




Article

# Anonymous Access System with Limited Number of Uses in a Trustless Environment

Francesc Garcia-Grau <sup>1,\*</sup> , Jordi Herrera-Joancomartí <sup>2</sup>  and Aleix Dorca Josa <sup>3</sup> <sup>1</sup> Escola de Doctorat, Universitat d'Andorra, AD600 Sant Julià de Lòria, Andorra<sup>2</sup> Departament d'Enginyeria de la Informació i les Comunicacions, Universitat Autònoma de Barcelona, CYBERCAT-Center, 08193 Barcelona, Spain; jordi.herrera@uab.cat<sup>3</sup> Departament de Serveis Informàtics, Universitat d'Andorra, AD600 Sant Julià de Lòria, Andorra; adorca@uda.ad

\* Correspondence: fgarcia@uda.ad

**Abstract:** This article proposes a novel method for managing usage counters within an anonymous credential system, addressing the limitation of traditional anonymous credentials in tracking repeated use. The method takes advantage of blockchain technology through Smart Contracts deployed on the Ethereum network to enforce a predetermined maximum number of uses for a given credential. Users retain control over increments by providing zero-knowledge proofs (ZKPs) demonstrating private key possession and agreement on the increment value. This approach prevents replay attacks and ensures transparency and security. A prototype implementation on a private Ethereum blockchain demonstrates the feasibility and efficiency of the proposed method, paving the way for its potential deployment in real-world applications requiring both anonymity and usage tracking.

**Keywords:** attribute-based credentials; pseudonyms; privacy-preserving credentials; self-blinded scheme; security proofs; user-centred system; zero knowledge proofs; trust-less; access counter



**Citation:** Garcia-Grau, F.; Herrera-Joancomartí, J.; Dorca Josa, A. Anonymous Access System with Limited Number of Uses in a Trustless Environment. *Appl. Sci.* **2024**, *14*, 8581. <https://doi.org/10.3390/app14198581>

Academic Editor: Douglas O'Shaughnessy

Received: 25 August 2024

Revised: 15 September 2024

Accepted: 17 September 2024

Published: 24 September 2024



**Copyright:** © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

## 1. Introduction

Privacy is, nowadays, one of the biggest challenges in our digital life [1]. Anonymity seems to be a way to solve the challenge of maintaining privacy. Anonymous credentials [2], which assert an identity while maintaining privacy, are the most widely accepted.

Anonymous credentials are commonly implemented as attribute-based credentials or ABC [3], and they guarantee the protection of user data and preserve privacy and anonymity. It is based on trusted entities, also known as the Attribute Provider  $\mathcal{AP}$ , that generates irrefutable cryptographic evidence about the user's attributes. These proofs are called Verifiable Attributes (VA). Furthermore, users store these VAs and use them wherever and whenever they want. Users can subsequently use the VAs to demonstrate possession of the attributes to a third party without revealing their identity if the VA does not identify them, and the third party can verify those attributes if it trusts  $\mathcal{AP}$ . For example, a user wants to purchase an age-restricted item on a website but wishes to remain anonymous, avoiding sharing their name, age, or date of birth. To do so, a VA can be obtained from the Civil Registry, which will act as  $\mathcal{AP}$ . However, as stated in article [4], these credentials are of no use when repeated uses must be detected. The article proposes a method for detecting credential reuse using a unique, anonymous identifier that cannot be linked to the user's real identity.

The main objective of this proposal is to extend the protocol presented in [4] in such a way that it allows the control of a maximum number of usages of a given credential. Such enforcement is carried out using blockchain technology through the implementation of Smart Contracts, a tool that provides the required robustness and transparency for authentication environments. It allows the user to have control over increments by forcing the need to provide a Zero Knowledge Proof (ZKP) that proves the possession of a private

key and the agreement on the value to be incremented. This, at the same time, prevents replay attacks. This is achieved by agreeing on the counter's increment.

The system involves three different entities: the User ( $\mathcal{U}$ ), an Attribute Provider ( $\mathcal{AP}$ ), accountable for proclaiming certain user attributes, and a Service Provider ( $\mathcal{SP}$ ), granting access to a service. As detailed in the use case in Section 6, consider a user ( $\mathcal{U}$ ) whose income is below the minimum wage, a requirement certified by  $\mathcal{AP}$ . This user is then granted thirty free underground trips per month. The process involves  $\mathcal{U}$  obtaining credentials from  $\mathcal{AP}$ , storing them locally, and presenting these blinded credentials along with proof of private key possession to  $\mathcal{SP}$ .

By use of a Smart Contract,  $\mathcal{SP}$  verifies  $\mathcal{U}$ 's data and initialises the counter on the blockchain. Subsequently, for each use,  $\mathcal{U}$  must provide a unique identifier and proof of access verification, which  $\mathcal{SP}$  will verify using the same Smart Contract.

Ethereum [5], a well-known blockchain technology, offers a decentralised, Turing-complete computer. This platform enables reliable and publicly verifiable execution of Smart Contracts, making it particularly suitable for enforcing security checks. This research proposes a set of Smart Contracts designed to manage the entire life cycle of a counter on the Ethereum network.

The rest of this article is structured as follows. Section 2 reviews the current state of the art in electronic ticketing proposals, focusing on privacy implementations and other key goals. Section 3 introduces the protocol to be enhanced with the improvements discussed above. Section 4 explains how to achieve the objectives and provides a detailed description of the protocol. Section 5 details the security evaluation of the proposed solution, and Section 6 explores, in detail, a use case of the proposed protocol. Finally, Section 7 presents the conclusions with additional ideas for future work.

## 2. Literature Review

Ethereum [5] provides a reliable and verifiable way to execute Smart Contracts, especially to enforce security checks, and ensures the life cycle of access counters. For this reason, the research is centred on Ethereum-based implementations.

In [4], the authors show a reusable anonymous credential protocol based on ABC [6,7], Verheul self-blinding schemes [8], and modified short ZSS (Zhang, Safavi-Naini, and Susilo) signatures on bilinear pairings over elliptic curves [9]. This approach aligns with the goals outlined in the cited article. One of the drawbacks of this solution is the use of elliptic curve arithmetic, which consumes a large amount of gas. However, with the use of pre-compiled contracts introduced with the Ethereum Improvement Proposal, it is possible to reduce this consumption. More specifically, EIP-196 [10] and EIP-197 [11] provide pre-compiled contracts for addition and scalar multiplication on the elliptic curve `alt_bn128`, and pre-compiled contracts for optimal Ate pairing [12] check on the elliptic curve `alt_bn128`. Another improvement is in the choice of the hash function used, using keccak instead of sha256, which is more expensive [5].

These access counters might be seen as electronic tickets, and for this purpose, a detailed analysis of electronic ticket systems in the literature is required. This research focuses on systems that use blockchain as the main backend.

Many electronic ticketing systems, usually for event tickets, face the bad behaviour of scalpers who damage the customer's interest and disrupt the normal order of the market. In [13], the authors describe a system based on the Consortium blockchain, where the control and management of the environment are handled by members of the consortium, and together with a regulatory mechanism, they effectively reduce the scalper's profits. This system does not face forward to a multi-usage system because it has no sense of its purpose.

In the same scope of event ticketing, in [14], the authors propose a ticket system with a QR code, but it is not multi-usage, does not use a blockchain backend, and is not anonymous. Another system for one-time ticketing has been developed in [15]. It uses Anonymous Credentials to guarantee user privacy, but again it does not use blockchain as the backend.

Feulner et al. [16] demonstrated that Self-Sovereign Identity [17] based event ticketing facilitated the practical implementation of the centralised exchange model to enable efficient secondary market control.

Another field where electronic tickets are introduced is the railway industry. J.D. Preece in [18] introduced a digital ticketing platform using blockchain. They used IBM's Hyperledger Fabric Framework to design an architecture that distributes tickets among all participants. However, in this approach, even if they took care of multi-usage through limited time, they did not check or implement the number of times the ticket was used.

Ricard Borges et al. in [19] proposed a system that includes reusability in the sense that a single electronic ticket allows one to make several journeys. The main goal of this system is to preserve privacy. As in the previous system, multi-usage is limited to time and not to the number of times it is used. There are other proposals [20–22] that allow transfers so that public transportation passengers can continue their journey on other buses or trains. In systems that limit the number of transfers, there is usually a mechanism in which an electronic ticket contains a hash chain whose size depends on the maximum number of transfers allowed. In this context, Stockburger et al. [23] proposed a solution involving a Self-Sovereign Identity. They proposed a credential to travel across Europe but not an electronic ticket system similar to an access counter, as the one described in this paper.

In [24], a blockchain-enhanced privacy-preserving electronic ticket system for IoT devices has been built. However, it lacks multi-use support, which is required to implement counter access.

After the evaluation of some of the research based on electronic ticketing services [13–16,18–22,24–26] found in the literature, the conclusion is that most of these are focused on granting access to events; solving problems on ticket reselling in the secondary market [14,16,26], privacy protection [15,16,23,24], bot protection and scalping [13,14], while some also envisage anonymity [16,23,25]. Other research has been oriented towards the transport industry re-usability [19–22]. Despite their strengths, existing solutions lack a crucial combination: anonymity, multi-use capability, and safeguards against malicious service providers ( $\mathcal{SP}$ ) potentially engaging in replay attacks.

### 3. Background

As introduced earlier, this proposal builds upon the anonymous credential authentication protocol described in [4] based on verifiable attributes. This section offers a general overview of that protocol. Table 1 provides a detailed guide to the notation used throughout this work.

The anonymous credential authentication protocol defines three distinct roles: the user ( $\mathcal{U}$ ), who seeks authentication, the attribute provider ( $\mathcal{AP}$ ), who holds information about the user, and the service provider ( $\mathcal{SP}$ ) who offers services to the user contingent upon predefined conditions. Each of these roles has corresponding pairs of private and public keys, denoted by  $(sk_{\mathcal{U}}, pk_{\mathcal{U}})$ ,  $(sk_{\mathcal{AP}}, pk_{\mathcal{AP}})$ , and  $(sk_{\mathcal{SP}}, pk_{\mathcal{SP}})$ , respectively.

The anonymous credential authentication protocol contains two sub-protocols. The credential issuance protocol, between  $\mathcal{U}$  and  $\mathcal{AP}$ , and the credential presentation protocol, between  $\mathcal{U}$  and  $\mathcal{SP}$ .

The detailed steps for the credential issuance protocol are the following:

1.  $\mathcal{U}$  sends the scope  $S$  and their public key  $pk_{\mathcal{U}}$  to  $\mathcal{AP}$ .
2.  $\mathcal{AP}$  computes the verifiable attribute  $\delta_{\mathcal{AP}}$  as follows:

$$\delta_{\mathcal{AP}} = (H(S) + sk_{\mathcal{AP}})^{-1} \cdot pk_{\mathcal{U}}$$

that links the scope  $S$  with  $\mathcal{U}$ 's public key through the private key of  $\mathcal{AP}$ .

3.  $\mathcal{AP}$  sends  $\mathcal{U}$  the computed  $\delta_{\mathcal{AP}}$ .
4.  $\mathcal{U}$  computes their  $id_{\mathcal{U}}$ :

$$id_{\mathcal{U}} = H(S)^{-1}(sk_{\mathcal{U}} + H(S))^{-1} \cdot pk_{\mathcal{U}} \quad (1)$$

**Table 1.** Notation guide.

Notation	Meaning
$\mathcal{AP}$	Attribute Provider
$\mathcal{SP}$	Service Provider
$id_{\mathcal{U}}$	User identifier
$id_{\mathcal{U}}^{S_i}$	User identifier for scope $S_i$
$sk_{\mathcal{U}}$	User secret key
$\tilde{sk}_{\mathcal{U}}$	Fake User secret key
$sk_{\mathcal{AP}}$	Attribute Provider secret key
$sk_{\mathcal{SP}}$	Service Provider secret key
$\delta_{\mathcal{AP}}$	Signature of Attribute Provider
$P$	Generator of the cyclic group $\mathbb{G}$
$sk'_{\mathcal{U}}$	Blinded User secret key
$\delta'_{\mathcal{AP}}$	Blinded signature
$n$	Maximum number of authentications
$e(,)$	Pairing function
$\mathcal{U}$	User
$\tilde{\mathcal{A}}$	Adversary
$S$	Scope (arbitrary string)
$\tilde{id}_{\mathcal{U}}$	Fake User identifier
$pk_{\mathcal{U}}$	User public key
$\tilde{pk}_{\mathcal{U}}$	Fake User public key
$pk_{\mathcal{AP}}$	Attribute Provider public key
$pk_{\mathcal{SP}}$	Service Provider public key
$H(S)$	Hash of scope
$b$	Random blind factor
$pk'_{\mathcal{U}}$	Blinded User public key
$(R', t_i)$	Modified NI-Schnorr ZKP
$i$	Auth counter in $[0, \dots, n]$

Once this protocol has been completed,  $\mathcal{U}$  can anonymously authenticate to  $\mathcal{SP}$  using the credential presentation protocol. This protocol has the following steps:

1.  $\mathcal{U}$  chooses  $b \in_R \mathbb{F}_q^*$  as a factor to blind the values needed to obtain the anonymous credentials and computes  $C'$ :

$$\begin{aligned}
 sk'_{\mathcal{U}} &= b \cdot sk_{\mathcal{U}} \\
 pk'_{\mathcal{U}} &= sk'_{\mathcal{U}} \cdot P \\
 \delta'_{\mathcal{AP}} &= b \cdot \delta_{\mathcal{AP}} \\
 P' &= b \cdot P \\
 pk'_{\mathcal{AP}} &= b \cdot pk_{\mathcal{AP}} \\
 C' &= b \cdot H(S) \cdot P
 \end{aligned}
 \tag{2}$$

2.  $\mathcal{U}$  also computes a NI-Schnorr ZKP by choosing  $r' \in_R \mathbb{F}_q^*$  and obtaining:

$$\begin{aligned}
 R' &= r' \cdot P \\
 h' &= H(R') \\
 t' &= h' \cdot sk'_{\mathcal{U}} + r'
 \end{aligned}
 \tag{3}$$

3.  $\mathcal{U}$  sends  $\mathcal{SP}$  the identifier  $id_{\mathcal{U}}$ , the anonymous credentials  $\delta'_{\mathcal{AP}}, pk'_{\mathcal{U}}, pk'_{\mathcal{AP}}, P', C'$ , and the NI-Schnorr ZKP proof of the private key  $(R', h', t')$ .
4.  $\mathcal{SP}$  verifies that  $pk'_{\mathcal{AP}}$  is the blinded value of  $pk_{\mathcal{AP}}$  by computing:

$$e(pk'_{\mathcal{AP}}, P) \stackrel{?}{=} e(pk_{\mathcal{AP}}, P')
 \tag{4}$$

5.  $\mathcal{SP}$  also verifies the correction of the credentials:

$$e(C' + pk'_{\mathcal{AP}}, \delta'_{\mathcal{AP}}) \stackrel{?}{=} e(P', pk'_{\mathcal{U}}) \quad (5)$$

6.  $\mathcal{SP}$  continues to verify the correction of  $id_{\mathcal{U}}$  by computing:

$$e(C' + pk'_{\mathcal{U}}, H(S) \cdot id_{\mathcal{U}}) \stackrel{?}{=} e(P, pk'_{\mathcal{U}}) \quad (6)$$

7. Finally,  $\mathcal{SP}$  verifies that  $\mathcal{U}$  possesses the private key that checks the correctness of the NI-Schnorr ZKP as follows:

$$t' \cdot P \stackrel{?}{=} R' + pk'_{\mathcal{U}} \cdot h' \quad (7)$$

#### 4. The Proposal

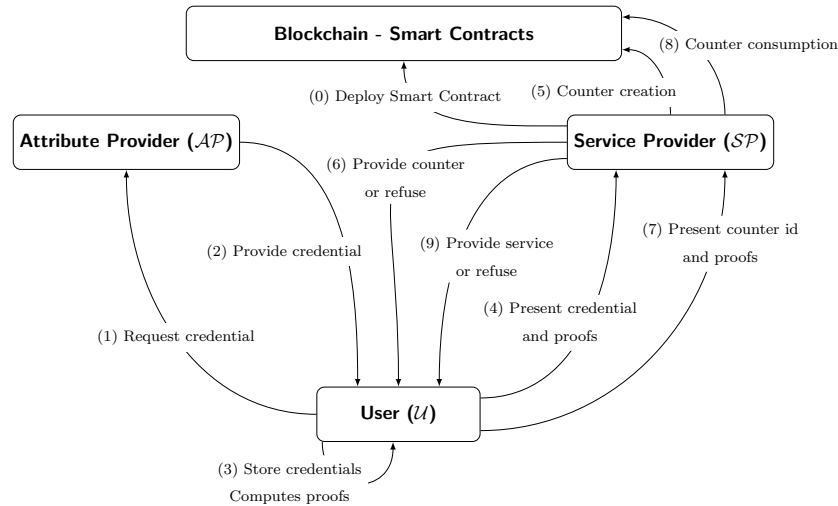
Although the protocol outlined in the previous section provides anonymity, unlinkability from real identity, and traceability within scope  $S$ , it lacks a mechanism for multiple authentications within the same scope. The only current solution is a naive approach where  $\mathcal{SP}$  counts the total authentications. However, this method fails to protect  $\mathcal{U}$  from malicious behaviour by  $\mathcal{SP}$ , who could potentially manipulate the counter.

##### 4.1. General Overview

In this proposal, blockchain technology is used through Smart Contracts to provide a secure implementation of the multiple authentication mechanism that protects  $\mathcal{U}$  in front of a malicious  $\mathcal{SP}$ . The idea is that the Smart Contract manages the life cycle of every authentication in order to properly restrict the total number of authentications to the predefined maximum  $n$ .

Figure 1 illustrates the overall protocol, outlining the three defined roles and highlighting the blockchain's new role within the system. A detailed explanation of each protocol step follows.

0. First and only once,  $\mathcal{SP}$  deploys the Smart Contract initialised with the public key of a trusted  $\mathcal{AP}$ .
1.  $\mathcal{U}$  begins the protocol by requesting  $\mathcal{AP}$  for a credential for a given scope  $S$ .
2. After user identity and attribute verification,  $\mathcal{AP}$  provides  $\mathcal{U}$  with a credential. This credential is valid only for the requested scope  $S$ .
3.  $\mathcal{U}$  stores the credential and computes the cryptography data and proofs required to continue with the protocol.
4.  $\mathcal{U}$  presents the credentials, proof of possession of the private key, and knowledge of the maximum value of the counter to obtain an access counter from  $\mathcal{SP}$ .
5.  $\mathcal{SP}$  calls the Smart Contract that verifies the credentials and the ZKPs, and all of them are valid, it stores a counter on the chain, identified by the hash of credentials, with a starting value of 0, the maximum value  $n$ , and the blinded public key of  $\mathcal{U}$ . If verification fails, the protocol is aborted.
6.  $\mathcal{SP}$  provides or refuses the counter.
7. To use the counter,  $\mathcal{U}$  presents their unique ID and proof of possession of the private key and knowledge of the next value.
8.  $\mathcal{SP}$  calls the Smart Contract that retrieves the actual value, the maximum value, and the blinded public key of  $\mathcal{U}$ . Then, it checks the proof of possession of the private key and the knowledge of the next value. The Smart Contract also checks that the number of iterations is less than the maximum allowed counter value. If the verification holds, it provides the service; otherwise, the service is refused.
9.  $\mathcal{SP}$  provides or refuses the service.



**Figure 1.** Overview of the protocol interactions.

#### 4.2. Protocol Design Rationale

Creating a counter begins with a Smart Contract requiring a ZKP demonstrating  $\mathcal{U}$ 's private key possession associated with their unique identifier ( $id_{\mathcal{U}}$ ) and agreement on a maximum usage limit. Each subsequent credential use, which increments the counter, also requires a ZKP proving private key possession and agreeing upon the increment value. This ZKP-based approach safeguards users against potential  $\mathcal{SP}$  misconduct by mandating user consent for counter value increments. To achieve this, the protocol uses parameters  $n$  (maximum uses) and  $i$  (increment value) within the computation of NI-Scnorr ZKPs.

The previously detailed protocol is then modified to introduce the use of the new variables. In Step 2 of the original protocol (Formula (3)), the number of times the credential has been used,  $i$ , and the hash of the anonymous credentials,  $H(id_{\mathcal{U}})$ , are added to the ZKP computation.  $\mathcal{U}$  then computes ZKPs by choosing  $r' \in_{\mathbb{R}} \mathbb{F}_q^*$  and modifying the computation as follows:

$$\begin{aligned} R' &= r' \cdot P \\ h_i &= (i + 1) \cdot H(id_{\mathcal{U}}) \\ t_i &= h_i \cdot sk'_{\mathcal{U}} + r' \end{aligned} \tag{8}$$

where  $H(id_{\mathcal{U}})$  is the counter identifier and  $t_i$  is the ZKP for each credential usage  $i$  for the total possible usages  $n$ . Verification in Step 7 (Formula (7)) is then to be carried out as follows:

$$\begin{aligned} t_i \cdot P &= (h_i \cdot sk'_{\mathcal{U}} + r') \cdot P \\ &= (i + 1) \cdot H(id_{\mathcal{U}}) \cdot sk'_{\mathcal{U}} \cdot P + r' \cdot P \\ &= r' \cdot P + (i + 1) \cdot H(id_{\mathcal{U}}) \cdot pk'_{\mathcal{U}} \\ &= R' + (i + 1) \cdot H(id_{\mathcal{U}}) \cdot pk'_{\mathcal{U}} \end{aligned} \tag{9}$$

#### 4.3. Credential Issuance Protocol

The credential issuance protocol comprises Steps 1, 2 and 3 of Figure 1 and is analogous to the credential issuance protocol proposed in [4]. The details of this protocol are described below.

1.  $\mathcal{U}$  sends the scope  $S$  and their public key  $pk_{\mathcal{U}}$  to  $\mathcal{AP}$ .
2.  $\mathcal{AP}$  computes the verifiable attribute  $\delta_{\mathcal{AP}}$  as follows:

$$\delta_{\mathcal{AP}} = (H(S) + sk_{\mathcal{AP}})^{-1} \cdot pk_{\mathcal{U}}$$

that links the scope  $S$  with  $\mathcal{U}$ 's public key through the private key of  $\mathcal{AP}$ , and sends  $\delta_{\mathcal{AP}}$  back to  $\mathcal{U}$ .

3.  $\mathcal{U}$  stores the computed  $\delta_{\mathcal{AP}}$  received from  $\mathcal{AP}$  and computes and stores  $id_{\mathcal{U}}$ :

$$id_{\mathcal{U}} = H(S)^{-1}(sk_{\mathcal{U}} + H(S))^{-1} \cdot pk_{\mathcal{U}} \tag{10}$$

#### 4.4. Access Counter Creation Protocol

Once the credential issuance protocol has been executed, and before  $\mathcal{U}$  accesses the service for the first time,  $\mathcal{U}$  must establish the maximum number of uses  $n$  for which its credential can be accepted. This value is determined through the access counter creation protocol performed by  $\mathcal{U}$  and  $\mathcal{SP}$ , and comprises Steps 4, 5, and 6 from Figure 1 as depicted in Figure 2.

The detailed steps for this protocol are the following:

1.  $\mathcal{U}$  currently has  $\delta_{\mathcal{AP}}$ ,  $H(S)$ ,  $sk_{\mathcal{U}}$ ,  $pk_{\mathcal{U}}$ ,  $n$ , and  $id_{\mathcal{U}}$
2.  $\mathcal{U}$  chooses  $b \in_{\mathbb{R}} \mathbb{F}_q^*$  as a factor to blind the values needed to obtain the anonymous credentials.
3.  $\mathcal{U}$  computes:

$$\begin{aligned} sk'_{\mathcal{U}} &= b \cdot sk_{\mathcal{U}} \\ pk'_{\mathcal{U}} &= sk'_{\mathcal{U}} \cdot P \\ \delta'_{\mathcal{AP}} &= b \cdot \delta_{\mathcal{AP}} \\ P' &= b \cdot P \\ pk'_{\mathcal{AP}} &= b \cdot pk_{\mathcal{AP}} \\ C' &= b \cdot H(S) \cdot P \end{aligned} \tag{11}$$

4.  $\mathcal{U}$  also computes a NI-Schnorr ZKP by choosing  $r' \in_{\mathbb{R}} \mathbb{F}_q^*$  and doing:

$$\begin{aligned} R' &= r' \cdot P \\ t_n &= (n + 1) \cdot H(id_{\mathcal{U}}) \cdot sk'_{\mathcal{U}} + r' \end{aligned} \tag{12}$$

5.  $\mathcal{U}$  sends  $\mathcal{SP}$  the identifier  $id_{\mathcal{U}}$ , the anonymous credentials  $\delta'_{\mathcal{AP}}$ ,  $pk'_{\mathcal{U}}$ ,  $pk'_{\mathcal{AP}}$ ,  $P'$ ,  $C'$ ,  $n$ , and the NI-Schnorr ZKP proof of the private key  $(R', t_n)$  by calling the Create function of  $\mathcal{SP}$  who, in turn, calls a Smart Contract with these values.
6. The Smart Contract verifies that  $pk'_{\mathcal{AP}}$  is the blinded value of  $pk_{\mathcal{AP}}$  by computing:

$$e(pk'_{\mathcal{AP}}, P) \stackrel{?}{=} e(pk_{\mathcal{AP}}, P') \tag{13}$$

7. The Smart Contract also verifies that the credentials are valid:

$$e(C' + pk'_{\mathcal{AP}}, \delta'_{\mathcal{AP}}) \stackrel{?}{=} e(P', pk'_{\mathcal{U}}) \tag{14}$$

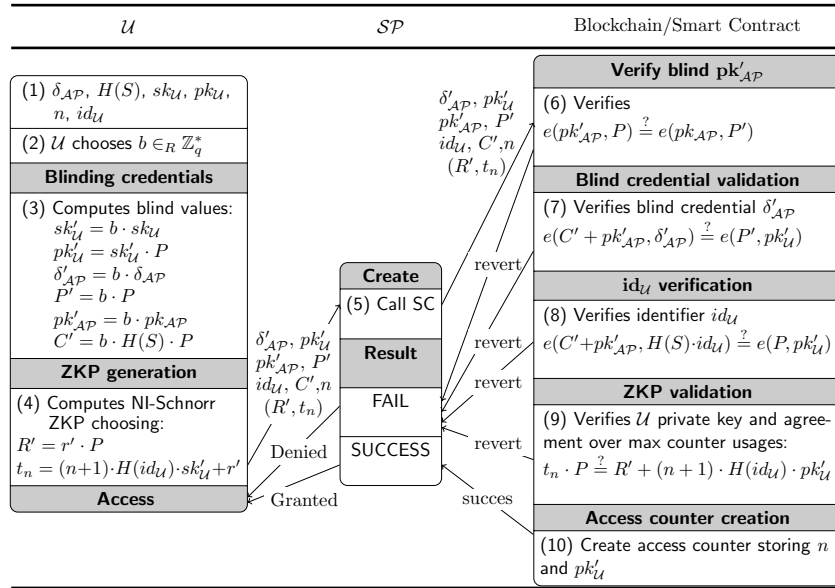
8. The Smart Contract verifies that  $id_{\mathcal{U}}$  is also valid by computing:

$$e(C' + pk'_{\mathcal{U}}, H(S) \cdot id_{\mathcal{U}}) \stackrel{?}{=} e(P, pk'_{\mathcal{U}}) \tag{15}$$

9. The Smart Contract verifies that  $\mathcal{U}$  possesses the private key checking the correctness of the NI-Schnorr ZKP as follows:

$$t_n \cdot P \stackrel{?}{=} R' + (n + 1) \cdot H(id_{\mathcal{U}}) \cdot pk'_{\mathcal{U}} \tag{16}$$

10. Finally, the Smart Contract persists on an array indexed by  $H(id_{\mathcal{U}})$  an object containing  $n$ , the maximum number of usages, the counter of usages (initialised to 0), and  $pk'_{\mathcal{U}}$ , the blinded public key of the user.

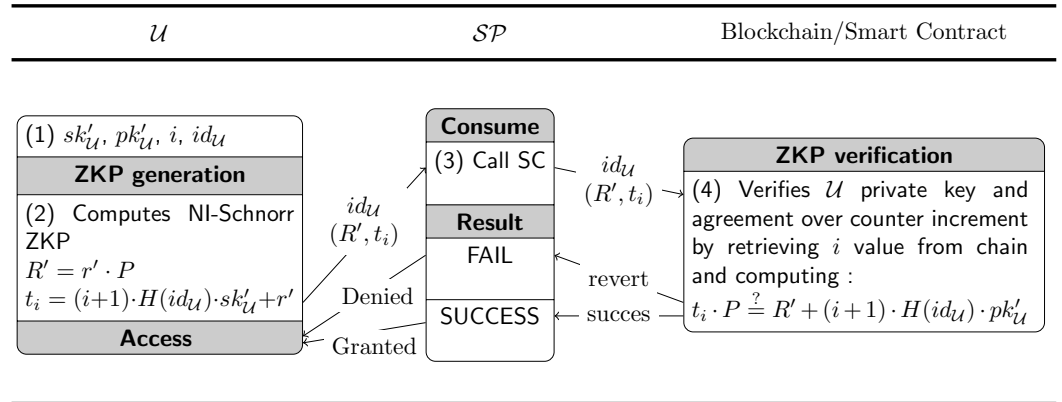


**Figure 2.** Data and call flow between  $\mathcal{U}$ ,  $\mathcal{SP}$  and the blockchain during the access counter creation protocol.

If the Smart Contract execution is successful, the access counter is ready to use; otherwise, execution is reverted and no counter is created.

#### 4.5. Access Counter Usage Protocol

Once the access counter creation protocol has been performed,  $\mathcal{U}$  can use the credentials  $n$  times executing the access counter usage protocol, depicted in Figure 3.



**Figure 3.** Message exchange between  $\mathcal{U}$  and  $\mathcal{SP}$  during the  $i$ -th iteration of the access counter usage protocol.

The details of the protocol are as follows:

1.  $\mathcal{U}$  currently has the secret key,  $sk'_{\mathcal{U}}$ , and the  $id_{\mathcal{U}}$ .
2. Using  $sk'_{\mathcal{U}}$ ,  $r'$ , and  $id_{\mathcal{U}}$ ,  $\mathcal{U}$  computes a NI-Schnorr ZKP by doing:

$$t_i = (i + 1) \cdot H(id_{\mathcal{U}}) \cdot sk'_{\mathcal{U}} + r' \tag{17}$$

3.  $\mathcal{U}$  sends  $id_{\mathcal{U}}$ ,  $R'$  and  $t_i$  to  $\mathcal{SP}$ , by calling the Consume service, and  $\mathcal{SP}$  calls a Smart Contract with these values.
4. The Smart Contract retrieves the object indexed by  $H(id_{\mathcal{U}})$  from storage. If it does not exist, it reverts the execution. Otherwise, it checks:

$$t_i \cdot P \stackrel{?}{=} R' + (i + 1) \cdot H(id_{\mathcal{U}}) \cdot pk'_{\mathcal{U}} \tag{18}$$



If the Smart Contract execution is successful,  $\mathcal{SP}$  could provide the service; otherwise, if the execution is reverted,  $\mathcal{SP}$  should deny the service.

#### 4.6. Pseudo-Code

This section provides the pseudo-code (Listing 1) of the implemented Smart Contract. Each formula in the previous sections has its own function  $TESTXX()$ . Each returns a Boolean value as the result of their evaluation. Note that the variable  $params$  is a *struct* with the parameters needed in the function. The *assert* function reverts execution of the Smart Contract if the evaluation is **false**.

```

storage bytes pkAP
struct record {
    integer max_value
    integer current_value
    bytes blinded_public_key_user
}
storage records array of record

constructor counter (params)
{
    pkAP = params.pkAP
}

function Create(params) {
    assert(not exist records[hash(params.idu)])
    assert(TEST13(params))
    assert(TEST14(params))
    assert(TEST15(params))
    assert(TEST16(params))
    create record r with:
        max_value = params.max_value
        current_value = 0
        blinded_public_key_user = params.blinded_public_
            ey_user
    store records[hash(params.idu)] = r
}

function Consume(params) {
    assert(exist records[hash(params.idu)])
    r = records[hash(params.idu)]
    assert(TEST18(params, r.blinded_public_key_user))
    assert(r.current_value < r.max_value)
    r.current_value++
    store records[hash(params.idu)] = r
}
}

```

**Listing 1.** Smart Contract pseudo-code.

## 5. Security Analysis

A security evaluation of the original authentication protocol can already be found in the reference article [4]. The security evaluation of the enhancements of the proposed protocol is provided in this section. Note that the checks are performed on the chain by using Smart Contracts, which offer integrity and reliability.

### 5.1. Design Decisions

I confirm that starting with step 0 indicates that this step is taken only once at the beginning and not repeated in subsequent iterations.

Three different choices to modify Formula (3) are presented in order to introduce  $n$  and  $i$ , denoting the maximum number of uses and the increment value, respectively, in the calculation of the NI-Scnorr ZKP.

The first option to introduce  $i$  in the computation of  $t_i$  is to take  $t$  and use it to compute each  $t_i$  by multiplying it by  $i$ . Formula (3) becomes  $t_i = i \cdot t = i \cdot (H(id_{\mathcal{U}}) \cdot sk'_{\mathcal{U}} + r')$ ,  $\forall i \in [0 \dots n]$ . The problem in this case is that when  $t_1$  is known, it is easy to forge  $t_2$  by calculating  $t_1 + t_1$ , and so on with  $t_3, \dots, t_n$ .

The second option to introduce  $i$  in the computation of  $t_i$  is to take  $t$  and multiply by  $i$  the term of  $t$  where  $r'$  appears. Formula (3) becomes  $t_i = h'_i \cdot sk'_{\mathcal{U}} + i \cdot r'$ ,  $\forall i \in [0 \dots n]$ . Note that, in this case

$$\begin{aligned} t_{i+1} &= (i + 2) \cdot R' + pk'_{\mathcal{U}} \cdot H(id_{\mathcal{U}}) \\ &= R' + (i + 1) \cdot R' + pk'_{\mathcal{U}} \cdot H(id_{\mathcal{U}}) \\ &= R' + t_i, \forall i \in [0 \dots n] \end{aligned}$$

the problem is knowing  $R'$  and adding it to  $t_i$ , anyone can then forge  $t_{i+1}$  and so on.

The third and final choice involves multiplying the term that contains the private key, Formula (3) becomes  $t_i = R' + i \cdot pk'_{\mathcal{U}} \cdot H(id_{\mathcal{U}})$ . Nobody can forge  $t_i$  without knowing the private key. This is the method chosen for this study.

Once the option has been chosen on how to introduce  $n$  and  $i$  into the ZKP, a problem appears when this value is equal to zero. The private key would be multiplied by zero and be rendered null. Therefore, in the computation of  $t_i$ , the multiplication factor becomes  $i + 1$  to avoid multiplication by 0.

All threats to the current protocol are based on the robustness of ZKP; therefore, this robustness has to be proved. From Formula (8), the proof is  $(R', t_i)$  with public values  $i$ ,  $id_{\mathcal{U}}$ , and private  $sk'_{\mathcal{U}}$  is calculated as:

$$\begin{aligned} R' &= r' \cdot P \\ t_i &= (i + 1) \cdot H(id_{\mathcal{U}}) \cdot sk'_{\mathcal{U}} + r' \end{aligned}$$

As long as  $sk'_{\mathcal{U}}$  and  $r'$  are private to  $\mathcal{U}$ , nobody except  $\mathcal{U}$ , can forge a ZKP. Because of the discrete logarithm problem (DLP) in additive groups, it is not feasible to compute  $sk'_{\mathcal{U}}$  nor  $r'$  from  $pk'_{\mathcal{U}}$  and  $R'$ , respectively.

With these results, the robustness of the protocol is proven in the presence of threats found. The main threats are described as:

- The use of an arbitrary  $id_{\mathcal{U}}$  where a malicious  $\mathcal{U}$  tries to forge a counter without the authentication creation phase.
- A misbehaviour of  $\mathcal{SP}$  in the counter creation phase that attempts to create a counter with an incorrect maximum usage value.
- A replay attack performed by  $\mathcal{SP}$ .
- A misbehaviour of  $\mathcal{SP}$  trying to forge an invalid  $t_i$ .

### 5.2. Use of Arbitrary $id_{\mathcal{U}}$ without Creation

Counter data are stored on the chain, and only the Smart Contract can add this data after successfully passing all the checks. To use the counter, it has to be created before. This ensures that no one can use an arbitrary counter before creation, where credentials checks are performed.

### 5.3. A Misbehaviour of $\mathcal{SP}$ in the Counter Creation Phase

By design, the Smart Contract verifies  $t_n$ . Since it is computed as:

$$t_n = (n + 1) \cdot H(id_U) \cdot sk_U + r'$$

it can only be provided by someone in possession of the private key,  $r'$  approving the value of  $n$ .

#### 5.4. A Replay Attack

By associating each  $t_i$  with a counter value  $i$  and including it in the computation, replay attacks can be prevented because a valid proof  $t_i$  for a counter value of  $i$  becomes invalid for a counter value of  $i + 1$ . Each  $t_i$  is invalidated after use. Only those with a private key can compute a valid  $t_{i+1}$  for the counter value of  $i + 1$ . Only the Smart Contract can alter the value of a counter after successfully passing all checks.

#### 5.5. A Misbehaviour of $\mathcal{SP}$

Enforced by the Smart Contract, without knowing the private key, nobody should be able to forge a valid  $t_i$ . Therefore, a malicious  $\mathcal{SP}$  cannot increment the counter without the user's agreement that provides the corresponding  $t_i$ .

### 6. Use Case in Detail

As noted in the Introduction, a government bill could grant users ( $\mathcal{U}$ ) earning below the minimum wage thirty free underground trips per month.

In this use case,  $\mathcal{AP}$  could be an entity like a tax agency or bank, and  $\mathcal{SP}$  could be the underground service company. The scope  $S$  is defined as the string consisting of the first letter of the service, the number of the month, and the year in which the users are entitled to the service. "U0123" would be the scope for January 2023, "U0223" for February 2023, and so on.

The steps for a given month are as follows:

1.  $\mathcal{U}$  generates their keys  $sk_U$  and  $pk_U$  that could always be the same or that could be changed every month. This does not affect the protocol.
2. In January,  $\mathcal{U}$  obtains (using Formula (1)) the unique identifier  $id_U$  for the scope  $S$  equal to "U0123".
3.  $\mathcal{U}$  refers then to  $\mathcal{AP}$  and identifies themselves, proving that in the previous month, it had income below the minimum wage, and obtains the  $\delta_{\mathcal{AP}}$  for the scope "U0123" that entitles them to thirty underground trips in January.  $\mathcal{AP}$  obtains  $\delta_{\mathcal{AP}}$  using Formula (2), binding the attribute to  $sk_U$ .
4. The  $\delta_{\mathcal{AP}}$  attribute is known by  $\mathcal{AP}$  that can associate it with a user to control whether it has already been given to them or not.
5. Before usage,  $\mathcal{U}$  blinds  $\delta_{\mathcal{AP}}$  and performs the calculations in Formula (2) to obtain the anonymous credentials from the scope "M0103" which entitles them to thirty underground trips during January 2023. These anonymous credentials are:  $\delta'_{\mathcal{AP}}$ ,  $pk'_U$ ,  $pk'_{\mathcal{AP}}$ ,  $P'$ , and  $C'$ .
6. To begin using trips,  $\mathcal{U}$  should obtain a counter. To do so,  $\mathcal{U}$  presents its anonymous credentials calculated in the previous step,  $id_U$ , and the proof of possession of their own private key. This proof is  $t_i$  and is calculated using Formula (8) with  $i$  equal to the number of trips, that is,  $t_{30}$ . This proof can be checked with Formula (9) given the anonymous credentials and  $id_U$ . These checks are implemented in the Create method of the Smart Contract.
7. With this data,  $\mathcal{SP}$  calls the Create function of the Smart Contract to create a counter valid for thirty accesses, no more, no less.
8. Only  $\mathcal{U}$  can use this counter, using  $t_i$  each time, where  $i$  is the number of uses stored in the Smart Contract, along with  $id_U$  and their public key. If  $\mathcal{U}$  keeps this  $t_i$ s secret, only they can consume the ticket each time. This prevents anyone who does not know the private key from incrementing the counter.
9. Once the ticket has been created,  $\mathcal{U}$  must calculate the  $t_i$  that corresponds to the current value of the counter and send it to  $\mathcal{SP}$  along with its  $id_U$  to be able to use

- it. Checks of these values are implemented in the Consume method of the Smart Contract, and the counter should be less than or equal to the maximum number.
10. When  $\mathcal{SP}$  receives a request to consume a counter, it only needs to call the Consume method of the Smart Contract with the parameters of the request. If the execution is successful, it grants access to the underground service; otherwise, it should deny access to the service and revert the transaction. The problem could be:
    - $t_i$  being incorrect
    - an attempted fraud
    - the ticket is sold out

## 7. Conclusions and Future Work

This article proposes a novel method for managing usage counters within a system designed to ensure user anonymity. The core concept revolves around establishing a predetermined maximum number of uses at the time of counter creation, with explicit agreement from the user. Each subsequent use also needs user consent, ensuring ongoing transparency and control.

To facilitate this secure and auditable process, all checks and validations are handled by a Smart Contract deployed on the Ethereum blockchain. This decentralised storage mechanism guarantees the immutability and public visibility of transaction records, fostering trust and fairness within the system.

To validate the practicality and efficiency of the proposed approach, a prototype implementation has been conducted on a private Ethereum blockchain. Preliminary experimental results denote that the proposed method can be executed with reasonable computational costs, paving the way for its potential deployment in real-world scenarios requiring both anonymity and usage tracking, which will be the focus of the next issue.

For the purpose of this study, the scenario has been simplified by assuming that the service provider ( $\mathcal{SP}$ ) deploys the Smart Contract initialised with the attribute provider's ( $\mathcal{AP}$ ) public key. This assumption has been made because  $\mathcal{SP}$  is the entity that directly uses the Smart Contract for authentication. However, it is important to note that this is not mandatory. In principle, any actor could deploy the Smart Contract. If a different party were to take on this role, an additional mechanism would be required to publicly disseminate the contract address. This may be explored in future research endeavours.

Another issue to be addressed in the future is the introduction of temporal restrictions in the use of ZKP to permit sequenced uses without user intervention, targeting periodic renewals, for example.

Replacing the usage of EIP-196 and EIP-197, which work with the elliptic curve `alt_bn128`, with EIP-2537 [27], which works with BLS12-381 curve, thus offering more security [28] and less gas consumption could also be explored in the future. At the time of writing this paper, EIP-2537 is in the peer review phase and not available on Ethereum Main Net.

**Author Contributions:** Investigation, F.G.-G.; Writing—original draft, F.G.-G.; Writing—review & editing, F.G.-G., J.H.-J. and A.D.J.; Supervision, J.H.-J. and A.D.J. All authors have read and agreed to the published version of the manuscript.

**Funding:** This research received no external funding.

**Institutional Review Board Statement:** Not applicable.

**Informed Consent Statement:** Not applicable.

**Data Availability Statement:** No new data were created or analyzed in this study.

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

## References

1. Vagle, J.L.; Bellovin, S.; Douglas, E.; Gandlur, S.; Hogan, K.; Listokin, S.; Ormerod, P.; Richards, N.; Seeman, J.; Tessono, C.; et al. Privacy's Commodification and the Limits of Antitrust. *Ark. Law Rev.* **2024**, *77*, 51.
2. Camenisch, J.; Camenisch, J.; Around, C.; Camenisch, J.; Marit, P. Preserving Attribute-Based Credentials. Concepts Around Privacy-Preserving Attribute-Based Credentials to Cite This Version: HAL Id: hal-01276046 Concepts around Privacy-Preserving Attribute-Based Credentials. 2016. Available online: <https://hal.science/hal-01276046> (accessed on 1 July 2024).
3. Camenisch, J.; Dubovitskaya, M.; Enderlein, R.R.; Lehmann, A.; Neven, G.; Paquin, C.; Preiss, F.S. Concepts and languages for privacy-preserving attribute-based authentication. In Proceedings of the 3rd IFIP WG 11.6 Working Conference of the Policies and Research in Identity Management (IDMAN 2013), London, UK, 8–9 April 2013. [CrossRef]
4. Garcia-Grau, F.; Herrera-Joanmartí, J.; Dorca Josa, A. Attribute Based Pseudonyms: Anonymous and Linkable Scoped Credentials. *Mathematics* **2022**, *10*, 2548. [CrossRef]
5. Wood, G. Ethereum: A Secure Decentralised Generalised Transaction Ledger. *Ethereum Proj. Yellow Pap.* **2014**, *151*, 1–32.
6. 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), Innsbruck, Austria, 6–10 May 2001; Pfitzmann, B., Ed.; Springer: Berlin/Heidelberg, Germany, 2001; pp. 93–118.
7. Bogatov, D.; Caro, A.D.; Elkhiyaoui, K.; Tackmann, B. Anonymous Transactions with Revocation and Auditing in Hyperledger Fabric; Cryptology ePrint Archive, Report 2019/1097. 2019. Available online: <https://eprint.iacr.org/2019/1097> (accessed on 1 July 2024).
8. Verheul, E.R. Self-Blindable Credential Certificates from the Weil Pairing. In Proceedings of the Advances in Cryptology (EUROCRYPT 2001), Innsbruck, Austria, 6–10 May 2001; Boyd, C., Ed.; Springer: Berlin/Heidelberg, Germany, 2001; pp. 533–551.
9. Zhang, F.; Safavi-Naini, R.; Susilo, W. An Efficient Signature Scheme from Bilinear Pairings and Its Applications. In Proceedings of the Public Key Cryptography (PKC 2004), Singapore, 1–4 March 2004; Bao, F., Deng, R., Zhou, J., Eds.; Springer: Berlin/Heidelberg, Germany, 2004; pp. 277–290.
10. Buterin, V.; Reitwiessner, C. EIP-196: EIP-196: Precompiled Contracts for Addition and Scalar Multiplication on the Elliptic Curve alt\_bn128. Ethereum Improvement Proposals, no. 196. February 2017. Available online: <https://eips.ethereum.org/EIPS/eip-196> (accessed on 1 July 2024).
11. Buterin, V.; Reitwiessner, C. EIP-197: Precompiled Contracts for Optimal ate Pairing Check on the Elliptic curve alt\_bn128. Ethereum Improvement Proposals, no. 197. February 2017. Available online: <https://eips.ethereum.org/EIPS/eip-197> (accessed on 1 July 2024).
12. Granger, R.; Hess, F.; Oyono, R.; Thériault, N.; Vercauteren, F. Ate pairing on hyperelliptic curves. In Proceedings of the 26th Annual International Conference on the Theory and Applications of Cryptographic Techniques—Advances in Cryptology (EUROCRYPT 2007), Barcelona, Spain, 20–24 May 2007; Proceedings 26; Springer: Cham, Switzerland, 2007; pp. 430–447.
13. Li, X.; Niu, J.; Gao, J.; Han, Y. Secure electronic ticketing system based on consortium blockchain. *KSII Trans. Internet Inf. Syst. (TIIS)* **2019**, *13*, 5219–5243.
14. Jie, T.H.; Alduais, N.A.M.; dan Teknologi Maklumat, F.S.K. Development of IoT-Based E-ticket Selling and Management System with QR code Scanner (Tickets.now). *Appl. Inf. Technol. Comput. Sci.* **2023**, *4*, 1907–1926.
15. Verslype, K.; Decker, B.D.; Naessens, V.; Nigusse, G.; Lapon, J.; Verhaeghe, P. A privacy-preserving ticketing system. *Lect. Notes Comput. Sci.* **2008**, *5094*, 97–112. [CrossRef]
16. Feulner, S.; Sedlmeir, J.; Schlatt, V.; Urbach, N. Exploring the use of self-sovereign identity for event ticketing systems. *Electron. Mark.* **2022**, *32*, 1759–1777. [CrossRef] [PubMed]
17. Sedlmeir, J.; Smethurst, R.; Rieger, A.; Fridgen, G. Digital identities and verifiable credentials. *Bus. Inf. Syst. Eng.* **2021**, *63*, 603–613. [CrossRef]
18. Preece, J.D.; Easton, J.M. Blockchain Technology as a Mechanism for Digital Railway Ticketing. *IEEE* **2019**, *12*, 3599–3606. [CrossRef]
19. Borges, R.; Sebe, F. A Construction for Providing Reusability to Mobile Phone-Based e-Tickets. *IEEE Access* **2020**, *8*, 101386–101397. [CrossRef]
20. Quercia, D.; Hailes, S. MOTET: Mobile Transactions using Electronic Tickets. In Proceedings of the 1st International Conference on Security and Privacy for Emerging Areas in Communications Networks (SECURECOMM'05), Athens, Greece, 5–9 September 2005; pp. 374–383. [CrossRef]
21. 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]
22. Heydt-Benjamin, T.S.; Chae, H.J.; Defend, B.; Fu, K. Privacy for public transportation. In Proceedings of the International Workshop on Privacy Enhancing Technologies, Cambridge, UK, 28–30 June 2006; Springer: Cham, Switzerland, pp. 1–19.
23. Stockburger, L.; Kokosioulis, G.; Mukkamala, A.; Mukkamala, R.R.; Avital, M. Blockchain-enabled decentralized identity management: The case of self-sovereign identity in public transportation. *Blockchain Res. Appl.* **2021**, *2*, 100014. [CrossRef]
24. 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. [CrossRef] [PubMed]

25. Chien, J.; Ho, L.; Lin, C.Y. An Anonymous On-Street Parking Authentication Scheme via Zero-Knowledge Set Membership Proof. *arXiv* **2021**, arXiv:2108.03629.
26. Sung, H.M.; Chen, T.; Tseng, H.C.; Prayogo, B.; Lin, J.Y.; Hung, Y.P. akaTick: Hybrid Mobile E-Ticketing System Based on Non-Fungible Tokens. In Proceedings of the 2023 IEEE International Conference on Metaverse Computing, Networking and Applications (MetaCom), Kyoto, Japan, 26–28 June 2023; pp. 686–687. [[CrossRef](#)]
27. Vlasov, A.; Olson, K.; Stokes, A.; Sanso, A. EIP-2537: Precompile for BLS12-381 Curve Operations [DRAFT]. Ethereum Improvement Proposals, no. 2537. February 2020. Available online: <https://eips.ethereum.org/EIPS/eip-2537> (accessed on 1 July 2024).
28. Menezes, A.; Sarkar, P.; Singh, S. Challenges with assessing the impact of NFS advances on the security of pairing-based cryptography. In Proceedings of the International Conference on Cryptolog, Kuala Lumpur, Malaysia, 1–2 December 2016; Springer: Cham, Switzerland, 2016; pp. 83–108.

**Disclaimer/Publisher’s Note:** The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.