the SP(s) that the user interacts with, nor it learns any information about such SP(s).

Lemma 3 and 4 establish that a malicious user does not succeed in impersonating a legitimate user, via either of the two attacks mentioned in Section 2.1.1.

Lemma 3. *A computationally bounded malicious adversary has a negligible probability of success in impersonating a user, using stolen biometric template(s).*

Proof: An imposter may attempt to bypass the biometric scanner by replaying any stolen biometric templates in order to impersonate a legitimate user. PEBASI addresses this by incorporating two additional authentication factors into the authentication protocol. The user's password is required to obtain the pseudo random vectors R_1 and R_3 and the access to the user's device is required to obtain W and the matching authentication artifacts that are provided by the BCA and that are compatible with the counterparts passed on to the SP during the refill phase. An imposter has negligible probability in stealing all three factors of authentication of the same legitimate user. Therefore, an imposter with stolen biometric template(s) has negligible probability of success in impersonating via PEBASI.

Lemma 4. *The authentication protocol protects against a malicious user who may deviate from the prescribed protocol to fake a successful authentication result.*

Proof: We follow two different approaches to enforce a malicious user to follow the prescribed protocol (i.e. to enforce the semi-honest behavior on a malicious user) in V1 and V2 of the protocol, as described in Section 2.2.4. The security of the approach used in V1 (i.e. remote attestation of integrity of the authentication software) depends on the security of the secure hardware (i.e. TEEs - Trusted Execution Environment or SEs - Secure Elements) used in the users' devices. On the other hand, the security of the approach used in V2 depends on the security of the underlying cryptographic primitives. We refer the reader to the SPDZ2k [55] paper for the security proof of the homomorphic MAC scheme used for verifying the integrity of the user's outputs in phase 1 (i.e. inputs to phase 2). Our proof also relies on the proof for the standard digital signature algorithm [70], which is used for verifying the authenticity of the GTTs in phase 2.

Lemma 5 establishes that neither the SP nor a non-legitimate user could breach biometric privacy of a legitimate user, via execution of the authentication protocol.

Lemma 5. *The authentication protocol protects a legitimate user's biometric privacy.*

Proof of Lemma 5: Given that a (non-legitimate) user is enforced to follow the prescribed protocol (see Lemma 4), the proof reduces to proving that a legitimate user's biometric privacy is protected against two semi-honest adversaries: the SP and a non-legitimate user, who may execute the authentication protocol. The notion of security against semi-honest adversaries, which is formalized in the real-ideal paradigm, requires that a party's view in the real world protocol execution be simulatable in the ideal world given its inputs and outputs [71], which implies that the party learns nothing about the other party's inputs from the protocol execution.

Definition: *Let parties* P_1 *and* P_2 *engage in a protocol* π *which computes a function* $f = (f_1,\ f_2)$*. Let* in_i *and* f_i *denote the input and output of party* P_i *respectively. Let* $\mathrm{view}_\mathrm{i}^\pi$ *denote the view of party* P_i *during the real execution of* π*, i.e.* $\mathrm{view}_\mathrm{i}^\pi = (\mathrm{in}_\mathrm{i},\ r_\mathrm{i},\ m_1, \ldots, m_t)$*, where* r_i *denotes* P_i*'s internal random tape contents and* m_j *denotes the* j^{th} *message received by* P_i *during the protocol execution.* π *securely computes* f *in the presence of semi-honest adversaries if there exist probabilistic polynomial time algorithms* S_i*, where* $\mathrm{i} \in \{1, 2\}$*, such that* $\{S_\mathrm{i}(\mathrm{in}_\mathrm{i},\ f_\mathrm{i}),\ f(\mathrm{in}_1, \mathrm{in}_2)\} \approx_c \{\mathrm{view}_\mathrm{i}^\pi, output^\pi(\mathrm{in}_1, \mathrm{in}_2)\}$*, where* $\approx_c$ *denotes computational indistinguishability [28], [71].*

To prove that the authentication protocol is secure according to the above definition, we show how to simulate the views of the two parties using their inputs and outputs in both versions of the protocol. Then we prove that such simulations are indistinguishable from the real execution of the protocol, which implies that both versions of the protocol are privacy preserving.

Protocol V1

Simulation of the SP's view: S_s^1 simulates the SP's view of protocol V1 (see Figure 2.4). S_s^1 simulates the messages received from the user in step 2, by generating and sending to the SP two vectors containing randoms in Z_{2^q} that are indistinguishable from the vectors containing the M values (i.e. M_P and M_Q) of the OLFE protocol (see Section 2.5.1) received in the real protocol for the secure multiplications: P and Q. Then S_s^1 simulates the messages received from the user in step 4, by generating and sending to the SP two vectors containing randoms in Z_{2^q} that are indistinguishable from the vectors containing the shares of E and F values of the Beaver's MT protocol (see Section 2.5.2) received in the real protocol for the secure multiplications of D. In order to simulate steps 6 and 7, S_s^1 generates random garbled label pairs for the inputs of the two parties into phase 2. Then S_s^1 generates random bits as the user's input bits and sends the garbled labels corresponding to those bits to the SP. Next S_s^1 simulates step 7 by executing OT-extension with the SP which transfers the garbled labels corresponding to the SP's inputs. Given a secure implementation of OT-extension, the SP cannot distinguish the interactions from the real protocol execution. S_s^1 simulates step 7 by generating the GTTs corresponding to the GC used in phase 2 and sending them over to the SP for evaluation. Given a secure implementation of Yao's GC protocol, the SP cannot distinguish the GTTs from those received in a real protocol execution.

Simulation of the user's view: S_u^1 simulates the user's view of protocol V1. S_u^1 simulates the messages received from the SP in step 2, by generating and sending to the user two vector pairs containing randoms in Z_{2^q} that are indistinguishable from the two vector pairs containing the Φ and Ω values (i.e. Φ_P, Ω_P, and Φ_Q, Ω_Q) of the OLFE protocol received in the real protocol for the secure multiplications: P and Q. S_u^1 simulates the messages received from the SP in step 4, similar to how S_s^1 simulates step 4, as described before. Then S_u^1 generates random bits to represent the SP's inputs to phase 2 and simulates step 7 by executing OT-extension with the user. Given a secure implementation of OT-extension, the user cannot distinguish the above interaction from the real protocol execution.

Protocol V2

Simulation of the SP's view: S_s^2 simulates the SP's view in protocol V2 (see Figure 2.5). S_s^2 simulates the messages received from the user in step 1 by generating and sending to the SP two vectors containing randoms in $Z_{2^{q+s}}$ that are indistinguishable from the two vectors containing ϵ values (see Section 2.5.7) sent by the user in the real protocol for sharing the two private inputs of the user. Then S_s^2 simulates the messages received from the user in step 2 and 4 by generating and sending three sets of vectors, each set consisted of two vector pairs containing randoms in $Z_{2^{q+s}}$ that are indistinguishable from the vectors containing the shares of E and F values and their MAC values received in the real protocol for the three sets of secure multiplications computed via Beaver's MTs. Next, S_s^2 simulates the messages received in step 6 by generating and sending two random values in Z_{2^s} and $Z_{2^{q+s}}$ respectively, that are indistinguishable from the values $\tilde{p^u}$ and $\tilde{z^u}$ received in the real protocol for 'BatchCheck' procedure. S_s^2 simulates steps 7 and 8 similar to how S_s^1 simulates steps 6 and 7 of protocol V1, as described above, except that S_s^2's simulation is based on the GC given in Figure 2.3.

Simulation of the user's view: S_u^2 simulates the user's view of protocol V2. S_u^2 simulates the messages received from the SP in steps 1, 2 and 4 similar to how S_s^2 simulates the same, as described above. S_u^2 simulates step 6 by generating and sending a random in Z_{2^s}, that is indistinguishable from $\tilde{p^u}$ received in the real protocol for 'BatchCheck' procedure. S_u^2 simulates step 8 similar to how S_u^1 simulates step 7 of protocol V1, as described above, except that S_u^2's simulation is based on the GC given in Figure 2.3.

Theorem 2.4.1. *PEBASI is correct, secure and privacy preserving.*

Proof: It follows from Lemmas 1, 2, 3, 4, and 5.

2.5 Cryptographic Building Blocks

In what follows, we introduce the cryptographic primitives used in PEBASI.

2.5.1 Oblivious Linear Function Evaluation

Let Alice and Bob have two private input vectors A and X respectively, of length n where $A[\text{i}], X[\text{i}] \in Z_{2^q}$ for i $\in \{0, 1, .., n-1\}$. Alice and Bob securely compute additive shares of $R = AX$, using Protocol 2.2, which is an efficient construction of Oblivious Linear Function Evaluation (OLFE) [25], [72]. Let A^*, X^*, B^* and Z^* be four vectors containing pre-computed correlated randoms, s.t. A^*, X^* and B^* are sampled from Z_{2^q} and $Z^* = A^*X^*$ + B^*. A^*, B^* are held by Alice, and X^*, Z^* are held by Bob. Such correlated randoms are provided in the pre-processing phase. Note that B is a vector containing randoms chosen by Alice herself. Then Z (computed by Bob) and $-B$ are additive shares of R (i.e. $R = AX = -B + Z$).

2.5.2 Beaver's MTs based Protocol

Let Alice and Bob have two private input vectors A and X respectively, s.t. $A[\text{i}], X[\text{i}] \in Z_{2^q}$ for i $\in \{0, 1, .., n-1\}$. They want to securely compute the additive shares of Z=AX.

Protocol 2.2: Secure multiplication with OLFE

Inputs: Alice: A, B, A^*, B^*; Bob: X, X^*, Z^*.
Output: Bob learns $Z = AX + B$.
Protocol Execution:
1. Bob computes $M = X^*$ - X and sends M to Alice.
2. Alice computes $\Phi = A^* + A$ and $\Omega = A^*M + B + B^*$ and sends Φ and Ω to Bob.
3. Bob computes $Z = \Phi X + \Omega$ - Z^*

Protocol 2.3: Secure multiplication with Beaver's MTs

Inputs: Alice: A_A, X_A and P_A, Q_A, R_A. Bob: A_B, X_B and P_B, Q_B, R_B.
Output: Alice: Z_A, Bob: Z_B
Protocol Execution:
1. Bob chooses a random vector S, computes $X_B = X$ - S and sends $X_A = S$ to Alice.
2. Alice chooses a random vector T, computes $A_A = A$ - T and sends $A_B = T$ to Bob.
3. Bob computes: $E_B = (X_B - P_B)$, $F_B = (A_B$ - $Q_B)$ and sends E_B, F_B to Alice.
4. Alice computes: $E_A = (X_A - P_A)$, $F_A = (A_A$ - $Q_A)$ and sends E_A, F_A to Bob.
5. Both Bob and Alice computes $E = \sum_{\mathrm{i}} E_{\mathrm{i}}$ and $F = \sum_{\mathrm{i}} F_{\mathrm{i}}$, where $\mathrm{i} \in \{A, B\}$.
6. Bob computes $Z_B = R_B + F.P_B + E.Q_B$ and Alice computes $Z_A = R_A + F.P_A + E.Q_A + E.F$.

Let P^*_A, Q^*_A, R^*_A and P^*_B, Q^*_B, R^*_B be correlated multiplication triples provided by the BCA to Alice and Bob respectively, such that $(P^*_A + P^*_B)(Q^*_A + Q^*_B) = (R^*_A + R^*_B)$. At the end of protocol 2.3 [33], Alice and Bob learns Z_A and Z_B respectively, as additive shares of Z (i.e. $Z = AX = Z_A + Z_B$).

2.5.3 Garbled Circuits and OT

Yao's garbled circuits (GC) [73] protocol allows two parties (say P_1 and P_2) to securely compute an arbitrary function that is represented as a boolean circuit, without leaking any information about their inputs beyond what is implied by the function output. The garbled circuit creator (say P_1), generates garbled input labels for both 0 and 1 values of each input wire of the circuit and generates the garbled truth tables (GTTs) for each gate, by encrypting output wire labels using the input wire labels, and randomly permuting the truth table entries. P_1 sends the GTTs (a.k.a garbled circuit) and the labels corresponding to her input bits to P_2. Then P_1 and P_2 engage in a 1-out-of-2 oblivious transfer (OT) protocol for P_2 to obliviously obtain the garbled input labels corresponding to her input bits. Using the garbled inputs of the two parties, P_2 then evaluates the garbled circuit, gate by gate and decrypts the final output.

M-times 1-out-of-2 oblivious transfer of l-bit strings (denoted by $m \times \frac{1}{2}OT_l$) is a critical component of the GC protocol, where m is the number of input wires in the circuit which are associated with the GC evaluator's inputs and l is the bit length of a garbled input label. The $m \times \frac{1}{2}OT_l$ used in the GC protocol is defined as follows. P_1 inputs m pairs of strings $x_\mathrm{i}^0, x_\mathrm{i}^1 \in \{0,1\}^l$ $(1 \leq \mathrm{i} \leq m)$, P_2 inputs a bit string $r = (r_1, .., r_m)$ of length m, and P_2 obtains $x_\mathrm{j}^{r_\mathrm{j}}$ $(1 \leq \mathrm{j} \leq m)$ as output. OT ensures that P_1 learns nothing about r and P_2 learns nothing about $x_\mathrm{j}^{1-r_\mathrm{j}}$ [62]. An optimization to OT called OT-extension [62], [74] implements the $m \times \frac{1}{2}OT_l$ using a small fixed (k) number of OTs based on public key primitives, referred to as 'base' OTs, which can be pre-computed and extended to any number of OTs using only symmetric key primitives in the online phase. OT-extension significantly improves the performance of the online phase of SMPC protocols based on GC.

2.5.4 Homomorphic MAC Scheme

The role of a MAC scheme in actively secure two party computation is to prevent parties from lying about the computations they perform. Following are the key properties of an homomorphic MAC scheme: 1. A value x authenticated by the MAC scheme is represented by $[x]$ and has three components: x, MAC value of x denoted by m_x and MAC key α. 2. Given $[x]$ and a public constant c, enables local computation of $[x + c]$ and $[x \cdot c]$. 3. Given $[x_1]$, $[x_2]$, .. , $[x_n]$, enables local computation of $[y] = [x_1] + [x_2] + .. + [x_n]$ (i.e. an authenticated linear combination of a set of authenticated values).

2.5.5 SPDZ2k Protocol

SPDZ2k [55] presents a secret-shared, information theoretic MAC scheme that is homomorphic modulo 2^q and is as efficient as the standard solutions over a field (e.g. SPDZ [33]). The main ideas of SPDZ2k that are relevant to PEBASI V2 are as follows:

1. A global MAC key $\alpha \in Z_{2^s}$ is randomly generated, where s is a security parameter. Each party gets a random share $\alpha^{\mathrm{i}} \in Z_{2^s}$ s.t. $\alpha = \sum_{\mathrm{i}} \alpha^{\mathrm{i}} \bmod 2^s$.
2. Each secret value $x \in Z_{2^q}$ is shared between the two parties over a large ring $Z_{2^{q+s}}$, s.t. $x = \sum_{\mathrm{i}} x^{\mathrm{i}} \bmod 2^{q+s}$ and $x = x \bmod 2^q$. MAC value of x is also shared between the two parties over $Z_{2^{k+s}}$, s.t. $m_x = \alpha x \bmod 2^{q+s}$. Accordingly, $[x] = (x^{\mathrm{i}}, m^{\mathrm{i}}, \alpha^{\mathrm{i}})_{\mathrm{i}=1}^{2} \in (Z_{2^{q+s}} \times Z_{2^{q+s}} \times Z_{2^s})^2$, $\sum_{\mathrm{i}} m^{\mathrm{i}} = (\sum_{\mathrm{i}} x^{\mathrm{i}}) \times (\sum_{\mathrm{i}} \alpha^{\mathrm{i}}) \bmod 2^{q+s}$. Section 2.5.7 shows how two parties compute shares of a secret $[x]$, using a random mask r and it's authenticated shares. Although the computation is done in the larger ring, only the lower q bits of the output is considered for correctness of the computation.
3. Having shares of $[x]$, $[y]$, the two parties obtain $[x \cdot y]$, via Beaver's MTs based secure multiplication which involves authenticated MTs. SPDZ2k assumes the existence of a pre-processing phase that provides authenticated MTs. (Section 2.5.6 describes the pre-processing phase of PEBASI V2 in which the BCA provides such artifacts to the two parties). The 'AffineComb' procedure of SPDZ2k [55] enables the two parties to obtain authenticated results of their local computations.
4. SPDZ2k proposes the 'SingleCheck' procedure and the 'BatchCheck' procedure to open and verify the MAC value(s) of a single shared output and of multiple shared outputs respectively, in a privacy preserving manner.

Security of the SPDZ2k [55], is given by the following game: Let $x, m_x \in Z_{2^{q+s}}$ and $\alpha \in Z_{2^s}$. The adversary, who is given x, specifies the errors e_x, e_α, e_m and defines the modified values: $x = x + \mathrm{e}_x$, $\alpha = \alpha + \mathrm{e}_\alpha$ and $m = m + \mathrm{e}_m$. The adversary wins if $m = \alpha x \bmod 2^{q+s}$. Since the actual information is stored in the least significant q bits only, this is a forgery only if $\mathrm{e}_x \bmod 2^q \neq 0$. From this fact and the fact that a winning adversary is able to compute $\mathrm{e}_x \alpha \bmod 2^{q+s}$, it follows that such adversary has to guess $\alpha \bmod 2^{-s}$, which is only possible with probability 2^s. SPDZ2k enables the design of actively secure computation of arithmetic circuits over rings with statistical security of s - $\log(s)$. We refer the reader to the SPDZ2k paper [55] for the full security proof.

2.5.6 Authentication Artifact Refill for V2

The BCA pre-computes the artifacts required by the two parties in protocol V2 as follows:

1. Generate a global MAC key $\alpha \in Z_{2^s}$ and sample two shares s.t. $\alpha = \sum_{\mathrm{i}=1}^{2} \alpha^{\mathrm{i}} \bmod 2^s$.

2. Generate random masks and their authenticated shares which enable the two parties to compute authenticated shares of their private inputs. For each party P_{i} and for each of their private inputs x;

i) sample $r \in Z_{2^{q+s}}$ and set $m_r = \alpha r \bmod 2^{q+s}$.

ii) sample two authenticated shares: $[r_1]$, $[r_2]$ s.t $r = \sum_{\mathrm{i}=1}^{2} r^{\mathrm{i}} \bmod 2^{q+s}$ and $m_r = \sum_{\mathrm{i}=1}^{2} m_r^{\mathrm{i}} \bmod 2^{q+s}$.

iii) allocate r and $[r_1]$ to P_{i}, where $\mathrm{i} \in \{1, 2\}$ and $[r_2]$ to the other party.

3. Generate authenticated MTs. For each set of MTs:

i) sample a_1, a_2, b_1, $b_2 \in Z_{2^{q+s}}$.

ii) compute $a = \sum_{\mathrm{i}}^{2} a_{\mathrm{i}} \bmod 2^q$, $b = \sum_{\mathrm{i}}^{2} b_{\mathrm{i}} \bmod 2^q$.

iii) compute $c = ab \bmod 2^q$.

iv) compute shares of MAC values for all $x \in \{a, b, c\}$:

- project x to the larger ring by sampling $r \in Z_{2^s}$ and computing $x = 2^k r + x \bmod 2^{q+s}$.
- compute $m_x = \alpha x \bmod 2^{q+s}$.
- sample m_x^1, m_x^2 s.t. $m_x = \sum_{\mathrm{i}=1}^{2} m_x^{\mathrm{i}} \bmod 2^{q+s}$.

v) sample c_1, $c_2 \in Z_{2^s}$ s.t. $c = \sum_{\mathrm{i}=1}^{2} c_{\mathrm{i}} \bmod 2^{q+s}$.

vi) allocate the shares of $[a]$, $[b]$, $[c]$ computed in above (i)-(v) steps, to the two parties.

4. For the 'BatchCheck' procedure:

i) generate a random seed S which enables both parties to derive of the required public randoms. ii) generate a random $r \in Z_s$, compute $m_r = \alpha r \bmod 2^{q+s}$ and sample two shares of $[r]$.

5. Generate the following artifacts based on the GC given Figure 2.3, for the user and the SP respectively: i) garbled labels corresponding to the input bits of the two parties into phase 2. ii) signed GTTs.

2.5.7 Computing Authenticated Shares

Two parties (i.e. the user and the SP) obtain the authenticated additive shares of a private input x of party P_1 as follows. The same steps apply when sharing each private input of both parties. Note that the random mask r and its authenticated shares $[r_1]$, $[r_2]$ are provided by the BCA during the offline phase.

1. $P_1 \rightarrow P_2$: $\epsilon = x - r \bmod 2^{q+s}$.
2. P_1 computes $[x_1] = [r_1] + \alpha_1\epsilon \bmod 2^{q+s}$.
3. P_2 computes $[x_2] = [r_2] + \alpha_2\epsilon \bmod 2^{q+s}$.

2.6 Supporting Other Biometric Traits

We presented PEBASI focusing mainly on iris because it is considered to be a very reliable biometric trait. In addition, since its features are represented in binary format, we could use POTP for encryption while enabling efficient secure biometric matching on encrypted data. PEBASI can also support face and fingerprint biometrics without architectural changes to the authentication system and with only slight modifications to the protocols defined in Section 2.2.5. Features of both face and fingerprint biometrics are represented as integer vectors after applying two different feature extraction mechanisms, namely, eigen faces [75] and FingerCode [76] respectively and after quantizing the eigen faces based features. Biometric matching on faces and fingerprints are performed by first computing the Euclidean distance between the feature vectors and then comparing the distance with a pre-defined threshold ϵ.

During enrollment, the user provides to the BCA, a capture of her biometrics: F_E and a random secret vector (S) derived from the user's password. The BCA extracts the feature vector L_E from F_E. Then the BCA computes: $L_E = L_E + S$. Next the BCA computes two vectors L_E^u and L_E^s s.t. $L_E = L_E^u + L_E^s$). The BCA includes L_E^s in the signed enrollment token (ET) and returns it to the user along with L_E^u and any meta-data required to be used during the authentication time (e.g. eigen faces subspace computed from a set of training images drawn form a public dataset).

During authentication, the user takes a fresh capture of her biometrics (F_A) and extracts the feature vector L_A locally. Then the user derives S from the password and computes L^u_E = L^u_E - S, which cancels the randomization effect applied by the BCA on L_E. Then the user and the SP perform secure computation of the squared Euclidean distance (E) between L_E and L_A (see eq. (2.13)). Note that q is the size of the feature vectors (i.e. q=12 for face biometrics [26] and q=640 for fingerprint biometrics [29]).

$$\begin{aligned} D &= ||L_A - L_E||^2 \\ &= ||L_A - (L^s_E + (L^u_E - S))||^2 \\ &= ||L_A - (L^s_E + L^u_E)||^2 \\ &= \sum_{\mathrm{i}=1}^{q} L_A[\mathrm{i}]^2 - 2\sum_{\mathrm{i}=1}^{q} L_A[\mathrm{i}](L^s_E[\mathrm{i}] + L^u_E[\mathrm{i}]) \\ &+ \sum_{\mathrm{i}=1}^{q}(L^s_E[\mathrm{i}] + L^u_E[\mathrm{i}])^2 \end{aligned} \tag{2.13}$$

Let $X_1 = \sum_{\mathrm{i}=1}^{q} L_A[\mathrm{i}]^2$, $X_2 = \sum_{\mathrm{i}=1}^{q} L_A[\mathrm{i}](L^s_E[\mathrm{i}] + L^u_E[\mathrm{i}])$ and $X_3 = \sum_{\mathrm{i}=1}^{q}(L^s_E[\mathrm{i}] + L^u_E[\mathrm{i}])^2$ in the right hand side of the eq. (2.13). In order to compute the additive shares of the squared Euclidean distance: E^u and E^s, by the user and the SP respectively, the user locally computes X_1 and the two parties collaboratively compute X_2 and X_3 as follows in the two versions of PEBASI. In V1 of the extended protocol, X_2 and X_3 are computed using OLFE. This is because X_2 involves secure multiplication of secret pairs of *Type 2* and X_3 involves those of *Type 3* (see Section 2.3.2), which are shown to be more efficiently computed using OLFE than using Beaver's MTs. In V2 of the extended protocol, X_2 and X_3 are computed using Beaver's MTs along with the MAC scheme of SPDZ2k for all the secure multiplications, similar to the phase 1 of V2 of the original protocol. Once the additive shares of the Euclidean distance are computed, the two parties perform secure comparison of the distance against the threshold ϵ via Yao's GC protocol in the two versions of the extended protocol, similar to the phase 2 of the two versions of the original protocol.

3. PRIVBIOMTAUTH: A PRIVACY PRESERVING GENERIC AUTHENTICATION SCHEME BASED ON BIOMETRICS

In this chapter, we present our proposed privacy preserving and biometrics based online authentication protocol, named PrivBioMTAuth, which we introduced in Section 1.3.2. This chapter is organized as follows. Section 3.1 introduces the main concepts used in our approach. Section 3.2 explains our approach in detail. We present the details of the prototype implementation and the experimental evaluation in Section 3.3. We analyze security and privacy in section 3.4.

3.1 Background

In what follows we introduce the main concepts and techniques which are used as building blocks in our approach.

3.1.1 Pedersen Commitment

The Pedersen commitment [77] is a secure commitment scheme whose security is based on the hardness of solving discrete logarithms. The operation of this commitment scheme, which involves a committer and a verifier, can be described by following three steps.

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

Commit: The committer creates the commitment of $x \in Z_q$ by choosing $r \in Z_q$ at random and computing: $C(x,r) = g^x h^r$ mod p $\in G_q$.

Open: To open the commitment, the committer reveals x and r and the verifier checks if $C = g^x h^r$ to verify the authenticity of the commitment.

Protocol 3.1: Zero Knowledge Proof of Knowledge

1. $U \rightarrow V$: U randomly picks $y, s \in Z_q$ and sends $d = g^y h^s \in G_q$ to V.
2. $V \rightarrow U$: V sends random challenge $\mathrm{e} \in Z_q$ to U.
3. $U \rightarrow V$: U sends $u = y + \mathrm{e}x$ and $v = s + \mathrm{e}r$ to V.
4. V: accepts if $g^u h^v = dC^{\mathrm{e}}$.

The Pedersen commitment has two properties: it is unconditionally hiding - every possible value of x is equally likely to be committed in C, and it is computationally binding - one cannot open the commitment with any $x \neq x$, unless one can compute $log_g h$.

3.1.2 Zero Knowledge Proof of Knowledge Protocol

A zero knowledge proof of knowledge (ZKPK) protocol is a protocol by which the owner of a secret can prove to a verifier his/her knowledge about the secret without making it any easier for the verifier to obtain the actual secret. In our work, we use the protocol listed in Protocol 3.1 to prove the knowledge of the two secret values x and r hidden in the Pedersen commitment, without revealing the actual values of x and r to the verifier. This protocol has three properties: completeness - if the committer and verifier are honest, the protocol succeeds with overwhelming probability; soundness - the protocol does not allow the committer to prove a false statement; and zero knowledge - the proof does not leak any information about the secrets. Let U denote the committer and V denote the verifier.

3.1.3 Key Derivation from a Password

Our approach uses three secrets ($S_{\mathrm{i}} : \mathrm{i} \in \{1, 2, 3\}$) in different steps of the protocol: S_1 of size 128 bits, S_2 of size 160 bits, and S_3 of size 256 bits. In order to address usability concerns, such as the user having to enter three passwords during the execution of the protocol, and security concerns, such as having to store the secrets somewhere and the secrets not being uniformly randomly distributed in the key space, we make use of the password based key derivation function 2 (PBKD2) for deriving the two secrets from a single password provided by the user, which involves PKCS#5 as the pseudo random function (PRF) and a salt value to make dictionary attacks harder. The key derivation algorithm is thus as follows. We first

generate a secret S as:
S = PBKDF2 (PKCS#5, Password, Salt, derived key length (=544 bits)).
We then partition S into three parts of aforementioned sizes in order to form the three secrets.

3.1.4 Eigen Faces based feature extraction

There are two main categories of face recognition techniques namely: (i) appearance based and (ii) geometric facial feature based [78]. The Eigen faces based face recognition technique [75] belongs to the first category. The Eigen faces based feature extraction mechanism extracts features by projecting face images on to a feature subspace called "eigen faces". This subspace is computed by applying Principal Component Analysis (PCA) on a set of training face images. PCA is a dimension reduction method which projects n dimensional data onto K dimensional subspace where $K < n$. This K dimensional subspace identifies the dimensions with the maximum variance.

In order to compute the eigen faces based features of a given image I, we need to first compute the "eigen faces" subspace denoted by W, and the mean image X_{mean} from a set of training face images X, and then project the given image onto W, after substracting X_{mean} from I. In what follows, we discuss how the computation of these two stages is performed.

Computing the eigen faces subspace [78]

Each face image in the training data set, which is represented as a pxq matrix of pixel values, is converted into a vector of $p * q$ rows.
Let $X = \{x_1, x_2, x_3, .., x_n\}$ be a matrix containing the vector representations of n such face images.

1. Compute the mean: $\mu = \frac{1}{n} \sum_{\mathrm{i}=1}^{n} x_{\mathrm{i}}$

2. Compute the covariance matrix:
$$S = \frac{1}{n} \sum_{\mathrm{i}=1}^{n} (x_{\mathrm{i}} - \mu)(x_{\mathrm{i}} - \mu)^T$$

3. Compute the eigen vectors v_{i} and eigen values λ_{i} represented by the following equation:

   $Sv_{\mathrm{i}} = \lambda_{\mathrm{i}} v_{\mathrm{i}}$

4. Normalize the eigen vectors.

5. Order the eigen vectors in descending order by their eigen values and select the K eigen vectors corresponding to the K largest eigen values. These K eigen vectors form the eigen faces subspace which is referred to as W.

Projecting an image onto eigen faces subspace

Given a face image I, its eigen faces based features is extracted via the following steps, using the eigen faces subspace W and the mean image $X_{mean}(= \mu)$ computed in the previous stage.

1. Normalize the image : $I_N = \frac{I}{\|\mathbf{I}\|}$

2. Substract the mean image of the training set:

   $I_S = I_N - X_{mean}$

3. Project I_S onto W : $F_I = W^T I_S$

F_I is the set of features extracted from the image I via the eigen faces based feature extraction mechanism which can then be used for face recognition tasks.

3.1.5 Support Vector Machine

Given a set of training examples composed of pairs of the form $\{x_{\mathrm{i}}, y_{\mathrm{j}}\}$, the SVM classification technique finds a function $f(x)$ that maps each attribute vector x_{i} to its associated class label y_{j}, $\mathrm{j} = 1, 2, 3 \ldots n$ where n is the total number of classes represented by training data. The SVM is a discriminative classifier defined by separating hyper planes, that is, given the labeled training data, the algorithm outputs the optimal hyper planes (i.e. maximum separating hyper planes) which categorize new samples which are also known as testing data. The SVM algorithm includes a kernel function which maps training data to improve its resemblance to a linearly separable set of data. This increases the dimensionality of data. We incorporate the Radial Basis Function (RBF) kernel with optimal values for C and γ parameters [1] selected based on k-fold cross validation accuracy in grid search, as we discuss in Section 3.3.

3.2 Approach

Our authentication approach involves three entities, namely: (i) user - the entity which is authenticating using the biometric identity, (ii) service provider (SP) - the entity which authenticates the user before allowing the user to perform any transactions, and (iii) identity provider (IDP) - which vouches for the user's biometric identity. Our approach consists of two main phases: (i) enrollment phase - by which the user obtains her/his biometrics-based cryptographic identity token digitally signed by the IDP; (ii) authentication phase - by which the user proves her/his biometrics-based identity at the SPs.

3.2.1 Enrollment Phase

During the enrollment phase, a user is given: i) an identity token (IDT) digitally signed by the IDP, which encodes in a cryptographic commitment a secret derived from the user's biometrics and a secret derived from the user's password and ii) some secure artifacts. These secure artifacts enable the user to regenerate the secrets during the authentication phase.

In what follows we discuss the key challenges in designing the enrollment phase followed by a detailed description of the enrollment protocol. In the discussion we refer to any enrolled user in the authentication system that is different to a particular enrolled user of interest, as an *imposter*.

[1]↑C trades off misclassification of training samples against simplicity of the decision surface in the SVM. A low value of C makes the decision surface smooth, while a high value of C aims at classifying all training examples correctly. γ is a kernel specific parameter which determines the RBF width.

Deriving the biometric identifier (BID)

Due to the dynamic nature of biometrics, deriving a repeatable secret from a user's biometric template is a challenge. This secret should also be unique (so that it is hard to be guessed and bruteforced) in order to preserve the inherited uniqueness of raw biometrics and revocable in order to cancel it in case of a compromise. The technique that we use to obtain a repeatable BID from a user's biometrics is to train at enrollment a multi-class classification based machine learning model which predicts the class label that best represents the enrolling user's biometric features. Depending on its robustness, the trained classifier is expected to predict the same class label at all the authentication attempts by a specific enrolling user and to predict a different class label at the authentication attempts by an *imposter*. We decided to use multi-class classification approach as opposed to binary classification approach because of the requirement that the class label associated with an enrolling user must be unique and hard to guess. Such requirements would not be met by a trained binary classifier which outputs either 0 or 1. A random set of binary strings that are 128 bits long, is selected as labels of the training biometric features used to train the multi-class classifier. The BID is created by concatenating the class label (l), which is 128 bits long, predicted by the trained classifier (on an input of biometric features of a specific enrolling user), with the secret $S1$, which is also 128 bits long, derived from the user's password through function PBKDF2. Therefore, the BID takes the format in equation 1 and it is 256 bits long (BID mod $q \in Z_q$).

$$BID = l|S1 \tag{3.1}$$

The user can revoke an existing biometric identity derived using the aforementioned approach and request the IDP to issue a new IDT that encodes a new BID generated using a new password and a new trained classifier which associates a different class label with the user's training biometric features. Therefore, the BID generated according to our approach is repeatable, unique and revocable.

Selecting training data to train the classifier

The method for selecting the training data to train the classifier needs to take into account both security and usability. On one hand, since popular multi-class classification techniques, such as kernel based SVMs, encode the training data in the trained model and since the trained model is stored in the user's mobile phone in order to support user centric authentication, preserving the security and privacy of the training data is important. Although the classifier will be stored securely in the user's device, it is critical to encode the minimum amount of sensitive data in the trained classification model to minimize the impact on the authentication system in case in which the user's mobile device is stolen and the classifier is compromised. On the other hand, the training data used to train the classification model should be discriminative in order to make the trained model robust enough to map at the authentication the correct class label with the genuine user's biometric features and to not map the genuine user's class label with the *imporsters'* biometric features.

We experimented with three potential methods for selecting the training data to train the classifier for a specific user enrolling in the authentication system. We have carried out experiments to empirically evaluate the performance of the classifiers trained with the data obtained from those three methods in terms of False Rejection Rate (FRR) and False Acceptance Rate (FAR).

Note that in the context of the multi-class classification model that we use, FRR is the rate at which the trained classification model predicts a class label different than the one associated with the genuine user's biometric features at enrollment, when the genuine user's biometric features is given as input to the model at authentication. Usability decreases as FRR increases because the genuine user finds difficulties in authenticating. FAR is the rate at which the trained classification model outputs the class label that is associated with the genuine user's biometric features at enrollment, when an *imposter's* biometric features is given as input to the model at authentication. An authentication application becomes less secure as FAR increases because the probability that an *imposter* authenticates as the genuine user increases.

Table 3.1. False Rejection Rates over four trials

Method	Trial 1		Trial 2		Trial 3		Trial 4	
	FRR	STDV	FRR	STDV	FRR	STDV	FRR	STDV
Method 1	0.556	0.233	0.644	0.257	0.622	0.239	0.622	0.239
Method 2	0.022	0.083	0.044	0.113	0.022	0.083	0.044	0.113
Method 3	0.05	0.1	0.033	0.084	0.033	0.084	0.033	0.084

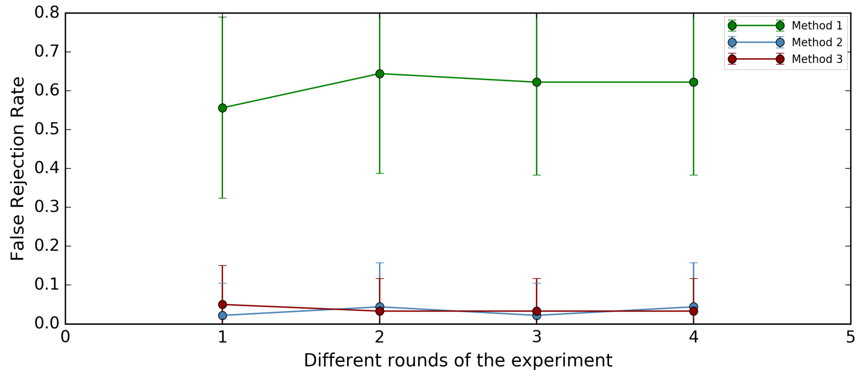


Figure 3.1. False Rejection Rates over four trials

Table 3.2. Variation of False Acceptance Rates with number of imposters

Method	3 Imposters		6 Imposters		9 Imposters		12 Imposters	
	FAR	STDV	FAR	STDV	FAR	STDV	FAR	STDV
Method 1	0.322	0.171	0.341	0.102	0.348	0.057	0.352	0.053
Method 2	0.0	0.0	0.022	0.036	0.053	0.065	0.056	0.044
Method 3	0.005	0.020	0.008	0.022	0.007	0.015	0.002	0.010

Based on the experimental results on the three methods summarized in the Tables: 3.1 and 3.2 and illustrated in the Figures: 3.1 and 3.2 (please refer the extended version of the paper [79] on PrivBioMTAuth for details on the three methods and their experiment results summarized here), we selected method 2 which achieves a balanced trade-off between security and usability, in order to select the data to train the machine learning model. This trained model is then used to derive the BID from the user's biometrics at authentication. In method 2, the training data is selected as the biometric features of a set of random users, who are outside the authentication system (so that they do not overlap with the space of potential *imposters*), and the biometric features of the current enrolling user.

With this method, we assume that the class label associated with a particular enrolling

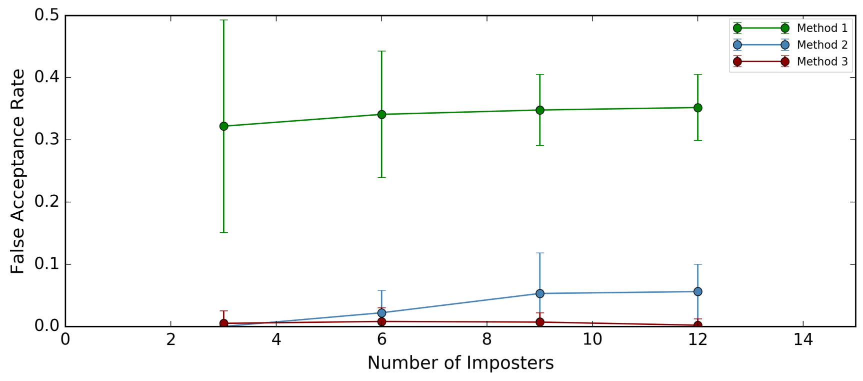


Figure 3.2. Variation of False Acceptance Rates with number of imposters

user's biometric features in the training data will be output by the trained model at the authentication attempts of the same enrolling user and that an *imposter's* biometric features will be matched with any of the random users' biometric features used to train the model. The only sensitive biometric features included in the classifier is the biometric features of the currently enrolling user. Therefore, in case of a compromise of the trained classifier, the biometric features of the other users enrolled in the authentication system will not be exposed. This method also allows us to achieve the desired FAR and FRR by varying the number of random users whose data is used to train the model (see Section 2.3.2 for experimental details).

Remarks: There are several approaches which can be used by the IDP to construct the dataset with random set of biometric features for training: (i) selecting one out of the many publicly available datasets; (ii) creating a set of synthetic biometric images; (ii) collecting a set of biometric images by the IDP itself for training purposes from a population which does not overlap with the population of enrolled users. In second and third approach, the IDP keeps the random biometric data set secret as such data sets become IDP's proprietary data. As we have already mentioned, the only requirement of such random biometric images is that, they are retrieved from subjects outside the authentication system, that is, subjects who are not the enrolled users of the authentication system. Therefore, the probability that the adversary knows which random images are involved in the training data of a particular users' classifier is negligible.

Protocol 3.2: Enrollment Protocol

Input from user: biometric images, password, meta-identity information (optional).

Input from IDP: biometric images of random set of users, private key of the IDP, public parameters of the Pedersen commitment.

Output from IDP: digitally signed IDT, trained model, salt-value for PBKDF2, key store.

Protocol execution:

```
User → IDP:
1. input of the user (over a secure channel)
IDP:
2. $F$ ← Training data prepared by extracting features from biometric images and
   assigning labels.
($l$ ← class label assigned to the training features of the enrolling user)
3. $M$ ← Classifier trained with with F.
4. $T$ ← Generated random salt value.
5. Derive three secrets from the user's password:
      S1, S2, S3 ← PBKDF2(PKCS#5, password, T, derived key length)
6. BID ← $l|S1$
7. Create IDT:
      i. Commitment (C) ← $g^x h^r modp$; where $x$ = BID and $r$ = S2.
      ii. IDT ← C | meta-data | Signature of IDP on {C | meta-data}
Generate a key pair, encrypt the trained model, protect the key store with S3.
IDP → User:
8. output of the IDP (over a secure channel)
```

Enrollment Protocol

In what follows we discribe the complete enrollment protocol designed based on the aforementioned design decisions. The enrollment protocol is executed between the user and the IDP and the steps are listed in Protocol 3.2. Note that when we refer to these entities, both human and software aspects related to them are involved. For example, when we refer to the user in the protocol, we refer to the actions taken by both the human user and the software installed in user's device.

When a request for enrolling a biometric identity is received at the IDP, along with the user inputs mentioned in Protocol 3.2, the IDP first selects a set of biometric images from a random set of users, following the method described in 3.2.1, in order to train the classification model for the current enrolling user. The number of such random users is decided based on the required assurance on FRR and FAR of the authentication application, as discussed in Section 2.3.2. Then the IDP extracts biometric features from the biometric images of both the enrolling user and the selected random users, using an appropriate feature extraction mechanism. The training data is constructed by assigning a random integer class label to the biometric features of each user and the classifier is trained using an appropriate learning algorithm.

In step 4 of Protocol 3.2, the IDP generates a random salt value which is given as input to the PBKDF2 function in step 5, along with the user's password and the required length of the secret to be derived. Then the BID is constructed in step 6 as described in Section 3.2.1. In step 7.i, the cryptographic commitment (C) is created by committing the two secrets, that is, the BID and the second secret derived from the user's password ($S2$), in the Pedersen commitment scheme. We leverage the properties of Pedersen commitment scheme described in Section 3.1.1 to hide the BID and $S2$ in the IDT. Finally, the IDT is created as the concatenation of the commitment, meta-data included in the IDT, and the digital signature of the IDP on the content of the IDT. Meta-data may include: serial number of the IDT, the public parameters of the Pedersen commitment (to be used in the authentication protocol), the expiration timestamp and any meta-identity information provided in the user input. The meta-identity information helps the SP to identify the user via some other identity attributes such as name, email, social security number etc., at authentication. We utilize the standard PKI based digital signature for digitally signing the IDT. The secure channel mentioned at the information exchange steps of Protocol 3.2 refers to a channel with message level security.

In order to allow a user to perform user-centric biometrics based authentication from her/his mobile phone without involvement of the IDP, the IDT, the trained model, the salt value (T) used in deriving the secrets based on the user's password, and the trusted software that executes the user's part of the authentication protocol are provided to the user by the IDP at the end of enrollment. The artifacts that contain sensitive information, such as the trained model are encrypted using the private key of a key pair generated for each enrolling user by the IDP. This key pair is stored in a keystore which is password-protected by S3. These secure artifacts are stored in the sandboxed internal storage of the IDP software application or in the Trusted Execution Environment (TEE) enabled in the modern mobile devices [80].

Remarks:

- The main motivation for using the user's password to generate the three secrets (S1, S2 and S3) used in the protocol is to avoid having to store the secret keys on the device and instead having them generated based on input by the actual user, each time the authentication is performed. This mitigates the risk of an attacker being able to steal the secrets kept in a storage.

- In order to prevent a malicious user from providing fake biometric images instead of her/his own biometric images in user input, for high assurance authentiation, the enrollment protocol should be executed at the IDP following the necessary legal processes such as requiring the user to visit the authority in person and proving her/his identity using a legal identifier that she/he possesses, such as a passport, which is outside the technical scope of our current work.

3.2.2 Revocation of the biometric identity

Revocation of an issued IDT can be initiated by either the IDP or the user. An IDT which is expired based on the expiration timestamp issued by the IDP at enrollment is also considered as revoked. Once an existing IDT is revoked, the user may request a new IDT by initiating the enrollment protocol and the IDP may issue a new IDT as mentioned in Section 3.2.1, by creating a new commitment using a new password provided by the user and a new trained classifier.

It is critical that SP be able to check for IDT revocation without undermining user's privacy. There are three main requirements: 1) user privacy - the identity of IDT owner should not be revealed to the IDP unless the IDT is in the revocation list; 2) revocation list hiding - the IDTs in the revocation list should not be revealed to the SP; 3) SP authorization - the SP cannot execute an arbitrary number of revocation checks without detection. The most common revocation checking mechanism is the Online Certificate Status Protocol (OCSP), in which the verifier sends the serial number of the certificate to the revocation authority and obtains the decision whether or not the given certificate is revoked. This violates the first requirement mentioned above, as it exposes a legitimate user's transaction patterns to the IDP. On the other hand, the IDP can let the SP obtain the entire revocation list to perform revocation check locally. While this addresses the first requirement, it violates the second and third requirements. The second and third requirements are important because someone who steals an arbitrary number of IDTs should not be able perform revocation checks on them in order to learn which of them are valid, without the risk of being detected. Therefore, a privacy preserving revocation checking mechanism that addresses all three requirements should be coupled with our authentication scheme. There are several approaches that address privacy-preserving revocation [81]–[83]. Both the approaches proposed in [81], [82] require the SP to query a range of K elements instead of the single token being verified. The degree of privacy depends on the size of this interval [83].

On the other hand, the approach proposed in [83] is based on an efficient private set intersection scheme [84]. The protocol consists of two phases: $Init$ and $Query$. During the $Init$ phase, the IDP creates a revocation list that hides the serial numbers of the revoked IDTs and sends it to the SP. During the $Query$ phase, the SP sends the blinded serial number of the IDT being verified, to the IDP, who then includes a RSA blind signature on it. The SP then unblinds the signature and checks it against the list sent by the IDP during the $Init$ phase. If there is a match, the IDT being verified is revoked. We refer the reader to [84] for further details. Communication complexity and computation complexity for the issuer of the $Init$ protocol is in the order of the size of the revocation list. No computation is required from the verifier in the $Init$ protocol. Both communication and computation complexity is constant for the $Query$ protocol. The cost of the $Init$ protocol is amortized if the same version of the revocation list is used over multiple runs of the $Query$ protocol. Such a scheme can be directly integrated into PrivBioMTAuth, as it addresses all the three requirements mentioned above.

3.2.3 Authentication Phase

At authentication, the user proves her/his biometric identity to the SP by proving the ownership of the IDT signed by the trusted IDP. This is accomplished by proving in zero knowledge, the knowledge of the two secrets, that is, the BID and the secret derived from the user's password ($S2$), that are encoded in the cryptographic commitment of the IDT. Accordingly, this is a three-factor authentication protocol which involves the user's IDT, her/his biometric image and the password. In what follows we present the basic authentication protocol in which the client authenticates to the SP, and then we extend it into a stronger authentication protocol which includes a key agreement phase that enables both parties to derive a secret key to verify authenticity of each other and encrypt the subsequent communications.

Basic Authentication Protocol

As shown in Protocol 3.3, the user initiates the authentication protocol by sending the authentication request along with the IDT, a helper commitment d and optionally, some meta-identity information. d encodes two random secrets y and s in a Pedersen commitment, which helps later in the protocol in proving the actual secrets encoded in the IDT. The SP first checks the validity of the IDT by verifying the signature of the IDP, the expiration timestamp, the revocation list and any meta-identity information sent by the user against the information included in the signed IDT. The meta-identity information may help the SP to associate the authenticating user with the user account held at the SP. Next the SP creates a challenge and sends it to the user. To prove the knowledge of the secrets against the challenge, the user first re-generates the BID using a newly captured biometric image, the password, the trained model, and the salt value, as shown in the steps 5-10 in Protocol 3.3. Then the user computes the proofs u and v as shown in step 11 and sends them to the SP. Upon verifying the zero knowledge proof as per step 13, the SP accepts or rejects the user's biometric identity based authentication. If the authentication succeeds, the user and the SP establish a session to carry out the transaction.

Protocol 3.3: Basic Authentication Protocol

Input from user: IDT, helper commitment ($d = g^y h^s mod p \in G_q$; where $y, s \in Z_q$ are random secrets), biometric image, password, meta-identity information(optional)

Output from SP: Authentication result: success/failure

Protocol execution:

```
User → SP:
1. authentication request with IDT, $d$ and meta-identity.
SP:
2. Verify the validity of the IDT.
3. Create a random challenge: $e \in Z_q$
SP → User:
4. challenge: e
User:
5. I ← Newly captured biometric image.
6. $S1$, $S2$, $S3$ ← PBKDF2(PKCS#5, password, salt-value(T), derived key length).
7. Decrypt the trained model (by using $S3$ to open the keystore).
8. f ← Features extracted from I.
9. $l$ ← Predicted class label for f.
10. $BID \leftarrow l|S1$
11. Computes: $u = y + ex$ and $v = s + er$; where x = $BID$ and r = $S2$
User → SP:
12. $u$ and $v$
SP:
13. Verifies if $g^u h^v = dC^e$; where $C$ is the commitment in the IDT.
SP → User:
14. Authentication result.
```

Based on the properties of standard ZKPK protocol described in Section 3.1.2, any information transmitted from the user to the SP in Protocol 3.3 does not help the SP to learn any additional information about the secrets encoded in the IDT. However, it helps the SP to verify with confidence whether or not the prover is the actual owner of the biometric identity encoded in the IDT.

Limitations of the Basic Authentication Protocol

Although none of the information exchanged in Protocol 3.3 is sensitive, we have to rely on Transport Layer Security (TLS) protocol, such as HTTPS, in order to verify the identity of the SP whom the user is interacting with and to establish a secure communication channel for exchanging any sensitive information of subsequent transactions. This is similar to the use of TLS in traditional user name and password based authentication protocols. However, even if Protocol 3.3 is followed by TLS, security issues still arise.

1. Identity theft attack by a malicious SP.

   This is a known man-in-the-middle type attack on ZKPK based identity verification protocols [39], also known as Mafia attack. This attack can be carried out by a malicious SP with whom the user performs a ZKPK based identity verification. When the user sends an authentication request to the malicious SP, this SP simultaneously initiates an authentication request to some other genuine SP, claiming the user's identity. When the genuine SP sends the challenge, the malicious SP simply forwards it to the user. When the user submits the identity proof, the malicious SP uses this proof to authenticate to the genuine SP by impersonating the user. (Protocol A in the extended version of the paper [79] on PrivBioMTAuth lists the steps of this attack based on the steps of the standard ZKPK based identity verification). This type of identity theft attack is possible because the basic authentication protocol does not include a mechanism for the two parties to verify that a man-in-the-middle attack has not taken place during authentication, before carrying out the transaction.

2. Exposure of sensitive information about the transactions at the intermediaries of the communication path.

   Because TLS only protects communication at the transport layer, confidentiality and integrity is not guaranteed at the intermediaries of the communicatin path. This could lead to certain attacks, such as session hijacking attacks, in which a malicious party can steal the session id of the user (which is provided by the SP at the end of the successful execution of Protocol 3.3) and perform transactions on behalf of the user, using the stolen session id. Therefore, it is preferable to have message level security in order to secure sensitive information.

Extended Authentication Protocol

The solution to those issues associated with the basic authentication protocol is to integrate a key agreement mechanism with the identity verification phase which serves two purposes, namely: i) helps mitigating the man-in-the-middle impersonation attack by allowing the user and the SP to verify the authenticity of each other ii) establishes a session based key for secure communication. We extend our basic authentication protocol to a strong authentication protocol which is listed in Protocol 3.4. The steps that are additional with respect to the basic authentication protocol are shown in bold font. Accordingly, in the step 4 of the identity verification phase, the SP sends two parameters a and b to the user, in addition to the challenge e. In the key agreement phase, the user derives the secret key using a, b and the secrets known to the user, while the SP derives the same key using the random secret w (which was used to create a and b), the commitment C in the IDT and the helper commitment d. The user and the SP then uses the derived key to: i) perform a handshake to verify that the impersonation attack has not taken place during the identity verification phase ii) secure the subsequent communication along with a symmetric encryption scheme (for confidentiality) and a keyed cryptographic hash function (for integrity). Secure communication enabled by Protocol 3.3 achieves forward secrecy as the communication is encrypted using per-session keys, any future compromise of the user's BID or the password will not compromise the past session keys, thereby preserving the secrecy of the past communication. A malicious SP could still perform the steps of the impersonation attack until the end of the identity verification phase of Protocol 3.3. However, it cannot continue the communication with the genuine SP beyond that point as it can not derive the secret key and hence cannot succeed in the handshake. It is important to note that tying the key agreement phase to the identity verification phase is critical. Otherwise, if the key agreement is performed independently after the identity verification is has been completed, malicious SP can derive two different secrets with the user and the genuine SP separately and carry out the impersonation attack.

Our authentication approach thus achieves our main goals: i) it avoids storing biometrics either at the IDP or at the SP ii) it avoids the revealing of biometrics at the SP at

```
Protocol 3.4: Extended Authentication Protocol
  Input from user: IDT, helper commitment ($d = g^y h^s$ mod p $\in G_q$; where $y, s \in Z_q$
    are random secrets), biometric image, password, meta-identity
    information(optional)
  Output from SP: Authentication result: success/failure
  Protocol execution:
  Identity Verification Phase:
  1. User $\rightarrow$ SP: authentication request with IDT and $d$.
  SP:
  2. Verify the validity of the IDT.
  3. i. Create a random challenge e $\in Z_q$
     ii. Create a random $w \in Z_q$
     iii. Compute $a = g^w$ mod p and $b = h^w$ mod p
  4. SP $\rightarrow$ User: e, a, b
  User:
  5. I $\leftarrow$ Newly captured a biometric image.
  6. $S1$, $S2$, $S3$ $\leftarrow$ PBKDF2(password, salt-value(T), derived key length)
  7. Decrypt the trained model (by using $S3$ to open the keystore).
  8. f $\leftarrow$ Features extracted from I.
  9. $l$ $\leftarrow$ Predicted class label for f.
  10. $BID \leftarrow l|S1$
  11. Computes: $u = y + \mathrm{e}x$ and $v = s + \mathrm{e}r$; where x = $BID$ and r = $S2$
  12. User $\rightarrow$ SP: $u$ and $v$
  SP:
  13. Verifies if $g^u h^v = dC^{\mathrm{e}}$
  14. SP $\rightarrow$ User: Authentication result.
  Key Agreement Phase:
  User:
  15. $K_{user} = a^{(x+y)}.b^{(r+s)}$ mod p $= g^{w(x+y)}.h^{w(r+s)}$ mod p.
  SP:
  $K_{sp} = C^w.d^w$ mod p $= (g^x h^r)^w.(g^y h^s)^w$ mod p $= g^{w(x+y)}.h^{w(r+s)}$ mod p
  16. User $\leftrightarrow$ SP: Secure handshake and communication.
```

authentication iii) it avoids the involvement of the IDP at authentication iv) it provides a mechanism to derive a unique, repeatable and revocable BID from the user's biometrics to be used for user-centric authentication and iv) it protects against known attacks on ZKPKP identity verification protocol.

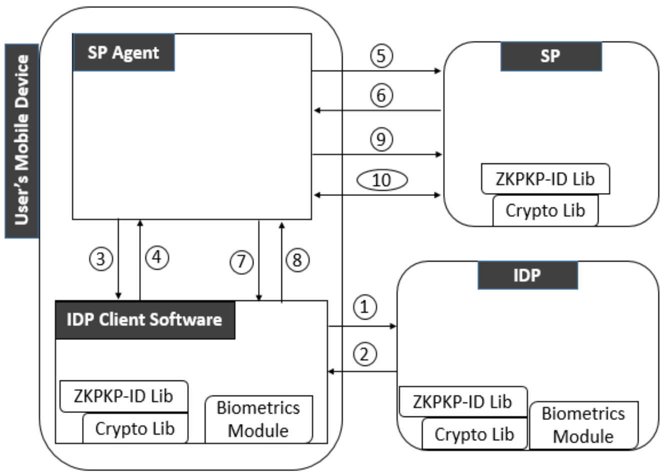


Figure 3.3. Overall Architecture of the Authentication Solution

3.3 Implementation and Experiments

In this section, we present the architecture of the prototype of our approach and the experimental evaluation.

3.3.1 Architecture

Figure 3.3 shows the main components of the authentication system and the flow of the execution of the protocols in high level. Details of the components and interactions among them are given in what follows.

Components

There are three main components that represent the three main entities described in Section 3.2. They are: i) the software component in the user's mobile phone ii) the IDP software component and iii) SP software component. The software in the user's mobile phone consists of two main sub components: IDP-client and SP-client. The IDP-client is provided by the trusted IDP and performs enrollment and facilitates authentication. The SP-client is the client application provided by the SP such as the client application provided by banking and e-commerce providers. Several SP-clients can be installed in a user's mobile phone. The software in the user's mobile device is divided into two applications for two reasons: i) *component re-use*: the IDP-client encapsulates the key steps of the user's part of the authentication protocol such as capturing and processing of biometrics, key derivation from the user's password and creation of the helper commitment and the zero-knowledge proofs. All the SP-clients installed in the user's mobile phone can consume this functionality when performing user authentication with their corresponding SPs. This ensures that the authentication related critical functionality is not duplicated and is transparent to the developers of SP-clients ii) *securing user credentials and authentication artifacts*: because the user enters the credentials only at the IDP-client and the authentication artifacts provided by the IDP is only accessed by the IDP-client during the authentication, they are not exposed to third party SP-clients.

We have developed three self-contained modules, namely: Crypto Lib, ZKPK-ID Lib and Biometrics module, which encapsulate different building blocks of the protocols and which are re-used across multiple components of the solution as illustrated in Figure 3.3. Further details about these modules and the implementation can be found in the extended version of the paper [79] on PrivBioMTAuth.

Flow

Numbered arrows in Figure 3.3 illustrate the order of the interactions between the entities during the enrollment and the authentication protocols. Steps 1 and 2 represent the enrollment protocol executed between the user and the IDP, at the end of which the user obtains the authentication artifacts mentioned in Protocol 3.2. When the user initiates authentication with a particular SP via the corresponding SP-client, the SP-client first sends a request to the IDP-client in step 3, in order to obtain the initial authentication elements, such as the IDT and the helper commitment. These are given to the SP-client by the IDP-client in Step 4. The SP-client then sends an authentication request to the SP in step 5 along with the obtained authentication elements. The SP returns the challenge (and the two parameters used for key derivation if the extended authentication protocol is used) in step 6. The SP-client forwards the challenge (and the two parameters) to the IDP-client in step 7. The IDP-client then prompts the user to enter her/his biometrics and password in order to create the zero-knowledge identity proofs (and the agreed key) which are handed back to the SP-client in step 8. The SP-client then forwards the proofs to the SP in step 9. If the zero-knowledge proof is successful, the SP-client and the SP establish a secure session (or carry out the secure handshake using the derived key, if the extended authentication protocol was used) in step 10 for further communication.

3.3.2 Experiments

The goals of our experiments are two folds: i) evaluating the performance of the classifiers trained using the training data obtained using the method described in Section 3.2.1 and observe how the number of random users involved in a classifier affects FRR and FAR ii) measuring the end-to-end execution times our authentication scheme and resource consumption of the mobile applications in order to identify any potential bottlenecks.

Experimental Setup

We hosted the IDP and the SP software in an Apache Tomcat web server running in a laptop machine with Ubuntu 14.04 OS, Intel Core i7-3537U CPU, and 5 GB memory. The two mobile applications were deployed in a mobile phone with the model number 'Moto G' [85] and Android version 5.1. The mobile applications communicated with the IDP and the SP web services over a wireless network with a speed of 140 Mbps. We used 'faces' as the biometric trait and utilized the publicly available 'AT&T' face dataset provided by AT&T Laboratories Cambridge **att** which contains face images of 40 individuals with 10 images from each. 'Eigen Faces' was used as the feature extraction mechanism [78] and 60 Eigen components were extracted from each image since most of the variation in the image data was captured in the first 60 eigen components [78]. Therefore, each biometric feature vector consisted of 60 elements. SVM was used as the classification technique and the Java version of LibSVM [86] library was used to implement the classification model. We used the 'C-SVC' SVM type which is intended for multi-class classification and the 'Radial Basis' function as the kernel function. After selecting the training data for each experiment, the optimal pair of values for the C and γ parameters were selected to train the SVM model by evaluating the 5-fold cross validation accuracy (CV accuracy) of each combination of values within the ranges of {-6, 6, 1} for C and {-10, 0, 1 } for γ, using grid search. Finally the SVM classifier was trained using the C and γ values selected with the best CV accuracy.

Evaluation of the trained classifiers

In what follows we discuss the experimental evaluation of the classifiers trained using the training data obtained from the method described in Section 3.2.1, in terms of FRR and FAR.

We performed the experiment by varying the number of random users involved in the trained classifier, for a given number of enrolling users in the authentication system. In this experiment, we assume three authentication systems with varying number of enrolled users (n) as 9, 12 and 15. In each such authentication system, we train six different classifiers for each enrolled user with varying number of random training users (x) as 9, 12, 15, 18, 21 and 24. For each of the six experiments related with each of the three authentication systems, the dataset was divided into two sets: i) set of random users with 40 - n number of users and ii) set of enrolling users with n number of users. The six classifiers of each enrolling user are trained using 6 images of randomly selected x number of users from the set (i) and 6 images of the enrolling user her/him self. Mean FRR was computed over the false rejections made on 3 images out of the 4 testing images of each of the n number of enrolling users by each of their six classifiers. Mean FAR was computed over the false acceptances made on the 4 testing images of n - 1 number of *imposters* by the six classifiers of each of the n enrolling users. FRR and FAR results are reported in the tables 3.3 and 3.4 and in the graphs 3.4 and 3.5.

Accordingly, FRR stays in a constant range, i.e: below 0.075, for the three authentication systems except for one case (i.e: the one with 12 random users in the trained classifier) in the authentication system with 9 enrolled users. High standard deviation associated with this case indicates that it is an outlier. There is no specific pattern in the variation of FRR based on the number of random users involved in the trained classifier, in an authentication system with a given number of enrolling users. In contrast, FAR decreases with the increasing number of random users in the classifier for all three authentication systems. FAR achieves 0.01 (commonly accepted FAR [87], [88]) when the ratio between number of enrolling users and number of random users is 1:2. This is proved for the authentication systems with 9 and 12 enrolling users. It could have been proved for the authentication system with 15 enrolling users as well, if there were 30 random training users, as the trend indicates. However, with 15 enrolling users, we could go only up to 25 random training users, because our data set size is 40.

The interesting result that we can observe from this experiment is that 'number of random users in the trained model' acts as a discriminative threshold for FAR, analogous to 'distance threshold' in a distance based biometric matching system (i.e: in a distance based biometric matching system, FAR increases and FRR decreases with the increased distance threshold). Therefore, based on the security and usability trade-off requirement of a particular application, one can select the appropriate ratio between the number of random users involved in the trained model and the number of enrolling users (or the number of imposters) in the authentication system.

End-to-end performance

As mentioned before, the second goal of the experiments is to measure the execution times of the main steps of both enrollment and authentication protocols as well as to observe the resource consumption by the IDP-client in the mobile phone during the execution of the authentication protocol. Since all the critical biometric processing steps of the enrollment protocol are executed by the IDP, we measured the execution times of such steps in the IDP software by taking the average execution times over 100 runs of the corresponding functions, as listed in Table 3.5. On the other hand, since all the critical biometric processing steps of the authentication protocol are executed in the mobile phone, we implemented a performance test suite by utilizing Android's instrumented test framework which automates the inter-application interactions, so that we could automatically run the end-to-end authentication protocol (which involves the aforementioned two mobile applications) in the mobile phone

Table 3.3. Variation of False Rejection Rates with number of enrolling users and number of random users in the model

No. of random users	9 enrolled users		12 enrolled users		15 enrolled users	
	FRR	STDV	FRR	STDV	FRR	STDV
9	0.0	0.0	0.027	0.092	0.0	0.0
12	0.111	0.179	0.0	0.0	0.022	0.083
15	0.037	0.104	0.027	0.092	0.022	0.083
18	0.037	0.104	0.027	0.092	0.022	0.083
21	0.074	0.138	0.0	0.0	0.044	0.113
24	0.037	0.104	0.027	0.092	0.044	0.113

for multiple times and take the average execution times. The breakdown of the execution time between the key steps of the authentication protocol is listed in Table 3.5.

Accordingly, the feature extraction step takes the most time as the eigen faces based feature extraction algorithm involves matrix multiplication which is a function known to be slow in Java, as discussed in **sof**. Performance benchmarks such as **perf** indicate that the running time could be improved by using certain perfromance enhanced libraries for matrix multiplication, which we did not experiment with, in the scope of this work, as our goal is to investigate whether it is feasible to carry out our proposed privacy preserving, user centric and biometrics based authentication approach from a mobile device, and not to achieve the best performance possible.

We compared our ZKPK based biometrics identity verification approach, with the ZKPK based static identity (such as: email, credit card number, social security number etc.) verification scheme in terms of their end-to-end execution times (see Table 3.6) and resource consumption in the mobile phone during the authentication phase (see Table 3.7).

ZKPK based static identity verification is a well known scheme which was first proposed by Feige et al [36]. It allows users to prove ownership of their static identities without revealing them to the verifiers. We built this functionality also into the IDP-client mobile application for comparison purposes. In the static identity based ZKPK authentication protocol, a user enters a static identity instead of the biometric trait when she/he is prompted to enter credentials by the IDP-client (in step 7 of Figure 3.3). Then such static identity is cryptographically hashed and used as secret x while the password is cryptographically hashed

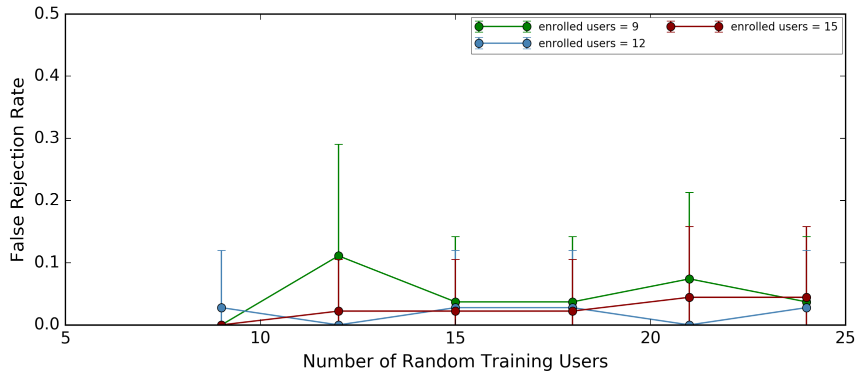


Figure 3.4. Variation of False Rejection Rates with number of enrolling users and number of random users in the model

Table 3.4. Variation of False Acceptance Rates with number of enrolling users and number of random users in the model

No. of random users	9 enrolled users		12 enrolled users		15 enrolled users	
	FAR	STDV	FAR	STDV	FAR	STDV
9	0.050	0.054	0.053	0.081	0.076	0.087
12	0.027	0.043	0.030	0.066	0.074	0.062
15	0.018	0.028	0.037	0.071	0.063	0.080
18	0.009	0.017	0.027	0.045	0.058	0.069
21	0.013	0.039	0.022	0.051	0.023	0.044
24	0.009	0.017	0.010	0.025	0.023	0.037

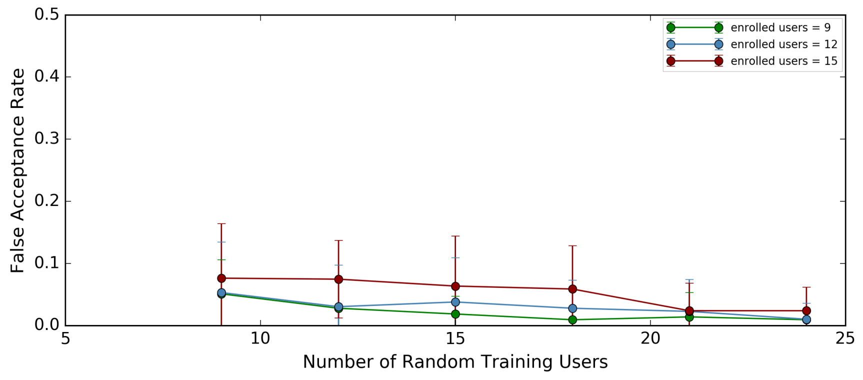


Figure 3.5. Variation of False Acceptance Rates with number of enrolling users and number of random users in the model

Table 3.5. Breakdown of the execution times of biometrics processing steps of the enrollment and authentication phases

Enrollment Phase	
Steps	Execution Time (ms)
Image reading and feature extraction (of 21*4 images)	4833.0
Parameter selection	7039
Training the classifier	23
Authentication Phase	
Steps	Execution Time (ms)
Reading the image	1.0
Feature extraction	13825.0
Prediction	120.0

and used as secret r in commitment C. Therefore, ZKPK based static identity verification does not involve any feature extraction step, classification based prediction step or derivation of multiple secrets from the password. It also does not involve any authentication artifacts stored in the user's mobile phone except the identity token given by the IDP at the end of the enrollment protocol.

According to the experimental data in Table 3.5 and Table 3.6, the additional time taken in the end-to-end execution of the ZKPK based biometric identity verification scheme when compared with the ZKPK based static identity verification scheme, in both enrollment and authentication phases, is due to the time taken by the biometric processing steps - specially the time taken by the feature extraction step, as previously discussed. According to the comparison of the resource consumption in the mobile phone during the authentication protocol, the IDP-client application shows higher performance requirements in the ZKPK based biometric identity verification scheme in terms of CPU usage, memory and storage, while the communication cost is the same for both the schemes. Both schemes exchange the

Table 3.6. Comparison of the end-to-end execution times of the authentication protocols

Protocol	Type of the identity verification	
	Static Identity	Biometric Identity
Enrollment	199.0(ms)	12194(ms)
Basic Authentication	2666.0(ms)	15423(ms)

Table 3.7. Comparison of the resource consumption in the mobile phone during the basic authentication protocol

Resource	Type of the identity verification	
	Static Identity	Biometric Identity
CPU	2%	47%
Memory	48598.0KB	112448.0
Communication Cost	4.776KB	4.776KB
Storage	1.6MB	7.5MB

same information between the prover and the verifier during the authentication protocol. The communication cost shown in Table 3.7 further breaks down as: 3.589KB of transferring cost and 1.187KB of receiving cost. Transfer cost is higher than the receiving cost because the prover sends more elements than what verifier does. Storage requirement is higher because in our implementation, we have used eigen faces as the feature extraction mechanism and SVM as the classification technique, which contribute eigen sub space W, the mean image X_{mean} (see Section 3.1.4) and the trained SVM classifier to the artifacts stored in the mobile phone. Note that in the prototype of our system which is used to measure end-to-end performance, we involved 20 random users in the trained classifier. CPU and memory usage is higher due to the expensive operations take place during the biometric processing phase, such as loading the artifacts into the memory and performing multiplication of large matrices.

In summarizing the information presented in this section, we first experimented with the proposed method for selecting training data to train the classification model which achieves the accepted performance of a biometrics authentication system, based on the AT&T face dataset. According to the paper: 'Guidelines for Best Practices in Biometrics Research' by A. Jain et al [87], accepted FAR is about 1.0% (i.e: 0.01) and according to the talk: 'Biometrics Shor Course' by Andrew S. et al [88], FRR reported at 1% FAR on the FVRT dataset is 25%. Figure 3.5 shows that the applications for which achieving a low FAR (at the cost of relatively high resource consumption) is a high priority, can achieve the aforementioned FAR by employing a classifier trained with the data obtained from the method proposed in 3.2.1 and an appropriate ratio between the number of random users involved in the trained model and the number of enrolled users in the authentication system. Second, we analyzed the performance of the end-to-end authentication protocol and observed that the only bottleneck is in the feature extraction algorithm that we employed. Overall the performance proves the feasibility of the proposed user centric and biometrics based authentication approach for mobile phones given that the experiments were carried out in a very low end mobile phone like 'Moto G', and that no optimization technique was applied in the implementation of the matrix multiplication step. It is important to note that the privacy preserving biometrics based authentication solution that we have presented in Section 3.2 is very generic so that one can plug-in any other better and optimized algorithms for commitment, feature extraction and classification steps by using the extension points provided in our prototype implementation.

3.4 Security and Privacy Analysis

In this section we formally define PrivBioMTAuth scheme and analyze its properties.

3.4.1 Modelling PrivBioMTAuth

There are three principal entities: User U, IDP and SP. We will also differentiate the human user and the user agent running in the user's device. Let the user's device be denoted by D. Before authentication can take place, user enrolls with the IDP. After the enrollment is completed, user can authenticate with SP using biometrics.

Definition I: We define two procedures of PrivBioMTAuth scheme: Enroll and Auth.

Enroll: is a protocol between U and IDP. If the enrollment is successful, U obtains the IDT and other artifacts, which are stored in D.

Auth: is a protocol between U and SP, where U uses biometrics, password, IDT and D to generate the proofs of authentication for SP to verify

Definition II: PrivBioMTAuth (Enroll, Auth) is a scheme with following properties:

1. *Complete*: A genuine user can authenticate successfully.

2. *Secure*: A computationally bounded adversary has a negligible success probability in breaking the Auth procedure and impersonating a genuine user.

3. *Privacy-preserving*: No sensitive information about the biometrics or the password is leaked from the IDT, D or from the transcripts of *Enroll* and *Auth* protocols. IDP will not learn any information about the user's interactions with SPs, as long as the IDP follows the protocol and does not collude with the SPs. (i.e: IDP is modelled as an honest but curious adversary against preserving user's transaction privacy).

Aforementioned properties cover the goals that we want to achieve. Now we define the building blocks of Enroll and Auth procedures and their properties.

Definition III: BID-Gen is a function f:{biometric, password} $\rightarrow$ BID ($\{0,1\}^{160}$); bound to a user U, with following properties:

1. *Unique*: BID is unique for each U. i.e: Pr[eq(x,y)=1 | x$\leftarrow$ BID-Gen($BIOM_u$, PW_u), y$\leftarrow$BID-Gen($BIOM_{u1}$, PW_{u1})] $\leq \epsilon_1$, for neglegibly small ϵ_1, where eq stands for equality, $BIOM$ stands for biometrics and PW stands for password.

2. *Repeatable*: Output of BID-Gen for the same user U, at two different times is the same. i.e: Pr[eq(x,y)=0 | x$\leftarrow$BID-Gen($BIOM_u$, PW_u), y$\leftarrow$BID-Gen($BIOM_u$, PW_u)] $\leq \epsilon_2$, for negligibly small ϵ_2.

3. *Revocable*: User U can safely replace BID1$\leftarrow$BID-Gen($BIOM_u$, $PW1_u$) with BID2$\leftarrow$BID-Gen($BIOM_u$, $PW2_u$). i.e: BID1$\neq$BID2.

We can build such a BID-Gen function using two main building blocks: a classification function (Cl-F) which is defined below and PBKDF2.

Definition IV: A classification based learning algorithm is given by two procedures: Train and Predict.

Train: Takes set of training data (Tr-Data), prepared as pairs of {class-label (cl), feature-vector (f)}, as inputs and outputs a classification function (Cl-F): {f} $\rightarrow$ {cl}. i.e: Train: {Tr-Data} $\rightarrow$ {Cl-F}.

Predict: Takes a feature-vector (f) and Cl-F as input and outputs the mapping class-label (cl) of f. i.e: Predict: {f, Cl-F} $\rightarrow$ {cl}. Predict must have two properties: low-FAR and low-FRR.

Low-FAR: Pr[eq(cl1, cl2)=1 | cl1$\leftarrow$ Cl-F(f1), cl2$\leftarrow$Cl-F(f2), f1$\not\approx$f2] $\leq \epsilon_1$, for negligibly small

ϵ_1.

Low-FRR: Pr[eq(cl1, cl2)=0 | cl1$\leftarrow$ Cl-F(f), cl2$\leftarrow$Cl-F(f'), f$\approx$f'] $\leq \epsilon_2$, for negligibly small ϵ_2. We call such a classification function, a (ϵ_1,ϵ_2) Cl-F, where ϵ_1 and ϵ_2 are false acceptance rate (FAR) and false rejection rate (FRR) respectively.

We define two functions: $PBKDF_{s1}$ and $PBKDF_{s2}$ which derives two secrets S1 and S2 used in PrivBioMTAuth scheme. $PBKDF_{s1}$(PW) = PBKDF2(PW)$|_{0-127}$ and $PBKDF_{s2}$(PW) = PBKDF2(PW)$|_{128-288}$. i.e: $PBKDF_{s1}$ and $PBKDF_{s2}$ extracts the first 128 bits and second 160 bits respectively, from the output of PBKDF2 on the input of the password and the salt value. We have ignored the salt value for brevity.

Definition V: Therefore, we can model BID-Gen as follows, using three sub functions: Cl-F, $PBKDF_{s1}$ and Concat:$\{(0,1)^{32}$, $(0,1)^{128}\} \rightarrow \{(0,1)^{160}\}$.

BID-Gen(BIOM, PW):

1. cl $\leftarrow$ Cl-F(BIOM).
2. S1 $\leftarrow$ $PBKDF_{s1}$(PW).
3. BID $\leftarrow$ Concat(cl, S1).
4. return BID.

Accordingly, Enroll procedure is built using BID-Gen, $PBKDF_{s2}$ and Pedersen Commitment, which cryptographically encodes the two secrets: BID and S2. Auth procedure is built using BID-Gen, $PBKDF_{s2}$ and ZKPK protocol.

3.4.2 Analyzing PrivBioMTAuth

In this section, we analyze the properties of PrivBioMTAuth and its building blocks defined above. First of all, low-FAR and low-FRR properties of Cl-F (see Definition IV) is empirically proven in Section 3.3.2, which implies uniqueness and repeatability properties of BID-Gen (see Definition III) respectively. Completeness property of PrivBioMTAuth (see Definition II) scheme is implied by three things: i) repeatability of BID-Gen ii) deterministic property of PBKDF2 iii) completeness property of ZKPK. We do not list the details of proof of completeness of ZKPK here and refer the reader to e.g. [89].

Proof of security of PrivBioMTAuth (see Definition II) involves showing that breaking PrivBioMTAuth scheme implies that the discrete log problem can be solved. Thus this implies (assuming that discrete log problem is hard) that a computationally bounded adversary has a negligible success probability in breaking the scheme.

Theorem: If $PBKDF_{s1}$ is a random oracle and there exists an adversary A that successfully authenticates with Auth procedure with non-negligible probability, then A contains a knowledge extractor that can solve the discrete log problem with non-negligible probability.

Proof: Assume that such an adversary A exists, and that an adversary B is given a discrete log problem instance: g, p, q, $g^x h^r$ (i.e: pedersen commitment of x and r). B is also given oracle access to Concat function, $PBKDF_{s2}$ and Cl-F trained for a random enrolling user. B sends a randomly chosen challenge e and $g^x h^r$ to A. A is expected to eventually output ZKPK of x and r against challenge e.

To simulate $PBKDF_{s1}$, B creates a set of tuples PSET. B initializes PSET by choosing a random password PW_B and adding (PW_B, $S1_B$) to PSET for randomly chosen $S1_B$ of length 128 bits. B initializes another set of tuples BSET which is initialized by choosing a random biometric feature vector $BIOM_B$, obtaining the output: $cl_B \leftarrow$ Cl-F($BIOM_B$) and adding ($BIOM_B$, cl_B) to BSET. When A queries each of Cl-F and $PBKDF_{s1}$ on a value m, B does the following. If there is a tuple (m, n) already in the corresponding tuple-set, it responds with n. Otherwise; 1) if it is PSET, it chooses a random r, adds (m, r) to PSET and responds to A with r, and 2) if it is BSET, it queries cl $\leftarrow$Cl-F(m), adds (m, cl) to BSET and responds to A with cl. When A queries Concat and $PBKDF_{s2}$ with (cl_A, $S1_A$) and PW_A respectively, B does the following. If $cl_A = cl_B$ and $S1_A = S1_B$, B outputs FAIL. Otherwise, B queries Concat and $PBKDF_{s2}$ with (cl_A, $S1_A$) and PW_A respectively, receives BID_A and $S2_A$ and sends them to A.

It is straight forward to see that if B does not output FAIL, then the above view is the same as the view when B is engaging in the protocol. In the following we show that: (i) B outputs FAIL with negligible probability and (ii) if B does not output FAIL and A succeeds with non-negligible probability, then B can use A to obtain x and r.

(i) B outputs FAIL only when A asks for Concat and $PBKDF_{s2}$ responses for inputs: (cl_A, $S1_A$) and PW_A and $cl_A = cl_B$ and $S1_A = S1_B$. In this FAIL scenario, two things can happen: (a) A queries Cl-F on $BIOM_B$ and $PBKDF_{s1}$ on PW_B (b) A does not query Cl-F on $BIOM_B$ and $PBKDF_{s1}$ on PW_B. Case (a) implies that A knows $BIOM_B$ and PW_B, which is an event with negligible probability, as A does not know B's biometrics and password. Case (b) implies that A randomly guesses cl_B and $S1_B$ which is negligible because cl is drawn from a space of size: 2^{32} and $S1$ is drawn from a space of size 2^{128}.

(ii) If B does not output FAIL and A can create a ZKPK of discrete logarithm of $g^x h^r$, then by properties of ZKPK, there must be a knowledge extractor for A that produces x and r. B uses this knowledge extractor to solve discrete log problem. If A succeeds, then so does B. However, assuming discrete log problem is hard, an adversary A does not exist.

Proof of privacy of PrivBioMTAuth (see Definition II) involves showing that no sensitive information about the biometrics or the password is leaked from: i) IDT ii) D iii) transcripts of *Enroll* and *Auth* protocols, and iv) no information about the interaction between U and SP is learned by the IDP. Case (i) directly follows from the perfectly hiding property of Pedersen Commitment scheme [77]. Case (ii) is ideally achieved by the use of Trusted Execution Environment (TEE) [90] of modern mobile devices for storing the artifacts given by the IDP to the user at the end of Enroll procedure. However, since the mobile device that we used for experiments did not allow developper access to TEE, we followed a defense-in-depth approach to secure the artifacts stored in D. First, the artifacts are stored in the IDP-client application's internal storage, which cannot be accessed by any other application, based on the application sandboxing mechanism provided by the Android kernel. Second, the artifacts are encrypted using a key stored in a keystore which also resides in the application's internal storage. Third, the keystore is secured with a secret ($S3$) which is derived from the user's password. Therefore, an attacker needs to break all the defense mechanisms in order to get hold of the artifacts and try to infer any sensitive information. We would like to emphasize that even if an attacker is able to break this defense-in-depth protection mechanism and get hold of the artifacts stored in D, the attacker can not break the security of PrivBioMTAuth scheme as biometrics and the password of the user is not stored in the device. Case (iii) directly follows from case i) that IDT in the protocol transcripts reveals nothing, from the zero-knowledge property of ZKPK scheme used in the Auth protocol such that u, v in the *Auth* protocol transcript reveals nothing and from the fact that no other sensitive information is exchanged in plaintext during the *Enroll* and *Auth* protocols. Case (iv) follows from the fact that we have adopted a user-centric architecture in the authentication protocol which avoids any user authentication attempt from going through the IDP and the assumption of a honest but curious IDP which does not collude with SPs.

Accordingly, we can conclude that PrivBioMTAuth(Enroll, Auth) scheme satisfies the three properties mentioned in Definition II, and hence it is complete, secure and privacy preserving.

4. RAHASNYM: A PSEUDONYMOUS AND FLEXIBLE IDENTITY VERIFICATION SYSTEM FOR PROTECTING AGAINST LINKABILITY

This chapter presents our privacy preserving solution, named RahasNym, for pseudonymous and flexible identity verification during the execution of online transactions, which we introduced in Section 1.4. This chapter is organized as follows. In Section 5.1 we present an overview of how a real world scenario is implemented with RahasNym.. In Section 5.2 we present our approach followed by the implementation details and performance analysis on RahasNym in Section 5.3. Section 5.4 presents the security and privacy analysis on RahasNym.

4.1 Overview of the Solution

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

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

RahasNym involves three main parties: user, SP and Identity Provider (IDP). During the initialization phase of RahasNym, both the user and the SP define their identity verification policies with fine-grained rules. When the user wants to perform a certain transaction, the user agent (i.e: the software in the user's device) obtains from the SP the identity verification policy related to that particular operation and checks if this policy matches the user policy. If they match, the user agent acquires the Identity Token(s) (IDT) required to perform the transaction from the corresponding IDP(s). For example, if CCN is required, the CCN-IDT is obtained from the user's credit card issuer and if the email address is required, the email-

IDT is obtained from the user's email provider. The IDP cryptographically encodes the user's identity information using the Pedersen Commitment scheme [77] and sends back to the user agent the digitally signed IDT. Then the user agent and the SP execute the agreed identity verification protocol through which the user proves his/her identity to the SP in zero-knowledge. If there is a conflict in combining the two policies, the user agent prompts the user to change her policy or aborts the operation. In what follows we analyze the identity

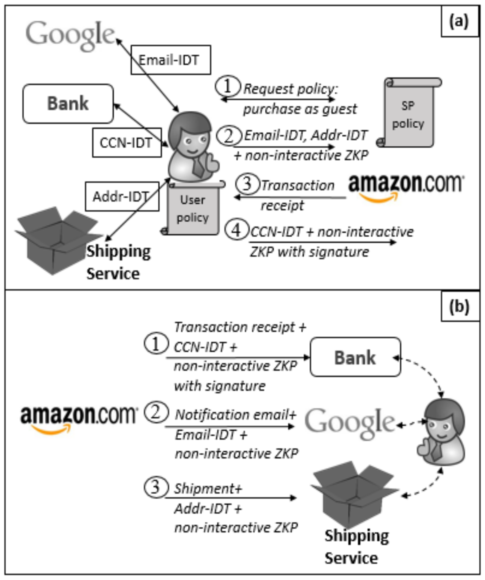

(a)
Google
Email-IDT
Bank
CCN-IDT
Addr-IDT
Shipping Service
User policy
1 Request policy: purchase as guest
SP policy
2 Email-IDT, Addr-IDT + non-interactive ZKP
3 Transaction receipt
amazon.com
4 CCN-IDT + non-interactive ZKP with signature
(b)
1 Transaction receipt + CCN-IDT + non-interactive ZKP with signature
Bank
amazon.com
2 Notification email+ Email-IDT + non-interactive ZKP
Google
3 Shipment+ Addr-IDT + non-interactive ZKP
Shipping Service

Figure 4.1. (a) Identity verification at on-line shopping scenario (b) The merchant performing activities related to the purchase with the IDPs.

verification carried out with regard to the operation, 'purchasing an item as a guest'. Assume that the user has active on-line accounts at the bank, shipping service, and email provider which act as IDPs issuing the respective identities to the user in the form of cryptographic IDTs. As shown in the step 2 of Figure 4.1(a), the first two attributes requested by the on-line merchant are email address and shipping address. However, since they are cryptographically encoded in the IDTs, there should be a way for the on-line merchant to verify the ownership of the IDTs and to use these IDTs for sending delivery notifications and shipping the items to the user. On the other hand, when the on-line merchant requests the email provider and the shipping service provider to perform the aforementioned operations by forwarding the user-provided IDTs, they should be able to link those IDTs to the user's accounts held at them, and verify that it is the actual owner who has given the IDTs to the particular merchant.

We have defined an identity verification protocol based on non-interactive ZKPK schemes to address requirements of these kinds of scenarios. After successful identity verification, the on-line merchant saves the IDTs and the associated non-interactive zero knowledge proofs provided by the user, in order to use them later to send email notifications and ship item(s) through the respective IDPs[1] as shown in steps 2 and 3 of Figure 4.1(b). Those IDPs, after verifying that the proofs of ownership have been created by the genuine owners of the IDTs, complete the operations requested by the on-line merchant. The 'return labels' feature offered by on-line shipping services such as FedEx, can easily adopt the proposed scheme to allow users to obtain the IDTs which cryptographically encode their shipping address, instead of the plain-text shipping address included in the return labels today. Here, FedEx acts as the IDP for the user's identity represented by his/her shipping address and once the packages with such IDTs are received, the shipping service can link them to the actual user's account via the auxiliary information contianed in the IDTs.

We have extended the basic non-interactive ZKPK protocol to provide more authenticity for payment related identity verifications. For an example, the purpose of requesting credit card information is to claim the payment from the user's bank account. If the bank approves the claimed payment based only on the IDT and associated proofs as in the previous case, a malicious on-line merchant could claim an arbitrary payment (with a higher amount) that is not approved by the user, simply by forwarding the IDTs and proofs provided by the user for a different transaction (associated with a lower amount). Therefore, it is important that the bank obtains a proof that the owner of the CCN-IDT has approved the current transaction, before performing the credit transfer to the merchant's account (step 1 of Figure 4.1(b)). In order to address such requirements, we have defined another protocol which requires to include the transaction receipt also in the signature created during the non-interactive ZKPK protocol carried out in order to prove the ownership of the CCN-IDT (step 4 of Figure 4.1(a)). In this way, both the on-line merchant and the bank can verify the proof and have assurance about the authenticity of the transaction while neither on-line merchant nor any intermediate entity in the credit card network can see the actual details of user's credit/debit card. Different transactions are unlinkable since each time a new IDT is used.

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

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

4.2 Our Approach

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

4.2.1 Key Requirements

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

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

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

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

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

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

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

1. Level 1: discouraging the sharing of IDTs by enforcing negative consequences of sharing such as having to share the password accociated with the credential or having to share the user's device itself whose TPM (Trusted Platform Module) contains the master secret of the anonymous credentials [91].

2. Level 2: preventing the sharing of IDTs by including the user's biometric identity in the IDT and requesting to verify it at the identity verification.

(6) Authenticity: This requirement is related to guarantee that the transaction has been approved by the legitimate owner of the IDT (e.g: CCN-IDT) which is used during the transaction. In many of today online transaction systems, any transaction performed through the logged-in session is considered valid, without any verification that the legitimate user has approved each transaction. If a malicious party steals the logged-in session, the authenticity of the transaction is at risk. Furthermore, as in the scenario discussed in Section 4.1, the user does not get to directly interact with the bank during the payment process. Hence there should be a way for all the parties involved in the transaction to verify that the legitimate user has actually approved the current transaction.

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

(8) Flexibility: Since the aforementioned required properties of pseudonymous identity verification vary based on different factors as mentioned in Section 1.4, a pseudonymous identity management framework should enable the participating parties to express their requirements and accommodate such requirements in a flexible manner.

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

4.2.2 The Policy Framework

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

Policy Language

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

Policy Elements: In our model we have three main policy elements, each of which has different options (see Table 4.1). How such options affect the protocol execution is explained in Section 4.2.3.

(1) Subject Verification: This element specifies how a pseudonymous IDT should be bound to the subject who presents the IDT and to the subject who verifies this IDT. This information is recorded in the IDT (in 'From:' and 'To:' fields respectively) and should be verified by the SP at authentication. For a particular transaction, this information can be included either in clear text or in hidden format (i.e: as a commitment). Based on the required level of assurance, this policy element can take different values as shown in Table 4.1. With the options available to specify how the 'From:' field of the IDT should be formed, one can achieve different levels of pseudonymity from full identification (real-name bound) to pseudonimity (pseudonym bound) to full anonymity (hidden pseudonym bound). Optionally, user's biometric identity is bound to require stronger level of non-shareability assurance. This policy element enforces the following subset of requirements: *ownership assurance, non-shareability, and confidentiality of pseudonyms.*

(2) Proof of Identity: This element specifies how the proof of identity should take place. RahasNym uses cryptographic commitments for encoding identity information and different variations of ZKPK-based protocols to prove the ownership of the identity. Therefore, the different values that this policy element takes correspond to the names of the three identity proof protocols in RahasNym, as listed in Table 4.1. This policy element allows one to enforce the following subset of requirements: *confidentiality of identity information, ownership assurance, and authenticity of the transaction.*

(3) Pseudonyms Cardinality: This element specifies whether the user has to use a single pseudonym or can use multiple pseudonyms when interacting with a particular SP. The two values it takes, shown in Table 4.1, indicate the level of unlinkability assured for the user. If it is *single*, the user has unlinkability assurance only at level 2 and level 3 but not at level 1. If the value is *multiple*, the user has unlinkability assurance at all three levels (see Section 4.2.1 for different levels of unlinkability).

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

Table 4.1. Options provided for the policy elements.

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

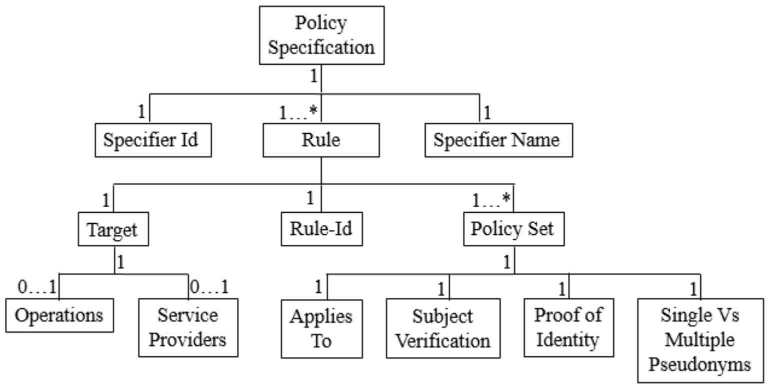
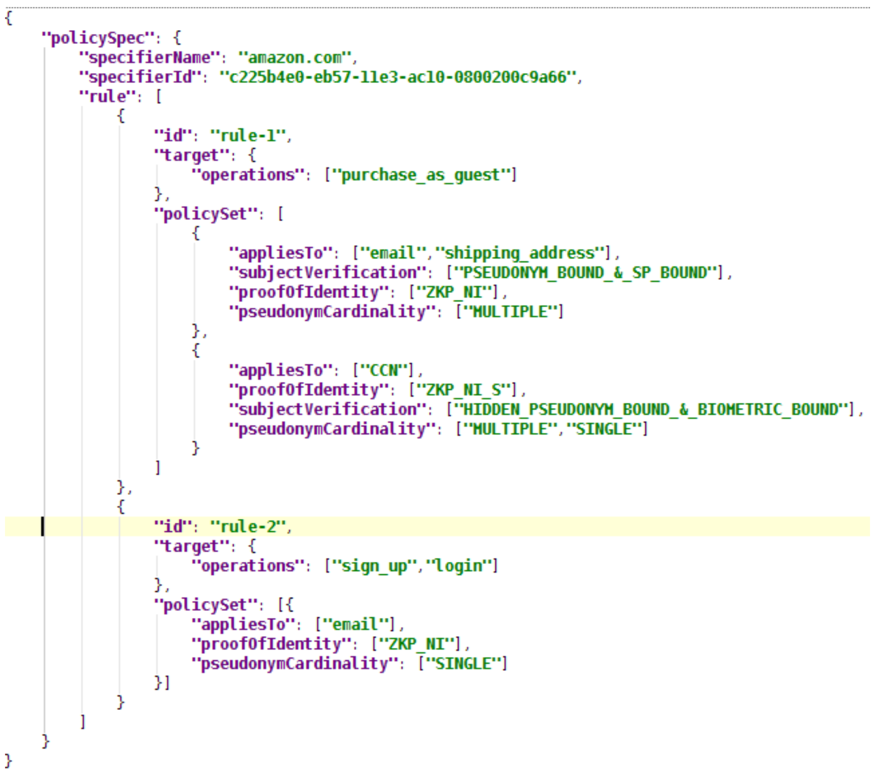


Figure 4.2. Structure of the Policy Language.

Policy Schema: Identity verification policies in RahasNym are expressed in a lightweight policy language built on top of the JSON data format. Figure 4.2 shows the hierarchical structure of the policy objects that constitute the policy language along with their relationship cardinality which defines the schema of a policy specification. 'Policy Specification' is the root object which consists of 'Specifier Id', 'Specifier Name' and one or more 'Rule' objects. One such 'Rule' object may define a fine-grained rule applied to verification of the identity attributes related to a particular transaction. Hence the 'Rule' object consists of a 'Rule Id', a 'Target' and one or more 'Policy Set' objects. The 'Target' object in a SP's policy specification defines the names of the operation(s) to which the 'Rule' is applied and that in a user's policy specification defines the names of the SP(s) to which the 'Rule' is applied. A 'Policy Set' object in a 'Rule' defines the policy elements (see Table 4.1) applied to verification of a particular identity attribute or set of identity attributes. We have provided sample policy specifications by SP and user for further illustration of the use of the policy language.

Example Policies

The policy specification of the on-line merchant (see Figure 4.3) has two rules: rule-1 applies when the user purchases an item as a guest, whereas rule-2 applies when the user signs up with the on-line merchant by creating an account and later logs into that account. Rule-1 has two policy sets each of which applies to different identity attributes. The first policy set of rule 1 applies to email and shipping addresses; it enforces non-interactive ZKPK protocol for proof of identity and accepts the IDT bound to the pseudonym of the user in the 'From' field of the token and to the on-line merchant's identity in the 'To' filed of the IDT. The second policy set of rule-1 applies to the IDTs associated with the user's credit card number. It specifies that the proof of identity should be carried out through non-interactive ZKPK with signature protocol, the IDT should be bound to the user's pseudonym in hidden format and to the biometric identity of the user, and the pseudonym cardinality can be either multiple or single. Rule-2 applies to sign-up and login operations which requires an



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

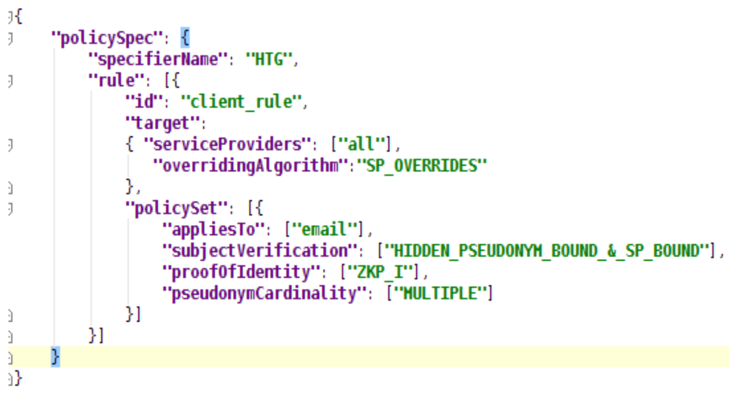
Figure 4.3. Example Policy Specification of On-line Merchant.

email address IDT under a single pseudonym since the on-line merchant needs to associate the transactions that the user performs, with the account created by the user.

The policy specification of the user (see Figure 4.4) illustrates the default policy set that is applied to one of the user's identity attributes which is email address and it is applied when carrying out transactions with all the SPs. The user can be prompted by the software in the client device to customize this default policy specification for different SPs and for different identity attributes as and when the user performs transactions. In this way, user's policy will get refined over time, based on the user's preferences and trust with different SPs.

Policy Combining Algorithm

In order to match the user's policy with the SP's policy and to obtain the agreed policy by both the parties, we provide a policy combining mechanism. Basically it goes through both the policies and compares the 'Policy Set' objects defined for the identity attributes



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

Figure 4.4. Example Policy Specification of User.

required to perform the intended transaction. In case of a conflict, the user can configure the conflict resolution mechanism as *Client Overrides* or *SP Overrides*, which is specified in the 'Target' object of a user's policy specification. If it is *SP Overrides*, the execution proceeds by overriding the user's policy with that of the SP's and if it is *Client Overrides*, the execution aborts giving user the option to change her policy to match that of the SP's if she prefers. In what follows we present two examples of how the policy combining algorithm resolves conflicts that are found during policy combining and helps negotiating preferences when multiple options are given as the policy element values.

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

If the user has specified *SP Overrides*, pertaining to the particular SP that the user is interacting with (i.e: in the *target* element defined for the particular SP), the policy combining algorithm will include the value *Single* for the policy element *Pseudonym Cardinality* in the corresponding policy set of the combined policy. Then the IDMM will request an IDT from the corresponding IDP, by sending the optional parameter: *single_pseudonym_certification_required* in the first step of Protocol 4.1, along with other details, including ID_{SP} in plain text. Then the IDP will check if the user has previously obtained any IDT to be presented for that particular SP, in which case the IDP will include in the issuing IDT, the same pseudonym used in such previous IDTs. Otherwise, the IDP will attach a new pseudonym (possibly sent by the user) to the issuing IDT and records it. The IDP will also include the certification: *single_pseudonym_certified* in the IDT created in the step 6 of Protocol 4.1. Once the IDT is received at the SP, it will check for this certification made by the IDP to make sure that the IDT adheres the SP's policy.

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

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

Negotiation of Policy Options:
When the user and/or the SP have specified more than one option for the three policy elements defined in Section 4.2.2, the combining algorithm will pick the option that enforces the highest level of assuarance w.r.t the corresponding requirement(s) of pseudonymous identity management. For example, if the *Proof of Identity* policy element in the user's policy specifies the options: *ZKP-NI, ZKP-NI-S* and the SP's policy specifies the options: *ZKP-I, ZKP-NI, ZKP-NI-S*, the policy combining algorithm will select: *ZKP-NI-S* as the option to be included in the combined policy, as it is included in both the policies and provides stronger authenticity assurance, as discussed in Section 4.2.3.

4.2.3 The Protocol Suite

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

Pedersen Commitment: We use this for cryptographically encoding the identity information. It is a secure commitment scheme [77] whose security is based on the hardness of solving discrete logarithms. The setup and the commitment creation can be described by the following steps.

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

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

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

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

(2) creating commitments of the user's pseudonyms and SP's identity in order to embed them in the IDT in hidden format,

(3) creating non-interactive zero knowledge proofs: as shown in Protocol 4.3 and Protocol 4.4.

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

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

W = PBKDF2 (PKCS#5, Password, Salt, derived key length). Salt is 8 bytes long and randomly generated during each run of PBKDF2 so that no two secrets generated from the

Table 4.2. Notation used in the protocol suite.

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

same password will be the same for a sufficiently long time, after which user can be forced to change the password.

In what follows we define the four protocols in the suite. Table 4.2 lists the notation used in the protocol definitions. First we explain the execution of the core protocols that correspond to the default options of the policy elements. Next, we discuss the variations of the core protocols based on the other options of the values taken by the three policy elements introduced in Section 4.2.2.

IDT Request Protocol

The core IDT request protocol corresponds to the basic policy options of: *Subject Verification = pseudonym-bound AND SP-bound*, *Pseudonym Cardinality = Single*. As shown in step 1 of Protocol 4.1, the user agent sends the IDT request along with the specified input. The user agent sends the secret r (derived from the user's password) encrypted with IDP's public key to be used in creating the identity commitment. The IDP creates the pseudonymous IDT by enclosing the user's identity 'a' corresponding to the requested identity attribute n, in the commitment Y (step 4 of Protocol 4.1). IDP also creates the auxiliary element W to be included in the IDT which helps the IDP to link an issued IDT with the user's account afterwards, if required. While the IDP can decide which value is included in W, it should not allow multiple IDTs of the same user being linked through W. RahasNym defines W as the user's pseudonym held at the IDP concatenated with the timestamp T and encrypted using IDP's public key. RSA-OAEP is used for encryption which provides both randomized encryption and security against chosen cipher text attacks (CCA-secure). In step 6 of Protocol 4.1, IDP vouches for the validity of the IDT by digitally signing it.

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

Protocol 4.1: IDT Request Protocol

1. $\underline{U \rightarrow IDP}$: authenticates to IDP and sends IDT request along with: p, n, $E_{IDP}(r)$, ID_{SP}.
2. $\underline{IDP}$: verifies the identity of U.
3. encodes a as $x \in Z_q$ using a cryptographic hash function.
4. creates identity commitment: $Y(x, r) = g^x h^r$ mod p.
5. creates auxiliary element: W = $E_{IDP}(f, T)$.
6. creates IDT: IDT = $f\ |ID_{SP}\ |n\ |Y\ |D\ |T\ |W\ |PPC\_Params\ |S_{IDP}(f\ |ID_{SP}\ |n\ |Y\ |D\ |T\ |W\ |PPC\_Params)\ |C_{IDP}$.
7. $\underline{IDP \rightarrow U}$: sends the IDT.

Interactive ZKPK Protocol

Protocol 4.2 lists the core Interactive ZKP Protocol which is used to verify IDTs, when *Proof of Identity* = *ZKP-I* in the agreed policy specification. In Protocol 4.2, the user agent and the SP engage in an interactive protocol in which the user proves in zero knowledge, her knowledge about the identity information hidden in the commitment of the IDT by responding to the SP's challenges in step 5. This protocol is to be used when the prover and the verifier directly interact with each other.

Protocol 4.2: Proof in Interactive Zero Knowledge

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

Non-Interactive ZKPK Protocol

Protocol 4.3 matches the policy element option: *Proof of Identity = ZKP-NI*. The difference in Protocol 4.3 compared to Protocol 4.2 is that instead of the challenge e sent by the SP in step 4 of Protocol 4.2, the user agent generates three challenges in step 3 of Protocol 4.3, based on which the non-interactive zero-knowledge proofs are created. One-wayness of the cryptographic hash function prevents the user agent from being able to predict the challenges beforehand and thereby prevents any cheating by the user during proof creation. Although we have set the number of challenges to three in the definition of Protocol 4.3, that number can be varied based on the level of difficulty that needs to be maintained against the user's ability to cheat (by using a cryptographic hash function with larger output length (e.g: SHA-1024) or by using PBKDF to obtain an output of arbitrary length). Replay attacks are avoided since the timestamp at the proof creation (T) is included in the input to the hash function. By knowing the inputs given to the hash function and the proofs created based on the output of the hash function, any party can verify the user's ownership of the identity by following the steps from 7-13 in Protocol 4.3.

Protocol 4.3: Proof in Non-Interactive Zero Knowledge

```
1. U: selects three pairs of $y_j, k_j \in Z_q$, and creates three commitments: $d_j = g^{y_j} h^{k_j}$
 mod p where j $\in \{1, 2, 3\}$.
2. Hash Output $(h) = H(Y, T, d_1, d_2, d_3)$.
3. obtains three challenge values: $e_j \in Z_q$ : j $\in \{1, 2, 3\}$, from $h$ above.
4. creates proofs based on $e_j$ values: $u_j = y_j + e_j x$ mod q and $v_j = k_j + e_j r$ mod q,
 where j $\in \{1, 2, 3\}$.
5. U→SP: sends $M$ that takes the form:
 M = $IDT$, $T$, $d_1$, $d_2$, $d_3$, $u_1$, $v_1$, $u_2$, $v_2$, $u_3$, $v_3$
6. SP: verifies $S_{IDP}$, $D$ and $T$ on the IDT and $T$ in M.
7. if initial verification is successful: then
8.   obtains $h = H(Y, T, d_1, d_2, d_3)$.
9.   obtains challenge values $e_j$ : j $\in \{1, 2, 3\}$ from $h$.
10.   for each $e_j$ do
11.     verifies if $Y^{e_j} d_j = g^{u_j} h^{v_j}$.
12.   end for
13.   if all the challenge-response verifications are passed, SP accepts $U$ as the owner
 of IDT.
14. end if
```

Non-Interactive ZKPK with Signature Protocol

Protocol 4.4 is used for identity verification when the agreed policy specifies *Proof of Identity = ZKP-NI-S*. Protocol 4.4 binds the transaction with the non-interactive proof of identity by requiring to include the transaction receipt as one of the inputs to the hash function, when creating and verifying the identity proof (step 3 and 9 of Protocol 4.4). Protocol 4.4 lists all the steps from the transaction receipt transmission by the SP to the user, up to linking the transaction with the user's account by the IDP. This protocol provides a higher level of assurance of the authenticity of the transaction (i.e: assurance that the legitimate owner of the identity has approved the transaction.), compared to the other two identity proof protocols in which the identity verification and the actual transaction happen in two different steps. This type of protocol for proof of identity is especially useful in scenarios where the IDP has to perform some critical operation on the user's account without having direct interaction with the user, such as debiting money from the user's account for a transaction claimed by an on-line merchant through the credit card network, as discussed under the Section 4.1.

Extensions to the Core Protocols

In what follows we list the extensions of the core protocols (defined in Section 4.2.3) which correspond to the different options taken by the policy elements other than the basic options.

(1) If *Subject Verification = hidden-pseudonym-bound* AND *hidden-SP-bound*, *Pseudonym Cardinality = Multiple*, the user agent sends f and ID_{SP} in hidden format by creating commitments on them using a cryptographic hash function: e.g: $H(f|s)$ and $H(ID_{SP}|s)$, in step 1 of Protocol 4.1. This protects the confidentiality of such information from the IDP and enables the user to prove such commitments only to the SP by revealing the associated secrets (s and s). In this case, the user agent sends the secrets encrypted with the SP's public key along with other information at the beginning of each identity proof protocols.

Protocol 4.4: Binding transactions with Proof of Identity

```
1. SP→U: Sends the receipt of the transaction: $I$.
2. U: selects $y_j, k_j \in Z_q$, and creates $d_j = g^{y_j} h^{k_j}$ mod p where j $\in \{1, 2, 3\}$.
3. checks the receipt $I$ and creates Hash Output $(h)$ = H($I$, $Y$, $T$, $d_1$, $d_2$, $d_3$).
4. obtains three challenge values: $e_j \in Z_q$ : j $\in \{1, 2, 3\}$, from $h$ above.
5. creates proofs based on $e_j$ values: $u_j = y_j + e_j x$ mod q and $v_j = k_j + e_j r$ mod q,
   where j $\in \{1, 2, 3\}$.
6. U→SP: sends M takes the form:
   M = $IDT$, $h$, $T$, $d_1$, $d_2$, $d_3$, $u_1$, $v_1$, $u_2$, $v_2$, $u_3$, $v_3$
7. SP: verifies $S_{IDP}$, $D$ and $T$ on the $IDT$ and $T$ in M.
8. if initial verification is successful: then
9.   runs steps 8-13 of Protocol 4.3 to verify the ZKPK made against the challenges
     obtained from the hash (h).
10.    if ZKPK verifications passed then
11.      SP→IDP: forwards I, M and W.
12.      IDP: runs the same steps: 7-9 above.
13.      if ZKPK verifications passed then
14.        - decrypts W and verifies the name/pseudonym associated with the user's
           account at IDP.
15.        - links the transaction with the user's account.
16.      end if
17.    end if
18. end if
```

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

(3) If *Subject Verification* includes *biometric-bound*, the user agent requests the IDP to include in the IDT, the user's biometric identity commitment (B) by sending the optional argument: *biometric_id_required*, in step 1 of Protocol 4.1. In RahasNym, we adopt the privacy preserving biometrics-based identity enrollment and verification mechanism proposed in **PrivBioMTAuth**. User's biometric identifier (bid) is encoded in the biometric identity commitment using the Pedersen Commitment scheme: $B(bid, r) = g^{bid}h^{r} mod p$. Please refer **PrivBioMTAuth** for details on how bid is derived from a user's biometric features. During the identity enrollment time at the IDP, user's biometrics is captured, features are extracted, bid is derived and B is created. After B is created, other sensitive information about the user's biometric identity is discarded and B is recorded by the IDP. When B is attached to the IDT, the user agent carries out the proof of ownership of the user's biometric identity as well during the verification of the IDT. This is done through the zero-knowledge biometrics-based identity verification protocol that has been proposed in **PrivBioMTAuth** which is compatible with the suite of identity proof protocols in RahasNym.

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

4.3 Implementation and Performance

In what follows we discuss the architecture of RahasNym and the performance based on its prototype implementation. The prototype implements the complete pseudonymous identity management system that simulates the on-line purchasing scenario discussed in Section 4.1. We also implemented a cryptographic library which underlies the key components of RahasNym. It implements the three versions of ZKPK protocols based on Pedersen Commitment, with Java 7 v1.7.0_25 SDK and with security parameters $p = 1024$ and $q = 160$ bits.

4.3.1 Architecture Overview

Figure 4.5 shows an high level overview of the architecture of RahasNym. The shaded areas represent the main components of RahasNym that are distributed across three main parties involved in the identity management system. The user agent (i.e: the software in the user's device) consists of two different RahasNym components: 1) The client application - that interacts with the SP and 2) The identity management module (IDMM) - that obtains IDTs from IDPs and creates proofs of identity on those IDTs. The client application that interacts with the SP can either be a web based application running in the user's web browser or a native client application installed in the user's device. The IDMM is a trusted application that runs as a background service in the user's device. The IDP and SP are server side applications.

The sequence of interactions between different entities (see Figure 4.5) is as follows: (1) The client application obtains the identity verification policy from the corresponding SP. (2) The client application hands over the SP's policy to the IDMM. (3) The IDMM compiles the policy agreed by both parties, by combining the SP's and the user's policies. It then contacts the corresponding IDP(s) to obtain the required IDT(s) which is(are) then handed over to the client application. (4),(5) The client application acts as a mediator between the SP and the IDMM during identity verification, by forwarding the challenges sent by the SP to the IDMM and forwarding the identity proofs provided by the IDMM to the SP. The intended transaction is completed after successful verification of identity. (6) The SP uses the proof(s) of identity if there are any subsequent activities to be performed with the corresponding IDP(s) related to the transaction.

4.3.2 Performance Analysis

The IDT request protocol and one of the identity proof protocols are executed during the identity verification phase of a transaction. Since the protocols involve expensive cryp-

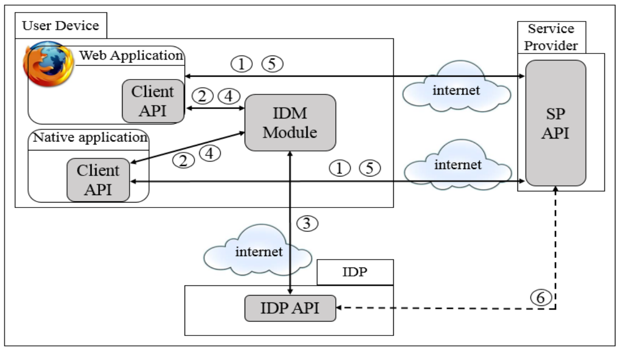


Figure 4.5. Deployment architecture of RahasNym.

tographic operations, such as modular exponentiations in creating commitments, we carried out experiments to measure the average execution time of all the major steps of the protocols executed at different entities of the identity management system as well as the average time taken for end-to-end execution of each protocol. The experiments were conducted in three machines: the client and the IDMM were deployed in a laptop machine with the Ubuntu 13.4 OS, Intel Core i7-3537U CPU, and 5 GB memory, and the RESTful web applications of IDP and SP were deployed in apache-tomcat-7.0.42 servers running in two other machines with Linux 3.15.10 based OS, Intel(R) Core(TM) i7-2600 CPU and 8 GB memory. They were connected through a wired network connection with download speed of 94.97 Mbps and upload speed of 21.41 Mbps and latency of 32ms.

IDT Request Protocol

The average execution time is: 2.945317 milli seconds (ms) which accounts for both IDT creation time and the network delay. Two significant steps in IDT creation are: 1) derivation of the secrets x and $r \in Z_q$ from the identity information and from the user's password respectively, 2) creation of the Pedersen commitment from those secrets. Average time taken by these two steps are: 1.760572 ms and 1.184745 ms respectively. Size of a basic IDT acquired from the core Protocol 4.1 is: 3.5 KB. In addition to the user identity information, the IDT includes PPC_Params which are required for proof creation by the user and proof verification by the SP, and C_{IDP} which is required for the signature verification by the SP.

Identity Proof Protocols

As shown in Table 4.3, we first measured the execution times w.r.t the set of common steps involved in all three identity proof protocols. The execution time of the helper commitment creation in Protocol 4.3 and Protocol 4.4 is three times that of Protocol 4.2, which is expected because the non-interactive zero knowledge proof creation requires three helper commitments where as interactive version needs only one such commitment to create the proof for the challenge sent by the verifier. Challenge (e) creation takes the longest execution time in

Table 4.3. Execution time and communication size of identity proof protocols.

Steps \Protocol	Protocol 4.2	Protocol 4.3	Protocol 4.4
Creating helper commitment(s): $d(y,k) {=} g^y h^k mod p$	1.24 (ms)	3.683 (ms)	3.679 (ms)
Challenge creation	0.018 (ms)	0.051 (ms)	0.696 (ms)
Proof creation	1.548 (ms)	1.532 (ms)	1.579 (ms)
Proof Verification	1.786 (ms)	5.231 (ms)	5.873 (ms)
Total execution	84.257(ms)	82.148(ms)	148.48(ms)
Comm. size	3.5 KB	3.5 KB	3.6 KB

Protocol 4.4 because it includes the transaction receipt also in the input to the hash function. For the the experiments on Protocol 4.4, we used transaction receipts of 100 KB in size. Proof creation does not show a significant difference across the three different protocols as it only involves computing values: u_{j} (= y_{j} + $\mathrm{e_j}x$ mod q) and v_{j} (= k_{j} + $\mathrm{e_j}r$ mod q). The execution times reported w.r.t proof verification reflect its increasing complexity in the three identity proof protocols. In addition to the time taken for the aforementioned key steps, the total execution times (i.e: end-to-end protocol execution times) also account for the network delay, policy request time, JSON message encoding and decoding time, policy combining time and IDT request time. Total execution times of Protocol 4.2 and Protocol 4.3 are roughly equal because although the former is less computationally expensive, it involves more communication steps than the latter, due to the interactive proof. Total execution time of Protocol 4.4 is higher than that of other two protocols as it involves binding of the transaction receipt with the ZKPK identity verification. Communication sizes involved with the three protocols are roughly equal since Protocol 4.2 involves more communication steps than other two protocols although its identity proof includes less information than other two protocols.

In summary, since the execution times of all three protocols are in the orders of milliseconds, they will not add a significant overhead to the overall latency of the transaction. According to the performance report at [92], the average total execution times of on-line transactions range from 5 seconds to 20 seconds at different on-line merchants.

4.4 Security and Privacy Analysis

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

Confidentiality: Identity information is cryptographically encoded using the Pedersen commitment [77] scheme whose security is based on the hardness of computing discrete logarithm. The Pedersen commitment
$(Y(x,r) = g^x h^r mod p)$ has two properties: unconditionally hiding - every possible value of x

is equally likely to be committed in Y, and computationally binding - one cannot open the commitment with any $x \neq x$, unless he can compute $log_g h$. Therefore, any external party, including the SP, who gets hold of an IDT or inspects a transcript of any protocol execution learns nothing about the identity values of its genuine owner. This mitigates the threat of identity theft present in online transaction systems today, in which users have to reveal the identity information in plain text to the third party SPs, even though the communication happens through the channels protected with transport level security. Using standard public key encryption schemes (such as RSA) to cryptographically encode the identity information in the IDT does not solve the problem because they do not provide means for proving the knowledge of identity information in zero knowledge during identity verification, which is essential in addressing ownership assurance requirement, which is discussed next.

Ownership Assurance: All three protocols defined for proof of identity, guarantee that any party without the knowledge of the secrets hidden in the IDT cannot successfully execute the identity proof protocols. In the interactive ZKPK protocol, the random challenge sent by the verifier prevents an attacker from crafting seemingly valid fake proofs. In the non-interactive ZKPK protocols, one-wayness of the cryptographic hash functions prevents an attacker from crafting challenges and seemingly valid fake proofs. Identity proof protocols mitigate the threat of impersonation attacks that is present in the online transaction systems today, in which the users' identity information is stored in identity repositories of various third party SPs which can be stolen by attackers as well as by malicious SPs themselves, to impersonate the user. RahasNym provides protection against the man-in-the-middle impersonation attacks, that could be carried out by malicious or compromised SPs (known as mafia attacks [93]) during the execution of interactive ZKPK protocols, by explicitly binding the pseudonym of the user and the identity of the SP to the IDT. The timestamps (T) used in the non-interactive ZKPK protocols protect against replay attacks by external parties.

Authenticity: RahasNym provides assurance on the authenticity of the transaction by binding the transaction receipt with the zero-knowledge based identity verification in Protocol 4.4. In current transaction systems, identity verification and the actual transaction happen as two separate steps allowing room for session hijacking attacks. In today's credit card transactions, there is no strong form of verification of the user's approval of the current transaction, except for requiring the expiration date, the CCV code etc, which are readily available on a stolen credit card. One might argue that it is undesirable for the IDP to see the transaction receipt in the perspective of level 3 unlinkability. However, this anyway happens in current systems and the advantage of Protocol 4.4 over these systems is that the transaction can be carried out without revealing the identity information such as credit card information to the third party SP while providing a stronger assurance on the authenticity of the transaction to all the parties. Furthermore, only the information required for the IDP to complete the transaction needs to be shown in the receipt sent to the IDP. This is an instance of a *trade-off* between authenticity and unlinkability.

Unlinkability: RahasNym supports unlinkability at level 1 and level 2 by using IDTs which cryptographically encode user's identity and which are bound to different pseudonyms (see Section 4.2.1 for different levels of unlinkability). We assume IDPs to be honest but curious parties and thereby RahasNym supports unlinkability at level 3 with the trust assumption placed on IDPs of not colluding with SPs to help them link different transactions of the same user. This is a reasonable assumption because an IDP is already trusted to vouch for the validity of the user's credential based on which the SPs authenticate the users and to protect the user's assets held at the IDPs such as user's money held at the bank, user's confidential emails held at the email providers, etc. It is important to note that we address unlinkability only at the application level. Concerns about linkability at the network level, such as through traffic analysis attacks etc., must be addressed by the use of network anonymizers such as Tor [94].

Accountability: The IDPs keep track of the issued IDTs in order to provide accountability. The same underlying mechanism used to address accountability requirement also enables SPs to complete certain user generated transactions by forwarding the IDTs and associated proofs to the IDPs, such as claiming the payment from the user's credit card, forwarding the notification emails and delivering the packages, as discussed in Section 4.1. The IDPs link the issued IDTs with the user's account using the W element included in the IDTs. This does not compromise unlinkability as the encryption mechanism used to create W is randomized and it does not compromise confidentiality of the user's pseudonym(s) held at the IDP(s) as the encryption mechanism is CCA-secure. Legal concerns related to the de-anonymization process is out of the scope of our work. Examples of such concerns are: under what circumstances a pseudonumous user is revealed and what the result of such disclosure is, etc.

Non-Shareability: RahasNym provides assurance that legitimate users are discouraged (by default) or prevented (optionally) from deliberately sharing their pseudonymous credentials with others, to exploit certain services provided by SPs (see Section 4.2.1 for different levels of non-shareability). By default, a secret (r) derived from user's password is bound to the identity commitment in the IDT which forces the user to share such secrets if he/she attempts to share the IDTs. Optionally, if the policy element *Subject-Verification* includes *biometrics bound*, user's biometric identity is encoded in an identity commitment similar to Y in Protocol 4.1, and it is attached to the pseudonymous IDT. During the identity verification carried out using such IDT, the ownership of the biometrics identity should also be proved. Since the biometrics is tightly coupled with each individual, the non-legitimate holder of a shared IDT can not use it for successful identity verification. RahasNym employs the privacy preserving biometrics based authentication protocol proposed in **PrivBioMTAuth**, to achieve non-shareability at level 2, as further described in the extensions to the core protocols.

Table 4.4 summarizes the mechanisms used in RahasNym to address the key requirements identified in Section 4.2.1 which are mainly related to user privacy and security and SP security.

Table 4.4. Summary of the mechanisms addressing the key requirements in RahasNym

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

5. CONCLUSION

In this dissertation, we develop privacy preserving protocols to address the privacy and security requirements in the current digital identity management ecosystem. Our protocols ensure that the users' online privacy is preserved while providing assurance to the SPs on the accountability of the online transactions and preserving utility of the identity management ecosystem.

First, we focus on privacy preserving and biometrics based online authentication. In order to overcome the privacy and security issues in the existing online authentication models, we introduce a user-centric authentication model for biometrics based online authentication which protects users' biometric privacy and transaction privacy. We propose two authentication protocols designed in this model, named, PEBASI and PrivBioMTAuth. Both protocols are designed as multi-factor authentication protocols s.t. even an attacker who steals the user's biometric template can not succeed in an impersonation attack. Each of the two protocols are preferred to be deployed in different contexts over the other due to their unique characteristics. In PEBASI, we enforce a potentially malicious user to follow the prescribed protocol in two different ways, in two versions of the protocol, where version 2 enforces the user to follow the prescribed protocol purely based on cryptographic primitives which provide active security. In PrivBioMTAuth, we demonstrate a methodology to securely derive a unique, repeatable and revocable biometric identifier from a user's biometric template, in order to make the usage of ZKPK practical with biometrics based authentication. We design techniques in PrivBioMTAuth to mitigate the threat of man-in-the-middle impersonation attack affecting the traditional ZKPK based identity verification. Such techniques also help to lay a foundation for securing any communication that takes place between the user and the SP following the authentication.

Second, we focus on privacy preserving exchange and verification of identity at two different stages of online transactions. Our solution named PrivIdEx enables different SPs to exchange and re-use in a privacy preserving manner, the identity assets created for a particular user, thereby eliminating the cost of repeated and lengthy due diligence processes that take place in the trust establishment stage of certain transactions. PrivIdEx is designed to be executed over a decentralized identity management ecosystem, thereby eliminating the privacy and security issues introduced by a centralized broker in the previous approaches, and providing better transparency to the participants. In our solution named RahasNym, we propose a pseudonymous and flexible identity management system which we believe is more practical than the previously proposed schemes, for protecting against linkability during the identity verification that takes place in the execution stage of online transactions. RahasNym is advantageous from the perspective of both users and SPs because both parties are able to specify, negotiate and implement their preferences w.r.t. identity verification requirements using the policy framework and the suite of protocols provided with RahasNym. Except for the initial overhead of defining the preferences for the identity verification policy elements, the whole process of obtaining IDTs and performing IDT verification is transparent to the user, as the IDMM in the user device takes care of the complexities of the cryptographic protocols. Despite the fact that the information that SPs obtain through linkability is directly linked to their revenue, it is advantageous for SPs to adopt Rahasnym because the SPs do not have to maintain and protect identity repositories of the users which are targets of the attackers.

The contributions of this dissertation can be developed further in four different directions. First, the techniques and protocols that we propose can be extended for better utility of the identity management ecosystem. For example, our PEBASI protocol can be extended by getting the user and the SPs to carry out the pre-processed authentication artifact re-fill phase by themselves, without requiring the BCA to supply such artifacts for them. Second, our protocols can be adopted in other areas of identity management which are different from the ones analyzed in this dissertation. For example, currently, in Sydney airport, biometrics based identity verification is performed by taking a biometric capture (i.e. a face image) of the passanger in the first step, scanning the passenger's passport in the second step and then matching the two images. Such a process compromises the passenger's privacy because the airport authority can store the captured biometric images of the passengers and more identity information than necessary is disclosed due to scanning of the passport. Implementation of privacy preserving biometrics based passenger identification at the airports can be explored by adopting PEBASI in conjunction with e-passports. Third, our privacy preserving protocols can be explored to be used in the emerging identity management paradigm called self-sovereign identity, which focuses on giving the total control of the user's identity to the user herself, in order to preserve the user's privacy. Fourth, certain protocols that we present, can be generalized to be used in other domains outside of digital identity management. For example, PrivIdEx can be generalized for privacy preserving exchange of other confidential digital assets, such as healthcare records of a patient exchanged between different healthcare providers to eliminate repeated medical tests.

REFERENCES

[1] T. Bernard *et al.*, *Equifax says cyberattack may have affected 143 million in the US*, Accessed: 16-Aug-2019, 2017. [Online]. Available: https://nyti.ms/2E5F6Kf.

[2] S. Kelly, *EBay's Massive Security Breach: What It Means for You*, Accessed: 16-Aug-2019, 2014. [Online]. Available: http://mashable.com/2014/05/21/ebay-breach-ramifications/.

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

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

[5] J. Condliffe, *Mark Zuckerberg says fixing Facebook's data scandal will "significantly impact" its profitability*, Accessed: 16-Aug-2019, 2018. [Online]. Available: https://tinyurl.com/ycgncfto.

[6] R. Metz, *Mark Zuckerberg explains Facebook user privacy to Congress*, Accessed: 16-Aug-2019, 2018. [Online]. Available: https://tinyurl.com/y8n8v6tq.

[7] E. Union, *General Data Protection Regulation*, Accessed: 22-Sept-2018, 2016. [Online]. Available: https://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=CELEX:32016R0679.

[8] daon.com, *IdentityX Platform*, Accessed: 14-Nov-2017. [Online]. Available: https://www.daon.com/products/identityx-platform.

[9] U. S. P. Office, *Fccx briefing*, Accessed: 22-Sept-2018, 2014. [Online]. Available:

.

[10] GOV.UK, *Introducing GOV.UK Verify*, Accessed: 22-Sept-2018, 2018. [Online]. Available: https://tinyurl.com/wu239nfk.

[11] L. T. Brandao, N. Christin, G. Danezis, and Anonymous., "Toward mending two nation-scale brokered identification systems," in *Proceedings on Privacy Enhancing Technologies*, 2015.

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

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

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

[15] A. Greenberg, *OPM now admits 5.6m feds' fingerprints were stolen by hackers*, Accessed: 10-Aug-2019, 2015. [Online]. Available: https://bit.ly/2P8O0wU.

[16] N. Cappella, *HSBC announces biometric banking with voice and fingerprints*, Accessed: 10-Jan-2017, 2016. [Online]. Available: https://thestack.com/world/2016/02/19/hsbc-voice-biometric-online-banking/.

[17] A. MacGregor, *Security in rich internet applications*, Accessed: 12-Jan-2017, 2016. [Online]. Available: https://thestack.com/iot/2016/02/22/mastercard-rolls-out-selfie-verification-for-mobile-payments/.

[18] A. MacGregor, *Amazon wants to replace passwords with selfies and videos*, Accessed: 15-Jan-2017, 2016. [Online]. Available: https://thestack.com/security/2016/03/15/amazon-wants-to-replace-passwords-with-selfies-and-videos/.

[19] OpenID, *Welcome to OpenID Connect*, Accessed: 22-Aug-2019, Jan. 2017. [Online]. Available: http://openid.net/connect/.

[20] OAuth, *Oauth 2.0*, Accessed: 22-Aug-2019, 2017. [Online]. Available: https://oauth.net/2/.

[21] H. Gunasinghe, M. Atallah, and E. Bertino, "Pebasi: A privacy preserving, efficient biometric authentication scheme based on irises.," in *In Submission*, 2021.

[22] H. Gunasinghe and E. Bertino, "Privbiomtauth: Privacy preserving biometrics-based and user centric protocol for user authentication from mobile phones.," in *IEEE TIFS, vol. 13, pp. 1042-1057*, 2018.

[23] A. Perala, *Iris ID Tech Helps Streamline Passenger Screening at Qatar Airport*, Accessed: 16-Aug-2019, 2017. [Online]. Available: https://bit.ly/2E5Nae3.

[24] J. Katz and Coursera, *The pseudo one-time pad*, Accessed: 01-Dec-2019, 2019. [Online]. Available: https://bit.ly/2E7D1xw.

[25] S. Wolf and J. Wullschleger, "Oblivious transfer is symmetric.," in *EUROCRYPT*, 2006.

[26] Z. Erkin *et al.*, "Privacy-preserving face recognition.," in *PETS'09*.

[27] A. Sadeghi *et al.*, "Efficient privacy-preserving face recognition," in *ICISC*, 2009.

[28] M. Blanton and P. Gasti., "Secure and Efficient Protocols for Iris and Fingerprint Identification.," in *ESORICS*, 2011.

[29] Y. Huang *et al.*, "Efficient privacy-preserving biometric identification.," in *NDSS*, 2011.

[30] Y. Luo *et al.*, "An efficient protocol for private iris-code matching by means of garbled circuits.," in *IEEE ICIP*, 2012.

[31] D. Demmler *et al.*, "ABY - a Framework for Efficient Mixed-Protocol Secure Two-Party Computation.," in *NDSS*, 2015.

[32] D. Beaver, "Efficient multiparty protocols using circuit randomization.," in *CRYPTO'91*, 1991.

[33] I. Damgard *et al.*, "
Multiparty computation from somewhat homomorphic encryption," in *CRYPTO*, 2012.

[34] N. Döttling *et al.*, "TinyOLE: Efficient actively secure two-party computation from oblivious linear function evaluation," in *CCS*, 2017.

[35] G. Droandi *et al.*, "Semba: SEcure Multi-biometric Authentication..," in *CoRR abs/1803.10758*, 2018.

[36] U. Feige and A. S. A. Fiat, "Zero-knowledge proofs of identity," *Journal of Cryptology*, 1988.

[37] A. Fiat and A. Shamir, "How to prove yourself: Practical solutions to identification and signature problems," in *CRYPTO 1986*, May 2015.

[38] F. Paci, "Veryidx - a digital identity management system for pervasive computing environments," in *Proceedings of 6th IFIP*, 2008.

[39] Y. Desmedt, C. Goutier, and S. Bengio, "Special uses and abuses of the fiat-shamir passport protocol," in *CRYPTO '87*, Aug. 1987.

[40] D. Birch, *Putting identity on the blockchain.* 2016. [Online]. Available: http://www.chyp.com/putting-identity-on-the-blockchain-part-1-find-a-problem/.

[41] H. Gunasinghe, A. Kundu, E. Bertino, H. Krawczyk, S. Chari, K. Singh, and D. Song., "Prividex: Privacy preserving and secure exchange of digital identity assets.," in *The World Wide Web Conference (WWW)*, 2019.

[42] H. Gunasinghe and E. Bertino, "Rahasnym: Pseudonymous identity management system for protecting against linkability.," in *The 2nd IEEE International Conference on Collaboration and Internet Computing (CIC)*, 2016.

[43] H. Gunasinghe and E. Bertino, "Rahasnym: Protecting against linkability in the digital identity ecosystem.," in *The 35th IEEE International Conference on Distributed Computing Systems (ICDCS)*, 2015.

[44] Nat, *Is Expressing Levels Enough for LOA2+?* Accessed: 22-Sept-2018, 2010. [Online]. Available: https://nat.sakimura.org/2010/09/03/is-expressing-levels-enough-for-loa2/.

[45] T. Reuters, *Thomson reuters 2016 know your customer surveys reveal escalating costs and complexity*, Accessed: 22-Sept-2018, 2016. [Online]. Available: https://www.thomsonreuters.com/en/press-releases/2016/may/thomson-reuters-2016-know-your-customer-surveys.html.

[46] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system.," 2008.

[47] E. Ben-Sasson, A. Chiesa, C. Garman, M. Green, I. Miers, E. Tromer, and M. Virza, "Zerocash: Decentralized anonymous payments from bitcoin," in *IEEE Symposium on Security and Privacy*, 2014.

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

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

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

[51] C. Paquin, *Privacy and accountability in identity systems: The best of both worlds.* [Online]. Available: http://research.microsoft.com/pubs/200815/Privacy%20and%20accountability%20-%20best%20of%20both%20worlds.pdf.

[52] J. Daugman., "How iris recognition works.," in *IEEE Transactions on Circuits and Systems for Video Technology*, 2004.

[53] L. Wood, *How it works: Iris scanning improves smartphone security*, Accessed: 16-Aug-2019, 2016. [Online]. Available: https://bit.ly/38swz24.

[54] R. Mayrhofer *et al.*, *The Android Platform Security Model.* Accessed: 10-Dec-2019. [Online]. Available: https://arxiv.org/abs/1904.05572.

[55] R. Cramer *et al.*, "SPDZ2k: Efficient Mpc mod $2k$ for Dishonest Majority," in *Crypto*, 2018.

[56] Y. Lindell, "Fast cut-and-choose based protocols for malicious and covert adversaries," in *CRYPTO '2013*, 2013.

[57] Y. Huang, J. Katz, and D. Evans, "Efficient secure two-party computation using symmetric cut-and-choose," in *CRYPTO '2013*, 2013.

[58] X. Wang, S. Ranellucci, and J. Katz, "Authenticated garbling and efficient maliciously secure two-party computation," in *Computer and Communications Security, CCS'17*, Nov. 2017.

[59] G. E. Suh and S. Devadas, "Physical unclonable functions for device authentication and secret key generation," in *44th Design Automation Conference. DAC'07*, ACM/IEEE, 2007.

[60] Y. Huang *et al.*, "Faster secure two-party computation using garbled circuits," in *20th USENIX Security Symposium*, 2011.

[61] Y. Huang, *FastGC*, Accessed: 16-Aug-2019, 2011. [Online]. Available: https://bit.ly/34golHa.

[62] G. Asharov *et al.*, "More efficient oblivious transfer and extensions for faster secure computation.," in *CCS*, ACM, 2013.

[63] T. Chou and C. Orlandi, "The simplest protocol for oblivious transfer," in *LATINCRYPT*, 2015.

[64] M. Bellare *et al.*, "Efficient garbling from a fixed-key blockcipher," in *34th IEEE SP*, 2013.

[65] S. Zahur and D. Evans, "Obliv-c: A language for extensible data-oblivious computation," in *Cryptology ePrint Archive: Report 2015:1153 [PDF]*, Nov. 2015.

[66] Y. Lindell and B. Pinkas, "An Efficient Protocol for Secure Two-Party Computation in the Presence of Malicious Adversaries.," in *EUROCRYPT*, 2007.

[67] T. Veugen *et al.*, "A framework for secure computations with two non-colluding servers and multiple clients, applied to recommendations," in *IEEE TIFS*, vol. 10, Mar. 2015.

[68] T. Nishide and K.Ohta, "Multiparty computation for interval, equality, and comparison without bit-decomposition protocol," in *PKC*, 2007.

[69] M. Bellare *et al.*, *Foundations of Garbled Circuits*, Accessed: 16-Aug-2019, 2012. [Online]. Available: https://tinyurl.com/8hna2ad2.

[70] M. Fersch *et al.*, "On the provable security of (EC)DSA signatures.," in *CCS*, 2016.

[71] C. Hazay and Y. Lindell, "Efficient secure two-party protocols - techniques and constructions," in *Springer*, 2010.

[72] A. Block *et al.*, "Secure computation with constant communication overhead using multiplication embeddings," in *INDOCRYPT*, 2018.

[73] A. C. Yao, "Protocols for secure computation.," in *IEEE FOCS*, 1982.

[74] Y. Ishai *et al.*, "Extending oblivious transfers efficiently.," in *CRYPTO*, 2003.

[75] M. Turk and A. Pentland, "Eigen faces for recognition," in *Journal of Cognitive Neuroscience*, vol. 3, Aug. 1991.

[76] A. Bazen *et al.*, "A correlation-based fingerprint verification system.," in *ProRISC Workshop on CSSP*, 2000.

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

[78] wihoho, *Implement face recognition using PCA, LDA and LPP*, Accessed: 10-Nov-2016, 2016. [Online]. Available: https://github.com/wihoho/FaceRecognition.

[79] H. Gunasinghe and E. Bertino, "Technical Report: PrivBioMTAuth: Privacy preserving biometrics-based and user centric protocol for user authentication from mobile phones," CERIAS, Tech. Rep. CERIAS TR 2017-4, Nov. 2017. [Online]. Available: .

[80] K. Kostiainen, J. Ekberg, *et al.*, "On-board credentials with open provisioning," in *Proceedings of ASIACCS'09*, 2009.

[81] J. Solis and G. Tsudik., "Simple and flexible revocation checking with privacy.," in *In Proceedings of the 6th international conference on Privacy Enhancing Technologies, PET'06.*, 2006.

[82] M. Narasimha, J. Solis, and G. Tsudik., "Privacy-preserving revocation checking.," in *Int. J. Inf. Secur.*, 2009.

[83] R. Peeters and A. Pashalidis., "Privacy-friendly checking of remote token blacklists.," in *3rd Policies and Research in Identity Management (IDMAN)*, 2013.

[84] E. D. Cristofaro and G. Tsudik., "Practical private set intersection protocols with linear complexity.," in *In 14th Financial Cryptography and Data Security, FC 2010)*, 2010.

[85] GSMARENA, *Motorola moto g - full phone specifications*, Accessed: 14-Nov-2017. [Online]. Available: .

[86] C.-C. Chang and C.-J. Lin, "LIBSVM: A library for support vector machines," *ACM Transactions on Intelligent Systems and Technology*, vol. 2, 27:1–27:27, 3 2011, Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.

[87] A. Jain, B. Klare, *et al.*, "Guidelines for best practices in biometric research," in *International Conference on Biometrics*, May 2015.

[88] A. Senior and N. Ratha, *Biometrics short course.* [Online]. Available: http://www.research.ibm.com/people/a/aws/documents/CVPR-BiometricsShortCourse-Part2.pdf.

[89] D. Chaum, J. Evertse, and J. Graaf, "An improved protocol for demonstrating possession of discrete logarithms and some generalizations.," in *EUROCRYPT*, 1987.

[90] N. Asokan, J. Ekberg, and K. Kostiainen, "The untapped potential of trusted execution environments on mobile devices," in *Proceedings of Financial Cryptography and Data Security*, Apr. 2013, pp. 293–294.

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

[92] E.-C. Times, *Online retail transaction performance indices.* [Online]. Available: http://www.ecommercetimes.com/web-performance/.

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

[94] Tor, *Tor project*, Accessed: 22-Aug-2019, 2015. [Online]. Available: https://www.torproject.org/.

VITA

Hasini Gunasinghe is a Ph.D. candidate in the Department of Computer Science at Purdue University. Her research focuses on the design and development of privacy preserving digital identity management protocols for real world use cases. Hasini's research has been supported by a Purdue Bisland Dissertation Fellowship, the Emil Stefanov Memorial Fellowship, an IBM PhD Fellowship, and a Summer Research Grant awarded by Purdue Graduate School.

Hasini's contributions as a Graduate Teaching Assistant have been recognized by the Raymond Boyce Graduate Teacher award by Purdue Computer Science Department. Hasini has interned with IBM Research and Salesforce.com. Prior to joining Purdue grad school, Hasini has been a Software Engineer for two years with WSO2 Inc., where she contributed to the open source WSO2 Identity Server product and where her contributions have been recognized by the Outstanding Contributors' awards by the founders. Hasini obtained her bachelor's degree in Computer Science and Engineering from University of Moratuwa.

Hasini grew up in the beautiful island of Sri Lanka. She enjoys meditation, swimming and learning to play piano.