Transferable, Auditable and Anonymous Ticketing Protocol

Pascal Lafourcade, Dhekra Mahmoud, Gael Marcadet, Charles Olivier-Anclin

Transferable, Auditable and Anonymous Ticketing Protocol

Pascal Lafourcade
Université Clermont-Auvergne, CNRS,
Clermont-Auvergne-INP, LIMOS
Clermont-Ferrand, France

Dhekra Mahmoud
Université Clermont-Auvergne, CNRS,
Clermont-Auvergne-INP, LIMOS
Clermont-Ferrand, France

Gael Marcadet
Université Clermont-Auvergne, CNRS,
Clermont-Auvergne-INP, LIMOS
Clermont-Ferrand, France

Charles Olivier-Anclin
be ys Pay and Université Clermont-Auvergne, CNRS,
Clermont-Auvergne-INP, LIMOS and LIFO, Université
d'Orléans, INSA Centre Val de Loire
Clermont-Ferrand and Bourges, France

ABSTRACT

Digital ticketing systems typically offer ticket purchase, refund, validation, and, optionally, anonymity of users. However, it would be interesting for users to transfer their tickets, as is currently done with physical tickets. We propose Applause, a ticketing system allowing the purchase, refund, validation, and transfer of tickets based on trusted authority, while guaranteeing the anonymity of users, as long as the used payment method provides anonymity. To study its security, we formalise the security of the transferable E-Ticket scheme in the game-based paradigm. We prove the security of Applause computationally in the standard model and symbolically using the protocol verifier ProVerif. Applause relies on standard cryptographic primitives, rendering our construction efficient and scalable, as shown by a proof-of-concept. In order to obtain Spotlight, an auditable version, proved to be secure, users will remain anonymous except for a trusted third party, which will be able to disclose their identity in the event of a disaster.

CCS CONCEPTS

• **Security and privacy → Privacy-preserving protocols**.

KEYWORDS

Ticketing System, Protocol, Auditability, Anonymity, Transfer

ACM Reference Format:
Pascal Lafourcade, Dhekra Mahmoud, Gael Marcadet, and Charles Olivier-Anclin. 2018. Transferable, Auditable and Anonymous Ticketing Protocol. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 17 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

Electronic tickets (*E-Ticket*) have become the standard. The rationale behind the digitization of the ticket industry is both practical and economical. More sales can be achieved by allowing users to purchase tickets from anywhere. The practical aspect of ticket digitization entails significant drawbacks, resulting in a negative impact on the second-hand market and the protection of users' privacy. In a world where privacy is a central concern, it is essential to preserve the multiple facets of a paper ticket, such as the right to trade them, and ensure full confidence in their validity. However, the original property of non-replicable paper tickets is often lost with the development of E-Tickets. A formal security model for electronic ticketing is needed to ensure their security, and a system that combines the best properties of both worlds is necessary.

The research has focused on multiple cryptographic technologies with properties similar to their physical counterparts. The subject of *e-cash* [3, 4, 49], *e-coupon* [10, 32, 38] (e-coupon are similar to e-cash with items chosen when redeeming an e-coupon remains unknown to the service provider) or *n-times anonymous authentication* [48, 7] appears to be closely associated with our issue. Even though they have been studied in the literature, their design makes them incompatible with being used as a ticketing system. In numerous electronic cash systems [3, 4, 49], the practice of double-spending, wherein a coin is spent twice, is mitigated by a central bank that maintains a record of the spent coins. When receiving a coin from a merchant, the bank checks to ensure that it does not belong to the list of spent coins. Therefore, the bank is able to detect if a coin has been double-spent, and can compensate the scammed merchant. This mechanism, however, is not suitable to construct a desirable ticketing system, since a scam would be left undetected by a second merchant accepting the same coin until they both reach the bank. This fraud can be generalised to transfer of coins in transferable e-cash. Putting it in different terms, cheaters could resell twice a ticket and this would be left undetected until they try to access the event. On the other hand, in *e-coupon* systems [10, 32, 38], the transfer may appear unfavorable or even atypical, as a coupon has been issued by a merchant to a specific customer with the intention of gaining their loyalty, whereas *n-times anonymous authentication* [48], cannot be safely transferred. With e-tickets, it is imperative to guarantee the validity of a ticket at any time in order to avoid double-spent tickets. This primary concern renders every electronic cash system that prevents double-spending at coin deposit irrelevant to our issue. Additionally, a coin is validated by the bank during the deposit process in e-cash [4]. In the case of e-ticketing, a ticket can be validated by one of the many terminals. All of these disparities substantiate the necessity for a specific approach to electronic ticketing. E-ticketing systems have

Conference'17, July 2017, Washington, DC, USA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/XXXXXXX.XXXXXXX

been studied both in the industry [2, 42] for practical purposes and in academia for understanding social behaviors regarding ticket resale and the incentives behind this process [29, 12]. The majority of the present ticketing systems that have been developed by the industry are centered on the "standard" functionality of electronic tickets, which entails the provision and validation of tickets. These systems are not designed to ensure secure ticket transfer. Hence, an honest client has high chances of buying a duplicated ticket from a malicious user. More advanced systems, attempting to address this issue, provide authentication via a distributed architecture, such as the blockchain [2, 42]. Even if ticket validity is now ensured during a transfer (by checking if the ticket is valid in the blockchain), it moves away from the current e-ticketing system organization where the tickets are stored in a database. These methods however, are in opposition to the currently centralized event organization and implies a larger energy consumption [13, 46]. Users' anonymity in e-ticketing systems has not been considered in the latest protocols [2, 24, 37, 42]. Nevertheless, anonymity is guaranteed by physical tickets, and it remains crucial to prevent the event organizers from collecting the identity of ticket purchasers unless necessary. Blockchain-based solutions can serve this purpose [2, 42], but for the aforementioned argumentation, we exclude them from our investigation.

Contribution. Considering the above problems and the lack of existing formalism, we design an e-ticket scheme with proven security. It features three central aspects as it is *centralised*, *transferable* and *anonymous*, with the latter property that can be mitigated by *auditable* feature at need. We discuss the requirements of such a protocol, its capabilities, and limitations, and encompass it in a model. We introduce the first security model for *E-Ticket Scheme* (ETS). The security is formalised through five experiments, modeling *unforgeability*, *ticket privacy*, *anonymity of users*, split in two experiments, and *no-double-spending*, preventing to execute a refund, transfer or validate twice with the same ticket.

Our *provably secure E-Ticket* scheme is called Applause. It allows users to purchase, refund, validate tickets, but also transfer their tickets to another user. During all the interactions, depicted in Fig. 1, users' anonymity from the event organiser point-of-view is ensured. We propose a protocol addressing all the above-mentioned properties. Its security is proven based on computational proofs, both in the random oracle and standard model. Moreover, we have used the protocol verifier tool ProVerif [6] to provide more security guarantees, see Fig. 2 for a description of ensured properties.

Payments can be made during at least three steps of our process: the purchase, the refund, and the transfer of tickets. Traditional payment does not guarantee anonymity, therefore, using it in our construction would trivially break the anonymity of users. For instance, the most standard payment protocol EMV [17] does not protect privacy. In our protocol, the payment is modelled as a generic cryptographic building block, requiring *anonymity*, a mandatory assumption only required to ensure full anonymity. More precisely, our protocol is as anonymous as the payment method in use. We stress that it is not specific for our protocol but can be applied for every protocol ensuring anonymity and involving payment. Alternative payment protocols that guarantee anonymity of users exist, ranging from simple anonymous payment method [40] to more advanced constructions [15, 33], can be used in our protocol. To reveal the identity of users, reaching *auditability*, we present Spotlight, an extended version of Applause, where the identity of *every* users can be revealed *only* by a third-party called the judge, employing certificates that can be randomised along with the certified keys.

Related Work. Most of the production-ready deployed systems guarantee the "standard" functionality of ticket payment and delivery. Our system provides users with the ability to transfer a ticket to another user, while maintaining anonymity, in addition to the standard functionalities. All existing systems that allow for additional functionalities that we identify rely on blockchains [2, 42]. Blockchain-based ticketing systems can easily achieve transfer of a ticket by checking if the ticket is still valid on the blockchain, and can achieve anonymity by the design of the blockchain. However, the blockchain requires the upkeep of a distributed ledger and numerous signatures, which means that numerous servers are needed to safeguard the validity of the network. Low-consumption and optimised computation has been a keystone of cryptography. As shown in [13, 46], blockchains are more computationally demanding than centralised design, but also require large-scale procedures. Our system is centralised to enhance efficiency and align with the existing topology of event organization.

Previous works, initiated by [30] in 2001 and followed up by a number of papers, essentially evaluating either the practicality [26], the interface design [50] or the security [47] of existing systems in the public transport. Subsequent works have proposed a novel ticketing system. They can be divided into three categories. The first category focuses on the design of a ticketing system [24, 37], ensuring the validity of a ticket, without considering privacy. The second category aims to guarantee the privacy of the user, however it does not permit transfer or auditability. In [28], they study the possibility of a ticketing system with privacy of users, including the billing step. The authors of [22, 23] have studied the possibility of using RFID and NFC while ensuring privacy of users, using unlinkable certified tokens [35] for public transport ticketing or anonymous credentials [25] for general purposes tickets. Furthermore, construction of [25] takes a few seconds whereas a full execution of our constructions requires at most 100 milliseconds and does not directly scale with the number of emitted tickets. Unlike [25], we include the payment process in the protocol as well. The third line of research proposes ticketing systems based on a distributed ledger, which are less efficient compared to a centralised setting, due to the number of involved servers and communication time. The first paper to propose such a system is presented in [31], which is based on blockchain. Recently, a new ticketing system has been proposed in [51] based on NFT. For practical and efficiency reasons depicted above, our work moves away from the distributed approach.

This work is compared to publications in less specialized areas, such as *E-cash* [3, 4, 49] and *e-coupon* [10, 32, 38], in which a trusted authority generates tokens attached to a value. Due to double-spending enforcement checked only by the bank, followed by a compensation process, they miss the guarantees of ticket validity upon transfer. On the other hand, *n-times anonymous authentication*[48, 7] allows a limited number of authentications before leaking user's identity, limiting the ticket purchasing, as well as the interest

of this approach for ticketing. In [7], restrictions of the concept are augmented by time frames that further limit the authentication process, yet it falls short in ensuring transfer guarantees.

Considerations could also be given to *Anonymous Credentials* [9, 11, 36] or *Attribute-Based Signatures* [5, 34]. These mechanisms are dedicated for authentication based on predicate matching, identifying information about the signer. Yet again, transferability remains unaddressed as a standalone aspect in this particular work.

2 ELECTRONIC TICKETS SYSTEMS

We move away from the decentralised approach used in previous work to focus on the currently used and centralised architecture. Our system divides the ticket handling into three phases and can be increased with a judges for auditability.

Architecture. A *user* $\mathcal{U}$ willing to purchase a ticket for an event, contacts the *ticket distributor* $\mathcal{D}$, which issues a ticket tk to $\mathcal{U}$ in exchange of a payment. Once $\mathcal{U}$ holds a ticket, it can authenticate itself to a *validator* terminal $\mathcal{V}$ in order to get access to the designated event. In some cases, an user $\mathcal{U}_1$ holding a ticket might not be able to benefit from its purchase, in such cases our protocol offers two options: a *refund*, or a *transfer* of its ticket to another user *e.g.,* to $\mathcal{U}_2$. The refund is proceeded between user $\mathcal{U}_1$ and the ticket distributor $\mathcal{D}$. The transfer scenario encompasses both the cases where $\mathcal{U}_1$ sells its ticket to another user $\mathcal{U}_2$ or offers it. The latter consisting of the same transfer protocol without the payments. The ticket transfer is modeled as an interaction between $\mathcal{U}_1$, $\mathcal{U}_2$, and a *transfer authority* $\mathcal{T}$ acting as a guaranty of the exchange. $\mathcal{T}$ ensures the validity of the ticket to $\mathcal{U}_2$, prevents $\mathcal{U}_1$ to resell a ticket for profit by controlling the price, but also prevents $\mathcal{U}_2$ to obtain a ticket without paying it. While ticket sale is a fairly straightforward process, its transfer can be achieved through multiple scenarios. It has been shown in [19] that a fair transfer between two users is impossible without a third party. Also see Appendix B for further considerations.

Note that the transfer authority $\mathcal{T}$ has been modeled as an independent entity but can be combined with the ticket distributor into a single entity without any loss of security. To make the designation of the ticket receiver possible, we assume a communication between $\mathcal{U}_1$ and $\mathcal{U}_2$ before the transfer of the ticket, allowing them to exchange keys. Last, to attend to an event, $\mathcal{U}$ has to validate a ticket against a validator $\mathcal{V}$ through an anonymous validation. At any time during the process, the *Judge* $\mathcal{J}$ can open a tickets to recovers the associated user's identity.

Anonymity and Auditability. Anonymity of users is ensured during every interaction with the system (*i.e.,* the ticket distributor $\mathcal{D}$, the transfer authority $\mathcal{T}$ and the validator $\mathcal{V}$) and all over the process. During the protocol, either authentication is not required or users authenticate themselves using randomised identities *i.e.,* randomised keys. If desired, our protocol Applause can be turned into an auditable version called Spotlight, where an external authority refers as the *judge* denoted $\mathcal{J}$, recovers the identity of $\mathcal{U}$. The auditability setting remains consistent with the one presented for *Auditable Anonymous Credentials* in [11]. In a nutshell, $\mathcal{U}$ provides a certificate attesting the validity of its randomised key to $\mathcal{D}$, $\mathcal{T}$ or $\mathcal{V}$. Given such a certificate, kept on a record, $\mathcal{J}$ retrieves the original key, hence the identity of $\mathcal{U}$. The anonymity still holds against $\mathcal{D}, \mathcal{T}$ or $\mathcal{V}$ as the certificate is randomised with the keys. If the judge is compromised, only the users' anonymity would be at risk. The tickets remain valid, ensuring that honest individuals can utilize their tickets without any issues. A compromised judge $\mathcal{J}$ would not have the capability to forge new tickets or alter the validity status of previously issued tickets. The purpose of auditability is not to prevent double spending, but rather to fulfill legal mandates concerning participant identity recovery. This aspect is crucial in unforeseen circumstances such as force majeure or disasters.

Validation Setting. Large scale events often result in an overloaded network due to the number of attendees. Multiple terminals are needed to validate the tickets simultaneously. Hence, communications coming to and from the validation terminals are limited in size and number. One would like to assume that the validator is offline after an initial setup. As a single ticket should be valid for any terminal, and without communication between the terminals, the same ticket could be accepted for each of them, constituting a forgery for any ticketing system. Hence, the validators must be online and communicate. In our case, only a single group element needs to be sent between a terminal and the central server for each approved verification. For efficiency, we rely on a central authority instead of a distributed ledger to agree on valid tickets at time t. A validation protocol has to ensure anonymity of users, and thus cannot authenticate the user. However, an adversary able to relay all communications between a legitimate user and a validator, can easily perform what is called a *relay attack* [43], where the adversary can have the access granted without paying a ticket, by simply blocking and sending all messages sent by a legitimate user to the validator, in its own name. As a result, the validator grants the access to the adversary instead of the legitimate user. Badly, neither the user or the validator can notice this attack. To prevent this issue, we have chosen to construct a validation protocol based on a physical channel (such as QRcode [27] scanned by the validator or Bluetooth's password shared verbally [39]) during the last step of the protocol to prevent relay attacks by an adversary. $\mathcal{U}$ gets a token, compared through this channel to the token sent by $\mathcal{V}$.

3 E-TICKET SCHEME MODEL

Most ticketing systems are designed to allow the sale of a seat at an event, with each seat being associated with a metadata such as the seat number. This can be more general, and we consider a scenario where tickets are characterised by an *event identifier* $\mathsf{ide} \in \mathsf{ID}_{\mathcal{E}} \subset \mathbb{N}$ and by a *serial number* $\mathsf{idp} \in \mathsf{ID}_{\mathcal{P}} \subset \mathbb{N}$. The set of identifiers and the set of serial numbers referred to event and seat number, and are assumed publicly known. Below we define an *E-Ticket Scheme* based on a security parameter 1^λ. Note that through this paper, all algorithms are assumed to run in probabilistic polynomial-time (PPT). By $\mathcal{A}^{\mathcal{O}}$, we denote the adversary having access to a set of oracles $\mathcal{O}$. By $\mathsf{P}\langle \mathsf{E}_1(i_1), .., \mathsf{E}_n(i_n)\rangle \rightarrow \mathsf{E}_1(o_1), .., \mathsf{E}_n(o_n)$; we denote the protocol P played between parties E_j, taking as input i_j and outputting o_j. For simplicity, we may omit a party in the output part of the notation if the party does not produce output. .

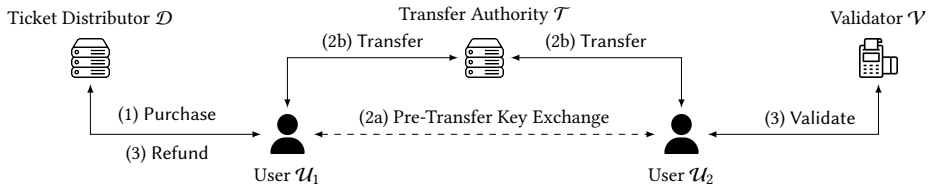


Figure 1: Representation of the ETS model.

	Unforgeability	Secrecy	No-double-spending	Anonymity	Auditability
Applause	✓	✓	✓	✓	×
Spotlight	✓	✓	✓	Partial	✓

Figure 2: Security properties.

*Definition 3.1 (*ETS*).* An *E-Ticket Scheme* Π = (DKeyGen, TKeyGen, VKeyGen, UKeyGen, Purchase, Refund, Transfer, Validate) is a tuple of PPT algorithms:

$\mathsf{DKeyGen/TKeyGen/VKeyGen/UKeyGen}(1^\lambda) \to (\mathsf{sk}, \mathsf{pk})$: Given a security parameter 1^λ, outputs a key pair.

$\mathsf{Purchase}\langle\mathcal{U}(\mathsf{sk}_\mathcal{U}, (\mathsf{ide}, \mathsf{idp})), \mathcal{D}(\mathsf{sk}_\mathcal{D}, st)\rangle \to \mathcal{U}(\mathsf{tk}), \mathcal{D}(b, (\mathsf{ide}, \mathsf{idp}), st)$: User $\mathcal{U}$ purchases the ticket identified by (ide, idp), to the ticket distributor $\mathcal{D}$. The user obtains the ticket tk, and $\mathcal{D}$ updates *st* and returns a success bit *b* and (ide, idp).

$\mathsf{Refund}\langle\mathcal{U}(\mathsf{sk}_\mathcal{U}, \mathsf{tk}), \mathcal{D}(\mathsf{sk}_\mathcal{D}, st)\rangle \to \mathcal{U}(b), \mathcal{D}(b, \mathsf{tk}, st)$: Given a ticket tk and its key $\mathsf{sk}_\mathcal{U}$, $\mathcal{U}$ asks the ticket distributor $\mathcal{D}$ for a refund. Both entities output a success bit *b*, while $\mathcal{D}$ additionally updates *st* and returns tk.

$\mathsf{Transfer}\langle\mathcal{U}_1(\mathsf{sk}_{\mathcal{U}_1}, \mathsf{pk}_{\mathcal{U}_2}, \mathsf{tk}_1), \mathcal{T}(\mathsf{sk}_\mathcal{T}, st), \mathcal{U}_2(\mathsf{sk}_{\mathcal{U}_2}, \mathsf{pk}_{\mathcal{U}_1}, (\mathsf{ide}, \mathsf{idp}))\rangle \to \mathcal{U}_1(b), \mathcal{T}(b, (\mathsf{ide}, \mathsf{idp}), \mathsf{tk}_1, st), \mathcal{U}_2(\mathsf{tk}_2)$: The transfer protocol allows $\mathcal{U}_1$ owning a ticket tk_1, the public key of $\mathcal{U}_2$ and key $\mathsf{sk}_{\mathcal{U}_1}$, to transfer it tk_2 holding the public key of $\mathcal{U}_1$ and a key $\mathsf{sk}_{\mathcal{U}_2}$ relying on $\mathcal{T}$ inputting $\mathsf{sk}_\mathcal{T}$ and *st*. As a result, $\mathcal{U}_1$ and $\mathcal{T}$ output a success bit *b*, while $\mathcal{T}$ also updates the shared state *st* and outputs (ide, idp) and tk_1. Finally $\mathcal{U}_2$ returns a ticket tk_2 for the identifier (ide, idp).

$\mathsf{Validate}\langle\mathcal{U}(\mathsf{sk}_\mathcal{U}, \mathsf{tk}), \mathcal{V}(\mathsf{sk}_\mathcal{V}, st)\rangle \to \mathcal{U}(b), \mathcal{V}(b, \mathsf{tk}, st)$: $\mathcal{U}$ inputs a ticket tk and its key $\mathsf{sk}_\mathcal{U}$, and interacts with $\mathcal{V}$ inputting its key $\mathsf{sk}_\mathcal{V}$ and a state *st* in order to validate the ticket tk. The protocol ends with $\mathcal{U}$ and $\mathcal{V}$ returning a validation bit *b*, a ticket tk and the state *st* from the validator $\mathcal{V}$.

The Shared State. The above model includes a state *st*, shared between the ticket distributor $\mathcal{D}$, the transfer authority $\mathcal{T}$ and the validator $\mathcal{V}$. The shared state allows to synchronize ticket status among the different entities. It prevents, for instance, a double validation or a validation after a refund. This shared state could be seen as a white-list or a blacklist, the latter being used in our protocol. In our model, this state is not kept secret: all adversaries have *read-only* access to the state at any time using an oracle called OLeakState. This state can be implemented as a dedicated server maintaining a database. Note that $\mathcal{D}$, $\mathcal{T}$, and $\mathcal{V}$ share a common state while being represented as distinct entities. This modeling choice aligns with the actual structure of the organization under consideration, reflecting its physical reality. In an alternate scheme, these entities might be perceived as independent entities without shared state. Therefore, our model is more general this way and remain valid for our purpose.

Security Model. Our security experiments are depicted in Fig. 3 and the associated oracles in Fig. 4. In Fig. 2, we provide a concise overview of the key security properties required in our model.

The security model for an ETS emulates realistic behaviors expected from a ticket service. Users possessing a ticket must be able to attend events or claim a refund, this is referred to as protocol *correctness*. In other properties, formulated as security games, $\mathcal{U} = \{\mathcal{U}_k\}_k$ denote the set of initialized users, potentially empty at the beginning. Each user $\mathcal{U}_k$ is associated with a set TK_k representing the tickets it owns. By *st*, we denote the shared state among $\mathcal{D}$, $\mathcal{T}$, and $\mathcal{V}$, initially empty and gradually populated with elements corresponding to revoked tickets. The system should withstand various attack scenarios across multiple executions. To simulate this, we consider a PPT adversary $\mathcal{A}$, allowed to call specific oracles. These oracles, including OCreateUser, OCorruptUser, OPurchase, OTransfer, OValidate, and ORefund, allow the adversary to execute actions. Depending on the expected security game, the adversary have access to specific oracles. Oracles in Fig. 4a allows the user to act as an arbitrary number of users and interact with the oracles simulating the system. At the opposite, oracles Fig. 4b, indexed with an additional bit *b* characterizing the execution for one user among a set of two, allows the adversary playing the roles of the system (and potentially other users), to interact with the one of two, of the two users simulated by the security game. For ticket unforgeability, ticket privacy and double-spending, we claim security even if the shared state is leaked. To address security concerns, we have introduced the OLeakState oracle in Fig. 4a. This read-only access grants insights into the state shared among $\mathcal{D}$, $\mathcal{T}$, and $\mathcal{V}$. This section provides a high-level description of the computational model, further detailed in Fig. 3.

Correctness. Under honest execution of the algorithms, a ticket tk bought by a user $\mathcal{U}$ through Purchase, or Transfer, can be either refund or validated, *i.e.,* Validate and Refund output a success bit *b* which equals 1.

Unforgeability. The *unforgeability* of a ticket prevents an adversary $\mathcal{A}$ from creating a new valid ticket. This property is described in $\mathsf{Exp}^{\mathsf{UF}}_{\mathcal{A}}$. It requires $\mathcal{A}$ to produce a ticket such that (1) the ticket has not been produced by the system *i.e.,* $\mathsf{tk} \notin \mathsf{TK}$, denoted as bit b_0 in the experiment and (2) make it accept either for a refund, a transfer, or a validation, this is denoted as bit b_1 in the experiment. This property must be ensured for any PPT adversary, with read-only access to the shared state and the oracles of Fig. 4a corresponding to possible actions of the users.

Ticket privacy. The *ticket privacy* prevents from an adversary $\mathcal{A}$, external to the system, stealing a ticket from a designated user. The adversary has the capability to generate, manipulate through oracles of Fig. 4a, and corrupt any user within the system. The ticket privacy is focused against entities that are external to the system. In the associated experiment, $\mathsf{Exp}^{\mathsf{PRIV}}_{\mathcal{A}}$, user $\mathcal{U}_1$ is the adversary's target. $\mathcal{U}_1$ purchases a ticket tk and $\mathcal{A}$ wins if (1) the purchase went through, (2) $\mathcal{A}$ outputted a ticket tk^* such that $\mathsf{tk}^* = \mathsf{tk}$ and (3) $\mathcal{A}$ did not corrupt $\mathcal{U}_1$. During this process, the adversary has *read-only* access to the shared state at any point during the experiment. The challenger simulates the user purchasing the ticket targeted for recovery by the adversary, but also $\mathcal{D}$, $\mathcal{T}$, and $\mathcal{V}$, otherwise making the ticket privacy trivially broken as the system requires the ticket for verification purposes.

No-double-spending. Once purchased, a ticket should be usable only once: the ticket can be refund once, transferred once or validated once. In other words, as done in our security model, none of Refund, Validate or Transfer executed by the challenger against a corrupted user would accept the same ticket twice. This notion differs from what has been formalised in *e-cash* [3]. Taking as example the protocol introduced by Baldimtsi *et al.* [3], their model allows execution of a function Spend twice for the same coin and postpone the double spending verification to a second algorithm call Deposit. Applied to ticketing, since the verification occurs after the ticket spent, the consequence for users is the possibility to buy already spent ticket, leading to a ticket rejection (during the second execution of Deposit of the same ticket). Following our notion, an honest client should not acquire a already transferred ticket.

The security notion preventing the adversary to double-transfer a ticket, and more generally to interact twice with the system using the same ticket is denoted as Double-Spending (DS), and is formally introduced in experiment $\mathsf{Exp}^{\mathsf{DS}}_{\mathcal{A}}$: The challenger simulates the system and allows $\mathcal{A}$ to invoke actions through oracles of Fig. 4a, thereby providing a view of the shared state. The adversary subsequently outputs a ticket tk and two actions, Alg_1 and Alg_2, selected from {Purchase, Refund, Transfer}. Remark that the two specified algorithms Alg_1 and Alg_2 are not required to be the same, modelling every possible combination of attack. The adversary succeeds if (1) both algorithm executions are successful, (2) the tickets tk_1 and tk_2 presented to the validator match the committed ticket tk, and (3) the tickets share identical event identifiers ide and serial numbers idp. Both executions are dependent since the shared state *st* is updated after the first execution and used the second time.

Anonymity. To model the properties of non-nominative physical tickets, an ETS should preserve the anonymity of a ticket holder $\mathcal{U}$ against the system. This is modelled by using two properties, one ensuring the *pseudonymity* of $\mathcal{U}$ and, as a complementary, we ensure *unlinkability* of the tickets purchased by a single user. This respectively guarantees that a ticket could not be linked by the system as coming from the same holder nor be linked to a user. In both cases, the adversary controls the system *i.e.,* $\mathcal{D}$, $\mathcal{T}$, and $\mathcal{V}$ and thus can generate and control other users. Experiment $\mathsf{Exp}^{\mathsf{PSE}}_{\mathcal{A}}$ models *pseudonymity*: the challenger generates two users, denoted as $\mathcal{U}_0$ and $\mathcal{U}_1$, represented by their respective public keys $\mathsf{pk}_{\mathcal{U}_0}$ and $\mathsf{pk}_{\mathcal{U}_1}$. $\mathcal{A}$ can invoke oracles $\mathsf{OPurchase}_b$, $\mathsf{ORefund}_b$, $\mathsf{OTransfer}_b$, and $\mathsf{OValidate}_b$ from Fig. 4b, for a given $b \in \{0, 1\}$. For $\mathcal{A}$ to succeed, it must produce a bit b^* that matches the input bit b provided in the oracles. Then determining which one of $\mathcal{U}_0$ and $\mathcal{U}_1$ responded to the oracle calls. Experiment $\mathsf{Exp}^{\mathsf{UNL}}_{\mathcal{A}}$ is designed to model *Unlinkability*. In this experiment, the challenger once again simulates the behaviors of two users, denoted as $\mathcal{U}_0$ and $\mathcal{U}_1$. Initially, the adversary interacts with both users by making calls to the oracles presented in Fig. 4b, denoted as $\mathsf{OPurchase}_i$, $\mathsf{ORefund}_i$, $\mathsf{OTransfer}_i$ and $\mathsf{OValidate}_i$, where $i \in \{0, 1\}$ is chosen by the adversary. Following this preliminary phase, the adversary provides three ticket identifiers $\{\mathsf{ide}_i, \mathsf{idp}_i\}_{i\in 0,1,2}$. Each of $\mathcal{U}_0$ and $\mathcal{U}_1$ performs a purchase and subsequently executes an action from the set {Validate, Refund, Transfer} for one ticket. Then, a bit b is sampled from $\{0, 1\}$, and $\mathcal{U}_b$ replicates the same pattern for the third ticket. The adversary succeeds in the experiment if it can correctly guess which user performed the last actions.

We mitigate anonymity by introducing *Auditable E-Ticket scheme* enabling user's identity recovery under the supervision of a judge.

Definition 3.2 (Auditable ETS*).* An *Auditable E-Ticket scheme* ETS is an E-Ticket scheme increased with an audit algorithm:

$\mathsf{JKeyGen}(1^\lambda) \rightarrow (\mathsf{sk}_{\mathcal{J}}, \mathsf{pk}_{\mathcal{J}})$: Given 1^λ, outputs a key pair.

$\mathsf{Audit}(\mathsf{sk}_{\mathcal{J}}, \mathsf{tk}) \rightarrow \mathsf{pk}_{\mathcal{U}}$: Given a secret key $\mathsf{sk}_{\mathcal{J}}$ and a state *st*, returns the public key associated to user $\mathcal{U}$.

Auditable E-Ticket scheme achieves the previous properties but Audit requests for $\mathcal{U}_0$ and $\mathcal{U}_1$ are not allowed for the adversary during the pseudonymity and unlinkability experiments.

4 CRYPTOGRAPHIC TOOLS

Applause is built upon an *asymmetric encryption*, a *signature scheme* and a *anonymous payment*.

An *asymmetric encryption* scheme $\mathsf{E} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ ensures confidentiality of messages. $\mathsf{KeyGen}(1^\lambda)$ generates the key pair (sk, pk). Given *plaintext* p, one computes the encryption of p using $\mathsf{Enc}_{\mathsf{pk}}(p)$ returning a *ciphertext* c. The ciphertext is decrypted using algorithm $\mathsf{Dec}_{\mathsf{sk}}(c)$ returning a *plaintext* p. We require the encryption scheme to achieve *Correctness* ensuring correct recovery of the plaintext, and *Indistinguishability under Chosen Plaintext Attack* (IND-CPA) against any PPT algorithm $\mathcal{A}$. The probability of breaking IND-CPA for an adversary $\mathcal{A}$ is given by $\mathsf{Adv}^{\text{IND-CPA}}_{\mathcal{A}}$ and must be negligible.

A *signature* scheme $\mathsf{S} = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verif})$ authenticates messages. Again $\mathsf{KeyGen}(1^\lambda)$ generates the key pair (sk, pk), sk being used to sign message m, producing an element $\mathsf{Sign}_{\mathsf{sk}}(m) = \sigma$ called the *signature*. A signature is verified using $\mathsf{Verif}_{\mathsf{pk}}(m, \sigma)$ outputting 1 on success, 0 otherwise. We require the signature to

$\mathsf{Exp}^{\mathsf{UF}}_{\mathcal{A}}(1^\lambda)$

$(\mathsf{sk}_{\mathcal{D}}, \mathsf{pk}_{\mathcal{D}}), (\mathsf{sk}_{\mathcal{T}}, \mathsf{pk}_{\mathcal{T}}), (\mathsf{sk}_{\mathcal{V}}, \mathsf{pk}_{\mathcal{V}})$
$\leftarrow \mathsf{DKeyGen/TKeyGen/VKeyGen}(1^\lambda)$
$\mathcal{U} \leftarrow \emptyset, st \leftarrow \emptyset$
$(\mathsf{tk}, \mathsf{Alg}) \leftarrow \mathcal{A}^{O}(\mathsf{pk}_{\mathcal{D}}, \mathsf{pk}_{\mathcal{T}}, \mathsf{pk}_{\mathcal{V}})$
$b_0 \leftarrow \mathsf{tk} \notin \mathsf{TK}$ // *Ticket not produced by challenger*
if $\mathsf{Alg} \in \{\mathsf{Refund}, \mathsf{Validate}\}$:
  $\mathsf{sk} \leftarrow \mathsf{sk}_{\mathcal{D}}$ **if** $\mathsf{Alg} = \mathsf{Refund}$ **else** $\mathsf{sk} \leftarrow \mathsf{sk}_{\mathcal{V}}$
  $\mathsf{Alg}\langle \mathcal{A}(\cdot,\cdot), C(\mathsf{sk}, st)\rangle \rightarrow \mathcal{A}(\cdot), C(b_1, \mathsf{tk}_0, st)$
  return $b_0 \wedge b_1$
else if $\mathsf{Alg} = \mathsf{Transfer}$:
  $\mathsf{Transfer}\langle \mathcal{A}(\cdot,\cdot), C(\mathsf{sk}_{\mathcal{T}}, st), \mathcal{A}(\cdot)\rangle$
    $\rightarrow \mathcal{A}(\cdot), C(b_1, (\mathsf{ide}_1, \mathsf{idp}_1), st), \mathcal{A}(\cdot)$
  return $b_0 \wedge b_1$
else : **return** 0

$\mathsf{Exp}^{\mathsf{DS}}_{\mathcal{A}}(1^\lambda)$

$(\mathsf{sk}_{\mathcal{D}}, \mathsf{pk}_{\mathcal{D}}), (\mathsf{sk}_{\mathcal{T}}, \mathsf{pk}_{\mathcal{T}}), (\mathsf{sk}_{\mathcal{V}}, \mathsf{pk}_{\mathcal{V}})$
$\leftarrow \mathsf{DKeyGen/TKeyGen/VKeyGen}(1^\lambda)$
$\mathcal{U} \leftarrow \emptyset, st \leftarrow \emptyset$
$(\mathsf{tk}, \mathsf{Alg}_1, \mathsf{Alg}_2) \leftarrow \mathcal{A}^{O}(\mathsf{pk}_{\mathcal{D}}, \mathsf{pk}_{\mathcal{T}}, \mathsf{pk}_{\mathcal{V}})$
for $k \in \{1,2\}$:
  if $\mathsf{Alg}_k \in \{\mathsf{Refund}, \mathsf{Validate}\}$:
    if $\mathsf{Alg}_k = \mathsf{Refund}$: $\mathsf{sk} \leftarrow \mathsf{sk}_{\mathcal{D}}$ **else** $\mathsf{sk} \leftarrow \mathsf{sk}_{\mathcal{V}}$
    $\mathsf{Alg}_k\langle \mathcal{A}(\cdot,\cdot), C(\mathsf{sk}, st)\rangle \rightarrow \mathcal{A}(\cdot), C(b_k, \mathsf{tk}_k, st)$
  if $\mathsf{Alg}_k = \mathsf{Transfer}$:
    $\mathsf{Transfer}\langle \mathcal{A}(\cdot,\cdot), C(\mathsf{sk}_{\mathcal{T}}, st), \mathcal{A}(\cdot)\rangle$
    $\rightarrow \mathcal{A}(\cdot), C(b_1, (\mathsf{ide}_k, \mathsf{idp}_k), \mathsf{tk}_k, st), \mathcal{A}(\cdot)$
  else : **return** 0
return $b_1 \wedge b_2 \wedge ((\mathsf{ide}_1, \mathsf{idp}_1) = (\mathsf{ide}_2, \mathsf{idp}_2))$
  $\wedge (\mathsf{tk} = \mathsf{tk}_1 = \mathsf{tk}_2 \neq \perp)$

$\mathsf{Exp}^{\mathsf{PRIV}}_{\mathcal{A}}(1^\lambda)$

$(\mathsf{sk}_{\mathcal{D}}, \mathsf{pk}_{\mathcal{D}}), (\mathsf{sk}_{\mathcal{T}}, \mathsf{pk}_{\mathcal{T}}), (\mathsf{sk}_{\mathcal{V}}, \mathsf{pk}_{\mathcal{V}})$
$\leftarrow \mathsf{DKeyGen/TKeyGen/VKeyGen}(1^\lambda)$
$(\mathsf{sk}_{\mathcal{U}_1}, \mathsf{pk}_{\mathcal{U}_1}) \leftarrow \mathsf{UKeyGen}(1^\lambda)$
$st \leftarrow \emptyset, \mathcal{U} \leftarrow \mathcal{U}_1 = \{(\mathsf{sk}_{\mathcal{U}_1}, \mathsf{pk}_{\mathcal{U}_1}, \perp, 0)\}$
$(\mathsf{ide}, \mathsf{idp}) \leftarrow \mathcal{A}^{O}(\mathsf{pk}_{\mathcal{U}}, \mathsf{pk}_{\mathcal{D}}, \mathsf{pk}_{\mathcal{T}}, \mathsf{pk}_{\mathcal{V}})$
$\mathsf{Purchase}\langle C(\mathsf{sk}_{\mathcal{U}}, (\mathsf{ide}, \mathsf{idp})), C(\mathsf{sk}_{\mathcal{D}}, st)\rangle$
  $\rightarrow C(\mathsf{tk}), C(b, (\mathsf{ide}, \mathsf{idp}), st)$
$\mathsf{tk}^* \leftarrow \mathcal{A}^{O}(\mathsf{pk}_{\mathcal{U}}, \mathsf{pk}_{\mathcal{D}}, \mathsf{pk}_{\mathcal{T}}, \mathsf{pk}_{\mathcal{V}})$
$\mathcal{U} \xrightarrow{p} \{\mathcal{U}_k\}_k, \mathcal{U}_1 \xrightarrow{p} (\mathsf{sk}_{\mathcal{U}_1}, \mathsf{pk}_{\mathcal{U}_1}, \mathsf{TK}_1, \mathsf{corr}_1),$
return $b \wedge (\mathsf{tk} = \mathsf{tk}^*) \wedge (\mathsf{corr}_1 = 0)$

$\mathsf{Exp}^{\mathsf{UNL}}_{\mathcal{A}}(1^\lambda)$

$(\mathsf{sk}_{\mathcal{U}_0}, \mathsf{pk}_{\mathcal{U}_0}), (\mathsf{sk}_{\mathcal{U}_1}, \mathsf{pk}_{\mathcal{U}_1}) \leftarrow \mathsf{UKeyGen}(1^\lambda)$
$O^{\mathsf{UNL}} \leftarrow O_0 \cup O_1$
$\{\mathsf{ide}_i, \mathsf{idp}_i\}_{i\in\{0,1,2\}} \leftarrow \mathcal{A}^{O^{\mathsf{UNL}}}(\mathsf{pk}_{\mathcal{U}_0}, \mathsf{pk}_{\mathcal{U}_1})$
$b \xleftarrow{\$} \{0,1\}$
for $(i,j) \in \{(0,0),(1,1),(2,b)\}$:
  $\mathsf{Purchase}\langle C(\mathsf{sk}_{\mathcal{U}_j}, (\mathsf{ide}_i, \mathsf{idp}_i)), \mathcal{A}(\cdot,\cdot)\rangle$
    $\rightarrow C(\mathsf{tk}_i), \mathcal{A}(\cdot,\cdot,\cdot)$
$\mathsf{pk}_{\mathcal{A}}, \{\mathsf{Alg}_i\}_{i\in\{0,1,2\}} \leftarrow \mathcal{A}^{O^{\mathsf{UNL}}}()$
for $(i,j) \in \{(0,0),(1,1),(2,b)\}$:
  if $\mathsf{Alg}_i \in \{\mathsf{Refund}, \mathsf{Validate}\}$:
    $\mathsf{Alg}_i\langle C(\mathsf{sk}_{\mathcal{U}_j}, (\mathsf{ide}_i, \mathsf{idp}_i)), \mathcal{A}(\cdot,\cdot)\rangle$
    $\rightarrow C(\mathsf{tk}_i), \mathcal{A}(\cdot,\cdot,\cdot)$
  if $\mathsf{Alg}_i = \mathsf{Transfer}$:
    $\mathsf{Transfer}\langle C(\mathsf{sk}_{\mathcal{U}_j}, \mathsf{pk}_{\mathcal{A}}, \mathsf{tk}_i), \mathcal{A}(\cdot,\cdot), \mathcal{A}(\cdot,\cdot,\cdot)\rangle$
    $\rightarrow C(b_i), \mathcal{A}(\cdot,\cdot,\cdot,\cdot), \mathcal{A}(\cdot)$
$b^* \leftarrow \mathcal{A}(b_0, b_1, b_2)$
return $b = b^*$

$\mathsf{Exp}^{\mathsf{PSE}}_{\mathcal{A}}(1^\lambda)$

$(\mathsf{sk}_{\mathcal{U}_0}, \mathsf{pk}_{\mathcal{U}_0}), (\mathsf{sk}_{\mathcal{U}_1}, \mathsf{pk}_{\mathcal{U}_1}) \leftarrow \mathsf{UKeyGen}(1^\lambda)$
$\mathsf{TK} \leftarrow \emptyset, b \xleftarrow{\$} \{0,1\}$
$b* \leftarrow \mathcal{A}^{O_b}(\mathsf{pk}_{\mathcal{U}_0}, \mathsf{pk}_{\mathcal{U}_1})$
return $b = b^*$

Figure 3: Experiments for Unforgeability $\mathsf{Exp}^{\mathsf{UF}}_{\mathcal{A}}$, Privacy of a ticket $\mathsf{Exp}^{\mathsf{PRIV}}_{\mathcal{A}}$ and Double-Spending $\mathsf{Exp}^{\mathsf{DS}}_{\mathcal{A}}$, Unlinkability $\mathsf{Exp}^{\mathsf{UNL}}_{\mathcal{A}}$ and Pseudonimity $\mathsf{Exp}^{\mathsf{PSE}}_{\mathcal{A}}$. O equals {OPurchase, ORefund, OTransfer, OValidate, OCreateUser, OCorruptUser, OLeakState} **and O_b equals** {$\mathsf{OPurchase}_b$, $\mathsf{ORefund}_b$, $\mathsf{OTransfer}_b$, $\mathsf{OValidate}_b$}

OPurchase($\mathcal{U}, \mathsf{sk}_{\mathcal{D}}; i, \mathsf{sk}_{\mathcal{U}_{\mathcal{A}}}, (\mathsf{ide}, \mathsf{idp})$)

$\mathcal{U} \xrightarrow{p} \{\mathcal{U}_k\}_k, \mathcal{U}_i \xrightarrow{p} (\mathsf{sk}_{\mathcal{U}_i}, \mathsf{pk}_{\mathcal{U}_i}, \mathsf{TK}_i, \mathsf{corr}_i)$
if $\mathsf{corr}_i = 1$: *// Corrupted user*
  $\mathsf{Purchase}\langle \mathcal{A}(\cdot,\cdot), C(\mathsf{sk}_{\mathcal{D}}, st)\rangle$
    $\rightarrow \mathcal{A}(\cdot), C(b, (\mathsf{ide}, \mathsf{idp}), st)$
  $\mathcal{U}_i \leftarrow (\mathsf{sk}_{\mathcal{U}_i}, \mathsf{pk}_{\mathcal{U}_i}, \mathsf{TK}_i \cup \{(\perp, (\mathsf{ide}, \mathsf{idp}), b)\}, \mathsf{corr}_i)$
else : $(\mathsf{ide}, \mathsf{idp}) \xleftarrow{\$} \mathsf{ID}_{\mathcal{E}} \times \mathsf{ID}_{\mathcal{P}}$ *// Honest user*
  $\mathsf{Purchase}\langle C(\mathsf{sk}_{\mathcal{U}_i}, (\mathsf{ide}, \mathsf{idp})), C(\mathsf{sk}_{\mathcal{D}}, st)\rangle$
    $\rightarrow C(\mathsf{tk}), C(b, (\mathsf{ide}, \mathsf{idp}), st)$
  $\mathcal{U}_i \leftarrow (\mathsf{sk}_{\mathcal{U}_i}, \mathsf{pk}_{\mathcal{U}_i}, \mathsf{TK}_i \cup \{(\mathsf{tk}, (\mathsf{ide}, \mathsf{idp}), b)\}, \mathsf{corr}_i)$

OValidate($\mathcal{U}, \mathsf{sk}_{\mathcal{V}}; i, \mathsf{sk}_{\mathcal{A}_i}, \mathsf{tk}_i, (\mathsf{ide}, \mathsf{idp})$)

$\mathcal{U} \xrightarrow{p} \{\mathcal{U}_k\}_k, \mathcal{U}_i \xrightarrow{p} (\mathsf{sk}_{\mathcal{U}_i}, \mathsf{pk}_{\mathcal{U}_i}, \mathsf{TK}_i, \mathsf{corr}_i)$
if $\mathsf{corr}_i = 1$: *// Corrupted user*
  $\mathsf{Validate}\langle \mathcal{A}(\cdot,\cdot), C(\mathsf{sk}_{\mathcal{V}}, st)\rangle \rightarrow \mathcal{A}(\cdot), C(b, \mathsf{tk}, st)$
  $\mathcal{U}_i \leftarrow (\mathsf{sk}_{\mathcal{U}_i}, \mathsf{pk}_{\mathcal{U}_i}, \mathsf{TK}_i \cup \{(\mathsf{tk}, (\mathsf{ide}, \mathsf{idp}), 1-b)\}, \mathsf{corr}_i)$
else : *// Honest user*
  $\mathsf{TK}_i \rightarrow \{\mathsf{tk}', (\mathsf{ide}, \mathsf{idp}), b\}$
  $\mathsf{Validate}\langle C(\mathsf{sk}_{\mathcal{U}_i}, \mathsf{tk}'), C(\mathsf{sk}_{\mathcal{V}}, st)\rangle \rightarrow C(b), C(b, \mathsf{tk}, st)$
  $\mathcal{U}_i \leftarrow (\mathsf{sk}_{\mathcal{U}_i}, \mathsf{pk}_{\mathcal{U}_i}, \mathsf{TK}_i \cup \{(\mathsf{tk}, (\mathsf{ide}, \mathsf{idp}), 1-b)\}, \mathsf{corr}_i)$

ORefund($\mathcal{U}, \mathsf{sk}_{\mathcal{D}}; i, \mathsf{sk}_{\mathcal{U}_{\mathcal{A}}}, \mathsf{tk}, (\mathsf{ide}, \mathsf{idp})$)

$\mathcal{U} \xrightarrow{p} \{\mathcal{U}_k\}_k, \mathcal{U}_i \xrightarrow{p} (\mathsf{sk}_{\mathcal{U}_i}, \mathsf{pk}_{\mathcal{U}_i}, \mathsf{TK}_i, \mathsf{corr}_i)$
if $\mathsf{corr}_i = 1$: *// Corrupted user*
  $\mathsf{Refund}\langle \mathcal{A}(\cdot,\cdot), C(\mathsf{sk}_{\mathcal{D}}, st)\rangle \rightarrow \mathcal{A}(\cdot), C(b, \mathsf{tk}, st)$
  $\mathcal{U}_i \leftarrow (\mathsf{sk}_{\mathcal{U}_i}, \mathsf{pk}_{\mathcal{U}_i}, \mathsf{TK}_i \cup \{\mathsf{tk}, (\mathsf{ide}, \mathsf{idp}), 1-b\}, \mathsf{corr}_i)$
else : *// Honest user*
  $\mathsf{TK}_i \xrightarrow{p} \{\mathsf{tk}', (\mathsf{ide}, \mathsf{idp}), b\}$
  $\mathsf{Refund}\langle C(\mathsf{sk}_{\mathcal{U}}, \mathsf{tk}'), C(\mathsf{sk}_{\mathcal{D}}, st)\rangle \rightarrow C(b), C(b, \mathsf{tk}', st)$
  $\mathcal{U}_i \leftarrow (\mathsf{sk}_{\mathcal{U}_i}, \mathsf{pk}_{\mathcal{U}_i}, \mathsf{TK}_i \cup \{\mathsf{tk}, (\mathsf{ide}, \mathsf{idp}), 1-b\}, \mathsf{corr}_i)$

OTransfer($\mathcal{U}, \mathsf{sk}_{\mathcal{T}}; i, j, \mathsf{sk}_{\mathcal{A}_i}, \mathsf{sk}_{\mathcal{A}_j}, \mathsf{tk}_i, (\mathsf{ide}, \mathsf{idp})$)

$\mathcal{U} \xrightarrow{p} \{\mathcal{U}_k\}_k, \mathcal{U}_i \xrightarrow{p} (\mathsf{sk}_{\mathcal{U}_i}, \mathsf{pk}_{\mathcal{U}_i}, \mathsf{TK}_i, \mathsf{corr}_i),$
$\mathcal{U}_j \xrightarrow{p} (\mathsf{sk}_{\mathcal{U}_j}, \mathsf{pk}_{\mathcal{U}_j}, \mathsf{TK}_i, \mathsf{corr}_i)$
if $\mathsf{corr}_i = 1 \wedge \mathsf{corr}_j = 1$: *// Corrupted users*
  $\mathsf{Transfer}\langle \mathcal{A}(\cdot,\cdot), C(\mathsf{sk}_{\mathcal{T}}, st), \mathcal{A}(\cdot)\rangle$
    $\rightarrow \mathcal{A}(\cdot), C(b, (\mathsf{ide}, \mathsf{idp}), st), \mathcal{A}(\cdot)$
  $\mathcal{U}_i \leftarrow (\mathsf{sk}_{\mathcal{U}_i}, \mathsf{pk}_{\mathcal{U}_i}, \mathsf{TK}_i \cup \{(\perp, (\mathsf{ide}, \mathsf{idp}), b)\}, \mathsf{corr}_i)$
if $\mathsf{corr}_i = 1$: *// Corrupted sender*
  $\mathsf{Transfer}\langle \mathcal{A}(\cdot,\cdot), C(\mathsf{sk}_{\mathcal{T}}, st), C(\mathsf{sk}_{\mathcal{U}_j}, (\mathsf{ide}, \mathsf{idp}))\rangle$
    $\rightarrow \mathcal{A}(\cdot), C(b, (\mathsf{ide}, \mathsf{idp}), st), C(\mathsf{tk})$
  $\mathcal{U}_i \leftarrow (\mathsf{sk}_{\mathcal{U}_i}, \mathsf{pk}_{\mathcal{U}_i}, \mathsf{TK}_i \cup \{(\mathsf{tk}', (\mathsf{ide}, \mathsf{idp}), b)\}, \mathsf{corr}_i)$
if $\mathsf{corr}_j = 1$: *// Corrupted receiver*
  $\mathsf{TK}_i \rightarrow \{\mathsf{tk}', (\mathsf{ide}, \mathsf{idp}), b'\}$
  $\mathsf{Transfer}\langle C(\mathsf{sk}_{\mathcal{U}_i}, \mathsf{tk}'), C(\mathsf{sk}_{\mathcal{T}}, st), \mathcal{A}(\cdot)\rangle$
    $\rightarrow C(b), C(b, (\mathsf{ide}, \mathsf{idp}), st), \mathcal{A}(\cdot)$
  $\mathcal{U}_i \leftarrow (\mathsf{sk}_{\mathcal{U}_i}, \mathsf{pk}_{\mathcal{U}_i}, \mathsf{TK}_i \cup \{(\perp, (\mathsf{ide}, \mathsf{idp}), b)\}, \mathsf{corr}_i)$

OCreateUser($\mathcal{U}$; pk)

if $\mathsf{pk} \neq \perp$: $i \leftarrow |\mathcal{U}|, \mathcal{U} \leftarrow \mathcal{U} \cup \{\mathcal{U}_i = (\perp, \mathsf{pk}, \perp, 1)\}$
  return $\perp$
else : $i \leftarrow |\mathcal{U}|, (\mathsf{sk}_{\mathcal{U}_i}, \mathsf{pk}_{\mathcal{U}_i}) \leftarrow \mathsf{UKeyGen}(\mathsf{pp})$
  $\mathcal{U} \leftarrow \mathcal{U} \cup \{\mathcal{U}_i = (\mathsf{sk}_{\mathcal{U}_i}, \mathsf{pk}_{\mathcal{U}_i}, \perp, 0)\}$
  return $\mathsf{pk}_{\mathcal{U}}$

OCorruptUser($\mathcal{U}; i$)

$\mathcal{U} \xrightarrow{p} \{\mathcal{U}_k\}_k, \mathcal{U}_i \xrightarrow{p} (\mathsf{sk}_{\mathcal{U}_i}, \mathsf{pk}_{\mathcal{U}_i}, \mathsf{TK}_i, \mathsf{corr}_i)$
$\mathcal{U}_i \leftarrow \{(\mathsf{sk}_{\mathcal{U}_i}, \mathsf{pk}_{\mathcal{U}_i}, \mathsf{TK}_i, 1)\}$
return $(\mathsf{sk}_{\mathcal{U}_i}, \mathsf{pk}_{\mathcal{U}_i}, \mathsf{TK}_i)$

OLeakState(st)

return st

(a) Oracles for the experiments $\mathsf{Exp}^{\mathsf{UF}}_{\mathcal{A}}$, $\mathsf{Exp}^{\mathsf{DS}}_{\mathcal{A}}$, $\mathsf{Exp}^{\mathsf{UNL}}_{\mathcal{A}}$ and $\mathsf{Exp}^{\mathsf{PRIV}}_{\mathcal{A}}$.

$\mathsf{OPurchase}_b(\mathcal{U}, \mathsf{sk}_{\mathcal{U}_b}; (\mathsf{ide}, \mathsf{idp}))$

$\mathsf{Purchase}\langle C(\mathsf{sk}_{\mathcal{U}_b}, (\mathsf{ide}, \mathsf{idp})), \mathcal{A}(\cdot,\cdot)\rangle \rightarrow C(\mathsf{tk}_{|\mathsf{TK}|}), \mathcal{A}(\cdot)$
$\mathsf{TK} \leftarrow \mathsf{TK} \cup \{\mathsf{tk}_{|\mathsf{TK}|}\}$

$\mathsf{ORefund}_b(\mathcal{U}, \mathsf{sk}_{\mathcal{U}_b}; i)$

if $i \geq |\mathcal{U}|$: **return** $\perp$
$\mathsf{TK} \xrightarrow{p} \{\mathsf{tk}_j\}_j$
$\mathsf{Refund}\langle C(\mathsf{sk}_{\mathcal{U}_b}, \mathsf{tk}_i), \mathcal{A}(\cdot,\cdot)\rangle \rightarrow C(b), \mathcal{A}(\cdot,\cdot,\cdot)$

$\mathsf{OValidate}_b(\mathcal{U}, \mathsf{sk}_{\mathcal{U}_b}; i)$

if $i \geq |\mathcal{U}|$: **return** $\perp$
$\mathsf{TK} \xrightarrow{p} \{\mathsf{tk}_j\}_j$
$\mathsf{Validate}\langle C(\mathsf{sk}_{\mathcal{U}_b}, \mathsf{tk}_i), \mathcal{A}(\cdot,\cdot)\rangle \rightarrow C(b), \mathcal{A}(\cdot,\cdot,\cdot)$

$\mathsf{OTransfer}_b(\mathcal{U}, \mathsf{sk}_{\mathcal{U}_b}; i, \mathsf{role}, \mathsf{pk}'_{\mathcal{U}_j}, (\mathsf{ide}, \mathsf{idp}))$

if $i \geq |\mathcal{U}|$: **return** $\perp$
if role = sell: $\mathsf{TK} \xrightarrow{p} \{\mathsf{tk}_j\}_j$
  $\mathsf{Transfer}\langle C(\mathsf{sk}_{\mathcal{U}_i}, \mathsf{pk}'_{\mathcal{U}_j}, \mathsf{tk}'), \mathcal{A}(\cdot,\cdot), \mathcal{A}(\cdot)\rangle$
    $\rightarrow C(b), \mathcal{A}(\cdot,\cdot,\cdot), \mathcal{A}(\cdot)$
if role = buy:
  $\mathsf{Transfer}\langle \mathcal{A}(\cdot,\cdot), \mathcal{A}(\cdot,\cdot), C(\mathsf{sk}_{\mathcal{A}_j}, (\mathsf{ide}, \mathsf{idp}))\rangle$
    $\rightarrow \mathcal{A}(\cdot), \mathcal{A}(\cdot,\cdot,\cdot), C(\mathsf{tk}_{|\mathsf{TK}|})$
  $\mathsf{TK} \leftarrow \mathsf{TK} \cup \{\mathsf{tk}_{|\mathsf{TK}|}\}$

(b) Oracles for the experiment $\mathsf{Exp}^{\mathsf{PSE}}_{\mathcal{A}}$.

Figure 4: In each oracle, arguments provided by the challenger C appear before the ';', while arguments specified after are provided by $\mathcal{A}$.

achieve *Correctness* of the algorithms, and *Existential Unforgeability under Chosen Message Attacks* (EUF-CMA) against any PPT algorithm $\mathcal{A}$. The probability of breaking EUF-CMA for an adversary $\mathcal{A}$ is given by $\mathsf{Adv}^{\text{EUF-CMA}}_{\mathcal{A}}$ and must be negligible. We stress that a signature does *not* provide access to the associated message. For both encryption and signature, we require the algorithm $\mathsf{RandKey}(\mathsf{sk}, \mathsf{pk}) \to (\mathsf{sk}', \mathsf{pk}')$ allowing to randomise a key pair. Using discrete log based keys $\mathsf{pk} \leftarrow g^{\mathsf{sk}}$ for some prime order group generator g, the randomisable key mechanism can be seen as $\mathsf{RandKey}(\mathsf{sk}, \mathsf{pk}) \to (\mathsf{sk} \cdot r, \mathsf{pk}^r)$ for a r randomly sampled at uniform ($r \xleftarrow{\$} G$ denotes a uniform sample of r from the set G).In many cases this is the same as generating a new key.

Ticket purchasing, transferring, and refunding necessitate payments. However, conventional online payment methods (such as card payments based on the EMV protocol [17]) disclose the user's identity, posing a challenge to ensure user anonymity in a ETS setting. Therefore, the anonymity within a ETS protocol is contingent on the anonymity provided by the underlying payment protocol. Rather than omitting the payment process, as observed in [25], we've opted to incorporate the payment protocol. Specifically, to maintain user anonymity, the incorporated payment, termed *Anonymous Payment* and utilized as a foundational component, must enable participants to make payments without disclosing their identities. This building block is voluntary generic to let any anonymous payment method, to be plugged in our protocol. Our *generic description* for *Payment Schemes* PS comprises the PPT algorithms:

$\underline{\mathsf{KeyGen}(1^\lambda)}$: Generate the key pair $(\mathsf{sk}, \mathsf{pk})$.

$\underline{\mathsf{Pay}\langle \mathcal{U}_1(\mathsf{sk}_{\mathcal{U}_1}), \mathcal{U}_2(\mathsf{sk}_{\mathcal{U}_2})\rangle \to \mathcal{U}_1(b), \mathcal{U}_2(b)}$: User $\mathcal{U}_1$ performs a payment to $\mathcal{U}_2$. At the end, a success bit b is returned.

For the sake of our demonstration and to facilitate our security reduction without mandating a specific payment method, we employ two generic definitions for *Unlinkability*, which ensures that two transactions from the same payee aren't linked, and *Pseudonymity*, which ensure that payee's identity remains undisclosed.

Pseudonymity. For any PPT adversary $\mathcal{A}$, it should not be possible to predict the user $\mathcal{U}_b$, for $b \in \{0,1\}$, proceeding to a payment, with a probability significantly different from 1/2. $\mathcal{A}$ has access to a $\mathsf{Pay}\langle C(\mathsf{sk}_b, \cdot), \mathcal{A}(\cdot)\rangle$ oracle. Anonymous payment ensures *pseudonymity* if, for any 1^λ,

$$\left|\Pr\left[\begin{array}{l}(\mathsf{sk}_i, \mathsf{pk}_i)_{i\in\{0,1\}} \xleftarrow{\$} \mathsf{KeyGen}(1^\lambda) \\ b \xleftarrow{\$} \{0,1\} \\ b^* \leftarrow \mathcal{A}^{\mathsf{Pay}\langle C(\mathsf{sk}_{\mathcal{U}_b}), \mathcal{A}\rangle}(\mathsf{pk}_0, \mathsf{pk}_1)\end{array} : b = b^*\right] - \tfrac{1}{2}\right| = \mathsf{Adv}^{\mathsf{PayPse}}_{\mathcal{A}} \leq \mathsf{negl}(\lambda).$$

Unlinkability. For any PPT adversary $\mathcal{A}$, it should not be possible to link payment to a user with a probability significantly different from 1/2. $\mathcal{A}$ has access to $\mathsf{Pay}\langle C(\mathsf{sk}_{\mathcal{U}_i}, \cdot), \mathcal{A}(\cdot)\rangle$ oracle for $i \in \{0,1\}$. An anonymous payment scheme ensures *Unlinkability* if, for any 1^λ,

$$\left|\Pr\left[\begin{array}{l}(\mathsf{sk}_i, \mathsf{pk}_i)_{i\in\{0,1\}} \xleftarrow{\$} \mathsf{KeyGen}(1^\lambda) \\ \mathcal{A}^{\mathsf{Pay}\langle C(\mathsf{sk}_{\mathcal{U}_i}), \mathcal{A}\rangle_{i\in\{0,1\}}}(\mathsf{pk}_0, \mathsf{pk}_1) \\ b \xleftarrow{\$} \{0,1\} \\ \mathsf{Pay}\langle C(\mathsf{sk}_{\mathcal{U}_b}), \mathcal{A}(\cdot)\rangle \\ b^* \leftarrow \mathcal{A}()\end{array} : b = b^*\right] - \tfrac{1}{2}\right| = \mathsf{Adv}^{\mathsf{PayUnl}}_{\mathcal{A}} \leq \mathsf{negl}(\lambda).$$

In this paper, we do not cover how to plug anonymous payment formally, since deciding which payment protocol to use is highly dependent of the context where our ticketing system is deployed. One straightforward yet inefficient payment method that maintains user anonymity is through a one-time payment card like PaySafeCard [40], purchasable locally with physical currency. Blockchain based methods such as Monero [44] provides payment unlinkability and pseudonymity using one-time payment identity in a sense that may differ slightly from our definitions. Another example uses cash or its digital counterpart like CashApp [8], where payments are executed using randomly generated identifiers. Anonymous transferable e-cash [3] also matches our requirements, providing a solution to exchange anonymously coins between users. EMV specification for *Tokenization* [18] provides pseudonymity, but falls short in providing unlinkability. It is uncertain if the current card system would allow anonymity. Our protocol, which are fully described later, benefits from reductions from their anonymity to the anonymity of the payment scheme. Therefore, they are as anonymous as the payment method used when they are deployed.

5 DESCRIPTION OF APPLAUSE

In the protocol, the participants interact using personal encryption keys $(\mathsf{sk}^{\mathsf{E}}, \mathsf{pk}^{\mathsf{E}})$ and signature keys $(\mathsf{sk}^{\mathsf{S}}, \mathsf{pk}^{\mathsf{S}})$ and proceed to payment using dedicated keys. A shared state *st* is updated by the ticket distributor $\mathcal{D}$, the transfer authority $\mathcal{T}$ and the validator $\mathcal{V}$, acting as a blacklist. To purchase a ticket through Purchase, the user first selects an event and a free seat (ide, idp), and it also draws a nonce r_c. It obtains a signature σ_c on the hash of the triple $(\mathsf{ide}, \mathsf{idp}, r_c)$. The nonce r_c and the signature σ_c represent the ticket of the client. The ticket validation process involves the showing of these two values to the validator and the agreement on a challenge value through a physical channel to authenticate the ticket owner. The first showing is similar to the Refund process before the client is given back its funds or during the Transfer, as the transfer authority $\mathcal{T}$ refunds the owner of the ticket, and performs the ticket purchase protocol with the new owner of the ticket.

Our E-Ticket Scheme. In Fig. 5b, we present the diagram for the Refund protocol. In Fig. 5c, we present the diagram for the Validate protocol. Finally, in Fig. 6, we present the diagram for the Transfer protocol. The diagrams present the non-auditable version of Applause. . We first introduce a KeyGen algorithm:

$\underline{\mathsf{KeyGen}(1^\lambda)}$: Generates signature keys $(\mathsf{sk}^{\mathsf{S}}, \mathsf{pk}^{\mathsf{S}}) \leftarrow \mathsf{S.KeyGen}(1^\lambda)$, encryption keys $(\mathsf{sk}^{\mathsf{E}}, \mathsf{pk}^{\mathsf{E}}) \leftarrow \mathsf{E.KeyGen}(1^\lambda)$, payment keys $(\mathsf{sk}^{\mathsf{P}}, \mathsf{pk}^{\mathsf{P}}) \leftarrow \mathsf{P.KeyGen}(1^\lambda)$, and return $(\mathsf{sk} \leftarrow (\mathsf{sk}^{\mathsf{S}}, \mathsf{sk}^{\mathsf{E}}, \mathsf{sk}^{\mathsf{P}}), \mathsf{pk} \leftarrow (\mathsf{pk}^{\mathsf{S}}, \mathsf{pk}^{\mathsf{E}}, \mathsf{pk}^{\mathsf{P}}))$.

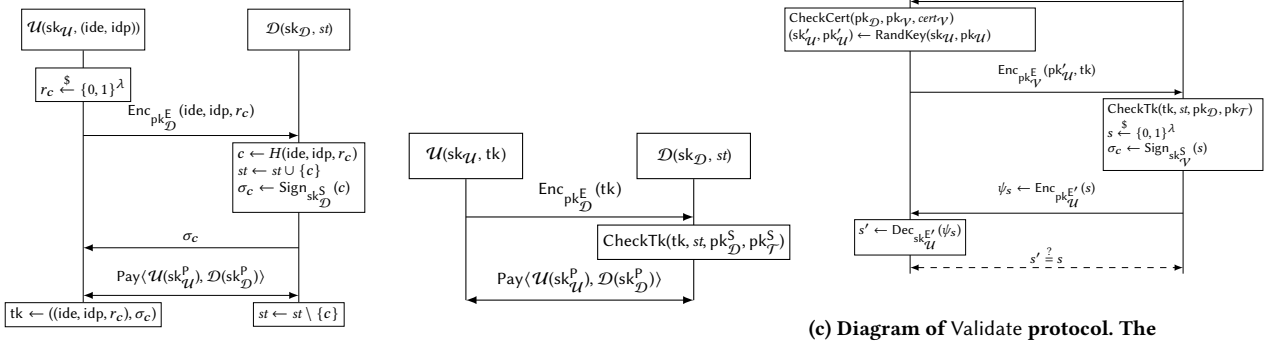
We are now ready to introduce the different key generation algorithms for the parties involved in our protocol:

$\underline{\mathsf{DKeyGen}(1^\lambda)}$: Outputs $(\mathsf{sk}_{\mathcal{D}}, \mathsf{pk}_{\mathcal{D}}) \leftarrow \mathsf{KeyGen}(1^\lambda)$.

$\underline{\mathsf{TKeyGen}(1^\lambda)}$: Outputs $(\mathsf{sk}_{\mathcal{T}}, \mathsf{pk}_{\mathcal{T}}) \leftarrow \mathsf{KeyGen}(1^\lambda)$.

$\underline{\mathsf{UKeyGen}(1^\lambda)}$: Outputs $(\mathsf{sk}_{\mathcal{U}}, \mathsf{pk}_{\mathcal{U}}) \leftarrow \mathsf{KeyGen}(1^\lambda)$.

$\underline{\mathsf{VKeyGen}(1^\lambda)}$: Generates the key pair $(\mathsf{sk}^{\mathsf{S}}_{\mathcal{V}}, \mathsf{pk}^{\mathsf{S}}_{\mathcal{V}}) \leftarrow \mathsf{S.KeyGen}(1^\lambda)$, and an encryption key pair $(\mathsf{sk}^{\mathsf{E}}_{\mathcal{V}}, \mathsf{pk}^{\mathsf{E}}_{\mathcal{V}}) \leftarrow \mathsf{E.KeyGen}(1^\lambda)$. Queries the signature $\mathsf{Sign}_{\mathsf{sk}^{\mathsf{S}}_{\mathcal{D}}}(\mathsf{pk}^{\mathsf{S}}_{\mathcal{V}}, \mathsf{pk}^{\mathsf{E}}_{\mathcal{V}}) \to \mathit{cert}_{\mathcal{V}}$ to $\mathcal{D}$



(a) Diagram of Purchase **protocol.** **(b) Diagram of** Refund **protocol.** **(c) Diagram of** Validate **protocol. The dashed line is the physical channel.**

Figure 5: Diagram of sequences for Purchase, Refund **and** Validate **protocols.**

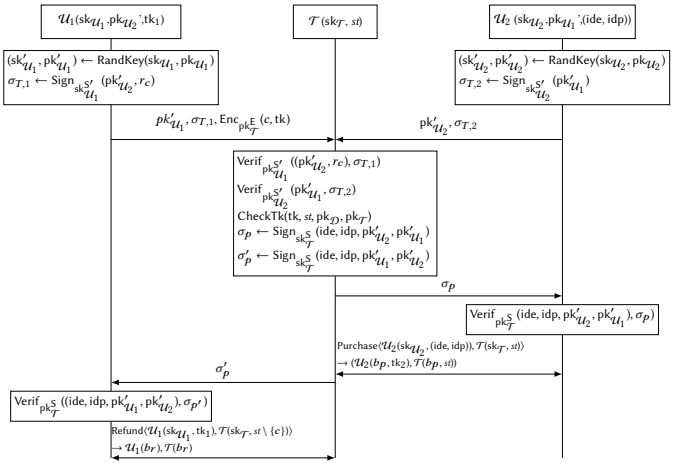


Figure 6: Diagram of sequences for the Transfer **protocol.**

on $\mathsf{pk}^\mathsf{E}_\mathcal{V}$. Set and return $\mathsf{sk}_\mathcal{V} \leftarrow (\mathsf{sk}^\mathsf{S}_\mathcal{V}, \mathsf{sk}^\mathsf{E}_\mathcal{V})$, $\mathsf{pk}_\mathcal{V} \leftarrow (\mathsf{pk}^\mathsf{S}_\mathcal{V}, \mathsf{pk}^\mathsf{E}_\mathcal{V}, cert_\mathcal{V})$.

Before purchasing a ticket, a user chooses the event of the ticket denoted by $\mathsf{ide} \leftarrow \mathsf{ID}_\mathcal{E}$, and a serial or seat number $\mathsf{idp} \leftarrow \mathsf{ID}_\mathcal{P}$.

$\mathsf{Purchase}\langle \mathcal{U}(\mathsf{sk}_\mathcal{U}, (\mathsf{ide}, \mathsf{idp})), \mathcal{D}(\mathsf{sk}_\mathcal{D}, st) \rangle$: $\mathcal{U}$ samples $r_c \xleftarrow{\$} \{0,1\}^\lambda$. Then it sends $\mathsf{Enc}_{\mathsf{pk}^\mathsf{E}_\mathcal{D}}(\mathsf{ide}, \mathsf{idp}, r_c)$ to the ticket distributor $\mathcal{D}$, which decrypts the message. After checking that the pair (idp, ide) has not been purchased before, $\mathcal{D}$ computes $c \leftarrow H(\mathsf{ide}, \mathsf{idp}, r_c)$, then sets $st \leftarrow st \cup \{c\}$ and signs $\sigma_c \leftarrow \mathsf{Sign}_{\mathsf{sk}^\mathsf{S}_\mathcal{D}}(c)$, before sending σ_c to $\mathcal{U}$. Once $\mathcal{U}$ verified the signature, $\mathcal{U}$ pays with $\mathsf{Pay}\langle \mathcal{U}(\mathsf{sk}^\mathsf{P}_\mathcal{U}), \mathcal{D}(\mathsf{sk}^\mathsf{P}_\mathcal{D}) \rangle \rightarrow \mathcal{U}(b), \mathcal{D}(b)$. If the payment works (*i.e.,* b equals 1) then $\mathcal{U}$ returns b and $\mathsf{tk} = ((\mathsf{ide}, \mathsf{idp}, r_c), \sigma_c)$. Finally, $\mathcal{D}$ updates the shared state st: if $b = 1$, $st \leftarrow st \setminus \{c\}$ and returns b and st.

For clarity, we describe the CheckTk subroutine:

$\mathsf{CheckTk}(\mathsf{tk}, st)$: Parses $\mathsf{tk} \xrightarrow{p} ((\mathsf{ide}, \mathsf{idp}, r_c), \sigma_c)$ ($\xrightarrow{p}$ denotes the parsing), gets $c \leftarrow H(\mathsf{ide}, \mathsf{idp}, r_c)$ and verifies $\mathsf{Verif}_{\mathsf{pk}^\mathsf{S}_\mathcal{D}}(c, \sigma_c)$ or $\mathsf{Verif}_{\mathsf{pk}^\mathsf{S}_\mathcal{T}}(c, \sigma_c)$. Checks that c is not a blacklisted ticket (*i.e.,* $c \notin st$). If all pass, it sets $st \leftarrow st \cup \{c\}$ and returns st.

$\mathsf{Refund}\langle \mathcal{U}(\mathsf{sk}_\mathcal{U}, \mathsf{tk}), \mathcal{D}(\mathsf{sk}_\mathcal{D}, st) \rangle$: $\mathcal{U}$ starts by sending $\mathsf{Enc}_{\mathsf{pk}^\mathsf{E}_\mathcal{D}}(\mathsf{tk})$ to $\mathcal{D}$. After the decryption, $\mathcal{D}$ checks the validity of the received ticket by executing $\mathsf{CheckTk}(\mathsf{tk}, st) \rightarrow st$. Both participants start a refund (*i.e.,* a payment with a negative amount) using $\mathsf{Pay}\langle \mathcal{U}(\mathsf{sk}^\mathsf{P}_\mathcal{U}), \mathcal{D}(\mathsf{sk}^\mathsf{P}_\mathcal{D}) \rangle \rightarrow \mathcal{U}(b), \mathcal{D}(b)$. If $b = 0$, $\mathcal{D}$ reverts its state to $st \leftarrow st \setminus \{c\}$. Both parties return b, and $\mathcal{D}$ additionally returns st along tk.

The Transfer protocol assumes that user $\mathcal{U}_1$ holds the randomised public key $\mathsf{pk}'_{\mathcal{U}_2}$ of user $\mathcal{U}_2$ and conversely.

$\mathsf{Transfer}\langle \mathcal{U}_1(\mathsf{sk}'_{\mathcal{U}_1}, \mathsf{pk}'_{\mathcal{U}_2}, \mathsf{tk}_1), \mathcal{T}(\mathsf{sk}_\mathcal{T}, st), \mathcal{U}_2(\mathsf{sk}'_{\mathcal{U}_2}, \mathsf{pk}'_{\mathcal{U}_1}, (\mathsf{ide}, \mathsf{idp})) \rangle$: $\mathcal{U}_1$ computes a signature $\mathsf{Sign}_{\mathsf{sk}^{\mathsf{S}\prime}_{\mathcal{U}_1}}(\mathsf{pk}'_{\mathcal{U}_2}, r_c) \rightarrow \sigma_{T,1}$ and sends $(\mathsf{pk}'_{\mathcal{U}_1}, \sigma_{T,1}, \mathsf{Enc}_{\mathsf{pk}^\mathsf{E}_\mathcal{T}}(c, \mathsf{tk}))$ to $\mathcal{T}$. In the meantime, $\mathcal{U}_2$ signs $\mathsf{Sign}_{\mathsf{sk}^{\mathsf{S}\prime}_{\mathcal{U}_2}}(\mathsf{pk}'_{\mathcal{U}_1}) \rightarrow \sigma_{T,2}$ and sends $(\mathsf{pk}'_{\mathcal{U}_2}, \sigma_{T,2})$ to $\mathcal{T}$. Once $\sigma_{T,1}$ and $\sigma_{T,2}$ received, $\mathcal{T}$ checks $\mathsf{Verif}_{\mathsf{pk}^{\mathsf{S}\prime}_{\mathcal{U}_1}}((\mathsf{pk}'_{\mathcal{U}_2}, r_c), \sigma_{T,1})$ and $\mathsf{Verif}_{\mathsf{pk}^{\mathsf{S}\prime}_{\mathcal{U}_2}}(\mathsf{pk}'_{\mathcal{U}_1}, \sigma_{T,2})$, and halts if it fails. Then, $\mathcal{T}$ checks the validity of the received ticket by executing the algorithm $\mathsf{CheckTk}(\mathsf{tk}, st) \rightarrow st$. If st is updated, $\mathcal{T}$ signs and sends the signature $\sigma_p \leftarrow \mathsf{Sign}_{\mathsf{sk}^\mathsf{S}_\mathcal{T}}(\mathsf{ide}, \mathsf{idp}, \mathsf{pk}'_{\mathcal{U}_2}, \mathsf{pk}'_{\mathcal{U}_1})$ to $\mathcal{U}_2$. Once σ_p is verified, $\mathcal{U}_2$ initiates the purchase of place (ide, idp) with $\mathsf{Purchase}\langle \mathcal{U}_2(\mathsf{sk}_{\mathcal{U}_2}, (\mathsf{ide}, \mathsf{idp})), \mathcal{T}(\mathsf{sk}_\mathcal{T}, st) \rangle \rightarrow (\mathcal{U}_2(b_p, \mathsf{tk}_2), \mathcal{T}(b_p, st))$ where $\mathcal{T}$ does not check that (ide,idp) where already attributed. If b_p equals 0, then $\mathcal{T}$ sets $st \leftarrow st \setminus \{c\}$ (previously added in st during the CheckTk execution) and halts. Otherwise, on success, $\sigma'_p \leftarrow \mathsf{Sign}_{\mathsf{sk}^\mathsf{S}_\mathcal{T}}(\mathsf{ide}, \mathsf{idp}, \mathsf{pk}'_{\mathcal{U}_1}, \mathsf{pk}'_{\mathcal{U}_2})$ is sent and verified by $\mathcal{U}_1$. Then $\mathcal{U}_1$ is refund by executing $\mathsf{Refund}\langle \mathcal{U}_1(\mathsf{sk}_{\mathcal{U}_1}, \mathsf{tk}_1), \mathcal{T}(\mathsf{sk}_\mathcal{T}, st \setminus \{c\}) \rangle \rightarrow \mathcal{U}_1(b_r), \mathcal{T}(b_r)$ with $\mathcal{T}$. Last $\mathcal{U}_1$ returns b_r, $\mathcal{T}$ updates st and returns b_r, while $\mathcal{U}_2$ returns tk_2. Note that between two honest users $\mathcal{U}_1$ and $\mathcal{U}_2$, the secret knowledge tk_1 could be simply transferred without the help of $\mathcal{T}$ [1].

The validation protocol requires for its last interaction, a physical channel, to prevent relay attacks, where an active external adversary blocks the ticket in the network, and play it in the validation process.

Validate$\langle \mathcal{U}(\mathsf{sk}_{\mathcal{U}}, \mathsf{tk}), \mathcal{V}(\mathsf{sk}_{\mathcal{V}}, st)\rangle$: $\mathcal{V}$ starts by providing $\mathsf{pk}_{\mathcal{V}}$ to $\mathcal{U}$, which parses the key and runs $\mathsf{Verif}_{\mathsf{pk}^{\mathsf{S}}_{\mathcal{D}}}((\mathsf{pk}^{\mathsf{S}}_{\mathcal{V}}, \mathsf{pk}^{\mathsf{E}}_{\mathcal{V}}), cert_{\mathcal{V}}) \rightarrow b$. We abort when b equals 0. Otherwise, $\mathcal{U}$ executes $\mathsf{RandKey}(\mathsf{sk}_{\mathcal{U}}, \mathsf{pk}_{\mathcal{U}}) \rightarrow (\mathsf{sk}'_{\mathcal{U}}, \mathsf{pk}'_{\mathcal{U}})$ and sends the ciphertext $\mathsf{Enc}_{\mathsf{pk}^{\mathsf{E}}_{\mathcal{V}}}(\mathsf{pk}'_{\mathcal{U}}, \mathsf{tk})$ to $\mathcal{V}$. After decryption, it executes the ticket verification $\mathsf{CheckTk}(\mathsf{tk}, st)$. If checks pass, $\mathcal{V}$ creates a challenge by sampling $s \xleftarrow{\$} \{0,1\}^{\lambda}$ and sending $\epsilon \leftarrow \mathsf{Enc}_{\mathsf{pk}^{\mathsf{E}'}_{\mathcal{U}}}(s)$ and a signature $\sigma_s \leftarrow \mathsf{Sign}_{\mathsf{sk}^{\mathsf{S}}_{\mathcal{V}}}(\epsilon)$ to $\mathcal{U}$, who obtains s' after decryption and verifying $\mathsf{Verif}_{\mathsf{pk}^{\mathsf{S}}_{\mathcal{V}}}(\epsilon, \sigma_s)$. Then, values s and s' are compared through a physical channel (see validation setting in Section 2). If the verification fails, then $\mathcal{V}$ removes c from *st i.e.,* $st \leftarrow st \setminus \{c\}$ and halts. Otherwise, both parties commonly return $s = s'$.

6 SECURITY ANALYSIS OF Applause

Security Analysis of Applause in the Computational Model Six properties are provided through computational arguments: *correctness*, *unforgeability*, *ticket privacy*, *no-double-spending* and *anonymity*, the latter being divided into *pseudonymity* and *unlinkability*. The associated games are sketched in Section 3. Below we summarise our hypothesis and arguments showing that these properties hold for Applause. We present the sketch of our proofs in this section. The full security proofs are presented in Appendix C.

Theorem 5.1. Applause *is correct under honest execution.*

Theorem 5.2. *Instantiated with an* EUF-CMA *signature, then for every* PPT $\mathcal{A}$, Applause *provides* unforgeability *and* $\Pr[\mathsf{Exp}^{\mathsf{UF}}_{\mathcal{A}}(1^{\lambda}) \rightarrow 1] \leq 2 \cdot \mathsf{Adv}^{\text{EUF-CMA}}_{\mathcal{A}}(1^{\lambda})$.

Recall that a ticket tk is defined by $\mathsf{ide}, \mathsf{idp}, r_c, \sigma_c$ where c is the hash of $\mathsf{ide}, \mathsf{idp}, r_c$, and σ_c is the signature of c under $\mathsf{sk}^{\mathsf{S}}_{\mathcal{D}}$ or $\mathsf{sk}^{\mathsf{S}}_{\mathcal{T}}$. Unforgeability of a ticket is ensured by the EUF-CMA property of the signature and the shared state *st* composed of invalidate tickets.

Theorem 5.3. *Instantiated with statistically indistinguishable randomisable keys, and an anonymous payment scheme,* Applause *provides* pseudonymity *under the following probability* $|\Pr[\mathsf{Exp}^{\mathsf{PSE}}_{\mathcal{A}}(1^{\lambda}) \rightarrow 1] - \frac{1}{2}| \leq \mathsf{Adv}^{\mathsf{PayPse}}_{\mathcal{A}}(1^{\lambda})$ *for every* PPT $\mathcal{A}$.

The identity of a user $\mathcal{U}$ is represented by its public key. It is involved in the computations through its randomised key $\mathsf{pk}'_{\mathcal{U}}$. Anonymity is ensured using the statistical indistinguishability of randomisable keys and an anonymous payment. Moreover, the numerous messages sent by $\mathcal{U}$ are unrelated to its public keys, hence unrelated to its identity. Our argument for the anonymity of $\mathcal{U}$ relies on all these facts.

Theorem 5.4. *Instantiated with an* IND-CPA *encryption scheme, for any PPT* $\mathcal{A}$, Applause *provides* privacy *under the random oracle model and the upper bound* $\Pr[\mathsf{Exp}^{\mathsf{PRIV}}_{\mathcal{A}}(1^{\lambda}) \rightarrow 1] \leq 2^{-\lambda+4} \cdot \mathsf{Adv}^{\text{IND-CPA}}_{\mathcal{A}}(1^{\lambda})$.

The privacy experiment for a ticket tk states that it is hard for a external adversary to recover tk from the transcript. In our protocol, we ensure the confidentiality of tk using an IND-CPA encryption scheme E. Recall that the adversary $\mathcal{A}$ has a read-only access to the state *st* using OLeakState. Then, at any time, $\mathcal{A}$ has every $c \leftarrow H(\mathsf{ide}, \mathsf{idp}, r_c)$ associated to each ticket. This property holds under the *Random Oracle Model* (ROM) assumption. Under this assumption the hash c is unrelated to $(\mathsf{ide}, \mathsf{idp}, r_c)$, meaning that $\mathcal{A}$ cannot learn any of ide, idp or r_c from c. To prove the privacy of tk, we rely on the secrecy of r_c the other values are accessible to the adversary but under the privacy of r_c a full ticket cannot be recover nor used. The nonce r_c is chosen uniformly at random in $\{0,1\}^{\lambda}$, the adversary can only be guessed r_c with probability $2^{-\lambda}$. Hence, r_c cannot be recovered by an adversary or with a negligible probability, hence the same apply to tk.

Theorem 5.5. Applause *provides* no-double-spending *unconditionally.*

The shared state *st* contains the hash of all tickets that have been refunded, transferred or validated, and tickets from other invalid runs. It is used to prevent a user from running the same algorithm successfully twice with the same input. The no-double-spending property is ensured by this state element. For each action, *st* has to be updated accordingly to serve as a blacklist. Based on the latter, the CheckTk function is used to ensure the validity of the provided ticket tk, it returns 1 under the condition that tk is not in *st*.

Theorem 5.6. *Instantiated with an anonymous payment having the property of randomisable keys, then for any PPT* $\mathcal{A}$, Applause *provides* unlinkability *under the probability* $|\Pr[\mathsf{Exp}^{\mathsf{UNL}}_{\mathcal{A}}(1^{\lambda}) \rightarrow 1] - \frac{1}{2}| \leq \frac{3}{2} \cdot \mathsf{Adv}^{\mathsf{PayUnl}}_{\mathcal{A}}(1^{\lambda})$.

The unlinkability ensures that a transaction performed with a ticket cannot be linked to another transaction, otherwise allowing an adversary to trace actions performed by a user. In our construction, a ticket is defined by random elements and signature of either $\mathcal{D}$ or $\mathcal{T}$, that are both unlinkable to the user. The usage of randomisable keys for signature and encryption schemes in Transfer constitute an argument for the unlinkability since a randomised key cannot be related to the original one. In the proof, we show that an adversary $\mathcal{A}$ cannot distinguish between playing at experiment $\mathsf{Exp}^{\mathsf{UNL}}_{\mathcal{A}}$ for a given b and playing at experiment $\mathsf{Exp}^{\mathsf{UNL}}_{\mathcal{A}}$ with $\bar{b}$. We obtain our result by replacing key of user $\mathcal{U}_b$ with key of user $\mathcal{U}_{\bar{b}}$. The last argument used in the proof is the unlinkability property of the anonymous payment. Without such a property, an attacker could link a transaction with the payment, breaking our definition of unlinkability. Hence, we require an unlinkable anonymous payment to ensure unlinkability.

Standard Instantiation of our Protocol. The security proofs, given in Appendix C.2, relies on the ROM, due to the usage of a hash function H to produce the digest $c \leftarrow H(\mathsf{ide}, \mathsf{idp}, r_c)$. The hash function is used to link the pair (ide, idp) identifying the ticket to the user knowing r_c. The user being the only one able to open this commitment under the pre-image resistance of the hash function, hence the only one to identify himself as the owner of the ticket. One may prefer a protocol proven secure in the standard model.

This can be easily achieved based on the Pedersen commitment [41]: consider three generators h, g_1, g_2 of a group $\mathbb{G}$ of prime order p where the discrete logarithm problem is assumed to be hard. Instead of computing $c \leftarrow H(\mathsf{ide}, \mathsf{idp}, r_c)$, commit to $c \leftarrow h^{r_c} \cdot g_1^{\mathsf{ide}} \cdot g_2^{\mathsf{idp}}$. This commitment is perfectly hiding and computationally binding. In our case, since the committed values ide and idp are already revealed, this process is computationally hiding and perfectly binding for r_c under the discrete logarithm problem. Proofs for this version of Applause can be found in Appendix C.2.

Formal Verification of Applause **in Symbolic Model**
ProVerif [6] version 2.04 has been used to formally verify the proposed protocol. This tool uses a process description based on the applied Π-calculus, a process calculus designed for the verification of cryptographic protocols. It has syntactical extension and is enriched with the notion of events, annotations that do not change the behavior of the protocol and which are inserted at precise locations to allow reasoning about the protocol's execution. Events allow checking reachability and correspondence properties. Reachability allows the investigation of which terms are available to the attacker and thus to check their secrecy. Correspondence properties have the following structure: "on every execution trace, the event e_1 is preceded by the event e_2". Authentication is formalised as correspondence properties. To check our authentication, for each sub-protocol and every entity $\mathcal{E}$, an event is inserted into the process to record the belief that $\mathcal{E}$ has accepted to run the protocol with another entity $\mathcal{E}'$, and another event to record the belief that $\mathcal{E}$ has terminated a protocol run. We refer to the authentication of $\mathcal{E}'$ to $\mathcal{E}$ by $\mathcal{E}' \rightarrow \mathcal{E}$ whenever $\mathcal{E}$ believes to complete the protocol with $\mathcal{E}'$. We then refer to the mutual authentication by $\mathcal{E} \leftrightarrow \mathcal{E}'$ when $\mathcal{E}' \rightarrow \mathcal{E}$ and $\mathcal{E} \rightarrow \mathcal{E}'$. In addition to that, ProVerif can check equivalence properties. We model privacy properties as equivalence properties. All along our analysis, we consider the Dolev-Yao attacker [14] which has a complete control of the network: the attacker eavesdrops, removes, substitutes, duplicates and delays any messages. ProVerif achieves a proof for anonymity of $\mathcal{U}_1$ and $\mathcal{U}_2$ takes less than a minute, while ticket secrecy and mutual authentication are achieved after only one second. Our Proverif formal verification files are available in [21].

7 AUDITABILITY WITH Spotlight

Applause ensures full anonymity for users. We show how to convert it into an *auditable E-ticket scheme* called Spotlight, using *auditable certificates on randomisable keys*, whose the construction of [11] is based on *Structure-preserving signatures on equivalence classes* [20]. We introduce a generic description of the algorithms for auditable certificates on randomisable keys:

OKeyGen(1^λ). Outputs certification keys (skc, pkc).
Certify$\langle \mathcal{U}(\mathsf{sk}), \mathcal{J}(\mathsf{skc}) \rangle$. Outputs a certificate *cert* for pk.
CheckCert(pkc, *cert*). Outputs a bit $b \in \{0, 1\}$.
RandID(sk, pk, *cert*). Outputs a randomised triple (sk′, pk′, *cert*′).
Audit(skc, *cert*). Opens *cert* based on skc and recovers pk.

Spotlight relaxes user anonymity achieved in Applause. We requires the possibility to randomise certificates with the keys related to this certificate, as suggested by the algorithm RandID. The main modification is to append a certificate with the previous elements of the shared state, it is now composed of a triples $(c, cert, b)$, where c is a hash, *cert* a certificate and $b \in \{0, 1\}$ is a bit describing the validity of the related ticket.

7.1 Overview of Spotlight

We start by given an highlight on the necessary modifications to apply on Applause to obtain an auditable version, giving the overall intuitions of Spotlight. The security analysis in Appendix D.

JKeyGen(1^λ): Runs and outputs OKeyGen(1^λ) $\rightarrow (\mathsf{sk}_{\mathcal{J}}, \mathsf{pk}_{\mathcal{J}})$.
UKeyGen: Additionally obtains the certificate $cert_{\mathcal{U}}$ from the certification algorithm Certify$\langle \mathcal{U}(\mathsf{sk}^{\mathsf{S}}_{\mathcal{U}}, \mathsf{pk}^{\mathsf{S}}_{\mathcal{U}}), \mathcal{J}(\mathsf{sk}_{\mathcal{J}}) \rangle$. It returns it as part of $\mathsf{pk}_{\mathcal{U}}$.
Purchase: $\mathcal{U}$ updates its signature's keys using the randomisation algorithm RandID$(\mathsf{sk}^{\mathsf{S}}_{\mathcal{U}}, \mathsf{pk}^{\mathsf{S}}_{\mathcal{U}}, cert_{\mathcal{U}}) \rightarrow (\mathsf{sk}^{\mathsf{S}'}_{\mathcal{U}}, \mathsf{pk}^{\mathsf{S}'}_{\mathcal{U}}, cert'_{\mathcal{U}})$, and sends $\mathsf{pk}'_{\mathcal{U}} = (\mathsf{pk}^{\mathsf{S}'}_{\mathcal{U}}, \mathsf{pk}^{\mathsf{E}'}_{\mathcal{U}})$, $cert'_{\mathcal{U}}$, $\psi = \mathsf{Enc}_{\mathsf{pk}^{\mathsf{E}}_{\mathcal{D}}}(\mathsf{ide}, \mathsf{idp}, r_c, c)$ and $\sigma_\psi = \mathsf{Sign}_{\mathsf{sk}^{\mathsf{S}'}_{\mathcal{U}}}(\psi)$. $\mathcal{D}$ executes the certification verification function CheckCert$(\mathsf{pk}_{\mathcal{J}}, cert'_{\mathcal{U}})$ and executes $\mathsf{Verif}_{\mathsf{pk}^{\mathsf{S}}_{\mathcal{U}}}(\psi, \sigma_\psi)$. $\mathcal{D}$ updates st as $st \leftarrow st \cup \{c, cert'_{\mathcal{U}}, 0\}$ and inverts the bit to obtain $\{c, cert'_{\mathcal{U}}, 1\}$ when the ticket is paid.
CheckTk: Additionally takes as input a certificate *cert*, executes CheckCert$(\mathsf{pk}_{\mathcal{J}}, cert)$ and checks the state $(c, \cdot, 1) \in st$ instead of $c \in st$. If all verification success, then we update $(c, \cdot, 1)$ to $(c, cert, 0)$.
Refund: $\mathcal{U}$ updates its keys as described in Purchase and sends $\mathsf{pk}'_{\mathcal{U}}, cert'_{\mathcal{U}}, \psi = \mathsf{Enc}_{\mathsf{pk}^{\mathsf{E}}_{\mathcal{D}}}(\mathsf{tk}), \mathsf{Sign}_{\mathsf{sk}^{\mathsf{S}}_{\mathcal{U}}}(\psi)$ to $\mathcal{D}$ as its first message. The signature is verified and then CheckTk executed with $cert'_{\mathcal{U}}$ as its additional input. In case of a payment failure, $\mathcal{D}$ reverts $(c, \cdot, 0)$ to $(c, cert'_{\mathcal{U}}, 1)$.
Transfer: Both $\mathcal{U}_1$ and $\mathcal{U}_2$ respectively send $cert'_{\mathcal{U}_1}$ and $cert'_{\mathcal{U}_2}$ to $\mathcal{T}$, and verify $cert'_{\mathcal{U}_1}$ and $cert'_{\mathcal{U}_2}$ using CheckCert. Eventually, if Purchase failed, $\mathcal{T}$ reverts $(c, \cdot, 0)$ to $(c, cert'_{\mathcal{U}_1}, 1)$.
Validate: $\mathcal{U}$ updates its keys and sends $\mathsf{Enc}_{\mathsf{pk}^{\mathsf{E}}_{\mathcal{V}}}(\mathsf{pk}'_{\mathcal{U}}, cert'_{\mathcal{U}}, \mathsf{tk})$ as its first message. $\mathcal{V}$ inputs $cert'_{\mathcal{U}}$ into CheckCert .
Audit$(\mathsf{sk}_{\mathcal{J}}, (\cdot, \cdot, cert))$ Returns Audit$(\mathsf{sk}_{\mathcal{J}}, cert) \rightarrow \mathsf{pk}_{\mathcal{U}}$.

Theorem 5.7. *Based an anonymous payment, an* EUF-CMA *signature with randomisable keys and an* IND-CPA *encryption scheme, auditable certificates on randomisable keys,* Spotlight *ensures anonymity, unlinkability, no-double-spending, unforgeability and ticket privacy.*

7.2 Formal Description of Spotlight

We have chosen to present Applause based on the Random Oracle Model, we stress that the modification to push Applause from the ROM into the standard model, can be apply in Spotlight in the standard model as well. To obtain auditability, we have to modify algorithms UKeyGen, Purchase, Transfer, Refund and Validate. In order to facilitate the readability, we repeat the definition of all the algorithm.

KeyGen(1^λ): Generates a signature keys S.KeyGen$(1^\lambda) \rightarrow (\mathsf{sk}^{\mathsf{S}}, \mathsf{pk}^{\mathsf{S}})$, an encryption keys E.KeyGen$(1^\lambda) \rightarrow (\mathsf{sk}^{\mathsf{E}}, \mathsf{pk}^{\mathsf{E}})$, an payment keys P.KeyGen$(1^\lambda) \rightarrow (\mathsf{sk}^{\mathsf{P}}, \mathsf{pk}^{\mathsf{P}})$, outputs $(\mathsf{sk} \leftarrow (\mathsf{sk}^{\mathsf{S}}, \mathsf{sk}^{\mathsf{E}}, \mathsf{sk}^{\mathsf{P}}), \mathsf{pk} \leftarrow (\mathsf{pk}^{\mathsf{S}}, \mathsf{pk}^{\mathsf{E}}, \mathsf{pk}^{\mathsf{P}}))$.

JKeyGen(1^λ): Returns OKeyGen(1^λ) $\rightarrow$ $(\mathsf{sk}_\mathcal{J}, \mathsf{pk}_\mathcal{J})$.

DKeyGen(1^λ): Returns $(\mathsf{sk}_\mathcal{D}, \mathsf{pk}_\mathcal{D}) \leftarrow \mathsf{KeyGen}(1^\lambda)$.

TKeyGen(1^λ): Returns $(\mathsf{sk}_\mathcal{T}, \mathsf{pk}_\mathcal{T}) \leftarrow \mathsf{KeyGen}(1^\lambda)$.

UKeyGen(1^λ): Computes $((\mathsf{sk}^{\mathsf{E}'}_\mathcal{U}, \mathsf{sk}^{\mathsf{S}'}_\mathcal{U}), (\mathsf{pk}^{\mathsf{E}'}_\mathcal{U}, \mathsf{pk}^{\mathsf{S}'}_\mathcal{U})) \leftarrow \mathsf{KeyGen}(1^\lambda)$, and then obtains a certificate $cert_\mathcal{U}$ from the certification $\mathsf{Certify}\langle\mathcal{U}(\mathsf{sk}^{\mathsf{S}'}_\mathcal{U}, \mathsf{pk}^{\mathsf{S}'}_\mathcal{U}), \mathcal{J}(\mathsf{sk}_\mathcal{J})\rangle$ the certification algorithm and returns $(\mathsf{sk} = (\mathsf{sk}^{\mathsf{E}'}_\mathcal{U}, \mathsf{sk}^{\mathsf{S}'}_\mathcal{U}), \mathsf{pk}_\mathcal{U} = (\mathsf{pk}^{\mathsf{E}'}_\mathcal{U}, \mathsf{pk}^{\mathsf{S}'}_\mathcal{U}, cert_\mathcal{U}))$.

VKeyGen(1^λ): Generates $\mathsf{S.KeyGen}(1^\lambda) \rightarrow (\mathsf{sk}^{\mathsf{S}}_\mathcal{V}, \mathsf{pk}^{\mathsf{S}}_\mathcal{V})$ a signature key pair, and $\mathsf{E.KeyGen}(1^\lambda) \rightarrow (\mathsf{sk}^{\mathsf{E}}_\mathcal{V}, \mathsf{pk}^{\mathsf{E}}_\mathcal{V})$ an encryption key pair. It obtains the signature $\mathsf{Sign}_{\mathsf{sk}^{\mathsf{S}}_\mathcal{D}}(\mathsf{pk}^{\mathsf{S}}_\mathcal{V}, \mathsf{pk}^{\mathsf{E}}_\mathcal{V}) \rightarrow cert_\mathcal{V}$ querying $\mathcal{D}$ to produce a certificate on its key. Returns $\mathsf{sk}_\mathcal{V} \leftarrow (\mathsf{sk}^{\mathsf{S}}_\mathcal{V}, \mathsf{sk}^{\mathsf{E}}_\mathcal{V})$, $\mathsf{pk}_\mathcal{V} \leftarrow (\mathsf{pk}^{\mathsf{S}}_\mathcal{V}, \mathsf{pk}^{\mathsf{E}}_\mathcal{V}, cert_\mathcal{V})$.

Purchase$\langle\mathcal{U}(\mathsf{sk}_\mathcal{U}, (\mathsf{ide}, \mathsf{idp})), \mathcal{D}(\mathsf{sk}_\mathcal{D}, st)\rangle$: $\mathcal{U}$ updates its signature's keys using the randomization algorithm $\mathsf{RandID}(\mathsf{sk}^{\mathsf{S}}_\mathcal{U}, \mathsf{pk}^{\mathsf{S}}_\mathcal{U}, cert_\mathcal{U}) \rightarrow (\mathsf{sk}^{\mathsf{S}'}_\mathcal{U}, \mathsf{pk}^{\mathsf{S}'}_\mathcal{U}, cert'_\mathcal{U})$. It samples $r_c \xleftarrow{\$} \{0,1\}^\lambda$. Then it sends $\mathsf{pk}'_\mathcal{U} = (\mathsf{pk}^{\mathsf{S}'}_\mathcal{U}, \mathsf{pk}^{\mathsf{E}'}_\mathcal{U})$, $cert'_\mathcal{U}$, $\psi = \mathsf{Enc}_{\mathsf{pk}^{\mathsf{E}}_\mathcal{D}}(\mathsf{ide}, \mathsf{idp}, r_c)$ and $\sigma_\psi = \mathsf{Sign}_{\mathsf{sk}^{\mathsf{S}'}_\mathcal{U}}(\psi)$. $\mathcal{D}$ starts by checking that the pair (idp, ide) has no been purchased. Then, $\mathcal{D}$ computes $c \leftarrow H(\mathsf{ide}, \mathsf{idp}, r_c)$ and executes the certification verification algorithm $\mathsf{CheckCert}(\mathsf{pkc}, cert'_\mathcal{U})$ and also verifies $\mathsf{Verif}_{\mathsf{pk}^{\mathsf{S}}_\mathcal{U}}(\psi, \sigma_\psi)$. $\mathcal{D}$ updates its state $st \leftarrow st \cup \{c, cert'_\mathcal{U}, 0\}$ signs $\sigma_c \leftarrow \mathsf{Sign}_{\mathsf{sk}^{\mathsf{S}}_\mathcal{D}}(c)$, before sending σ_c to $\mathcal{U}$. Once the signature verified, $\mathcal{U}$ pays with $\mathsf{Pay}\langle\mathcal{U}(\mathsf{sk}^{\mathsf{P}}_\mathcal{U}), \mathcal{D}(\mathsf{sk}^{\mathsf{P}}_\mathcal{D})\rangle \rightarrow \mathcal{U}(b), \mathcal{D}(b)$. If the payment successful (*i.e.,* b equals 1) then $\mathcal{U}$ returns b and the ticket $\mathsf{tk} \leftarrow ((\mathsf{ide}, \mathsf{idp}, r_c), \sigma_c)$, $\mathcal{D}$ returns b and an updated state $st \leftarrow st \cup \{c, cert'_\mathcal{U}, 1\}$.

CheckTk(tk, *st*, *cert*): Parses $\mathsf{tk} \xrightarrow{p} ((\mathsf{ide}, \mathsf{idp}, r_c), \sigma_c)$, computes $c \leftarrow H(\mathsf{ide}, \mathsf{idp}, r_c)$ and verifies if the signature is valid by executing $\mathsf{Verif}_{\mathsf{pk}^{\mathsf{S}}_\mathcal{D}}(c, \sigma_c)$ or $\mathsf{Verif}_{\mathsf{pk}^{\mathsf{S}}_\mathcal{T}}(c, \sigma_c)$ validate. Also verify the certificate by executing $\mathsf{CheckCert}(\mathsf{pk}_\mathcal{J}, cert)$ Then, checks that c is not a blacklisted ticket *i.e.,* $(c, \cdot, 1) \in st$. If all passes, it updates $(c, \cdot, 1)$ to $(c, cert, 0)$ and returns st.

Refund$\langle\mathcal{U}(\mathsf{sk}_\mathcal{U}, \mathsf{tk}), \mathcal{D}(\mathsf{sk}_\mathcal{D}, st)\rangle$: $\mathcal{U}$ executes $\mathsf{RandID}(\mathsf{sk}^{\mathsf{S}}_\mathcal{U}, \mathsf{pk}^{\mathsf{S}}_\mathcal{U}, cert_\mathcal{U}) \rightarrow (\mathsf{sk}^{\mathsf{S}'}_\mathcal{U}, \mathsf{pk}^{\mathsf{S}'}_\mathcal{U}, cert'_\mathcal{U})$ and sends $\mathsf{pk}'_\mathcal{U}, cert'_\mathcal{U}, \psi = \mathsf{Enc}_{\mathsf{pk}^{\mathsf{E}}_\mathcal{D}}(\mathsf{tk}), \mathsf{Sign}_{\mathsf{sk}^{\mathsf{S}}_\mathcal{U}}(\psi)$ to $\mathcal{D}$ as its first message. After the decryption and verification of the signature $\mathcal{D}$ checks the received ticket using $\mathsf{CheckTk}(\mathsf{tk}, st, cert'_\mathcal{U}) \rightarrow st$.
Both participants starts a refund (*i.e.,* a payment with a negative amount) using $\mathsf{Pay}\langle\mathcal{U}(\mathsf{sk}^{\mathsf{P}}_\mathcal{U}), \mathcal{D}(\mathsf{sk}^{\mathsf{P}}_\mathcal{D})\rangle \rightarrow \mathcal{U}(b), \mathcal{D}(b)$. If $b = 0$, $\mathcal{D}$ reverts its state by updating $(c, \cdot, 0)$ to $(c, cert'_\mathcal{U}, 1)$. Both parties return b, and $\mathcal{D}$ additionally returns st and tk.

Transfer$\langle\mathcal{U}_1(\mathsf{sk}'_{\mathcal{U}_1}, \mathsf{tk}_1, \mathsf{pk}'_{\mathcal{U}_2}), \mathcal{T}(\mathsf{sk}_\mathcal{T}, st), \mathcal{U}_2(\mathsf{sk}'_{\mathcal{U}_2}, \mathsf{pk}'_{\mathcal{U}_1}, (\mathsf{ide}, \mathsf{idp}))\rangle$: $\mathcal{U}_1$ computes the signature $\mathsf{Sign}_{\mathsf{sk}^{\mathsf{S}'}_{\mathcal{U}_1}}(\mathsf{pk}'_{\mathcal{U}_2}, r_c) \rightarrow \sigma_{T,1}$ and sends $(\mathsf{pk}'_{\mathcal{U}_1}, cert'_{\mathcal{U}_1}, \sigma_{T,1}, \mathsf{Enc}_{\mathsf{pk}^{\mathsf{E}}_\mathcal{T}}(c, \mathsf{tk}))$ to $\mathcal{T}$. In the meantime, $\mathcal{U}_2$ signs $\mathsf{Sign}_{\mathsf{sk}^{\mathsf{S}'}_{\mathcal{U}_2}}(\mathsf{pk}'_{\mathcal{U}_1}) \rightarrow \sigma_{T,2}$ and sends the message $(\mathsf{pk}'_{\mathcal{U}_2}, cert'_{\mathcal{U}_2}, \sigma_{T,2})$ to $\mathcal{T}$. Once it received $\sigma_{T,1}$ and $\sigma_{T,2}$, $\mathcal{T}$ checks $\mathsf{Verif}_{\mathsf{pk}^{\mathsf{S}'}_{\mathcal{U}_1}}((\mathsf{pk}'_{\mathcal{U}_2}, r_c), \sigma_{T,1})$ and $\mathsf{Verif}_{\mathsf{pk}^{\mathsf{S}'}_{\mathcal{U}_2}}(\mathsf{pk}'_{\mathcal{U}_1}, \sigma_{T,2})$, and halts any fails. Then $\mathcal{T}$ executes the certificate verification $\mathsf{CheckCert}(\mathsf{pk}_\mathcal{J}, cert'_{\mathcal{U}_2})$ and checks the validity of the received ticket with $\mathsf{CheckTk}(\mathsf{tk}, st, cert'_{\mathcal{U}_1}) \rightarrow st$ the ticket verification algorithm. If st was updated, $\mathcal{T}$ signs $\sigma_p \leftarrow \mathsf{Sign}_{\mathsf{sk}^{\mathsf{S}}_\mathcal{T}}(\mathsf{ide}, \mathsf{idp}, \mathsf{pk}'_{\mathcal{U}_2}, \mathsf{pk}'_{\mathcal{U}_1})$. σ_p is transmitted to $\mathcal{U}_2$ and verified. $\mathcal{U}_2$ initiates the purchase for the place (ide, idp) with $\mathsf{Purchase}\langle\mathcal{U}_2(\mathsf{sk}_{\mathcal{U}_2}, (\mathsf{ide}, \mathsf{idp})), \mathcal{T}(\mathsf{sk}_\mathcal{T}, st)\rangle$ resulting to $(\mathcal{U}_2(b_p, \mathsf{tk}_2), \mathcal{T}(b_p, st))$ where $\mathcal{T}$ does not check that (ide,idp) where already attributed. If b_p equals 0, then $\mathcal{T}$ revert the state of c from $(c, \cdot, 0)$ to $(c, cert'_{\mathcal{U}_1}, 1)$ (previously added in st during the CheckTk execution) and halts. Otherwise, on success, $\sigma'_p = \mathsf{Sign}_{\mathsf{sk}^{\mathsf{S}}_\mathcal{T}}(\mathsf{ide}, \mathsf{idp}, \mathsf{pk}'_{\mathcal{U}_1}, \mathsf{pk}'_{\mathcal{U}_2})$ is sent to $\mathcal{U}_1$, verified by $\mathcal{U}_1$. $\mathsf{Refund}\langle\mathcal{U}_1(\mathsf{sk}_{\mathcal{U}_1}, \mathsf{tk}_1), \mathcal{T}(\mathsf{sk}_\mathcal{T}, st \setminus \{c\})\rangle \rightarrow \mathcal{U}_1(b_r), \mathcal{T}(b_r)$ is executed between $\mathcal{U}_1$ and $\mathcal{T}$. Last $\mathcal{U}_1$ returns b_r, $\mathcal{T}$ update the shared state st and returns b_r along st, and $\mathcal{U}_2$ returns tk_2.

Validate$\langle\mathcal{U}(\mathsf{sk}_\mathcal{U}, \mathsf{tk}), \mathcal{V}(\mathsf{sk}_\mathcal{V}, st)\rangle$: $\mathcal{V}$ starts by providing $\mathsf{pk}_\mathcal{V}$ to $\mathcal{U}$. The latter parses it and runs $\mathsf{Verif}_{\mathsf{pk}^{\mathsf{S}}_\mathcal{D}}(\mathsf{pk}_\mathcal{V}, cert_\mathcal{V}) \rightarrow b$. On b equals 0, the protocol is aborted. Otherwise, $\mathcal{U}$ executes $\mathsf{RandID}(\mathsf{sk}^{\mathsf{S}}_\mathcal{U}, \mathsf{pk}^{\mathsf{S}}_\mathcal{U}, cert_\mathcal{U}) \rightarrow (\mathsf{sk}^{\mathsf{S}'}_\mathcal{U}, \mathsf{pk}^{\mathsf{S}'}_\mathcal{U}, cert'_\mathcal{U})$ and sends $cert'_\mathcal{U}$, $\mathsf{Enc}_{\mathsf{pk}^{\mathsf{E}}_\mathcal{V}}(\mathsf{pk}'_\mathcal{U}, \mathsf{tk})$ to $\mathcal{V}$. It decrypts the message and executes $\mathsf{CheckTk}(\mathsf{tk}, st, cert'_\mathcal{U})$. If both checks passes, $\mathcal{V}$ sample a challenge value $s \xleftarrow{\$} \{0,1\}^\lambda$ and sends $\epsilon = \mathsf{Enc}_{\mathsf{pk}^{\mathsf{E}'}_\mathcal{U}}(s)$ and a signature $\sigma_s = \mathsf{Sign}_{\mathsf{sk}^{\mathsf{S}}_\mathcal{V}}(\epsilon)$ to $\mathcal{U}$ who obtain s' after decryption and verifying $\mathsf{Verif}_{\mathsf{pk}^{\mathsf{S}}_\mathcal{V}}(\epsilon, \sigma_s)$. The values s and s' are compared through a physical channel (as explained in Section 2). Both parties commonly return $s = s'$. If the verification has fail the $\mathcal{V}$ also removes c from st *i.e.,*$st \leftarrow st \setminus \{c\}$.

Audit$(\mathsf{sk}_\mathcal{J}, (\cdot, \cdot, cert))$: Returns $\mathsf{Audit}(\mathsf{sk}_\mathcal{J}, cert) \rightarrow \mathsf{pk}_\mathcal{U}$.

8 CONCLUSION

We presented Applause, a ticketing system that preserves the physical aspects of paper tickets while ensuring that any user can buy, refund, validate, or transfer a ticket. The system additionally safeguards the privacy of users. Applause achieves unforgeability, no-double-spending, privacy of the tickets, and anonymity of users. In particular, anonymity ensured by Applause highly depends on the anonymity property of the used payment method, that we chosen to include in a generic fashion. We have also formally verified the security properties of Applause using ProVerif, proving its security in the symbolic model for an unbounded number of sessions. We have additionally extended Applause to obtain an auditable version called Spotlight, whereby the judge can reveal the identity of all users, whereas other entities are unable to discern whether a user has purchased a ticket and for which event.

ACKNOWLEDGMENTS

The authors would like to thank the anonymous reviewers for their constructive comments. This work has been partially the French BPI project D4N and by the SEVERITAS project ANR-20-CE39-0009.

REFERENCES

[1] N. Asokan, Victor Shoup, and Michael Waidner. 1998. Asynchronous protocols for optimistic fair exchange. English (US). *Proceedings of the IEEE Computer Society Symposium on Research in Security and Privacy*, 86–99. Proceedings of the 1998 IEEE Symposium on Security and Privacy ; Conference date: 03-05-1998 Through 06-05-1998.

[2] Aventus. 2016. https://aventus.io/. (2016).

[3] Foteini Baldimtsi, Melissa Chase, Georg Fuchsbauer, and Markulf Kohlweiss. 2015. Anonymous transferable e-cash. In *International Workshop on Public Key Cryptography*. Springer.

[4] Balthazar Bauer, Georg Fuchsbauer, and Chen Qian. 2021. Transferable e-cash: a cleaner model and the first practical instantiation. In *IACR International Conference on Public-Key Cryptography*. Springer, 559–590.

[5] Osman Bicer and Alptekin Kupcu. 2019. Versatile abs: usage limited, revocable, threshold traceable, authority hiding, decentralized attribute based signatures. Cryptology ePrint Archive, Paper 2019/203. https://eprint.iacr.org/2019/203. (2019). https://eprint.iacr.org/2019/203.

[6] Bruno Blanchet. 2014. Automatic Verification of Security Protocols in the Symbolic Model: The Verifier ProVerif. In *Foundations of Security Analysis and Design VII*. Springer. DOI: 10.1007/978-3-319-10082-1_3.

[7] Jan Camenisch, Susan Hohenberger, Markulf Kohlweiss, Anna Lysyanskaya, and Mira Meyerovich. 2006. How to win the clonewars: efficient periodic n-times anonymous authentication. In *Proceedings of the 13th ACM Conference on Computer and Communications Security* (CCS '06). Association for Computing Machinery, Alexandria, Virginia, USA. DOI: 10.1145/1180405.1180431.

[8] CashApp. 2018. CashApp. https://www.cash.app/. (2018).

[9] David Chaum. 1985. Security without identification: transaction systems to make big brother obsolete. *Communications of the ACM*, 28, 10, 1030–1044.

[10] Liqun Chen, Matthias Enzmann, Ahmad-Reza Sadeghi, Markus Schneider, and Michael Steiner. 2005. A privacy-protecting coupon system. In *Financial Cryptography and Data Security: 9th International Conference, FC 2005, Roseau, The Commonwealth Of Dominica, February 28–March 3, 2005. Revised Papers 9*. Springer, 93–108.

[11] Aisling Connolly, Jérôme Deschamps, Pascal Lafourcade, and Octavio Perez Kempner. 2022. Protego: efficient, revocable and auditable anonymous credentials with applications to hyperledger fabric. In *International Conference on Cryptology in India*. Springer, 249–271.

[12] Yao Cui, Izak Duenyas, and Özge Şahin. 2014. Should event organizers prevent resale of tickets? *Management Science*, 60, 9, 2160–2179.

[13] Alex de Vries. 2018. Bitcoin's growing energy problem. 2, (May 2018), 801–805. DOI: 10.1016/j.joule.2018.04.016.

[14] D. Dolev and A. Yao. 1983. On the security of public key protocols. *IEEE Transactions on Information Theory*. DOI: 10.1109/TIT.1983.1056650.

[15] Yuhao Dong, Ian Goldberg, Sergey Gorbunov, and Raouf Boutaba. 2022. Astrape: anonymous payment channels with boring cryptography. In *International Conference on Applied Cryptography and Network Security*. Springer.

[16] Taher ElGamal. 1985. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE transactions on information theory*.

[17] EMVCo. 2011. Book 1: application independent icc to terminal interface requirements.

[18] LLC EMVCo. 2022. Emv payment tokenisation specification technical framework v2.3. *EMVCo: Foster City, CA, USA*.

[19] Shimon Even. 1983. A protocol for signing contracts. *SIGACT News*, 15, 1, (Jan. 1983), 34–39. DOI: 10.1145/1008908.1008913.

[20] Georg Fuchsbauer, Christian Hanser, and Daniel Slamanig. 2019. Structure-preserving signatures on equivalence classes and constant-size anonymous credentials. *Journal of Cryptology*.

[21] [n. d.] Git for transferable, auditable and anonymous ticketing protocol. https://gitlab.limos.fr/palafour/asiaccs-24-applause/.

[22] Ivan Gudymenko. 2013. On protection of the user's privacy in ubiquitous e-ticketing systems based on RFID and NFC technologies. In *PECCS 2013 - Proceedings of the 3rd International Conference on Pervasive Embedded Computing and Communication Systems*. SciTePress.

[23] Ivan Gudymenko, Felipe Sousa, and Stefan Kopsell. 2014. A simple and secure e-ticketing system for intelligent public transportation based on NFC. In *The First International Conference on IoT in Urban Space, Urb-IoT*. ICST. https://xxdoi.org/10.4108/icst.urb-iot.2014.257244.

[24] N. Abdul Hamid, M. F. Al A'zhim, and M. L. Yap. 2012. E-ticketing system for football events in malaysia. In *7th International Conference for Internet Technology and Secured Transactions, ICITST*. IEEE. https://ieeexplore.ieee.org/document/6470872/.

[25] Jinguang Han, Liqun Chen, Steve Schneider, Helen Treharne, and Stephan Wesemeyer. 2021. Privacy-preserving electronic ticket scheme with attribute-based credentials. *IEEE Trans. Dependable Secur. Comput.* https://xxdoi.org/10.1109/TDSC.2019.2940946.

[26] Wan Huzaini Wan Hussin, Paul Coulton, and Reuben Edwards. 2005. Mobile ticketing system employing trustzone technology. In *International Conference on Mobile Business*. IEEE Computer Society. https://xxdoi.org/10.1109/ICMB.2005.71.

[27] International Organization for Standardization. 2006. Information technology — automatic identification and data capture techniques — qr code 2005 bar code symbology specification. ISO/IEC 18004:2006. (2006).

[28] Florian Kerschbaum, Hoon Wei Lim, and Ivan Gudymenko. 2013. Privacy-preserving billing for e-ticketing systems in public transportation. In *Proceedings of the 12th annual ACM Workshop on Privacy in the Electronic Society, WPES 2013*. ACM. https://xxdoi.org/10.1145/2517840.2517848.

[29] Phillip Leslie and Alan Sorensen. 2014. Resale and rent-seeking: an application to ticket markets. *Review of Economic Studies*, 81, 1, 266–300.

[30] Xiaoshan Li, Zhiming Liu, and Zhensheng Guo. 2001. Formal object-oriented analysis and design of an online ticketing system. In *8th Asia-Pacific Software Engineering Conference (APSEC 2001)*. IEEE Computer Society. https://xxdoi.org/10.1109/APSEC.2001.991486.

[31] Xuelian Li, Jie Niu, Juntao Gao, and Yue Han. 2019. Secure electronic ticketing system based on consortium blockchain. *KSII Trans. Internet Inf. Syst.* https://xxdoi.org/10.3837/tiis.2019.10.022.

[32] Weiwei Liu, Yi Mu, and Guomin Yang. 2014. An efficient privacy-preserving e-coupon system. In *International Conference on Information Security and Cryptology*. Springer.

[33] Akash Madhusudan, Mahdi Sedaghat, Philipp Jovanovic, and Bart Preneel. 2022. Nirvana: instant and anonymous payment-guarantees. *IACR Cryptol. ePrint Arch.*, 872.

[34] Hemanta K Maji, Manoj Prabhakaran, and Mike Rosulek. 2011. Attribute-based signatures. In *Cryptographers' track at the RSA conference*. Springer, 376–392.

[35] Milica Milutinovic, Koen Decroix, Vincent Naessens, and Bart De Decker. 2015. Privacy-preserving public transport ticketing system. In *Data and Applications Security and Privacy- 29th Annual IFIP WG 11.3 Working Conference, DBSec*. Springer. https://xxdoi.org/10.1007/978-3-319-20810-7%5C_9.

[36] Omid Mir, Daniel Slamanig, and René Mayrhofer. 2023. Threshold delegatable anonymous credentials with controlled and fine-grained delegation. *IEEE Transactions on Dependable and Secure Computing*.

[37] Lekshmi S. Nair, V. S. Arun, and Sijo Joseph. 2015. Secure e-ticketing system based on mutual authentication using RFID. In *Proceedings of the Third International Symposium on Women in Computing and Informatics, WCI 2015*. ACM. https://xxdoi.org/10.1145/2791405.2791573.

[38] Lan Nguyen. 2006. Privacy-protecting coupon system revisited. In *Financial Cryptography and Data Security: 10th International Conference, FC 2006 Anguilla, British West Indies, February 27-March 2, 2006 Revised Selected Papers 10*. Springer, 266–280.

[39] Johanna Nieminen, Teemu Savolainen, Markus Isomaki, Basavaraj Patil, Zach Shelby, and Carles Gomez. 2015. IPv6 over BLUETOOTH(R) Low Energy. RFC 7668. (Oct. 2015). DOI: 10.17487/RFC7668.

[40] PaySafeCard. 2018. Paysafecard. https://www.paysafecard.com/fr/. (2018).

[41] Torben Pryds Pedersen. 2001. Non-interactive and information-theoretic secure verifiable secret sharing. In *Advances in Cryptology—CRYPTO'91: Proceedings*. Springer, 129–140.

[42] GET Protocol. 2016. GUTS Ticketing. https://guts.tickets/. (2016).

[43] Jason Reid, Juan M Gonzalez Nieto, Tee Tang, and Bouchra Senadji. 2007. Detecting relay attacks with timing-based protocols. In *Proceedings of the 2nd ACM symposium on Information, computer and communications security*, 204–213.

[44] Nicolas Van Saberhagen. 2013. Monero Research Paper. (2013). https://github.com/monero-project/research-lab/blob/master/whitepaper/whitepaper.pdf.

[45] Claus-Peter Schnorr. 1991. Efficient signature generation by smart cards. *Journal of cryptology*.

[46] Johannes Sedlmeir, Hans Buhl, Gilbert Fridgen, and Robert Keller. 2020. The energy consumption of blockchain technology: beyond myth. *Business & Information Systems Engineering*, 62, (Dec. 2020). DOI: 10.1007/s12599-020-00656-x.

[47] Marc Sel, Stefaan Seys, and Eric R. Verheul. 2008. The security of mass transport ticketing systems. In *ISSE 2008 - Securing Electronic Busines Processes, Highlights of the Information Security Solutions Europe 2008 Conference*. https://xxdoi.org/10.1007/978-3-8348-9283-6%5C_37.

[48] Isamu Teranishi, Jun Furukawa, and Kazue Sako. 2004. K-times anonymous authentication (extended abstract). In *Advances in Cryptology - ASIACRYPT 2004*. Springer.

[49] Hitesh Tewari and Arthur Hughes. 2016. Fully anonymous transferable ecash. *Cryptology ePrint Archive*.

[50] Minhui Xie, Mark Tomlinson, and Bobby Bodenheimer. 2004. Interface design for a modern software ticketing system. In *Proceedings of Annual Southeast Regional Conference*. ACM, Huntsville, Alabama. https://xxdoi.org/10.1145/986537.986566.

[51] Yuan YuanJiang and Ji Ting Zhou. 2022. Ticketing system based on NFT. In *24th IEEE International Workshop on Multimedia Signal Processing, MMSP 2022*. IEEE. https://xxdoi.org/10.1109/MMSP55362.2022.9948706.

A IMPLEMENTATION OF Applause

A proof-of-concept implementation of Applause in Rust is available in [21]. This implementation showcases the protocol's completeness and highlights the anticipated low computational cost of the cryptographic operations within our proposal. In our implementation, we have measure the impact of up to a thousand participants event purchasing a ticket. After the purchase, all acquired tickets are transferred to load the shared state while generating the new (transferred) tickets. Subsequently, the refund and validation processes are executed with these tickets.

Our analysis specifically focuses on cryptographic operations, excluding communication time and payment processes. Communication time is highly dependent on the network infrastructure, and our protocol allows for multiple anonymous payment methods to be integrated. These presented execution times should be considered as a baseline.

We rely on standard cryptographic primitives: we used the curve25519 curve for ElGamal [16] and Schnorr signatures [45]. To achieve key randomisation for Schnorr scheme, we rely on the generic algorithm generating a new key pair. Benchmarks over 200 iterations from 1 ticket in the database up to 1000 tickets in it have highlight the constant time execution of all process. On an Ubuntu laptop equipped with an Intel i7-12800H processor and 32 GB of RAM give average execution times of 10 ms for a purchase, 40 ms for a transfer, 15 ms for a refund and 30 ms for a validation, hence emphasizing the high efficiency of all cryptographic operations. Hence, the overhead brought by securing the protocol is acceptable. Moreover, these timings are within the same order of magnitude as 1 *Round-Trip Time*. The overhead introduced by securing the protocol appears acceptable at any step of the process. Finally, the measurement of the shared state update shows that it is negligible, always lower than 1 millisecond. Therefore, our instantiation is efficient and scalable.

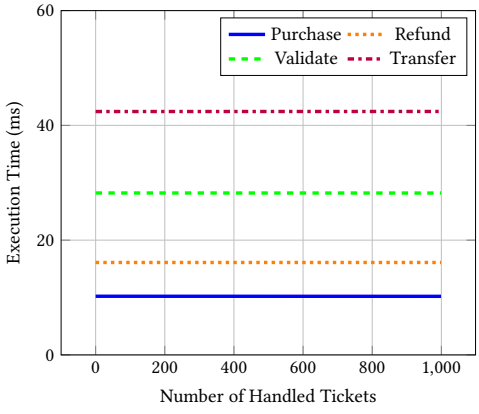


Figure 7: Mean execution times (in milliseconds) of the cryptographic operations in the protocol over 200 iterations, depending on the number of handled tickets.

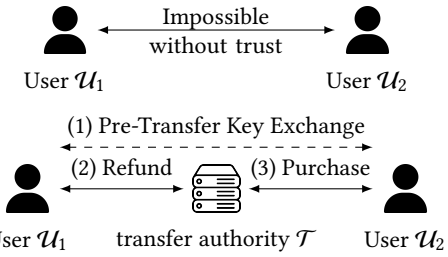


Figure 8: Exchange without trust and based on central transfer authority.

B TRANSFER SETTINGS

While the sale of a ticket is relatively straightforward, ticket transfer can be realized under multiple scenarios. In *e-cash* system, only two entities are typically involved in the transfer, but the verification is not strong enough to guarantee that, at any time, two entities does not hold the same coin. This is required in *e-ticket*. It was demonstrated in [19] that achieving a fair transfer in the setting depicted in Fig. 8, between two users without a trusted third party is impossible. Another way to come to this result, in our context, is described below.

Property 1. *Consider two users $\mathcal{U}_1$ and $\mathcal{U}_2$. Assume that $\mathcal{U}_1$ holds a honestly generated ticket and that $\mathcal{U}_2$ tries to acquire it via a transfer protocol where it trades it against a payment. Assuming that no external interaction is executed during the protocol. Their exist no protocol leading to a secure exchange between $\mathcal{U}_1$ and $\mathcal{U}_2$ ending with $\mathcal{U}_2$ being the only holder of the exchanged ticket.*

Proof. Considering $\mathcal{U}_1$ selling its ticket to $\mathcal{U}_2$, without any of the two users interacting with any third party. As $\mathcal{U}_1$ holds at least the same data as it did before interacting with $\mathcal{U}_2$, if it had probability p to execute the exchange protocol with $\mathcal{U}_2$, in general, the same success probability holds for a second execution of the transfer protocol with another user $\mathcal{U}_3$ with inputs distributed similarly to $\mathcal{U}_2$'s inputs. If we assume that p is non-negligible, then, $\mathcal{U}_1$ has probability p^2 to sell twice its ticket, which is not negligible and contradicts the security of the exchange regarding $\mathcal{U}_2$'s perspective. □

To ensure a secure transfer between the two users, we rely on a transfer authority $\mathcal{T}$. The considered communication setup is depicted in Fig. 8. In our case the ticket transfer can be viewed as a purchase of the same ticket for $\mathcal{U}_2$, followed by the refund of $\mathcal{U}_1$.

Other architectures based on a trusted third party could be considered. For example, the above analysis does not prevent only one of the two users $\mathcal{U}_1$ from contacting the transfer authority, while $\mathcal{U}_2$ contacts only $\mathcal{U}_1$. Indeed, the security analysis of our protocol would ensure that such a process can be practically achieved: messages replayed by $\mathcal{U}_1$ from $\mathcal{U}_2$ and then received by $\mathcal{T}$ would be well-formed, and our security proofs guarantee that $\mathcal{U}_2$ does not need to contact the transfer authority. security proofs guarantee that $\mathcal{U}_2$'s identity cannot be usurped by another entity in another execution.

C SECURITY PROOFS OF Applause

We present the security proofs for Applause, with respect to the security model presented in Section 3. Through this section, we operate proofs divided into sequences of games. We refer to this games as $\mathrm{G}^i_{\mathcal{A}}(1^\lambda)$ for a given PPT algorithm $\mathcal{A}$ and a security parameter 1^λ. Moreover, we characterise by W_i the winning event where a PPT algorithm $\mathcal{A}$ makes $\mathrm{G}^i_{\mathcal{A}}(1^\lambda)$ outputs 1. At first we present the security proofs of the version of Applause presented in Section 5, which relies on the *Random Oracle Model* (ROM). Then we present the security proofs for the version of Applause in the standard model. Recall that both versions of Applause differ only in few points, as explained in Section 6.

C.1 Security Proofs of Applause in Random Oracle Model

Proof of Unforgeability. Let $\mathcal{A}$ be a PPT adversary and assume for contradiction that the following probability $\Pr[\mathsf{Exp}^{\mathsf{UF}}_{\mathcal{A}}(1^\lambda) \rightarrow 1]$ is non-negligible. This proof relies on the ROM, meaning that $\mathcal{A}$ has access to an oracle Ohash replacing the function H and programmed by the challenger C, which on call on a message m, samples at random a value h representing the hash associated to m. This process is repeated if the value is already associated to another message m'. Then the sampled value h is returned to $\mathcal{A}$.

Game G^0. The initial game represents $\mathsf{Exp}^{\mathsf{UF}}_{\mathcal{A}}$ defined in Fig. 3, we observe that $\Pr[W_0] = \Pr[\mathsf{Exp}^{\mathsf{UF}}_{\mathcal{A}}(1^\lambda) \rightarrow 1]$.

Game G^1. In this game, we now reject tickets with valid signatures from $\mathcal{D}$ that where not produced by the challenger. To distinguish such signature from the originally produced ones, C keeps updated a set $\mathcal{S}_{\mathcal{D}} = \{m_i, \sigma_i\}$ containing all the signature produced by $\mathcal{D}$ during the protocol. A signature is deemed valid if the message-signature pair is contained in this set, otherwise invalid. This prevents $\mathcal{A}$ from forging a signature, which was previously possible with probability $\mathsf{Adv}^{\text{EUF-CMA}}_{\mathcal{A}}$. Therefore, we obtain the following bound $|\Pr[W_0] - \Pr[W_1]| \leq \mathsf{Adv}^{\text{EUF-CMA}}_{\mathcal{A}}(1^\lambda)$.

Game G^2. The same action as in G^1 is performed on the signature produced by $\mathcal{T}$. Hence $|\Pr[W_1] - \Pr[W_2]| \leq \mathsf{Adv}^{\text{EUF-CMA}}_{\mathcal{A}}(1^\lambda)$.

In G^2, the adversary $\mathcal{A}$ is not able to produce a ticket containing a forged signature from $\mathcal{D}$ or $\mathcal{T}$. A valid ticket is requires to contain a signature either by $\mathcal{D}$ or $\mathcal{T}$ as prescribed by the CheckTk algorithm executed in Validate, Refund or Transfer. $\mathcal{A}$ can still reused a signature produced by the Challenger. Under the ROM, this implies using the hash $c \leftarrow \mathsf{Ohash}(\mathsf{ide}, \mathsf{idp}, r_c)$ corresponding to a random value, already signed by C. Either $\mathsf{tk} \notin \mathsf{TK}$ and $\mathcal{A}$ looses the game, otherwise $\mathsf{tk} \in \mathsf{TK}$ and in such case, c appears in the state st of the Challenger C and the ticket will fail to verify in the last verification of the CheckTk algorithm. This can been seen by analyzing the inclusion of c in st through the processes. In both cases, a condition of CheckTk is not achieved and the protocol aborts for C. This shows that $\Pr[W_2] = 0$, and then, we obtain $\Pr[\mathsf{Exp}^{\mathsf{UF}}_{\mathcal{A}}(1^\lambda) \rightarrow 1] \leq 2 \cdot \mathsf{Adv}^{\text{EUF-CMA}}_{\mathcal{A}}(1^\lambda)$.

By hypothesis on the EUF-CMA property of the signature S, we know that $\mathsf{Adv}^{\text{EUF-CMA}}_{\mathcal{A}}$ is negligible, leading to a negligible quantity on the right hand side of the inequation, giving a direct contraction with the initial hypothesis. Hence, $\Pr[\mathsf{Exp}^{\mathsf{UF}}_{\mathcal{A}}(1^\lambda) \rightarrow 1]$ is negligible and Applause has Unforgeability. □

Proof of Pseudonymity. Assume a PPT algorithm $\mathcal{A}$, we show that it breaks at least one of the assumed properties of the used primitives, and that such an algorithm cannot exist under the given hypothesis. We start by considering an initial game G^0.

Game G^0. The initial game corresponds to experiment $\mathsf{Exp}^{\mathsf{PSE}}_{\mathcal{A}}$. Observe that $\Pr[W_0] = \Pr[\mathsf{Exp}^{\mathsf{PSE}}_{\mathcal{A}}(1^\lambda) \rightarrow 1]$.

Game G^1. We focus on the anonymous payment. At the beginning, the challenger generates a new key pair and uses this key $\mathsf{sk}^{\mathsf{P}}_{\mathsf{new}}$ instead of using $\mathsf{sk}^{\mathsf{Pay}}_{\mathcal{U}_b}$ in all payments of the experiment. By hypothesis, an adversary has a negligible advantage $\mathsf{Adv}^{\mathsf{PayPse}}_{\mathcal{A}}(1^\lambda)$ of distinguishing between which one among the two key pairs used in a payment. This property is directly transferred into our case and leaves the two key pairs of $\mathsf{sk}^{\mathsf{Pay}}_{\mathcal{U}_0}$ and $\mathsf{sk}^{\mathsf{Pay}}_{\mathcal{U}_1}$ unused. Hence, we obtain the following bound $|\Pr[W_0] - \Pr[W_1]| \leq \mathsf{Adv}^{\mathsf{PayPse}}_{\mathcal{A}}(1^\lambda)$.

Now, we notice that the user's key is not used in Purchase and Refund. Only Transfer and Validate involve the user's key, excluding the payment key. Hence, a call to oracles $\mathsf{OPurchase}_b$, $\mathsf{ORefund}_b$ does not reveal any information on the key pair $(\mathsf{sk}_{\mathcal{U}_b}, \mathsf{pk}_{\mathcal{U}_b})$ in use, ignoring the payment key that we already modified we apply the following:

Game G^2. Instead of randomizing the key pair in Validate, when $\mathcal{A}$ calls $\mathsf{OValidate}_b$, we generate a new signature and encryption key pairs. The resulting keys are unrelated to the key pair of the user. Provided with statistically randomizable key pairs, they where already unrelated after randomization, resulting in $\Pr[W_2] = \Pr[W_1]$.

Game G^3. On each call of $\mathcal{A}$ to $\mathsf{OTransfer}_b$, the same modification is put in place in Transfer. As for the previous modification, games are therefore statistically indistinguishable. This is again a bridging step with $\Pr[W_3] = \Pr[W_2]$.

In this game G^3, none of $(\mathsf{sk}_{\mathcal{U}_0}, \mathsf{pk}_{\mathcal{U}_0})$ or $(\mathsf{sk}_{\mathcal{U}_1}, \mathsf{pk}_{\mathcal{U}_1})$ are used by the challenger during the experiment. Hence, the adversary has no mean to recover b and then no advantage in breaking the pseudonymity. Hence, we obtain $\Pr[W_3] = \frac{1}{2}$. Summing up, $\mathcal{A}$ is a PPT algorithm with non-negligible advantage of winning against G^0 *i.e.,* a non-negligible advantage in breaking the anonymity of $\mathcal{U}_b$. Hence, we observe that $|\Pr[\mathsf{Exp}^{\mathsf{PSE}}_{\mathcal{A}}(1^\lambda) \rightarrow 1] - \frac{1}{2}| \leq \mathsf{Adv}^{\mathsf{PayPse}}_{\mathcal{A}}(1^\lambda)$.

By hypothesis on the anonymity property of considered anonymous payment, we know that $\mathsf{Adv}^{\mathsf{Pay}}_{\mathcal{A}}$ is negligible, implying that no such $\mathcal{A}$ could exist. Therefore, pseudonymity holds for our ETS protocol Applause. □

Proof of Privacy. We prove secrecy of r_c through this proof. Assume $\mathcal{A}$ has a PPT algorithm. If an adversary $\mathcal{A}$ is unable to recover r_c, then it is unable to win against $\mathsf{Exp}^{\mathsf{PRIV}}_{\mathcal{A}}(1^\lambda)$, or at least with a negligible probability. Notice that $\mathcal{U}$, the user simulated by C which has generated tk, cannot be corrupted, otherwise $\mathcal{A}$ would fail to the experiment under $\mathsf{corr}_1 = 0$. The following modification applied to the oracles are assumed to append only when the ticket tk is involved during the execution of the oracle.

Game G^0. The initial game corresponds to the experiment $\mathsf{Exp}^{\mathsf{PRIV}}_{\mathcal{A}}$, hence $\Pr[W_0] = \Pr[\mathsf{Exp}^{\mathsf{PRIV}}_{\mathcal{A}}(1^\lambda) \rightarrow 1]$.

Game G^1. First, we modify the Purchase algorithm happening in the experiment. When tk is purchased, the challenger sends a random element as the first message from $\mathcal{U}_1$, thus replacing $\mathsf{Enc}_{\mathsf{pk}^{\mathsf{E}}_{\mathcal{D}}}(\mathsf{ide}, \mathsf{idp}, r_c, c)$ by a random sampled uniformly from $[\mathsf{Enc}_{\mathsf{pk}^{\mathsf{E}}_{\mathcal{D}}}]$. $\mathcal{D}$ being also simulated by C it still has access to ide, idp, r_c, c and then is able to continue the Purchase protocol.

We observe that the difference between G^0 and G^1 occurs only in the encryption of this message. Under the IND-CPA hypothesis, a distinguisher would have negligible chances to distinguish between these two experiments. Hence $|\Pr[W_0]-\Pr[W_1]| \leq \mathsf{Adv}^{\text{IND-CPA}}_{\mathcal{A},\mathsf{E}}(1^\lambda)$.

Game G^2. The second game replaces the encryption of tk under the key $\mathsf{pk}^{\mathsf{E}}_{\mathcal{D}}$ during a call to ORefund by a random element sampled uniformly from $[\mathsf{Enc}_{\mathsf{pk}^{\mathsf{E}}_{\mathcal{D}}}]$. This is the same argument as before, we directly conclude to $|\Pr[W_1] - \Pr[W_2]| \leq \mathsf{Adv}^{\text{IND-CPA}}_{\mathcal{A},\mathsf{E}}(1^\lambda)$.

Game G^3. The third game replaces the encryption of tk under the key $\mathsf{pk}^{\mathsf{E}}_{\mathcal{T}}$ during a call to OTransfer by a random element sampled uniformly from $[\mathsf{Enc}_{\mathsf{pk}^{\mathsf{E}}_{\mathcal{D}}}]$. As seen in G^1 and in G^2, we have $|\Pr[W_2] - \Pr[W_3]| \leq \mathsf{Adv}^{\text{IND-CPA}}_{\mathcal{A},\mathsf{E}}(1^\lambda)$.

Game G^4. In the last game, concluding this proof, we replace the encryption of $\mathsf{pk}'_{\mathcal{U}}$, tk, $cert'_{\mathcal{U}}$ under $\mathsf{pk}^{\mathsf{E}}_{\mathcal{V}}$ by a random element sampled uniformly from $[\mathsf{Enc}_{\mathsf{pk}^{\mathsf{E}}_{\mathcal{D}}}]$. Still using the indistinguishability argument we have $|\Pr[W_3] - \Pr[W_4]| \leq \mathsf{Adv}^{\text{IND-CPA}}_{\mathcal{A},\mathsf{E}}(1^\lambda)$.

Since $\mathcal{A}$ has no more advantage, and since r_c is randomly picked at uniformly in $\{0,1\}^\lambda$, $\mathcal{A}$ has probability $2^{-\lambda}$ to guess correctly, hence $\Pr[W_4] = 2^{-\lambda}$. Finally, we observe that: $\mathsf{Adv}^{\mathsf{PRIV}}_{\mathcal{A}}(1^\lambda) = \Pr[\mathsf{Exp}^{\mathsf{PRIV}}_{\mathcal{A}}(1^\lambda) \rightarrow 1] \leq 4 \cdot \mathsf{Adv}^{\text{IND-CPA}}_{\mathcal{A},\mathsf{E}} + 2^{-\lambda}$. □

Proof of No-Double-Spending. Let $\mathcal{A}$ be a PPT adversary. We show that the probability $\Pr[\mathsf{Exp}^{\mathsf{DS}}_{\mathcal{A}}(1^\lambda) \rightarrow 1]$ is negligible. In the game $\mathsf{Exp}^{\mathsf{DS}}_{\mathcal{A}}$, $\mathcal{A}$ is free to take advantage of any sequence of execution it has chosen. We show that under any chosen sequence, it is impossible for $\mathcal{A}$ to success to the same algorithm twice with the same ticket tk. The proof relies on the consistency of the state *st*, updated by the challenger C during the protocol. We argue that this state prevent any double spending, indeed acting as a blacklist containing every ticket tk used in any of the protocols. We proceed by analyzing the update of the state through the algorithms: In the Refund protocol, on the reception of tk, $\mathcal{D}$ executes the CheckTk algorithm which checks that σ_c is signed either by $\mathcal{D}$ or by $\mathcal{T}$, but also that c is not contained in *st*. If not, c is inserted in *st*. Suppose now that $\mathcal{A}$ replays the refund protocol again, then the signature still verifies, but since tk is now contained in *st*, then the refund fails. The ticket transfer holds on the same principle. When transferring tk, algorithm CheckTk is executed, ensuring that tk is not already contained in *st*. Exactly as the previous protocols, on the reception of ticket tk, the validation executes the CheckTk checking that tk is not contained in *st*. When the algorithm successfully ends, then tk is required to be contained in *st* to notify that the ticket is now considered as invalid.

All algorithms taking a ticket tk as an input ends with a state *st* containing tk. Since the state *st* is fully controlled by C and cannot be modified by $\mathcal{A}$, we can directly conclude that for every PPT adversary, $\Pr[\mathsf{Exp}^{\mathsf{DS}}_{\mathcal{A}}(1^\lambda) \rightarrow 1]$ is zero, meaning that the double-spending, in any sequence, is prevented. □

Recall that in our construction, a ticket tk is defined by a the place (ide, idp), the random r_c and the signature σ_c. From an information theory perspective, tk is completely unrelated from any user $\mathcal{U}$. Then, our proof simply consists to show that at any moment, the identity of $\mathcal{U}$ (*e.g.,*the public key $\mathsf{pk}_{\mathcal{U}}$, a signature using $\mathsf{sk}_{\mathcal{U}}$) is used during the communication.

Proof of Unlinkability. Our proof is designed as a sequence of five games G^i for i ranging from 0 to 4, where the initial game G^0 corresponds the game $\mathsf{Exp}^{\mathsf{UNL}}_{\mathcal{A}}$ for a bit b, and our last game G^4 corresponds to the game $\mathsf{Exp}^{\mathsf{UNL}}_{\mathcal{A}}$ with C using $1-b$. We show that both game are indistinguishable through a sequence of negligible modifications. We only need to apply the modifications where the third ticket tk_3 (purchased by $\mathcal{U}_b$) is involved. Let $\mathcal{A}$ be a PPT adversary, we show that it has negligible probability to succeeds to $\mathsf{Exp}^{\mathsf{UNL}}_{\mathcal{A}}$ for a probability non-negligible close to 1/2. For more clarity, let $\mathsf{Exp}^{\mathsf{UNL}}_{\mathcal{A},b}$ be the game $\mathsf{Exp}^{\mathsf{UNL}}_{\mathcal{A}}$ except that we set the bit b as a parameter instead of being chosen randomly by the challenger.

Game G^0. The initial game corresponds to the experiment $\mathsf{Exp}^{\mathsf{UNL}}_{\mathcal{A},b}$ with the bit b (still chosen randomly outside of the game) is provided as a parameter. Hence, for every adversary $\mathcal{A}$, we have $\Pr[W_0] = \Pr[\mathsf{Exp}^{\mathsf{UNL}}_{\mathcal{A},b}(1^\lambda) \rightarrow 1]$.

Game G^1. We focus on the Validate protocol, where we have $\mathsf{pk}'_{\mathcal{U}_b}$ and tk sent from $\mathcal{U}_b$ to $\mathcal{V}$. Since tk is not related to $\mathcal{U}_b$, then using the statistical indistinguishability of the randomised keys, we replace $\mathsf{pk}'_{\mathcal{U}_b}$ by $\mathsf{pk}'_{\mathcal{U}_{1-b}}$. This modification implies that in the subsequent modification, we have $\mathsf{Enc}_{\mathsf{pk}^{\mathsf{E}'}_{\mathcal{U}_{1-b}}}(s)$ which does not help $\mathcal{A}$ using the statistical indistinguishability of the randomised keys. Therefore, we have $\Pr[W_0] = \Pr[W_1]$.

Game G^2. We focus on Refund, which contains only a payment interaction between $\mathcal{U}_b$ and $\mathcal{D}$. Again, we replace the secret payment key $\mathsf{sk}^{\mathsf{P}}_b$ of $\mathcal{U}_b$ with the secret payment key of $\mathsf{sk}^{\mathsf{P}}_{1-b}$. The difference from the point of view of $\mathcal{A}$ relies on the capacity to link a payment either to $\mathcal{U}_b$ or to $\mathcal{U}_{1-b}$, which we quantify as the advantage $\mathsf{Adv}^{\mathsf{PayUnl}}_{\mathcal{A}}$. Then, we have $|\Pr[W_1] - \Pr[W_2]| \leq \mathsf{Adv}^{\mathsf{PayUnl}}_{\mathcal{A}}$.

Game G^3. We focus on Transfer, where $\mathcal{A}$ is expected to have as an input the randomised public key $\mathsf{pk}'_{\mathcal{U}_b}$. Using the same strategy of the game G^1, we replace $\mathsf{pk}'_{\mathcal{U}_b}$ with $\mathsf{pk}'_{\mathcal{U}_{1-b}}$. This modification of the input implies that $\sigma_{T,1}$ is now produces with the signature key $\mathsf{sk}^{\mathsf{S}'}_{\mathcal{U}_{1-b}}$, which is verifiable using $\mathsf{pk}^{\mathsf{S}'}_{\mathcal{U}_{1-b}}$. Again, using the statistical indistinguishability of the randomised keys, $\mathcal{A}$ cannot notice the difference. However, Latter in the protocol, we have a call to the Refund protocol as a subroutine. As shown, Refund involves an anonymous payment. Since we replace the key $\mathcal{U}_b$ with key of $\mathcal{U}_{1-b}$, the secret payment key is replaced as well, which can be distinguished by $\mathcal{A}$ with the advantage of breaking the unlinkability of the anonymous payment. Hence $|\Pr[W_2] - \Pr[W_3]| \leq \mathsf{Adv}^{\mathsf{PayUnl}}_{\mathcal{A}}$.

Game G^4. In the last game, we focus on Purchase, on which the two first interactions does not involve the public key $\mathsf{pk}_{\mathcal{U}}$. The third interaction, however, is the anonymous payment. As in game G^2 and G^3, we replace $\mathsf{sk}^{\mathsf{P}}_b$ by $\mathsf{sk}^{\mathsf{P}}_{1-b}$. The probability for $\mathcal{A}$ to distinguish between G^3 and G^4 is based on the advantage to break the unlinkability of the anonymous payment. Hence, we have $|\Pr[W_3] - \Pr[W_4]| \leq \mathsf{Adv}^{\mathsf{PayUnl}}_{\mathcal{A}}$.

Remark that in G^4, we have replaced the key pair $(sk_{\mathcal{U}_b}, pk_{\mathcal{U}_b})$ with $(sk_{\mathcal{U}_{1-b}}, pk_{\mathcal{U}_{1-b}})$, when tk_3 is used. This demonstrate that $\mathcal{A}$ is unable to distinguish the experiment $\mathsf{Exp}^{\mathsf{UNL}}_{\mathcal{A},b'}$ when b' equals b or $1-b$. We can conclude that

$$|\Pr[\mathsf{Exp}^{\mathsf{UNL}}_{\mathcal{A},b}(1^\lambda) \to 1] - \Pr[\mathsf{Exp}^{\mathsf{UNL}}_{\mathcal{A},\bar{b}}(1^\lambda) \to 1]| \leq 3 \cdot \mathsf{Adv}^{\mathsf{PayUnl}}_{\mathcal{A}}$$

$$|\Pr[\mathsf{Exp}^{\mathsf{UNL}}_{\mathcal{A}}(1^\lambda) \to 1] - \frac{1}{2}| \leq \frac{3}{2} \cdot \mathsf{Adv}^{\mathsf{PayUnl}}_{\mathcal{A}}$$

□

C.2 Security Proofs of Applause in the Standard Model

We present the security proofs of Applause in the standard model, for which we replace the hash $c \leftarrow H(\mathsf{ide}, \mathsf{idp}, r_c)$ for a place $(\mathsf{ide}, \mathsf{idp})$ and a random r_c, by a commitment $c \leftarrow h^{r_c} g_1^{\mathsf{ide}} g_2^{\mathsf{idp}}$ where h, g_1 and g_2 are three generators of a group $\mathbb{G}$, assumed secure under the Discrete Logarithm (DL) problem.

Pseudonimity, *Double-spending* and *Unlinkability* have already been shown under standard assumption as the previous proof does not involve any argument based on the random oracle. We need no further consideration to guarantee these properties for the standard model version of Applause, the proof being the same as before. We only have to modify the proofs for *Unforgeability* and *Privacy*.

Proof of Unforgeability. We start with the same sequence of games as for the proof of Theorem 5.2. We consider G^2, where the challenger keeps record of the signatures produced by $\mathcal{D}$ and $\mathcal{T}$, and refuses any forged valid signature under $pk^{\mathsf{S}}_{\mathcal{D}}$ or $pk^{\mathsf{S}}_{\mathcal{T}}$.

During the execution of the algorithm Alg chosen by $\mathcal{A}$ the CheckTk is always executed, hence $\mathcal{A}$ must have transmitted a signature σ_c for c taken from tk returned by $\mathcal{A}$. We recall that in the current case $c = h^{r_c} g_1^{\mathsf{ide}} g_2^{\mathsf{idp}}$. As a forged signature would be refused by C, $\mathcal{A}$ needs to output a signature generated for another ticket. It has been produced for this ticket, then $\mathsf{tk} \in \mathsf{TK}_i$ holds true and the game is lost. Hence $\mathcal{A}$ must used a previously made signature and need to find a triple $r_c, \mathsf{ide}, \mathsf{idp}$ such that $c = h^{r_c} g_1^{\mathsf{ide}} g_2^{\mathsf{idp}}$ holds true for one of the signature produced for another ticket tk'. Equivalently, $c = h^{r_c} g_1^{\mathsf{ide}} g_2^{\mathsf{idp}} = h^{r'_c} g_1^{\mathsf{ide}'} g_2^{\mathsf{idp}'}$. Based on such an answer, we can recover the discrete logarithm of one of the generator based on the two others. For example, we can obtain $h = g_1^{\frac{\mathsf{ide}-\mathsf{ide}'}{r_c - r'_c}} g_2^{\frac{\mathsf{idp}-\mathsf{idp}'}{r_c - r'_c}}$. Now assuming a challenger for the discrete logarithm problem sending g, g^x as a challenge, we can sample a random $s \xleftarrow{\$} \mathbb{Z}_p^*$ and set $h = g^x$, $g_1 = g$ and $g_2 = g^s$, all three are known to be generators of the prime order group if $g^x \neq 1$. Based on a correct answer from an adversary winning against game G^2, we recover $g^x = g_1^{\frac{\mathsf{ide}-\mathsf{ide}'}{r_c - r'_c}} g_2^{\frac{\mathsf{idp}-\mathsf{idp}'}{r_c - r'_c}} = g^{\frac{\mathsf{ide}-\mathsf{ide}'}{r_c - r'_c} + s \cdot \frac{\mathsf{idp}-\mathsf{idp}'}{r_c - r'_c}}$. Hence, recovering the discrete logarithm of g^x. This allows us to conclude that $\Pr[W_2] \leq \mathsf{Adv}^{\mathsf{DL}}_{\mathcal{A}}(1^\lambda)$. And based on the previous reduction, we observe that: $\Pr[\mathsf{Exp}^{\mathsf{UF}}_{\mathcal{A}}(1^\lambda) \to 1] \leq 2 \cdot \mathsf{Adv}^{\text{EUF-CMA}}_{\mathcal{A}}(1^\lambda) + \mathsf{Adv}^{\mathsf{DL}}_{\mathcal{A}}(1^\lambda)$. □

Proof of Privacy. The same sequence of game as in the proof of Theorem 5.4 can be applied. We follow up on the previous reduction. In G^4, among other things, $\mathcal{A}$ has to return the value r_c. This element is only involved in the computation of c, as in this version of the protocol $c = h^{r_c} g_1^{\mathsf{ide}} g_2^{\mathsf{idp}}$. Using the state leakage, $\mathcal{A}$ can have access to c during the game, similarly, ide and idp are both accessible to $\mathcal{A}$. In order to recover r_c from the latest, $\mathcal{A}$ is expected to compute a discrete logarithm on $c \cdot g_1^{\mathsf{ide}} g_2^{\mathsf{idp}} = h^{r_c}$. Hence, given a challenger for the discrete logarithm problem sending $X = g^x$, during the purchase of $(\mathsf{ide}, \mathsf{idp})$, the challenger C replace h^{r_c} by X. Receiving the answer from an adversary $\mathcal{A}$, we can return the value returned for r_c to the challenger for the discrete logarithm problem. If $\mathcal{A}$ has non-negligible chances to break the privacy of the ticket, it would either break the discrete logarithm problem or the IND-CPA property of the encryption. A random guess can still be possible, leading to the following advantage: $\Pr[\mathsf{Exp}^{\mathsf{PRIV}}_{\mathcal{A}}(1^\lambda) \to 1] \leq \frac{1}{q} + 4 \cdot \mathsf{Adv}^{\text{IND-CPA}}_{\mathcal{A}}(1^\lambda) + \mathsf{Adv}^{\mathsf{DL}}_{\mathcal{A}}(1^\lambda)$. □

D SECURITY PROOFS OF Spotlight

The unforgeability of the ticket, its privacy and prevention of double spending property remains unaffected by the additional authentication information on this new version of the protocol. The proposed changes in Section 7, at a high level, adds a certificate that must to be sent by a user at the beginning of the protocols' execution. Since we only add authentication material to the protocol, it is fairly straightforward to see that the same proof strategies for unforgeability, privacy and no-double spending are valid in this latest version of the protocol. Note that the Judge $\mathcal{J}$ does not take part to the game in this case. The challenger is simulating it instead.

Proof of Pseudonymity. Based on the above described modifications in the game (the judge is added and controlled by the challenger), we follow the same sequence of game as in proof of unforgeability. Let us denote it as Game G^1.

We have assumed statistical randomization of the certificates of $\mathcal{U}_0$ with key pair $(sk_{\mathcal{U}_0}, pk_{\mathcal{U}_0})$ and $\mathcal{U}_1$ with key pair $(sk_{\mathcal{U}_1}, pk_{\mathcal{U}_1})$ through the RandID algorithm. Hence, there is no link between the keys and the identity. Now, let us assume a distinguisher between G^1 with bit b and $\mathsf{Exp}^{\mathsf{PSE}}_{\mathcal{A}}$ with bit $1-b$. Assuming existence of such a distinguisher, we can directly distinguish two randomised certificates. Hence, under the above hypothesis, we can conclude that pseudonymity holds for the auditable version of Applause. We precise an upper bound: $|\Pr[\mathsf{Exp}^{\mathsf{PSE}}_{\mathcal{A}}(1^\lambda) \to 1] - \frac{1}{2}| \leq \mathsf{Adv}^{\mathsf{PayPse}}_{\mathcal{A}}(1^\lambda)$. □

Proof of Unlinkability. The challenger executes the actions of the judge, hence, the adversary cannot recover the user's identity based on the judge's keys $(sk_{\mathcal{J}}, pk_{\mathcal{J}})$. Only an additional certificate is shared with the adversary during Purchase, Refund or Transfer. Even so, this is linked to the user's public key, that we have assumed statistically indistinguishable randomization, leading to impossibility of the recovery of the original identity of the user. Then, $|\Pr[\mathsf{Exp}^{\mathsf{UNL}}_{\mathcal{A}}(1^\lambda) \to 1] - \frac{1}{2}| \leq 3 \cdot \mathsf{Adv}^{\mathsf{PayUnl}}_{\mathcal{A}}(1^\lambda)$. □

For the auditable version of Applause in the standard model, the same changes presented in Section C.2 can be applied to the above proof to show its security.