sensors

MDPI

Article

PriTKT: A Blockchain-Enhanced Privacy-Preserving Electronic Ticket System for IoT Devices

Yonghua Zhan [1], Feng Yuan [2], Rui Shi [3,*], Guozhen Shi [3] and Chen Dong [1]

[1] College of Computer and Data Science, Fuzhou University, Fuzhou 350108, China; 190310007@fzu.edu.cn (Y.Z.); dongchen@fzu.edu.cn (C.D.)
[2] Institute 706, The Second Academy China Aerospace Science & Industry Corp, Beijing 100854, China; fyuan1234@aliyun.com
[3] Beijing Electronic Science and Technology Institute, Beijing 100070, China; sgz1974@163.com
* Correspondence: ruishi_mail@126.com

Abstract: Electronic tickets (e-tickets) are gradually being adopted as a substitute for paper-based tickets to bring convenience to customers, corporations, and governments. However, their adoption faces a number of practical challenges, such as flexibility, privacy, secure storage, and inability to deploy on IoT devices such as smartphones. These concerns motivate the current research on e-ticket systems, which seeks to ensure the unforgeability and authenticity of e-tickets while simultaneously protecting user privacy. Many existing schemes cannot fully satisfy all these requirements. To improve on the current state-of-the-art solutions, this paper constructs a blockchain-enhanced privacy-preserving e-ticket system for IoT devices, dubbed PriTKT, which is based on blockchain, structure-preserving signatures (SPS), unlinkable redactable signatures (URS), and zero-knowledge proofs (ZKP). It supports flexible policy-based ticket purchasing and ensures user unlinkability. According to the data minimization and revealing principle of GDPR, PriTKT empowers users to selectively disclose subsets of (necessary) attributes to sellers as long as the disclosed attributes satisfy ticket purchasing policies. In addition, benefiting from the decentralization and immutability of blockchain, effective detection and efficient tracing of double spending of e-tickets are supported in PriTKT. Considering the impracticality of existing e-tickets schemes with burdensome ZKPs, we replace them with URS/SPS or efficient ZKP to significantly improve the efficiency of ticket issuing and make it suitable for use on smartphones.

Keywords: electronic tickets; privacy-preserving; blockchain; IoT; double-spending detection

Citation: Zhan, Y.; Yuan, F.; Shi, R.; Shi, G.; Dong, C. PriTKT: A Blockchain-Enhanced Privacy-Preserving Electronic Ticket System for IoT Devices. *Sensors* **2024**, *24*, 496. https://doi.org/10.3390/s24020496

Academic Editor: Rongxing Lu

Received: 14 December 2023
Revised: 6 January 2024
Accepted: 7 January 2024
Published: 13 January 2024

1. Introduction

E-ticketing has emerged as a popular method of ticket entry, processing, and marketing for companies in the airline [1], railway [2,3], and other transportation and entertainment industries. The integration of blockchain technology in e-ticketing has further enhanced the security and transparency of transactions. Blockchain [4–7] ensures that each transaction is recorded in a decentralized and tamper-proof ledger, reducing the risk of fraud and ensuring the authenticity of e-tickets. Unlike traditional paper-based tickets, e-tickets offer two major advantages. First, paper-based tickets are disposable and require significant resources for their production, leading to an increased negative impact on the environment. By contrast, e-tickets serve as an eco-friendly replacement by minimizing paper waste produced from ticketing activities, thereby aligning with the principles of the Paris Agreement [8]. Second, e-tickets provide customers with the flexibility to reserve, issue, and refund their tickets online anytime and anywhere, eliminating the need for in-person lineups. This convenience and ease for customers is further heightened by the integration of IoT technology. Customers can easily access their e-tickets through dedicated mobile applications, making the entire ticketing process more streamlined and accessible. Seamless integration with IoT devices not only enhances the overall customer experience, it reflects

the evolving nature of ticketing systems in the digital age. As a result, e-tickets, powered by blockchain and IoT technology, offer significant advantages in terms of convenience, environmental sustainability, and security.

Despite the increasing popularity of e-tickets, they face numerous practical challenges, particularly with regards to privacy. In e-ticket systems, user data such as name, identity number, purchase date, and other personal attributes may be collected and misused. Thus, it is crucial to minimize the collection of personal data in line with the recently introduced General Data Protection Regulations (GDPR) [9]. In this regard, various privacy-protecting e-ticket systems have been proposed which use randomizable signatures with efficient proofs [10–14], pseudonyms [15,16], and anonymous credentials [13,17,18]. However, some of these systems lack formal proof of security and cannot guarantee the integrity of the token [12,13,16], while others are inefficient in their invoicing operations, making them impossible to deploy on IoT devices [18].

An essential feature of any e-ticket system is the ability to issue tickets based on user attributes. Attribute-based e-ticket systems have potential in various real-world applications. For example, they can enable students, soldiers, and individuals with disabilities to purchase tickets at discounted rates without revealing sensitive information such as a student ID number, unit number, or health conditions. In an attribute-based e-ticketing system, a user credential used to purchase e-tickets is parameterized with a vector of the user's attributes, such as date of birth, affiliation, or occupation. During ticket purchase, users can prove that they possess a credential meeting a given attribute policy, such as age or disability, without revealing any additional information beyond the satisfaction of the attribute policy. Unfortunately, most existing e-ticketing systems [12–14,16] do not support attribute-based ticket issuance protocols.

Credentials issued by governmental bodies, schools, or companies typically include only the basic attributes of the user, such as name, gender, educational background, address, and position. However, in many cases additional attributes can be defined as well. For instance, in platform configuration-based access control services, attributes may refer not only to the user's personal data but to hardware platform and configuration information. Moreover, one user may hold several roles, each with unique credentials issued by a single entity. The Privacy by Design Foundation's Anonymous Credentials (IRMA) [19] offers a broad range of real-world attributes, such as diplomas, passports, cards, and even membership IDs for online services, which are relevant to governing bodies and various businesses. Such scenarios imply that a user may have hundreds or more potential attributes. Integrating IoT devices into the e-ticketing system allows for the inclusion of additional attributes related to the user while presenting an opportunity to streamline the issuance process. IoT devices capabilities, such as bio-metric authentication and secure storage, can enhance the security and efficiency of handling a wide array of attributes. This ensures that the e-ticketing system remains practical and user-friendly even in scenarios with a large number of potential attributes. While Han et al. [18] designed the first attribute-based e-ticketing system, their ticket issuance algorithm's computational cost and communication overhead increases linearly with the number of user attributes, making their scheme impractical to deploy on IoT devices when users have hundreds of attributes.

Paper-based tickets can be made unique, and are easily distinguishable from copies; however, distinguishing original e-tickets from their copied versions is challenging, making it necessary to prevent and detect double spending in e-ticketing systems. This challenge is particularly pertinent in the digital realm, where the ease of replication poses a unique set of security concerns. The integration of blockchain technology offers a promising solution to the issue of double spending in e-ticketing. By utilizing a decentralized and tamper-proof ledger, blockchain ensures the immutability and transparency of transaction records. Each e-ticket transaction can be securely recorded on the blockchain, creating a verifiable and unforgeable trail. This prevents unauthorized duplication of e-tickets while allowing for efficient tracking and identification of any double-spending attempts. Moreover, in the context of IoT technology, blockchain can be seamlessly integrated to

enhance the security of e-tickets. IoT devices can serve as secure digital wallets, storing and managing e-tickets in a tamper-resistant environment. The combination of blockchain and IoT devices ensures that the integrity of e-tickets is maintained, reducing the risk of duplication and unauthorized use. In the event of a double-spending attempt, the decentralized nature of blockchain enables the identification and tracing of responsible users without compromising the anonymity of honest users. The transparency of the blockchain allows for swift and accurate resolution of security incidents, thereby bolstering the overall trust and reliability of the e-ticketing system.

Based on the above requirements and intuition, we propose PriTKT, a blockchain-enhanced privacy-preserving e-ticketing system for IoT devices that supports ticket issuance based on user attributes. PriTKT significantly reduces the computational cost and communication overhead, making it suitable for use on IoT devices, and utilizes blockchain [4] to achieve effective malicious user tracking.

Our Contributions

This work makes the following contributions:

• **Attribute-based ticketing**: We propose an e-ticket system that supports attribute-based ticketing. This system securely and seamlessly integrates attribute-based anonymous credentials [20], unlinkable redactable signatures (URS) [20], structure-preserving signatures (SPS) [21], Pedersen commitment, and zero-knowledge signature of knowledge (ZKSoK) [22] in its design. In our e-ticket system, a trusted party generates a ticket purchasing policy set during system initialization. The ticket purchasing policy set regulates how tickets are issued. User privacy is preserved, as users may disclose a subset of their attributes as long as they satisfy a ticket purchasing policy in the ticket purchasing policy set for the tickets they purchase.

• **Efficient ticketing for IoT devices**: Our system improves the efficiency of the ticketing algorithm to ensure that it can be efficiently executed on an IoT device (such as smartphone) while presenting a privacy-preserving e-ticketing solution that does not require expensive Zero-Knowledge Proofs (ZKP) to validate user credentials during ticket purchase. Instead, we use the URS signature, which reduces the computational overhead in the ticket-issuing algorithm from $O(N)$ to $O(N-K)$ and the communication overhead from $O(N)$ to $O(K)$, where N and K represent the number of user attributes and exposed attributes, respectively. Compared to the closest scheme in [18], our method improves both computational and communication performance. We implemented PriTKT on a smartphone under AES-100-bit security and compared it with the state-of-the-art scheme from [18], finding that our issuing algorithm and showing algorithm were 250% and 240% more efficient, respectively.

• **Blockchain-enhanced double-spending detection and efficient trace**: We implement double-spending detection and efficient user tracking. The verifier uploads each valid token into the blockchain. The decentralization and immutability of the blockchain [4] ensures the correctness of the token storage. Every ticket must disclose its unique identity when it is shown, and ZKSoK is used to guarantee the correctness of its identity disclosure. With the ticket identity, the verifier can traverse the tokens stored in the blockchain and quickly detect any double-spending. Using the "Schnorr trick" [23], the verifier can quickly compute the public key of any double-spending user for efficient tracing.

• **Unlinkability and framing resistance**: Our solution ensures the unlinkability of tickets. Using the unlinkability of URS signatures, we prove that no tickets of the same user can be linked. Our solution ensures the framing resistance of tickets. Framing resistance ensures that corrupt sellers cannot falsely accuse any honest users of double-spending.

In addition, we offer formal security definitions for e-tickets that can be reduced to known complexity assumptions or the security of established cryptography primitives. The performance of PriTKT was measured on a smartphone.

2. Related Work

2.1. Electronic Tickets on IoT Devices

Mut-Puigserver et al. [24] conducted a study on the functional and security requirements of electronic tickets. These requirements include offline verification, expiration dates, reduced size, portability, flexibility, unlinkability, unforgeability, and non-overspending. They explored various types of e-tickets, such as single-use, multi-use, transferable, and non-transferable. Our research specifically focuses on analyzing the unlinkability, unforgeability, and non-overspending features of single-use and non-transferable electronic tickets leveraging the capabilities of IoT devices and the security of blockchain technology.

Previous studies by Heydt-Benjamin et al. [12] employed electronic cash, anonymous credentials [25], and proxy re-encryption [26] to enhance privacy in public transit systems that use electronic tickets. The study proposed a theoretical framework to examine the security of payments and the privacy of information in transit systems. They asserted that their system was capable of safeguarding user privacy but was not compatible with attribute-based ticketing; however, the paper did not provide any formal proof to support the system's security. In contrast, the integration of IoT technology and blockchain in modern e-ticketing systems, such as PriTKT, can offer improved security measures, including attribute-based ticketing, while maintaining user privacy. Following Rupp et al. [14], Jager et al. [27] presented Black-Box Accumulation (BBA) to establish cryptography payment systems. Later, BBA was enhanced to BBA+ [28] by Hartung et al. and Black-Box Wallet (BBW) by Hoffmann et al. [29]. Nonetheless, these payment schemes [27–29] differ from attribute-based e-ticketing systems, as they pertain to electronic payments.

Vives-Guasch et al. [16] introduced an e-ticketing system that considers user privacy requirements as well as security requirements that include exculpability and re-usability. Additionally, by utilizing lightweight cryptography and mobile phones equipped with Near-Field Communication (NFC) technology, their system accommodates the computational limitations of users. Regrettably, however, their system does not support attribute-based ticketing. The integration of IoT devices (smartphones) in e-ticketing systems, as demonstrated by PriTKT, can contribute to enhanced features such as de-anonymization prevention and secure ticket non-transferability, while blockchain ensures the security and transparency of transactions.

The system of Milutinovic et al. [13] depends on certified tokens that are impossible to relate, and on different cryptographic primitives such as commitment schemes [30], partially blind signatures [31], and anonymous credentials [32] to tackle privacy concerns. However, their system does not provide de-anonymization capabilities after double spending or support ticket non-transferability in the way that PriTKT does. The integration of IoT devices capabilities in PriTKT enhances security measures and provides a practical solution for these challenges.

Han et al. [18] introduced attribute-based credentials derived from the Boneh–Boyen signature (BBS) [33] and efficient set membership proof and range proofs [34] to issue attribute credentials and tickets. However, due to the use of a signature with the NIZK protocol, their system's NIZK proof computation increases linearly with the number of attributes in the ticket-issuing algorithm, limiting its practical use. The integration of lightweight and efficient IoT technology, along with the security of blockchains, can potentially address these computational constraints, making attribute-based e-ticketing more practical and user-friendly.

2.2. Blockchain-Enhanced Double-Spending Detection

Double-spending problems exist not only in e-ticket system [12–14,16,18], but in electronic payment systems [23,27–29] and blockchain cryptocurrency schemes [5–7]. Blockchain technology plays a crucial role in mitigating the double-spending issue by providing a decentralized and transparent ledger that ensures the integrity and uniqueness of transactions.

The most direct way to prevent double-spending is to bind each spending operation with a unique identifier that cannot be forged to ensure that that verifiers can detect double-spending based on the unique identifier. Rupp et al. [14] used a digital signature as the identifier. Sasson et al. [5] utilized hash-based commitments to compute identifiers. Androulaki et al. [6] utilized verifiable random functions to generate identifiers. Han et al. [18] and Sun et al. [7] used elements randomly mapped to elliptic curve groups as identifiers. Jager et al. [27] and Bobolz et al. [23] directly used a random number as the identifier. Although there are many ways to generate identifiers, these schemes all need to compute a zero-knowledge proof in the spending protocol to prove the correctness of the disclosed identifier.

Certain schemes [5–7,12,13,16,27] simply terminate the spending protocol when a double-spend is detected, while others [14,18,23,28,29] support further tracing and identification of malicious users. The method used by Han et al. [18] was to hide users' identity in ElGamal ciphertext, then use a verifier to recover their identities through the spending algorithms. The scheme in [14,23,28,29] achieved double-spending tracing through the "Schnorr trick", which is more efficient than scheme in [18]. The integration of blockchains into these schemes can enhance the overall security and transparency of their double-spending prevention and tracing mechanisms.

2.3. Attribute-Based Credentials

Anonymized attribute-based credentials that enable selective disclosure of attributes can be obtained in a manner akin to the use of randomizable signatures. Every user receives a signature on (commitments to) a list of attributes from a centralized authority. When the credential is presented, the user randomizes the signature (ensuring that the resulting signature and the published signature cannot be linked) and proves the correspondence of this signature to the revealed and hidden attributes in zero-knowledge proofs [35–41]. From a privacy perspective, this solution is perfectly satisfactory; however, it is not very efficient, as the user's unshared attributes impose more cost than the user's revealed attributes.

Fuchsbauer and Hanser [42,43] randomized both the signature and the signed message (which is a set commitment to the user's attributes) with a structure-preserving signature on equivalence classes (SPS-EQ), then used subset opening of the set commitments to selectively disclose attributes. In this way, they avoided the need to perform costly ZKP over the hidden attributes. Unfortunately, while attributes can be disclosed in this solution once they are signed, it cannot be proven that they are hidden while satisfying certain relations.

Camenisch et al. [44] presented a novel unlinkable redactable signature (URS) that allows part of the signed message to be redacted while proving that the signature is valid for the disclosed attributes. Unfortunately, their scheme can only be instantiated by Groth–Sahai proofs [45], and it is difficult to compete with the most effective solution in practice. Sanders followed the URS approach from [44] and constructed a flexible redactable signature scheme that achieves unlinkability at almost zero cost. Unlike the methods in [43,44], the URS presented by Sanders can prove complex relationships between attributes and does not rely on zero-knowledge proofs for partial verification.

3. Preliminaries

3.1. Bilinear Pairing

Suppose that $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$ are groups of prime-order p with generators $g \in \mathbb{G}_1$ and $\widetilde{g} \in G_2$. A mapping $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is a bilinear map if it satisfies three requisite properties. (1) *Bilinearity*: for all $g \in \mathbb{G}_1, \widetilde{g} \in \mathbb{G}_2$, and $a, b \in \mathbb{Z}_p$, we have $e(g^a, \widetilde{g}^b) = e(g, \widetilde{g})^{ab}$; (2) *Nondegeneracy*: $e(g, \widetilde{g}) \neq 1$; and (3) *Computability*: e is an efficiently computable function. The PriTKT scheme is based on the Type III bilinear pairing [46], which by definition does not admit an efficiently computable homomorphism between $\mathbb{G}_1$ and $\mathbb{G}_2$.

3.2. Computational Assumptions

Discrete Logarithm (DL) Assumption. Let $\mathbb{G}$ be a prime-order cyclic group and let g be a generator of $\mathbb{G}$. Given $(g, g^x) \in \mathbb{G}^2$, the DL assumption holds in $\mathbb{G}$ if no efficient algorithm or adversary can practically compute x with non-negligible probability.

The DDH Assumption. Let $\mathbb{G}$ be a prime-order cyclic group and let g be a generator of $\mathbb{G}$. Given $(g, g^x, g^y, g^z) \in \mathbb{G}^4$, the DDH assumption holds in G if no efficient adversary can distinguish the product $z = x \cdot y$ from a random element in $\mathbb{G}$.

3.3. Unlinkable Redactable Signatures

The Unlinkable Redactable Signature (URS) [20] consists of a tuple (Setup, KeyGen, Sign, Derive, Verify) of probabilistic polynomial-time (PPT) algorithms. The URS is used to issue credentials for a user's attributes.

• Setup(1^λ): given a security parameter λ, this algorithm generates the public parameters $pp = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g, \widetilde{g}, p, e)$.

• KeyGen(n): given an input integer n, this algorithm selects $(x, y_1, \ldots, y_n) \xleftarrow{R} \mathbb{Z}_p^{n+1}$ and computes $X = g^x$, $Y_i = g^{y_i}$, $\widetilde{Y}_i = \widetilde{g}^{y_i}$, $1 \leqslant i \leqslant n$, and $Z_{i,j} = g^{y_i \cdot y_j}$, $1 \leqslant i \neq j \leqslant n$. Then, the secret key is $sk = (x, y_1, \ldots, y_n)$ and the public key is $pk = (X, \{Y_i, \widetilde{Y}_i\}_{i=1}^n, \{Z_{i,j}\}_{1 \leqslant i \neq j \leqslant n})$.

• Sign$(sk, \mathbb{A} = \{a_i\}_{i=1}^n)$: to sign n messages $a_1, \ldots, a_n$ ($a_i \in \mathbb{Z}_p, 1 \leqslant i \leqslant n$), this algorithm selects $\widetilde{\sigma}_1 \xleftarrow{R} \mathbb{G}_2$ and computes $\widetilde{\sigma}_2 = \sigma_1^{x + \sum_{i=1}^n y_i \cdot a_i}$. It sets $\sigma_1 = 1_{\mathbb{G}_1}$ and $\sigma_2 = 1_{\mathbb{G}_1}$, then outputs $\sigma = (\sigma_1, \sigma_2, \widetilde{\sigma}_1, \widetilde{\sigma}_2)$.

• Derive$(pk, \sigma, \mathbb{A}, \mathcal{D})$: given an input signature $\sigma = (\sigma_1, \sigma_2, \widetilde{\sigma}_1, \widetilde{\sigma}_2)$ on $\{a_i\}_{i=1}^n$, the public key pk, and a subset $\mathcal{D} \subseteq [n]$, this algorithm selects $(r, t) \xleftarrow{R} \mathbb{Z}_p^2$ and computes $\widetilde{\sigma}_1' = \widetilde{\sigma}_1^r$, $\widetilde{\sigma}_2' = \widetilde{\sigma}_2^r (\widetilde{\sigma}_1')^t$, $\sigma_1' = g^t \prod_{i \in [n] \setminus \mathcal{D}} Y_i^{a_i}$, and $\sigma_2' = (\prod_{i \in \mathcal{D}} Y_i)^t \prod_{i \in \mathcal{D}, j \in [n] \setminus \mathcal{D}} Z_{i,j}^{a_j}$. If $\mathcal{D} = [n]$, then $[n] \setminus \mathcal{D} = \emptyset$ and $\sigma_1' = g^t$ and $\sigma_2' = (\prod_{i \in [n]} Y_i)^t$. The algorithm returns the derived signature $\sigma' = (\sigma_1', \sigma_2', \widetilde{\sigma}_1', \widetilde{\sigma}_2')$ on $\mathcal{D} \subseteq [n]$.

• Verify$(pk, \sigma, \mathbb{D} = \{a_i\}_{i \in \mathcal{D}})$: a signature$(\sigma_1, \sigma_2, \widetilde{\sigma}_1, \widetilde{\sigma}_2)$ on $\mathcal{D} \subseteq [n]$ is valid if the following equations hold: $e(X\sigma_1 \cdot \prod_{i \in \mathcal{D}} \widetilde{Y}_i^{a_i}, \widetilde{\sigma}_1) = e(g, \widetilde{g}_2)$ and $e(\sigma_1, \prod_{i \in \mathcal{D}} \widetilde{Y}_i) = e(\sigma_2, \widetilde{g})$, in which case the algorithm returns 1; otherwise, it returns 0.

The URS [20] is a redactable signature scheme with forgeability and unlinkability in the generic group model.

3.4. Structure-Preserving Signatures

The Structure-Preserving Signatures (SPS) proposed by Groth [21] consist of a tuple (Setup, KeyGen, Sign, Verify) of PPT algorithms that sets messages in $\mathbb{G}_2$. This scheme is used to issue credentials for a seller's public key.

• Setup(1^λ): given a security parameter 1^λ, this algorithm generates public parameters $pp = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g, \widetilde{g}, p, e)$.

• KeyGen(n): this algorithm generates a secret key $sk = (x, y_1, \ldots, y_n)$ and a public key $pk = (h; \widetilde{X}; \{\widetilde{Y}_i\}_{i=1}^{n-1})$ when given an input integer n. It selects $(x, y_1, \ldots, y_{n-1}) \xleftarrow{R} \mathbb{Z}_p^n$ and a random generator $h \xleftarrow{R} \mathbb{G}_1$, and computes $\widetilde{X} = \widetilde{g}^x$, $\widetilde{Y}_i = \widetilde{g}^{y_i}$, and $1 \leqslant i \leqslant n-1$.

• Sign$(sk, \{m_i\}_{i=1}^n)$: to sign n messages $(m_1, \ldots, m_n)$, where m_i is an element of group $\mathbb{G}_1$ for $1 \leq i \leq n$, the signer randomly selects $r \xleftarrow{R} \mathbb{Z}_p$ and computes $\widetilde{\delta}_1 = \widetilde{g}^{r^{-1}}$, $\delta_2 = (h \cdot g^x)^r$, and $\delta_3 = (h \cdot m_n \cdot \prod_{i=1}^{n-1} m_i^{y_i})^r$. Next, the signer outputs the signature $\delta = (\widetilde{\delta}_1, \delta_2, \delta_3)$.

• Verify$(pk, \delta, \{m_i\}_{i=1}^n)$: for verification of the signature $\delta = (\widetilde{\delta}_1, \delta_2, \delta_3)$ associated with messages $\{m_i\}_{i=1}^n$, two pairing equations need to be evaluated: $e(\delta_2, \widetilde{\delta}_1) = e(h, \widetilde{g}) e(g, \widetilde{X})$ and $e(\delta_3, \widetilde{\delta}_1) = e(h, \widetilde{X}) e(m_n, \widetilde{g}) \prod_{i=1}^{n-1} e(m_i, \widetilde{Y}_i)$. If at least one of these equations is not satisfied, the algorithm outputs 0; otherwise, it outputs 1.

3.5. Zero-Knowledge Signature of Knowledge

The ZKSoK protocol [22] for an NP-relation $\mathcal{R}$ is composed of the following algorithms to ensure knowledge of the language $L_{\mathcal{R}} = \{y : \exists x, (x, y) \in \mathcal{R}\}$.

• Gen(1^{λ}), which returns a public parameter pp when provided a security parameter λ;

• Sign(m, x, y), which returns a ZKSoK Π = ZKSoK$\{x : (x, y) \in \mathcal{R}\}$ when given a message m and a relation $(x, y) \in \mathcal{R}$;

• Verify(m, Π, y), which, when given a message m, ZKSoK Π, and statement y, returns 1 if Π is valid and 0 otherwise.

4. System and Security Model

4.1. System Model

Electronic ticketing systems involve five entities, as presented in Figure 1: the Central Authority (CA), Ticket Seller (S), User (U) with IoT device (such as smartphones), Blockchain ($\mathcal{BC}$), and Ticket Verifier (V). This section describes each individual function in turn.

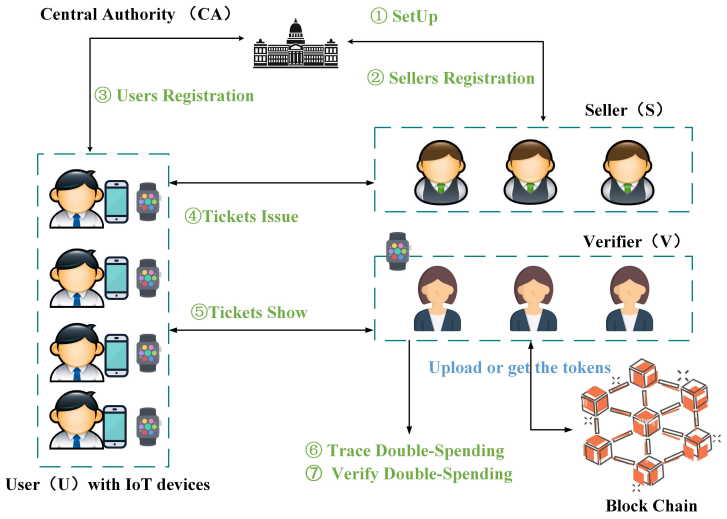

Figure 1. E-ticket system.

• CA bears the responsibility of being the globally trusted entity that establishes the electronic ticketing system (step ①). Here, the CA creates policy set for purchasing tickets. In addition, the CA must offer registration services to ticket sellers (step ②) and users (step ③).

• S is an independent ticket seller who is required to register with CA in order to participate in the ticketing system (step ②). During the ticket purchasing process, S is responsible for verifying each user's credentials and issuing tickets based on the ticket purchasing policy set forth by CA (step ④).

• U has a designated set of attributes stored in their IoT device that must be registered with CA in order to participate in the ticketing system (step ③). When purchasing a ticket from S, U is only required to disclose a subset of their attributes that satisfy the ticket purchasing policy selected by S (step ④). Subsequently, U presents the ticket to V in an anonymous manner (step ⑤).

• The blockchain $\mathcal{BC}$ acts as an immutable database in the electronic ticket system; using the decentralized features of the blockchain, the verifier can upload all valid token information. Any participant can then query the presentation tokens.

• V offers ticket verification services to all users, and has the ability to detect instances of double-spending. Upon receiving verifiable data from users, V verifies the authenticity of their tickets (step ⑤) and uploads the token to the blockchain while simultaneously detecting any attempts at double-spending. If a double-spending ticket is identified,

V traces the corresponding user's public key (step ⑥) and generates double-spending blaming information.

• Any participant can download the token information from the blockchain and verify the correctness of the double-spending tracing based on the double-spending blaming information generated by V (step ⑦).

4.2. Formal Definition

The notations used in the system are listed in Table 1, and the algorithms are defined formally below.

Table 1. Summary of notation.

Notation	Description
$1^\lambda/\epsilon(\lambda)$	security number/negligible function
$[N]$	set $\{1, 2, \cdots, N\}$
$\mathcal{D}$	subset of $[N]$
$x \xleftarrow{R} \mathbb{S}$	x is randomly selected from the set $\mathbb{S}$
CA	central authority
S/U/V	ticket seller/user/verifier
$\mathcal{BC}$	the blockchain
$\mathbb{P}$	ticket purchasing policy set
$\mathbb{A}/\mathbb{D}$	attributes/disclosed attributes set
N/K	number of user attributes/disclosed attributes
pp/msk	public parameters/master key of the system
usk/upk	secret/public key of U
ssk/spk	secret/public key of S
$cred_u/cred_s$	credential of U/S
tkt	ticket of U
VP_{tkt}	valid period for the ticket of U
$dsid/dstrace$	double-spending identity/trace infomation
$dsblame$	double-spending blaming information
H	collision resistant hash function
$\perp$	failed identifier

• Setup$(1^\lambda) \rightarrow (msk, pp, \mathbb{P})$: the algorithm is executed by CA, which takes a security parameter 1^λ as input and returns a secret key msk, system parameter pp, and set of ticket purchasing policies $\mathbb{P}$.

• SKeyGen$(pp) \rightarrow (ssk, spk)$: the algorithm is executed by S, which takes a system parameter pp as input and returns a private key ssk and corresponding public key spk as output.

• UKeyGen$(pp) \rightarrow (usk, upk)$: the algorithm is executed by U, which takes a system parameter pp as input and returns a private key usk and corresponding public key upk as output.

• SReg$(\text{S}(ssk, spk, pp) \leftrightarrow \text{CA}(msk, spk, pp)) \rightarrow (cred_s, \perp)$: to obtain a credential for a given public key spk, S and CA engage in an interactive algorithm. S provides (ssk, spk) and pp as inputs, while CA provides msk, spk, and pp as inputs. If the algorithm executes successfully, a credential $cred_s$ is issued to S. If the algorithm fails to execute, $\perp$ is returned as the output.

• UReg$(\text{U}(usk, upk, \mathbb{A}, pp) \leftrightarrow \text{CA}(msk, upk, pp)) \rightarrow (cred_u, \perp)$: U and CA engage in an interactive algorithm to obtain a credential for the attribute set $\mathbb{A} = \{a_i\}_{i=1}^N$. U provides (usk, upk), $\mathbb{A}$, and pp as inputs, while CA provides msk, upk, and pp as inputs. If the algorithm executes successfully, a credential $cred_u$ is issued to U. If the algorithm fails to execute, $\perp$ is returned as output.

• Issue$(\text{U}(usk, cred_u, \mathcal{D}, \mathbb{A}, pp) \leftrightarrow \text{S}(ssk, cred_s, pp, \mathbb{P})) \rightarrow ((tkt, VP_{tkt}), b)$: the algorithm for obtaining an anonymous ticket involves an interaction between U and S. To initiate this process, U provides usk, $cred_u$, $\mathcal{D}$, $\mathbb{A}$, and pp as inputs, while S provides ssk, $cred_s$, pp, and $\mathbb{P}$ as inputs. Here, we define $\mathbb{D} = \{a_i\}_{i \in \mathcal{D}}, \mathcal{D} \subseteq [N]$ for convenience. U

must then provide proof that a subset $\mathbb{D}$ of their attributes $\mathbb{A}$ has been certified and that $\mathbb{D} \in \mathbb{P}$ in accordance with the ticket purchasing policy selected by S. The final output of the algorithm consists of a bit b indicating the validity of $cred_u$ as well as an anonymous ticket tkt and its associated valid period VP_{tkt} for U.

• Show(U($usk, tkt, VP_{tkt}, spk, pp$) $\leftrightarrow$ V(spk, pp)) $\rightarrow$ (($dsid, dstrace$), b): this algorithm relies on the interaction between U and V. U accepts usk, tkt, VP_{tkt}, spk, and pp as inputs, while V only takes spk and pp. When the algorithm is completed, V uploades the token to $\mathcal{BC}$ and produces a bit b with a value of either 1 or 0 (1 indicates that tkt is valid and that VP_{tkt} falls within the validity period, whereas 0 indicates the opposite) along with a double-spending identity $dsid$ and double-spending trace information $dstrace$.

• DSTrace($dsid, dstrace, \overline{dsid}, \overline{dstrace}, spk, pp$) $\rightarrow$ (($dsblame$, upk'), $\perp$): V can execute this algorithm under the condition that the double-spending identity of two tickets is identical ($dsid = \overline{dsid}$), which can be correctly verified by traversing the token information in $\mathcal{BC}$. By taking two trace information $dstrace$ and $\overline{dstrace}$ as input, V can deterministically generate two outputs: the unique public key upk' of the double-spending user, and the double-spending blaming information ($dsblame$ if successful or $\perp$ if unsuccessful).

• VerifyDS($dsblame, upk'$) $\rightarrow b$: any party can operate this algorithm. With $dsblame$ and upk' as inputs, it generates an output of $b = 1$ if $dsblame$ proves that upk' has double-spending and $b = 0$ otherwise.

Note that instead of embedding the user ID as an attribute in the user credential, in our e-ticketing system we use each user's public key as that user's unique identifier. This is because in our system each user has a unique public key. When the DSTrace algorithm is executed, the user's ID can always be determined in the real world according to the public key issued by V and the credential information maintained by CA.

Definition 1. *The correctness of the e-ticket system depends on two conditions: (1) the tickets produced by the Issue algorithm must be verifiable by the Show algorithm, and (2) the DSTrace algorithm must be able to track users who attempt double-spending behavior. The formal definition of correctness can be found in Supplementary Material Section S4.*

4.3. Security Model

We assume that the central authority in the system, CA, is fully trustworthy. The ticket seller, S, is honest enough to issue tickets to users according to a specific purchase policy but may attempt to obtain users' undisclosed attributes and real IDs. The user, U, may forge tickets, attempt to spend them twice, or transfer them illegally. The ticket verifier is honest in verifying tickets and detecting users who attempt to spend an electronic ticket more than once, but may attempt to obtain the undisclosed attributes and real IDs of honest users.

The e-ticket system should satisfy the following security requirements: unforgeability of user credentials and tickets, unlinkability of honest users, framing resistance of honest users, and non-transferability of tickets. The security model is defined following the works in [20,23,43,47], and we provide formal definitions of security requirements.

The following global variables and oracles are used in all security definitions.

Global Variables.

$\mathcal{HU}$: the set of honest users' identities; $\mathcal{CU}$: the set of corrupt users identities; ($\mathcal{UPK}$, $\mathcal{USK}$): the list of users' public and secret keys; ($\mathcal{CRED}_U$, $\mathcal{ATTR}_U$, $\mathcal{CID}_U$): the list of user credentials, user attributes sets, and user identities; ($\mathcal{TKT}$, $\mathcal{VP}_{tkt}$, $\mathcal{TID}_U$, $\mathcal{TID}_S$): the list of tickets, valid period of tickets, user identities, and seller identities; $\mathcal{HS}$: the set of honest sellers' identities; $\mathcal{CS}$: the set of corrupt sellers' identities; ($\mathcal{SPK}$, $\mathcal{SSK}$): the list of sellers' public and secret keys; ($\mathcal{CRED}_S$, $\mathcal{CID}_S$): the list of sellers' credentials and identities.

Oracles.

• $\mathcal{O}_{HU}(i)$: an oracle that can be used to generate keys for an honest user i. If $i \in \mathcal{HU}$ or $i \in \mathcal{CU}$, then it returns $\perp$; otherwise, it creates honest user i by running ($\mathcal{USK}[i]$,$\mathcal{UPK}[i]$) $\leftarrow$ UKeyGen($\cdot$). It adds i to $\mathcal{HU}$ and returns $\mathcal{UPK}[i]$.

• $\mathcal{O}_{HS}(j)$: an oracle that can be used to generate keys for an honest seller j. If $j \in \mathcal{HS}$ or $j \in \mathcal{CS}$, then it returns $\perp$; otherwise, it creates honest seller j by running $(\mathcal{SSK}[j], \mathcal{SPK}[j]) \leftarrow \text{SKeyGen}(\cdot)$. It adds j to $\mathcal{HS}$ and returns $\mathcal{SPK}[j]$.

• $\mathcal{O}_{CU}(i, upk)$: an oracle that can (optionally) be used to corrupt an honest user i with the public key upk. If $i \in \mathcal{CU}$, then it returns $\perp$. If $i \in \mathcal{HU}$, then it removes i from $\mathcal{HU}$ and adds i to $\mathcal{CU}$; it searches u to fulfill the condition $\mathcal{CID}_U[u] = i$ and returns $(\mathcal{USK}[i], \mathcal{ATTR}_U[u], \mathcal{CRED}_U[u])$. Otherwise, it adds i to $\mathcal{CU}$ and sets $\mathcal{UPK}[i] \leftarrow upk$.

• $\mathcal{O}_{CS}(j, spk)$: an oracle that can (optionally) be used to corrupt an honest seller j with the public key spk. If $j \in \mathcal{CS}$, then it returns $\perp$. If $j \in \mathcal{HS}$, then it removes j from $\mathcal{HS}$, adds j to $\mathcal{CS}$, and returns $\mathcal{SSK}[j]$. Otherwise, it adds j to $\mathcal{CS}$ and sets $\mathcal{SPK}[j] \leftarrow spk$.

• $\mathcal{O}_{UReg}(i, \mathbb{A})$: an oracle that can be used to issue a credential for an honest user i with the attribute set $\mathbb{A}$. If $i \notin \mathcal{HU}$, it returns $\perp$. Otherwise, it issues a credential to i by running $\text{UReg}(\text{U}(\mathcal{USK}[i], \mathcal{UPK}[i], \mathbb{A}, pp) \leftrightarrow \text{CA}(msk, \mathcal{UPK}[i], pp)) \rightarrow cred_u$ and appends $(cred_u, \mathbb{A}, i)$ to $(\mathcal{CRED}_U, \mathcal{ATTR}_U, \mathcal{CID}_U)$.

• $\mathcal{O}_{SReg}(j)$: an oracle that can be used to issue a credential for an honest seller j. If $j \notin \mathcal{HS}$, it returns $\perp$; otherwise, it issues a credential to j by running $\text{SReg}(\text{S}(msk, \mathcal{SPK}[j], \mathcal{SPK}[j], pp) \leftrightarrow \text{CA}(\mathcal{SSK}[j], pp)) \rightarrow cred_s$ and appends $(cred_s, j)$ to $(\mathcal{CRED}_S, \mathcal{CID}_S)$.

• $\mathcal{O}_{Iss}(i, j, \mathcal{D})$: an oracle that can be used to play an honest seller j issuing a ticket to an honest user i with the disclosed attributes set $\mathbb{D} = \{a_i\}_{i \in \mathcal{D}}$. If $i \notin \mathcal{HU}$ or $j \notin \mathcal{HS}$, it returns $\perp$; otherwise, it searches u and v to fulfill conditions $\mathcal{CID}_U[u] = i$ and $\mathcal{CID}_S[v] = j$, then j issues a ticket to i by running $\text{Issue}(\text{U}(\mathcal{USK}[i], \mathcal{CRED}_U[u], \mathcal{D}, \mathcal{ATTR}_U[u], pp) \leftrightarrow \text{S}(\mathcal{SSK}[j], \mathcal{CRED}_S[v], pp, \mathbb{P})) \rightarrow ((tkt, VP_{tkt}), b)$. If $b = 0$, it returns $\perp$; otherwise, it appends (tkt, VP_{tkt}, i, j) to $(\mathcal{TKT}, \mathcal{VP}_{tkt}, \mathcal{TID}_U, \mathcal{TID}_S)$.

• $\mathcal{O}_{Iss_U}(i, j, \mathcal{D})$: an oracle that can be used to play a curious seller j issuing a ticket to an honest user i with the disclosed attributes set $\mathbb{D}$. If $i \notin \mathcal{HU}$ or $j \notin \mathcal{CS}$, it returns $\perp$; otherwise, it searches u to fulfill the condition $\mathcal{CID}_U[u] = i$ and runs $\text{Issue}(\text{U}(\mathcal{USK}[i], \mathcal{CRED}_U[u], \mathcal{D}, \mathcal{ATTR}_U[u], pp) \leftrightarrow \mathcal{A}(j, \cdot)) \rightarrow ((tkt, VP_{tkt}), b)$, where the seller's side is executed by the adversary. If $b = 0$, it returns $\perp$; otherwise, it appends (tkt, VP_{tkt}, i, j) to $(\mathcal{TKT}, \mathcal{VP}_{tkt}, \mathcal{TID}_U, \mathcal{TID}_S)$.

• $\mathcal{O}_{Iss_S}(j, i, \mathcal{D})$: an oracle that can be used to play an honest seller j issuing a ticket to a malicious user i with the disclosed attributes set $\mathbb{D}$. If $i \notin \mathcal{CU}$ or $j \notin \mathcal{HS}$, it returns $\perp$; otherwise, it searches v to fulfill the condition $\mathcal{CID}_S[v] = j$ and runs $\text{Issue}(\mathcal{A}(i, \cdot) \leftrightarrow \text{S}(\mathcal{SSK}[j], \mathcal{CRED}_S[v], pp, \mathbb{P})) \rightarrow ((tkt, VP_{tkt}), b)$, where the user's side is executed by the adversary. If $b = 0$, it returns $\perp$; otherwise, it appends (tkt, VP_{tkt}, i, j) to $(\mathcal{TKT}, \mathcal{VP}_{tkt}, \mathcal{TID}_U, \mathcal{TID}_S)$.

• $\mathcal{O}_{Shw}(i, j)$: an oracle that can be used to play a malicious verifier verifying a ticket for an honest seller user i. If $i \notin \mathcal{HU}$, it returns $\perp$; otherwise, it searches u to fulfill conditions $\mathcal{TID}_U[u] = i$ and $\mathcal{TID}_S[u] = j$ and runs $\text{Show}(\text{U}(\mathcal{USK}[i], \mathcal{TKT}[u], \mathcal{VP}_{tkt}[u], pp) \leftrightarrow \mathcal{A}(\mathcal{SPK}[j], \cdot)) \rightarrow ((dsid, dstrace), b)$, where the verifier's side is executed by the adversary. If $b = 0$, it returns $\perp$; otherwise, it returns $(dsid, dstrace)$.

We define the security model of the PriTKT system as follows.

Unforgeability. Unforgeability can protect honest sellers and verifiers from malicious users. It guarantees that users cannot forge credentials in the Issue algorithm or tickets in the Show algorithm. An adversary can interact with the CA and honest seller oracles as a corrupted user to model this property. The adversary wins when they can forge a credential or a ticket of either an honest or an unregistered user in the Issue or Show algorithm. Unforgeability is defined by dividing it into credential unforgeability and ticket unforgeability.

Definition 2. *Experiment $Exp^{uf_{cred}}$ in Pse. 1 defines the unforgeability of credentials. The users' credentials are considered unforgeable if any PPT adversary $\mathcal{A}$ can access the oracle*

$\mathcal{O} = \{\mathcal{O}_{HU}(i), \mathcal{O}_{CU}(i), \mathcal{O}_{HS}(j), \mathcal{O}_{UReg}(i, \mathbb{A}), \mathcal{O}_{SReg}(j), \mathcal{O}_{Iss}(i, j, \mathcal{D}), \mathcal{O}_{Iss_S}(j, i, \mathcal{D}), \mathcal{O}_{Shw}(i, j)\}$. *A negligible function* $\epsilon(\lambda)$ *exists such that*

$$Adv^{uf_{cred}} = |\Pr\left[Exp^{uf_{cred}}(\mathcal{A}, \lambda)\right] = 1| \leqslant \epsilon(\lambda).$$

$Exp^{uf_{cred}}(\mathcal{A}, \lambda)$:
1. $(msk, pp, \mathbb{P}) \leftarrow \text{Setup}(1^\lambda)$;
2. $(i^*, j^*, st) \leftarrow \mathcal{A}^{\mathcal{O}}(pp, \mathbb{P})$;
3. $\text{Issue}(\mathcal{A}(st, i^*) \leftrightarrow \text{S}(\mathcal{SPK}[j^*], pp)) \rightarrow (\cdot, b)$;
4. If ($b = 0 \vee i^* \in \mathcal{CU} \vee j^* \notin \mathcal{HS}$), return 0;
5. Return 1.

Pseudocode 1. Unforgeability of Credentials

Definition 3. *The experiment* $Exp^{uf_{tkt}}$ *in Pse. 2 defines the unforgeability of tickets. In order for users' tickets to be considered unforgeable, it must be the case that any PPT adversary, denoted as* $\mathcal{A}$*, who has the oracle* $\mathcal{O}$ *that contains* $\mathcal{O}_{HU}(i)$*,* $\mathcal{O}_{CU}(i)$*,* $\mathcal{O}_{HS}(j)$*,* $\mathcal{O}_{UReg}(i, \mathbb{A})$*,* $\mathcal{O}_{SReg}(j)$*,* $\mathcal{O}_{Iss}(i, j, \mathcal{D})$*,* $\mathcal{O}_{Iss_S}(j, i, \mathcal{D})$*, and* $\mathcal{O}_{Shw}(i, j)$*, will have a negligible function* $\epsilon(\lambda)$ *such that*

$$Adv^{uf_{tkt}} = |\Pr\left[Exp^{uf_{tkt}}(\mathcal{A}, \lambda)\right] = 1| \leqslant \epsilon(\lambda).$$

$Exp^{uf_{tkt}}(\mathcal{A}, \lambda)$:
1. $(msk, pp, \mathbb{P}) \leftarrow \text{Setup}(1^\lambda)$;
2. $(i^*, j^*, st) \leftarrow \mathcal{A}^{\mathcal{O}}(pp, \mathbb{P})$;
3. $\text{Show}(\mathcal{A}(st, i^*) \leftrightarrow \text{V}(\mathcal{SPK}[j^*], pp)) \rightarrow (\cdot, b)$;
4. If ($b = 0 \vee i^* \in \mathcal{CU} \wedge j^* \notin \mathcal{HS}$), return 0;
5. Return 1.

Pseudocode 2. Unforgeability of Tickets

Unlinkability. Unlinkability is necessary to safeguard honest users from inquisitive sellers and verifiers. Its primary purpose is to prevent an adversary from controlling a corrupt seller (who also verifies) from associating a specific credential with a particular user in the Issue algorithm or from tying a specific ticket to a particular user in the Show algorithm. To formally define this property, we allow the adversary to interact with the CA and honest user oracles playing the part of the corrupted seller. In the subsequent challenge phase, the adversary can invoke extra interactions and attempt to determine which user they are interacting with. If they correctly guess the user, the adversary wins. To define this property, we divide unlinkability into unlinkability of credentials and unlinkability of tickets.

Definition 4. *The experiment* $Exp^{ano_{cred}-b}$ *depicted in Pse. 3 defines the unlinkability of credentials. We consider users' credentials to be unlinkable if, for any PPT adversary,* $\mathcal{A}$ *with access to the oracle* $\mathcal{O} = \{\mathcal{O}_{HU}(i), \mathcal{O}_{HS}(j), \mathcal{O}_{CS}(j), \mathcal{O}_{UReg}(i, \mathbb{A}), \mathcal{O}_{SReg}(j), \mathcal{O}_{Iss}(i, j, \mathcal{D}), \mathcal{O}_{Iss_U}(i, j, \mathcal{D}), \mathcal{O}_{Shw}(i, j)\}$ *there exists a negligible function* $\epsilon(\lambda)$ *such that*

$$Adv^{ano_{cred}} = |\Pr\left[Exp^{ano_{cred}-1}(\mathcal{A}, \lambda) = 1\right] - \Pr\left[Exp^{ano_{cred}-0}(\mathcal{A}, \lambda) = 1\right]| \leqslant \epsilon(\lambda).$$

$Exp^{ano_{cred}-b}(\mathcal{A}, \lambda)$:
1. $(msk, pp, \mathbb{P}) \leftarrow \text{Setup}(1^\lambda)$;
2. $(i_0, i_1, \mathcal{D}^*, j^*, st) \leftarrow \mathcal{A}^{\mathcal{O}}(pp, \mathbb{P})$;
3. If ($i_0 \notin \mathcal{HU} \vee i_1 \notin \mathcal{HU} \vee j^* \notin \mathcal{CS}$), return $\perp$;
4. $b^* \leftarrow \mathcal{A}^{\mathcal{O}_{Iss_U}(i_b, j^*, \mathcal{D}^*)}(pp, \mathbb{P})$;
5. Return b^*.

Pseudocode 3. Anonymity of Credentials

Definition 5. *Experiment $Exp^{ano_{tkt}-b}$ in Pse. 4 defines the unlinkability of tickets. Users' tickets are unlinkable if any PPT adversary $\mathcal{A}$ with the oracle $\mathcal{O} = \{\mathcal{O}_{HU}(i), \mathcal{O}_{HS}(j), \mathcal{O}_{CS}(j), \mathcal{O}_{UReg}(i, \mathbb{A}), \mathcal{O}_{SReg}(j), \mathcal{O}_{Iss}(i, j, \mathcal{D}), \mathcal{O}_{Iss_U}(i, j, \mathcal{D}), \mathcal{O}_{Shw}(i, j)\}$ cannot associate a specific ticket with a particular user. There exists a negligible function $\epsilon(\lambda)$ such that*

$$Adv^{ano_{tkt}} = |\Pr\left[Exp^{ano_{tkt}-1}(\mathcal{A}, \lambda) = 1\right] - \Pr\left[Exp^{ano_{tkt}-0}(\mathcal{A}, \lambda) = 1\right]| \leqslant \epsilon(\lambda).$$

$Exp^{ano-b}(\mathcal{A}, \lambda)$:
1. $(msk, pp, \mathbb{P}) \leftarrow \text{Setup}(1^\lambda)$;
2. $(i_0, i_1, j^*, st) \leftarrow \mathcal{A}^{\mathcal{O}}(pp, \mathbb{P})$;
3. If ($i_0 \notin \mathcal{HU} \vee i_1 \notin \mathcal{HU} \vee j^* \notin \mathcal{CS}$), return $\perp$;
4. $b^* \leftarrow \mathcal{A}^{\mathcal{O}_{Shw}(i_b, j^*)}(pp, \mathbb{P})$;
5. Return b^*.

Pseudocode 4. Anonymity of Tickets

Framing Resistance. In defining framing resistance, we follow the idea of Bobolz et al. [23]. Framing resistance ensures that corrupt sellers cannot falsely accuse honest users of double spending. To model this property, we allow the adversary to interact with the CA and the oracles of the honest users in the role of a corrupt seller. Honest users will not engage in double spending. The adversary outputs a double-spending blame information *dsblame*. If there is an honest user's public key upk' that verifies *dablame* and upk' with VerifyDS, then the adversary wins.

Definition 6. *The framing resistance property is defined by the experiment Exp^{fr} in Pse. 5. The e-ticket system is framing-resistant if, for any PPT adversary $\mathcal{A}$ having access to the oracle $\mathcal{O} = \{\mathcal{O}_{HU}(i), \mathcal{O}_{HS}(j), \mathcal{O}_{CS}(j), \mathcal{O}_{UReg}(i, \mathbb{A}), \mathcal{O}_{SReg}(j), \mathcal{O}_{Iss}(i, j, \mathcal{D}), \mathcal{O}_{Iss_U}(i, j, \mathcal{D}), \mathcal{O}_{Shw}(i, j)\}$, there is a negligible function $\epsilon(\lambda)$ such that*

$$Adv^{fr} = |\Pr\left[Exp^{fr}(\mathcal{A}, \lambda) = 1\right]| \leqslant \epsilon(\lambda).$$

$Exp^{fr}(\mathcal{A}, \lambda)$:
1. $(msk, pp, \mathbb{P}) \leftarrow \text{Setup}(1^\lambda)$;
2. $(i^*, dsblame^*, st) \leftarrow \mathcal{A}^{\mathcal{O}}(pp, \mathbb{P})$;
3. If ($i^* \in \mathcal{HU} \wedge \text{VerifyDS}(dsblame^*, \mathcal{UPK}[i^*]) = 1$), return 1;
4. Return 0.

Pseudocode 5. Framing Resistance

$$\mathrm{SReg}(\mathrm{S}(ssk, spk, pp) \leftrightarrow \mathrm{CA}(msk, spk, pp)) \rightarrow cred_s$$

Seller: S		**Central Authority**: CA
1: compute $\Pi_1 = \mathrm{ZKSoK}\{(a, b_0, b_1, \ldots,$	$\xrightarrow{spk, \Pi_1}$	
2: $b_3) : A = g^a \wedge B_i = g^{b_i}, 0 \leqslant i \leqslant 3\}$.		verify Π_1, verify $e(B_i, \widetilde{g}) \stackrel{?}{=} e(g, \widetilde{B}_i), 0 \leqslant i \leqslant 3$,
3:		$e(B_2, \widetilde{B}_0) \stackrel{?}{=} e(C_{0,2}, \widetilde{g}), e(B_2, \widetilde{B}_1) \stackrel{?}{=} e(C_{1,2}, \widetilde{g})$,
4:		$e(B_2, \widetilde{B}_3) \stackrel{?}{=} e(C_{3,2}, \widetilde{g})$, select $r_s \xleftarrow{R} \mathbb{Z}_p$,
5:		compute the credential $cred_s = (\widetilde{\delta}_1, \delta_2, \delta_3) \leftarrow$
6:	$\xleftarrow{cred_s}$	$(\widetilde{g}^{r_s^{-1}}, (hg^d)^{r_s}, (h^d A \prod_{i=0}^{3} (B_i)^{e_i})^{r_s}$.
7: verify $e(\delta_3, \widetilde{\delta}_1) \stackrel{?}{=} e(h, \widetilde{D}) e(A, \widetilde{g}) \cdot$		
8: $\prod_{i=0}^{3} e(B_i, \widetilde{E}_i), e(\delta_2, \widetilde{\delta}_1) \stackrel{?}{=} e(h, \widetilde{g}) \cdot$		
9: $e(g, \widetilde{D})$.		
10: store $cred_s = (\widetilde{\delta}_1, \delta_2, \delta_3)$.		

Pseudocode 6. Seller Registration Algorithm

5. Our Construction

5.1. Overflow of PriTKT

PriTKT's workflow, illustrated in Figure 1, operates in the following manner. The CA initializes the system by issuing a system parameter pp and a set of ticket purchase policies $\mathbb{P}$ (Setup, step ①). Upon joining the system, the Seller (S) creates a private–public key pair (ssk, spk) by running SKeyGen, authenticates to the CA, and obtains their public key credentials $cred_s$ (SReg, step ②). The User (U) generates the private–public key pair (usk, upk) using UKeyGen, authenticates to the CA, and obtains their attribute-based credentials $cred_u$ (UReg, step ③). To obtain a ticket, U verifies their identity to S by providing a disclosure proof for selective attributes (including attributes $\mathbb{D} \subset \mathbb{A}$) and S generates a ticket tkt for U (Issue, step ④). Upon presenting the ticket, U anonymously proves its validity to Verifier (V) and discloses its validity period VP_{tkt} and double-spending identity $dsid$ (Show, step ⑤). V can traverse the token information in the blockchain and verify whether a double-spending event has occurred. In case of attempted double-spending, V can trace U's public key upk' using a ticket tkt and generate blame information (DSTrace, step ⑥). Using this blame information, any entity involved can verify the accuracy of the double-spending trace (VerifyDS, step ⑦).

5.2. High-Level Overview

In this section, we present a specific implementation of the e-ticket system defined in Section 4. The main challenge in developing the e-ticket system is creating efficient and unlinkable ticket issuance and show algorithms. The classical solution used in this regard consists of zero-knowledge proofs that prove the knowledge of hidden attributes, where those attributes are signed by a certification authority or a signer. Han et al. [18] have proposed an attribute-based e-ticketing system using this approach, and to date this remains the only such system. Han et al.'s system includes a privacy-preserving e-ticketing system with attribute-based credentials, BBS signatures, and NIZK. While this solution fulfills privacy requirements, the system is very costly, with the Issue algorithm's computational complexity being $O(N)$, where N is the number of user attributes. This significantly limits its practical use. Another strategy is to use specific signatures to prove knowledge of a subset of user attributes. The URS signature [20,44] is a constant-size signature that proves K out of N attributes, and its computational complexity is $O(N - K)$ for the prover and $O(K)$ for the verifier. Sanders [20] has proposed an extremely efficient URS scheme that can easily unlink without cost. Therefore, we have chosen to use Sanders' URS scheme to create the attribute credentials $cred_u$ and tickets tkt. URS ensures that the displayed credentials cannot be linked and supports the proof of attribute disclosure. The most important advantage of URS is that it uses randomization proof technology, avoiding the need for complex zero-knowledge proofs.

Because the private key of S is used to issue tickets to U, S generates the public–private key pair of URS, and the CA must issue the credential for the public key of S. SPS [21] allows us to effectively implement credential issuance for URS public keys.

The tickets are designed as single-use tickets, with a need to prevent double-spending by users. Moreover, in the case of double-spending, a user's public key needs to quickly be traced. To meet these requirements, we use the "Schnorr trick" [23]. When presenting tickets, a user discloses their double-spending identity *dsid* to the verifier; the verifier can instantly detect double-spending if a similar identity was used earlier. Specifically, every time a user shows a ticket, they create a challenge value c' and calculate $s' = dsrnd + usk \cdot c'$, employing the "Schnorr trick". When double-spending occurs, the user is required to show $s' = dsrnd + usk \cdot c'$ in the first show and $\overline{s}' = dsrnd + usk \cdot \overline{c}'$ in the second show. The zero-knowledge proof of knowledge ensures that *usk* and *dsrnd* are the same for both shows, while $(s', \overline{s}', c', \overline{c}')$ enables the verifier to calculate *usk*. If a user does not double-spend, *usk* remains perfectly hidden in s' (because *dsrnd* is only used once) and each displayed *dsid* is simply a random identity.

$$\mathrm{UReg}(\mathrm{U}(usk, upk, \mathbb{A}, pp) \leftrightarrow \mathrm{CA}(msk, upk, pp)) \rightarrow cred_u$$

User: U		**Central Authority**: CA
define $\mathbb{A} = \{a_i\}_{i=1}^N$ as a set of attributes.		
compute $\Pi_2 =$ ZKSoK$\{usk : upk = \widetilde{g}^{usk}\}$.	$\xrightarrow{upk, \Pi_2, \mathbb{A}}$	
		verify Π_2,
		select $r_u \xleftarrow{R} \mathbb{Z}_p$, compute credential
	$\xleftarrow{cred_u}$	$cred_u = (\widetilde{\sigma}_1, \widetilde{\sigma}_2) = (\widetilde{g}^{r_u}, (upk)^{r_u \cdot y_0} \cdot$
verify $e(g, \widetilde{\sigma}_2) \stackrel{?}{=} e(X(Y_0)^{usk} \prod_{i=1}^N Y_i^{a_i}, \widetilde{\sigma}_1)$.		$\widetilde{g}^{r_u(x + \sum_{i=1}^N y_i \cdot a_i)})$.
store the credential $cred_u = (\widetilde{\sigma}_1, \widetilde{\sigma}_2)$.		

Pseudocode 7. User Registration Algorithm

$$\mathrm{Issue}(\mathrm{U}(usk, cred_u, \mathcal{D}, \mathbb{A}, pp) \leftrightarrow \mathrm{S}(ssk, cred_s, pp, \mathbb{P})) \rightarrow (tkt, VP_{tkt})$$

User: U		**Seller**: S
		select $nonce \xleftarrow{R} \mathbb{Z}_p$.
	$\xleftarrow{\Pi_1', nonce}$	compute $\Pi_1' =$ ZKSoK$\{(a, b_0, b_1, \ldots, b_3) :$
		$A = g^a \wedge B_i = g^{b_i}, 0 \leqslant i \leqslant 3\}$.
verify Π_1'.		
set attributes disclose policy $\mathbb{D} \subset \mathbb{A}$,		
select $k, r, t \xleftarrow{R} \mathbb{Z}_p$, computes :		
$\sigma_1 = g^t \prod_{i \in [N] \setminus \mathcal{D}} Y_i^{a_i}$		
$\sigma_2 = [\prod_{i \in \mathcal{D} \cup \{0\}} Y_i]^t \prod_{i \in \mathcal{D} \cup \{0\}, j \in [N] \setminus \mathcal{D}} Z_{i,j}^{a_j}$		
$\widetilde{\sigma}_1' = \widetilde{\sigma}_1^r, \widetilde{\sigma}_2' = \widetilde{\sigma}_2^r (\widetilde{\sigma}_1')^t$		
$\sigma_3' = e(Y_0^k, \widetilde{\sigma}_1')$		
$c = H(\sigma_1 \| \sigma_2 \| \widetilde{\sigma}_1' \| \widetilde{\sigma}_2' \| \sigma_3' \| nonce)$		
$s = k + c \cdot usk$	$\xrightarrow{s, psu, \mathbb{D}, \Pi_3}$ $\xrightarrow{\sigma_1, \sigma_2, \widetilde{\sigma}_1', \widetilde{\sigma}_2', \sigma_3'}$	
$\omega = e(Y_0^{usk}, \widetilde{\sigma}_1')$		compute $c = H(\widetilde{\sigma}_1 \| \widetilde{\sigma}_2 \| \sigma_1' \| \sigma_2' \| \sigma_3' \| nonce)$,
$psu = \widetilde{B}_0^{usk} \widetilde{B}_1^{dsid'} \widetilde{B}_2^{dsrnd}$		verify $e(Y_0^s, \widetilde{\sigma}_1')(\sigma_3')^{-1} \stackrel{?}{=} [e(g, \widetilde{\sigma}_2') e(X\sigma_1 \cdot$
$\Pi_3 =$ ZKSoK$\{(usk, dsid', dsrnd) :$		$\prod_{i \in \mathcal{D}} Y_i^{a_i}, \widetilde{\sigma}_1')^{-1}]^c$, and $e(\sigma_1, \prod_{i \in \mathcal{D} \cup \{0\}} \widetilde{Y}_i) \stackrel{?}{=}$
$\omega = e(Y_0^{usk}, \widetilde{\sigma}_1') \wedge psu = \widetilde{B}_0^{usk} \widetilde{B}_1^{dsid'}$.		$e(\sigma_2, \widetilde{g})$.
$\widetilde{B}_2^{dsrnd}\}(nonce)$		compute $\omega = [(\sigma_3')^{-1} \cdot e(Y_0^s, \widetilde{\sigma}_1')]^{c^{-1}}$, verify
		Π_3. select $(dsid'', z) \xleftarrow{R} \mathbb{Z}_p^2$, set $VP_{tkt} \in$
		$\mathbb{Z}_p$, compute the ticket $tkt = (\widetilde{T}_1, \widetilde{T}_2) \leftarrow$
	$\xleftarrow{tkt, dsid'', VP_{tkt}}$	$(\widetilde{g}^z, psu^z \cdot \widetilde{g}^{z(a + b_1 \cdot dsid'' + b_3 \cdot VP_{tkt})})$.
compute $dsid = dsid' + dsid''$.		
verify $e(g, \widetilde{T}_2) \stackrel{?}{=} e(AB_0^{usk} B_1^{dsid}$.		
$B_2^{dsrnd} B_3^{VP_{tkt}}, \widetilde{T}_1)$.		
store the ticket $tkt = (\widetilde{T}_1, \widetilde{T}_2)$		

Pseudocode 8. Issue Tickets Algorithm

$\text{Show}(\text{U}(usk, tkt, VP_{tkt}, spk, pp) \leftrightarrow \text{V}(spk, pp)) \rightarrow ((dsid, dstrace), b)$

User: U		**Verifier**: V
1:	$\xleftarrow{nonce}$	select $nonce \xleftarrow{R} \mathbb{Z}_p$
2: select $r', t' \xleftarrow{R} \mathbb{Z}_p$ and compute:		
3: $T_1 = g^{t'} B_2^{dsrnd}$.		
4: $T_2 = (B_0 B_1 B_3)^{t'} \cdot (C_{0,2} C_{1,2} C_{3,2})^{dsrnd}$.		
5: $\widetilde{T}_1' = \widetilde{T}_1^{r'}; \widetilde{T}_2' = \widetilde{T}_2^{r'} (\widetilde{T}_1')^{t'}$.		
6: $T_3' = e(B_0^{dsrnd}, \widetilde{T}_1')$.		
7: $c' = H(T_1 \| T_2 \| \widetilde{T}_1' \| \widetilde{T}_2' \| T_3' \| nonce)$.		
8: $s' = dsrnd + c' \cdot usk$.	$\xrightarrow[\widetilde{T}_1, \widetilde{T}_2, T_1', T_2', T_3']{s', dsid, VP_{tkt}, \Pi_4}$	
9: $\Pi_4 = \text{ZKSoK}\{(t', dsrnd) : T_1 = g^{t'}$		compute $c' = H(T_1 \| T_2 \| \widetilde{T}_1' \| \widetilde{T}_2' \| T_3' \| nonce)$.
10: $B_2^{dsrnd} \wedge T_3' = e(B_0^{dsrnd}, \widetilde{T}_1')$		verify $e(B_0^{s'}, \widetilde{T}_1')(T_3')^{-1} \stackrel{?}{=} [e(g, \widetilde{T}_2') e(AT_1 B_1^{dsid} \cdot$
11: $\}(nonce)$.		$B_3^{VP_{tkt}}, (\widetilde{T}_1')^{-1})]^{c'}, e(T_1, \widetilde{B}_0 \widetilde{B}_1 \widetilde{B}_3) \stackrel{?}{=} e(T_2, \widetilde{g})$.
12:		verify Π_4, output $dsid, dstrace = (s', c')$ and
13:		$b = 1$; upload $(dsid, dstrace)$ to $\mathcal{BC}$.

Pseudocode 9. Show Tickets Algorithm

5.3. Concrete Construction

$\text{Setup}(1^\lambda) \rightarrow (msk, pp, \mathbb{P})$: CA takes the security parameter 1^λ as input to create the system master private key msk, public parameters pp, and a ticket purchasing policy set $\mathbb{P}$. CA defines a ticket purchasing policy set $\mathbb{P}$ and generates Type-III bilinear pair parameters $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g, \widetilde{g}, p, e)$. CA selects $(x, y_0, y_1, \ldots, y_N) \xleftarrow{R} \mathbb{Z}_p^{N+2}$, $(d, e_0, e_1, \ldots, e_3) \xleftarrow{R} \mathbb{Z}_p^5$, $h \xleftarrow{R} \mathbb{G}_1$, and computes $X = g^x, Y_i = g^{y_i}, \widetilde{Y}_i = \widetilde{g}^{y_i}, Z_{i,j} = g^{y_i \cdot y_j}$ for $0 \leqslant i \neq j \leqslant N$. In addition, CA computes $\widetilde{D} = \widetilde{g}^d$ and $\widetilde{E}_i = \widetilde{g}^{e_i}, 0 \leqslant i \leqslant 3$. CA then outputs $msk = \{x, y_0, y_1, \ldots, y_N, d, e_0, e_1, \ldots, e_3\}$ and $pp = \{g, \widetilde{g}, h, X, Y_0, \ldots, Y_N, \widetilde{Y}_0, \ldots, \widetilde{Y}_N, \{Z_{i,j}\}_{0 \leqslant i \neq j \leqslant N}, \widetilde{D}, \widetilde{E}_0, \ldots, \widetilde{E}_3\}$.

$\text{SKeyGen}(pp) \rightarrow (ssk, spk)$: S takes public parameters pp as input to generate its secret key ssk and public key spk. S selects $(a, b_0, b_1, \ldots, b_3) \xleftarrow{R} \mathbb{Z}_p^5$ and computes $A = g^a$, $B_i = g^{b_i}$, $\widetilde{B}_i = \widetilde{g}^{b_i}$, $C_{0,2} = g^{b_0 \cdot b_2}$, $C_{1,2} = g^{b_1 \cdot b_2}$, and $C_{3,2} = g^{b_3 \cdot b_2}$, where $0 \leqslant i \leqslant 3$. S outputs $ssk = \{a, b_0, b_1, \ldots, b_3\}$ and $spk = \{A, B_0, \ldots, B_3, \widetilde{B}_0, \ldots, \widetilde{B}_3, C_{0,2}, C_{1,2}, C_{3,2}\}$.

$\text{UKeyGen}(pp) \rightarrow (usk, upk)$: U takes public parameters pp as input to generate the secret key usk and public key upk. U selects $usk \xleftarrow{R} Z_p$ and computes $upk = \widetilde{g}^{usk}$.

$\text{SReg}(\text{S}(ssk, spk, pp) \leftrightarrow \text{CA}(msk, spk, pp)) \rightarrow cred_s$: As shown in Pse. 6, S interacts with CA to generate the seller's public key credential $cred_s$. S sends CA the public key spk along with a ZKSoK of ssk (Π_1) to prove that S knows the secret key ssk. S should authenticate to CA and provide (online or offline) evidence to demonstrate that it can operate as a seller. If Π_1 is verified as valid and the authentication is accepted, CA computes an SPS signature $cred_s$ as the credential of the public key spk. $cred_s$ is then sent back to S, who uses its private and public keys and their associated credential to verify that CA has authorized it as a seller.

$\text{UReg}(\text{U}(usk, upk, \mathbb{A}, pp) \leftrightarrow \text{CA}(msk, upk, pp)) \rightarrow cred_u$: as shown in Pse. 7, U interacts with CA to generate the user's attribute credential $cred_u$. U sends CA the public key upk along with a ZKSoK of usk (Π_2) and a set of attributes $\mathbb{A} = \{a_i\}_{i=1}^N$ which allow U to purchase tickets. If Π_2 is verified as valid and the attributes are authentic, CA computes a URS signature $cred_u$ as the credential of the public key upk and user attributes $\mathbb{A}$. $cred_u$ is sent back to U, who uses it to verify that they are now a legitimate user and that their attributes have been signed by CA.

$\text{Issue}(\text{U}(usk, cred_u, \mathcal{D}, \mathbb{A}, pp) \leftrightarrow \text{S}(ssk, cred_s, pp, \mathbb{P})) \rightarrow (tkt, VP_{tkt})$: as shown in Pse. 8, U interacts with S to obtain the ticket tkt. To prevent S from collecting user information maliciously, S computes the signature of knowledge Π_1' to prove to the user that it has been authenticated by CA. To prevent replay attacks, S chooses a random $nonce$ to send to U. U anonymously proves to S that they have been certified by CA as a legitimate user and

selects an attributes set $\mathbb{D}$ to disclose in order to satisfy a ticket purchasing policy selected by S from the policy set defined by CA. U then generates a new Pedersen commitment *psu* which commits a private key *usk*, double-spending identity $dsid'$, and double-spending random *dsrnd*. U constructs a ZKSoK (Π_3) to prove that U knows the knowledge of $(usk, dsid', dsrnd)$ and that *psu* and $cred_u$ have the same *usk*. If Π_3 is verified as valid, S can update the Pedersen commitment *psu* homomorphically without knowing the opening, then produces the ticket *tkt*, contributes to double-spending identity $dsid''$ of S, and clarifies the ticket's valid period VP_{tkt}. S then sends *tkt*, $dsid''$, and VP_{tkt} to U, who uses them along with U's private key to verify the validity of the ticket.

$\text{Show}(\text{U}(usk, tkt, VP_{tkt}, spk, pp) \leftrightarrow \text{V}(spk, pp)) \rightarrow ((dsid, dstrace), b)$: as shown in Pse. 9, U interacts with V to show the ticket *tkt*. To prevent replay attacks, V first chooses a random *nonce* to send to U. U anonymously proves the legitimacy of *tkt* to V and discloses *dsid* and VP_{tkt}. Then, U computes the challenge c' and $s' = dsrnd + usk \cdot c'$ to enable V to reveal the user's public key in case of double-spending. Finally, U needs to compute Π_4 to ensure that the *dsrnd* in the *tkt* is the same as in s'. If Π_4 is verified, V checks all tickets in the history with the same *dsid* to detect whether the ticket has been double-spent. If not, V uploads $(dsid, dstrace)$ to $\mathcal{BC}$ and outputs $b = 1$; otherwise, $b = 0$. Then, V traverses the token information in the blockchain and verifies whether a double-spending event has occurred. In case of double-spending being detected, V outputs *dsid* to link double-spending tickets and uses *dstrace* to trace the double-spending user's public key.

$\text{DSTrace}(dsid, dstrace, \overline{dsid}, \overline{dstrace}, spk, pp) \rightarrow (dsblame, upk')$: if U spends the same ticket a second time, this algorithm can be operated by V. If U only spends their ticket once, then *usk* is perfectly hidden. However, in the case of double-spending V can detect whether the *dsid* of the current ticket is the same as the $\overline{dsid}$ of a ticket that was spent before. This allows V to compute the private key *usk* of a double-spending user based on the fact that double-spending of the same ticket involves the same *dsid* and two different challenges $(c', \overline{c}')$ with $s' = dsrnd + usk \cdot c'$, $\overline{s}' = dsrnd + usk \cdot \overline{c}'$, allowing V to extract *usk* with overwhelming probability by parsing $\{dsid, dstrace = (s', c')\}$ and $\{\overline{dsid}, \overline{dstrace} = (\overline{s}', \overline{c}')\}$. If $dsid = \overline{dsid}$, $dsblame = \frac{s'-\overline{s}'}{\overline{c}'-c'}$ is output and $upk' = \widetilde{g}^{dsblame}$; otherwise, $\perp$ is output. It should be noted that because the user generates c' by using a random *nonce* selected by V every time a ticket is shown, there is an overwhelming probability that generation of two different c' will be forced when a ticket is double-spent.

$\text{VerifyDS}(dsblame, upk') \rightarrow b$; given double-spending blaming information *dsblame* and a public key *upk*, this algorithm outputs $b = 1$ if $upk' = \widetilde{g}^{dsblame}$ and $b = 0$ otherwise.

The details of the zero-knowledge signature of knowledge of the proposed system are shown in Supplemental Material Section S3.

6. Security Analysis

In Supplementary Material Section S1, we analyze the correctness of the proposed system. To formalize that our construction from Section 4.2 satisfies all the desired security guarantees defined in Section 3.2, we define the following theorems. Let $\Pi_1, \Pi_2, \Pi_3, \Pi_4$ be ZKSoKs. See Supplementary Material Section S2 for the formal proofs.

Theorem 1. *In the PriTKT system, the user's credential is unforgeable if the DL assumption holds in $\mathbb{G}_2$ and if the URS is unforgeable.*

Theorem 2. *In the PriTKT system, the user's tickets are unforgeable if the DL assumption holds in $\mathbb{G}_2$ and if the URS is unforgeable.*

Theorem 3. *In the PriTKT system, the user's credential is unlinkable if the DDH assumption holds in $\mathbb{G}_2$.*

Theorem 4. *In the PriTKT system, the user's tickets are unlinkable if the DDH assumption holds in $\mathbb{G}_2$.*

Theorem 5. *The PriTKT system is framing-resistant if the DL assumption holds in* $\mathbb{G}_2$.

7. Performance Analysis

7.1. Theoretical Analysis and Comparison

Table 2 presents a detailed comparison of PriTKT and related works, including four e-ticket systems [12–14,16], one attribute-based issuance e-ticket system [18], and five attribute-based credential schemes [35,36,39,40,43]. The comparison evaluates each system in terms of formal security proof, double-spending detection, double-spending trace, attribute-based issuance, and attribute disclosure. Formal proof refers to an e-ticketing system's security verification through formal methods. Double-spending detection ensures that an e-ticket cannot be reused after it has been spent, while double-spending trace allows for the identification of the responsible user when a ticket is spent twice. Attribute-based issuance allows for the issuance of tickets or credentials based on user attributes. Finally, attribute disclosure refers to the method used by users to reveal a subset of their attributes in order to purchase or display tickets. Of the e-ticketing systems included in the study, all those listed in [12–14,16,18] feature double-spending detection, while the systems in [14,18] offer double-spending trace and formal security proof. Furthermore, Ref. [18] employs complex zero-knowledge proofs (ZKP) to issue tickets based on attributes, while the attribute-based credential schemes discussed in [35,36,39,40] use ZKP to issue credentials. Lastly, the attribute-based credential schemes in [43] offer formal proof and attribute-based issuance, with the efficiency of attribute disclosure improved by replacing ZKP with SPS-EQ. However, this scheme fails to prove that the user's hidden attributes satisfy certain requirements. In comparison, PriTKT boasts formal proof, double-spending detection, double-spending trace, attribute-based issuance, and attribute disclosure. The URS signature used in PriTKT avoids the complexity associated with the use of ZKP in [18].

Table 2. Functional comparison with related works.

Scheme	Formal Proof	Double-Spending Detection	Double-Spending Trace	Attribute-Based Issuance	Disclosing Attributes
[12]	×	√	×	×	–
[16]	×	√	×	×	–
[13]	×	√	×	×	–
[14]	√	√	√	×	–
[18]	√	√	√	√	ZKP
[35]	√	–	–	√	ZKP
[36]	√	–	–	√	ZKP
[39]	√	–	–	√	ZKP
[40]	√	–	–	√	ZKP
[43]	√	–	–	√	SPS-EQ
PriTKT	√	√	√	√	URS

√: supported feature; ×: unsupported feature; –: not applicable.

Tables 3–5 feature a comparison of the PriTKT system and the only existing e-ticketing system [18] in terms of computation, storage, and communication overhead. The sizes of the elements in the groups $\mathbb{G}_1$, $\mathbb{G}_2$, $\mathbb{G}_T$, and $\mathbb{Z}_p$ are represented by $|\mathbb{G}_1|$, $|\mathbb{G}_2|$, $|\mathbb{G}_T|$, and $|\mathbb{Z}_p|$, respectively. The time costs for exponentiation in groups $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$, and the bilinear pairing maps are denoted by t_{e_1}, t_{e_2}, t_{e_T}, and t_p, respectively. The only difference in implementation between these two systems is that PriTKT uses high-efficiency Type-III pairing, where $\mathbb{G}_1 \neq \mathbb{G}_2$, while the system presented in [18] uses Type-I pairing with $\mathbb{G}_1 = \mathbb{G}_2$.

Table 3. Computation overhead comparison.

Algorithms	Entity	PriTKT	[18]
Setup	CA	$(N^2+1)t_{e_1}+(N+5)t_{e_2}$	Nt_{e_1}
UKeyGen	User	t_{e_2}	t_{e_1}
SKeyGen	Seller	$8t_{e_1}+4t_{e_2}$	t_{e_1}
UReg_U	User	$(N+1)t_{e_1}+1t_{e_2}+2t_p$	$(N+5)t_{e_1}+(N+4)t_p$
UReg_{CA}	CA	$5t_{e_2}$	$(N+2)t_{e_1}$
SReg_S	Seller	$5t_{e_1}+10t_p$	$4t_{e_1}+5t_p$
SReg_{CA}	CA	$8t_{e_1}+3t_{e_2}+14t_p$	$3t_{e_1}$
Issue_U	User	$(2(N-K)+20)t_{e_1}+9t_{e_2}+5t_p$	$(N+3K+37)t_{e_1}+(N+3K+21)t_p$
Issue_S	Seller	$(K+3)t_{e_1}+7t_{e_2}+4t_{e_T}+6t_p$	$(3K+31)t_{e_1}+Kt_{e_T}+(2K+17)t_p$
Show_U	User	$8t_{e_1}+3t_{e_2}+2t_p$	$24t_{e_1}+8t_p$
Show_V	Verifier	$7t_{e_1}+2t_T+6t_p$	$19t_{e_1}+8t_p$
DSTrace	Verifier	t_{e_2}	$3t_{e_1}$

Table 4. Storage complexity comparison.

Variates	Entity	PriTKT	[18]
pp	CA	$(N^2+3)\lvert\mathbb{G}_1\rvert+(N+6)\lvert\mathbb{G}_2\rvert$	$(N+13)\lvert\mathbb{G}_1\rvert$
upk	User	$1\lvert\mathbb{G}_2\rvert$	$1\lvert\mathbb{G}_1\rvert$
spk	Seller	$8\lvert\mathbb{G}_1\rvert+4\lvert\mathbb{G}_2\rvert$	$1\lvert\mathbb{G}_1\rvert$
$cred_u$	User	$2\lvert\mathbb{G}_1\rvert$	$2\lvert\mathbb{Z}_p\rvert+1\lvert\mathbb{G}_1\rvert$
$cred_s$	Seller	$2\lvert\mathbb{G}_1\rvert+1\lvert\mathbb{G}_2\rvert$	$2\lvert\mathbb{Z}_p\rvert+1\lvert\mathbb{G}_1\rvert$
tkt	User	$4\lvert\mathbb{Z}_p\rvert+2\lvert\mathbb{G}_2\rvert$	$5\lvert\mathbb{Z}_p\rvert+2\lvert\mathbb{G}_1\rvert$
tok	User	$5\lvert\mathbb{Z}_p\rvert+2\lvert\mathbb{G}_1\rvert+2\lvert\mathbb{G}_2\rvert+1\lvert\mathbb{G}_T\rvert$	$10\lvert\mathbb{Z}_p\rvert+8\lvert\mathbb{G}_1\rvert$

Table 5. Communication complexity comparison.

Algorithms (Variates)	Entity	PriTKT	[18]
UReg_U	User	$(N+1)\mathbb{Z}_p\rvert+1\lvert\mathbb{G}_2\rvert$	$(N+3)\lvert\mathbb{Z}_p\rvert+3\lvert\mathbb{G}_1\rvert$
UReg_{CA}	CA	$1\lvert\mathbb{Z}_p\rvert+2\lvert\mathbb{G}_2\rvert$	$3\lvert\mathbb{Z}_p\rvert+1\lvert\mathbb{G}_1\rvert$
SReg_S	Seller	$6\lvert\mathbb{Z}_p\rvert+8\lvert\mathbb{G}_1\rvert+4\lvert\mathbb{G}_2\rvert$	$2\lvert\mathbb{Z}_p\rvert+2\lvert\mathbb{G}_1\rvert$
SReg_{CA}	CA	$2\lvert\mathbb{Z}_p\rvert+1\lvert\mathbb{G}_1\rvert$	$3\lvert\mathbb{Z}_p\rvert+1\lvert\mathbb{G}_1\rvert$
Issue_U	User	$(K+5)\lvert\mathbb{Z}_p\rvert+3\lvert\mathbb{G}_1\rvert+2\lvert\mathbb{G}_2\rvert+1\lvert\mathbb{G}_T\rvert$	$(2N+K+8)\lvert\mathbb{Z}_p\rvert+(2K+5)\lvert\mathbb{G}_1\rvert$
Issue_S	Seller	$9\lvert\mathbb{Z}_p\rvert+2\lvert\mathbb{G}_2\rvert$	$6\lvert\mathbb{Z}_p\rvert+1\lvert\mathbb{G}_1\rvert$
Show_U	User	$6\lvert\mathbb{Z}_p\rvert+2\lvert\mathbb{G}_1\rvert+2\lvert\mathbb{G}_2\rvert+1\lvert\mathbb{G}_T\rvert$	$(K+10)\lvert\mathbb{Z}_p\rvert+9\lvert\mathbb{G}_1\rvert$
Show_V	Verifier	$3\lvert\mathbb{Z}_p\rvert$	$2\lvert\mathbb{Z}_p\rvert+4\lvert\mathbb{G}_1\rvert$
DSTrace	Verifier	$\lvert\mathbb{Z}_p\rvert+\lvert\mathbb{G}_2\rvert$	$\lvert\mathbb{G}_1\rvert$

Tables 3–5 show that the system in [18] is inefficient; a ticket issued on an attribute-based credential (Issue_U) requires at least $O(N)$ operations in order for a user to disclose K attributes out of a total of N attributes. In addition, it is necessary to prove knowledge of all N attributes, which implies that at least $O(N)$ elements must be sent in communications during the execution of the Issue algorithm. Our algorithm avoids this problem, significantly reducing the computational cost to $O(N-K)$ operations and the communication cost to $O(K)$ elements.

In PriTKT, the computational overhead of Setup and the storage overhead of pp are both $O(N^2)$. Fortunately, system initialization only needs to be performed once, and the system parameters pp are stored on users' IoT devices (such as smartphones), which have more than enough storage space. In PriTKT, the user credential $cred_u$, seller credential $cred_s$, and ticket tkt all have constant sizes. While the operations on Issue_U are related to the number of user attributes, they decrease with the number of disclosed attributes. The operations for Issue_S depend only on the number of disclosed attributes. The computation and communication overhead of Show and DSTrace are constant in PriTKT.

7.2. Experimental Analysis of PriTKT

We further evaluated the performance of the PriTKT system through objective tests. We implemented the system and measured its performance on an Android 9.0 operating system running on a HUAWEI Honor 9i smartphone, which had a Hisilicon Kirin 659 (ARMv8-A) CPU with a clock speed of 2.36 GHz and 1.7 GHz and with 4 GB of RAM.

We performed experiments utilizing MIRACL [48] and Type-III pairing. We used SHA256 to implement the $H : \{0,1\}^* \rightarrow \mathbb{Z}_p$ hash functions required by PriTKT (see Pseudocodes 8 and 9). To accurately evaluate the computational and storage/communication overhead of each of PriTKT's algorithms, we used the Barreto–Naehrig curve (BN-256) [49]. BN-256 was used to test the system's performance at the AES 100-bit security level, and we compared it to the performance of a scheme [18] using the Security Supersingular Elliptic Curve (SSP-1536) [50] at the same security level.

In practice, the total number of user attributes N can reach hundreds, and is usually much larger than the number of a user's disclosed attributes K. For example, a user may have *name*, *id*, *student*, *phone*, *number*, *occupation*, *home*, *address*, and many other attributes. If the user buys a student discount ticket, they only need to provide their *student* attribute. Therefore, we compared the computation and storage/communication costs with a larger number of N and a constant number of $K = 5$.

Tables 6–8 compare the computational and storage/communication overheads of all algorithms between PriTKT and [18] at $N = 50$, where each time result is averaged over 50 iterations.

As shown in Table 6, in PriTKT the Setup algorithm takes 44.2 s. For key generation, UKeyGen and SKeyGen cost 18.1 ms and 143.6 ms, respectively. For user registration, UReg_U and UReg_{CA} cost 642.8 ms and 95.5 ms, respectively. For seller registration, SReg_S and SReg_{CA} cost 1024.3 ms and 1543.2 ms, respectively. The DSTrace algorithm takes 18.5 ms. Most frequently used of e-tickets rely on two algorithms, namely, Issue and Show. When issuing a ticket, the computation overheads of the Issue_U algorithm are 1971.4 ms for PriTKT and 561,049 ms for the scheme in [18], while the computation overheads of the Issue_S algorithm are 908.2 ms for PriTKT and 201,506 ms for the scheme in [18]. When showing a ticket, the computation overheads of the Show_U algorithm are 320.4 ms for PriTKT and 79,544 ms for the scheme in [18], while the computation overheads of the Show_V algorithm are 804.3 ms for PriTKT and 69,924 ms for the scheme in [18].

As shown in Table 7, the storage of public parameters pp in PriTKT is 348,816 bytes, which is significantly larger than the 2584 bytes required by the scheme in [18]; however, we note that public parameters can be stored in smartphones with sufficient storage space. The other size overheads are similar for both PriTKT and the scheme in [18].

As shown in Table 8, the communication overheads of the Issue_U algorithm are 1856 bytes for PriTKT and 6640 bytes for the scheme in [18], while the communication overheads of the Show_U algorithm are 1568 bytes for PriTKT and 1824 bytes for the scheme in [18]. For both the Issue_S and Show_V algorithms, the communication overheads of both PriTKT and the scheme in [18] are less than 1000 bytes.

Table 6. Experimental computation costs (ms).

ALGO	Entity	PriTKT	[18]
Setup	CA	44,245	98,124
UKeyGen	User	18.1	1924
SKeyGen	Seller	143.6	1924
UReg_U	User	642.8	337,149
UReg_{CA}	CA	95.5	101,972
SReg_S	Seller	1024.3	28,551
SReg_{CA}	CA	1543.2	5772
Issue_U	User	1971.4	561,049
Issue_S	Seller	908.2	201,506
Show_U	User	320.4	79,544
Show_V	Verifier	804.3	69,924
DSTrace	Verifier	18.5	5772

Table 7. Experimental storage costs (B).

ALGO	Entity	PriTKT	[18]
pp	CA	348,816	2584
upk	User	272	136
spk	Seller	2112	136
$cred_u$	User	256	216
$cred_s$	Seller	528	216
tkt	User	704	472
tok	User	1528	1488

Table 8. Experimental communication costs (B).

ALGO	Entity	PriTKT	[18]
UReg_U	User	2352	2568
UReg_{CA}	CA	584	256
SReg_S	Seller	1584	3,52
SReg_{CA}	CA	208	256
Issue_U	User	1856	6640
Issue_S	Seller	904	376
Show_U	User	1568	1824
Show_V	Verifier	120	624
DSTrace	Verifier	312	136

Tables 9 and 10 compare the computational and communication overheads of the Issue and Show algorithms between PriTKT and the scheme in [18]; the parameters were set to $K = 5$, while N varied from 20 to 100.

Table 9. Computation overhead of Issue and Show (ms).

ALGO	System	Number of Attributess				
		20	**40**	**60**	**80**	**100**
Issue_U	PriTKT	1200.5	1723	2242.1	2767.1	3283.5
	[18]	378,196	500,098	622,002	743,903	865,801
Issue_S	PriTKT	905.6	906.7	908.1	904.5	906.7
	[18]	201,506	201,508	201,505	201,506	201,507
Show_U	PriTKT	325.9	323.1	324.2	320.2	321.4
	[18]	79,544	79,541	79,545	79,543	79,542
Show_V	PriTKT	801.1	803.9	802.4	802.3	803.6
	[18]	69,924	69,923	69,925	69,922	69,924

Table 10. Communication overheads of Issue and Show (B).

ALGO	System	Number of Attributess				
		20	**40**	**60**	**80**	**100**
Issue_U	PriTKT	1856	1856	1856	1856	1856
	[18]	4240	5840	7440	9040	10,640
Issue_S	PriTKT	904	904	904	904	904
	[18]	376	376	376	376	376
Show_U	PriTKT	1586	1568	1586	1586	1586
	[18]	1824	1824	1824	1824	1824
Show_V	PriTKT	120	120	120	120	120
	[18]	624	624	624	624	624

Table 9 shows that the computational overhead of the Issue_U algorithm in both PriTKT and the scheme of Han et al. [18] increases linearly with the number of user attributes. However, PriTKT performs over 250 times more efficiently than the scheme of Han et al. For the Issue_S algorithm, the computational overhead of both schemes are independent of the number of user attributes. Nonetheless, PriTKT is around 200 times more efficient than the scheme of Han et al. In the case of the Show_U and Show_V algorithms, the computational overhead of both schemes are independent of the number of user attributes. However, PriTKT executes around 240 times and 80 times more efficiently in the Show_U and Show_V algorithms, respectively, compared to the scheme of Han et al.

As shown in Table 10, the communication overheads of PriTKT are constant for both the Issue and Show algorithms. The communication overheads of the Issue_U, Issue_S, Show_U, and Show_V algorithms in PriTKT are as low as 1856 bytes, 904 bytes, 1586 bytes and 120 bytes, respectively. In comparison, while the scheme in [18] achieves a constant communication overhead for the Show algorithm, for the Issue_U algorithm its output size increases linearly with the number of user attributes.

Compared to the state-of-the-art described in Han et al. [18], the above analysis and comparison reveal that PriTKT incurs significantly less computational and communication overhead.

8. Conclusions

This paper presents a blockchain-enhanced privacy-preserving e-ticketing system for IoT devices. The proposed system permits users to purchase tickets anonymously by revealing some of their attributes while concealing others. The proposed system presents significantly reduced computational cost and communication overhead compared to state-

of-the-art e-ticketing systems, making it suitable for use in IoT devices. Moreover, it utilizes blockchain technology to achieve effective malicious user tracking. The system possesses robust security properties such as unlinkability, unforgeability, non-double spending, non-transferability, and framing resistance. The security properties are formally defined, and we have reduced them to well-known complexity assumptions or the security of proven cryptography primitives. We implemented the algorithms of the e-ticketing system on a smartphone, demonstrating that the system generates significantly lower computational and communication overhead compared to the state-of-the-art.

Supplementary Materials: The following supporting information can be downloaded at: https://www.mdpi.com/article/10.3390/s24020496/s1, Section S1: Correctness Analysis; Section S2: Security Proof; Section S3: Zero-Knowledge Signature of Knowledge; Section S4: Formal Correctness Definition.

Author Contributions: Conceptualization, Y.Z. and R.S.; Methodology, Y.Z.; Software, F.Y.; Validation, G.S. and C.D.; Formal analysis, F.Y.; Investigation, F.Y.; Resources, R.S.; Data curation, G.S.; Writing—original draft, Y.Z. and R.S.; Writing—review and editing, G.S. and C.D.; Funding acquisition, C.D. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by National Natural Science Foundation of China under Grant No. 62372110 and by the Fujian Provincial Natural Science of Foundation under Grant 2023J02008.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Iata.org. TATA. 2014. Available online: https://www.iata.org/en/services/certification/operations-safety-security/ (accessed on 1 January 2024).
2. Group, R.D. Rail Technical Strategy Capability Delivery Plan. 2017. Available online: https://www.rssb.co.uk/ (accessed on 1 January 2024).
3. Group, T.S.L. The Future Railway—The Industry's Rail Technical Strategy 2012. 2012. Available online: https://www.rssb.co.uk/ (accessed on 1 January 2024).
4. Nakamoto, S. Bitcoin: A Peer-to-Peer Electronic Cash System. 2008. Available online: https://bitcoin.org/bitcoin.pdf (accessed on 1 January 2024).
5. Sasson, E.B.; Chiesa, A.; Garman, C.; Green, M.; Miers, I.; Tromer, E.; Virza, M. Zerocash: Decentralized anonymous payments from bitcoin. In Proceedings of the 2014 IEEE Symposium on Security and Privacy, Berkeley, CA, USA, 18–21 May 2014; pp. 459–474.
6. Androulaki, E.; Camenisch, J.; Caro, A.D.; Dubovitskaya, M.; Elkhiyaoui, K.; Tackmann, B. Privacy-preserving auditable token payments in a permissioned blockchain system. In Proceedings of the 2nd ACM Conference on Advances in Financial Technologies, New York, NY, USA, 21–23 October 2020; pp. 255–267.
7. Sun, S.F.; Au, M.H.; Liu, J.K.; Yuen, T.H. Ringct 2.0: A compact accumulator-based (linkable ring signature) protocol for blockchain cryptocurrency monero. In Proceedings of the Computer Security–ESORICS 2017: 22nd European Symposium on Research in Computer Security, Oslo, Norway, 11–15 September 2017; Proceedings, Part II 22; Springer: Cham, Switzerland, 2017; pp. 456–474.
8. Agreement, P. Paris agreement. In *Report of the Conference of the Parties to the United Nations Framework Convention on Climate Change (21st Session, 2015: Paris). Retrived December*; HeinOnline: Getzville, NY, USA, 2015; Volume 4, p. 2017.
9. Regulation, P. General data protection regulation. *Intouch* **2018**, *25*, 1–5.
10. Chaum, D. Blind signatures for untraceable payments. In *Advances in Cryptology: Proceedings of Crypto 82*; Springer: Boston, MA, USA, 1983; pp. 199–203.
11. Chaum, D.; Van Heyst, E. Group signatures. In Proceedings of the Advances in Cryptology—EUROCRYPT'91: Workshop on the Theory and Application of Cryptographic Techniques, Brighton, UK, 8–11 April 1991; Proceedings 10; Springer: Berlin/Heidelberg, Germany, 1991; pp. 257–265.
12. Heydt-Benjamin, T.S.; Chae, H.J.; Defend, B.; Fu, K. Privacy for public transportation. In Proceedings of the Privacy Enhancing Technologies: 6th International Workshop, PET 2006, Cambridge, UK, 28–30 June 2006, Revised Selected Papers 6; Springer: Berlin/Heidelberg, Germany, 2006; pp. 1–19.
13. Milutinovic, M.; Decroix, K.; Naessens, V.; De Decker, B. Privacy-preserving public transport ticketing system. In Proceedings of the Data and Applications Security and Privacy XXIX: 29th Annual IFIP WG 11.3 Working Conference, DBSec 2015, Fairfax, VA, USA, 13–15 July 2015; Proceedings 29; Springer: Cham, Switzerland, 2015; pp. 135–150.

14. Rupp, A.; Baldimtsi, F.; Hinterwälder, G.; Paar, C. Cryptographic theory meets practice: Efficient and privacy-preserving payments for public transport. *ACM Trans. Inf. Syst. Secur. (TISSEC)* **2015**, *17*, 10. [CrossRef]
15. Lysyanskaya, A.; Rivest, R.L.; Sahai, A.; Wolf, S. Pseudonym systems. In Proceedings of the Selected Areas in Cryptography: 6th Annual International Workshop (SAC'99), Kingston, ON, Canada, 9–10 August 1999; Proceedings 6; Springer: Berlin/Heidelberg, Germany, 2000; pp. 184–199.
16. Vives-Guasch, A.; Payeras-Capellà, M.M.; Mut-Puigserver, M.; Castella-Roca, J.; Ferrer-Gomila, J.L. A secure e-ticketing scheme for mobile devices with near field communication (NFC) that includes exculpability and reusability. *IEICE Trans. Inf. Syst.* **2012**, *95*, 78–93. [CrossRef]
17. Chaum, D. Security without identification: Transaction systems to make big brother obsolete. *Commun. ACM* **1985**, *28*, 1030–1044. [CrossRef]
18. Han, J.; Chen, L.; Schneider, S.; Treharne, H.; Wesemeyer, S. Privacy-preserving electronic ticket scheme with attribute-based credentials. *IEEE Trans. Dependable Secur. Comput.* **2019**, *18*, 1836–1849. [CrossRef]
19. by Design Foundation, P. IRMA Attributes. 2023. Available online: https://privacybydesign.foundation/attribute-index/en/ (accessed on 1 January 2024).
20. Sanders, O. Efficient redactable signature and application to anonymous credentials. In Proceedings of the Public-Key Cryptography–PKC 2020: 23rd IACR International Conference on Practice and Theory of Public-Key Cryptography, Edinburgh, UK, 4–7 May 2020; Proceedings, Part II; Springer: Cham, Switzerland, 2020; pp. 628–656.
21. Groth, J. Efficient fully structure-preserving signatures for large messages. In Proceedings of the Advances in Cryptology–ASIACRYPT 2015: 21st International Conference on the Theory and Application of Cryptology and Information Security, Auckland, New Zealand, 29 November–3 December 2015; Proceedings, Part I; Springer: Berlin/Heidelberg, Germany, 2016; pp. 239–259.
22. Chase, M.; Lysyanskaya, A. On signatures of knowledge. In Proceedings of the Advances in Cryptology-CRYPTO 2006: 26th Annual International Cryptology Conference, Santa Barbara, CA, USA, 20–24 August 2006; Proceedings 26; Springer: Berlin/Heidelberg, Germany, 2006; pp. 78–96.
23. Bobolz, J.; Eidens, F.; Krenn, S.; Slamanig, D.; Striecks, C. Privacy-preserving incentive systems with highly efficient point-collection. In Proceedings of the 15th ACM Asia Conference on Computer and Communications Security, Taipei, Taiwan, 5–9 October 2020; pp. 319–333.
24. Mut-Puigserver, M.; Payeras-Capellà, M.M.; Ferrer-Gomila, J.L.; Vives-Guasch, A.; Castellà-Roca, J. A survey of electronic ticketing applied to transport. *Comput. Secur.* **2012**, *31*, 925–939. [CrossRef]
25. Damgård, I.; Dupont, K.; Pedersen, M.Ø. Unclonable group identification. In Proceedings of the Advances in Cryptology-EUROCRYPT 2006: 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, St. Petersburg, Russia, 28 May 28–1 June 2006; Proceedings 25; Springer: Berlin/Heidelberg, Germany, 2006; pp. 555–572.
26. Ateniese, G.; Fu, K.; Green, M.; Hohenberger, S. Improved proxy re-encryption schemes with applications to secure distributed storage. *ACM Trans. Inf. Syst. Secur. (TISSEC)* **2006**, *9*, 1–30. [CrossRef]
27. Jager, T.; Rupp, A. Black-Box Accumulation: Collecting Incentives in a Privacy-Preserving Way. *Proc. Priv. Enhancing Technol.* **2016**, *2016*, 62–82. [CrossRef]
28. Hartung, G.; Hoffmann, M.; Nagel, M.; Rupp, A. BBA+ Improving the Security and Applicability of Privacy-Preserving Point Collection. In Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, Dallas, TX, USA, 30 October–3 November 2017; pp. 1925–1942.
29. Hoffmann, M.; Klooß, M.; Raiber, M.; Rupp, A. Black-box wallets: Fast anonymous two-way payments for constrained devices. *Proc. Priv. Enhancing Technol.* **2020**, *2020*, 165–194. [CrossRef]
30. Pedersen, T.P. Non-interactive and information-theoretic secure verifiable secret sharing. In *Advances in Cryptology—CRYPTO'91: Proceedings*; Springer: Berlin/Heidelberg, Germany, 2001; pp. 129–140.
31. Abe, M.; Okamoto, T. Provably secure partially blind signatures. In Proceedings of the Advances in Cryptology—CRYPTO 2000: 20th Annual International Cryptology Conference, Santa Barbara, CA, USA, 20–24 August 2000; Proceedings 20; Springer: Berlin/Heidelberg, Germany, 2000; pp. 271–286.
32. Camenisch, J.; Lysyanskaya, A. An efficient system for non-transferable anonymous credentials with optional anonymity revocation. In Proceedings of the Advances in Cryptology—EUROCRYPT 2001: International Conference on the Theory and Application of Cryptographic Techniques, Innsbruck, Austria, 6–10 May 2001; Proceedings 20; Springer: Berlin/Heidelberg, Germany, 2001; pp. 93–118.
33. Boneh, D.; Boyen, X. Short signatures without random oracles. In Proceedings of the Advances in Cryptology-EUROCRYPT 2004: International Conference on the Theory and Applications of Cryptographic Techniques, Interlaken, Switzerland, 2–6 May 2004; Proceedings 23; Springer: Berlin/Heidelberg, Germany, 2004; pp. 56–73.
34. Camenisch, J.; Chaabouni, R.; Shelat, A. Efficient protocols for set membership and range proofs. In Proceedings of the 14th International Conference on the Theory and Application of Cryptology and Information Security, Melbourne, Australia, 7–11 December 2008; Springer: Berlin/Heidelberg, Germany, 2008; pp. 234–252.
35. Camenisch, J.; Lysyanskaya, A. A signature scheme with efficient protocols. In Proceedings of the Security in Communication Networks: Third International Conference (SCN 2002), Amalfi, Italy, 11–13 September 2002; Revised Papers 3; Springer: Berlin/Heidelberg, Germany, 2003; pp. 268–289.

36. Camenisch, J.; Lysyanskaya, A. Signature schemes and anonymous credentials from bilinear maps. In Proceedings of the Advances in Cryptology–CRYPTO 2004: 24th Annual International Cryptology Conference, Santa Barbara, CA, USA, 15–19 August 2004; Proceedings 24; Springer: Berlin/Heidelberg, Germany, 2004; pp. 56–72.
37. Boneh, D.; Boyen, X.; Shacham, H. Short group signatures. In Proceedings of the CRYPTO 2004: Annual International Cryptology Conference, Santa Barbara, CA, USA, 15–19 August 2004; Springer: Berlin/Heidelberg, Germany, 2004; Volume 3152, pp. 41–55.
38. Boneh, D.; Boyen, X. Short signatures without random oracles and the SDH assumption in bilinear groups. *J. Cryptol.* **2008**, *21*, 149–177. [CrossRef]
39. Au, M.H.; Susilo, W.; Mu, Y. Constant-size dynamic k-TAA. In Proceedings of the Security and Cryptography for Networks: 5th International Conference (SCN 2006), Maiori, Italy, 6–8 September 2006; Proceedings 5; Springer: Berlin/Heidelberg, Germany, 2006; pp. 111–125.
40. Pointcheval, D.; Sanders, O. Short randomizable signatures. In Proceedings of the Topics in Cryptology-CT-RSA 2016: The Cryptographers' Track at the RSA Conference 2016, San Francisco, CA, USA, 29 February–4 March 2016; Proceedings; Springer: Cham, Switzerland, 2016; pp. 111–126.
41. Camenisch, J.; Drijvers, M.; Lehmann, A.; Neven, G.; Towa, P. Short threshold dynamic group signatures. In Proceedings of the Security and Cryptography for Networks: 12th International Conference, SCN 2020, Amalfi, Italy, 14–16 September 2020; Proceedings; Springer: Cham, Switzerland, 2020; pp. 401–423.
42. Fuchsbauer, G.; Hanser, C.; Slamanig, D. Practical round-optimal blind signatures in the standard model. In Proceedings of the Advances in Cryptology–CRYPTO 2015: 35th Annual Cryptology Conference, Santa Barbara, CA, USA, 16–20 August 2015; Proceedings, Part II; Springer: Berlin/Heidelberg, Germany, 2015; pp. 233–253.
43. Fuchsbauer, G.; Hanser, C.; Slamanig, D. Structure-preserving signatures on equivalence classes and constant-size anonymous credentials. *J. Cryptol.* **2019**, *32*, 498–546. [CrossRef]
44. Camenisch, J.; Dubovitskaya, M.; Haralambiev, K.; Kohlweiss, M. Composable and modular anonymous credentials: Definitions and practical constructions. In Proceedings of the Advances in Cryptology–ASIACRYPT 2015: 21st International Conference on the Theory and Application of Cryptology and Information Security, Auckland, New Zealand, 29 November–3 December 2015; Proceedings, Part II 21; Springer: Berlin/Heidelberg, Germany, 2015; pp. 262–288.
45. Groth, J.; Sahai, A. Efficient non-interactive proof systems for bilinear groups. In Proceedings of the Advances in Cryptology–EUROCRYPT 2008: 27th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Istanbul, Turkey, 13–17 April 2008; Proceedings 27; Springer: Berlin/Heidelberg, Germany, 2008; pp. 415–432.
46. Galbraith, S.D.; Paterson, K.G.; Smart, N.P. Pairings for cryptographers. *Discret. Appl. Math.* **2008**, *156*, 3113–3121. [CrossRef]
47. Bellare, M.; Shi, H.; Zhang, C. Foundations of group signatures: The case of dynamic groups. In Proceedings of the Topics in Cryptology–CT-RSA 2005: The Cryptographers' Track at the RSA Conference 2005, San Francisco, CA, USA, 14–18 February 2005; Proceedings; Springer: Berlin/Heidelberg, Germany, 2005; pp. 136–153.
48. MIRACL Ltd. 2019. Available online: https://github.com/miracl/MIRACL (accessed on 1 January 2024).
49. Fan, J.; Vercauteren, F.; Verbauwhede, I. Faster Fp-arithmetic for cryptographic pairings on Barreto-Naehrig curves. In Proceedings of the CHES 2009: International Workshop on Cryptographic Hardware and Embedded Systems, Lausanne, Switzerland, 6–9 September 2009; Springer: Berlin/Heidelberg, Germany, 2009; Volume 9, pp. 240–253.
50. Blake, I.; Seroussi, G.; Smart, N. *Elliptic Curves in Cryptography*; Cambridge University Press: Cambridge, UK, 1999; Volume 265.