Transferable, Auditable and Anonymous Ticketing Protocol (Long Version)

Pascal Lafourcade$^{2[0000-0002-4459-511X]}$, Dhekra Mahmoud$^{2[0009-0002-0555-0581]}$, Gael Marcadet$^{2[0000-0003-1194-1343]}$, and Charles Olivier-Anclin$^{1,2[0000-0002-9365-3259]}$

^{1} be ys Pay
^{2} Université Clermont-Auvergne, CNRS, Clermont-Auvergne-INP, LIMOS, Clermont-Ferrand, France

Abstract. Digital ticketing systems usually provide ticket purchase, refund, validation, and optionally the anonymity of users. However, it is interesting for users to transfer their tickets, as currently done with physical tickets. We propose Applause, the first ticketing system allowing purchase, refund, validation, and transfer of tickets, while ensuring anonymity of users. To study its security we formalise the security of transferable E-Ticket scheme in the game-based paradigm and prove the security of Applause computationally in the standard model and symbolically using the protocol verification tool ProVerif. Applause relies on standard cryptographic primitives, rendering our construction efficient and scalable as shown by a proof-of-concept.
Depending on the context, anonymity of E-Tickets schemes may have to be mitigated. We extend Applause to obtain Spotlight, ensuring auditability of the identities by adding a third-party called a judge and prove the security of our auditable protocol.

Keywords: Ticketing System, Protocol, Auditability, Anonymity, Transfer

1 Introduction

Electronic tickets (*E-Ticket*) have become the standard. The incentives behind this digitalisation of the ticket industry are practical and economical. A user can easily purchase tickets from anywhere, leading to more sales. The practical aspect of ticket digitalisation comes with major drawbacks impacting the second-hand market and the privacy of the costumers. In a world where privacy is a central concern, preserving the multiple facets of a paper ticket, such as the right to trade them and ensuring full confidence in their validity must be maintained. But the original property of non-replicable paper tickets is often lost with the development of E-Ticket. Thus, a formal security model for electronic ticketing is needed to be unsured of their security and designing a system merging the best properties of both worlds is necessary.

The research has been focused on multiple cryptographic technologies with properties analogous to their physical counterpart. Topics such as *transferable e-cash* [4], *transferable e-coupon* [19] or *n-times anonymous authentication* [28] seems closely related to our problematic. Despite being deeply studied in the literature, their design makes them incompatible for being used as a ticketing system. *Transferable e-cash* [3,4] ensures the same guarantees as paper cash. We stress that, even if the two subjects seem to be closely related, the properties needed by digital ticketing system are fundamentally different to what intended in the design of an e-cash protocol. In many E-cash systems [3,4], double-spending (spending the same coin twice) is prevented by a central bank maintaining a list of spent coins. On receiving a coin the bank checks if the coin belongs to the list of spent coins. If so, the same coin has been spent twice, and the bank can now sue the client for its fraud. The bank can directly punish the fraudster and compensate a scammed client. For physical ticket system, it can only be detected at the time it tries to access to the event, which is too late, and it might not even be possible to trace the cheater. The same applies for *transferable e-coupon* [19], while *n-times anonymous authentication* [28] cannot be securely transferred. In contrary, with e-tickets, it is imperative to guarantee the validity of a ticket at any time to avoid double-spent tickets. This main issue makes every e-cash system preventing the double-spending problem at coin deposit irrelevant for our problem. Finally, in e-cash [4], a coin is validated by the bank during the deposit. In the case of e-ticketing, a ticket can be validated by one of the many terminals at the entrance to the event. All these differences justify the need for a specific approach to e-ticketing.

The e-ticketing has been more studied by the industry [2,25] for obvious economic interests. Majority of existing ticketing systems developed by the industry are focused on the “standard” functionality of e-ticket: providing a ticket and validating it. These systems are simply not designed for secure ticket transfer. Hence, an honest client has a high chance to buy a duplicated ticket from a malicious user, which is badly common. More sophisticated systems, trying to address this issue, provides authentication using a distributed architecture such as Blockchain [2,25]. Even if the ticket validity is now ensured during a transfer (by checking if the ticket is valid in the Blockchain), we move away from current e-ticketing system organization where the tickets are stored in a database. These methods, even if they could ensure all the required security, are in opposition with the centralised event organization and lead to complex and computationally demanding procedures. As an alternative our proposed protocol is efficient, and require at most 100 milliseconds for a full execution (see Appendix A), and hence can be executed from any device. The anonymity of users in e-ticketing systems is a property that has not been considered in the latest protocols [2,13,22,25]. Nevertheless, anonymity is guaranteed by physical tickets, and it remains crucial to prevent the event organizers to collect names of ticket purchasers unless necessary. It is well known that blockchain can serve this purpose, but for the aforementioned argumentation, we rule them out of the scope of our investigation.

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

Our *provably secure e-ticketing* protocol is called Applause. It allows users to purchase, get a refund, and to validate tickets, but also transfer their tickets to another user. During all these interactions, depectied in Fig. 1, users' anonymity from the event organiser point-of-view is ensured. We propose the first protocol addressing all the above mentioned properties and prove its security based on computational proofs under the random oracle model, but also using formal methods with ProVerif [5]. Payments can occur during at least three steps of our process: the purchase, the refund, and the transfer of tickets. Traditional payment does not ensure anonymity of buyer, hence, using it in our construction would trivially break the anonymity of users. For instance, the most standard payment protocol EMV [9] does not preserve any kind of privacy. In our protocol, the payment is modelised as a cryptographic building block, requiring *anonymity*, a mandatory assumption only required to ensure the full-anonymity. Alternative payment protocols ensuring anonymity of users exist [20,23] and can be used in our protocol. To reveal the identity of users, and hence reaching auditability, we present Spotlight, an extended version of Applause, where the identity of users is revealed by a third-party called the judge, using randomisable certificates which are certificates that can be randomised along with the certified keys.

Related Work. Most of the production-ready deployed systems guarantee the "standard" functionalities: ticket payment and delivery. More than offering the standard functionalities, our system allows users to transfer a ticket to another user, while preserving anonymity. All existing systems allowing additional functionalities that we identify rely on blockchains [2,25]. Blockchain-based ticketing systems can easily achieve transfer of a ticket by checking if a ticket is still valid on the blockchain and can achieve anonymity by the design of the blockchain. However, it requires the maintenance of a distributed ledger and numerous signatures, meaning that many servers are involved to ensure the integrity of the blockchain. Low-consumption and optimised computation has been a keystone of cryptography. It is known that blockchains are outperformed by more classical and centralised design. Our system is centralised to be efficient and to match the existing topology of event organisation.

Previous works, initiated by [17] in 2001 and followed up by several papers, essentially evaluating either the practicality [15], the interface design [29] or the security [27] of existing systems in the public transport. Subsequent works have proposed for new ticketing system. They can be divided into three categories.

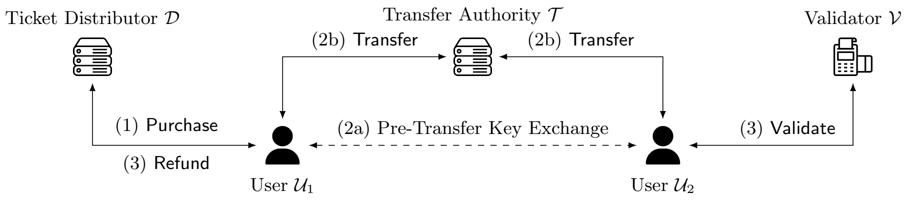


Fig. 1: Representation of the ETS model.

The first category focuses on the design of ticketing system [13,22], ensuring the validity of a ticket, without considering privacy. The second category proposes to provide privacy of user, but does not allow transfer nor auditability. In [16], they study the possibility of ticketing system with privacy of users, including respectively the billing step. The authors of [11,12] have studied the possibility to use RFID and NFC while ensuring privacy of users, using unlinkable certified tokens [21] for public transport ticketing or using anonymous credentials [14] for general purposes tickets. Moreover, construction of [14] requires 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 [14], we encompass the payment process in the protocol. The third line of research proposes ticketing systems based on a distributed ledger, less efficient compared to the centralised setting, due to the number of involved servers and communication time. The first paper to propose such a system is presented in [18], both based on blockchain. Recently, a new ticketing system has been proposed in [30] based on NFT. Our work moves away from the distributed approach for the practical and efficiency reasons depicted above.

Outline. The general ideas and the setting of our ticketing service are provided in Section 2. In section 3, we introduce a new model for *E-Ticket schemes*. In Section 4, we present the cryptographic tools used in Section 5 to describe Applause. The security analysis of our protocol is given in Section 7. We describe Spotlight, the auditable version of Applause, in Section 8 and we conclude in Section 9. Full proofs of the protocol without auditability are given in Appendix E. Then a full description of the protocol with auditability is given in Appendix D and followed by the associated proofs in Appendix E.

2 Technical Overview

Entities Organisation. When a *User* $\mathcal{U}$ wants to purchase a ticket for an event, the user contacts a *Ticket Distributor* $\mathcal{D}$, which issues a tickets 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. Miscellaneous reasons can lead to $\mathcal{U}_1$ not to be able to benefit from its purchase. Our protocol

Purchase$\langle \mathcal{U}, \mathcal{D} \rangle$	Refund$\langle \mathcal{U}, \mathcal{D} \rangle$	Transfer$\langle \mathcal{U}_1, \mathcal{T}, \mathcal{U}_2 \rangle$	Validate$\langle \mathcal{U}, \mathcal{V} \rangle$
Tickets Secrecy	Tickets Secrecy	Tickets Secrecy	Tickets Secrecy
Ano. $\mathcal{U} \rightarrow \mathcal{D}$	Ano. $\mathcal{U} \rightarrow \mathcal{D}$	Ano. $\mathcal{U}_i \rightarrow \mathcal{T}$, $\mathcal{U}_1 \leftrightarrow \mathcal{U}_2$	Ano. $\mathcal{U} \rightarrow \mathcal{V}$
Auth. $\mathcal{U} \leftrightarrow \mathcal{D}$	Auth. $\mathcal{U} \leftrightarrow \mathcal{D}$	Auth. $\mathcal{U}_i \leftrightarrow \mathcal{T}$,$\mathcal{U}_1 \leftrightarrow \mathcal{U}_2$	Auth. $\mathcal{U} \leftrightarrow \mathcal{V}$
Unlinkability	Unlinkability	Unlinkability	Unlinkability
	Unforgeability	Unforgeability	Unforgeability
	No-Double-Refund	No-Double-Transfer	No-Double-Validation

Fig. 2: Security properties, where *Ano.* = *Anonymity* and *Auth.* = *Authentication.*

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 $\mathcal{U}_1$ in interaction with 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 case consists of the same 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 guarantor 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. The Transfer Authority $\mathcal{T}$ has been mod[2]eled as an independent entity but can be combined with the Ticket Distributor into a single entity without any loss of security. To attend for an event, $\mathcal{U}$ has to validate a ticket against a validator $\mathcal{V}$.

To ensure a secure transfer between the two users, we rely on a Transfer Authority $\mathcal{T}$. 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 $\mathcal{U}_1$. To make possible the designation of the ticket receiver, we assume a communication between $\mathcal{U}_1$ and $\mathcal{U}_2$ before the transfer of the ticket, allowing them to exchange keys.

Anonymity and Auditability. Anonymity of users is ensured during every interaction with the system (*i.e.,* the Ticket Distributor $\mathcal{D}$, the ticket Transfer Authority $\mathcal{T}$ and the Validator $\mathcal{V}$). This is ensured during all the process as no authentication is required during the majority of our protocol or only under a randomised identity *i.e.,* using randomised keys. If desired, our protocol Applause can be turned into an auditable version called Spotlight, where an external authority refers as the *judge* and denoted $\mathcal{J}$, can recover the identity of $\mathcal{U}$. In a nutshell, $\mathcal{U}$ provides a certificate attesting the validity of its randomised key. Given such a certificate, kept on a record by the authorities, $\mathcal{J}$ retrieves the original key, hence the identity of the user $\mathcal{U}$. The anonymity remains against $\mathcal{D}$, $\mathcal{T}$ or $\mathcal{V}$ as the certificate is randomised with the keys.

Validation Setting. During large scale events, the network is often overloaded due to the number of attendees. Moreover, not one but many terminals are validating the tickets simultaneously. Requiring large communications coming to and from the validation terminal for each validation might be difficult, limiting the transmitted data is thus required. One another hand, one would like to assume the Validator to be 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. We rely on a central authority instead of using a distributed ledger to agree on the valid ticket at time t. In our case only a single group element needs to be transmitted between a terminal and the central server for each approved verification.

Our validation protocol has to ensure anonymity of the users, but the Validator requires a mean to ensures that the system is not fooled by an adversary relaying all communications of a legitimate user. We have chosen to rely on a physical channel during the last step of the protocol to prevent a block-and-send attack of the ticket by an adversary. $\mathcal{U}$ receives a token which is compared through this channel with the token own and sent by $\mathcal{V}$. This makes the authentication provided by the protocol holds under the secrecy of the token encrypted under $\mathcal{U}$'s randomised key.

3 E-Ticket Model

Most ticketing systems are designed to allow the sale of a seat at an event, where each seat is associated to a metadata such as the seat number. This can also be more general, and we consider a scenario where tickets are characterized 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 can be through as event and seat number, and are assumed publicly known. In Fig. 2, we sum up security properties for each protocol involved in our model. Below we define our protocol ETS based on a security parameter 1^λ. Through this paper all algorithms runs in probabilistic polynomial-time (PPT).

Definition 1 (ETS). *An* ETS *scheme* $\Pi = (\mathsf{DKeyGen}, \mathsf{TKeyGen}, \mathsf{VKeyGen}, \mathsf{UKeyGen}, \mathsf{Purchase}, \mathsf{Refund}, \mathsf{Transfer}, \mathsf{Validate})$ *is a tuple of PPT algorithms, where:*

$\underline{\mathsf{DKeyGen}/\mathsf{TKeyGen}/\mathsf{VKeyGen}/\mathsf{UKeyGen}(1^\lambda) \rightarrow (\mathsf{sk}, \mathsf{pk})}$ *: Given the unary representation of the security parameter* 1^λ*, outputs a key pair.*

$\underline{\mathsf{Purchase}\langle \mathcal{U}(\mathsf{sk}_{\mathcal{U}}, (\mathsf{ide}, \mathsf{idp})), \mathcal{D}(\mathsf{sk}_{\mathcal{D}}, \mathrm{st})\rangle \rightarrow \mathcal{U}(\mathsf{tk}), \mathcal{D}(b, (\mathsf{ide}, \mathsf{idp}), \mathrm{st})}$ *: User* $\mathcal{U}$ *purchases the ticket identified by the pair* $(\mathsf{ide}, \mathsf{idp})$*, to the Ticket Distributor* $\mathcal{D}$*. As a result, the user obtains the ticket* tk*, and* $\mathcal{D}$ *updates* st *and returns a success bit* b *and* $(\mathsf{ide}, \mathsf{idp})$*.*

$\underline{\mathsf{Refund}\langle \mathcal{U}(\mathsf{sk}_{\mathcal{U}}, \mathsf{tk}), \mathcal{D}(\mathsf{sk}_{\mathcal{D}}, \mathrm{st})\rangle \rightarrow \mathcal{U}(b), \mathcal{D}(b, \mathsf{tk}, \mathrm{st})}$ *: 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*.*

$\underline{\mathsf{Transfer}\langle \mathcal{U}_1(\mathsf{sk}_{\mathcal{U}_1}, \mathsf{pk}_{\mathcal{U}_2}, \mathsf{tk}_1), \mathcal{T}(\mathsf{sk}_{\mathcal{T}}, \mathrm{st}), \mathcal{U}_2(\mathsf{sk}_{\mathcal{U}_2}, \mathsf{pk}_{\mathcal{U}_1}, (\mathsf{ide}, \mathsf{idp}))\rangle \rightarrow \mathcal{U}_1(b),}$
$\underline{\mathcal{T}(b, (\mathsf{ide}, \mathsf{idp}), \mathsf{tk}_1, \mathrm{st}), \mathcal{U}_2(\mathsf{tk}_2)}$*: 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* $(\mathsf{ide}, \mathsf{idp})$ *and* tk_1*.* $\mathcal{U}_2$ *returns a ticket* tk_2 *for the identifier* $(\mathsf{ide}, \mathsf{idp})$*.*

$\underline{\mathsf{Validate}\langle\mathcal{U}(\mathsf{sk}_{\mathcal{U}},\mathsf{tk}),\mathcal{V}(\mathsf{sk}_{\mathcal{V}},\mathrm{st})\rangle \to \mathcal{U}(b),\mathcal{V}(b,\mathsf{tk},\mathrm{st})}$: *$\mathcal{U}$ 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 tickets 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 a *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 local database.

Security Model. The security model of an ETS scheme must represent the realistic model expected from the real world behaviors of a ticket service: one holding a ticket must be able to attempt the event or get a refund, this is called the *correctness* of the protocol. In the meantime, no client should be able to falsify a ticket, our model encompass it in a property called *unforgeability.* The inability to infer the value of one of the user's ticket, *i.e.,* steal it, based on the transcripts holds under the name of *ticket-privacy.* Moreover a ticket might only be validated by a single holder, no two users should be accepted using the same ticket, this is called *no-double-spending.* And last, we expect some privacy for the ticket's holders. Users should remain unlinkable to the tickets they purchased, this property is called *pseudonymity.* It should also be hard to link tickets purchased by the same entity, this property is refered to as the *unlinkability.* Achieving these two properties is often refered to as the *anonymity* of the ticket's owners. The full formalism of the properties is given in Appendix B while a high level description is provided below.

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 to create a new valid ticket by observing and controlling users: the adversary has the ability to request users creations, purchases, refunds, transfers, validations, and user corruptions. It is expected to produce a new ticket not yet produced by the system and make it accepted either for a refund, a transfer or a validation. This property must be ensured for any PPT adversary, with a read-only access to the shared state, choosing among the Refund, Transfer and Validate algorithms.

Ticket privacy. An ETS scheme is *ticket private*, when a malicious entity, external to the system, cannot recover one of the other's client produced ticket. The adversary is assumed to be able to create, control the behaviors and corrupt any user of its choice. During the whole process, the adversary has a *read-only* access to the shared state at any point during the experiment. We simulate the user purchasing the ticket that have to be recovered by the adversary, but also $\mathcal{D}$, $\mathcal{T}$ and $\mathcal{V}$ since the system has to get the ticket to verify it. Therefore, the ticket privacy does not hold against the system and is rather focused against entities that are external to it, while still providing the full view of the shared state.

No-double-spending. A ticket should be deemed valid only once, hence, no PPT adversary $\mathcal{A}$ should be able to execute twice Refund, $\mathsf{Validate}$ and $\mathsf{Transfer}$ for the same ticket against a system simulated by the challenger. Once again, the adversary can request actions or corruption of clients, and has ready-only access to the shared state. Note that both execution are not independent as an updated shared state st is returned by 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 modeled by using two property 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 as well as the other users. *Pseudonymity* is modeled by a user $\mathcal{U}_b$ out of two users $\mathcal{U}_0$ and $\mathcal{U}_1$ executing multiple actions, the adversary trying to distinguish which one is acting. *Unlinkability* allows the adversary, during a preliminary stage, to interact both with $\mathcal{U}_0$ and $\mathcal{U}_1$. Then, the adversary interacts with the user $\mathcal{U}_b$, first for a purchase of a ticket, and then either to refund, transfer or validate the purchased ticket. The adversary has to guess which user acts last. We mitigate anonymity by introducing *Auditable E-Ticket scheme* enabling the recovery of the identity of users under the supervision of a judge.

Definition 2 (Auditable ETS). *An* Auditable E-Ticket scheme ETS *is an E-Ticket scheme increased with an audit algorithm:*
$\underline{\mathsf{JKeyGen}(1^\lambda) \rightarrow (\mathsf{sk}_\mathcal{J}, \mathsf{pk}_\mathcal{J})}$: *Given* 1^λ*, outputs a key pair.*
$\underline{\mathsf{Audit}(\mathsf{sk}_\mathcal{J}, \mathsf{tk}) \rightarrow \mathsf{pk}_\mathcal{U}}$ *: 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 should achieve the five described properties where requests of validation for $\mathcal{U}_0$ and $\mathcal{U}_1$ are not permitied through the oracles of the pseudonymity and unlinkability experiments.

4 Cryptographic Tools

$\mathsf{Applause}$ 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 $(\mathsf{sk}, \mathsf{pk})$. 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 $\mathsf{Adv}_{\mathcal{A}}^{\text{IND-CPA}}$ 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 $(\mathsf{sk}, \mathsf{pk})$, 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 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 $\mathsf{Adv}_{\mathcal{A}}^{\text{EUF-CMA}}$ 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 $\mathsf{RandKey}(\mathsf{sk}, \mathsf{pk}) \rightarrow (\mathsf{sk}', \mathsf{pk}')$ allowing to randomize 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 $\mathsf{RandKey}(\mathsf{sk}, \mathsf{pk}) \rightarrow (\mathsf{sk} \cdot r, \mathsf{pk}^r)$ for a r randomly sampled at uniform ($r \xleftarrow{\$} G$ denotes a uniform sample of r from the set G).

Ticket purchasing, transferring and refunding require payments. ETS must ensure the anonymity of users, but a standard online payment (*e.g.,* EMV payments [9]) reveals the user's identity. To address this issue, we consider a building block that we call *Anonymous Payment*, allowing a participant to perform a payment without revealing its identity. This building block is voluntary generic to let any anonymous payment method, such as [20,23], to be plugged in our protocol. An anonymous payment ensures *pseudonymity* preserving the identity of the user, and *unlinkability* preventing linkability of the payments. A payment scheme $\mathsf{P} = (\mathsf{KeyGen}, \mathsf{Pay})$ is a tuple of PPT algorithms, where:

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

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

This building block must achieve the two following properties:

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

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

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 $\mathsf{Pay}\langle \mathcal{C}(\mathsf{sk}_{\mathcal{U}_i}, \cdot), \mathcal{A}(\cdot)\rangle$ oracle for $i \in \{0, 1\}$. An Anonymous Payment scheme ensures *Unlinkability* if, for any 1^λ, advantage $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{PayUnl}}$ is

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

There exist numerous instantiations of anonymous payment. Distributed ledgers are good candidates, at condition that the ledger ensures anonymity such as Monero or Z-Cash. A more centralised approach called *Tokenization* payment [23], proposed to delegate the payment to a trusted third party called *Token Service Provider* (TSP), acting as a payment proxy. A TSP generates a token from the card number of the users, which does not reveal some information from the user identity. Plugging tokenization payment can be applied in our construction by setting the card number of the user as the secret payment key. At each payment, a new token is generated and used as the public key. This construction reaches our definition of anonymous payment. Indeed, since the payment is performed by a proxy and that the only information hold by the merchant during the transaction is the token, the pseudonymity property is ensured. The unlinkability is ensured since a token, unlinkable to another token, is used once.

5 Description of Applause

Informal Description. In the protocol, the participants interact using personal encryption keys $(\mathsf{sk}^{\mathsf{E}},\mathsf{sk}^{\mathsf{E}})$ and signature keys $(\mathsf{sk}^{\mathsf{S}},\mathsf{pk}^{\mathsf{S}})$ and proceed to the payments 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 $\mathsf{Purchase}$, the user first selects an event and a free seat $(\mathsf{ide},\mathsf{idp})$ 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 validation of a tickets goes through the showing of these two values to the validator and the agreement on a challenge value through a physical channel to authenticate the owner of the ticket. The first showing is similar for the refund process before the client is given back its funds or during the $\mathsf{Transfer}$ as the transfer authority $\mathcal{T}$ refunds the owner of the ticket and generates a new one with the new owner.

Our ETS ***Protocol.*** . A full diagram description of the protocol is given in Fig. 3 and Fig. 4 We first introduce a KeyGen algorithm:

$\underline{\mathsf{KeyGen}(\lambda)}$: Generates a signature key pair $(\mathsf{sk}^{\mathsf{S}},\mathsf{pk}^{\mathsf{S}}) \leftarrow \mathsf{S.KeyGen}(1^\lambda)$, an encryption key pair $(\mathsf{sk}^{\mathsf{E}},\mathsf{pk}^{\mathsf{E}}) \leftarrow \mathsf{E.KeyGen}(1^\lambda)$, a payment key pair $(\mathsf{sk}^{\mathsf{P}},\mathsf{pk}^{\mathsf{P}}) \leftarrow \mathsf{P.KeyGen}(1^\lambda)$, and outputs $(\mathsf{sk} \leftarrow (\mathsf{sk}^{\mathsf{S}},\mathsf{sk}^{\mathsf{E}},\mathsf{sk}^{\mathsf{P}}),\mathsf{pk} \leftarrow (\mathsf{pk}^{\mathsf{S}},\mathsf{pk}^{\mathsf{E}},\mathsf{pk}^{\mathsf{P}}))$.

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

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

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

$\underline{\mathsf{VKeyGen}(\lambda)}$: Generates a signature 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 $(\mathsf{sk}^{\mathsf{E}}_{\mathcal{V}}, \mathsf{pk}^{\mathsf{E}}_{\mathcal{V}}) \leftarrow \mathsf{E.KeyGen}(1^{\lambda})$. It queries the signature $\mathsf{Sign}_{\mathsf{sk}^{\mathsf{S}}_{\mathcal{D}}}(\mathsf{pk}^{\mathsf{S}}_{\mathcal{V}}, \mathsf{pk}^{\mathsf{E}}_{\mathcal{V}}) \rightarrow \mathit{cert}_{\mathcal{V}}$ to $\mathcal{D}$ on $\mathsf{pk}^{\mathsf{E}}_{\mathcal{V}}$ and outputs $(\mathsf{sk}_{\mathcal{V}} \leftarrow (\mathsf{sk}^{\mathsf{S}}_{\mathcal{V}}, \mathsf{sk}^{\mathsf{E}}_{\mathcal{V}}),$ $\mathsf{pk}_{\mathcal{V}} \leftarrow (\mathsf{pk}^{\mathsf{S}}_{\mathcal{V}}, \mathsf{pk}^{\mathsf{E}}_{\mathcal{V}}, \mathit{cert}_{\mathcal{V}}))$.

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 $\mathsf{idp} \leftarrow \mathsf{ID}_{\mathcal{P}}$.

$\underline{\mathsf{Purchase}\langle \mathcal{U}(\mathsf{sk}_{\mathcal{U}}, (\mathsf{ide}, \mathsf{idp})), \mathcal{D}(\mathsf{sk}_{\mathcal{D}}, st)\rangle}$: $\mathcal{U}$ 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 $(\mathsf{idp}, \mathsf{ide})$ 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, $\mathcal{U}$ proceeds to the payment 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 share state st: if $b = 1$, $st \leftarrow st \setminus \{c\}$ and returns b and st.

For clarity, we describe the $\mathsf{CheckTk}$ subroutine used in the following algorithms:

$\underline{\mathsf{CheckTk}(\mathsf{tk}, st)}$: Parses $\mathsf{tk} \xrightarrow{p} ((\mathsf{ide}, \mathsf{idp}, r_c), \sigma_c)$ ($\xrightarrow{p}$ denotes the parsing operation), gets $c \leftarrow H(\mathsf{ide}, \mathsf{idp}, r_c)$ and verifies $\mathsf{Verif}_{\mathsf{pk}^{\mathsf{S}}_{\mathcal{D}}}(c, \sigma_c)$ or $\mathsf{Verif}_{\mathsf{pk}^{\mathsf{S}}_{\mathcal{T}}}(c, \sigma_c)$. It 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.

$\underline{\mathsf{Refund}\langle \mathcal{U}(\mathsf{sk}_{\mathcal{U}}, \mathsf{tk}), \mathcal{D}(\mathsf{sk}_{\mathcal{D}}, st)\rangle}$: $\mathcal{U}$ 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 the ticket verification $\mathsf{CheckTk}(\mathsf{tk}, st) \rightarrow st$. 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 $\mathsf{Transfer}$ protocol assumes that user $\mathcal{U}_1$ holds the randomised public key $\mathsf{pk}'_{\mathcal{U}_2}$ of user $\mathcal{U}_2$ and conversely.

$\underline{\mathsf{Transfer}\langle \mathcal{U}_1(\mathsf{sk}'_{\mathcal{U}_1}, \mathsf{pk}'_{\mathcal{U}_2}, \mathsf{tk}_1), \mathcal{T}(\mathsf{sk}_{\mathcal{T}}, st), \mathcal{U}_2(\mathsf{sk}'_{\mathcal{U}_2}, \mathsf{pk}'_{\mathcal{U}_1}, (\mathsf{ide}, \mathsf{idp}))\rangle}$: $\mathcal{U}_1$ computes a signature $\mathsf{Sign}_{\mathsf{sk}^{\mathsf{S}'}_{\mathcal{U}_1}}(\mathsf{pk}'_{\mathcal{U}_2}, c) \rightarrow \sigma_{T,1}$ 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 $\mathsf{Sign}_{\mathsf{sk}^{\mathsf{S}'}_{\mathcal{U}_2}}(\mathsf{pk}'_{\mathcal{U}_1}) \rightarrow \sigma_{T,2}$ 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 $\mathsf{Verif}_{\mathsf{pk}^{\mathsf{S}'}_{\mathcal{U}_1}}(\mathsf{pk}'_{\mathcal{U}_2}, c, \sigma_{T,1})$ and $\mathsf{Verif}_{\mathsf{pk}^{\mathsf{S}'}_{\mathcal{U}_2}}(\mathsf{pk}'_{\mathcal{U}_1}, c, \sigma_{T,2})$, 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, $\mathcal{D}$ 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 $(\mathsf{ide}, \mathsf{idp})$ 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 $\mathsf{CheckTk}$ 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.

The validation protocol requires for its last interaction, a physical channel. This is assumed to prevent block-and-send attack, where an active external adversary blocks the ticket in the network, and play it in the validation process.

$\underline{\mathsf{Validate}\langle\mathcal{U}(\mathsf{sk}_{\mathcal{U}}, \mathsf{tk}), \mathcal{V}(\mathsf{sk}_{\mathcal{V}}, st)\rangle}$: $\mathcal{V}$ 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.,* $st \leftarrow st \backslash \{c\}$ and halts. Otherwise, both parties commonly return $s = s'$.

6 Diagram of Applause

In addition to the description of the protocol, we present a diagram of sequences to increase the readability and comprehension of our construction. In Fig. 3a, we present the diagram for the Purchase protocol. In Fig. 3b, we present the diagram for the Refund protocol. In Fig. 3c, we present the diagram for the Validate protocol. Finally, in Fig. 4, we present the diagram for the Transfer protocol. The diagrams present the non-auditable version of Applause.

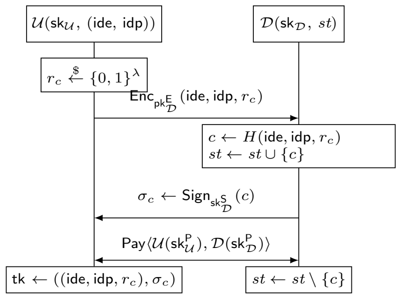
7 Security Analysis of Applause

7.1 Security Analysis of Applause in the Computational Model

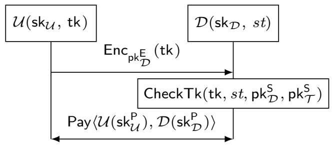
Six properties are provided through computational arguments: *correctness*, *unforgeability*, *ticket privacy*, *no-double-spending* and *anonymity*, the latter being divided into *pseudonymity* and *unlinkability*. The associated games are sketched in Section 3. Below we summarise our hypothesis and 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 1. Applause *achieves correctness under honest execution.*

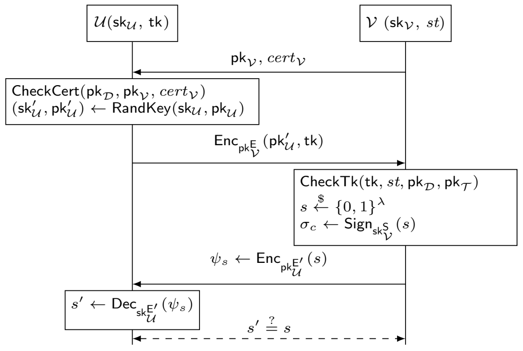
Assuming correctness of the encryption, correctness of the hash function and correctness of the signature scheme, then we clearly observe a correspondence between the signature of the hashed random r_c (produced either by $\mathcal{D}$ or $\mathcal{T}$), and the validator able to verify the signature. An analysis of the update of the shared state also allows to conclude that the shared state remains coherent with the status of the tickets.


$\mathcal{U}(\mathsf{sk}_\mathcal{U},\ (\mathsf{ide},\ \mathsf{idp}))$
$\mathcal{D}(\mathsf{sk}_\mathcal{D},\ st)$
$r_c \xleftarrow{\$} \{0,1\}^\lambda$
$\mathsf{Enc}_{\mathsf{pk}^\mathsf{E}_\mathcal{D}}(\mathsf{ide}, \mathsf{idp}, r_c)$
$c \leftarrow H(\mathsf{ide}, \mathsf{idp}, r_c)$
$st \leftarrow st \cup \{c\}$
$\sigma_c \leftarrow \mathsf{Sign}_{\mathsf{sk}^\mathsf{S}_\mathcal{D}}(c)$
$\mathsf{Pay}\langle \mathcal{U}(\mathsf{sk}^\mathsf{P}_\mathcal{U}), \mathcal{D}(\mathsf{sk}^\mathsf{P}_\mathcal{D})\rangle$
$\mathsf{tk} \leftarrow ((\mathsf{ide}, \mathsf{idp}, r_c), \sigma_c)$
$st \leftarrow st \setminus \{c\}$

(a) Diagram of Purchase protocol.


$\mathcal{U}(\mathsf{sk}_\mathcal{U},\ \mathsf{tk})$
$\mathcal{D}(\mathsf{sk}_\mathcal{D},\ st)$
$\mathsf{Enc}_{\mathsf{pk}^\mathsf{E}_\mathcal{D}}(\mathsf{tk})$
$\mathsf{CheckTk}(\mathsf{tk}, st, \mathsf{pk}^\mathsf{S}_\mathcal{D}, \mathsf{pk}^\mathsf{S}_\mathcal{T})$
$\mathsf{Pay}\langle \mathcal{U}(\mathsf{sk}^\mathsf{P}_\mathcal{U}), \mathcal{D}(\mathsf{sk}^\mathsf{P}_\mathcal{D})\rangle$

(b) Diagram of Refund protocol.


$\mathcal{U}(\mathsf{sk}_\mathcal{U},\ \mathsf{tk})$
$\mathcal{V}\ (\mathsf{sk}_\mathcal{V},\ st)$
$\mathsf{pk}_\mathcal{V}, cert_\mathcal{V}$
$\mathsf{CheckCert}(\mathsf{pk}_\mathcal{D}, \mathsf{pk}_\mathcal{V}, cert_\mathcal{V})$
$(\mathsf{sk}'_\mathcal{U}, \mathsf{pk}'_\mathcal{U}) \leftarrow \mathsf{RandKey}(\mathsf{sk}_\mathcal{U}, \mathsf{pk}_\mathcal{U})$
$\mathsf{Enc}_{\mathsf{pk}^\mathsf{E}_\mathcal{V}}(\mathsf{pk}'_\mathcal{U}, \mathsf{tk})$
$\mathsf{CheckTk}(\mathsf{tk}, st, \mathsf{pk}_\mathcal{D}, \mathsf{pk}_\mathcal{T})$
$s \xleftarrow{\$} \{0,1\}^\lambda$
$\sigma_c \leftarrow \mathsf{Sign}_{\mathsf{sk}^\mathsf{S}_\mathcal{V}}(s)$
$\psi_s \leftarrow \mathsf{Enc}_{\mathsf{pk}^{\mathsf{E}\prime}_\mathcal{U}}(s)$
$s' \leftarrow \mathsf{Dec}_{\mathsf{sk}^{\mathsf{E}\prime}_\mathcal{U}}(\psi_s)$
$s' \overset{?}{=} s$

(c) Diagram of sequences for the Validate protocol. The dashed line is the physical channel.

Fig. 3: Diagram of sequences for Purchase, Refund and Validate protocols.

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

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 $\mathsf{sk}^\mathsf{S}_\mathcal{T}$. The unforgeability of a ticket is prevented by the EUF-CMA property of the signatures and the shared state st composed of invalidate tickets.

Theorem 3. *Instantiated with statistically indistinguishable randomisable keys, and an anonymous payment scheme,* Applause *provides* pseudonymity *under the following probability* $|\Pr[\mathsf{Exp}^{\mathsf{PSE}}_\mathcal{A}(1^\lambda) \to 1] - \frac{1}{2}| \leq \mathsf{Adv}^{\mathsf{PayPse}}_\mathcal{A}(1^\lambda)$ *for every* PPT $\mathcal{A}$.

The identity of a user $\mathcal{U}$ is involved through its randomised key $\mathsf{pk}'_\mathcal{U}$ and during the payment. Anonymity is ensured using the statistical indishtinguishability of randomisable keys and an anonymous payment. Moreover, the numerous mes-

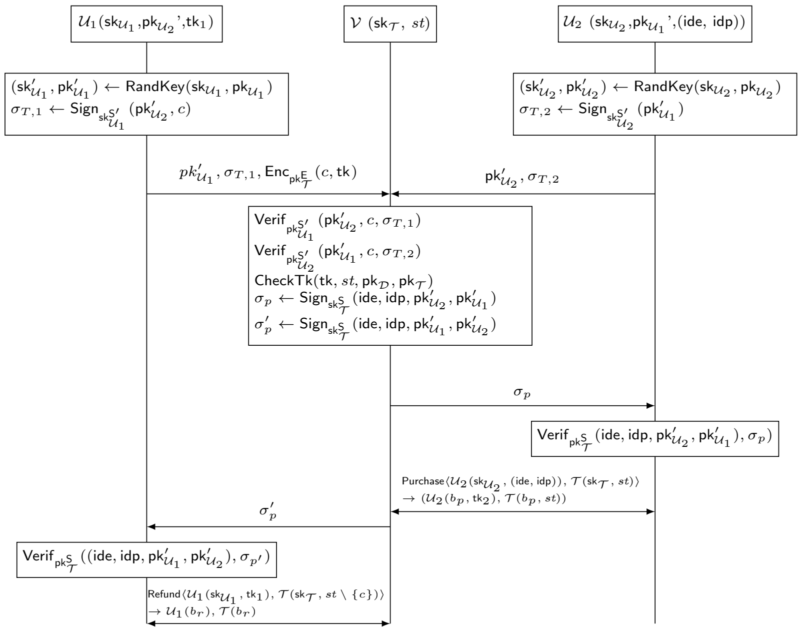

$\mathcal{U}_1(\mathsf{sk}_{\mathcal{U}_1}, \mathsf{pk}_{\mathcal{U}_2}', \mathsf{tk}_1)$
$\mathcal{V}\ (\mathsf{sk}_{\mathcal{T}},\ st)$
$\mathcal{U}_2\ (\mathsf{sk}_{\mathcal{U}_2}, \mathsf{pk}_{\mathcal{U}_1}', (\mathsf{ide},\ \mathsf{idp}))$
$(\mathsf{sk}'_{\mathcal{U}_1}, \mathsf{pk}'_{\mathcal{U}_1}) \leftarrow \mathsf{RandKey}(\mathsf{sk}_{\mathcal{U}_1}, \mathsf{pk}_{\mathcal{U}_1})$
$\sigma_{T,1} \leftarrow \mathsf{Sign}_{\mathsf{sk}^{\mathsf{S}\prime}_{\mathcal{U}_1}}(\mathsf{pk}'_{\mathcal{U}_2}, c)$
$(\mathsf{sk}'_{\mathcal{U}_2}, \mathsf{pk}'_{\mathcal{U}_2}) \leftarrow \mathsf{RandKey}(\mathsf{sk}_{\mathcal{U}_2}, \mathsf{pk}_{\mathcal{U}_2})$
$\sigma_{T,2} \leftarrow \mathsf{Sign}_{\mathsf{sk}^{\mathsf{S}\prime}_{\mathcal{U}_2}}(\mathsf{pk}'_{\mathcal{U}_1})$
$pk'_{\mathcal{U}_1}, \sigma_{T,1}, \mathsf{Enc}_{\mathsf{pk}^{\mathsf{E}}_{\mathcal{T}}}(c, \mathsf{tk})$
$\mathsf{pk}'_{\mathcal{U}_2}, \sigma_{T,2}$
$\mathsf{Verif}_{\mathsf{pk}^{\mathsf{S}\prime}_{\mathcal{U}_1}}(\mathsf{pk}'_{\mathcal{U}_2}, c, \sigma_{T,1})$
$\mathsf{Verif}_{\mathsf{pk}^{\mathsf{S}\prime}_{\mathcal{U}_2}}(\mathsf{pk}'_{\mathcal{U}_1}, c, \sigma_{T,2})$
$\mathsf{CheckTk}(\mathsf{tk}, st, \mathsf{pk}_{\mathcal{D}}, \mathsf{pk}_{\mathcal{T}})$
$\sigma_p \leftarrow \mathsf{Sign}_{\mathsf{sk}^{\mathsf{S}}_{\mathcal{T}}}(\mathsf{ide}, \mathsf{idp}, \mathsf{pk}'_{\mathcal{U}_2}, \mathsf{pk}'_{\mathcal{U}_1})$
$\sigma'_p \leftarrow \mathsf{Sign}_{\mathsf{sk}^{\mathsf{S}}_{\mathcal{T}}}(\mathsf{ide}, \mathsf{idp}, \mathsf{pk}'_{\mathcal{U}_1}, \mathsf{pk}'_{\mathcal{U}_2})$
σ_p
$\mathsf{Verif}_{\mathsf{pk}^{\mathsf{S}}_{\mathcal{T}}}(\mathsf{ide}, \mathsf{idp}, \mathsf{pk}'_{\mathcal{U}_2}, \mathsf{pk}'_{\mathcal{U}_1}), \sigma_p)$
$\mathsf{Purchase}\langle \mathcal{U}_2(\mathsf{sk}_{\mathcal{U}_2}, (\mathsf{ide}, \mathsf{idp})), \mathcal{T}(\mathsf{sk}_{\mathcal{T}}, st)\rangle \rightarrow (\mathcal{U}_2(b_p, \mathsf{tk}_2), \mathcal{T}(b_p, st))$
σ'_p
$\mathsf{Verif}_{\mathsf{pk}^{\mathsf{S}}_{\mathcal{T}}}((\mathsf{ide}, \mathsf{idp}, \mathsf{pk}'_{\mathcal{U}_1}, \mathsf{pk}'_{\mathcal{U}_2}), \sigma_{p'})$
$\mathsf{Refund}\langle \mathcal{U}_1(\mathsf{sk}_{\mathcal{U}_1}, \mathsf{tk}_1), \mathcal{T}(\mathsf{sk}_{\mathcal{T}}, st \setminus \{c\})\rangle \rightarrow \mathcal{U}_1(b_r), \mathcal{T}(b_r)$

Fig. 4: Diagram of sequences for the Transfer protocol.

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

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 $\{0,1\}^{\lambda}$ and cannot be recovered. Hence, tk cannot be recovered by an adversary, as r_c can be guessed only with probability $2^{-\lambda}$.

Theorem 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 $\mathsf{CheckTk}$ 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 6. *Instantiated with an anonymous payment having the property of randomisable keys, then for any PPT $\mathcal{A}$,* $\mathsf{Applause}$ *provides* unlinkability *under the probability* $|\Pr[\mathsf{Exp}_{\mathcal{A}}^{\mathsf{UNL}}(\lambda) \to 1] - \frac{1}{2}| \leq \frac{3}{2} \cdot \mathsf{Adv}_{\mathcal{A}}^{\mathsf{PayUnl}}(1^{\lambda})$.

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 $\mathsf{Transfer}$ 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 $\mathsf{Exp}_{\mathcal{A}}^{\mathsf{UNL}}$ for a given b and playing at experiment $\mathsf{Exp}_{\mathcal{A}}^{\mathsf{UNL}}$ 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 (sketched) proofs of security above 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 $(\mathsf{ide}, \mathsf{idp})$ identifying the ticket to the user knowing r_c. The user being the only one able to open this commitment under the preimage resistance of the hash function, hence the only one to identify himself as the owner of the ticket. One may prefer a protocole proven secure in the standard model. This can be easily achieved based on the Pedersen commitment [24]: 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, here, as 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 $\mathsf{Applause}$ can be found in Appendix C.2.

7.2 Formal Verification of Applause in Symbolic Model

To perform a symbolic verification of the protocol, we use ProVerif [5]. 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

Security Properties	Proving Time (ProVerif)
Authentication: $\mathcal{U} \leftrightarrow \mathcal{D}$	$< 1s$
Authentication: $\mathcal{U}_1 \leftrightarrow \mathcal{U}_2$	$< 1s$
Authentication: $\mathcal{U} \leftrightarrow \mathcal{T}$	$< 1s$
Authentication: $\mathcal{U} \leftrightarrow \mathcal{V}$	$< 1s$
Secrecy of ticket	$< 1s$
Unforgeability of ticket	$< 1s$
Double-(Refund/Transfer/Validate)	< 1s
Anonymity of $\mathcal{U}_1$ and $\mathcal{U}_2$	56s

Fig. 5: Execution time to prove the security properties.

the behavior of the protocol and which are inserted at precise locations to allow reasoning about the protocol's execution. Events allow to check 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 $\mathcal{E}' \to \mathcal{E}$ 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 $\mathcal{E}' \to \mathcal{E}$ and $\mathcal{E} \to \mathcal{E}'$. 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 [7] which has a complete control of the network: the attacker eavesdrops, removes, substitutes, duplicates and delays any messages. ProVerif achieves a proof for all stated properties: anonymity of $\mathcal{U}_1$ and $\mathcal{U}_2$ takes less than a minute, while all others are achieved after only one second of execution. Our model is available in [1] and a full recap of the proven properties is given in Figure 5.

8 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 [6] is based on *Structure-preserving signatures on equivalence classes* [10]. We introduce a generic description of the algorithms for auditable certificates on randomisable keys:

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

$\mathsf{Spotlight}$ achieves the same level of security compared to $\mathsf{Applause}$, only relaxing the anonymity of the users. 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. In this section we highlight the modification to apply on $\mathsf{Applause}$ to obtain an auditable version. The full auditable protocol description is presented in Appendix D, and the security analysis in Appendix E.

$\underline{\mathsf{JKeyGen}(1^\lambda):}$ Runs $\mathsf{OKeyGen}(1^\lambda) \rightarrow (\mathsf{sk}_{\mathcal{J}}, \mathsf{pk}_{\mathcal{J}})$ and outputs the key pair.

$\underline{\mathsf{UKeyGen}:}$ Additionally obtains the certificate $\mathit{cert}_{\mathcal{U}}$ from the certification algorithm $\mathsf{Certify}\langle \mathcal{U}(\mathsf{sk}^{\mathsf{S}}_{\mathcal{U}}, \mathsf{pk}^{\mathsf{S}}_{\mathcal{U}}), \mathcal{J}(\mathsf{sk}_{\mathcal{J}}) \rangle$. It returns it as part of $\mathsf{pk}_{\mathcal{U}}$.

$\underline{\mathsf{Purchase}:}$ $\mathcal{U}$ updates its signature's keys using the randomisation algorithm $\mathsf{RandID}(\mathsf{sk}^{\mathsf{S}}_{\mathcal{U}}, \mathsf{pk}^{\mathsf{S}}_{\mathcal{U}}, \mathit{cert}_{\mathcal{U}}) \rightarrow (\mathsf{sk}^{\mathsf{S}'}_{\mathcal{U}}, \mathsf{pk}^{\mathsf{S}'}_{\mathcal{U}}, \mathit{cert}'_{\mathcal{U}})$, and sends the message $\mathsf{pk}'_{\mathcal{U}} = (\mathsf{pk}^{\mathsf{S}'}_{\mathcal{U}}, \mathsf{pk}^{\mathsf{E}'}_{\mathcal{U}})$, $\mathit{cert}'_{\mathcal{U}}$, $\psi = \mathsf{Enc}_{\mathsf{pk}^{\mathsf{E}}_{\mathcal{D}}}(\mathsf{ide}, \mathsf{idp}, r_c, c)$ and $\sigma_\psi = \mathsf{Sign}_{\mathsf{sk}^{\mathsf{S}'}_{\mathcal{U}}}(\psi)$. $\mathcal{D}$ executes the certification verification function $\mathsf{CheckCert}(\mathsf{pk}_{\mathcal{J}}, \mathit{cert}'_{\mathcal{U}})$ and executes $\mathsf{Verif}_{\mathsf{pk}^{\mathsf{S}}_{\mathcal{U}}}(\psi, \sigma_\psi)$. $\mathcal{D}$ updates st as $\mathit{st} \leftarrow \mathit{st} \cup \{c, \mathit{cert}'_{\mathcal{U}}, 0\}$ and inverts the bit to obtain $\{c, \mathit{cert}'_{\mathcal{U}}, 1\}$ when the ticket is payed.

$\underline{\mathsf{CheckTk}:}$ Additionally takes as input a certificate cert, executes $\mathsf{CheckCert}(\mathsf{pk}_{\mathcal{J}}, \mathit{cert})$ and checks the state by verifying $(c, \cdot, 1) \in \mathit{st}$ instead of $c \in \mathit{st}$. If all verification success, then we update $(c, \cdot, 1)$ to $(c, \mathit{cert}, 0)$.

$\underline{\mathsf{Refund}:}$ $\mathcal{U}$ updates its keys as described in $\mathsf{Purchase}$ and sends $\mathsf{pk}'_{\mathcal{U}}, \mathit{cert}'_{\mathcal{U}}, \psi = \mathsf{Enc}_{\mathsf{pk}^{\mathsf{E}}_{\mathcal{D}}}(\mathsf{tk}), \mathsf{Sign}_{\mathsf{sk}^{\mathsf{S}}_{\mathcal{U}}}(\psi)$ to $\mathcal{D}$ as its first message. The signature is verified and then $\mathsf{CheckTk}$ executed with $\mathit{cert}'_{\mathcal{U}}$ as its additional input. In case of a payment failure, $\mathcal{D}$ reverts $(c, \cdot, 0)$ to $(c, \mathit{cert}'_{\mathcal{U}}, 1)$.

$\underline{\mathsf{Transfer}:}$ Both $\mathcal{U}_1$ and $\mathcal{U}_2$ respectively send $\mathit{cert}'_{\mathcal{U}_1}$ and $\mathit{cert}'_{\mathcal{U}_2}$ to $\mathcal{T}$, and verify $\mathit{cert}'_{\mathcal{U}_1}$ and $\mathit{cert}'_{\mathcal{U}_2}$ using $\mathsf{CheckCert}$. Eventually, if $\mathsf{Purchase}$ failed, $\mathcal{T}$ reverts $(c, \cdot, 0)$ to $(c, \mathit{cert}'_{\mathcal{U}_1}, 1)$.

$\underline{\mathsf{Validate}:}$ $\mathcal{U}$ updates its keys as above and sends $\mathsf{Enc}_{\mathsf{pk}^{\mathsf{E}}_{\mathcal{V}}}(\mathsf{pk}'_{\mathcal{U}}, \mathit{cert}'_{\mathcal{U}}, \mathsf{tk})$ as its first message. $\mathcal{V}$ inputs $\mathit{cert}'_{\mathcal{U}}$ into $\mathsf{CheckCert}$.

$\underline{\mathsf{Audit}(\mathsf{sk}_{\mathcal{J}}, (\cdot, \cdot, \mathit{cert}))}$ Executes $\mathsf{Audit}(\mathsf{sk}_{\mathcal{J}}, \mathit{cert}) \rightarrow \mathsf{pk}_{\mathcal{U}}$ and returns $\mathsf{pk}_{\mathcal{U}}$.

Theorem 7. *Instantiated with an anonymous payment, an* EUF-CMA *signature with randomisable keys and an* IND-CPA *encryption scheme, auditable certificates on randomisable keys, our construction ensures anonymity, unlinkability, no-double-spending, unforgeability and ticket privacy.*

The proof of this theorem is given in Appendix E.

9 Conclusion

We presented $\mathsf{Applause}$ a ticketing system preserving the physical aspects of paper tickets while ensuring that any user can buy, refund, validate and transfer a ticket. The system preserves also the privacy of users. The security proofs ensure unforgeability, no-double-spending, privacy of the tickets and anonymity of users in the random oracle model. We have also formally verified the security properties of Applause using ProVerif, proving the security even with parallel sessions. The efficiency and the scalability of Applause has been demonstrated by an implementation in Rust. We have extended Applause to obtain an auditable version called Spotlight, where the judge is able to reveal the identity of users, whereas other entities are not able to deduce whether a user has purchased a ticket and for which event.

References

1. Anonymous. Transferable, auditable and anonymous ticketing protocol. `https://anonymous.4open.science/r/ETS-085F`.
2. Aventus. `https://aventus.io/`, 2016.
3. F. Baldimtsi, M. Chase, G. Fuchsbauer, and M. Kohlweiss. Anonymous transferable e-cash. In *International Workshop on Public Key Cryptography*. Springer, 2015.
4. B. Bauer, G. Fuchsbauer, and C. Qian. Transferable e-cash: A cleaner model and the first practical instantiation. Cryptology ePrint Archive, Paper 2020/1400, 2020.
5. B. Blanchet. Automatic Verification of Security Protocols in the Symbolic Model: The Verifier ProVerif. In *Foundations of Security Analysis and Design VII*. Springer, 2014.
6. A. Connolly, J. Deschamps, P. Lafourcade, and O. P. Kempner. Protego: Efficient, revocable and auditable anonymous credentials with applications to hyperledger fabric. Cryptology ePrint Archive, Paper 2022/661, 2022.
7. D. Dolev and A. Yao. On the security of public key protocols. *IEEE Transactions on Information Theory*, 1983.
8. T. ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE transactions on information theory*, 1985.
9. EMVCo. Book 1: Application independent icc to terminal interface requirements. 2011.
10. G. Fuchsbauer, C. Hanser, and D. Slamanig. Structure-preserving signatures on equivalence classes and constant-size anonymous credentials. *Journal of Cryptology*, 2019.
11. I. Gudymenko. 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, 2013.
12. I. Gudymenko, F. Sousa, and S. Kopsell. 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, 2014.
13. N. A. Hamid, M. F. A. A'zhim, and M. L. Yap. e-ticketing system for football events in malaysia. In *7th International Conference for Internet Technology and Secured Transactions, ICITST*. IEEE, 2012.
14. J. Han, L. Chen, S. Schneider, H. Treharne, and S. Wesemeyer. Privacy-preserving electronic ticket scheme with attribute-based credentials. *IEEE Trans. Dependable Secur. Comput.*, 2021.
15. W. H. W. Hussin, P. Coulton, and R. Edwards. Mobile ticketing system employing trustzone technology. In *International Conference on Mobile Business*. IEEE Computer Society, 2005.
16. F. Kerschbaum, H. W. Lim, and I. Gudymenko. 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, 2013.
17. X. Li, Z. Liu, and Z. Guo. Formal object-oriented analysis and design of an online ticketing system. In *8th Asia-Pacific Software Engineering Conference (APSEC 2001)*. IEEE Computer Society, 2001.
18. X. Li, J. Niu, J. Gao, and Y. Han. Secure electronic ticketing system based on consortium blockchain. *KSII Trans. Internet Inf. Syst.*, 2019.
19. W. Liu, Y. Mu, and G. Yang. An efficient privacy-preserving e-coupon system. In *International Conference on Information Security and Cryptology*. Springer, 2014.

20. A. Madhusudan, M. Sedaghat, P. Jovanovic, and B. Preneel. Nirvana: Instant and anonymous payment-guarantees. *IACR Cryptol. ePrint Arch.*, page 872, 2022.
21. M. Milutinovic, K. Decroix, V. Naessens, and B. D. Decker. Privacy-preserving public transport ticketing system. In *Data and Applications Security and Privacy-29th Annual IFIP WG 11.3 Working Conference, DBSec*. Springer, 2015.
22. L. S. Nair, V. S. Arun, and S. Joseph. 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, 2015.
23. C. Nugier, D. Leblanc-Albarel, A. Blaise, S. Masson, P. Huynh, and Y. B. Wandji Piugie. An Upcycling Tokenization Method for Credit Card Numbers. In *SECRYPT - 18th International Conference on Security and Cryptography*, 2021.
24. T. P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In *Advances in Cryptology—CRYPTO'91: Proceedings*, pages 129–140. Springer, 2001.
25. G. Protocol. GUTS Ticketing. `https://guts.tickets/`, 2016.
26. C.-P. Schnorr. Efficient signature generation by smart cards. *Journal of cryptology*, 1991.
27. M. Sel, S. Seys, and E. R. Verheul. The security of mass transport ticketing systems. In *ISSE 2008 - Securing Electronic Busines Processes, Highlights of the Information Security Solutions Europe 2008 Conference*, 2008.
28. I. Teranishi, J. Furukawa, and K. Sako. k-times anonymous authentication (extended abstract). In *Advances in Cryptology - ASIACRYPT 2004*. Springer, 2004.
29. M. Xie, M. Tomlinson, and B. Bodenheimer. Interface design for a modern software ticketing system. In *Proceedings of Annual Southeast Regional Conference*. ACM, 2004.
30. Y. YuanJiang and J. T. Zhou. Ticketing system based on NFT. In *24th IEEE International Workshop on Multimedia Signal Processing, MMSP 2022*. IEEE, 2022.

A Scalability of Applause

We study the scalability of our implementation of Applause in Rust, proving its possible usage at large scale. In our implementation, we have measure the impact of a thousand participants event purchasing a ticket. After the purchase, every purchased tickets are transferred, in order to load the shared state while producing the new (transferred) tickets. Such approach allows us to take in account the impact of the shared state, which is negligible as confirmed by the results. Finally, we execute the refund and validation with the tickets. Presented execution times are computed as the mean of 10 iterations on an Ubuntu laptop, embedding an Intel i7-12800H processor and 32Gb of memory. We exclude the communication time. We rely on standard cryptographic primitives: we used the curve25519 curve for ElGamal [8] and Schnorr signatures [26]. The results are depicted in in Fig 6. We show that Transfer requires an execution time around 40ms, even with a thousand of already transferred tickets as the execution time remains constant regarding the number of already emitted tickets. We require 30ms for Validate, 15ms for Refund and 10ms for Purchase. Hence, the overhead brought by securing the protocol seems acceptable. Therefore, our instanciation is efficient and scalable.

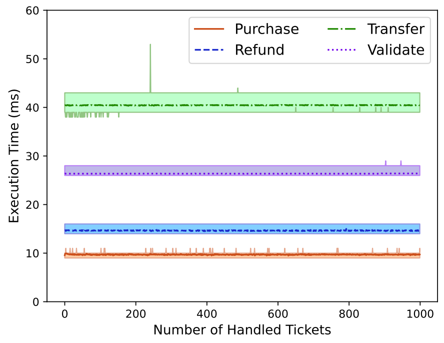


Fig. 6: Execution times (in milliseconds) of the protocol, depending on the number of handled tickets. The curves represent the mean of 10 iterations, whereas the edges of the areas represents the min and max execution times.

B Security Experiments

In Fig. 7, we present the oracles used in the experiments presented in Fig. 8. We recall that an informal overview of the security experiments is presented in Section 3.

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 $\mathsf{G}^i_{\mathcal{A}}(1^\lambda)$ 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 $\mathsf{G}^i_{\mathcal{A}}(1^\lambda)$ outputs 1. At first we presents the security proof 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 the both versions of Applause differ only in few points, as explained in Section 7.

C.1 Security Proofs of Applause in Random Oracle Model

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

Proof. Let $\mathcal{A}$ be a PPT adversary and assume for contradiction that the following probability $\Pr[\mathsf{Exp}^{\mathsf{UF}}_{\mathcal{A}}(1^\lambda) \to 1]$ is non-negligible. This proof relies on the ROM, meaning that $\mathcal{A}$ has access to an oracle Ohash replacing the function H and

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

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

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

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

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

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

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

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

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

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 $\mathsf{pk}_{\mathcal{U}}$
```

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

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

```
$\mathsf{OLeakState}(st)$

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 $\mathsf{Exp}^{\mathsf{PRIV}}_{\mathcal{A}}$.

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

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

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

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

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

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

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

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

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

Fig. 7: In each oracle, arguments provided by the challenger $\mathcal{C}$ appear before the ';', while arguments specified after are provided by $\mathcal{A}$.

```
$\mathsf{Exp}^{\mathsf{UF}}_{\mathcal{A}}(1^\lambda)$
$(\mathsf{sk}_\mathcal{D}, \mathsf{pk}_\mathcal{D}), (\mathsf{sk}_\mathcal{T}, \mathsf{pk}_\mathcal{T}), (\mathsf{sk}_\mathcal{V}, \mathsf{pk}_\mathcal{V})$
    $\leftarrow \mathsf{DKeyGen}/\mathsf{TKeyGen}/\mathsf{VKeyGen}(1^\lambda)$
$\mathcal{U} \leftarrow \emptyset, st \leftarrow \emptyset, \mathsf{TK} \leftarrow \emptyset$
$(\mathsf{tk}, \mathsf{Alg}) \leftarrow \mathcal{A}^{\mathcal{O}}(\mathsf{pk}_\mathcal{D}, \mathsf{pk}_\mathcal{T}, \mathsf{pk}_\mathcal{V})$
$b_0 \leftarrow \mathsf{tk} \notin \mathsf{TK}$ // *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** $\mathsf{sk} \leftarrow \mathsf{sk}_\mathcal{V}$
    $\mathsf{Alg}\langle \mathcal{A}(\cdot,\cdot), \mathcal{C}(\mathsf{sk}, st)\rangle \to \mathcal{A}(\cdot), \mathcal{C}(b_1, \mathsf{tk}_0, st)$
    **return** $b_0 \wedge b_1$
**else if** $\mathsf{Alg} = \mathsf{Transfer}$ :
    $\mathsf{Transfer}\langle \mathcal{A}(\cdot,\cdot), \mathcal{C}(\mathsf{sk}_\mathcal{T}, st), \mathcal{A}(\cdot)\rangle$
        $\to \mathcal{A}(\cdot), \mathcal{C}(b_1, (\mathsf{ide}_1, \mathsf{idp}_1), st), \mathcal{A}(\cdot)$
    **return** $b_0 \wedge b_1$
**else** : **return** 0
```

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

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

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

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

Fig. 8: 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 $\mathsf{Exp}^{\mathsf{PSE}}_{\mathcal{A}}$. $\mathcal{O}$ equals {OPurchase, ORefund, OTransfer, OValidate, OCreateUser, OCorruptUser, OLeakState} and $\mathcal{O}_b$ equals {$\mathsf{OPurchase}_b$, $\mathsf{ORefund}_b$, $\mathsf{OTransfer}_b$, $\mathsf{OValidate}_b$}

programmed by the challenger $\mathcal{C}$, 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 $\mathsf{Exp}_{\mathcal{A}}^{\mathsf{UF}}$ defined in Fig. 8, we observe that

$$\Pr[W_0] = \Pr[\mathsf{Exp}_{\mathcal{A}}^{\mathsf{UF}}(1^\lambda) \to 1]. \tag{1}$$

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, $\mathcal{C}$ 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 $\mathsf{Adv}_{\mathcal{A}}^{\text{EUF-CMA}}$. Therefore, we obtain the following bound:

$$|\Pr[W_0] - \Pr[W_1]| \leq \mathsf{Adv}_{\mathcal{A}}^{\text{EUF-CMA}}(1^\lambda). \tag{2}$$

Game G^2. The same action as in G^1 is performed on the signature produced by $\mathcal{T}$, leading to the bound:

$$|\Pr[W_1] - \Pr[W_2]| \leq \mathsf{Adv}_{\mathcal{A}}^{\text{EUF-CMA}}(1^\lambda). \tag{3}$$

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 $\mathsf{CheckTk}$ algorithm executed in $\mathsf{Validate}$, Refund or $\mathsf{Transfer}$. $\mathcal{A}$ 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 $\mathcal{C}$. 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 $\mathcal{C}$ and the ticket will fail to verify in the last verification of the $\mathsf{CheckTk}$ algorithm. This can been seen by analysing the inclusion of c in st through the processes. In both cases, a condition of $\mathsf{CheckTk}$ is not achieved and the protocol aborts for $\mathcal{C}$. This shows that $\Pr[W_0] = 0$, and then

$$\Pr[\mathsf{Exp}_{\mathcal{A}}^{\mathsf{UF}}(1^\lambda) \to 1] \leq 2 \cdot \mathsf{Adv}_{\mathcal{A}}^{\text{EUF-CMA}}(1^\lambda) \tag{4}$$

By hypothesis on the EUF-CMA property of the signature S, we know that $\mathsf{Adv}_{\mathcal{A}}^{\text{EUF-CMA}}$ is negligible, leading to a negligible quantity on the right hand side of Equation 4, in direct contraction with the initial hypothesis. Hence, $\Pr[\mathsf{Exp}_{\mathcal{A}}^{\mathsf{UF}}(1^\lambda) \to 1]$ is negligible and $\mathsf{Applause}$ has Unforgeability.

Theorem 3. *Instantiated with statistically indistinguishable randomisable keys, and an anonymous payment scheme,* $\mathsf{Applause}$ *provides* pseudonymity *under the following probability* $|\Pr[\mathsf{Exp}_{\mathcal{A}}^{\mathsf{PSE}}(1^\lambda) \to 1] - \frac{1}{2}| \leq \mathsf{Adv}_{\mathcal{A}}^{\mathsf{PayPse}}(1^\lambda)$ *for every* PPT $\mathcal{A}$.

Proof. 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 $\mathsf{Exp}_{\mathcal{A}}^{\mathsf{PSE}}$. Observe that

$$\Pr[W_0] = \Pr[\mathsf{Exp}_{\mathcal{A}}^{\mathsf{PSE}}(1^\lambda) \to 1]. \tag{5}$$

Game G^1. We focus on the anonymous payment. At the beginning, the challenger generates a new key pair and uses this key $\mathsf{sk}_{\mathsf{new}}^{\mathsf{P}}$ instead of using $\mathsf{sk}_{\mathcal{U}_b}^{\mathsf{Pay}}$ in all payments of the experiment. By hypothesis, an adversary has a negligible advantage $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{PayPse}}(1^\lambda)$ 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 $\mathsf{sk}_{\mathcal{U}_0}^{\mathsf{Pay}}$ and $\mathsf{sk}_{\mathcal{U}_1}^{\mathsf{Pay}}$ unused. Hence,

$$|\Pr[W_0] - \Pr[W_1]| \leq \mathsf{Adv}_{\mathcal{A}}^{\mathsf{PayPse}}(1^\lambda) \tag{6}$$

Now, we notice that the user's key is not used in $\mathsf{Purchase}$ and Refund. Only $\mathsf{Transfer}$ and $\mathsf{Validate}$ 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 randomising the key pair in $\mathsf{Validate}$, 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 randomisable key pairs, they where already unrelated after randomisation, hence resulting in a bridging step with

$$\Pr[W_2] = \Pr[W_1]. \tag{7}$$

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

$$\Pr[W_3] = \Pr[W_2]. \tag{8}$$

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

$$\Pr[W_3] = \frac{1}{2}. \tag{9}$$

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 $\mathcal{U}_b$. By equations (5) to (9), we observe that

$$|\Pr[\mathsf{Exp}_{\mathcal{A}}^{\mathsf{PSE}}(1^\lambda) \to 1] - \frac{1}{2}| \leq \mathsf{Adv}_{\mathcal{A}}^{\mathsf{PayPse}}(1^\lambda).$$

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

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

Proof. 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 $\mathsf{Exp}_{\mathcal{A}}^{\mathsf{PRIV}}(1^{\lambda})$, or at least with a negligible probability. Notice that $\mathcal{U}$, the user simulated by $\mathcal{C}$ 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 games corresponds to the experiment $\mathsf{Exp}_{\mathcal{A}}^{\mathsf{PRIV}}$, hence

$$\Pr[W_0] = \Pr[\mathsf{Exp}_{\mathcal{A}}^{\mathsf{PRIV}}(1^{\lambda}) \to 1] \tag{10}$$

Game G^1. First we modify the $\mathsf{Purchase}$ 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 $\mathsf{Enc}_{\mathsf{pk}_{\mathcal{D}}^{\mathsf{E}}}(\mathsf{ide}, \mathsf{idp}, r_c, c)$ by a random sampled uniformly from $[\mathsf{Enc}_{\mathsf{pk}_{\mathcal{D}}^{\mathsf{E}}}]$. $\mathcal{D}$ being also simulated by $\mathcal{C}$ it still has access to $\mathsf{ide}, \mathsf{idp}, r_c, c$ and then is able to continue the $\mathsf{Purchase}$ 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. Leading to,

$$|\Pr[W_0] - \Pr[W_1]| \leq \mathsf{Adv}_{\mathcal{A},\mathsf{E}}^{\text{IND-CPA}}(1^{\lambda}). \tag{11}$$

Game G^2. The second game replaces the encryption of tk under the key $\mathsf{pk}_{\mathcal{D}}^{\mathsf{E}}$ during a call to $\mathsf{ORefund}$ by a random element sampled uniformly from $[\mathsf{Enc}_{\mathsf{pk}_{\mathcal{D}}^{\mathsf{E}}}]$. This is the same argument as before, we directly conclude to

$$|\Pr[W_1] - \Pr[W_2]| \leq \mathsf{Adv}_{\mathcal{A},\mathsf{E}}^{\text{IND-CPA}}(1^{\lambda}). \tag{12}$$

Game G^3. The third game replaces the encryption of tk under the key $\mathsf{pk}_{\mathcal{T}}^{\mathsf{E}}$ during a call to $\mathsf{OTransfer}$ by a random element sampled uniformly from $[\mathsf{Enc}_{\mathsf{pk}_{\mathcal{D}}^{\mathsf{E}}}]$. As seen in game in G^1, we have

$$|\Pr[W_2] - \Pr[W_3]| \leq \mathsf{Adv}_{\mathcal{A},\mathsf{E}}^{\text{IND-CPA}}(1^{\lambda}). \tag{13}$$

Game G^4. In the last game, concluding this proof, we replace the encryption of $\mathsf{pk}'_{\mathcal{U}}, \mathsf{tk}, \mathit{cert}'_{\mathcal{U}}$ under $\mathsf{pk}_{\mathcal{V}}^{\mathsf{E}}$ by a random element sampled uniformly from $[\mathsf{Enc}_{\mathsf{pk}_{\mathcal{D}}^{\mathsf{E}}}]$. Still using the indishtinguishability argument we have

$$|\Pr[W_3] - \Pr[W_4]| \leq \mathsf{Adv}_{\mathcal{A},\mathsf{E}}^{\text{IND-CPA}}(1^{\lambda}). \tag{14}$$

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

$$\Pr[W_4] = 2^{-\lambda} \tag{15}$$

From equations (10) to (15), we observe that:

$$\mathsf{Adv}_{\mathcal{A}}^{\mathsf{PRIV}}(1^\lambda) = \Pr[\mathsf{Exp}_{\mathcal{A}}^{\mathsf{PRIV}}(1^\lambda) \to 1] \leq 4 \cdot \mathsf{Adv}_{\mathcal{A},\mathsf{E}}^{\text{IND-CPA}} + 2^{-\lambda}.$$

Theorem 5. Applause *provides* no-double-spending *unconditionally.*

Proof. Let $\mathcal{A}$ be a PPT adversary. We show that the probability $\Pr[\mathsf{Exp}_{\mathcal{A}}^{\mathsf{DS}}(1^\lambda) \to 1]$ is negligible. In the game $\mathsf{Exp}_{\mathcal{A}}^{\mathsf{DS}}$, $\mathcal{A}$ 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 $\mathcal{C}$ 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 analysing the update of the state through the algorithms.

Refund. 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.

Transfer. The ticket transfer holds on the same principle. When transfering tk, algorithm CheckTk is executed, ensuring that tk is not already contained in *st*.

Validate. Exactly as the previous algorithms, 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 $\mathcal{C}$ and cannot be modified by $\mathcal{A}$, we can directly conclude that for every PPT adversary, $\Pr[\mathsf{Exp}_{\mathcal{A}}^{\mathsf{DS}}(1^\lambda) \to 1]$ is zero, meaning that the double-spending, in any sequence, is prevented.

Theorem 6. *Instantiated with an anonymous payment having the property of randomisable keys, then for any PPT $\mathcal{A}$,* Applause *provides* unlinkability *under the probability* $|\Pr[\mathsf{Exp}_{\mathcal{A}}^{\mathsf{UNL}}(\lambda) \to 1] - \frac{1}{2}| \leq \frac{3}{2} \cdot \mathsf{Adv}_{\mathcal{A}}^{\mathsf{PayUnl}}(1^\lambda)$.

An adversary $\mathcal{A}$ wins the unlinkability experiment $\mathsf{Exp}_{\mathcal{A}}^{\mathsf{UNL}}$ if $\mathcal{A}$ is able to link a ticket purchased by an user $\mathcal{U}_b$ among users $\mathcal{U}_0$ and $\mathcal{U}_1$. At the beginning of the experiment, both $\mathcal{U}_0$ and $\mathcal{U}_1$ purchase a ticket for places specified by $\mathcal{A}$. Then, $\mathcal{U}_b$ purchase a place where b is randomly chosen by the challenger $\mathcal{C}$. Then, $\mathcal{A}$

specifies the algorithm to execute among Refund, $\mathsf{Transfer}$ and $\mathsf{Validate}$. Then $\mathcal{U}_0$, $\mathcal{U}_1$ and $\mathcal{U}_b$ executes the algorithm with their purchased ticket. Finally, $\mathcal{A}$ is required to return a bit b^*, used by the challenger $\mathcal{C}$ returning 1 if $b = b^*$ and 0 otherwise. We let $\mathcal{A}$ to create the environment of its choice using oracles. We stress that $\mathcal{A}$ cannot corrupt $\mathcal{U}_0$ and $\mathcal{U}_1$ to avoid trivial attack.

Recall that in our construction, a ticket tk is defined by a the place $(\mathsf{ide}, \mathsf{idp})$, 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. 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 $\mathcal{C}$ 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-negligibly 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 $\mathcal{C}$.

Game G^0. The initial games 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

$$\Pr[W_0] = \Pr[\mathsf{Exp}^{\mathsf{UNL}}_{\mathcal{A},b}(1^\lambda) \to 1] \tag{16}$$

Game G^1. We focus on the $\mathsf{Validate}$ 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 indishtinguishability 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 indishtinguishability of the randomised keys. Therefore, we have

$$\Pr[W_0] = \Pr[W_1] \tag{17}$$

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

$$\Pr[W_1] - \Pr[W_2] \leq \mathsf{Adv}^{\mathsf{PayUnl}}_{\mathcal{A}} \tag{18}$$

Game G^3. We focus on $\mathsf{Transfer}$, 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 indishtinguishability of the randomsied 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, we have

$$\Pr[W_2] - \Pr[W_3] \leq \mathsf{Adv}^{\mathsf{PayUnl}}_{\mathcal{A}} \tag{19}$$

Game G^4. In the last game, we focus on $\mathsf{Purchase}$, 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

$$\Pr[W_3] - \Pr[W_4] \leq \mathsf{Adv}^{\mathsf{PayUnl}}_{\mathcal{A}} \tag{20}$$

Remark that in G^4, we have replaced the key pair $(\mathsf{sk}_{\mathcal{U}_b}, \mathsf{pk}_{\mathcal{U}_b})$ with $(\mathsf{sk}_{\mathcal{U}_{1-b}}, \mathsf{pk}_{\mathcal{U}_{1-b}})$, 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

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

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. Therefore, we have only to modify the proofs for *Unforgeability* and *Privacy.*

Theorem 8. *Instantiated with an* EUF-CMA *signature, and assuming the Discrete Logarithm (DL) problem,* Applause *provides* unforgeability *with the probability* $\Pr[\mathsf{Exp}^{\mathsf{UF}}_{\mathcal{A}}(1^\lambda) \to 1] \leq 2\cdot\mathsf{Adv}^{\mathrm{EUF\text{-}CMA}}_{\mathcal{A}}(1^\lambda) + \mathsf{Adv}^{DL}_{\mathcal{A}}(1^\lambda)$ *for all* PPT *algorithm* $\mathcal{A}$.

Proof. We start with the same sequence of games as for the proof of Theorem 2. We consider G^2, where the challenger keeps record of the signatures produced by $\mathcal{D}$ and $\mathcal{T}$, and refuses any signature under $\mathsf{pk}_{\mathcal{D}}^{\mathsf{S}}$ or $\mathsf{pk}_{\mathcal{T}}^{\mathsf{S}}$ that it does not produce.

During the execution of the algorithm Alg chosen by $\mathcal{A}$ the $\mathsf{CheckTk}$ 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 $\mathcal{C}$, $\mathcal{A}$ 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 against 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

$$\Pr[W_2] \leq \mathsf{Adv}_{\mathcal{A}}^{\mathsf{DL}}(1^\lambda). \tag{21}$$

And based on the previous reduction, we observe that

$$\Pr[\mathsf{Exp}_{\mathcal{A}}^{\mathsf{UF}}(1^\lambda) \to 1] \leq 2 \cdot \mathsf{Adv}_{\mathcal{A}}^{\text{EUF-CMA}}(1^\lambda) + \mathsf{Adv}_{\mathcal{A}}^{\mathsf{DL}}(1^\lambda). \tag{22}$$

Theorem 9. *Instantiated with an* IND-CPA *encryption scheme, and assuming the Discrete Logarithm (DL) problem,* $\mathsf{Applause}$ *provides* privacy *with the probability* $\Pr[\mathsf{Exp}_{\mathcal{A}}^{\mathsf{PRIV}}(\lambda) \to 1] \leq \frac{1}{q} + 4 \cdot \mathsf{Adv}_{\mathcal{A}}^{\text{IND-CPA}}(1^\lambda) + \mathsf{Adv}_{\mathcal{A}}^{\mathit{DL}}(1^\lambda)$*, for all* PPT *adversary* $\mathcal{A}$.

Proof. The same sequence of game as in the proof of Theorem 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 $\mathcal{C}$ 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:

$$\Pr[\mathsf{Exp}_{\mathcal{A}}^{\mathsf{PRIV}}(\lambda) \to 1] \leq \frac{1}{q} + 4 \cdot \mathsf{Adv}_{\mathcal{A}}^{\text{IND-CPA}}(1^\lambda) + \mathsf{Adv}_{\mathcal{A}}^{\mathsf{DL}}(1^\lambda).$$

D Description of Spotlight

In this section, we formally present Spotlight, the auditable version of Applause. We have chosen to present it in 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.

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

$\underline{\mathsf{JKeyGen}(1^\lambda)}$: Runs $\mathsf{OKeyGen}(1^\lambda) \to (\mathsf{sk}_{\mathcal{J}}, \mathsf{pk}_{\mathcal{J}})$ and outupts the key pair.

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

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

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

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

$\underline{\mathsf{Purchase}\langle \mathcal{U}(\mathsf{sk}_{\mathcal{U}}, (\mathsf{ide}, \mathsf{idp})), \mathcal{D}(\mathsf{sk}_{\mathcal{D}}, st)\rangle}$: $\mathcal{U}$ updates its signature's keys using the randomisation algorithm $\mathsf{RandID}(\mathsf{sk}_{\mathcal{U}}^{\mathsf{S}}, \mathsf{pk}_{\mathcal{U}}^{\mathsf{S}}, cert_{\mathcal{U}}) \to (\mathsf{sk}_{\mathcal{U}}^{\mathsf{S}'}, \mathsf{pk}_{\mathcal{U}}^{\mathsf{S}'}, cert'_{\mathcal{U}})$. It samples $r_c \xleftarrow{\$} \{0,1\}^\lambda$. Then it sends $\mathsf{pk}'_{\mathcal{U}} = (\mathsf{pk}_{\mathcal{U}}^{\mathsf{S}'}, \mathsf{pk}_{\mathcal{U}}^{\mathsf{E}'})$, $cert'_{\mathcal{U}}$, $\psi = \mathsf{Enc}_{\mathsf{pk}_{\mathcal{D}}^{\mathsf{E}}}(\mathsf{ide}, \mathsf{idp}, r_c)$ and $\sigma_\psi = \mathsf{Sign}_{\mathsf{sk}_{\mathcal{U}}^{\mathsf{S}'}}(\psi)$. $\mathcal{D}$ starts by checking that the pair $(\mathsf{idp}, \mathsf{ide})$ 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 $\mathsf{Verif}_{\mathsf{pk}_{\mathcal{U}}^{\mathsf{S}}}(\psi, \sigma_\psi)$. $\mathcal{D}$ updates its state $st \leftarrow st \cup \{c, cert'_{\mathcal{U}}, 0\}$ signs $\sigma_c \leftarrow \mathsf{Sign}_{\mathsf{sk}_{\mathcal{D}}^{\mathsf{S}}}(c)$, before sending σ_c to $\mathcal{U}$. Once the signature verified, $\mathcal{U}$ proceeds to the payment with $\mathsf{Pay}\langle \mathcal{U}(\mathsf{sk}_{\mathcal{U}}^{\mathsf{P}}), \mathcal{D}(\mathsf{sk}_{\mathcal{D}}^{\mathsf{P}})\rangle \to \mathcal{U}(b), \mathcal{D}(b)$. 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 $st \leftarrow st \cup \{c, cert'_{\mathcal{U}}, 1\}$.

$\underline{\mathsf{CheckTk}(\mathsf{tk}, st, cert)}$: 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 $\mathsf{Verif}_{\mathsf{pk}_{\mathcal{D}}^{\mathsf{S}}}(c, \sigma_c)$ or $\mathsf{Verif}_{\mathsf{pk}_{\mathcal{T}}^{\mathsf{S}}}(c, \sigma_c)$ validate. Also verify the certificat 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.

$\underline{\mathsf{Refund}\langle\mathcal{U}(\mathsf{sk}_{\mathcal{U}},\mathsf{tk}),\mathcal{D}(\mathsf{sk}_{\mathcal{D}},st)\rangle}$: $\mathcal{U}$ executes $\mathsf{RandID}(\mathsf{sk}^{\mathsf{S}}_{\mathcal{U}},\mathsf{pk}^{\mathsf{S}}_{\mathcal{U}},cert_{\mathcal{U}})\to(\mathsf{sk}^{\mathsf{S}'}_{\mathcal{U}},\mathsf{pk}^{\mathsf{S}'}_{\mathcal{U}},cert'_{\mathcal{U}})$ 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 validity of the received ticket using $\mathsf{CheckTk}(\mathsf{tk},st,cert'_{\mathcal{U}})\to st$.
Both participants starts a refund (*i.e.,* a payment with a negative amount) using $\mathsf{Pay}\langle\mathcal{U}(\mathsf{sk}^{\mathsf{P}}_{\mathcal{U}}),\mathcal{D}(\mathsf{sk}^{\mathsf{P}}_{\mathcal{D}})\rangle\to\mathcal{U}(b),\mathcal{D}(b)$. 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.

$\underline{\mathsf{Transfer}\langle\mathcal{U}_1(\mathsf{sk}'_{\mathcal{U}_1},\mathsf{tk}_1,\mathsf{pk}'_{\mathcal{U}_2}),\mathcal{T}(\mathsf{sk}_{\mathcal{T}},st),\mathcal{U}_2(\mathsf{sk}'_{\mathcal{U}_2},\mathsf{pk}'_{\mathcal{U}_1},(\mathsf{ide},\mathsf{idp}))\rangle}$: $\mathcal{U}_1$ computes the signature $\mathsf{Sign}_{\mathsf{sk}^{\mathsf{S}'}_{\mathcal{U}_1}}(\mathsf{pk}'_{\mathcal{U}_2},c)\to\sigma_{T,1}$ 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 $\mathsf{Sign}_{\mathsf{sk}^{\mathsf{S}'}_{\mathcal{U}_2}}(\mathsf{pk}'_{\mathcal{U}_1})\to\sigma_{T,2}$ and sends $(\mathsf{pk}'_{\mathcal{U}_2},cert'_{\mathcal{U}_2},\sigma_{T,2})$ to $\mathcal{T}$. Once it received $\sigma_{T,1}$ and $\sigma_{T,2}$, $\mathcal{T}$ checks $\mathsf{Verif}_{\mathsf{pk}^{\mathsf{S}'}_{\mathcal{U}_1}}(\mathsf{pk}'_{\mathcal{U}_2},c,\sigma_{T,1})$ and $\mathsf{Verif}_{\mathsf{pk}^{\mathsf{S}'}_{\mathcal{U}_2}}(\mathsf{pk}'_{\mathcal{U}_1},c,\sigma_{T,2})$, 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 $\mathsf{CheckTk}(\mathsf{tk},st,cert'_{\mathcal{U}_1})\to st$ the ticket verification algorithm. If st was updated, $\mathcal{D}$ 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 $(\mathsf{ide},\mathsf{idp})$ by executing $\mathsf{Purchase}\langle\mathcal{U}_2(\mathsf{sk}_{\mathcal{U}_2},(\mathsf{ide},\mathsf{idp})),\mathcal{T}(\mathsf{sk}_{\mathcal{T}},st)\rangle$ resulting to $(\mathcal{U}_2(b_p,\mathsf{tk}_2),\mathcal{T}(b_p,st))$ where $\mathcal{T}$ does not check that $(\mathsf{ide},\mathsf{idp})$ 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 $\mathsf{CheckTk}$ 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 $\mathcal{U}_1$. $\mathsf{Refund}\langle\mathcal{U}_1(\mathsf{sk}_{\mathcal{U}_1},\mathsf{tk}_1),\mathcal{T}(\mathsf{sk}_{\mathcal{T}},st\setminus\{c\})\rangle\to\mathcal{U}_1(b_r),\mathcal{T}(b_r)$ is executed in 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.

$\underline{\mathsf{Validate}\langle\mathcal{U}(\mathsf{sk}_{\mathcal{U}},\mathsf{tk}),\mathcal{V}(\mathsf{sk}_{\mathcal{V}},st)\rangle}$: $\mathcal{V}$ starts by providing $\mathsf{pk}_{\mathcal{V}}$ to $\mathcal{U}$. The latter parses it and runs $\mathsf{Verif}_{\mathsf{pk}^{\mathsf{S}}_{\mathcal{D}}}(\mathsf{pk}_{\mathcal{V}},cert_{\mathcal{V}})\to b$. On b equals 0, the protocol is aborted. Otherwise, $\mathcal{U}$ executes $\mathsf{RandID}(\mathsf{sk}^{\mathsf{S}}_{\mathcal{U}},\mathsf{pk}^{\mathsf{S}}_{\mathcal{U}},cert_{\mathcal{U}})\to(\mathsf{sk}^{\mathsf{S}'}_{\mathcal{U}},\mathsf{pk}^{\mathsf{S}'}_{\mathcal{U}},cert'_{\mathcal{U}})$ 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 pases, $\mathcal{V}$ creates a challenge by sampling a random 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 $\mathsf{Verif}_{\mathsf{pk}^{\mathsf{S}}_{\mathcal{V}}}(\epsilon,\sigma_s)$. 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 $s=s'$.

$\underline{\mathsf{Audit}(\mathsf{sk}_{\mathcal{J}},(\cdot,\cdot,cert))}$: Executes $\mathsf{Audit}(\mathsf{sk}_{\mathcal{J}},cert)\to\mathsf{pk}_{\mathcal{U}}$ and returns $\mathsf{pk}_{\mathcal{U}}$.

E 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 8, 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.

Theorem 10. *Instantiated with statistically indistinguishable randomisable keys, and an anonymous payment scheme,* $\mathsf{Spotlight}$ *provides* pseudonymity *under probability* $|\Pr[\mathsf{Exp}_{\mathcal{A}}^{\mathsf{PSE}}(1^\lambda) \rightarrow 1] - \frac{1}{2}| \leq \mathsf{Adv}_{\mathcal{A}}^{\mathsf{PayPse}}(1^\lambda)$*, for every* PPT $\mathcal{A}$*.*

Proof. 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 Theorem 10. Let us denote it as Game G^1.

We have assumed statistical randomisation of the certificats of $\mathcal{U}_0$ with key pair $(\mathsf{sk}_{\mathcal{U}_0}, \mathsf{pk}_{\mathcal{U}_0})$ and $\mathcal{U}_1$ with key pair $(\mathsf{sk}_{\mathcal{U}_1}, \mathsf{pk}_{\mathcal{U}_1})$ throught 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 $\mathsf{Exp}_{\mathcal{A}}^{\mathsf{PSE}}$ 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 $\mathsf{Applause}$. We precise an upper bound for the advantages of an adversary:

$$|\Pr[\mathsf{Exp}_{\mathcal{A}}^{\mathsf{PSE}}(1^\lambda) \rightarrow 1] - \frac{1}{2}| \leq \mathsf{Adv}_{\mathcal{A}}^{\mathsf{PayPse}}(1^\lambda).$$

Theorem 11. *Instantiated with an anonymous payment and using randomisable keys, then for any PPT* $\mathcal{A}$*,* $\mathsf{Spotlight}$ *provides* unlikability *under the probability* $|\Pr[\mathsf{Exp}_{\mathcal{A}}^{\mathsf{UNL}}(\lambda) \rightarrow 1] - \frac{1}{2}| \leq 3 \cdot \mathsf{Adv}_{\mathcal{A}}^{\mathsf{PayUnl}}(1^\lambda)$*.*

Proof. The challenger executes the actions of the judge, hence, the adversary is not able to recover the user's identity based on the judge's keys $(\mathsf{sk}_{\mathcal{J}}, \mathsf{pk}_{\mathcal{J}})$. Only an additional certificate shared with the adversary during $\mathsf{Purchase}$, Refund or $\mathsf{Transfer}$ when called by the adversary through the oracles. Even so, this is linked to the user's public key, that we have assumed statistically indistinguishable randomisation, leading to impossibility of the recovery of the original identiy of the user. Then, we conclude to the same advantage as for Theorem 11 :

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

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