Private Mobile Pay-TV From Priced Oblivious Transfer

Wouter Biesmans, Josep Balasch, Alfredo Rial, Bart Preneel, Member, IEEE, Ingrid Verbauwhede, Member, IEEE

Abstract—In pay-TV, a service provider offers TV programs and channels to users. To ensure that only authorized users gain access, conditional access systems (CAS) have been proposed. In existing CAS, users disclose to the service provider the TV programs and channels they purchase. We propose a pay-per-view and a pay-per-channel CAS that protect users’ privacy. Our pay-per-view CAS employs priced oblivious transfer (POT) to allow a user to purchase TV programs without disclosing which programs were bought to the service provider. In our pay-per-channel CAS, POT is employed together with broadcast attribute-based encryption (BABE) to achieve low storage overhead, collusion resistance, efficient revocation and broadcast efficiency. We propose a new POT scheme and show its feasibility by implementing and testing our CAS on a representative mobile platform.

Index Terms—ADP-PPRO, ADP-APPS, APC-MMED

1 INTRODUCTION

In pay-TV, a service provider offers TV programs and channels to users. To ensure that only authorized users gain access, conditional access systems (CAS) have been proposed [1], [2]. CAS can implement pay-per-channel, where users subscribe to channels or groups of channels for a period of time, or pay-per-view, where users pay for individual TV programs. In flexible pay-per-channel [3], users can subscribe to any combination of channels and change their subscription during the subscription period.

A CAS should provide some functionalities and security properties [4]. Fine-grained access control is required to ensure that only authorized users can get access. For example, it should be possible to apply access control policies that deny access to underage users. As in any commercial transaction, non-repudiation is necessary to solve disputes. Backward secrecy ensures that revoked users cannot get access. Additionally, a CAS should be efficient, which involves efficient transmission, low storage overhead, and efficient key redistribution when users are revoked or change their subscription.

Traditional CAS can roughly be divided into two classes: group key based symmetric schemes and public key based schemes. In group key based schemes [1], [3], [5], [6], [7], users in the same group share a group key. To transmit a video to users in different groups, the service provider determines which group keys should be used and encrypts the video employing those keys. Group key based schemes offer efficient broadcast transmission. However, they suffer collusion attacks, lack of non-repudiation, and inefficient key distribution [4]. Public key based schemes [8], [9] offer non-repudiation, lighter storage overhead and efficient key redistribution. However, transmission is inefficient and fine-grained access control is difficult to implement.

Recent CAS which enjoy both efficient broadcast transmission and the security guarantees of public key based schemes have been proposed for pay-TV [10] and mobile pay-TV [4]. They are based on attribute-based encryption (ABE) [11], [12]. The scheme in [4] is compatible with the digital video broadcast standard DVB-H1 and DVB-SH2. However, as explained in Section 2, these CAS have shortcomings.

To the best of our knowledge, user privacy has not been considered in existing CAS. Currently, in pay-per-channel, users disclose to the service provider the channels to which they subscribe, and in pay-per-view, they disclose the TV programs they purchase. This may reveal users’ sensitive information, such as political views, religious beliefs and sexual orientation.

Our Contribution. We propose a pay-per-view and a pay-per-channel CAS that protect users’ privacy. Both CAS employ priced oblivious transfer (POT) [13] in order to allow the user to purchase channels and programs without disclosing which channels and programs were bought to the service provider. POT employs a prepayment mechanism, where the user makes an initial deposit and, at each purchase, subtracts the price paid from the deposit without the service provider learning the price paid or the new value of the deposit.

In our pay-per-channel CAS, POT is employed together with broadcast attribute-based encryption (BABE) [14], which allows our CAS to enjoy low storage overhead, collusion resistance, efficient revocation and broadcast efficiency. With respect to the CAS in [4], our CAS employs an efficient direct revocation method in which non-revoked users do

1. http://www.dvb.org/standards/dvb-h
not perform any operation. With respect to the CAS in [10], our CAS avoids the use of an ad-hoc revocation method based on an expiration time attribute, which, as explained in Section 2, requires a large ciphertext size.

To show the feasibility of our approach, we propose a new POT scheme. Our POT scheme is similar to the scheme in [15], but replaces the signature scheme employed there by a structure-preserving signature scheme [16], which allows to shorten the ciphertext size. We implement both our pay-per-view and a pay-per-channel CAS and show performance measurements in an ARM Cortex-A8 processor core. Our results demonstrate that our approach can be deployed in practical settings even when using platforms from early-generation mobile devices.

Outline of the paper. In Section 2, we describe and analyze related work. In Section 3, we describe our model for CAS and its properties. In Section 4, we describe the concepts of priced oblivious transfer and broadcast attribute-based encryption. We describe our pay-per-channel and our pay-per-view CAS in Section 5 and Section 6 respectively. We discuss their properties in Section 7. In Section 8, we propose a new priced oblivious transfer scheme. In Section 9, we analyze the efficiency of our CAS. We describe an implementation and performance measurements of both BABE and POT schemes. We conclude in Section 10.

2 Related Work

The group key based CAS described in the DVB standard\(^3\) consists of four layers: registration, rights management, key stream and traffic encryption. In the registration layer, the user’s device is given several decryption keys for authentication and broadcast decryption purposes. In the rights management layer, the service provider determines the inferred encryption key (IEK) that is used to encrypt the service encryption key (SEK). The choice of IEK depends on which devices should receive the SEK, i.e., on which users have subscribed for a particular channel. Zero message broadcast encryption [17] is employed to encrypt the right object (RO), which contains the SEK, on input the IEK. In the key stream layer, the SEK is used to encrypt the traffic encryption key TEK. In the traffic encryption layer, the TEK is used to encrypt the video content. The TEK is updated frequently, e.g., every second.

The DVB CAS, as well as other group key based CAS [1], [3], [5], [6], [7], suffers from lack of non-repudiation and inefficient key distribution. Furthermore, for security reasons, the size of the groups should be small, thus reducing the advantage of broadcast encryption efficiency. Public key based CAS [8], [9] employ inefficient one to one transmission and hinder fine-grained access control.

The CAS proposed in [4] is based on the key-policy attribute-based encryption scheme (KP-ABE) in [12]. The key stream and traffic encryption layers remain unchanged from the DVB standard. During registration, each user obtains a KP-ABE key that binds her to a subscribers category and to other attributes, such as her age. The rights management layer works as follows. To encrypt the RO, the service provider employs KP-ABE. Each KP-ABE ciphertext is associated to a category and to an age rating, such that only users with the right category and age can decrypt it and get the RO. In order to revoke a user, a mechanism in which all the users but the revoked user update their KP-ABE keys is proposed, which is inefficient.

The CAS proposed in [10] is based on ciphertext policy attribute set based encryption (ASBE). The key stream and traffic encryption layers remain unchanged from the DVB standard. During registration, each user obtains an ASBE key for attributes of the form \((\text{channel}, \text{start}, \text{end})\), where \(\text{channel}\) identifies a channel, and \(\text{start}\) and \(\text{end}\) denote the start and end dates of the subscription. In ASBE, the attributes have a hierarchy, so that the attributes \(\text{start}\) and \(\text{end}\) of a channel cannot be used for other channels. In the rights management layer, to encrypt the RO, the user employs a policy \((\text{channel}, \leq \text{starttime}, \geq \text{endtime})\), i.e., the \(\text{start}\) and \(\text{end}\) attributes of the user should be lower and greater than the \(\text{starttime}\) and \(\text{endtime}\) attributes in the policy. This mechanism is inefficient and also needs key updates.

The problems of the CAS in [4] and [10] can be solved by using attribute-based encryption with revocation. An ABE with revocation scheme was first proposed in [18]. As described in [19], there are two main revocation methods: direct and indirect. Indirect revocation does not require the senders to know the revocation list, but it requires the keys of non-revoked users to be updated. Direct revocation requires the sender to know the revocation list but no key updates are needed. Therefore, direct revocation is more suitable for a CAS, where the service provider is the only sender and already knows the revocation list. We employ the ABE with direct revocation proposed in [14] in our CAS.

We summarize now recent progress in ABE applicable to the construction of CAS. In [20], a more efficient ABE with direct revocation scheme is proposed. The ABE with direct revocation schemes in [14] and [20] are selectively secure but, in [21], a fully-secure one is presented. In [22], a forward-secure ABE scheme is shown. This scheme gives protection against key exposure and is adequate for settings where the provider does not know the revocation list. Otherwise, ABE with direct revocation is better suited because it avoids key updates. In [23] (resp. [24]), an ABE scheme with white-box (resp. black-box) traceability is presented, which allows one to identify users that leak their secret keys (resp. their decryption box).

Other recent works on CAS include an authentication and subscription protocol between the provider and users [25] and a CAS for portable devices based on selective encryption that provides interoperability between providers [26].

To the best of our knowledge, there is no CAS that protects user privacy. POT was proposed in [13]. Existing POT schemes offer different security levels: half-simulation security [13], full-simulation security [27], and universal composability [15]. We propose a new universally composable POT scheme that shortens the ciphertext size of the scheme in [15].

Oblivious transfer with access control (OTAC) [28], [29] allows the service provider to enforce access control policies in OT protocols. Some OTAC schemes are based on combining OT with ABE [30], [31]. Although POT can be seen as a form of oblivious transfer with access control, OTAC
schemes do not allow for an efficient implementation of POT. The reason is that, in OTAC schemes, a user possesses some attributes that are certified only once by a credential issuer, while in POT, the user account is an attribute that changes each time the user makes a purchase in a way that depends on the item purchased. Therefore, existing OTAC schemes cannot be applied without modification to construct a privacy-preserving CAS. We note that some OTAC schemes also hide the access control policy from the user [32], but this is not necessary in our case since the price of a channel or program should be public.

3 Definition of Our Conditional Access System

3.1 Pay-Per-Channel CAS

Our pay-per-channel conditional access system (PPC-CAS) is run by a provider \( P \) and users \( U \) and consists of a registration layer, a rights management layer, a key stream layer and a traffic encryption layer, which we describe in Figure 1. The last two layers work as in the DVB standard, which we summarize in Section 2.

In the registration layer, at setup, the provider creates public parameters and a master secret key. After that, a user and the provider run a protocol by means of which the user obtains right objects that contain parts of a secret key associated with the user’s identity and attributes. The user attributes are TV channels for which the user subscribes. The user obtains her secret key without revealing to the provider the TV channels she subscribes for, while still paying the right price for the subscription. Each right object is associated with an access control policy that describes a rating (i.e., an age restriction) for a TV channel, so that a user cannot subscribe for a channel whose rating is not adequate for the user’s age.

In the rights management layer, the provider receives as input a SEK, an attribute associated with a TV channel and a revocation list. The provider encrypts the SEK under the identities of non-revoked subscribers and under the attribute.

A PPC-CAS should fulfill the following security properties.

Privacy. No coalition of parties, which can include the provider, must be able to learn the TV channels for which a user subscribes.

Revocation. \( P \) must have the ability to end the subscription of a user when it expires or when it must be terminated because of misbehavior. A revoked user must not be able to access channels for which her subscription is revoked.

Collusion Resistance. A group of colluding users (which could include revoked users) must not be able to access channels that none of them is able to access individually.

Non repudiation. Non repudiation is an important property when a dispute between \( P \) and \( U \) occurs. It requires that \( P \) should be able to prove that \( U \) made a subscription if \( U \) did make the subscription, while \( U \) should be able to prove that she has made the corresponding payment if \( U \) did make the payment. When there is a dispute between \( P \) and \( U \), they can ask a trusted adjudicator to resolve it.

These security properties must hold against an active adversary. The adversary controls the network and is able to eavesdrop, delay and modify messages exchanged between parties. In addition, the adversary is allowed to corrupt parties and control their behavior.

Throughout this paper, we use several abbreviations. They are summarized in Table 1.

3.2 Pay-Per-View CAS

Our pay-per-view conditional access system (PPV-CAS) is run by a provider \( P \) and users \( U \) and consists of a rights management layer, a key stream layer and a traffic encryption layer, which we describe in Figure 2. The last two layers work as in the DVB standard, which we summarize in Section 2.

In the rights management layer, the provider gets as input, for each TV program, a right object RO and an access control policy that describes the age rating. The user inputs as attributes its age and a TV program of her choice. The provider and the user run a protocol by means of which
the user decrypts the right object associated with the TV program of her choice without revealing the TV program to the provider. A right object contains the SEK needed to decrypt a TV program.

A PPV-CAS should fulfill the properties of privacy, collusion-resistance, and non-repudiation defined in Section 3.1. In our PPV-CAS, the SEK is only valid for the duration of a TV program and there are no long term keys, so revocation is less necessary.

4 Building Blocks

4.1 Broadcast Attribute-Based Encryption

Let \( U \) be the set of all user indexes and \( S \) be the set of all attributes. Let \( Q \) be the set of all policies that are allowed over \( S \). We denote by \( \mathcal{A} \vdash \mathcal{P} \) the fact that a set of attributes \( \mathcal{A} \) satisfies the policy \( \mathcal{P} \).

We use broadcast ciphertext policy attribute based encryption (BABE) [14], where the secret key of a user is associated with a user index \( id \) and to a set of attributes \( \mathcal{A} \), and a ciphertext is associated to a set of user indexes \( I \) and to a policy \( \mathcal{P} \). A BABE scheme consists of the following algorithms:

- **Setup(1^k)**. On input the security parameter \( 1^k \), output the public parameters \( \text{par} \) and a master secret key \( \text{msk} \).
- **KeyGen(par, msk, id, A)**. On input the parameters \( \text{par} \), the master secret key \( \text{msk} \), the user index \( id \) ∈ \( U \) and the set of attributes \( \mathcal{A} \in \mathcal{S} \), output a secret key \( sk(id, A) \) for the user index \( id \) and the attributes \( \mathcal{A} \).
- **Enc(par, I, P, m)**. On input the parameters \( \text{par} \), the user index set \( I \in U \), the policy \( \mathcal{P} \in Q \), and the message \( m \), output a ciphertext \( ct(I, P) \).
- **Dec(par, sk(id, A), ct(I, P))**. On input the parameters \( \text{par} \), a secret key \( sk(id, P) \) and a ciphertext \( ct(I, P) \), output a message \( m \) if \( id \in I \) and if \( \mathcal{A} \vdash \mathcal{P} \).

In the rights management layer of our pay-per-channel CAS, users are assigned a user identifier \( id \), and the set of attributes of a user will be the set of TV channels to which the user subscribes. When a user subscribes to a new TV channel, it must be possible to add the associated TV channel attribute to an existing secret key. For this purpose, we split up the algorithm KeyGen into two algorithms KeyGen_0 and KeyGen_1.

- **KeyGen_0(par, msk, id)**. On input the parameters \( \text{par} \), the master secret key \( \text{msk} \), and the user index \( id \in U \), output a secret key part \( sk(id) \) for the user index \( id \) and auxiliary information \( aux \).
- **KeyGen_1(aux, s)**. On input the auxiliary information \( aux \) and the attribute \( s \), output a secret key part \( sk(s) \).

Therefore, a secret key for the user index \( id \in U \) and the set of attributes \( \mathcal{A} \in \mathcal{S} \) is computed by first running \((sk(id), aux) \leftarrow \text{KeyGen}_0(\text{par}, \text{msk}, id)\) and, for each \( s \in \mathcal{A} \), \((sk(s) \leftarrow \text{KeyGen}_1(aux, s)) \), and then setting \((sk(id), (sk(s)))_{s \in \mathcal{A}}\). The algorithm KeyGen of the broadcast ciphertext policy attribute based encryption in [14] can be split up into KeyGen_0 and KeyGen_1. We summarize the symbols used for BABE in Table 2 (left).

4.2 Priced Oblivious Transfer

Priced oblivious transfer (POT) is a two-party protocol that provides privacy in e-commerce of digital goods by hiding from the vendor which items are bought. More formally, a provider \( \mathcal{P} \) sells a set of messages \( \{m_1, \ldots, m_N\} \) with prices \( \{p_1, \ldots, p_N\} \) to a user \( \mathcal{U} \). At each purchase, \( \mathcal{U} \) chooses \( i \in \{1, \ldots, N\} \), gets \( m_i \) and pays \( p_i \). The following security properties must hold.

Provider security. For any message \( m_i \), when \( \mathcal{U} \) buys \( m_i \), \( \mathcal{P} \) learns nothing about \( m_i \) or \( p_i \).

User privacy. For any message \( m_i \), when \( \mathcal{U} \) buys \( m_i \), \( \mathcal{P} \) must not learn any information about messages that \( \mathcal{U} \) has not bought.

A POT scheme consists of an initialization phase, a payment phase, a request phase and a sell phase, and it is parameterized by a message space \( \mathcal{M} \), a number of messages \( N \), a universe of prices \( \mathcal{P} \), and a universe of accounts \( \mathcal{A} \).

The initialization phase consists of the algorithms InitP and InitU.

- **InitP(1^k, m_1, \ldots, m_N, p_N)**. On input the security parameter \( 1^k \), the messages \( \{m_1, \ldots, m_N\} \) and the prices \( \{p_1, \ldots, p_N\} \), output state information \( st_p \) for \( \mathcal{P} \) and a message \( M_{init} \) to be sent to \( \mathcal{U} \).
- **InitU(1^k, M_{init})**. On input the security parameter \( 1^k \) and the message \( M_{init} \), output state information \( st_u \) for \( \mathcal{U} \) and prices \( \{p_1, \ldots, p_N\} \).

The payment phase consists of the algorithms PayP and PayU.

- **PayU(st_u, p)**. On input state information \( st_u \) and payment \( p \), output a payment message \( M_{pay} \) and updated state information \( st_u \).
- **PayP(st_p, M_{pay})**. On input state information \( st_p \) and message \( M_{pay} \), output updated state information \( st_p \) and payment \( p \).

The request phase consists of the algorithms ReqP and ReqU.
• \text{ReqU}(st_u, i)$. On input state information $st_u$ and index $i$, output state information $st_u$ and message $M_{req}$ to be sent to $P$.
• \text{ReqP}(st_p, M_{req})$. On input state information $st_p$ and message $M_{req}$, output updated state information $st_p$.

The request phase consists of the algorithms \text{SellP} and \text{SellU}.
• \text{SellP}(st_p)$. On input state information $st_p$, output updated state information $st_p$ and message $M_{sell}$ to be sent to $U$.
• \text{SellU}(st_u, M_{sell})$. On input state information $st_u$, output updated state information $st_u$ and message $m_u$.

We summarize the symbols used for POT in Table 2 (right).

5 Pay-Per-Channel CAS

We describe a mobile pay-per-channel conditional access system (PPC-CAS). The key stream and traffic encryption layers remain as in the DVB standard. We thus describe a protocol for the registration and rights management layer of our PPC-CAS.

The service encryption key (SEK) of a TV channel is updated periodically, e.g., every week. The user $U$ purchases a subscription to a TV channel by running a priced oblivious transfer protocol (POT) with the provider $P$. The right object $RO_i$ for a TV channel, which the user $U$ receives as output of the POT protocol, contains a key that allows the user to obtain the SEK each time the provider $P$ updates the SEK.

The provider $P$ employs broadcast ciphertext-policy attribute based encryption (BABEL) in order to release updates of the SEK. The reason why $P$ employs BABEL is threefold. First, it is more efficient than purchasing the SEK each time it is updated by employing the POT protocol. Second, BABEL allows for efficient revocation. When a user is revoked, the provider simply removes the user identifier $id$ from the set $I$ used to compute the ciphertext that encrypts the updated SEK. Third, the user secret key of a BABEL scheme can be stored in a tamper-resistant element of the mobile device in order to prevent the user from sharing her keys. This is possible thanks to the fact that the key $sk(id, A)$ can be split up into $(sk(id), sk(s))_{s \in A}$. The key part $sk(id)$, which depends only on the user identifier $id$, can be stored permanently in a tamper-resistant element. The key parts $(sk(s))_{s \in A}$, which are purchased through the POT protocol, are useless without $sk(id)$. We note that the collusion resistance property of the BABEL scheme ensures that users cannot combine their keys to produce a key for more attributes.

The protocol for the registration layer of our PPC-CAS consists of a setup phase, an initialization phase, a payment phase, and a purchase phase. The rights management layer consists of a SEK update phase and a revocation phase. We depict them in Figure 3.

Setup Phase. $P$ runs $(par, msk) \leftarrow \text{Setup}(1^k)$. $P$ keeps $msk$ and publishes $par$.

Initialization Phase. The user $U$ receives as input a certified age $age$. The provider $P$ receives as input a list of right objects $RO_i$. Each right object $RO_i$ contains a channel attribute $s_i$, a price $p_i$ and a rating $rt_i$. The rating indicates the minimum age $age$ a user must have in order to be allowed to subscribe to a TV channel. The interaction between $U$ and $P$ is as follows:
• $P$ assigns an identifier $id$ to the user $U$ and runs $(sk(id), aux) \leftarrow \text{KeyGen}_i(par, msk, id)$. $P$ stores the tuple $(U, id, aux)$ and includes $id$ in a set of user identifiers. $P$ stores the key part $sk(id)$ in a tamper-resistant element of the user mobile device.
• $U$ sends age to $P$ through an authenticated channel.
• $P$ verifies the certified age $age$. We leave open the way the attribute age is verified.
• $P$ creates a list $(RO_1, \ldots, RO_N)$ of right objects such that the attribute age of $U$ fulfills the rating $rt$ of the right object. $P$ retrieves $aux$ from the tuple $(U, id, aux)$. For each $RO_i$, $P$ runs $sk(s_i) \leftarrow \text{KeyGen}_i(aux, s_i)$ and appends $s_i$ to $RO_i$.
• $P$ runs $(st_p, M_{init}) \leftarrow \text{InitP}(1^k, RO_1, p_1, \ldots, RO_N, p_N)$, stores $st_p$ and sends $M_{init}$ to $U$ through an authenticated channel.
• $U$ receives $M_{init}$, runs $(st_u, p_1, \ldots, p_N) \leftarrow \text{InitU}(1^k, M_{init})$ and stores $st_u$.

Payment Phase. The user $U$ receives as input a payment $p$. The provider $P$ receives no input. The interaction between $U$ and $P$ is as follows:
• $U$ runs $(st_u, M_{pay}) \leftarrow \text{PayU}(st_u, p)$, updates the stored $st_u$ and sends $M_{pay}$ to $P$ through an authenticated channel.
• $P$ receives $M_{pay}$ and runs $(st_p, p) \leftarrow \text{PayP}(st_p, M_{pay})$. $P$ updates the stored $st_p$ and verifies that a payment of value $p$ has been received through any existing payment protocol.

Purchase Phase. The user $U$ receives as input an index $i \in [1, N]$. The provider $P$ receives no input. The interaction between $U$ and $P$ is as follows:
• $U$ runs $(st_u, M_{req}) \leftarrow \text{ReqU}(st_u, i)$, stores $st_u$ and sends $M_{req}$ to $P$ through an authenticated channel.
• $P$ receives $M_{req}$, runs $st_p \leftarrow \text{ReqP}(st_p, M_{req})$ and updates the stored $st_p$. $P$ runs $(st_p, M_{sell}) \leftarrow \text{SellP}(st_p)$, updates the stored $st_p$ and sends $M_{sell}$ to $U$ through an authenticated channel.
• $U$ receives $M_{sell}$, runs $(st_u, RO_i) \leftarrow \text{SellU}(st_u, M_{sell})$ and updates the stored $st_u$. $U$ retrieves $sk(s_i)$ from $RO_i$, sets $A \leftarrow A \cup s_i$ and adds $sk(s_i)$ to her secret key $sk(id, A)$.

SEK Update Phase. The provider $P$ receives as input a channel policy $P$ and a service encryption key SEK. The channel policy allows any user to obtain the SEK if the user possesses a secret key for the channel attribute. $P$ and $U$ do the following:
• $P$ retrieves the stored user identifier set $I$ and runs $ct(I, P) \leftarrow \text{Enc}(par, I, P, SEK)$. $P$ broadcasts the ciphertext $ct(I, P)$ to the users.
• A user $U$ receives $ct(I, P)$ and checks whether, for her secret key $sk(id, A)$, $id \in I$ and $A \leftarrow P$. In that case, $U$ runs SEK $\leftarrow \text{Dec}(par, sk(id, A), ct(I, P))$ and outputs SEK.

Revocation Phase. The provider $P$ receives as input the identity $U$ of a user. To revoke this user, $P$ retrieves $id$ from
the stored tuple \((U, id, aux)\) and sets \(\Pi \leftarrow \Pi \setminus id\).

The provider \(P\) establishes subscription periods of, e.g., one year. At the beginning of a new period, \(P\) changes the identifiers of the channel attributes \(s_i\) and the corresponding policies \(\mathcal{P}\) used to encrypt the SEK. Therefore, at the beginning of a new period, users must renew their subscriptions by running the POT protocol to obtain key parts \(sk(s_i)\) for the new channel attributes \(s_i\) in order to be able to obtain the updated SEK’s. We note that the key part \(sk(id)\) does not need to be updated. We also note that a user can subscribe to a channel at any time. If the subscription is not carried out at the beginning of the subscription period, the prices of the channels are adjusted for the remaining of the period.

### 6 Pay-Per-View CAS

In our pay-per-view CAS (PPV-CAS), the key stream and traffic encryption layers remain as in the DVB standard, which we briefly describe in Section 2. A user \(U\) needs only one SEK to decrypt the TEK’s throughout the broadcast. The SEK is also updated every week, but we consider that TV programs last shorter.

Through the rights management layer, a user obtains right objects, which contain the SEK, for TV programs of her choice. The rights management layer uses priced oblivious transfer to let users purchase right objects without revealing the TV programs bought to the provider.

The protocol consists of an initialization phase, a payment phase and a purchase phase. Basically, it consists on executing a POT protocol where the provider sells right objects that contain the SEK of TV programs.

**Initialization Phase.** The user \(U\) receives as input a certified age \(age\). The provider \(P\) receives as input a list of right objects \(RO_i\). Each right object \(RO_i\) contains a program attribute \(s_i\), the SEK, a price \(p_i\) and a rating \(rt_i\). The rating indicates the minimum age \(age\) a user must have in order to be allowed to buy a TV program. The interaction between \(U\) and \(P\) is as follows:

- \(U\) sends \(age\) to \(P\) through an authenticated channel.
- \(P\) verifies the certified age \(age\). We leave open the way the attribute \(age\) is verified.
- \(P\) creates a list \((RO_1, \ldots, RO_N)\) of right objects such that the attribute \(age\) of \(U\) fulfills the rating \(rt\) of the right object. \(P\) runs \((st_p, M_{init}) \leftarrow \text{InitP}(1^k, RO_1, p_1, \ldots, RO_N, p_N)\), stores \(st_p\) and sends \(M_{init}\) to \(U\) through an authenticated channel.
- \(U\) receives \(M_{init}\), runs \((st_u, p_1, \ldots, p_N) \leftarrow \text{InitU}(1^k, M_{init})\) and stores \(st_u\).

**Payment Phase.** The payment phase works as the payment phase of our PPC-CAS in Section 5.

**Purchase Phase.** The user \(U\) receives as input an index \(i \in [1, N]\). The provider \(P\) receives no input. The interaction between \(U\) and \(P\) is as follows:

- \(U\) runs \((st_u, M_{req}) \leftarrow \text{ReqU}(st_u, i)\), stores \(st_u\) and sends \(M_{req}\) to \(P\) through an authenticated channel.
- \(P\) receives \(M_{req}\), runs \((st_p, M_{req}) \leftarrow \text{ReqP}(st_p, M_{req})\) and updates the stored \(st_p\). \(P\) runs \((st_p, M_{sell}) \leftarrow \text{SellP}(st_p)\), updates the stored \(st_p\) and sends \(M_{sell}\) to \(U\) through an authenticated channel.
- \(U\) receives \(M_{sell}\), runs \((st_u, RO_i) \leftarrow \text{SellU}(st_u, M_{sell})\) and updates the stored \(st_u\). \(U\) retrieves the SEK contained in \(RO_i\).

### 7 Discussion of Our CAS

We discuss the properties of our pay-per-view and our pay-per-channel CAS.

**Privacy.** The user privacy property of the priced oblivious transfer scheme guarantees that the service provider does not learn any information on the TV channels for which a user subscribes.

**Revocation.** The BABE scheme employs a direct revocation method. \(P\) associates each user to a user index \(id\). Each user receives a secret key that allows to decrypt ciphertexts that include \(id\). \(P\), who knows the revocation list, encrypts the SEK of each channel on input the list of non-revoked users. A revoked user will not be able to obtain the next SEK for that channel, which is updated weekly. Therefore, our CAS provides backward secrecy.

**Collusion Resistance.** The BABE scheme also ensures that users cannot collude to gain access to channels that they cannot access on their own. The user secret keys in a BABE scheme cannot be combined to obtain a secret key that encompasses the attributes of each of the secret keys. Additionally, by storing the key part \(sk(id)\) in a tamper-resistant element, a user is prevented from sharing her key.

**Non repudiation.** For POT schemes, the problem of non repudiation has been addressed in [33], which provides a method to turn any POT scheme into a POT scheme that offers fair exchange. This method can be applied to our POT scheme, so that, at the end of a purchase, \(P\) gets a signature by \(U\) that acknowledges that the purchase has been made, while \(U\) receives a signature by \(P\) that acknowledges that the payment was received. These signatures should be used in case of a dispute between \(P\) and \(U\).

**Low Storage Overhead.** The storage overhead of our pay-per-channel CAS requires the storage of a BABE secret key...
Fig. 3. Our Pay-Per-Channel CAS.

for a user identifier $id$ and for the TV channels to which the user has subscribed, and of one SEK for each TV channel to which the user has subscribed. Our pay-per-view CAS requires to store one SEK for each purchased TV program.

**Key Redistribution.** Our scheme enjoys efficient key redistribution when a user is revoked. Thanks to the use of a direct revocation method, other users do not need to perform any operation. Users only need to renew their keys at the beginning of each subscription period, e.g., every year. We note that this update does not involve the key part stored in the tamper-resistant element of the mobile device.

**Broadcast Efficiency.** We note that, in our pay-per-view and pay-per-channel CAS is lost if the video content is transmitted on demand rather than being broadcast to users. In that case, users can employ an anonymous communication network [34] to access the video content in order to protect their privacy.

**Fine-Grained Access Control.** The BABE scheme allows the provider $\mathcal{P}$ to apply fine-grained access control. In our pay-per-channel CAS, the provider enforces a simple policy that requires users to subscribe to a channel. However, the BABE scheme allows to apply any policy described by a CNF or DNF formula, and threshold predicates, which allows the provider to enforce more complex access control policies.

In Table 3, we compare the features of our PPC-CAS with those of the PPC-CAS in [4] and [10]. As can be seen, the main advantages of our CAS in comparison with those in [4] and [10] are that our CAS provides user privacy and efficient key redistribution. The former is due to the use of POT, which allows users to subscribe to channels without revealing the channels they purchase to the provider. The latter is due to the use of BABE, which allows for a revocation method in which non-revoked users do not need to update their keys.

We use the symbol $\perp$ for properties that none of the analyzed PPC-CAS attains, but that can be achieved by extending them with suitable tools. The non-repudiation claim of [4] is not completely sound. They should use a signature scheme or message authentication code to prove
TABLE 3
Feature comparison of our CAS with those in [4] and [10].

<table>
<thead>
<tr>
<th>Feature</th>
<th>Our CAS</th>
<th>[4]</th>
<th>[10]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Privacy</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Revocation</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Collusion Resistance</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Non-repudiation</td>
<td>⊥</td>
<td>⊥</td>
<td>⊥</td>
</tr>
<tr>
<td>Efficient Key Redistribution</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Broadcast Efficiency</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Broadcast Anonymity</td>
<td>⊥</td>
<td>⊥</td>
<td>⊥</td>
</tr>
</tbody>
</table>

that the ciphertexts are not tampered. In our CAS, broadcast anonymity can be achieved by using an anonymous communication network. Non-repudiation can be achieved by using digital signature schemes and fair exchange protocols.

8 PRICED OBLIVIOUS TRANSFER

8.1 Building Blocks of Our POT Protocol

Bilinear Maps. Let $G_1$, $G_2$ and $G_T$ be groups of prime order $p$. A map $e : G_1 \times G_2 \to G_T$ must satisfy bilinearity, i.e., $e(g^i, g^j) = e(g, g)^{ij}$; non-degeneracy, i.e., for all generators $g \in G_1$ and $\tilde{g} \in G_2$, $e(g, \tilde{g})$ generates $G_T$; and efficiency, i.e., there exists an efficient algorithm BMSetup($1^k$) that outputs the pairing group setup $(p, G_1, G_2, G_T, e, g, \tilde{g})$ and an efficient algorithm to compute $e(a, b)$ for any $a \in G_1$, $b \in G_2$.

Non-Interactive ZKPK. Let $L$ be a polynomial time computable binary relation. For tuples $(wilt, ins) \in R$ we call $wilt$ the witness and $ins$ the instance. Let $L$ be the NP-language consisting of the instances of $ins$ for which there exist witnesses $wilt$ such that $(wilt, ins) \in R$. A non-interactive zero-knowledge proof of knowledge (NIPK) system for $R$ consists of the algorithms PKSetup, PKProve and PKVerify. On input a security parameter $1^k$, PKSetup($1^k$) outputs the parameters $parPK$. PKProve($parPK, wilt, ins$) checks whether $(wilt, ins) \in R$ and if that case outputs a proof $\pi$. PKVerify($parPK, ins, \pi$) outputs 1 if $\pi$ is a valid proof that $ins \in L$ or 0 if that is not the case. A NIPK system must fulfill the zero-knowledge and simulation-sound extractability properties [35].

Signatures. A signature scheme consists of the algorithms KeyGen, Sign, and VfSig. KeyGen($1^k$) outputs a secret key $sk$ and a public key $pk$, which include a description of the message space $M$. Sign($sk, m$) outputs a signature $s$ on a message $m \in M$. VfSig($pk, s, m$) outputs 1 if $s$ is a valid signature on $m$ and 0 otherwise. This definition can be extended to blocks of messages $\tilde{m} = (m_1, \ldots, m_n)$. In this case, KeyGen($1^k, n$) receives the maximum number of messages as input. A signature scheme must be existentially unforgeable [36].

Commitments. A commitment scheme consists of the algorithms CSetup, Com and VfCom. CSetup($1^k$) generates the parameters $par_c$, which include a description of the message space $M$ and of the randomness space $R$. Com($par_c, x, open$) outputs a commitment $com$ to $x \in M$ and random value $open \in R$. VfCom($par_c, com, x, open$) outputs 1 if $com$ is a commitment to $x \in M$ and random value $open \in R$ or 0 otherwise. A commitment scheme must fulfill the hiding and binding properties [37].

8.2 Our Priced Oblivious Transfer Scheme

Our construction POT for priced oblivious transfer involves a provider $P$ and a user $U$ and is parameterized by a message space $M = G_1$, a number of messages $N$, a universe of prices $U_{pr} = [0, p_{max}]$ and a universe of accounts $U_{ac} = [0, ac_{max}]$, where $ac_{max} = p_{max} \cdot N$. Construction POT uses a commitment scheme (CSetup, Com, VfCom), a signature scheme (KeyGen, Sign, VfSig) that signs messages in $G_1$ and $G_2$, a signature scheme (KeyGen, Sign, VfSig) that signs messages in $Z_p$, a double trapdoor public-key encryption scheme, and a NIPK scheme (PKSetup, PKProve, PKVerify) for the relations $R_1$, $R_2$ and $R_3$ described in our construction. We employ a NIPK scheme where the setup algorithm PKSetup is common for those relations. Our scheme uses a common references string, which is computed by a trusted party as follows.

- Run $\Phi = (p, G_1, G_2, G_T, e, g, \tilde{g}) \leftarrow$ BMSetup($1^k$), $par_c \leftarrow$ CSetup($1^k$) and $parPK \leftarrow$ PKSetup($1^k$).
- Pick random $a, b, c \leftarrow Z_p$ and set $(h_1, h_2, h_3, h_2, h_3^2) = (g^a, g^b, g^c, g^{a+b}, g^{a+b+c})$.
- Set $crs \leftarrow (\Phi, h_1, h_2, h_3, h_2, \tilde{h}_3, par_c, parPK)$.

The $crs$ is used by $P$ and $U$. In the following, we describe the algorithms of our scheme.

InitP($1^k, m_1, p_1, \ldots, m_N, p_N$).

- $P$ picks random $x_1, x_2 \leftarrow Z_p$ and sets $(w_1, w_2) = (h_3^{-1/4}, h_1^{1/4})$ and $(\tilde{w}_1, \tilde{w}_2) = (h_3^{1/2}, \tilde{h}_3^{1/2})$.
- $P$ runs KeyGen($1^k, \tilde{S}$) to obtain $(pk_1, sk_1)$ and KeyGen($1^k, 1$) to obtain $(pk_2, sk_2)$.
- For $i = 1$ to $N$, $P$ picks $r_1, r_2 \leftarrow Z_p$, runs $s_i \leftarrow$ Sign$_1(sk_1, (w_1, w_2, \tilde{w}_1, \tilde{w}_2, \tilde{g}^p))$ and sets $t_i \leftarrow (w_1^r, w_2^r, \tilde{w}_1^r, \tilde{w}_2^r, m_i, h_3^{r+1}, s_i, p_i)$. The tuple $(w_1, w_2, \tilde{w}_1, \tilde{w}_2, m_i, h_3^{r+1})$ is a double trapdoor encryption of $m_i$.
- If $j = 0$ to $ac_{max}$, $P$ runs $s'_j \leftarrow$ Sign$_2(sk_2, j)$.
- $P$ sets $com_s \leftarrow \bot$ and $st_p \leftarrow (x_1, x_2, \tilde{w}_1, \tilde{w}_2, pk_1, pk_2, com)$.
- $P$ sets $M_{init} \leftarrow (w_1, w_2, \tilde{w}_1, \tilde{w}_2, pk_1, pk_2, (s'_j)_{j=0}^{ac_{max}}, t_1, \ldots, t_N)$.
- $P$ outputs $(st_p, M_{init})$.

InitU($1^k, M_{init}$).

PayU($st_u, p$).

- $U$ aborts if $p + ac \notin [0, ac_{max}]$.
- If $com = \bot$ and $open = \bot$, $U$ sets $open' \leftarrow 0$ else picks random $open' \leftarrow R$.
- $U$ sets $ac' = ac + p$, runs $com' \leftarrow$ Com($par_c, ac', open'$), picks $l_1, l_2 \leftarrow Z_p$ and encrypts $p$ as $(c_1, c_2, c_3) \leftarrow (w_1^{l_1}, w_2^{l_2}, p \cdot h_3^{l_1+l_2})$. 
• If \( com \neq \perp \) and \( open \neq \perp \), \( U \) sets \( ins \leftarrow (par_c, com, com', p, pk_2) \) and \( \text{wit} \leftarrow (ac, ac', \sigma_{ac}', open, open') \), and runs \( \pi \leftarrow \text{PKProve}(par_{PK}, \text{wit}, ins) \) for the relation
  \[
  R_1 = \{(ins, wit) : 1 = VfCom(par_c, com, ac, open) \land \\
  1 = VfCom(par_c, com', ac', open') \land \ ac' = ac + p \land \\
  1 = VfSig_2(pk_2, ac', \sigma_{ac}') \}\]

  \( U \) replaces \((ac, com, open)\) in \( s_{tu} \) by \((ac', com', open')\).

  If \( com = \perp \) and \( open = \perp \), \( U \) sets \( M_{pay} \leftarrow (c_1, c_2, c_3) \), else sets \( M_{pay} \leftarrow (\pi, com', c_1, c_2, c_3) \).

  \( U \) outputs \( st_{tu} \) and \( M_{pay} \).

\[\text{PayP}(st_p, M_{pay}).\]

  If \( com = \perp \), \( \mathcal{P} \) parses \( M_{pay} \) as \((c_1, c_2, c_3)\), else parses \( M_{pay} \) as \((\pi, com', c_1, c_2, c_3)\).

  \( \mathcal{P} \) decrypts \( p \leftarrow c_3/(c_1^2 c_2^2) \).

  If \( com \neq \perp \), \( \mathcal{P} \) sets \( ins \leftarrow (par_c, com, com', p, pk_2) \) and aborts if \( 1 \neq \text{PKVerify}(par_{PK}, ins, \pi) \).

  If \( com = \perp \), \( \mathcal{P} \) sets \( open' = 0 \) and runs \( com' \leftarrow \text{Com}(par_c, open') \).

  \( \mathcal{P} \) replaces \( com \) by \( com' \) in \( st_p \).

  \( \mathcal{P} \) outputs \( st_p \) and \( p \).

\[\text{Requ}(st_u, i).\]

  \( U \) aborts if \( i \notin [1, N] \) or if \( st_u \) contains the values \((i', y_1, y_2, d_1, d_2)\).

  \( U \) parses \( t_i \) as \((c_1, c_2, c_3, c_4, c_5, s_i, p_i) \) and aborts if \( ac < p_i \).

  \( U \) picks random \( y_1, y_2 \leftarrow \mathbb{Z}_p \) and computes \((d_1, d_2) \leftarrow (c_1 \cdot w_1^{y_1}, c_2 \cdot w_2^{y_2}) \) and \((u_1, u_2) \leftarrow (h_1^{y_1}, h_2^{y_2}) \).

  \( U \) sets \( ac' \leftarrow ac - p_i, \) picks random \( open' \leftarrow \mathcal{R} \) and computes \( com' \leftarrow \text{Com}(par, open') \).

  \( U \) sets \( ins \leftarrow (par_c, com, com', pk_1, pk_2, d_1, d_2, h_3, \tilde{w}_1, \tilde{w}_2) \) and \( \text{wit} \leftarrow (ac, ac', open', p_i, s_i, \sigma_{ac}', c_1, c_2, u_1, u_2) \) and runs \( \pi \leftarrow \text{PKProve}(par_{PK}, \text{wit}, ins) \) for a relation
  \[
  R_2 = \{(ins, wit) : 1 = VfCom(par_c, com, ac, open) \land \\
  1 = VfCom(par_c, com', ac', open') \land \ ac' = ac - p_i \land \\
  1 = VfSig_2(pk_1, (c_1, c_2, \tilde{g}^{p_i}, s_i)) \land \\
  1 = VfSig_2(pk_2, ac', \sigma_{ac}') \land \\
  e(c_1, h_3)(u_1, \tilde{w}_1) = e(d_1, h_3) \land \\
  e(c_2, h_3)(u_2, \tilde{w}_2) = e(d_2, h_3) \}\]

  \( U \) sets \( st_u \leftarrow st_{tu} \cup (i, y_1, y_2, d_1, d_2) \).

  \( U \) sets \( M_{req} \leftarrow (d_1, d_2, \pi, com') \).

  \( U \) outputs \( st_u \) and \( M_{req} \).

\[\text{ReqP}(st_p, M_{req}).\]

  \( \mathcal{P} \) aborts if \( st_p \) contains the values \((d_1', d_2') \).

  \( \mathcal{P} \) sets \( ins \leftarrow (par_c, com, com', pk_1, pk_2, d_1, d_2, h_3, \tilde{w}_1, \tilde{w}_2) \) and aborts if \( 1 \neq \text{PKVerify}(par_{PK}, ins, \pi) \).

  \( \mathcal{P} \) replaces \( com \) by \( com' \) in \( st_p \) and sets \( st_p \leftarrow st_p \cup (d_1, d_2) \).

  \( \mathcal{P} \) outputs \( st_p \).

\[\text{SellP}(st_p).\]

  \( \mathcal{P} \) aborts if \( st_p \) does not contain the values \((d_1, d_2) \).

  \( \mathcal{P} \) calculates \((z_1, z_2) \leftarrow (d_1^{-1}, d_2^{-1}) \) and \( z \leftarrow z_1 \cdot z_2 \).

8.3 Security Analysis

In the full version, we prove that our construction POT securely realizes our ideal functionality for priced oblivious transfer \( \mathcal{F}_{\text{POT}} \) in the universal composability framework. Our construction is secure in a hybrid model where parties in the real world use ideal functionalities for common reference string \( \mathcal{F}_{\text{CRS Setup}} \) and for authenticated channels \( \mathcal{F}_{\text{AUTH}} \).

The user privacy property holds if the commitment scheme (\( \text{CSetup}, \text{Com}, \text{VfCom} \)) is hiding and if the scheme (\( \text{PKSetup}, \text{PKProve}, \text{PKVerify} \)) is zero-knowledge and sound. We note that soundness is implied by simulation-sound exttractability.

The sender security property holds if the scheme (\( \text{PKSetup}, \text{PKProve}, \text{PKVerify} \)) is zero-knowledge and simulation-sound extractable, if the signature schemes (\( \text{KeyGen}_1, \text{Sign}_1, \text{VfSig}_1 \)) and (\( \text{KeyGen}_2, \text{Sign}_2, \text{VfSig}_2 \)) are existentially unforgeable, if the double trapdoor encryption scheme fulfills the property of indistinguishability under chosen plaintext attack, and if the commitment scheme (\( \text{CSetup}, \text{Com}, \text{VfCom} \)) is binding.

9 Efficiency Analysis of Our CAS

In this section we analyze the efficiency of our CAS proposals. Results for PCC-CAS are provided in Section 9.1, along with a comparison to the scheme in [10]. The results for PPV-CAS are summarized in Section 9.2.

Our platform selection for the provider side is an off-the-shelf laptop running Linux on an Intel Core i5-3317U CPU at 1.70 GHz. On the user side, we select a Beaglebone Black equipped with a 32-bit ARM Cortex-A8 processor and clocked at 1 GHz. This platform is representative of mobile devices, as the same processor core can be found in early-generation smartphones such as Apple iPhone 4, Samsung Galaxy S or Google Nexus 5. Our implementation is coded in C language to ensure platform independence. We leverage on the Pairing Based Cryptography (PBC) library developed by Ben Lynn to implement bilinear maps. In particular, we employ type A1 curves to instantiate the ABE constructions and type D curves with a base field length of

201 bits and an embedding degree of 6 to instantiate the POT constructions. Note that PBC can be ported and executed in Android platforms, see e.g. [38], and thus we expect the performance of an Android application to be very similar to that of our Beaglebone implementation.

To instantiate the cryptographic building blocks of our POT protocol, we employ the commitment scheme in [37] for (CSetup, Com, VfCom), the signature scheme in [16] for (KeyGen_1, Sign_1, VfSig_1), and the weakly secure signature scheme in [39] for (KeyGen_2, Sign_2, VfSig_2). For the sake of efficiency, to instantiate (PKSetup, PKProve, PKVerify), we employ the Fiat-Shamir transform, which is simulation-sound extractable in the random oracle model [40]. Therefore, this instantiation does not achieve universal composability, which is achieved when using other simulation-sound extractable NIPK scheme or the Groth-Sahai proof system [41] in the manner of [15].

9.1 Efficiency Analysis of PPC-CAS

In Table 4, we summarize the performance results for each phase of the PPC-CAS schemes. Results on the left hand side of the table are given depending on the parameters of the system, namely, the number of users \( |U| \), the number of TV channels \( |N| \), the number of channels the user is subscribed to \( |A| \), and the average number of user subscriptions per channel \( |I| \). The results on the right hand side correspond to a particular initialization of the system with \( |U| = 100k \) users, \( |N| = 100 \) TV channels, \( |A| = 10 \) channels/user (in average), and \( |I| = 10k \) users/channel (in average, assuming uniform distribution). Note that in our scheme, the value of \( |A| \) corresponds to the number of attributes. This is however not the case in [10], where the number of attributes is fixed to four.

In our scheme, the running time of the setup algorithm is dominated by the number of users \( |U| \), which, at the same time, determines the size of the public parameters. The running time of the key generation algorithm and the key size grow with \( |A| \), which in this case corresponds to the number of attributes in BABE. More specifically, the execution time of KeyGen_0 is 50 ms and is independent of the number of attributes, whereas the running time of KeyGen_1 demands 0.8 ms for each user attribute. The running time of the POT initialization phase is dominated by the loops iterating over the number of messages \( N \) and the universe of accounts \( U_{ac} \) which in our application correspond to the number of channels \( |N| \) and users \( |U| \), respectively. Timings on the user side are significantly larger due to the use of a less powerful computing platform, but also due to the fact that our building blocks demand the computation of bilinear maps during signature verification but not during signature generation. The running times of the payment and purchase algorithms are constant and independent of any parameter of the scheme. The running time of the encryption algorithm grows with the number of users \( |I| \) that have subscribed to that particular channel. Note however that the ciphertext size is constant at 1040 bytes. Finally for decryption, the running time grows similarly with \( |I| \).

The comparison with the results obtained from the scheme in [10] yields some clear differences. First, we observe that both running time and communication in the setup algorithm are independent of any parameter in the PPC-CAS construction and significantly lower than in our scheme. Second, the complexity of the user registration phase (i.e. invoking user key generation) increases with \( |A| \). This is the same behaviour as our PPC-CAS, yet in this case the scheme in [10] performs worse. Third, the payment and purchase algorithms are not present in the comparison, since, in contrast to our work, the scheme in [10] does not protect users’ privacy and therefore omits any description of the payment mechanisms. Finally we observe that only the encryption algorithm depends on \( |I| \), and that both encryption and decryption are more costly than in our scheme. Moreover, the CAS in [10] needs key updates for revocation purposes.

In summary, except for the setup phase, our CAS is more efficient. The reason is that our CAS uses BABE, which provides an efficient direct revocation method. The CAS in [10] needs to include attributes for revocation purposes in the user keys and in the ciphertexts, which affects negatively the efficiency of the user registration and SEK update phases.

Independently of this observation, the results provided in Table 4 prove that our PPC-CAS scheme is practical. The time required for encrypting and decrypting the SEK is perfectly assumable, and the size of the ciphertext that is broadcast is only 1040 bytes. Recall that the SEK is updated weekly. The most time consuming result corresponds to the verification for \( U \), which requires over 2 hours for \( |U| = 100k \). While this seems a prohibitive result, we note that, in practice, at initialization the mobile pay-tv users receive the same message from the provider. Therefore, the verification step can be carried out by a trusted authority. The authority can verify the message once, compute its hash and sign the hash. That way, each user only needs to compute the hash of the message and verify the signature computed by the authority.

9.2 Efficiency Analysis of PPV-CAS

The results of our PPV-CAS implementation are summarized in Table 5 (initialization phase) and Table 6 (payment and purchase phases). Recall that our PPV-CAS leverages on POT to achieve privacy but, in contrast to our PPC-CAS scheme, it does not rely on BABE.

Similar to the case of PPC-CAS, the running time of the initialization phase is dominated by the loops iterating over the number of messages \( N \) and the universe of accounts \( U_{ac} \) which in our application correspond to the number of channels \( |N| \) and users \( |U| \), respectively. Timings on the user side are significantly larger due to the use of a less powerful computing platform, but also due to the fact that our building blocks demand the computation of bilinear maps during signature verification but not during signature generation. The running times of the payment and purchase algorithms are constant and independent of any parameter of the scheme. The running time of the encryption algorithm grows with the number of users \( |I| \) that have subscribed to that particular channel. Note however that the ciphertext size is constant at 1040 bytes. Finally for decryption, the running time grows similarly with \( |I| \).

The comparison with the results obtained from the scheme in [10] yields some clear differences. First, we observe that both running time and communication in the setup algorithm are independent of any parameter in the PPC-CAS construction and significantly lower than in our scheme. Second, the complexity of the user registration phase (i.e. invoking user key generation) increases with \( |A| \). This is the same behaviour as our PPC-CAS, yet in this case the scheme in [10] performs worse. Third, the payment and purchase algorithms are not present in the comparison, since, in contrast to our work, the scheme in [10] does not protect users’ privacy and therefore omits any description of the payment mechanisms. Finally we observe that only the encryption algorithm depends on \( |I| \), and that both encryption and decryption are more costly than in our scheme. Moreover, the CAS in [10] needs key updates for revocation purposes.

In summary, except for the setup phase, our CAS is more efficient. The reason is that our CAS uses BABE, which provides an efficient direct revocation method. The CAS in [10] needs to include attributes for revocation purposes in the user keys and in the ciphertexts, which affects negatively the efficiency of the user registration and SEK update phases.

Independently of this observation, the results provided in Table 4 prove that our PPC-CAS scheme is practical. The time required for encrypting and decrypting the SEK is perfectly assumable, and the size of the ciphertext that is broadcast is only 1040 bytes. Recall that the SEK is updated weekly. The most time consuming result corresponds to the verification for \( U \), which requires over 2 hours for \( |U| = 100k \). While this seems a prohibitive result, we note that, in practice, at initialization the mobile pay-tv users receive the same message from the provider. Therefore, the verification step can be carried out by a trusted authority. The authority can verify the message once, compute its hash and sign the hash. That way, each user only needs to compute the hash of the message and verify the signature computed by the authority.

9.2 Efficiency Analysis of PPV-CAS

The results of our PPV-CAS implementation are summarized in Table 5 (initialization phase) and Table 6 (payment and purchase phases). Recall that our PPV-CAS leverages on POT to achieve privacy but, in contrast to our PPC-CAS scheme, it does not rely on BABE.

Similar to the case of PPC-CAS, the running time of the initialization phase is dominated by the loops iterating over the number of messages \( N \) and the universe of accounts \( U_{ac} \) which in our application correspond to the number of channels \( |N| \) and users \( |U| \), respectively. Timings on the user side are significantly larger due to the use of a less powerful computing platform, but also due to the fact that our building blocks demand the computation of bilinear maps during signature verification but not during signature generation. The running times of the payment and purchase algorithms are constant and independent of any parameter of the scheme. The running time of the encryption algorithm grows with the number of users \( |I| \) that have subscribed to that particular channel. Note however that the ciphertext size is constant at 1040 bytes. Finally for decryption, the running time grows similarly with \( |I| \).

The comparison with the results obtained from the scheme in [10] yields some clear differences. First, we observe that both running time and communication in the setup algorithm are independent of any parameter in the PPC-CAS construction and significantly lower than in our scheme. Second, the complexity of the user registration phase (i.e. invoking user key generation) increases with \( |A| \). This is the same behaviour as our PPC-CAS, yet in this case the scheme in [10] performs worse. Third, the payment and purchase algorithms are not present in the comparison, since, in contrast to our work, the scheme in [10] does not protect users’ privacy and therefore omits any description of the payment mechanisms. Finally we observe that only the encryption algorithm depends on \( |I| \), and that both encryption and decryption are more costly than in our scheme. Moreover, the CAS in [10] needs key updates for revocation purposes.

In summary, except for the setup phase, our CAS is more efficient. The reason is that our CAS uses BABE, which provides an efficient direct revocation method. The CAS in [10] needs to include attributes for revocation purposes in the user keys and in the ciphertexts, which affects negatively the efficiency of the user registration and SEK update phases.

Independently of this observation, the results provided in Table 4 prove that our PPC-CAS scheme is practical. The time required for encrypting and decrypting the SEK is perfectly assumable, and the size of the ciphertext that is broadcast is only 1040 bytes. Recall that the SEK is updated weekly. The most time consuming result corresponds to the verification for \( U \), which requires over 2 hours for \( |U| = 100k \). While this seems a prohibitive result, we note that, in practice, at initialization the mobile pay-tv users receive the same message from the provider. Therefore, the verification step can be carried out by a trusted authority. The authority can verify the message once, compute its hash and sign the hash. That way, each user only needs to compute the hash of the message and verify the signature computed by the authority.
TABLE 4
Performance analysis of our PPC-CAS and comparison.

<table>
<thead>
<tr>
<th>Protocol Phase</th>
<th>Timings (in ms)</th>
<th>Communication (in bytes)</th>
<th>Timings</th>
<th>Communication</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>P</td>
<td>U</td>
<td>P ↔ U</td>
<td>P ↔ U</td>
</tr>
<tr>
<td>SETUP</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>InitP,InitU</td>
<td>50[U]</td>
<td>—</td>
<td>2 401 + 50[U]</td>
<td>5 000 s —</td>
</tr>
<tr>
<td>User Registration</td>
<td>100[N] + [U]</td>
<td>882[N] + 78[U]</td>
<td>1040[N] + 52[U]</td>
<td>110 s 7 888 s</td>
</tr>
<tr>
<td>KeyGen, KeyGen</td>
<td>50 + 0.8[A]</td>
<td>—</td>
<td>520 + 260 ·</td>
<td>58 ms —</td>
</tr>
<tr>
<td>PAYMENT</td>
<td></td>
<td></td>
<td>[A]</td>
<td></td>
</tr>
<tr>
<td>PayU, PayP</td>
<td>61</td>
<td>462</td>
<td>288</td>
<td>61 ms 462 ms</td>
</tr>
<tr>
<td>PURCHASE</td>
<td>174</td>
<td>2 390</td>
<td>1 235</td>
<td>1 235 bytes</td>
</tr>
<tr>
<td>ReqU, ReqP</td>
<td>71</td>
<td>720</td>
<td>491</td>
<td>71 ms 720 ms</td>
</tr>
<tr>
<td>SellP, SellU</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SEK UPDATE</td>
<td>106 + 0.016[U]</td>
<td>—</td>
<td>1 040</td>
<td>266 ms —</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>OUR SCHEME</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SETUP</td>
<td>140</td>
<td>—</td>
<td>1 560</td>
<td>1 560 bytes</td>
</tr>
<tr>
<td>User Registration</td>
<td>2 352 ·</td>
<td>—</td>
<td>2 340 ·</td>
<td>2 340 bytes</td>
</tr>
<tr>
<td>SEK UPDATE</td>
<td>2 380</td>
<td>—</td>
<td>3 120</td>
<td>3 120 bytes</td>
</tr>
</tbody>
</table>

TABLE 5
PPV-CAS: Initialization phase.

<table>
<thead>
<tr>
<th>PARAMETERS</th>
<th>TIMINGS</th>
<th>COMMUNICATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>ac max</td>
<td>N</td>
<td>P</td>
</tr>
<tr>
<td>50</td>
<td>15 s</td>
<td>824 s</td>
</tr>
<tr>
<td>10 k</td>
<td>20 s</td>
<td>868 s</td>
</tr>
<tr>
<td>200</td>
<td>30 s</td>
<td>956 s</td>
</tr>
<tr>
<td>50</td>
<td>105 s</td>
<td>7 844 s</td>
</tr>
<tr>
<td>100 k</td>
<td>110 s</td>
<td>7 888 s</td>
</tr>
<tr>
<td>200</td>
<td>120 s</td>
<td>7 976 s</td>
</tr>
</tbody>
</table>

TABLE 6
PPV-CAS: Payment, request and sell phases.

<table>
<thead>
<tr>
<th>INTERFACE</th>
<th>TIMINGS</th>
<th>COMMUNICATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>PayP</td>
<td>61 ms</td>
<td>462 ms</td>
</tr>
<tr>
<td>ReqP</td>
<td>174 ms</td>
<td>2 390 ms</td>
</tr>
<tr>
<td>SellP</td>
<td>71 ms</td>
<td>720 ms</td>
</tr>
</tbody>
</table>

10 CONCLUSION

We proposed a pay-per-view and a pay-per-channel CAS that protect user privacy. Both our CAS employ priced oblivious transfer to allow the user to purchase programs and to subscribe to channels without disclosing sensitive information to the service provider. Our pay-per-channel CAS also employs broadcast attribute-based encryption, which provides broadcast efficiency, simple revocation, low storage overhead and protection against user collusion. Our implementation shows that our CAS is perfectly feasible for mobile devices.

REFERENCES


