BRNO UNIVERSITY OF TECHNOLOGY

FACULTY OF ELECTRICAL ENGINEERING
AND COMMUNICATION

DEPARTMENT OF TELECOMMUNICATIONS

UNIVERSITAT JAUME I

INSTITUTE OF NEW IMAGING TECHNOLOGIES

Raúl Casanova-Marqués

PRIVACY-ENHANCING TECHNOLOGIES AND PRIVACY-ENHANCING CRYPTOGRAPHY FOR WEARABLES

SHORTENED VERSION OF PH.D. THESIS

Program:	DPAD-EIT Electronics and Information Technologies (Double-Degree)
Supervisors:	doc. Ing. Jan Hajný, Ph.D. Dr. Michael Gould Carlson Dr. Joaquín Torres-Sospedra
Opponents:	Łukasz Michał Chmielewski, Ph.D. Prof. Dr. Cristiano Gonçalves Pendão
Date of Defense:	29 April 2024

KEYWORDS

Cryptographic protocols, attribute-based authentication, attribute-based credentials, privacy protection, anonymity, user revocation, smart cards, wearable architectures, Internet of Things, collaborative indoor positioning systems, elliptic curve cryptography.

ARCHIVED IN

Dissertation is available at the Science Department of Dean's Office FEEC, Brno University of Technology, Technická 10, Brno, 616 00.

1 INTRODUCTION

Recently, the proliferation of electronic systems and devices has led to an exponential increase in the amount of digital information being generated and exchanged. While this trend has opened up new opportunities for communication, commerce, and social interaction, it has also raised significant concerns about privacy and digital identity protection. Cryptography, the science of secure communication, has emerged as a crucial tool for addressing these concerns, offering techniques to encode and decode information in ways that can only be accessed by authorized parties.

However, traditional cryptographic methods are not always suitable for the complex and dynamic environments of modern electronic systems, particularly in the case of wearable devices. These devices, which are often resource-constrained and operate in uncontrolled environments, pose significant challenges for cryptographic protocols, such as the need to maintain user authenticity and prevent unauthorized access. To address these challenges, researchers are developing novel cryptographic technologies that provide attribute-based authentication, enabling more granular control over access to digital information based on user characteristics.

1.1 Thesis motivation

In safeguarding user identity, cryptographic algorithms are pivotal for ensuring the security and privacy of sensitive information, offering properties like confidentiality, integrity, and authenticity. However, in heterogeneous networks such as the *Internet of Things* (IoT), implementing standard cryptographic algorithms becomes challenging due to limited computational resources in devices. Asymmetric ciphers such as *Rivest-Shamir-Adleman* (RSA), *Digital Signature Algorithm* (DSA), or *Diffie-Hellman* (DH) may not be supported by *Central Processing Units* (CPUs) and microcontrollers, and implementing them in software may be difficult due to a lack of computational power. Fortunately, advanced cryptographic schemes like *Attribute-based Credentials* (ABCs) [1, 2, 3], known for their efficient and lightweight design, prove valuable in protecting user privacy on resource-constrained devices.

Attribute-based Credentials redefine authentication by using attributes, such as legal age or citizenship, instead of traditional credentials, empowering users to selectively disclose information for specific transactions. Compared to traditional authentication systems, ABCs [4, 5, 6] present notable advantages in addressing identity theft, data breaches, and privacy concerns, finding applications in finance and education. Their implementation on devices like smart cards was impractical until very recently, according to [7, 8, 3, 9], while efficient large-scale revocation remains a challenge. Given the widespread use of smart cards in critical domains, the development of privacy-enhancing technologies becomes crucial for securing user identities in the digital age.

As the IoT and industrial networks expand, with wearable devices gaining increased power, ensuring security and privacy in real-world applications becomes paramount. This is particularly crucial in applications like the *Collaborative Indoor Positioning System* (CIPS), which presently jeopardizes user security and privacy. CIPSs rely sensors and devices for locating individuals and objects within indoor environments through inter-user communication. However, these systems confront significant challenges such as user tracking, unauthorized access to location data, and data manipulation. Additionally, there are inherent risks of malicious attacks like spoofing or jamming [10]. Ensuring stringent

security measures, spanning from data collection to processing, is imperative to fortify the integrity of data and protect user identities within CIPSs. Inaccurate or corrupted location data can have serious consequences, especially in safety-critical applications such as emergency response or industrial settings. The prospective integration of ABCs in CIPSs holds the promise of elevating privacy and security. The development of cryptographic protocols, encompassing user authentication and confidentiality measures, is essential to instill confidence in CIPSs, positioning them as invaluable tools for multi-user collaboration. This approach adeptly addresses security and privacy concerns, unlocking the full potential of CIPSs while diligently safeguarding sensitive data and user identities.

1.2 Research questions and objectives

The advancement of wearable technology has unlocked novel opportunities for user authentication and access control in various applications. However, guaranteeing the privacy and security of user identities in such dynamic and resource-constrained environments poses considerable challenges. To enhance the comprehension of attribute-based anonymous credential schemes and their application in wearable environments, this thesis endeavors to explore these research questions:

- *How can anonymous credential schemes be adapted to support user revocation while maintaining privacy?*
- *What strategies can be employed to enable attribute-based authentication protocols on smart cards with limited support for elliptic curve cryptography?*
- *What are the usability challenges associated with using anonymous credentials in various applications, and how can they be addressed?*
- *How can anonymous credential schemes be integrated into collaborative indoor positioning systems to enhance privacy and security?*
- *How can anonymous credential schemes be implemented in resource-constrained environments, such as* IoT devices?
- *Are attribute-based authentication schemes suitable for ensuring user authenticity in dynamic wearable architectures?*

This thesis aims to contribute valuable insights into the effectiveness and scalability of attribute-based anonymous credential schemes in ensuring user authenticity and security in dynamic wearable architectures. The thesis focuses on addressing research questions and aims to design novel cryptographic algorithms that efficiently protect user privacy and digital identity in electronic systems. To attain this goal, the study tackles challenges such as inefficient revocation of invalid users, missing identification of malicious users, and performance limitations on constrained devices like wearables. The developed algorithms undergo testing and benchmarking on existing wearable hardware devices, including smart cards, smartwatches, and smartphones.

1.3 Contribution

This dissertation presents the following contributions:

- The *Revocable Keyed-Verification Anonymous Credential* (RKVAC) protocol, which allows for user authentication using anonymous credentials and supports efficient revocation, even on resource-constrained devices like smart cards.
- The implementation of the *Keyed-Verification Anonymous Credential* (KVAC) and RKVAC protocols for smart cards, leveraging Java Card technology. This involves exploiting the restricted cryptographic *Application Programming Interface* (API) of the Java Card platform and applying various optimization and acceleration techniques to further reduce execution times.
- The introduction of the *Privacy-Enhancing Authentication System* (PEAS) to assess the potential of ABC schemes for real-world applications, evaluating their maturity and readiness for deployment.
- The development of a novel decentralized privacy-preserving authentication mechanism for CIPS, offering several key benefits, including anonymized location data sharing, decentralized authentication, and offline revocation.

2 REVOCABLE ATTRIBUTE-BASED CREDENTIALS ON SMART CARDS

The practical identification and revocation of misbehaving users are crucial components of ABC schemes. Unfortunately, the cryptographic protocols employed for verifying personal attributes and revoking invalid users have been developed independently, resulting in considerable difficulties with their integration. Despite the existence of efficient attribute verification [9] and generic ABC revocation schemes [3], the intricate nature of these protocols poses significant complexity challenges to their combination. This chapter presents the RKVAC protocol as a solution to the challenge of integrating cryptographic protocols for attribute verification and revocation while preserving computational feasibility for smart cards.

2.1 Cryptographic scheme

In this section, we aim to provide a comprehensive yet accessible overview of the key entities shaping the RKVAC protocol design. Additionally, we offer a high-level exploration of the `Show` and `Verify` algorithms. These algorithms are central to the protocol, playing a crucial role in enabling secure and privacy-preserving information exchange.

To improve the clarity and legibility of this chapter, we included a table of all the symbols used in our cryptographic protocol. Table 2.1 defines each symbol and its associated meaning, enabling a thorough comprehension of the protocol's components.

Tab. 2.1: Table of symbols

Symbol	Definition
$q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \mathbf{e}, g_1, g_2$	parameters for the selected pairing-friendly elliptic curve.
$j, \alpha_1, \ldots, \alpha_j, h_1, \ldots, h_j$	parameters for the revocation authority.
$k, e_1, \ldots, e_k, \sigma_{e_1}, \ldots, \sigma_{e_k}$	user randomizers, and signed randomizers.
RL, RH, RD	revocation list, list of revocation handlers, and revocation database.
m_{ID}, m_r	user identifier and attribute for revocation.
$m_1, \ldots, m_n$	attributes with the user's information.
sk_{RA}, pk_{RA}	key pair of the revocation authority (private and public keys).
σ_{RA}	signature of the revocation authority.
sk_I, sk_V	private keys of the issuer and verifier (identical keys).
$\sigma, \sigma_{x_1}, \ldots, \sigma_n, \sigma_{x_r}$	cryptographic credential issued to the user.
$\hat{\sigma}, \hat{\sigma}_{e_I}, \hat{\sigma}_{e_{II}}, \bar{\sigma}_{e_I}, \bar{\sigma}_{e_{II}}$	cryptographic credential randomized by the user.
$\mathcal{D}$	keys to the attributes to be disclosed.
ρ	random number used to randomize the cryptographic credential.
$\rho_v, \rho_i, \rho_{m_r}, \rho_{m_{z \notin \mathcal{D}}}, \rho_{e_I}, \rho_{e_{II}}$	random numbers used to compute the protocol commitments and responses.
$t_{verify}, t_{revoke}, t_{sig}, t_{sigI}, t_{sigII}$	cryptographic commitments computed by the user.
e	challenge used in the cryptographic protocol.
$s_{m_{z \notin \mathcal{D}}}, s_v, s_{m_r}, s_i, s_{e_I}, s_{e_{II}}$	responses obtained during the execution of the cryptographic protocol.
i	unique per-session value.
C	unique one-time pseudonym.
ψ	alias of $\hat{\sigma}, \hat{\sigma}_{e_I}, \hat{\sigma}_{e_{II}}, \bar{\sigma}_{e_I}, \bar{\sigma}_{e_{II}}$.
π	alias of $e, s_{m_{z \notin \mathcal{D}}}, s_v, s_{m_r}, s_i, s_{e_I}, s_{e_{II}}$.
$t'_{verify}, t'_{revoke}, t'_{sig}, t'_{sigI}, t'_{sigII}$	cryptographic commitments reconstructed by the verifier.

The RKVAC protocol operates within a precisely defined system model consisting of several entities with distinct roles and responsibilities. To facilitate a more comprehensive understanding of the protocol's mechanisms, we hereby expound upon the entities involved in the system model depicted in Figure 2.1:

- *Revocation Authority*: assigns and issues a unique revocation handler to each user during the `IssueRA` phase. This attribute enables the revocation authority to revoke users when necessary.
- *Issuer*: assigns and emits personal attributes to users in a cryptographic credential, digitally signing them to ensure their authenticity and integrity. This process is carried out using the `IssueI` algorithm.
- *User*: acquires the cryptographic credential containing the personal attributes and leverages the `Show` algorithm to anonymously prove the possession of necessary attributes to the verifier. The user must provide evidence of the pseudonym's non-revocation status, validating its legitimacy for secure communication within the system.
- *Verifier*: validates the possession of the necessary attributes and the revocation status of the revocation handler through the use of the `Verify` algorithm. By confirming the validity of the attributes and the revocation status, the verifier can grant or deny access to the requested service, maintaining the security and privacy of the system.

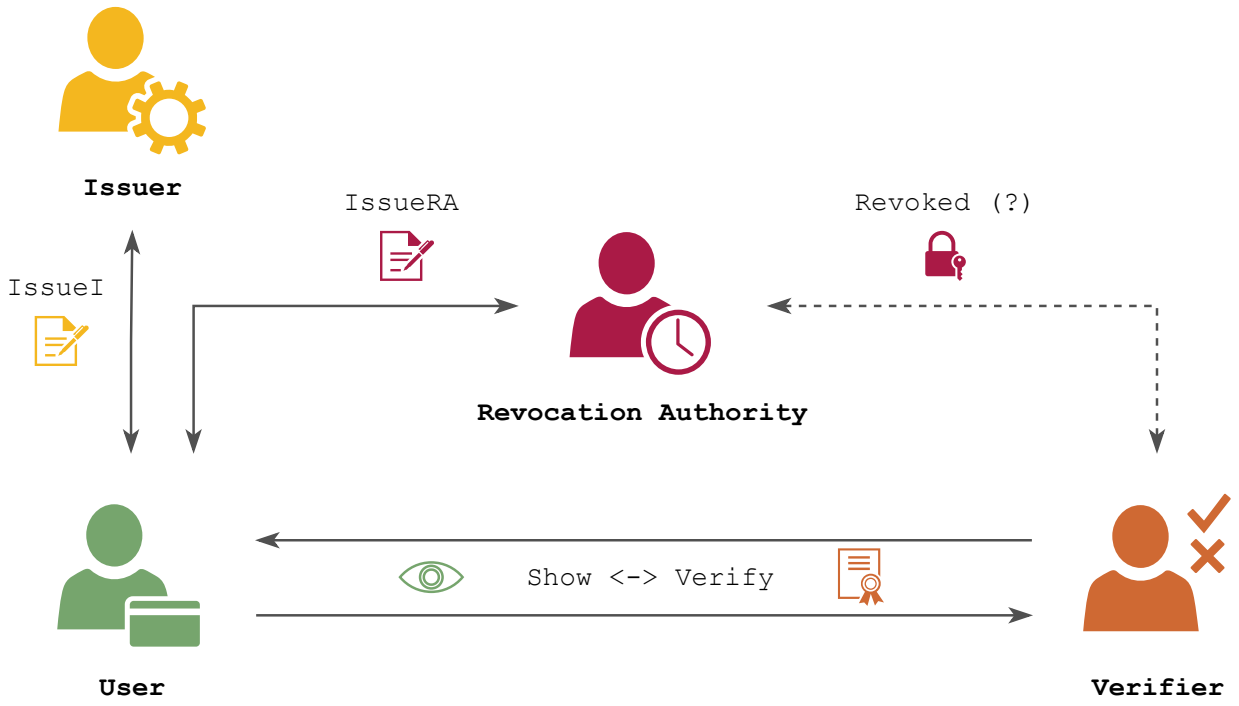
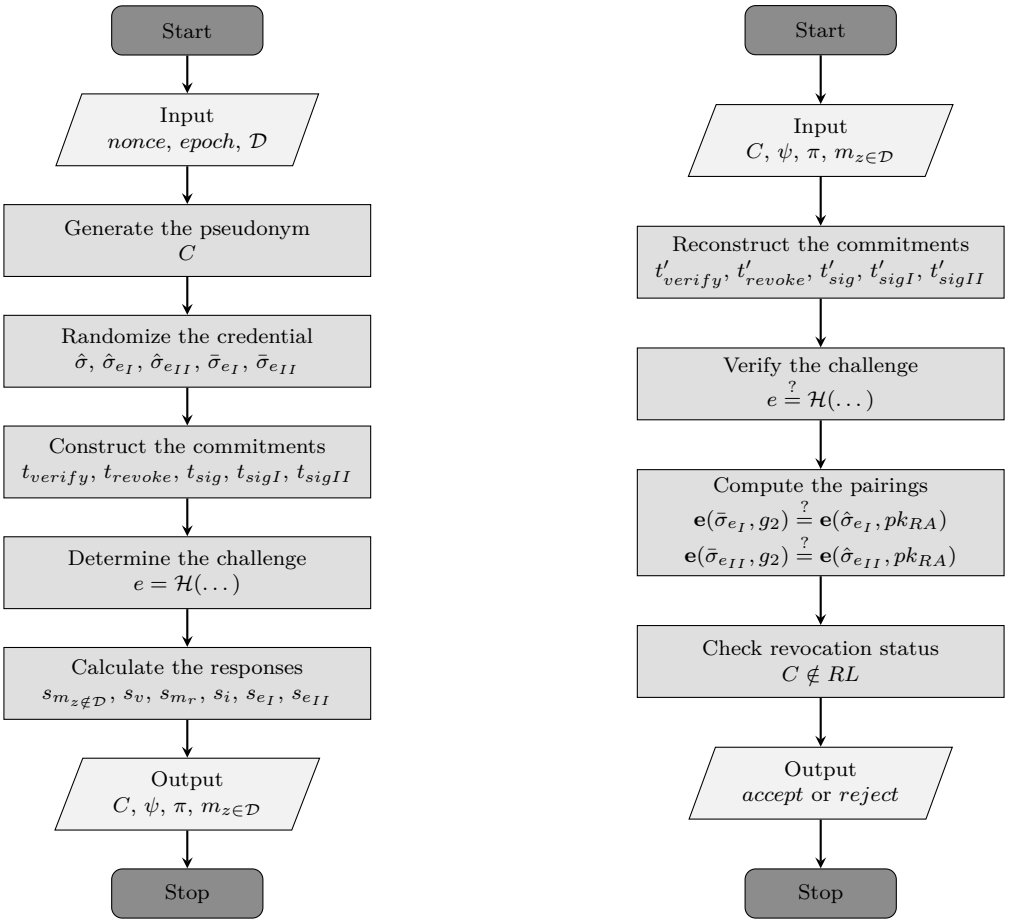


Fig. 2.1: Entities and algorithms constituting the RKVAC protocol

The flowcharts depicted in Figure 2.2 provide a high-level description of the `Show` and `Verify` algorithms and are a valuable resource for gaining a thorough understanding of the protocols, their underlying mechanisms, and the succession of steps involved in their execution. By illustrating the key components of each algorithm and their interrelationships, the flowcharts enable readers to rapidly comprehend the fundamental concepts of our cryptographic protocol.



(a) High-level `Show` algorithm

(b) High-level `Verify` algorithm

Fig. 2.2: High-level definition of the `Show` and `Verify` algorithms

2.2 Implementation details

In this cryptographic protocol implementation, the system is divided into two distinct parts: the desktop application and the smart card application. The desktop application acts as the revocation authority, issuer, and verifier, while the smart card application represents the user.

The protocol was designed and implemented using elliptic curve cryptography. Specifically, for the desktop application, we utilized the *Barreto-Naehrig 254-bit* (BN254) curve supplied by the `mcl` library [11]. Moreover, we used the `OpenSSL` library [12] for the hash functions required by the protocol. In contrast, to design the elliptic curve functions for the smart card application, we employed the MULTOS assembly provided by the SmartDeck *Software Development Kit* (SDK) [13, 14, 15, 16]. The MULTOS assembly reference [15] was utilized to construct essential functionalities like modular addition, subtraction, multiplication, elliptic curve addition, elliptic curve scalar multiplication, and other pertinent functions. Through this SDK, the smart card application can proficiently execute the required cryptographic computations while ensuring high-level security. By harnessing industry-standard libraries and SDKs, the system can conduct the essential cryptographic computations with high accuracy and dependability.

2.3 Experimental results

To ascertain the practicality and assess the efficiency and speed of the algorithms, we conducted a series of experiments, with a focus on benchmarking the `Show` algorithm. Figure 2.3 presents the results of these experiments, showcasing the benchmarks in milliseconds and accounting for protocol run times and communication overhead.

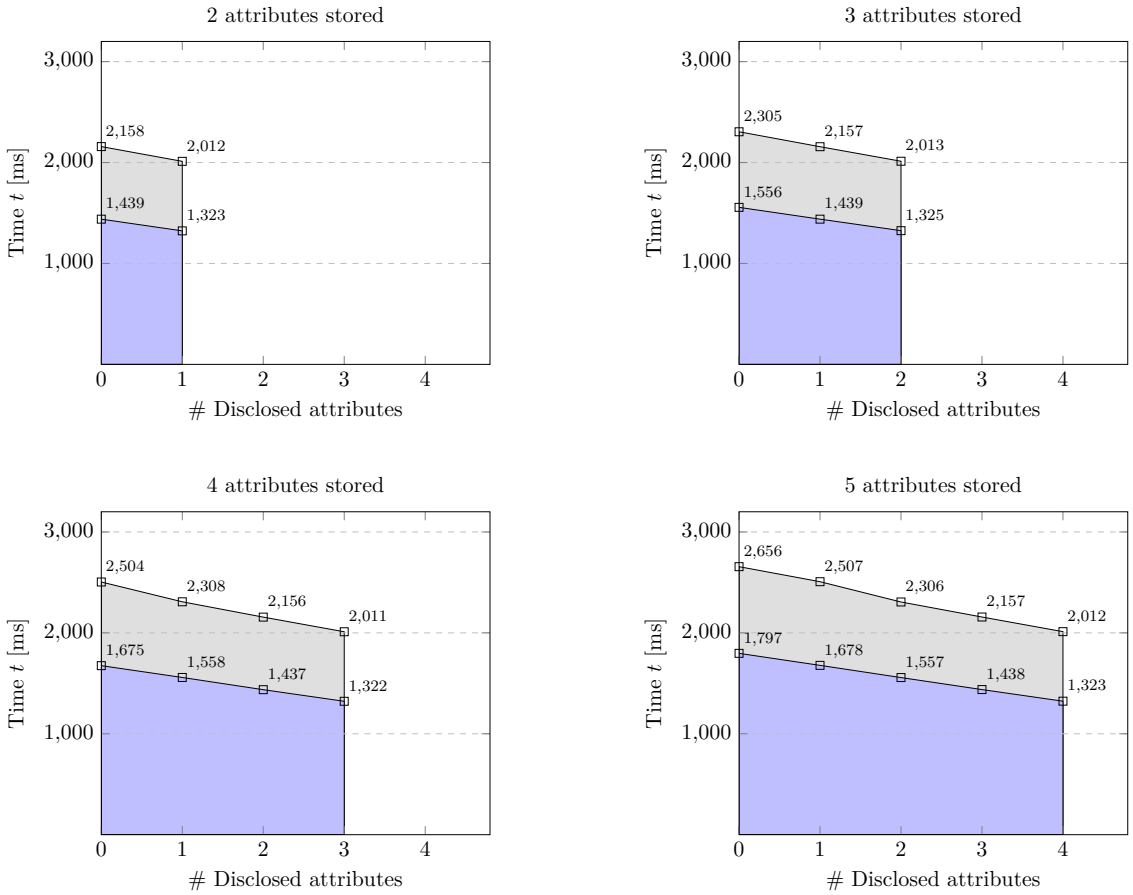


Fig. 2.3: Speed comparison of the `Show` algorithm and the transmission overhead

The graphical representations illustrate the runtime of the `Show` algorithm across a range of 2 to 5 attributes. It is worth noting that the revocation attribute remains hidden at all times; therefore, the maximum number of attributes that can be disclosed is $n-1$. Consistency in runtime is observed when all attributes are disclosed; however, for each undisclosed attribute, the proof generation time increases by approximately 150 milliseconds due to additional cryptographic operations. Credentials with a higher number of attributes exhibit longer computation times. The swiftest `Show` algorithm execution records at 1,322 milliseconds, expanding to around 2 seconds when factoring in communication overhead. Conversely, the slowest execution time demonstrates a linear growth pattern with an increasing number of attributes. Communication overhead for all transmissions amounts to approximately 700 milliseconds. Since the T=0 protocol is used, it is not feasible to reduce communication time. Owing to their microsecond-scale execution on a Raspberry Pi 4 Model B, `Verify` algorithm benchmarks are omitted from the analysis.

3 BOOSTING REVOCABLE ATTRIBUTE-BASED CREDENTIALS ON JAVA CARDS

In the previous chapter, we introduced an ABC scheme for smart cards and demonstrated its performance on MULTOS-based smart cards. While these cards offer numerous benefits, they are less accessible to the general consumer. Additionally, support for cryptographic operations on elliptic curves is typically only available upon request. In contrast, Java Cards are more widely available. Unfortunately, their support for elliptic curve cryptography is severely limited. This chapter outlines strategies for optimizing the coprocessor's capabilities to enable ABC schemes on Java Cards.

3.1 Java Card technology

In this section, we present a comprehensive overview of the Java Card technology, its usability for the development of security applications, and its cryptographic support.

Java Card technology is a subset of the Java programming language that is specifically designed for use in small, resource-constrained devices such as smart cards. It enables the secure execution of Java-based applets on these devices. Java Card achieves great interoperability and platform independence by combining the *Java Card Virtual Machine* (JCVM) and standard libraries with a well-defined and documented API. This allows the same applet to run on different smart cards while maintaining the highest certification levels and standard compliance, and permits applets to be developed on one platform and deployed on another with minimal modification. The JCVM is a highly optimized runtime environment that is designed to run on resource-constrained devices. It includes a set of core Java classes, such as the `Object` class, as well as Java card-specific classes that provide access to the smart card's hardware resources.

Java Card Applets are small programs that run on the JCVM and provide specific functionality to the smart card. They are written in Java and are compiled into bytecode that is compatible with the JCVM. Applets are loaded onto the smart card and can be updated or deleted as needed. They are state machines that respond to commands received via the reader device by transmitting and receiving status codes and data.

One of the key features of Java Card technology is its security architecture. Java Card applets are executed within a secure sandbox environment that isolates them from the rest of the smart card's operating system and other applets. This provides a high degree of security and prevents unauthorized access to sensitive data or functions. In addition to its security features, Java Card technology provides several benefits for smart card developers. Because it is based on the Java programming language, it is easy to learn and use and provides a familiar development environment for Java developers.

Java Card technology is used in applications such as secure access control systems, electronic payment systems, and public transportation systems. These applications rely on the security, interoperability, and platform independence of Java Card technology to provide secure and reliable services to users.

3.1.1 Cryptographic support

The Java Card API provides extensive support for a multitude of standard cryptographic algorithms, including symmetric encryption protocols, public-key cryptosystems, and message digest algorithms. One particularly notable feature of the Java Card API is its support for elliptic curve cryptography, which has gained increasing attention as a viable alternative to conventional public-key cryptography algorithms.

Elliptic Curve Cryptography (ECC) is based on the mathematical theory of elliptic curves and is known for its superior security characteristics compared to traditional public-key cryptosystems. Regrettably, the Java Card API lacks built-in support for essential elliptic curve primitives, including point addition, scalar point multiplication, and modular arithmetic, among other crucial algebraic operations necessary for ECC implementation.

It is worth noting that the Java Card API provides only partial access to the underlying algebraic operations required for cryptographic algorithms. As a result, it is impractical to construct efficient non-standard ECC applications using Java Card. Consequently, the current level of elliptic curve operations support provided by the Java Card API is inadequate for implementing non-standard cryptographic applications that require the use of ECC, such as attribute-based credential schemes.

3.2 Implementation details

This section discusses the application's design, the implementation of modular and elliptic curve operations, and the optimization techniques for KVAC and RKVAC to enhance their performance.

First and foremost, we used `JCAlgTest` [17], an automated testing tool, to determine the cryptographic algorithms supported by Java-based smart cards. Based on these results and considering the algorithms we need for the implementation of KVAC and RKVAC schemes, we chose the Java Card J3H145. One of the key advantages of this card is the combination of accessible data memory and support for the required cryptographic algorithms. On the other hand, its reduced *Random Access Memory* (RAM) capacity makes it impossible to run the whole protocol on it. This slows down the computation and, therefore, the execution time.

3.2.1 Application design

We built the Java Card application using the *Model-View-Controller* (MVC) design paradigm. This approach allowed us to partition the code into three main components: *models* for data storage, *views* for *Application Protocol Data Unit* (APDU) messages, and *controllers* for program logic. The application's models are crucial for processing and storing application data. To represent the system entities, the program uses three core models: the revocation authority, the issuer, and the user. Additionally, three auxiliary models separate the user data during the computation of the proof of knowledge. Controllers direct the application logic and serve as a bridge between views and models. Each core model has its own controller, with the user's controller also managing their auxiliary models. The application has only one view, which is the main class that drives the applet. This is due to the smart card's lack of a graphical interface, and thus, the transmission and reception of data through APDU messages are considered views.

In addition to the MVC components, we developed auxiliary code to enhance Java Card functionality. This code is divided into three parts: *(i)* arithmetic operations; *(ii)* memory management; and *(iii)* utility classes. The arithmetic operations group comprises classes that implement modular and elliptic curve operations. We designed a class to provide domain parameters for BN254 curves [18], which are not officially supported by the Java Card API. We also defined data types to represent elliptic curve information and implemented arithmetic functions to utilize the coprocessor as much as possible. The memory management classes manage smart card resources, partitioning RAM into segments for mathematical operations. We created a handler to temporarily lock these segments to execute mathematical operations on them and speed up execution. Once the segment is no longer required, it can be unlocked for future use. Lastly, due to constraints in the Java Card version, we developed utility classes and methods for dealing with lists, bit processing, and other functions. These utility classes, while not part of the MVC design, are crucial for the application's development.

3.2.2 Arithmetic operations

Basic algebraic operations, such as modular arithmetic or elliptic curve calculations, are not supported by the Java Card API. We set out to leverage the coprocessor as much as possible when implementing the operations required by the KVAC and RKVAC schemes.

Hardware-accelerated execution of arithmetic operations is available through the `BigNumber` class within the `javacardx.framework.math` package of the Java Card API. However, this package is optional and not available on most smart cards. Therefore, we combined software implementation with various hardware algorithms. Simple operations like addition, subtraction, negation, and division do not exert excessive strain on the CPU. They do not demand any type of hardware acceleration because the CPU can handle them. For this reason, we adopted the implementation of the `Bignat` library [19] and tweaked it to meet our requirements. Nevertheless, multiplication and inversion, being more costly operations, require the use of the coprocessor. We used RSA encryption with padding disabled for these computations. To compute the modular multiplication, we combined software execution with the coprocessor, using the RSA routine for square computations and the CPU for addition, subtraction, and division by two. For modular inversion, we used the value $q - 2$ as the exponent in the RSA encryption routine to obtain the inverse of a number. To expedite multiplication, we may reuse square computations (a^2 and b^2) for common operands. Similarly, we could precompute the common $q - 2$ value to accelerate inverse execution. Note that q is the prime order of the *Elliptic Curve* (EC) base point. Finally, for modular reduction, we divided the value to be reduced by the order of the elliptic curve.

The implementation of elliptic curve operations presented a challenge since execution on the CPU would bring considerable overhead. Indeed, it was crucial to use hardware acceleration. We harnessed the coprocessor's capabilities through the utilization of the *Password Authenticated Connection Establishment* (PACE) *Generic Mapping* (GM) algorithm and the variant of *Elliptic Curve Diffie-Hellman* (ECDH) that yields the plain `X` and `Y` coordinates. Notably, these functionalities are officially accessible as of Java Card version 3.0.5; nonetheless, we found their availability from version 3.0.4 onwards. Thus, we rely on the class `KeyAgreement` of the Java Card API to perform point addition and scalar point multiplication operations.

3.2.3 Acceleration techniques

We propose an off-card precomputation approach to accelerate the execution of the `Show` algorithm, which is shown in Figure 2.2a at a high level. By precomputing roughly three of the four stated stages, i.e., selecting a unique per-session value, generating the transaction pseudonym, and randomizing the credential, we can minimize the execution times shown in Figure 3.1b by more than 50%. The `Show` algorithm enables anonymous user authentication. However, for the same epoch value, the number of authentications is limited to 100. Table 3.1 illustrates the amount of storage space necessary to store each precomputation, as well as the precomputations for the 100 authentications.

Tab. 3.1: Space required for authentication precomputations

Stage	$n = 1$	$n = 100$
1) Select a unique per-session value	32 bytes	3.2 KB
2) Generate the transaction pseudonym	65 bytes	6.5 KB
3) Randomize the credential	517 bytes	51.7 KB
4) Compute the proof of knowledge (*partial*)	325 bytes	32.5 KB
Total	939 bytes	93.9 KB

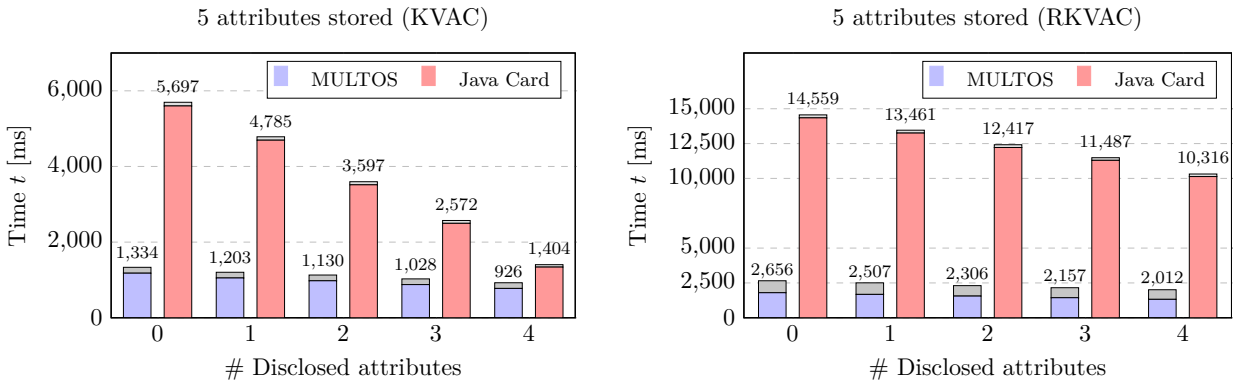
The computation of the proof of knowledge for undisclosed attributes was omitted from the precomputation due to session data requirements. Specifically, the nonce and attributes to disclose are not known in advance. Despite space constraints on the smart card, it was feasible to store the 100 precomputations. We also explored the inclusion of undisclosed attribute computations, considering five disclosure states. However, due to space limitations, this was only implemented for testing and benchmarking purposes.

3.3 Experimental results

To ascertain the feasibility of our approach, we conducted several experiments, measuring and comparing the execution speed with that of the MULTOS-based smart card implementation. Figure 3.1a and Figure 3.1b illustrate the performance of the KVAC and RKVAC protocols using MULTOS and Java Card technologies, respectively. Both figures depict the benchmarks in milliseconds and include both computation time and communication overhead.

We can observe that the Java Card implementation of the KVAC protocol takes approximately the same amount of time to disclose all of its attributes as the MULTOS implementation does without disclosure. The protocol takes roughly one second longer to run for each attribute that is not disclosed. The complexity of the RKVAC protocol is mirrored in the execution speed of the Java Card implementation, which is close to five times slower than the MULTOS implementation. Additionally, likewise with the KVAC protocol, the execution time is increased by around one second for each undisclosed attribute. However, we can appreciate that the communication overhead in Java Card is significantly inferior to that of MULTOS. We achieve this gain by using the T=1 communication protocol, which enables us to transmit all the data in a single message rather than split it into several packets.

We applied the acceleration techniques described in Subsection 3.2.3, obtaining the results depicted in Figure 3.2. Both acceleration techniques fully precalculate the first three phases of the `Show`



(a) Execution of KVAC scheme

(b) Execution of RKVAC scheme

Fig. 3.1: Speed comparison between MULTOS and Java Card implementations

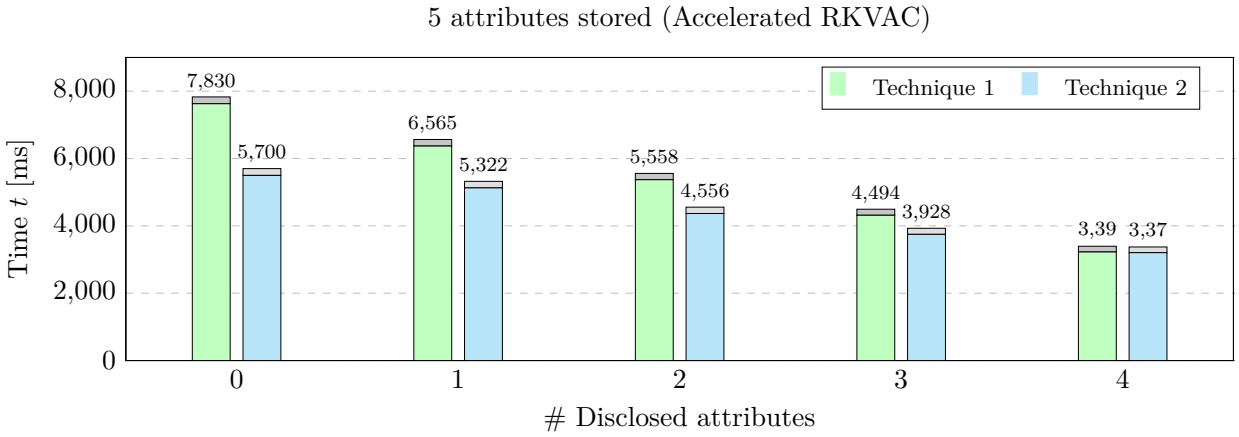


Fig. 3.2: Speed comparison between acceleration techniques for RKVAC

protocol (see Table 3.1). Concerning the fourth phase, the first technique precomputes the values $t_{revoke}, t_{sig}, t_{sigI}, t_{sigII}$ and partially t_{verify}, since the number of attributes to be disclosed is unknown. Alternatively, the second technique precomputes the five possible values of t_{verify}, i.e., when no attributes are disclosed, when one attribute is disclosed, and so forth, and employs the one that is necessary at the time. Through the use of these acceleration techniques, we detected a remarkable decrease in the execution duration of the protocol.

Note that the issuer (i.e., an external device such as a Raspberry Pi) performs the authentication precalculations. This information is stored on the smart card during the personalization phase for use in the subsequent authentication process. We did not include the time necessary to precompute the 100 authentications since it is negligible, requiring less than a half-second for both acceleration techniques. The execution times achieved are still far from those we found with the MULTOS-based smart cards. Evaluating both acceleration techniques, we can glean that the gain obtained by precomputing the attribute disclosure is impractical if we consider the memory space it requires. Moreover, due to space limitations, it is not possible to store all 100 precomputations, including attribute disclosure, on the smart card. We can state that the optimal configuration is the first acceleration technique.

4 PRIVACY-ENHANCING AUTHENTICATION SYSTEM

In the previous chapter, we presented a comprehensive demonstration of how to implement an ABC scheme using Java Card technology, given the challenges of procuring MULTOS smart cards. Nonetheless, the use of smart cards poses some functional challenges, including the lack of visibility regarding the attributes requested and the absence of an option to accept or decline their disclosure. In this chapter, we introduce the PEAS, a robust solution that is fully prepared for deployment in real-world scenarios. PEAS is compatible with smart cards and works seamlessly with smartphones and smartwatches.

4.1 System architecture and technical aspects

This section elucidates the technical aspects of our PEAS, which we have developed to enable secure and anonymous access to both electronic services and physically protected areas. By leveraging our technology, users can preserve their anonymity without disclosing their complete identity or becoming subject to tracking or profiling by system administrators. Nevertheless, in the event of malicious activities, the revocation authority plays a vital role in detecting and identifying any rogue users. By collaborating with the revocation authority, PEAS guarantees that, although user identities are anonymous, any illegal activity can still be traced back to the perpetrator.

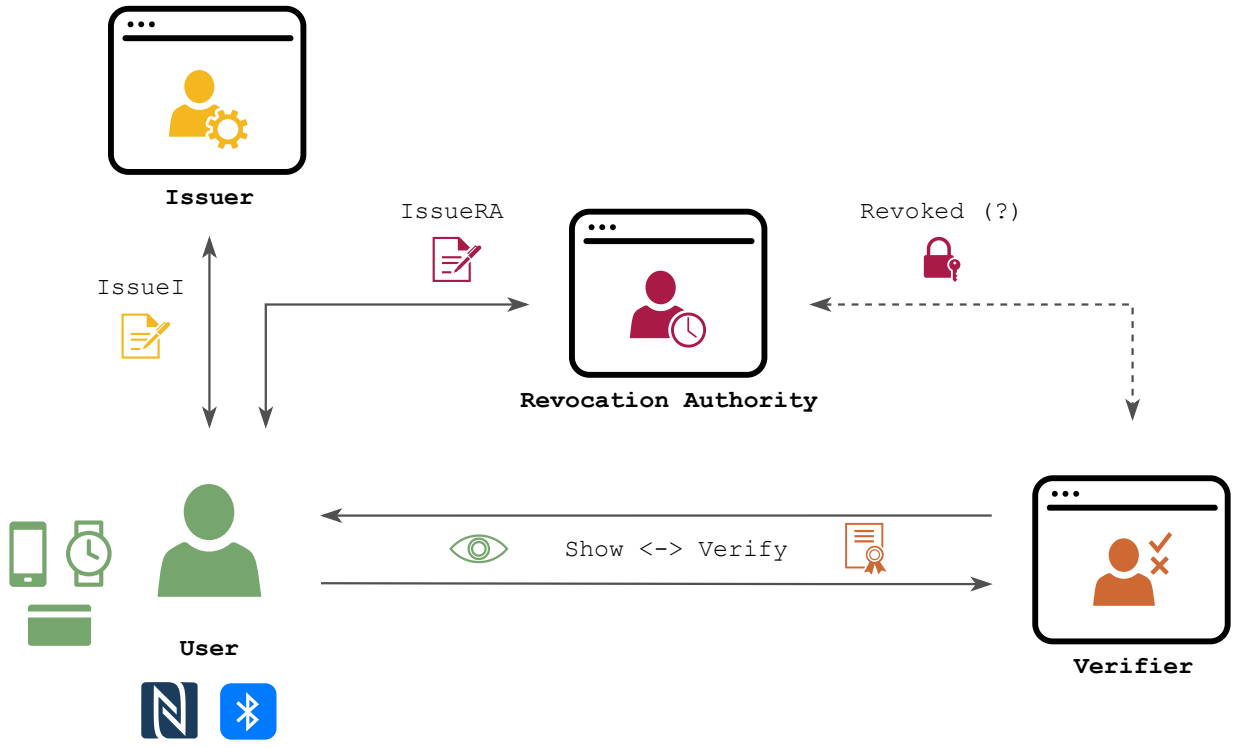


Fig. 4.1: High-level topology of PEAS

PEAS represents user identity through multiple attributes, including but not limited to name, surname, age, gender, and job title. During the verification process, only necessary identity attributes, such as age, gender, and job title, are disclosed, ensuring that the user's privacy is safeguarded.

Furthermore, the authentication sessions of PEAS are mutually unlinkable, guaranteeing the non-profiling and non-tracking of users.

The system is built upon the RKVAC technology proposed by Hajny et al. [20] and presented in Chapter 2, which serves as the cryptographic core of the system. Likewise, we optimized and expanded the RKVAC technology for deployment in real-world scenarios. PEAS is highly practical and adaptable to various platforms, including smart cards, smartphones, smartwatches, and single-board computers, commonly employed in IoT environments. Moreover, PEAS adheres to a user-centric approach, where users have complete authority over their personal data, deciding which information to provide and when to offer it to service providers. This approach ensures that irrelevant personal data is not stored, aligning with the *General Data Protection Regulation* (GDPR).

Our implementation of PEAS comprises software applications for all entities involved in the authentication process, namely the revocation authority, the issuer of attributes, the verifier of attributes, and user devices holding personal attributes. Figure 4.1 provides a detailed illustration of the PEAS architecture.

4.2 Implementation details

This section describes the development process and summarizes the main key points considered during the design of the system. We utilized personal devices that were equipped with *Personal Computer / Smart Card* (PC/SC) and Bluetooth interfaces to establish communication with the system. Specifically, the smart card interfaced with the system through the PC/SC interface in contact mode, whereas the smartphone supported both contactless PC/SC (i.e., *Near-Field Communication* (NFC)) and Bluetooth connectivity. In contrast, the smartwatch exclusively relied on Bluetooth for communication with the system.

We divided the implementation of the application into three components: *(i)* the *libpeas* library for core functionality; *(ii)* the *server-side* to operate with the revocation authority, the issuer, and the verifier; and *(iii)* the *client-side* for user functionality.

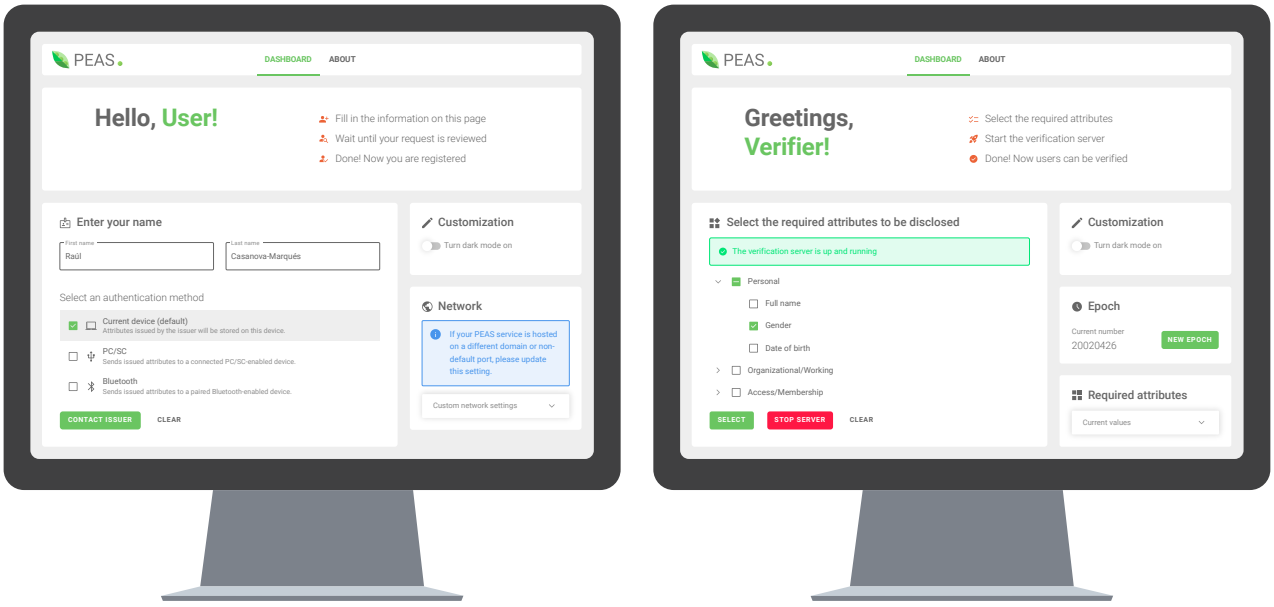
4.2.1 Core library

At the core of our application is the purpose-built *libpeas* library, a C-based framework with optimized cryptographic functions for user revocation, authentication, verification, and attribute issuance. It facilitates efficient data transmission across interfaces like PC/SC, Bluetooth, and *Transmission Control Protocol* (TCP), as well as secure database storage. This library ensures unparalleled performance and security on various devices and relies on several third-party libraries. Specifically, we utilized `mcl` [11] for elliptic curve cryptography; `openssl` [12] for cryptographic hash algorithm support; `pcsc-lite` [21], `ccid` [22], and `bluetooth` [23] for device communications; and `libcjson` [24] and `libwebsockets` [25] for web server connections.

The protocol was designed and implemented using elliptic curve cryptography. Among various libraries, `mcl` stood out for its promising performance and high security [26], supporting *Barreto-Naehrig* (BN) [18] pairing-friendly elliptic curves. We adopted the BN254 curve from `mcl` for our implementation.

4.2.2 Server-side specifics

The server implementation features a highly modular design, facilitating easy maintenance and expansion. It consists of a front-end and a back-end. The front-end, developed with `Node.js` [27] and `Vue.js` [28], provides a user-friendly interface with distinct applications for each system entity: revocation authority, issuer, user, and verifier. Each entity has specific functionalities, enhancing the system's security and user experience. The back-end uses the *libpeas* library to support cryptographic operations based on the RKVAC. It also ensures robust system management, secure data storage, and seamless communication with all entities.



(a) Dashboard for users

(b) Dashboard for verifiers

Fig. 4.2: Example of web-based dashboards

Figures 4.2a and 4.2b depict the user and verifier web-based dashboards, respectively. These dashboards offer a spectrum of authentication methods, customizable attribute disclosure options, the capability to modify the current epoch, and the ability to scrutinize the user's access history. Communication between the frontend, backend, and other entities is facilitated through various interfaces, including the *REpresentational State Transfer* (REST) API, *Command Line Interface* (CLI), TCP socket, and WebSocket. Lastly, we used `Docker` for straightforward deployment.

4.2.3 Client-side specifics

The implementation of PEAS for user devices involves the deployment of two distinct applications, namely the MULTOS application for smart cards and the Android application for smartphones and smartwatches. These applications are designed to be fully interoperable with each other and other PEAS entities. Each of the applications can store up to nine personal attributes, with the tenth reserved for the revocation attribute.

We utilized the MULTOS ML4 smart card for the smart card application, chosen for its hardware acceleration for modular arithmetic and elliptic curve operations. However, it only supports the T=0 transmission protocol and has limited RAM, which can impact its effectiveness. The Android applications, developed in Java and C, require at least SDK version 24 and use the `mcl` library, along with standard Android and Java libraries. The smartphone version provides a 4-digit *Personal Identification Number* (PIN) code for security, with an option for fingerprint access. Users can disable attribute issuance for security reasons, and the application maintains a history of events (i.e., logs). Communication with other system entities is facilitated using NFC and Bluetooth technologies. Users can attach their smartphones to an NFC reader or select a paired terminal from a drop-down list and confirm it by clicking on the `Connect` button if they intend to use Bluetooth.

(a) Application for smartphones (b) Application for smartwatches

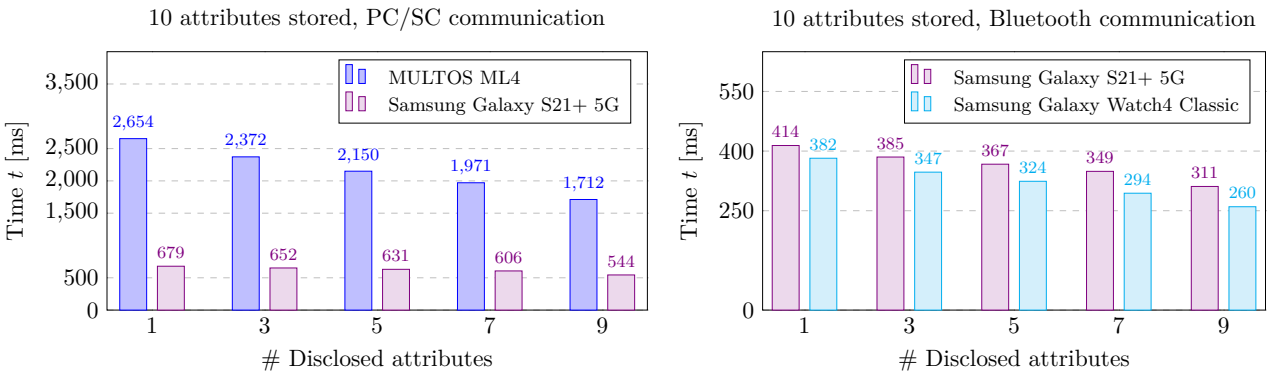
Fig. 4.3: Example of Android-based applications

Figure 4.3a and Figure 4.3b illustrate the graphical interfaces of the PEAS application for Android-based smartphones and smartwatches, respectively. The application shares fundamental similarities with its smartphone counterpart, but with one significant distinction: it lacks a login dialog consisting of a PIN code and fingerprint to enhance user-friendliness and convenience when accessing the application on the go.

4.3 Experimental results

This section presents the benchmarks of the `Show` and `Verify` algorithm implementations. The findings are organized according to the communication interface deployed. Figure 4.4 depicts the benchmarks for devices enabled with PC/SC and Bluetooth capabilities.

In each scenario, we assume the cryptographic credential stores 10 attributes, of which 1, 3, 5, 7, and 9 are disclosed while the remaining attributes remain concealed. To prove knowledge of each hidden attribute, the user's device must compute a corresponding proof. Our benchmarks consider both compute and transmission latency and are reported in milliseconds. The entities involved in the tests were the issuer, verifier, and revocation authority, run on a conventional personal computer, and the user entity, represented by a smart card, smartphone, and smartwatch.



(a) Execution using PC/SC communication

(b) Execution using Bluetooth communication

Fig. 4.4: Speed comparison of PEAS execution (client-side)

The smart card operates at a rate three to four times slower than the smartphone, as depicted in Figure 4.4a. The smartphone and smartwatch demonstrate similar performance, as shown in Figure 4.4b. When comparing the results obtained from the smartphone with PC/SC and Bluetooth interfaces, Bluetooth proves faster due to reduced communication overhead.

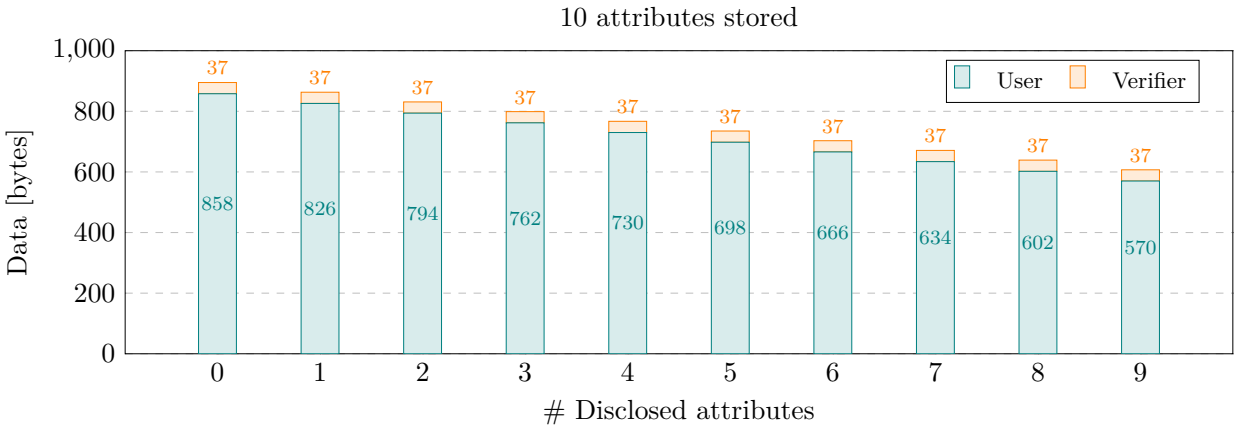


Fig. 4.5: Total amount of transferred data during the authentication phase

Figure 4.5 presents the aggregate amount of data that must be transmitted during the authentication phase between the user and verifier. We consider all possible scenarios where the cryptographic credential stores ten attributes and the disclosure process starts from none to all nine attributes. The tenth attribute, which is the revocation attribute and is confidential to the user, is never disclosed. It is worth noting that the amount of data transmitted by the user is significantly greater than the amount transmitted by the verifier. The verifier only sends the authentication challenge, epoch identifier, and 1 byte of data that defines the disclosed attributes. In contrast, the user sends at least 570 bytes (all attributes are disclosed) and up to 858 bytes (all attributes are concealed). It is important to note that the PEAS implementation does not transmit disclosed attributes from the user's device to the verifier. The verifier is aware of the disclosed attributes and only verifies if the user holds them or not.

5 ZERO-KNOWLEDGE PROOFS FOR SECURE COOPERATIVE INDOOR POSITIONING

In the context of CIPS, safeguarding privacy and security is of paramount importance, given that these systems process and handle sensitive location data that can be exploited to monitor and infringe upon individuals' privacy. However, existing solutions for CIPS do not provide adequate privacy protection, primarily due to their reliance on centralized data sources, which may expose users to data breaches and unauthorized access. Moreover, the positioning information in CIPS is often transmitted in plaintext, further exacerbating the risk of privacy violations. This chapter introduces a novel approach to addressing privacy and security concerns in CIPSs.

5.1 Cryptographic scheme

We propose an innovative decentralized attribute-based authentication protocol. This section discusses the entities involved, as well as the high-level cryptographic design of the `Show` and `Verify` algorithms.

To aid in the clarity and readability of this chapter, we provide a table of symbols used throughout our cryptographic protocol. Table 5.1 defines each symbol and its associated meaning, allowing for a comprehensive understanding of the protocol's components.

Tab. 5.1: Table of symbols

Symbol	Definition
$q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \mathbf{e}, g_1, g_2$	parameters for the selected pairing-friendly elliptic curve.
m_{ID}	attribute with the user's identifier.
m_r	attribute for revocation based on the week and year.
$sk_I, \mathcal{X}_0$	key pair of the issuer (private and public keys).
$i_r, \mathcal{I}_r$	shared key pair among system users (private and public keys).
$\sigma, \sigma_{x_0}, \sigma_{x_r}, \sigma_{x_{ID}}$	cryptographic credential issued to the user.
$\hat{\sigma}, \hat{\sigma}_{x_0}, \hat{\sigma}_{x_r}, \hat{\sigma}_{x_{ID}}$	cryptographic credential randomized by the user.
ρ	random number used to randomize the cryptographic credential.
ρ_κ, ρ_{ID}	random numbers used to compute the protocol commitment and responses.
t_κ	cryptographic commitment computed by the user.
e	challenge used in the cryptographic protocol.
s_κ, s_{ID}	responses obtained during the execution of the cryptographic protocol.
τ	random number used to randomize the transaction identifier.
$\mathcal{R}$	transaction identifier.
ψ	alias of $\hat{\sigma}, \hat{\sigma}_{x_0}, \hat{\sigma}_{x_r}, \hat{\sigma}_{x_{ID}}$.
π	alias of e, s_κ, s_{ID}.
λ	information transmitted or received in plaintext.
ξ	information transmitted or received in ciphertext.
t'_κ	cryptographic commitment reconstructed by the verifier.

The following entities comprise the system model presented in Figure 5.1.

- *Issuer*: issues the personal attribute gathered in a cryptographic credential using the `Issue` algorithm. In our design, the issuer is also responsible for revoking invalid users from the system. This is done to simplify interactions with other entities. The revocation attribute is additionally aggregated to the cryptographic credential. The cryptographic credential is signed by the issuer.

- *User*: acquires the unique credential, including the attributes issued by the issuer, to gain access to the system or service. Using the `Show` algorithm, the user then anonymously demonstrates ownership of the attributes to the verifier. The user may additionally transmit information to the verifier in either plaintext or encrypted format.
- *Verifier*: utilizes the `Verify` algorithm to confirm the user's possession of the attributes. If the ownership of the attributes is successfully validated and the user's access has not been revoked, the verifier accepts the received information. If not, the information will be rejected.

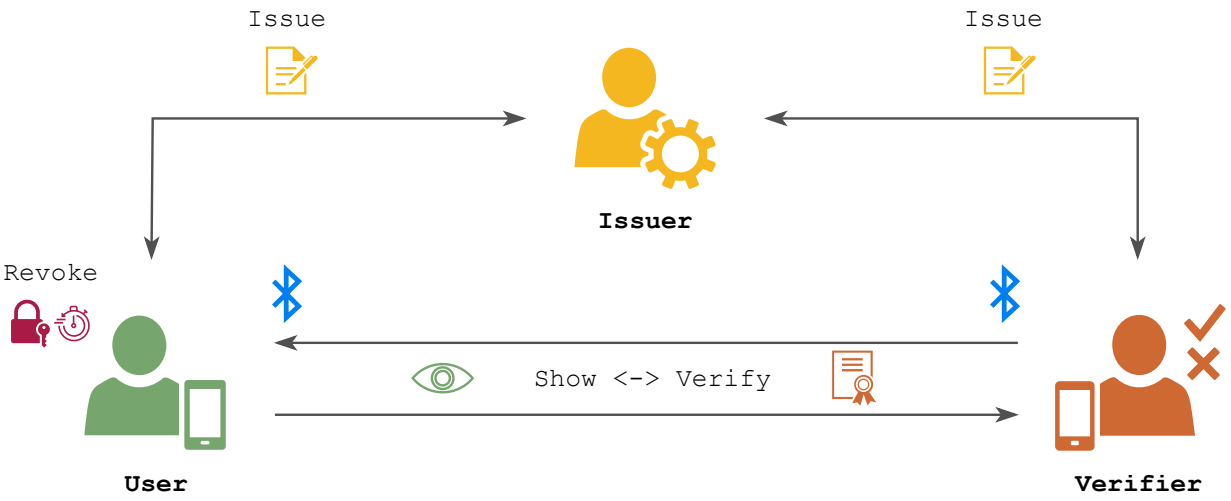


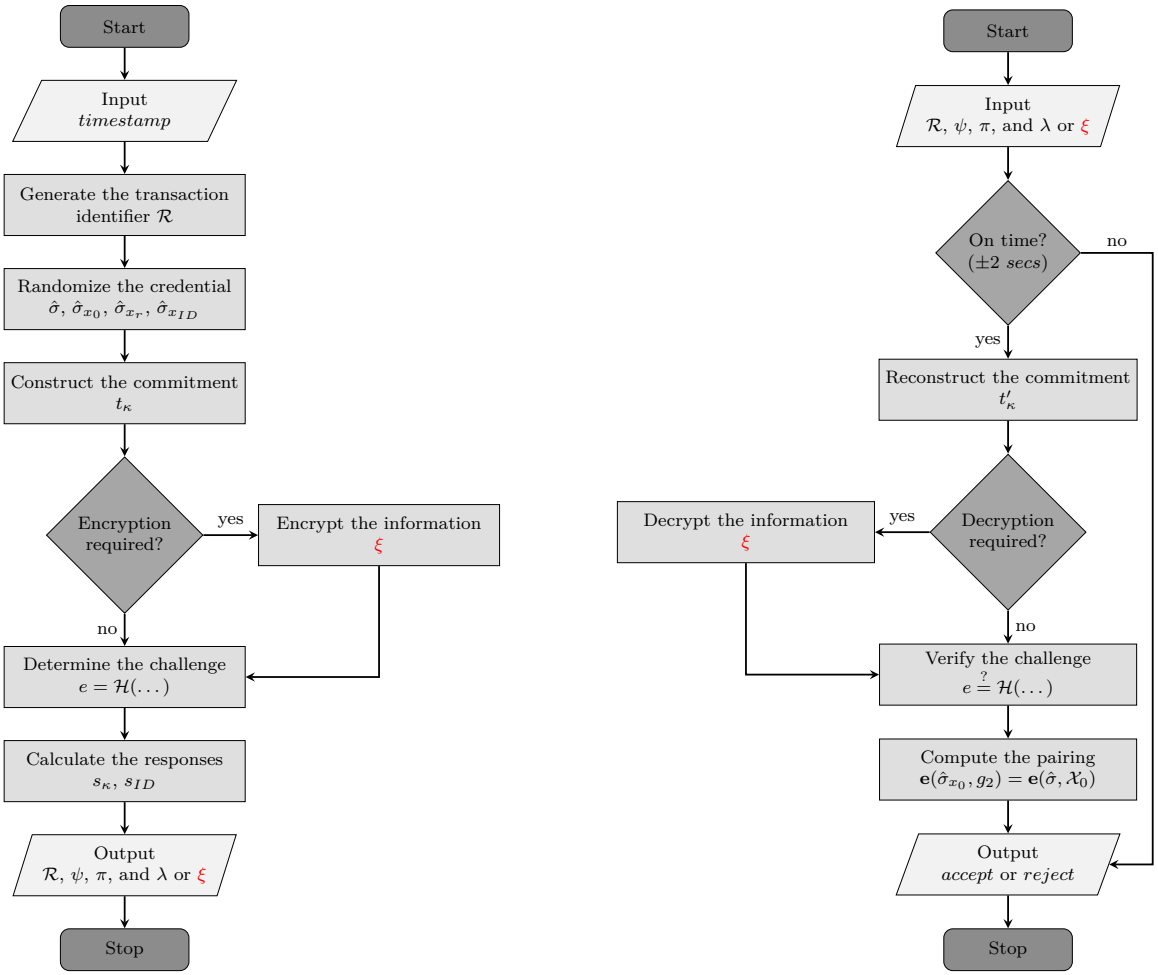
Fig. 5.1: Entities and algorithms constituting the proposed protocol

The flowcharts presented in Figure 5.2 provide a high-level definition of the `Show` and `Verify` algorithms and are a valuable tool for facilitating a comprehensive understanding of the protocols, their underlying mechanisms, and the sequence of steps involved in their execution. By visually presenting the key components of each algorithm and the relationships between them, the flowcharts enable readers to quickly grasp the fundamental concepts of our cryptographic protocol.

5.2 Implementation details

The application was designed for several platforms, including single-board computers, smartphones, smartwatches, and microcontrollers. Such heterogeneous device types are common in IoT ecosystems and are representative of the different use cases that our protocol can support. While smartphones and smartwatches are ideal for obtaining positioning data actively by users, single-board computers and microcontrollers are better suited for industrial settings. They can enable autonomous vehicle positioning or enhance the location sensing capabilities of other devices, further extending the reach and flexibility of our solution.

We divided the implementation of the application into three components: *(i)* the Libre Collaborative Indoor Positioning (LibreCIP) library for core functionality; *(ii)* the *device wrappers* to ensure compatibility with the selected IoT devices; and *(iii)* the *Bluetooth Low Energy* (BLE) integration for transmitting and receiving data using Bluetooth Low Energy.



(a) High-level `Show` algorithm

(b) High-level `Verify` algorithm

Fig. 5.2: High-level definition of the `Show` and `Verify` algorithms

5.2.1 Core library

The application relies on the LibreCIP library, a purpose-built software package implementing the protocol. LibreCIP offers optimized cryptographic functions, including user authentication, data encryption, and compression. It ensures efficient data transmission and storage while prioritizing security and privacy. The library, designed for performance and security across various devices, is written in C and utilizes third-party libraries such as `mcl` [11], `relic-toolkit` [29], `lz4` [30], and the native `crypto` library of RIOT [31]. Device-specific cryptographic backends, `mcl` and `relic-toolkit`, are chosen during compilation. The protocol employs elliptic curve cryptography, specifically the BN254 curve from `mcl` and `relic-toolkit`. Data compression using `lz4` is selected for its compatibility with RIOT and its ability to reduce the original data size. The RIOT `crypto` library is utilized for encryption, decryption, and cryptographic hashes, employing *Advanced Encryption Standard* (AES) in *Cipher Block Chaining* (CBC) mode with 128-bit keys and *Secure Hash Algorithm 3* (SHA-3) with 256-bit digests.

5.2.2 Device wrappers

The wrappers for each device are written in their respective native languages. They consist of the *User Interfaces* (UIs) of the devices as well as the BLE routines and libraries required for communication. The UIs are designed based on the device type. For Android, iOS, Wear OS, and watchOS devices, we created *Graphical User Interfaces* (GUIs). In contrast, we developed CLIs for the Raspberry Pi and PinePhone Pro. We did not create UIs for the microcontrollers or the PineTime. Figure 5.3 depicts the different device wrappers and the technologies used by each to call LibreCIP library functions.

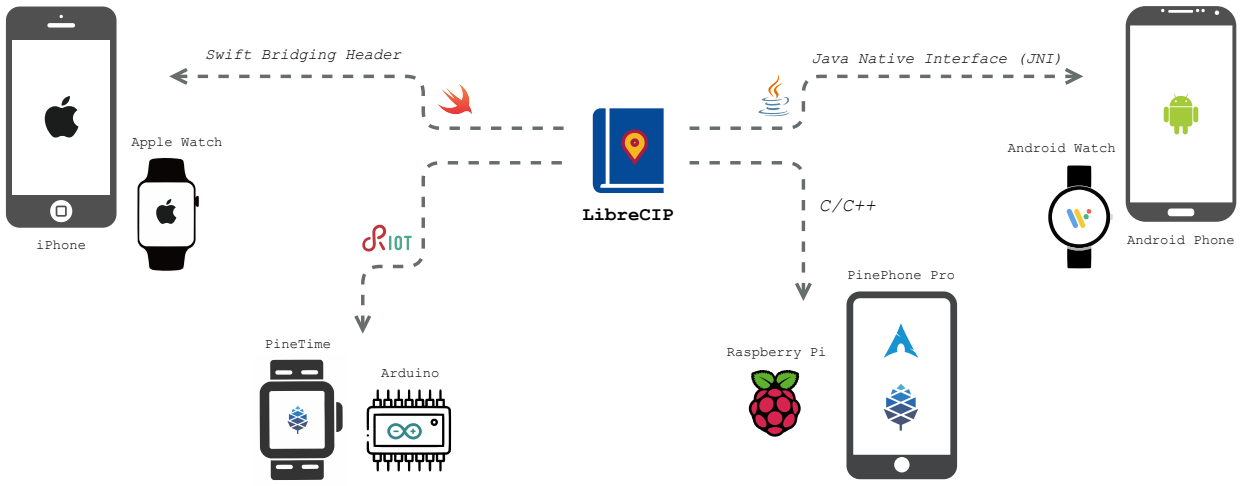


Fig. 5.3: Interoperability between LibreCIP and different devices

5.2.3 Bluetooth Low Energy integration

Our protocol utilizes BLE advertising packets for data transfer. While Bluetooth 4.2 [32] and earlier versions specified a 31-byte payload limit, Bluetooth 5.0 [33] expanded this capacity to 254 bytes with Low-Energy Advertising Extensions. Considering device compatibility issues, we designed a packet format compatible with Bluetooth 4.2. This structure allows data transmission by fragmenting it into smaller packets, facilitating identification of the packet count and sequence for receivers.

5.3 Experimental results

To ascertain the feasibility of our approach and evaluate the performance and speed of the algorithms, we conducted several experiments, benchmarking the entire protocol. Figure 5.4 illustrates the results of the execution in milliseconds and includes BLE communication overhead. Interpreting the study's results requires considering certain limitations. The data collection took place in a controlled laboratory environment, potentially limiting the generalizability of the results to real-world scenarios with crowded transmitting devices. Despite this, the findings suggest the protocol's potential for privacy protection in contemporary CIPSs. Future studies should aim to replicate these results in larger and more diverse environments to comprehensively assess performance.

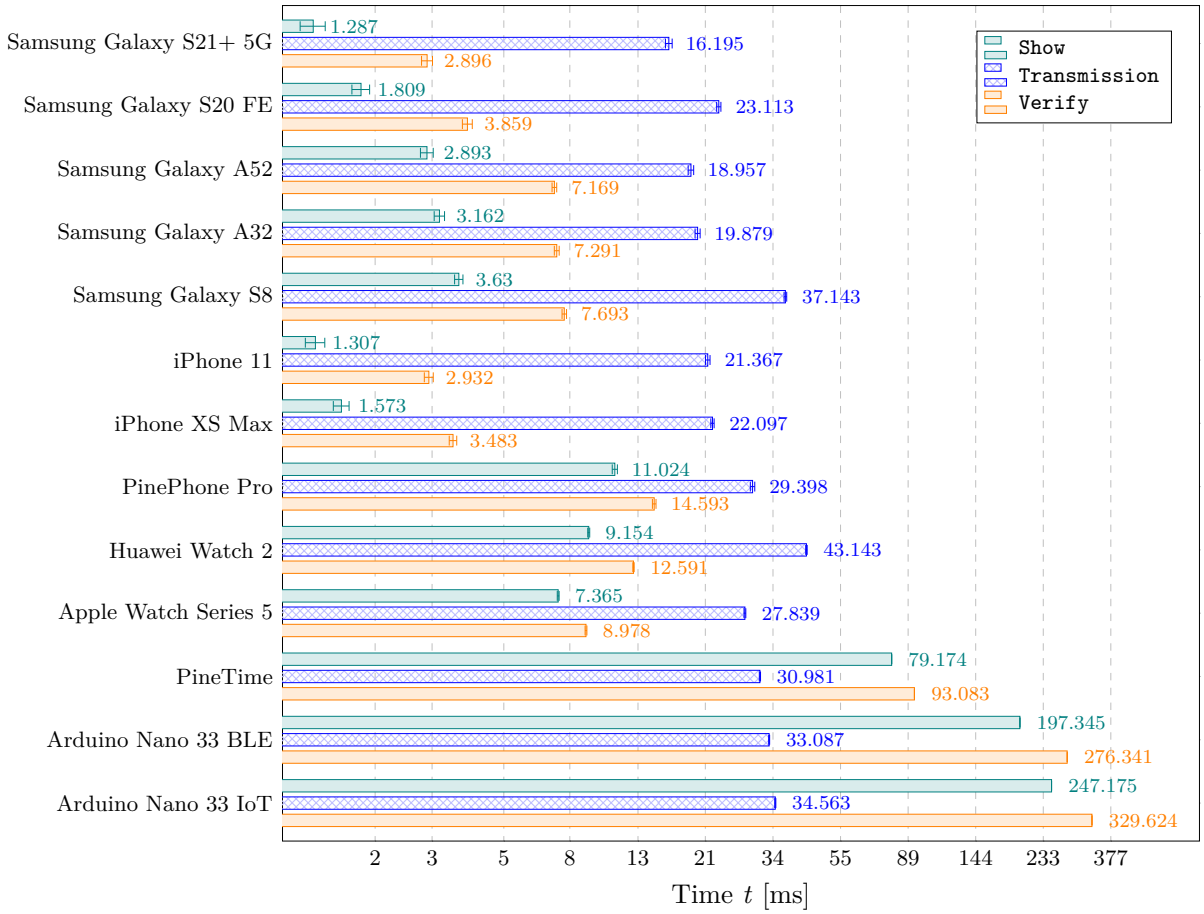


Fig. 5.4: Speed comparison of the `Show` and `Verify` algorithms, and the transmission overhead

The computation times reveal notable differences across devices. High-end smartphones, like the Samsung Galaxy S21+ 5G and iPhone 11, exhibit the fastest processing times (around 1.3 milliseconds for `Show` and 2.9 milliseconds for `Verify`). In contrast, microcontroller boards like Arduino Nano 33 BLE and Arduino Nano 33 IoT, designed for IoT applications, demonstrate considerably slower computation times (ranging from 197.345 to 329.624 milliseconds). This discrepancy arises from the limited processing power and memory resources of microcontroller boards compared to more advanced smartphone hardware. All widely used smartphones and smartwatches maintain `Show` and `Verify` processing times below 15 milliseconds. The fastest device executes `Show` in 1.287 milliseconds, while the slowest takes 9.154 milliseconds. For the `Verify` algorithm, the fastest execution is 2.896 milliseconds, and the slowest is 12.591 milliseconds. The Raspberry Pi 4 benchmarks, with execution times in the order of microseconds, are omitted due to their negligible impact. Notably, the variability in measurements, represented by error bars, is larger for smartphones, attributed to the complexity of their operating systems introducing more variability compared to microcontrollers. Devices like the Huawei Watch 2, Samsung Galaxy S8, PineTime, and Arduino Nano exhibit slower transmission times, exceeding 30 milliseconds. This can be attributed to the utilization of outdated Bluetooth technology. During development, energy profiling tools in Android Studio and Xcode were employed to assess energy consumption, indicating no adverse effects on energy utilization.

6 CONCLUSION

The research questions presented in Chapter 1 are addressed and answered in this section.

- *How can anonymous credential schemes be adapted to support user revocation while maintaining privacy?*
  Chapter 2 presents a novel attribute-based authentication scheme with pseudonymous-based revocation. This design maintains user privacy while allowing for efficient revocation in case of malicious behavior. The protocol has been successfully implemented and tested on MULTOS smart cards, demonstrating its effectiveness even on resource-constrained devices.
- *What strategies can be employed to enable attribute-based authentication protocols on smart cards with limited support for elliptic curve cryptography?*
  Chapter 3 introduces techniques to adapt mathematical operations to the limitations of the Java Card API. This has enabled the implementation of the attribute-based authentication protocol from Chapter 2 on these cards. While not yet ready for real-world deployment, these results open up new avenues for future research.
- *What are the usability challenges associated with using anonymous credentials in various applications, and how can they be addressed?*
  The inability to visualize requested attributes or provide consent for disclosure hinders the seamless integration of anonymous credentials in real-world environments. Chapter 4 introduces a platform that addresses the usability challenges of attribute-based authentication on smart cards. It utilizes the cryptographic core from Chapter 2 and operates on mobile devices and smartwatches, bridging the gap between attribute-based authentication and practical application.
- *How can anonymous credential schemes be integrated into collaborative indoor positioning systems to enhance privacy and security?*
  Collaborative indoor positioning systems face inherent vulnerabilities due to their interaction with unknown devices and the lack of cryptographic protocols to safeguard user privacy and security. Chapter 5 presents a cryptographic scheme for collaborative indoor positioning systems. It provides anonymity through randomized identities, encrypted data transmission, and automatic user revocation, enhancing privacy and security while fostering trust among users.
- *How can anonymous credential schemes be implemented in resource-constrained environments, such as* IoT devices?
  The cryptographic scheme from Chapter 5 has been implemented on Arduino boards, demonstrating its applicability in IoT environments. This implementation utilized specialized libraries tailored for IoT environments and optimized operating systems designed for low-power devices.
- *Are attribute-based authentication schemes suitable for ensuring user authenticity in dynamic wearable architectures?*
  The thesis affirms the suitability of attribute-based authentication schemes for wearable architectures. Extensive implementations on various wearable devices have demonstrated their practical viability and efficiency, ensuring user authenticity in dynamic environments.

BIBLIOGRAPHY

[1] Jan Hajny and Lukas Malina. Unlinkable Attribute-based Credentials with Practical Revocation on Smart Cards. In *Smart Card Research and Advanced Applications*, pages 62–76, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg. doi: 10.1007/978-3-642-37288-9_5.

[2] Jan Camenisch, Stephan Krenn, Anja Lehmann, Gert Læssøe Mikkelsen, Gregory Neven, and Michael Østergaard Pedersen. Formal Treatment of Privacy-Enhancing Credential Systems. In *Selected Areas in Cryptography – SAC 2015*, pages 3–24, Cham, 2016. Springer International Publishing. doi: 10.1007/978-3-319-31301-6_1.

[3] Jan Camenisch, Manu Drijvers, and Jan Hajny. Scalable Revocation Scheme for Anonymous Credentials Based on n-Times Unlinkable Proofs. In *Proceedings of the 2016 ACM on Workshop on Privacy in the Electronic Society*, pages 123–133, Vienna Austria, October 2016. ACM. doi: 10.1145/2994620.2994625.

[4] David Chaum. Security without Identification: Transaction Systems to Make Big Brother Obsolete. *Communications of the ACM*, 28(10):1030–1044, October 1985. doi: 10.1145/4372.4373.

[5] Stefan A. Brands. *Rethinking Public Key Infrastructures and Digital Certificates: Building in Privacy*. MIT Press, Cambridge, Mass, 2000. ISBN 978-0-262-02491-4.

[6] Jan Camenisch and Anna Lysyanskaya. An Efficient System for Non-transferable Anonymous Credentials with Optional Anonymity Revocation. In *Advances in Cryptology – EUROCRYPT 2001*, pages 93–118, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg. doi: 10.1007/3-540-44987-6_7.

[7] Wojciech Mostowski and Pim Vullers. Efficient U-Prove Implementation for Anonymous Credentials on Smart Cards. In *Security and Privacy in Communication Networks*, pages 243–260, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg. doi: 10.1007/978-3-642-31909-9_14.

[8] Pim Vullers and Gergely Alpár. Efficient Selective Disclosure on Smart Cards using Idemix. In *Policies and Research in Identity Management*, pages 53–67, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg. doi: 10.1007/978-3-642-37282-7_5.

[9] Jan Camenisch, Manu Drijvers, Petr Dzurenda, and Jan Hajny. Fast Keyed-Verification Anonymous Credentials on Standard Smart Cards. In *ICT Systems Security and Privacy Protection*, pages 286–298, Cham, 2019. Springer International Publishing. doi: 10.1007/978-3-030-22312-0_20.

[10] Mouna S. Chebli, Heba Mohammad, and Khalifa Al Amer. An Overview of Wireless Indoor Positioning Systems: Techniques, Security, and Countermeasures. In *Internet and Distributed Computing Systems*, pages 223–233, Cham, 2019. Springer International Publishing. doi: 10.1007/978-3-030-34914-1_22.

[11] Mitsunari Shigeo. MCL Library. `https://github.com/herumi/mcl`, 2023.

[12] OpenSSL Project Authors. OpenSSL: Cryptography and SSL/TLS Toolkit. https://www.openssl.org, 2023.

[13] MULTOS Limited. MULTOS SmartDeck Developer's Reference Manual. Technical Report v3.4.0.0, MULTOS, 2021. URL https://multos.com/wp-content/uploads/2021/06/MDG.pdf.

[14] MULTOS Limited. MDG: MULTOS Developer's Guide. Technical Report MAO-DOC-TEC-005 v1.43, MULTOS, 2021. URL https://multos.com/wp-content/uploads/2021/06/MDG.pdf.

[15] MULTOS Limited. MDRM: MULTOS Developer's Reference Manual. Technical Report MAO-DOC-TEC-006 v1.59, MULTOS, 2022. URL https://multos.com/wp-content/uploads/2022/11/MDRM.pdf.

[16] MULTOS Limited. C-API V2: MULTOS Standard C-API. Technical Report MAO-DOC-TEC-016 v2.3, MULTOS, 2021. URL https://multos.com/wp-content/uploads/2021/06/CAPIv2.pdf.

[17] Petr Svenda. Java Card Algorithm Test: Detailed Analysis of Cryptographic Smart Cards Running with Java Card Platform. https://github.com/crocs-muni/JCAlgTest, 2013.

[18] Paulo S. L. M. Barreto and Michael Naehrig. Pairing-Friendly Elliptic Curves of Prime Order. In *Selected Areas in Cryptography*, pages 319–331, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg. doi: 10.1007/11693383_22.

[19] Hendrik Tews. OV-Chip 2.0 Hacker's Guide. https://www.sos.cs.ru.nl/ovchip/, 2010.

[20] Jan Hajny, Petr Dzurenda, Raúl Casanova-Marqués, and Lukas Malina. Privacy ABCs: Now Ready for Your Wallets! In *2021 IEEE International Conference on Pervasive Computing and Communications Workshops and other Affiliated Events (PerCom Workshops)*, pages 686–691, Kassel, Germany, March 2021. IEEE. doi: 10.1109/PerComWorkshops51409.2021.9431139.

[21] Ludovic Rousseau. Middleware to Access a Smart Card using SCard API (PC/SC). https://pcsclite.apdu.fr, 2023.

[22] Ludovic Rousseau. Generic USB CCID and ICCD Driver for Unix Systems. https://ccid.apdu.fr, 2023.

[23] BlueZ Project. BlueZ: Official Linux Bluetooth Protocol Stack. https://www.bluez.org, 2022.

[24] tspspi. ANSI C99 JSON Serializer / Deserializer. https://github.com/tspspi/libcjson, 2019.

[25] Andy Green. Flexible and Lightweight Pure C Library for Implementing Modern Network Protocols. https://github.com/warmcat/libwebsockets, 2023.

[26] Jan Hajny, Petr Dzurenda, Sara Ricci, Lukas Malina, and Kamil Vrba. Performance Analysis of Pairing-based Elliptic Curve Cryptography on Constrained Devices. In *2018 10th International Congress on Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT)*, pages 1–5, Moscow, Russia, November 2018. IEEE. doi: 10.1109/ICUMT.2018.8631228.

[27] OpenJS Foundation. Open-source and Cross-platform JavaScript Runtime Environment. https://nodejs.org, 2023.

[28] Evan You. The Progressive JavaScript Framework. https://vuejs.org, 2023.

[29] Diego de Freitas Aranha, Conrado Porto Lopes Gouvêa, Tobias Markmann, Riad S. Wahby, and K. Liao. RELIC is an Efficient LIbrary for Cryptography. https://github.com/relic-toolkit/relic, 2023.

[30] Yann Collet. Lossless Compression Algorithm. https://lz4.github.io/lz4/, 2023.

[31] RIOT Community. RIOT OS: The Friendly Operating System for the Internet of Things. https://riot-os.org/, 2023.

[32] Bluetooth Special Interest Group. Bluetooth Core Specification v4.2. Specification, Bluetooth Special Interest Group, 2014. URL https://www.bluetooth.com/specifications/specs/core-specification-4-2/.

[33] Bluetooth Special Interest Group. Bluetooth Core Specification v5.0. Specification, Bluetooth Special Interest Group, 2016. URL https://www.bluetooth.com/specifications/specs/core-specification-5-0/.

Curriculum Vitae

Raúl Casanova-Marqués

E-mail	casanova@vut.cz raul.casanova@uji.es
WWW	https://orcid.org/0000-0001-6653-867X

EDUCATION

2019–2023	**Joint Ph.D.** at Brno University of Technology, Czech Republic, and Universitat Jaume I, Spain. Doctoral thesis topic: *Privacy-Enhancing Technologies and Privacy-Enhancing Cryptography for Wearables*.
2017–2019	**Master's degree in Security of Information and Communication Technologies** at Universitat Oberta de Catalunya, Spain. Thesis topic: *Replay Attacks in Access to Restricted Zones*.
2017–2018	**Indra Master's degree in Cybersecurity** at U-tad Centro Universitario de Tecnología y Arte Digital, Spain. Thesis topic: *Bluetooth Pentesting with Metasploit*.
2013–2017	**Bachelor's degree in Computer Science and Engineering** at Universitat Rovira i Virgili, Spain. Thesis topic: *Secure Bluetooth: Secure Implementation?*

PROFESSIONAL CAREER

2019–2022	Doctoral researcher at Brno University of Technology, Czech Republic.
2017–2018	Security researcher at CRISES Research Group, Universitat Rovira i Virgili, Spain.

RESEARCH ACTIVITIES

Specialization	Research and development in the field of advanced cryptographic protocols for attribute-based authentication, privacy protection mechanisms for users in electronic systems, and secure algorithms on wearable devices.
Summary of publications	2 scientific journals 8 international conferences

ABSTRACT

In response to escalating privacy concerns and the need for secure digital communication, cryptographic mechanisms have been developed to ensure impervious information exchange. However, traditional cryptographic approaches are inadequate in dynamic and resource-constrained environments, such as wearable devices. This thesis investigates attribute-based credential schemes, offering fine-grained access control based on user-specific attributes. Specifically, it assesses the effectiveness and scalability of attribute-based anonymous credential schemes within dynamic wearable device architectures. The study focuses on enhancing these schemes by incorporating user revocation while preserving privacy. Additionally, the research develops methods for attribute-based authentication protocols on smart cards with limited elliptic curve cryptography support and addresses usability challenges. Furthermore, the thesis explores the integration of anonymous authentication in collaborative indoor positioning systems to ensure privacy and security. It also delves into implementing attribute-based authentication in resource-constrained environments, including Internet of Things devices, and evaluating its feasibility in dynamic wearable device architectures.