PIMA: a privacy-preserving identity management system based on an unlinkable MAlleable signature

Souha Masmoudi, Maryline Laurent, Nesrine Kaaniche

PIMA: A Privacy-Preserving Identity Management System based on an Unlinkable MAlleable Signature

Souha Masmoudi, Maryline Laurent, Nesrine Kaaniche
SAMOVAR, Télécom SudParis, Institut Polytechnique de Paris, France

Abstract

This paper presents a privacy-preserving identity management system, referred to as Pima. The proposed system is built over a novel unlinkable malleable signature scheme, called $\mathcal{UMS}$. Pima supports pseudonymity, as it is more in line with today's web service interactions than anonymity. The originality of the approach is manifold. First, Pima satisfies both users' and providers' basic security and privacy needs as it provides a user centric approach which permits the users to keep control over their revealed attributes, and the service providers to get attributes certified by an identity provider and associated with different pseudonymous sessions. Second, Pima helps service providers to comply with the data minimization principle required by the E.U. General Data Protection Regulation, through the enforcement of both sanitizable and redactable features. Third, the proposed signature scheme $\mathcal{UMS}$ fulfills main security and strong privacy requirements at a constant pairing computation overhead. Fourth, Pima's concrete construction is proven as secure under the generic group model. Fifth, the implementation results demonstrate high efficiency w.r.t. most closely related work, while considering resource-constrained devices, i.e., Android-10 smartphone.

Keywords: Sanitizable signature, Redactable signature, Unlinkability, Invisibility, Privacy-preserving identity management system

1. Introduction

With the appearance of the Covid-19 crisis and the urgent need to rely on digital identity solutions to access remote services. Different identity models have been deployed to help users manage their relationships with service providers, namely the centralized and federated models. While the centralized model is mainly deployed in a corporate context, the federated model can be implemented in different settings (e.g., OPEN ID, OAuth, Fast Identity Online (FIDO), Single-Sign-On (SSO)). Relying on these identity management solutions, users' activities are easily traced and profiled by a single identity provider or across multiple service providers. Users are also compelled to disclose more information than necessary. This raises several privacy challenges, which are amplified by a growing awareness among users who are more and more eager to reduce the amount of revealed data. These challenges include (i) the empowerment of users with the full control over their identities and personal data (i.e., adopting a user-centric identity model), (ii) the protection of users' privacy and their personal data from collection's abuses and (iii) the introduction of a trusted digital identity (i.e., delivered by a trustworthy entity) that can be used to access multiple online services in a pseudonymous manner. Indeed, there are many situations where a controlled linkability of pseudonyms is desirable. For instance, in a chat-bot use case scenario, web services prefer to maintain state information per user in order to keep a conversation thread with the same person that they started it with [1].

In this context, privacy-preserving identity management systems are considered as promising solutions. Several systems have been designed to support users' anonymity relying on several cryptographic primitives, in particular digital signatures and proof algorithms. Four main signatures' families have been widely considered by the scientific community and different industrial actors. For instance, IBM proposed the Identity Mixer prototype [2] that relies on a variant of group signatures, referred to as Camenish-Lysyanskaya [3]. Microsoft introduced the UProve system [4] that relies on Brands signatures [5], following

the main design of blind signatures. Several research prototypes have been built over either attribute-based primitives [6] or sanitizable signatures [7]. To satisfy identity management privacy requirements, namely the selective disclosure property, group and blind signatures have to be extended with other cryptographic blocks, such as zero-knowledge proofs and accumulators, which overburden the communication overhead, mainly for resource-constrained devices. For attribute-based and sanitizable signatures, this extension is not ineluctably required. Both primitives support malleability properties, by essence, namely the predicate-based and sanitizing features, respectively. They can be deployed as the main building block while supporting the major privacy requirements of privacy-preserving identity management systems.

Like [2], attribute-based primitives [8, 9, 10] have been extended to achieve controlled linkability relying on pseudonyms. While of interest, they do not support the modification of the attributes' values, the user is only allowed to present attributes as they were initially certified by the issuing entity. From this perspective, unlike group, blind and attribute-based signatures, sanitizable signatures are promising candidates of interest for privacy-preserving identity management systems. Indeed, sanitizable signatures support strong malleability features allowing a sanitizer to perform modifications on admissible blocks. They have been implemented in identity management systems, for the first time, by Canard and Lescuyer [7]. They allow users to modify, in a controlled way, their certified attributes' values while certificates remain valid. Nevertheless, in existing sanitizable signature-based identity management solutions, attributes' names remain accessible to service providers while only attributes' values, that users do not want to disclose, are hidden (e.g., by special characters). As such, service providers are given the ability to distinguish modified attributes and to decide whether a message has been modified. Unfortunately, these solutions do not match our needs to disclose only the necessary attributes and attributes' values and also to achieve a controlled level of linkability. Moreover, further security and privacy properties need to be addressed when empowering users with the full control over their identities and attributes, as defined below:

- **Controlled and restricted modifications** – users should not be allowed to neither perform arbitrary modifications nor permutations on attributes' values in the same message.

- **Privacy** – when having access to a modified signature, no party is able to determine whether the message was modified or not, which and how many parts have been modified. As a result, profiling possibilities are mitigated[1].

To meet all the above challenges, this paper proposes a user centric identity management system called PIMA, based on a novel unlinkable malleable signature ($\mathcal{UMS}$) that satisfies both sanitizable and redactable schemes' features. With the combination of both primitives, we aim to enable a designated sanitizer to remove admissible parts of a message in a controlled way (i.e., he is not able to modify as he wants admissible blocks), while his public key is required for verification. Indeed, we propose that each attribute may have several values. The sanitizer is able to choose one of the values of the attribute to be disclosed and remove the others (i.e, attributes and attributes' values).

In the perspective of designing a malleable signature as a main single building block of PIMA, the signer's role is assigned to the identity provider (IdP) and the sanitizer's role to the user (U). U receives from IdP a signed message certifying his identity (i.e., attributes), associated with a pseudonym. To authenticate with a service provider (SP), for a particular session, U is able to generate a randomized pseudonym and modify his attributes w.r.t. admissible modifications and/or remove the unnecessary ones. SP can then verify the received information, relying on the public parameters of the identity provider, considered as trusted.

Let us illustrate with a company providing an employee E with a pseudonym

[1]If modified blocks are known, a service provider relying on several sessions initialized by the same user, is able to combine several modified versions of the same original message and to retrieve the accurate profile.

'DAN' and a signature σ on his attributes constituting the message $m = \{a_1 =$ 'Name: Dan Joe', $a_2 =$ 'DoB:18 May 1986', $a_3 =$ 'DoB:May 1986', $a_4 =$ 'DoB:1986', $a_5 =$ 'City:Nice', $a_6 =$ 'City:In France', $a_7 =$ 'City: In Europe', $a_8 =$ 'Job:Engineer', $a_9 =$ 'Job:Developer', $a_{10} =$ 'Job: Scientist'$\}$. E needs access to a development library for which the service provider asks him to prove that he is a developer with his location (for statistical purposes). E first derives from DAN a specific pseudonym for next interacting with the service provider. Second, he removes the blocks of the message (i.e., attributes) he does not want to disclose, resulting in the new following message $m' = \{a_1 =$'City:in France', $a_2 =$'Job:Developer'$\}$. He also modifies σ to fit m'. The service provider checks the validity of the signature under the company's keys and verifies the eligibility of E to the requested service.

Contribution – PIMA satisfies several properties of interest. First, the user is able to control his identity and the attributes disclosed to service providers, thanks to sanitization and redaction features. These modifications can only be performed by a designated user over the signature received from an identity provider. Second, from a service provider perspective, PIMA supports the legislation requirements, i.e., data minimization, while querying only necessary information to access services. Third, users may rely on self-generated pseudonyms for personalizing interactions with service providers, thus they remain under pseudonymity and unlinkable across providers. Fourth, the proposed malleable signature scheme supports strong privacy and unlinkability properties at a constant pairing computational cost. A complete implemented prototype of PIMA shows its practical usability for identity management systems adapted to resource-constrained devices. PIMA proposes a concrete construction relying on a modified variant of the Pointcheval-Sanders signature (PS) scheme [11].

Paper organization – Section 2 gives the motivation for the PIMA system through three practical and topical illustrative scenarios.Section 3 discusses the related work and Section 4 presents an overview of the proposed unlinkable

malleable signature $\mathcal{UMS}$. Section 5 gives a high-level description of Pima and presents the identified threat models. Section 6 details the concrete construction and Section 7 gives a formal security analysis w.r.t. the proposed threat models. Section 8 demonstrates, over both laptop and smartphone devices, high level performance measurements of Pima compared to related work, before concluding in Section 9.

2. Motivation Through Topical Illustrative Scenarios

To illustrate the privacy properties and functional requirements to be supported by Pima, let us consider the three following practical use case scenarios.

The first one is a company scenario where employees are eligible to remotely access multiple services offered by several service providers. For this purpose, the company X (acting as an identity provider) provides each employee (i.e., referred to as a user) with a certified identity over his attributes (e.g. name, age, phone number, position/roles in the company, fields of expertise, information about his laptop). To keep his information protected, the company X is accustomed to updating the antivirus installed on the employee's devices when being connected to its network. Now that teleworking is the norm, the company encourages their employees to use their identity to authenticate to the antivirus provider (acting as a service provider) and to benefit from the recent update. The employee usually finds himself obliged to reveal personally identifiable information, which can be sold to third parties, thus exposing him to profiling or unwanted job advertising, while he only needs to prove that he is using a device owned by the company X. Using the Pima solution, the employee, previously provided with an electronic document certified by the company X, is able to modify the document for only showing the attribute, i.e., his laptop is owned by the company X, and proving so thanks to the still valid certificate.

The second one is a healthcare application where a health organization (i.e., identity provider) provides their patients (i.e., users) with certified credentials including sensitive health information (e.g. X-rays, treatments, pathologies)

known as Electronic Healthcare Records (EHR), and personally identifiable data (e.g. name, home address, date of birth, and Social Security Number). Patients share these credentials with medical applications (i.e., service providers) to get relevant diagnosis and recommendations through their mobile devices without visiting a doctor. Although, patients have gained several advantages, e.g., saving time through these applications, their medical data have been exposed to increasing security and privacy risks. On the one hand, service providers are collecting sensitive data that can be sold to third parties, which harms patients' privacy. On the other hand, the huge amount of patients' data collected are exposed to high risk of leakage. These risks can be mitigated using the novel PIMA system. Indeed, patients are given the full control over their sensitive medical data and they can select information to be disclosed when accessing the medical applications. For example, for a diabetes application, the patient can only reveal, in a pseudonymous session, the results of the diabetic balance sheet and the age group to which he belongs. Hence, the patient can get relevant diagnosis while preserving his privacy. In case of medical data breaches, he cannot be identified and his requests cannot be linked together across several medical applications.

The last illustrative scenario is about online purchase. Let us consider a student (i.e., user) who wants to benefit from the student fare when buying a transport ticket/pass online. The transport agency (i.e., service provider) may ask him to provide an enrollment certificate or a student card delivered by his university (i.e., identity provider) to ensure that he is a student under the age of 25. It is known that the enrollment certificate contains more personal information than needed, e.g. nationality, studies' level, specialty. Such information can be used for profiling and tracking. To tackle this situation, PIMA system assumes that the student is given an enrollment certificate signed by his university, and he is able to modify his enrollment certificate to only show the transport agency that he belongs to that university and that he is younger than 25 without revealing his date of birth. As such, the student remains anonymous inside the group of students belonging to the same university and being younger

than 25 years old.

3. Related Work

This section introduces malleable signatures, discusses privacy-preserving identity management systems based on sanitization features, and presents a detailed comparison between PIMA, the proposed signature primitive and related work.

Malleable Signatures – A signature is called malleable if it is possible to derive, from a signature σ on a message m, a signature σ' on a message m' where m' is an admissible transformation of m [12]. A variety of primitives of malleable signatures have been developed, namely sanitizable and redactable signatures.

Sanitizable signatures have been introduced by Ateniese et al. [13] allowing a designated party, called the sanitizer, to change parts, chosen by the signer, of a signed message while the corresponding signature remains valid under the signer's key. Several works have been proposed to improve this concept and the variety of supported security properties. For example, Brzuska et al. [14] propose a threat formalization for the security requirements introduced in [13], namely unforgeability, immutability, privacy, transparency and accountability. In [15], [16], [17] and [18], authors extend the security properties of sanitizable signatures to cover the unlinkability property. However, in the schemes proposed by [16] and [18], several sanitized signatures, generated from the same signature, can be linked to each other and even to the original signature, since the signature of the fixed part of the message (i.e., the part that cannot be modified by the sanitizer) is unchangeable. In [19], Bultel et al. show that the unlinkability and invisibility properties can be achieved simultaneously. In [20], Canard et al. propose an extension to sanitizable signatures supporting multi signers and sanitizers.

In order to avoid the sanitizer selection at the signature generation phase, authors in [21] propose a policy-based sanitizable signature scheme that allows a

sanitizer to modify a signature based on a set of attributes it has. The proposed signature scheme does not support unlinkability. [22] presents an extension of sanitizable signature that allows to fix both the admissible blocks and the number of admissible blocks that can be sanitized in a single sanitization.

Redactable signatures have been introduced by Steinfeld et al. [23] allowing any party to remove parts of a signed message while maintaining the corresponding signature valid under the signer's key. In [24], Brzuska et al. formalize the security properties supported by redactable signatures that include unforgeability, privacy and transparency. [25] and [26] introduce redactable signature schemes fulfilling the unlinkability property. Accountability is also considered, in [27], by integrating the `Judge` functions of sanitizable signature. Relying only on redactable signatures is not sufficient to design identity management systems. Indeed, identity providers needs to design the party that modifies a signed message and to restrict the modifications.

Identity Management Systems and Sanitization – An identity management system, IMS for short, can be defined as a framework used in computer systems for implementing users' digital identities enrolment, authentication and authorization management, access control to applications and resources, and measures to achieve security and privacy of identities [28]. There are several works designing privacy-preserving IMS for disclosing only the necessary information (i.e., users' attributes) and hiding the others. For example, in [29], Sherman et al. propose a privacy-preserving IMS that relies on two group signatures (Water's signatures and Groth-Sahai proof system) as a technique to hide the users' attributes. The main idea behind this work is that a user registers only once with the identity provider and obtains a source certificate that he locally stores. Then, upon requesting a service to a service provider, he randomizes and sanitizes the certificate. The service provider is able to authenticate the user by checking the validity of the certificate. The problem is that the disclosure of attributes with some real values being revealed and shared between transactions can prevent transactions to remain unlinkable. That is why, later on, sanitizable signatures have been introduced, for the first time, by Canard

and Lescuyer [7] to design an original anonymous credential system. In their construction, authors use the signature of knowledge for avoiding the signature validity verification to require the sanitizer's key, thus ensuring the anonymity of the sanitizer. However, [7] introduces a tracing algorithm to re-identify the user as a sanitizer of a particular signature, thus, unlinkability is limited to trace-restricted unlinkability. Additionally, Canard and Lescuyer suggest that only the attributes' values are hidden with a symbol (like "#") covering the full length of the values, while the attributes' names remain visible. Thus, based on the names of attributes, their numbers, the length of their hidden values and the values of disclosed attributes during different sessions, transactions can be linkable between different service providers, and beyond that, service providers can try to extract some user's features. Service providers are also able to distinguish sanitized signatures from original ones based on the message form, which contradicts the privacy requirement. Sanitizable features were also used to support privacy properties in smart mobile medical scenarios [30]. In their scheme, Xu et al. consider sanitizers as a honest party, which makes the scheme weakly unforgeable. Also, unlinkability and immutability are not supported in [30]. Later, in [31], authors proposes a redactable signature scheme that allows users to remove sensitive parts of their healthcare data when sharing them with third parties. The proposed scheme does not support neither unlinkability nor anonymity or pseudonymity. [31] only ensures that third parties are not able to distinguish a redacted signature from an original one.

Comparison Between PIMA and Closely Related Work – Table 1 presents a comparison between the proposed PIMA system and the $\mathcal{UMS}$ signature scheme with closely-related works. The first part of Table 1 identifies the security properties achieved by PIMA vs other identity management systems relying on sanitization and redaction, namely **unforgeability** of signatures unless secret keys are known, **unlinkability** between either modified signatures or modified signature and their origin, **strong privacy** of users' information (i.e., no information can be extracted more than what was disclosed), and **anonymity/pseudonymity** of users towards service providers (i.e., a for-

Table 1: Comparison between PIMA / $\mathcal{UMS}$ schemes and related works

		PIMA	[29]	[7]	[31]	$\mathcal{UMS}$	[17]	[15]	[19]	[26]
Identity management systems properties	Unforgeability	✓	✓	✓	✓					
	Unlinkability	✓	✓[a]	✓[c]	✗					
	Strong privacy	✓	✗	✗	✓[d]					
	Ano.(1)/Pym.(2)	(2)	(1)	(1)	✗					
Malleable signatures properties	Unforgeability					✓	✓	✓	✓	✓
	Immutability					✓	✓	✓	✓	✗
	Unlinkability					✓	✓[a]	✓[a]	✓[b]	✓
	Invisibility					✓	✗	✗	✓	N.A
Computational costs	`SigKeyGen`	2E	N.A	2E	1E	2E	32E+P	7E	(l+1)E	(3l+1)E
	`UserKeyGen`	2E	3E	1E	N.A	2E	2E	1E	2E	N.A
	`Sign`	3E	(l+2)E	3lE	(l+3)E	3E	1E 103E+10P	15E	(5l+13)E	1E
	`Modify`	(3s+5)E	(l+5)E	2(l-s)E + δ	Null	(3s+5)E	102E+10P	14E	(3l+16)E	($l \cdot s$)E
	`Verify`	(l-s)E+6P	(l+5)P	γ	1E+(s+2)P	(l-s)E+6P	2E+148P	17E	8E+(4l+6)P	sE+4P

NOTE: Ano. and Pym. denote respectively anonymity and pseudonymity; E and P stand for group exponentiations and pairing costs respectively; N.A is the abbreviation for Not Applicable; l denotes the message length; s denotes the length of the modifications' set; [a], [b] and [c] respectively indicate that unlinkability is limited to (i) the incapacity to link several transactions of the same user, (ii) the incapacity to link sanitized signatures to their origin and (iii) trace-restricted unlinkability; [d] states that only the impossibility to extract information about the modified parts is satisfied; δ and γ respectively refers to the computation costs of a signature of knowledge generation and verification.

mal definition of security and privacy properties is given in Section 5.2). Table 1 shows that PIMA is the first one to achieve strong unlinkability and privacy properties. Indeed, [29] ensures partially unlinkability, while in [7], the unlinkability property is limited to trace-restricted unlinkability. Note that, unlike [7] and [29] that support users' anonymity, PIMA is designed for pseudonymous sessions. [31] partially fulfills the strong privacy property while ensuring the inability to distinguish whether a signature was redacted or it represents the original one. The second part of the table presents the security properties satisfied by the $\mathcal{UMS}$ signature scheme vs other existing sanitizable and redactable signature schemes (i.e., [17], [15], [19] and [26]). The security and privacy properties that have been addressed in this comparison include **unforgeability** of signatures by unauthorized entities, **immutability** of modifications (i.e., only admissible modifications can be performed), **unlinkability** of modified signatures to each other or to their origin, and **invisibility** of what is modifiable or what has been

modified in a message. Table 1 shows that, apart from [26], $\mathcal{UMS}$ and other works satisfy security properties (i.e., unforgeability and immutability). Moreover, the $\mathcal{UMS}$ scheme ensures strong privacy properties (i.e., unlinkability and invisibility). Indeed, other works only guarantee partial unlinkability, and only [19] satisfies the invisibility requirement. The third part details the computational costs of the system's algorithms. Note that the computation cost of group element multiplication is ignored as it is considered as negligible compared with the exponentiation and pairing computation costs. From Table 1, it can be shown that the PIMA signing algorithm is very efficient as the computation cost does not depend on the message's length (i.e., number of attributes) denoted by l. In addition, the PIMA verification algorithm has a constant computational cost in terms of pairings as all the attributes are certified in a single signature, unlike [7] which produces one signature per attribute.

4. Overview of the Unlinkable Malleable UMS Signature

4.1. UMS Algorithms

The $\mathcal{UMS}$ signature scheme relies on the following PPT algorithms (Setup, KeyGen, Sign, Modify, Verify) inspired from redactable signature scheme [32]:

$(pp, \mathtt{pk}_{sig}, \mathtt{sk}_{sig}, \mathtt{w}, \mathcal{W}) \leftarrow \mathsf{Setup}(1^{\lambda})$ takes as input a security parameter λ and outputs the public parameters pp. It also generates $(\mathtt{pk}_{sig}, \mathtt{sk}_{sig})$ the signer's pair of keys, and $\mathtt{w}$ a set of private weights associated with each type of message (i.e., attributes), and it derives the set of public weights $\mathcal{W}$.

$(\mathtt{pk}_{san}, \mathtt{sk}_{san}) \leftarrow \mathsf{KeyGen}(pp, \mathtt{pk}_{sig}, \mathtt{sk}_{sig})$ is run by the signer. It takes as input the public parameters pp and the signer's keys $(\mathtt{pk}_{sig}, \mathtt{sk}_{sig})$ and outputs the pair of keys $(\mathtt{pk}_{san}, \mathtt{sk}_{san})$ of users referred to as sanitizers.

$(m, \sigma) \leftarrow \mathsf{Sign}(pp, m, \mathtt{w}, \mathtt{sk}_{sig}, \mathtt{sk}_{san}, ADM)$ takes as input the signer's and sanitizer's private keys ($\mathtt{sk}_{sig}$ and $\mathtt{sk}_{san}$), a message m and a description of admissible modifications ADM. It then associates each message block with a specific weight in order to prevent the sanitizer either from modifying the fixed

part or adding non admissible values in the modifiable part. Finally, it signs the message with the two keys and outputs the message m and a signature σ.

$(m', \sigma', \mathtt{pk}'_{san}, \mathtt{sk}'_{san}) \leftarrow \mathsf{Modify}(pp, m, MOD, \sigma, \mathtt{sk}_{san}, \mathtt{pk}_{sig})$ first checks that a set of modifications MOD is admissible (i.e., $ADM(MOD)$=1) and accordingly modifies the message. It then randomizes the sanitizer's pair of keys w.r.t. a random r and adjusts the signature according to the new keys. The whole signature is randomized in order to ensure unlinkability between different sanitized signatures of the same message that are generated by the same source. The algorithm outputs the modified message $m' \leftarrow MOD(m)$ and the signature σ'.

$b \leftarrow \mathsf{Verify}(pp, m, \sigma, \mathtt{pk}_{sig}, \mathtt{pk}'_{san})$ takes as input the signature σ, the message m, the signer's public key $\mathtt{pk}_{sig}$ and the sanitizer's randomized public key $\mathtt{pk}'_{san}$. It outputs a bit $b \in \{0, 1\}$ to state whether σ is valid or not.

4.2. Security Properties

The $\mathcal{UMS}$ signature scheme has to support the following security properties:

Unforgeability – states that a sanitizer not holding the appropriate keys and not being authorized by the signer is not able to create valid signatures even if he colludes with other sanitizers.

Unlinkability – states that it is infeasible either to link modified versions of the same signature, or to link a modified signature to the original one.

Immutability – ensures that only admissible modifications can be performed by the sanitizers over the message received from the signer. In our case, this means that a user is not able, neither to modify a fixed part of a message, nor to modify the modifiable parts of the message with disallowed values.

Invisibility – states that a curious entity is not able to decide which parts of the message are neither modifiable nor sanitized. That means that the admissible modifications are hidden from an outsider party.

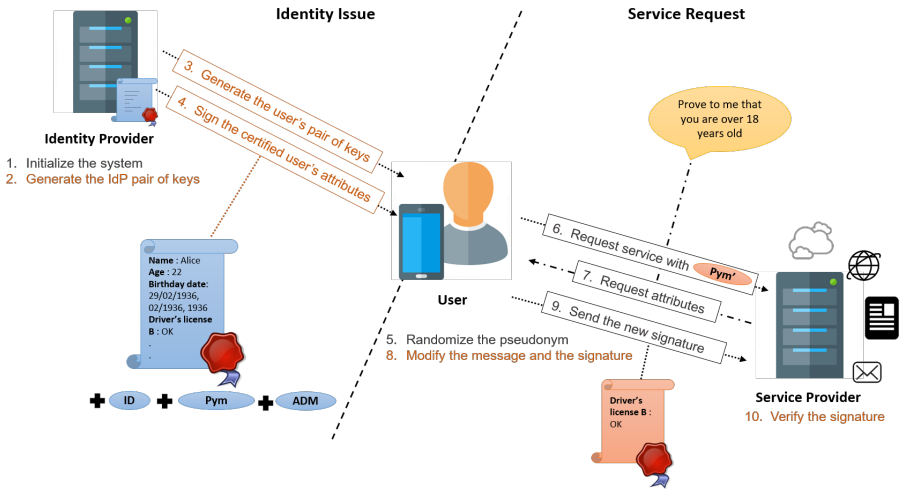


Figure 1: PIMA architecture and key interactions between entities

5. PIMA System and Threat Models

This section describes PIMA system model, including the involved entities and the high-level algorithms, and it formally defines the threat model.

5.1. System Model

ENTITIES – The proposed IMS involves three main entities namely the user (U), the identity provider (IdP) and the service provider (SP). U is the central entity that obtains her certified identity (credentials and attributes) from IdP and aims to pseudonymously get access to services offered by SPs. We assume that IdP is a fully trusted authority designed to issue identities, e.g., a prefecture or a company. In order to give access to resources and services, SPs have certain access control policies that force users to disclose some of their information. Hence, U has to select the appropriate information, show the information to the SP and prove it has been delivered by a trusted authority, i.e., the IdP.

PROCEDURES AND ALGORITHMS – PIMA system relies on the three following procedures: Setup and Initialization, Identity issue and Service request. The architecture of the new system is depicted in Figure 1.

Setup and Initialization – This procedure consists of setting up and initializing the whole system. The PIMA.$\mathtt{Setup}$ algorithm runs the $\mathcal{UMS}$.$\mathtt{Setup}$ algorithm that outputs the system global parameter pp, likely to the $\mathcal{UMS}$ scheme. The IdP key pair $(\mathtt{pk}_{IdP}, \mathtt{sk}_{IdP})$ corresponds to the signer's one generated during the $\mathcal{UMS}$.$\mathtt{Setup}$ algorithm. Next, the IdP generates a set of public parameters $\{P_n\}_{n=1}^{N}$ (N is the maximum number of possible types of attributes delivered by the IdP) that correspond to the weights associated with each type of attribute [2]. Then, the signer executes the PIMA.$\mathtt{UserKeyGen}$ algorithm using the $\mathcal{UMS}$.KeyGen one in order to generate the key pair $(\mathtt{pk}_{U_j}, \mathtt{sk}_{U_j})$ of a user U_j (referred to as the sanitizer in the $\mathcal{UMS}$.KeyGen algorithm).

Identity issue – This procedure occurs when IdP issues the identity of a requesting user U_j. Indeed, IdP generates an identifier ID to which he associates a pseudonym Pym and a set of attributes $Attr = \{a_k\}_{k=1}^{l}$ where l is the number of attributes describing the identity of U_j and $Attr \subseteq \mathcal{A}$ ($\mathcal{A}$ is the attributes' universe). IdP then defines a message m in the form of $m = \{ID_{IdP}, D, opt, Attr\}$, where ID_{IdP} is the identifier of IdP, D is a set of date values such as the identity issuing and expiration dates and opt represents other options specific to IdP. These elements form the fixed part of the message m. Next, IdP performs the PIMA.$\mathtt{Sign}$ algorithm by using the exact same algorithm as $\mathcal{UMS}$.Sign, taking as input the message m, a set of admissible modifications ADM, the secret keys of respectively IdP and U_j, the pseudonym Pym and a set of weights associated with the U_j's attributes. It produces a signature σ_{U_j}. Finally, IdP sends the tuple $(ID, Pym, m, \sigma_{U_j}, ADM)$ to U_j who locally stores them.

Service request – This procedure occurs once U_j needs to get access to a service offered by a service provider $\mathrm{SP}_j \in \mathcal{SP}$ where $\mathcal{SP}$ is the SP' universe.

[2] The objective behind the use of weights associated with each type of attribute is to prevent a malicious sanitizer from (i) modifying the message blocks with non admissible inputs as each block is associated with a specific weight known to the SP, and (ii) exchanging the attributes' values, e.g. the knowledge of the user's name attribute value "Florence" (provided with a weight P_1) cannot help to change the town attribute value to "Florence" as it is associated to a weight $P_2 \neq P_1$.

Then U_j first solicits SP_j using a new pseudonym that he derives from his local pseudonym Pym. This new pseudonym is written as $Pym_{U_j}@\mathrm{SP}_j$ for the pseudonym of U_j at SP_j. We assume that each user has different pseudonyms at different SPs in order to ensure unlinkability between several transactions. Once receiving the pseudonym $Pym_{U_j}@\mathrm{SP}_j$, SP_j picks a set of attributes $AttrReq = \{attr_j\}_{j=1}^{n}$ allowing U_j to have access to the requested service if he succeeds in proving their possession. We assume that $AttrReq$ contains the minimum number of attributes that must be provided by the user which fits the data minimization requirement w.r.t. the GDPR[3]. Next, according to the $AttrReq$ received set, U_j defines a set of possible modifications MOD (w.r.t. the set of admissible modifications ADM) and accordingly sanitizes the original signature by executing the PIMA.`Modify` algorithm which is the exact same algorithm as $\mathcal{UMS}$.Modify, taking as input the message m, the signature σ_{U_j}, the IdP public key $\mathtt{pk}_{IdP}$, the U_j's secret key sk_{U_j} and the modifications' set MOD. The algorithm then outputs the modified message m', the sanitized signature σ'_{U_j} and the randomized pair of keys $(\mathtt{sk}'_{U_j}, \mathtt{pk}'_{U_j})$ of U_j. U_j then sends the tuple $(m', \sigma'_{U_j}, \mathtt{pk}'_{U_j})$ to SP_j. This latter takes these elements with the IdP public key $\mathtt{pk}_{IdP}$ as inputs for the PIMA.`Verify` algorithm which relies on both the $\mathcal{UMS}$.Verify algorithm and two other verifications w.r.t. the user's randomized public key. SP_j checks whether the attributes of U_j are certified by IdP and provides the service if the verification succeeds, otherwise he rejects the request.

5.2. Threat Model

Two main adversaries are identified. First, malicious users that attempt to override their rights and authorizations. This model is considered against the unforgeability requirement. Second, curious service providers that attempt to collect extra information about users in order to identify the entity behind the request. This threat model is considered against both strong privacy and

[3] 2016/679 of the European parliament on the protection of natural persons with regard to the processing of personal data and on the free movement of such data.

unlinkability requirements.

5.2.1. Unforgeability

In the PIMA system, unforgeability means that it is not possible to produce a valid message/signature pair unless the private key of the IdP and the secret values of attributes' weights are known. Formally, this is defined in a game $\mathbf{Exp}_{\mathcal{A}}^{unforg}$ where an adversary $\mathcal{A}$, acting a malicious user, has access to a PIMA.`Sign` oracle. Then, $\mathcal{A}$ chooses a message m^* that has neither given before nor obtained by modifying given messages. $\mathcal{A}$ succeeds if it outputs a valid message/signature pair (m^*, σ^*) such that the PIMA.`Verify` verification holds.

Definition 1. Unforgeability – We say that PIMA satisfies the unforgeability property, if for every PPT adversary $\mathcal{A}$, there exists a negligible function κ such that $Pr[\mathbf{Exp}_{\mathcal{A}}^{unforg}(1^\lambda) = 1] \leq \kappa(\lambda)$ where $\mathbf{Exp}_{\mathcal{A}}^{unforg}$ is detailed hereafter.

$\boldsymbol{Exp}_{\mathcal{A}}^{unforg}(\lambda)$
$(pp, \mathtt{pk}_{IdP}, \mathtt{sk}_{IdP}, \mathtt{w}, \mathcal{W}) \leftarrow$ PIMA.`Setup`(λ)
$(\mathtt{pk}_u, \mathtt{sk}_u) \leftarrow$ PIMA.`UserKeyGen`$(pp, \mathtt{pk}_{IdP}, \mathtt{sk}_{IdP})$
$\mathcal{O} \leftarrow$ PIMA.`Sign`$(\cdot,\cdot,\mathtt{sk}_{\mathtt{IdP}},\cdot)$
$(m^*, \sigma^*) \leftarrow \mathcal{A}^{\mathcal{O}}\ (\mathtt{pk}_\mathtt{u},\mathtt{sk}_\mathtt{u},\mathtt{pk}_{\mathtt{IdP}},pp,\mathcal{W}, ADM)$
  letting m^i *and* σ_u^i *denote the queries and answers to and from oracle* PIMA.`Sign`
  and $m^* \notin \{MOD(m^i) \mid MOD \ \ with\ ADM^i(MOD) = 1\}$
If PIMA.`Verify`$(pp, m^*, \sigma^*, \mathtt{pk}_{\mathtt{IdP}},\mathtt{pk}_\mathtt{u}, \mathcal{W}) = 1$
  return 1
Else return 0

Remark 1. Note that we say that PIMA guarantees the strong unforgeability property as it satisfies not only the original definition of a signature scheme unforgeability, but also ensures the unforgeability of admissible modifications, i.e., which refers to the immutability property of $\mathcal{UMS}$ defined in Section 4.2.

5.2.2. Unlinkability

The unlinkability property covers two sub-properties: (i) the multi-transactions unlinkability guarantees that two or several service providers are not able to collude and link several modified signatures derived from a signature over the

same message and transmitted over several transactions, (ii) the to-original unlinkability ensures that an adversary cannot link the sanitized signature to the original one even if this latter is known.

Formally, the multi-transactions unlinkability property is defined in a game $\mathbf{Exp}_{\mathcal{A}}^{MT-Game}$ where an adversary $\mathcal{A}$ acting as a colluding curious SPs has access to a PIMA.`Modify` oracle on the same message m^* and modifications MOD^* for two service providers SP_1 and SP_2. A left-or-right oracle LoRMT is initialized with a secret random bit b and returns to $\mathcal{A}$ two modified signatures derived either from the same signature or two different signature over the same message m^* and modifications MOD^*. The adversary wins the game if he successfully predicts the bit b.

Definition 2. Multi-Transactions Unlinkability – We say that PIMA satisfies the multi-transactions unlinkability, if for every PPT adversary $\mathcal{A}$, there exists a negligible function κ, such that $Pr[\mathbf{Exp}_{\mathcal{A}}^{MT-Game}(1^\lambda) = 1] = \frac{1}{2} \pm \kappa(\lambda)$, where $\mathbf{Exp}_{\mathcal{A}}^{MT-Game}$ is represented as follows.

$\boldsymbol{Exp}_{\mathcal{A}}^{MT-Game}(\lambda)$	LoRMT $(pp,\ m^*,\ MOD^*,\ \sigma,\ \mathtt{sk}_u,\ \mathtt{sk}_{IdP},$ $\mathtt{pk}_{IdP},\ \mathtt{w},\ MOD^*,\ b)$
$(pp, \mathtt{pk}_{IdP}, \mathtt{sk}_{IdP}, \mathtt{w}, \mathcal{W}) \leftarrow$ PIMA.`Setup`(λ)	
$(\mathtt{pk}_u, \mathtt{sk}_u) \leftarrow$ PIMA.`UserKeyGen`$(pp,\ \mathtt{pk}_{IdP},$ $\mathtt{sk}_{IdP})$	**if** $(b = 0)$ **then** {
$(m^*, \sigma) \leftarrow$ PIMA.`Sign` $(pp,$ $m^*, \mathtt{w}, \mathtt{sk}_{IdP}, \mathtt{sk}_u, ADM^*)\}$	$(m'^*, \sigma_1'^*) \leftarrow$ PIMA.`Modify`$(pp,\ m^*,\ MOD^*,$ $\sigma,\ \mathtt{sk}_u,\ \mathtt{pk}_{IdP})$ for SP_1
$b \leftarrow \{0, 1\}$	$(m'^*, \sigma_2'^*) \leftarrow$ PIMA.`Modify`$(pp,\ m^*,\ MOD^*,$ $\sigma,\ \mathtt{sk}_u,\ \mathtt{pk}_{IdP})$ for SP_2 }
$\mathcal{O} \leftarrow \{$PIMA.`Modify` $(\cdot, \mathtt{sk}_u, \cdot)$ for SP_1 and SP_2, LoRMT$(\cdot, \cdot, b)\}$	**else** {
$b' \leftarrow \mathcal{A}^{\mathcal{O}}\ (pp, \mathtt{pk}_{IdP},\ \mathcal{W})$	$(m'^*, \sigma_1'^*) \leftarrow$ PIMA.`Modify`$(pp,\ m^*,\ MOD^*,$ $\sigma,\ \mathtt{sk}_u,\ \mathtt{pk}_{IdP})$ for SP_1
If $b = b'$	$(m^*, \sigma') \leftarrow$ PIMA.`Sign`$(pp,\ m^*,\ \mathtt{w},\ \mathtt{sk}_u,$ $\mathtt{sk}_{IdP},\ \sigma,\ ADM^*)$
return 1	$(m'^*, \sigma_2'^*) \leftarrow$ PIMA.`Modify`$(pp,\ m^*,\ MOD^*,$ $\sigma',\ \mathtt{sk}_u,\ \mathtt{pk}_{IdP})$ for SP_2 }
Else return 0	**return** $(\sigma_1'^*, \sigma_2'^*)$

The to-original unlinkability holds if $\mathcal{A}$, acting as a curious service provider[4], is given access to PIMA.`Modify` oracle on the same message m^* and modifications MOD^* and two signatures σ_0 and σ_1 for the same user. $\mathcal{A}$ also gets access to a left-or-right oracle LoRTO which is initialized with a secret random bit $b \in \{0,1\}$. $\mathcal{A}$ is given back a modified signature over the same message m^*, modifications MOD^* and signature σ_b. To win this game, $\mathcal{A}$ should successfully predict b.

Definition 3. To-Original Unlinkability – We say that PIMA satisfies the to-original unlinkability property, if for every PPT adversary $\mathcal{A}$, there exists a negligible function κ, such that $Pr[\mathbf{Exp}_{\mathcal{A}}^{TO-Game}(1^\lambda) = 1] = \frac{1}{2} \pm \kappa(\lambda)$, where $\mathbf{Exp}_{\mathcal{A}}^{TO-Game}$ is defined as follows.

$\boldsymbol{Exp}_{\mathcal{A}}^{TO-Game}(\lambda)$
$(pp, \mathtt{pk}_{IdP}, \mathtt{sk}_{IdP}, \mathtt{w}, \mathcal{W}) \leftarrow$ PIMA.`Setup`(λ)
$(\mathtt{pk}_u, \mathtt{sk}_u) \leftarrow$ PIMA.`UserKeyGen`$(pp,\ \mathtt{pk}_{IdP},\ \mathtt{sk}_{IdP})$
$(m^*, \sigma_0) \leftarrow$ PIMA.`Sign` $(pp, m^*, \mathtt{w}, \mathtt{sk}_{IdP}, \mathtt{sk}_u, ADM^*)\}$
$(m^*, \sigma_1) \leftarrow$ PIMA.`Sign` $(pp, m^*, \mathtt{w}, \mathtt{sk}_{IdP}, \mathtt{sk}_u, ADM^*)\}$
$b \leftarrow \{0,1\}$
$\mathcal{O} \leftarrow$ {PIMA.`Modify` $(\cdot, \sigma_0, \mathtt{sk}_u, \cdot)$, {PIMA.`Modify` $(\cdot, \sigma_1, \mathtt{sk}_u, \cdot)$, LoRTO$(\cdot, \cdot, b)$}
$b' \leftarrow \mathcal{A}^{\mathcal{O}}\ (pp, \mathtt{pk}_{IdP},\ \mathcal{W})$
If $b = b'$
  return 1
Else return 0

LoRTO $(pp,\ m^*,\ MOD^*,\ \sigma_0,\ \sigma_1,\ \mathtt{sk}_u,\ \mathtt{sk}_{IdP},\ MOD^*,\ b)$

$(m'^*, \sigma_b'^*) \leftarrow$ PIMA.`Modify`$(pp,\ m^*,\ MOD^*,\ \sigma_b,\ \mathtt{sk}_u,\ \mathtt{pk}_{IdP})$
return $\sigma_b'^*$

[4]The adversary considered against the to-original unlinkability property refers to as a curious service provider or colluding service providers. A curious IdP could be also considered against this property. This assumption does not pose plausible threats to the proposed system unless a collusion between IdP and different SPs occurs in order to trace users' transactions. Nevertheless, this assumption of collusion between the IdP and service providers contradicts the fact that the IdP is honest.

5.2.3. Strong Privacy

The strong privacy property refers to the impossibility to extract information about neither the user nor the exchanged messages, more than what was voluntarily revealed. Indeed, it is unfeasible for a curious provider neither to identify a particular user based on various transactions' information, nor to decide which data have been modified or deleted from a given message, i.e., sanitized signature. This property will be discussed informally in Section 7.

6. Concrete Construction of the PIMA System

After presenting the basics about bilinear maps, this section presents the Pointcheval-Sanders scheme and introduces the proposed modifications to be securely integrated in the PIMA system.

6.1. Bilinear Maps

Let $\mathbb{G}_1=\langle g\rangle$ and $\mathbb{G}_2=\langle \tilde{g}\rangle$ be two cyclic groups of prime order q so there exists a bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_3$ that satisfies the following properties: (i) bilinearity for all $g \in \mathbb{G}_1$, $\tilde{g} \in \mathbb{G}_2$, (ii) non-degeneracy: $e(g, \tilde{g}) \neq 1$ and (iii) $e(g, \tilde{g})$ is efficiently computable for any $g \in \mathbb{G}_1$ and $\tilde{g} \in \mathbb{G}_2$.

6.2. Pointcheval-Sanders Signature Scheme

The PS signature [11] is defined under a bilinear group of type-3 and is proven unforgeable under the LRSW assumption. It is presented as follows: The secret key is selected as $(x, y) \leftarrow {\mathbb{Z}^*_q}^2$ and the public key is computed as $(X, Y) \leftarrow (\tilde{g}^x, \tilde{g}^y)$. To sign a message $m \in \mathbb{Z}^*_q$, the signer picks a random $h \leftarrow \mathbb{G}^*_1$, computes $a := h$ and $b := h^{(x+y\cdot m)}$, and outputs the signature $\sigma = (a, b)$. To verify the validity of the signature, a verifier checks if $e(a, X \cdot Y^m) = e(b, \tilde{g})$ holds.

Modification 1 to Pointcheval-Sanders scheme – As the original scheme does not support the unlinkability features [5], based on [33] works, we propose a

[5] The randomization of the PS scheme states that based on a random $t \in \mathbb{Z}^*_q$, the randomized

modified version as follows. The generation of the secret and public keys remains the same. The signature of a message $m \in \mathbb{Z}_q^*$, requires the signer to select a random $h \leftarrow \mathbb{G}_1^*$ and a random $s \leftarrow \mathbb{Z}_q^*$, to compute $a = h^s$ and $b = h^{(x+y \cdot m)}$, and to output the signature $\sigma = (s, a, b)$. To verify that the signature is valid, a verifier checks if Equation 1 holds.

$$e(a, X \cdot Y^m) = e(b, \tilde{g})^s \tag{1}$$

Correctness. $e(a, X \cdot Y^m) = e(h^s, \tilde{g})^{x+ym} = e(h^{x+ym}, \tilde{g})^s = e(b, \tilde{g})^s$

Modification 2 to get a randomizable signature scheme – To make unlinkable several presentations of the same signature, this latter has to be randomizable. This can be obtained as follows. Given a signature $\sigma = (s, a, b)$, the signer (or any other entity) chooses two randoms $r_1, r_2 \in \mathbb{Z}_q^*$, computes $s' = r_2 s$, $a' = a^{r_1 r_2}, b' = b^{r_1}$ and outputs the new signature $\sigma' = (s', a', b')$. We can easily check that the equation 1 still holds.

Modification 3 to get randomizable keys – Randomization of keys aims at hiding the signer's identity. This can be achieved as follows. Given a signature $\sigma = (s, a, b)$, the signer selects a random $\rho \in \mathbb{Z}_q^*$, runs two algorithms `RandSK` and `RandPK` for randomizing respectively the signer's secret and public key and outputs $(x', y') = \texttt{RandSK}((x, y), \rho) = (x\rho, y\rho)$ and $(X', Y') = \texttt{RandPK}((X, Y), \rho) = (\tilde{g}^{x'}, \tilde{g}^{y'}) = (\tilde{g}^{x\rho}, \tilde{g}^{y\rho})$. Then, the signer computes $\tilde{b} = b^\rho = h^{(x\rho + my\rho)} = h^{(x' + my')}$ and outputs the signature $\tilde{\sigma} = (s, a, \tilde{b})$, which can be verified if 1 holds using the public key (X', Y').

6.3. Concrete Algorithms

This section details the concrete construction of PIMA system based on the modifications of the PS signature scheme proposed in Section 6.2.

Setup and Initialization includes two main algorithms referred to as PIMA.`Setup` (c.f., Algorithm 1) and PIMA.`UserKeyGen` (c.f., Algorithm 2).

signature is denoted as $\sigma' = (a^t, b^t)$. Relying on pairing functions, a randomized signature can be linked to its origin, which contradicts the *To-Original unlinkability* property

Algorithm 1 PIMA.`Setup` algorithm

Inputs: the security parameter λ

Output: the parameters pp, the IdP's keys ($\mathtt{pk}_{IdP}$,$\mathtt{sk}_{IdP}$) and the weights $(\mathtt{w}, \mathcal{W})$

set an asymmetric bilinear group of type 3 environment $(q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_3, e)$ where $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_3$ is an asymmetric type 3 pairing function;

pick at random $g \in \mathbb{G}_1$, $\tilde{g} \in \mathbb{G}_2$ and $x, y \in \mathbb{Z}_q^*$ and compute $X := \tilde{g}^x$; $Y := \tilde{g}^y$;

set $pp = (q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_3, e, g, \tilde{g})$; $\mathtt{pk}_{IdP} = (X, Y)$ and $\mathtt{sk}_{IdP} = (x, y)$;

pick at random $\{p_j\}_{j\in[1,P]}, \{q_j\}_{j\in[1,P]} \in \mathbb{Z}_q^*$ where P is the maximum number of attributes' types supported by the IdP;

for all $j \in \{1, \cdots, P\}$ **do**

    $P_j := \tilde{g}^{p_j y}$; $Q_j := \tilde{g}^{q_j}$;

end for

$\mathtt{w} = \{(p_j, q_j)\}_{j\in[1,P]}$; $\mathcal{W} = \{(P_j, Q_j)\}_{j\in[1,P]}$ where $\mathtt{w}$ (resp. $\mathcal{W}$) is the set of secret (resp. public) weights associated to attributes

return $(pp, \mathtt{pk}_{IdP}, \mathtt{sk}_{IdP}, \mathtt{w}, \mathcal{W})$

Note that the tuple $(q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_3, e, g, \tilde{g}, X, Y, \mathcal{W})$ is known by all the system's entities while x, y and $\mathtt{w}$ are kept secret at the IdP. Afterwards, the IdP generates the pair of keys for each of its users by running the PIMA.`UserKeyGen` algorithm (c.f., Algorithm 2). Note that the pair of keys $(\mathtt{pk}_u, \mathtt{sk}_u)$ associated with the

Algorithm 2 PIMA.`UserKeyGen` algorithm

Inputs: the public parameters pp and the key pair $(\mathtt{pk}_{IdP}, \mathtt{sk}_{IdP})$ of IdP

Output: the pair of public and private keys of U

pick two random values $\alpha, \beta \leftarrow \mathbb{Z}_q^*$;

compute $x_u = \alpha x$; $y_u = \beta y$; $(X_u = \tilde{g}^{x_u}$; $Y_u = \tilde{g}^{y_u}$;

set $\mathtt{pk}_u = (X_u, Y_u)$ and $\mathtt{sk}_u = (x_u, y_u)$;

return $(\mathtt{pk}_u, \mathtt{sk}_u)$

user's identifier ID_u is stored at both the IdP's and the user's sides [6].

Identity issue relies on the PIMA.`Sign` algorithm that is run by IdP, upon receiving a list of attributes $Attr$=$\{a_j\}_{j\in[1,N]}$ from U_j, where N is the number of

[6]The fact that the user's secret key is generated by and stored at the IdP does not affect the security of PIMA as the IdP is considered as a fully trusted authority.

user's attributes. The IdP generates an ID_u and the corresponding pseudonym Pym by picking a random value $s_u \leftarrow \mathbb{Z}_q^*$, and derives a message m, as follows: The message can be written as $m = m_{FIX} + m_{MOD}$ where m_{FIX} is the fixed part (i.e., non-modifiable blocks introduced by the IdP) and m_{MOD} is the modifiable part (i.e., the set of attributes that can be modified by the user according to the admissible modifications). This part can be represented as follows: $m_{MOD} = \sum_{j\in[1,N]}(a_j + \sum_k mod_k(a_j))$ where mod_k are the different possible modifications for a given attribute. The set of possible admissible modifications ADM represents the indices of the modifiable blocks. In our case, it is considered to be the same as $[1, N]$ since we assume that all attributes can be modified to one of the given modifications mod_k.

In the following, we assume that m is represented as $m = \{m_j\}_{j\in\{1..n\}}$ where n is the length of the message m and m_j represents either the attribute a_j, its possible modification or the fixed parts of the message. This message is signed by the identity provider as shown in Algorithm 3. Note that in order to sanitize the signature, U needs further elements than only the delivered signature. Thus, IdP generates a set $\mathcal{H}_\mathcal{U} = \{H_j\}_{j\in ADM}$ where $H_j = h_u^{p_j m_j y + q_j}$ and sets the tuple $(m, \sigma_u, g^\alpha, g^\beta, h_u^y, \mathcal{H}_\mathcal{U})$ that he sends to U.

Algorithm 3 PIMA.`Sign` algorithm

1: **Inputs:** the public parameters pp, the message m, the set of secret weights $\mathtt{w}$, secret keys $\mathtt{sk}_{IdP}$ and $\mathtt{sk}_u$ of respectively IdP and U and admissible modifications ADM
2: **Output:** the message m and the corresponding signature σ_u
3: pick at random $s_u \in \mathbb{Z}_q^*$ and $h_u \leftarrow \mathbb{G}_1^*$ where h_u is reinitialized for each issued signature ;
4: compute $a_u = h_u^{s_u}$; $b_u = h_u^{(x+\sum_{j=1}^n (m_j y + p_j m_j y + q_j))}$; $c_u = h_u^{(\alpha x + \sum_{j=1}^n m_j \beta y)}$;
5: set $\sigma_u = (s_u, a_u, b_u, c_u)$;
6: **return** (m, σ_u)

Service request involves two algorithms, i.e., PIMA.`Modify` (c.f., Algorithm 4) and PIMA.`Verify` (c.f., Algorithm 5). When U wants to access a service offered by a given SP_1, he first selects a random $r_{SP1} \in \mathbb{Z}_q^*$, computes $s_{SP1} = r_{SP1} s_u$ to be the pseudonym of U at SP_1 (c.f., Section 5) and contacts SP_1 via the generated pseudonym. SP_1 sends back the set of requested attributes

$AttrReq \in \{0,1\}^*$. Then, according to $AttrReq$ and ADM, U defines the set of modifications MOD to be performed on m, and runs PIMA.`Modify`. Note

Algorithm 4 PIMA.`Modify` algorithm

Inputs: the message m, the signature σ_u, the set of modifications MOD, the $\mathcal{H}_{\mathcal{U}}$ set, the tuple $(g^{\alpha}, g^{\beta}, h_u^y)$, the secret key $\mathtt{sk}_u$ of U and the public key $\mathtt{pk}_{IdP}$ of IdP

Output: the modified message m', the corresponding signature σ'_u and the randomized pair of keys $(\mathtt{pk}'_u, \mathtt{sk}'_u)$ of U

set $m' = \{m_j\}_{j\in\{1..n\}\setminus MOD}$;

pick a random value $r_{SP1} \leftarrow \mathbb{Z}_q^*$;

compute $a'_u = a_u^{r_{SP1}}$; $b'_u = b_u \cdot \prod_{k\in MOD}(H_k{}^{-1})(h_u^y)^{-m_k}$; $c'_u = c_u \cdot \prod_{k\in MOD}(h_u^{\beta y})^{-m_k}$);

pick two random values $\rho, z \in \mathbb{Z}_q^*$;

set $\mathtt{sk}'_u = (x'_u, y'_u) = (\alpha\rho x, \beta\rho y)$; $\mathtt{pk}'_u = (X'_u, Y'_u) = (\tilde{g}^{\alpha\rho x}, \tilde{g}^{\beta\rho y}) = (X_u^{\rho}, Y_u^{\rho})$;

compute $g^{\alpha z}$; $g^{\beta z}$; $g^{\frac{z}{\rho}}$;

pick a random value $r_1 \in \mathbb{Z}_q^*$;

compute $s_{SP1} = s_u \cdot r_{SP1}$; $\tilde{a_u} = {a'_u}^{r1}$; $\tilde{b_u} = {b'_u}^{r_1}$; $\tilde{c_u} = {c'_u}^{\rho r_1}$; $\tilde{d_u} = \tilde{b_u} \cdot \tilde{c_u}$;

set $\sigma'_u = (s_{SP1}, \tilde{a_u}, \tilde{d_u})$;

return $(m', \sigma'_u, \mathtt{pk}'_u, \mathtt{sk}'_u)$

that the user keeps secret the randomized private key $\mathtt{sk}'_u$ and sends the tuple $(m', \sigma'_u, X'_u, Y'_u, g^{\alpha z}, g^{\beta z}, g^{\frac{z}{\rho}})$ to SP_1. This latter should check the validity of the received signature, by executing the PIMA.`Verify`.

Algorithm 5 PIMA.`Verify` algorithm

Inputs: the public parameters pp, the message m', the signature σ'_u, the public keys $\mathtt{pk}_{IdP}$ and $\mathtt{pk}'_u$ of IdP and U, the $\mathcal{W}$ set, and the elements $g^{\alpha z}$, $g^{\beta z}$ and $g^{\frac{z}{\rho}}$

Output: a bit $b \in \{0,1\}$

extract a subset $\mathcal{W}_l$ of length l from $\mathcal{W}$ according to the message m' blocks;

if ($e(\tilde{a_u}, X_{IdP} \cdot X'_u \cdot \prod_{j=1}^{l} Q_j(P_j \cdot Y_{IdP} \cdot Y'_u)^{m'_j}) = e(\tilde{d_u}, \tilde{g})^{s_{SP1}}$

  and $e(g^{\frac{z}{\rho}}, X'_u) = e(g^{\alpha z}, X_{IdP})$

  and $e(g^{\frac{z}{\rho}}, Y'_u) = e(g^{\beta z}, Y_{IdP})$)

then ($b = 1$)

else ($b = 0$)

return b

Correctness. The first verification equation holds as

$$e(\tilde{a_u}, X_{IdP}.X'_u \prod Q_j(P_j Y_{IdP}.Y'_u)^{m_j}) = e(h_u^{s_u r_{SP1} r_1}, \tilde{g})^{(x+\alpha\rho x+\sum q_j+(p_j y+y+\beta\rho y)m_j)}$$

$$= e(h_u^{r_1(x+\alpha\rho x+\sum(m_j y+p_j m_j y+m_j\beta\rho y+q_j))}, \tilde{g})^{s_{SP1}}$$
$$= e(\tilde{d_u}, \tilde{g})^{s_{SP1}}$$

The second one holds as $e(g^{\frac{z}{\rho}}, X'_u) = e(g^{\frac{z}{\rho}}, \tilde{g}^{\alpha x \rho}) = e(g^{\alpha z}, \tilde{g}^{x}) = e(g^{\alpha z}, X_{IdP})$, and the last one holds as $e(g^{\frac{z}{\rho}}, Y'_u) = e(g^{\frac{z}{\rho}}, \tilde{g}^{\beta y \rho}) = e(g^{\beta z}, \tilde{g}^{y}) = e(g^{\beta z}, Y_{IdP})$.

7. Security Analysis

This section shows that PIMA satisfies the unforgeability, unlinkability and privacy requirements w.r.t. the threat models defined in Section 5.2. Furthermore, it deduces from the formal proofs of PIMA's properties that the proposed $\mathcal{UMS}$ satisfies the properties introduced in Section 4.2. Indeed, on the one hand, the strong unforgeability property of the PIMA system implies unforgeability and immutability of $\mathcal{UMS}$. On the other hand, unlinkability and strong privacy properties of PIMA imply unlinkability and invisibility of $\mathcal{UMS}$.

7.1. Unforgeability

Theorem 1 (Unforgeability)**.** The PIMA system satisfies the unforgeability requirement, with respect to $\mathbf{Exp}_{\mathcal{A}}^{unforg}$ experiment.

Proof. In this proof, we suppose that for each session i, $\mathcal{A}$ receives two signatures σ_1^i and σ_2^i over two one-block messages m_1^i and m_2^i, respectively, for a user (U). The signatures can be parsed as $(s, {a_{1u}}^i = (h_{1u})^s$, ${b_{1u}}^i = (h_{1u})^{x+m_1^i y+pm_1^i y+q}$, ${c_{1u}}^i = (h_{1u})^{\alpha * x+m_1^i \beta y}$ and $(s, {a_{2u}}^i = (h_{2u})^s$, ${b_{2u}}^i = (h_{2u})^{x+m_2^i y+pm_2^i y+q}$, ${c_{2u}}^i = (h_{2u})^{\alpha * x+m_2^i \beta y}$ with $h_{1u} = g^{v_{1u}}$ and $h_{2u} = g^{v_{2u}}$, where v_{1u} and $v_{2u} \in Z_q^*$. We suppose that $\mathcal{A}$ is extremely strong that he knows the exponents of the signature's elements. Then, $\mathcal{A}$ gets access to the following linear system.

$$v_{1u}(x + m_1^i y + pm_1^i y + q), \forall i \quad (2) \qquad v_{2u}(x + m_2^i y + pm_2^i y + q), \forall i \quad (4)$$
$$v_{1u}(\alpha * x + m_1^i \beta * y), \forall i \quad (3) \qquad v_{2u}(\alpha * x + m_2^i \beta * y), \forall i \quad (5)$$

$\mathcal{A}$ deduces the values of v_{1u} and v_{2u} as it knows αx and βy and obtains:

$$A_i = x + m_1^i y + p m_1^i y + q, \forall i \quad (6) \qquad B_i = x + m_2^i y + p m_2^i y + q, \forall i \quad (7)$$

$\mathcal{A}$ aims at forging $x + m^* y + p m^* y + q$ where m^* has not been given before. Computing (7)-(6), the linear system becomes:

$$C_i/y = 1 + p, \forall i \quad (8) \qquad D_i = x + q, \forall i \quad (9)$$

Thus, for each session i, $\mathcal{A}$ attempts to solve the same linear system, independent from m_1^i and m_2^i, with 2 equations and 4 variables (i.e., x, y, p and q), which is infeasible. As such, $\mathcal{A}$ is not able to forge $x + m^* y + p m^* y + q$. Thus, PIMA satisfies the unforgeability property. □

7.2. Unlinkability

Theorem 2 (Unlinkability)**.** The PIMA system satisfies the unlinkability requirement, with respect to multi-transactions and to-original unlinkability.

Proof. Let us start with the multi-transactions unlinkability presented in Section 5.2. We suppose that for each session i, the modification of the message m^* results in a one-block message m (i.e., $MOD^*(m^*) = m$), thus $\mathcal{A}$ receives the sanitized signatures $\sigma_1'^i$ for SP_1 and $\sigma_2'^i$ for SP_2. The signatures can be respectively parsed as $(s_{1u} = S_1, \tilde{a_{1u}}^i = (h_u)^{r'_{1i} S_1}, \tilde{d_{1u}}^i = (h_u)^{r'_{1i}(E + \rho_{1i} F)}$ and $(s_{2u} = S_2, \tilde{a_{2u}}^i = (h_u)^{r'_{2i} s_{2u}}, \tilde{d_{2u}}^i = (h_u)^{r'_{2i}(E + \rho_{2i} F)}$ with $E = x + my + mpy + q$, $F = \alpha x + m\beta y$, $h_u = g^{v_u}$ where $v_u \in \mathbb{Z}_q^*$.

We suppose that $\mathcal{A}$ is extremely strong that he knows the exponents of the signature's elements, and gets access to the following linear system of $4i$ equations and $4i + 3$ variables.

$$A_{1i}/S_1 = v_u r'_{1i}, \forall i \quad (10) \qquad A_{2i}/S_2 = v_u r'_{2i}, \forall i \quad (12)$$

$$D_{1i} = v_u r'_{1i}(E + \rho_{1i}F), \forall i \quad (11) \qquad D_{2i} = v_u r'_{2i}(E + \rho_{2i}F), \forall i \quad (13)$$

$\mathcal{A}$ is then given two sanitized signatures σ'_1 for SP_1 and σ'_2 for SP_2 that can be respectively parsed as $(s_{1u} = S_1, \tilde{a_{1u}} = (h_u)^{r'_1 S_1}, \tilde{d_{1u}} = (h_u)^{r'_1(E+\rho_1 F)})$ and $(s_{2u} = S_2, \tilde{a_{2u}} = (h_{2u})^{r'_2 S_2}, \tilde{d_{2u}} = (h_{2u})^{r'_2(E+\rho_2 F)})$ with $h_{2u} = g^{v_{2u}}$, i.e., $v_{2u} \in \mathbb{Z}_q^*$.

To break the multi-transactions unlinkability property, $\mathcal{A}$ should compare the values of v_u and v_{2u} and if $v_u = v_{2u}$, $\mathcal{A}$ can deduce that the two sanitized signatures come from the same signature, else from two different signatures. Thus, $\mathcal{A}$, being a strong adversary, establishes the following linear system and tries to make the right choice, while relying on the previous sessions results.

$$A_1/S_1 = v_u r'_1 \quad (14) \qquad A_2/S_2 = v_{2u} r'_2 \quad (16)$$

$$D_1 = v_u r'_1(E + \rho_1 F) \quad (15) \qquad D_2 = v_{2u} r'_2(E + \rho_2 F) \quad (17)$$

Combining the two linear systems, $\mathcal{A}$ attempts to solve a linear system with $4i+4$ equations and $4i+8$ unknown variables, i.e., r'_1, r'_2, ρ_1, ρ_2, E, F, v_u, v_{2u}, r'_{1i}, r'_{2i}, ρ_{1i} and ρ_{2i} $\forall i$, which is unfeasible.

For the to-original unlinkability, $\mathcal{A}$ tries to link sanitized signatures to their original signature, while relying on different sessions. $\mathcal{A}$ receives two signatures belonging to the same user (U) and over a one-block message m^*. The two signatures are denoted by σ_1 and σ_2 and can be respectively parsed as $(s_u, a_{1u} = (h_{1u})^{s_u}, b_{1u} = (h_{1u})^E, c_{1u} = (h_{1u})^F)$ and $(s_u, a_{2u} = (h_{2u})^{s_u}, b_{2u} = (h_{2u})^E, c_{2u} = (h_{2u})^F)$, with $E = x + m^* y + m^* p y + q$, $F = \alpha x + m^* \beta y$, $h_{1u} = g^{v_1}$ and $h_{2u} = g^{v_2}$ where $v_1, v_2 \in \mathbb{Z}_{||}^*$. We suppose that $MOD(m^*) = m^*$, so that for each session i, $\mathcal{A}$ receives, two sanitized signatures σ'^i_1 from σ_1 and σ'^i_2 from σ_2, that can be respectively parsed as $(S = r_s s_u, \tilde{a_{1u}}^i = (h_{1u})^{r'_{1i} S}, \tilde{d_{1u}}^i = (h_{1u})^{r'_{1i}(E+\rho_{1i}F)})$ and

$(S = r_s s_u, \tilde{a_{2u}}^i = (h_{2u})^{r'_{2i} S}, \tilde{d_{2u}}^i = (h_{2u})^{r'_{2i}(E+\rho_{2i}F)})$. We consider $\mathcal{A}$ as a strong adversary that can compute the exponents of $b_{1u}, c_{1u}, b_{2u}, c_{2u}, \tilde{a_{1u}}^i, \tilde{a_{2u}}^i, \tilde{d_{1u}}^i$ and $\tilde{d_{2u}}^i$. As such, $\mathcal{A}$ knows the following linear system of $4i+4$ equations and $4i+4$ variables where the first four equations with 4 variables gives an infinite number of solutions (v_1, v_2) where $v_2 = B_2/B_1 v_1 = C_2/C_1 v_1$.

$$B_1 = v_1 E \quad (18) \qquad A_{1i}/S = v_1 r'_{1i}, \forall i \quad (22)$$

$$C_1 = v_1 F \quad (19) \qquad D_{1i} S/A_{1i} = E + \rho_{1i} F, \forall i \quad (23)$$

$$B_2 = v_2 E \quad (20) \qquad A_{2i}/S = v_2 r'_{2i}, \forall i \quad (24)$$

$$C_2 = v_2 F \quad (21) \qquad D_{2i} S/A_{2i} = E + \rho_{2i} F, \forall i \quad (25)$$

Afterwards, $\mathcal{A}$ is given a sanitized signature σ'_b from either σ_1 or σ_2. The signature can be parsed as $(S = r_s s_u, \tilde{a_{bu}} = (h_{bu})^{r'_b S}, \tilde{d_{bu}} = (h_{bu})^{r'_b(E+\rho_b F)})$ with $h_{bu} = g^{v_b}$. The strong adversary $\mathcal{A}$ then knows the following two equations.

$$A_b/S = v_b r'_b \quad (26) \qquad D_b S/A_b = E + \rho_b F \quad (27)$$

To successfully link the sanitized signature to the associated origin, $\mathcal{A}$ should determine whether $v_b = v_1$ or $v_b = v_2$. As such, $\mathcal{A}$ tries to solve the linear system with $4i+6$ equations and $4i+7$ unknown variables, i.e., r'_b, ρ_b, v_b, E, F, v_1, v_2, r'_{1i}, r'_{2i}, ρ_{1i} and ρ_{2i} $\forall i$, which is unfeasible. □

7.3. Strong Privacy

Theorem 3 (Strong Privacy)**.** The PIMA system satisfies the strong privacy requirement, with respect to the invisibility property of $\mathcal{UMS}$ scheme.

Proof. We first detail the support of the privacy property. Then, we prove that PIMA guarantees the **strong** privacy.

In a nutshell, the notion of privacy is related to the (i) indistinguishability of signatures and (ii) the anonymity of the originator.

(i) The indistinguishability of signatures is inherited from the multi-transactions unlinkability property satisfied by PIMA, as proven in Theorem 2.

(ii) For the anonymity of the originator, we consider an adversary $\mathcal{A}$ that is allowed to request, as many times as he wants, the PIMA.`Modify` on the same message m^* and the modifications MOD^* for two users U_0 and U_1. Note that for each session, $\mathcal{A}$ receives randomized versions of the users' public keys and pseudonyms (i.e., $\mathcal{A}$ has no access to the original public keys and pseudonyms of U_0 and U_1). $\mathcal{A}$ is then given a new sanitized signature on the message m^* and modifications MOD^* either for user U_0 or user U_1. $\mathcal{A}$ outputs a bit $b \in \{0, 1\}$ and wins the game if he can successfully guess with a probability greater than $\frac{1}{2}$ the user originating the sanitized signature. Let us emphasize that thanks to the randomization of both public key and pseudonym, $\mathcal{A}$ is not able to decide which user originated the signature with a probability greater than $\frac{1}{2}$.

For the **strong** privacy feature, we show that a curious provider should not be able to decide which data has been modified or deleted. Indeed, $\mathcal{A}$ can query the PIMA.`Modify` on a same message m^* and different modifications $MOD_i \in ADM^*$. Note that the message m^* is secret from the adversary; $\mathcal{A}$ has only access to admissible modifications ADM^*.

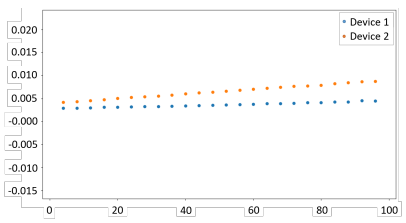
For a given challenge pairs of modifications MOD_0 and MOD_1 on the same message m^*, such that $\{MOD_0, MOD_1\} \notin \bigcup_i MOD_i$ and $|MOD_0(m^*)| = |MOD_1(m^*)|$, $\mathcal{A}$ should be able to decide which modifications are applied on the message m^*. $\mathcal{A}$ receives a new message $m_b^{'*} = MOD_b(m^*)$, such that $m_b^{'*} \notin \{MOD_i(m^*)\}$. As such, $\mathcal{A}$ has access to a new message that has not been given before and there is no combination of $\{MOD_i(m^*)\}$ that allows to derive $m_b^{'*}$. Thus, $\mathcal{A}$ is not able to successfully guess the modifications that have been applied with a probability greater than $\frac{1}{2}$. Thus, PIMA satisfies the strong privacy property. □

8. Performance Analysis

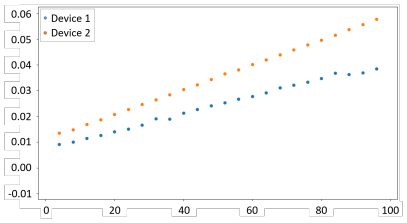
This section discusses the experimental results of PIMA, and demonstrates the practical usability of the detailed construction. All the algorithms[7] have been implemented and lead to several performance measurements relying on the two devices, i.e., Device1 is a laptop running Ubuntu 18.04.3 LTS (64 bits) with Intel Core i7 @1.30 GHz processor and 8 GB memory, and Device2 is a Smartphone running Android 10 with Snapdragon 730 Octa-Core processor and 8 GB memory. The two devices run Python v.3.6 and the associated cryptographic library, supporting bilinear pairings, *bplib* [8]. Note that these two devices represent an example of test beds. PIMA algorithms could be also deployed on iOS smartphones. The implementation relies on a bilinear elliptic curve group of 616-bits group order, which corresponds approximately to a 308-bit security level. All the experiments refer to an amount of admissible blocks to 50% and an amount of modified blocks to 25% of all blocks (i.e., 50% of admissible blocks). The processing time of PIMA algorithms is expressed according to the message blocks number, from 4 to 100 blocks. As shown in Figure 2, the tests' results are fully in line with the theoretical computational costs.

For both devices, the generation time of the key pairs of both the signer and the sanitizer remains constant, regardless of the number of blocks. Indeed, the computation time reaches 3.8ms on Device1 (resp. 4.7ms on Device2) when generating the key pair of the signer, while it is evaluated at 6.3ms on Device1 (resp. 8.3ms on Device2) for the generation of the sanitizer's pair of keys. From Figure 2, it is worth stating that the computation times of `Sign`, `Modify` and `Verify` algorithms are linear functions of the blocks' number, with different slopes. Figure 2a shows that the time for generating a signature (i.e., by IdP) takes 2 to 4ms on Device1 and 4 to 8.7ms on Device2, which is low and can satisfy many practical use cases. These results are thanks to the signing algorithm of l blocks' signature that only involves 3 modular exponentiation and l

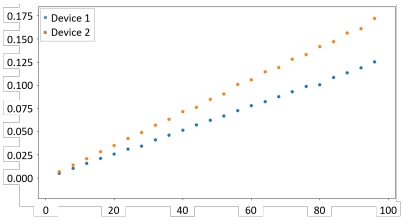
[7] The source code is available at https://github.com/soumasmoudi/malleable_unlinkable_sig
[8] https://pypi.org/project/bplib/



(a) PIMA signature processing time



(b) PIMA modification processing time



(c) PIMA verification processing time

Figure 2: Computation time (in ms) with a number of blocks varying from 4 to 100

multiplications (cf. Table 1). With consideration to the keys' generation and the message signature times, we deduce that PIMA is efficient and practical at identity providers side.

Figure 2b shows that the computation time for modifying 25% of a message is increasing significantly, varying from 9 to 40ms on Device1 and from 13.5 to 57.7ms on Device2. Indeed, as reported in Table 1, the computation of the new signature requires $3s + 5$ modular exponentiations, where s is the number of modified blocks (i.e., 25% of l). Despite the increasing computation time, a very good efficiency for PIMA can be observed on the `Modify` operation, even when running on a smartphone, thus confirming the usability of PIMA.

Finally, in Figure 2c, the processing time to verify the validity of a signature over an $(l - s)$-blocks message increases with a slope steeper than the `Modify`'s one (pursuant to computational costs in Table 1). This results in a computation time varying from 5 to 130ms on Device1 and from 6 to 172ms on Device2, due to $(l - s)$ modular exponentiations and 6 pairings. Advanced hardware features and paralleled calculations at service providers can be used to improve results.

9. Conclusion

To protect users from loosing control over their identities, and preventing service providers from building a precise profile by linking their transactions, the paper proposes PIMA, a privacy-preserving and user centric identity management system based on a new unlinkable malleable signature. The proposed system enables a user to select and hide the attributes he does not want to reveal, while proving to a service provider that the disclosed information is certified by a trusted entity in each pseudonymous session. Additionally, PIMA addresses a critical privacy concern, i.e., unlinkability, thanks to our proposed variant of the PS scheme which supports randomness of both the signature and the user's keys. Experimental results show the practical usability of our solution. Thus, PIMA meets the requirements of the modern digital society, reinforced by the new regulations, e.g., eIDAS, especially in terms of security, trust, privacy and scalability. Thanks to properties it supports, PIMA has the potential to be used as an individual wallet solution with certified attributes.

From this perspective, further experiments will be performed relying on existing users attributes' wallets with several numbers of admissible and modifiable blocks. Future research will also consider multiple identity providers such that a user is able to prove to a service provider different attributes certified by different identity providers.

References

[1] P. Bichsel, J. Camenisch, M. Dubovitskaya, R. Enderlein, S. Krenn, I. Krontiris, A. Lehmann, G. Neven, J. D. Nielsen, C. Paquin, et al., D2. 2 architecture for attribute-based credential technologies-final version, ABC4TRUST project deliverable. Onlien version available at `https://abc4trust.eu/index.php/pub`.

[2] J. Camenisch, E. Van Herreweghen, Design and implementation of the idemix anonymous credential system, in: CCS 2002, Association for Computing Machinery, New York, NY, USA, 2002, p. 21–30.

[3] J. Camenisch, A. Lysyanskaya, An efficient system for non-transferable anonymous credentials with optional anonymity revocation, in: EUROCRYPT 2001, Springer, Berlin, 2001, pp. 93–118.

[4] Microsoft, U-prove community technology, 2013.

[5] S. A. Brands, Rethinking Public Key Infrastructures and Digital Certificates: Building in Privacy, MIT Press, 2000.

[6] N. Kaaniche, M. Laurent, Attribute-based signatures for supporting anonymous certification, in: ESORICS 2016, Springer, 2016, pp. 279–300.

[7] S. Canard, R. Lescuyer, Protecting privacy by sanitizing personal data: A new approach to anonymous credentials, ASIA CCS 2013 (2013) 381–392.

[8] A. El Kaafarani, L. Chen, E. Ghadafi, J. Davenport, Attribute-based signatures with user-controlled linkability, in: Cryptology and Network Security, Springer International Publishing, Cham, 2014, pp. 256–269.

[9] M. Urquidi, D. Khader, J. Lancrenon, L. Chen, Attribute-based signatures with controllable linkability, in: M. Yung, J. Zhang, Z. Yang (Eds.), Trusted Systems, Springer International Publishing, Cham, 2016, pp. 114–129.

[10] A. El Kaafarani, E. Ghadafi, Attribute-based signatures with user-controlled linkability without random oracles, in: Cryptography and Coding, Springer International Publishing, Cham, 2017, pp. 161–184.

[11] D. Pointcheval, O. Sanders, Short randomizable signatures, in: Topics in Cryptology - CT-RSA 2016, Springer International Publishing, 2016, pp. 111–126.

[12] M. Chase, M. Kohlweiss, A. Lysyanskaya, S. Meiklejohn, Malleable signatures: New definitions and delegatable anonymous credentials, in: 2014 IEEE 27th Computer Security Foundations Symposium, 2014, pp. 199–213.

[13] G. Ateniese, D. H. Chou, B. de Medeiros, G. Tsudik, Sanitizable signatures, in: Computer Security – ESORICS 2005, Springer, Berlin, 2005, pp. 159–177.

[14] C. Brzuska, M. Fischlin, T. Freudenreich, A. Lehmann, M. Page, J. Schelbert, D. Schröder, F. Volk, Security of sanitizable signatures revisited, in: PKC 2009, Springer, Berlin, 2009, pp. 317–336.

[15] N. Fleischhacker, J. Krupp, G. Malavolta, J. Schneider, D. Schröder, M. Simkin, Efficient unlinkable sanitizable signatures from signatures with re-randomizable keys, in: PKC 2016, Springer, Berlin, 2016, pp. 301–330.

[16] C. Brzuska, M. Fischlin, A. Lehmann, D. Schröder, Unlinkability of sanitizable signatures, in: PKC 2010, Springer, Berlin, 2010, pp. 444–461.

[17] R. W. F. Lai, T. Zhang, S. S. M. Chow, D. Schröder, Efficient sanitizable signatures without random oracles, in: ESORICS 2016, Springer, 2016, pp. 363–380.

[18] C. Brzuska, H. C. Pöhls, K. Samelin, Efficient and perfectly unlinkable sanitizable signatures without group signatures, in: Public Key Infrastructures, Services and Applications, Springer, Berlin, 2014, pp. 12–30.

[19] X. Bultel, P. Lafourcade, R. W. F. Lai, G. Malavolta, D. Schröder, S. A. K. Thyagarajan, Efficient invisible and unlinkable sanitizable signatures, in: PKC 2019, Springer International Publishing, 2019, pp. 159–189.

[20] S. Canard, A. Jambert, R. Lescuyer, Sanitizable signatures with several signers and sanitizers, in: AFRICACRYPT 2012, Springer, Berlin, 2012, pp. 35–52.

[21] K. Samelin, D. Slamanig, Policy-based sanitizable signatures, in: Topics in Cryptology – CT-RSA 2020, Springer International Publishing, 2020, pp. 538–563.

[22] A. Bossuat, X. Bultel, Unlinkable and invisible γ-sanitizable signatures, in: ACNS 2021, Springer International Publishing, 2021, pp. 251–283.

[23] R. Steinfeld, L. Bull, Y. Zheng, Content extraction signatures, in: Information Security and Cryptology — ICISC 2001, Springer, Berlin, 2002, pp. 285–304.

[24] C. Brzuska, H. Busch, O. Dagdelen, M. Fischlin, M. Franz, S. Katzenbeisser, M. Manulis, C. Onete, A. Peter, B. Poettering, D. Schröder, Redactable signatures for tree-structured data: Definitions and constructions, in: ACNS, Springer, Berlin, 2010, pp. 87–104.

[25] J. Camenisch, M. Dubovitskaya, K. Haralambiev, M. Kohlweiss, Composable and modular anonymous credentials: Definitions and practical constructions, in: ASIACRYPT 2015, Springer, Berlin, 2015, pp. 262–288.

[26] O. Sanders, Efficient redactable signature and application to anonymous credentials, in: Public-Key Cryptography – PKC 2020: 23rd IACR International Conference on Practice and Theory of Public-Key Cryptography, Edinburgh, UK, May 4–7, 2020, Proceedings, Part II, Springer-Verlag, Berlin, Heidelberg, 2020, p. 628–656.

[27] H. C. Pöhls, K. Samelin, Accountable redactable signatures, in: 2015 10th International Conference on Availability, Reliability and Security, 2015, pp. 60–69.

[28] B. Chao, L. Yang, A survey of identity management technology, IEEE Int. Conference on Information and Theory Information Security (2011) 287 – 293.

[29] S. S. M. Chow, Y.-J. He, L. C. K. Hui, S. M. Yiu, Spice – simple privacy-preserving identity-management for cloud environment, in: ACNS 2012, Springer, Berlin, 2012, pp. 526–543.

[30] Z. Xu, M. Luo, N. Kumar, P. Vijayakumar, L. Li, Privacy-protection scheme based on sanitizable signature for smart mobile medical scenarios, Wireless Communications and Mobile Computing (2020) 1–10.

[31] F. Zhu, X. Yi, A. Abuadbba, I. Khalil, S. Nepal, X. Huang, Cost-effective authenticated data redaction with privacy protection in iot, IEEE Internet of Things Journal 8 (14) (2021) 11678–11689.

[32] A. Bilzhause, H. C. Pöhls, K. Samelin, Position paper: The past, present, and future of sanitizable and redactable signatures, Proceedings of the 12th International Conference on Availability, Reliability and Security (2017) 1–9.

[33] S. Zhou, D. Lin, Unlinkable randomizable signature and its application in group signature, in: Information Security and Cryptology, Springer, 2008, pp. 328–342.