Universally Composable Efficient Priced Oblivious Transfer from a Flexible Membership Encryption*

Pratish Datta, Ratna Dutta, and Sourav Mukhopadhyay

Department of Mathematics
Indian Institute of Technology Kharagpur
Kharagpur-721302, India
{pratishdatta,ratna,sourav}@maths.iitkgp.ernet.in

Abstract. Membership encryption is a newly developed cryptographic primitive that combines membership proof and encryption into an unified setting. This paper presents a new flexible membership encryption scheme which is provably secure and significantly more efficient than the previous scheme. Further we apply our proposed membership encryption to construct a round optimal 1-out-of-n priced oblivious transfer (POT) protocol which, unlike the existing 1-out-of-n POT schemes, is proven secure under the universally composable (UC) security model and thus preserves security when it is executed with multiple protocol instances that run concurrently in an adversarily controlled way. Moreover, using our membership encryption, the POT protocol exhibits constant communication complexity on the buyer's side and $O(n)$ communication cost on the vendor's side, which is so far the best known in the literature.

Keywords: membership encryption, priced oblivious transfer, universally composable security, bilinear maps, non-interactive proof of knowledge, P-Signature, non-interactive range proof.

1 Introduction

Membership proof and *membership encryption* are two important cryptographic primitives of which membership encryption has been developed very recently. Membership proof [9], [13], [2], [11] is useful and nontrivial particularly when protecting the privacy is at prime concern. Membership encryption combines encryption and membership proof into a unified setting, thereby improving the communication efficiency. Further, while a membership proof cannot be converted to a membership encryption, a successful decryption of the ciphertext in membership encryption naturally serves as a proof of membership. The idea of membership encryption is that, if a message is encrypted using an attribute and a privacy preserving token for a group attribute, decryption of the ciphertext is possible if and only if the used attribute is a member of the used group attribute. The concept of membership encryption has been introduced by Guo et al. [18] and, to the best of our knowledge, is so far the only membership encryption available in the literature. Membership encryption is applicable in advanced cryptographic protocols where privacy protection is important, e.g., priced oblivious transfer.

Priced oblivious transfer (POT) protocol aims at protecting the privacy of customers purchasing digital goods. More specifically, POT allows a buyer to purchase digital goods from a vendor without letting the vendor learn what it is buying. Usually after making a pre-payment to the vendor, the buyer engages in an unlimited number of transactions such that, as long as the buyer's balance contains sufficient funds it will successfully retrieve the selected item and its balance will be debited by the item's price. However, the buyer should be unable to retrieve an item whose cost exceeds its remaining balance.

The first priced oblivious transfer scheme [1], as well as subsequent works [21] analyse security in the half-simulation model, where simulation security is required against the buyer only and stand-alone privacy is needed against the vendor. As explained in [14], these protocols fail even under sequential composition and are shown to be vulnerable to practical attacks. Afterwards, a number of universally composable (UC)-secure priced oblivious transfer protocols have been proposed [19], [10], [20]. The UC-security paradigm [12] provides a framework for representing cryptographic protocols and analysing their security. Protocols that are proven UC-secure maintain their security even when they are run concurrently with an unbounded number of arbitrary protocol instances controlled by an adversary.

Our contributions: Our contribution in this paper is two fold:

* This is the full version of the paper that appeared in Proceedings of the 19th Australasian Conference on Information Security and Privacy (ACISP 2014), LNCS 8544, pp. 98–114, Springer.

- *Firstly*, we introduce a cost-effective flexible membership encryption scheme secure in the standard model that outperforms the existing one [18].
- *Secondly*, we use our membership encryption scheme to construct a UC-secure 1-out-of-n priced oblivious transfer protocol having the best computational and communication efficiency over the previous similar schemes.

To be precise, our membership encryption scheme is built on a prime order bilinear group setup. Our scheme is proven to be secure in the selective security model of [18] without using random oracles under the **Square Decisional Bilinear Diffie-Hellman** and **Simultaneous Square Decisional Bilinear Diffie-Hellman** assumptions. Unlike [18], our scheme is flexible in the sense that the universe of attributes $\mathbb{A}$ can be changed at any time keeping the setup unaltered. This property is crucial for an application such as priced oblivious transfer where item prices may change with time. Further in our membership encryption, the group token and ciphertext respectively have one and two group elements, whereas, those for [18] consists of three group elements each. Also, our scheme requires $(n^2+n)/2+2k+3$ exponentiations and 2 pairings compared to $n^2 + 3n + 4k + 12$ exponentiations and 3 pairings for [18], where k is the size of the group attribute. On a more positive note, our scheme, when applied with a fixed universe of attributes, results in constant computational cost.

Next, we apply our proposed membership encryption scheme to construct an 1-out-of-n priced oblivious transfer protocol that is UC-secure under the assumption that there is an honestly generated common reference string, employing Groth-Sahai non-interactive proof techniques [17], P-Signatures [3] and non-interactive range proof [19]. Security is proven in a static corruption model without relying on random oracles under the **Hidden Strong Diffie-Hellman**, **Triple Diffie-Hellman**, **Decisional linear** and **Square Decisional Bilinear Diffie-Hellman** assumptions. Our protocol allows more than one item to have the same price. After an initialization of complexity $O(n)$, each transfer phase is optimal in terms of rounds of communication. Moreover, the complexities of computation and communication are constant on the buyer's side whereas $O(n)$ on the vendor's side. When compared with [1], [21], which are the only two existing 1-out-of-n priced oblivious transfer schemes, our construction is concrete, provides significantly strong security guarantees and achieves the best known computational and communication efficiency.

Outline of the paper: The rest of the paper is organized as follows. We review the definitions and security notions of membership encryption and priced oblivious transfer together with the security assumptions we use in Section 2. Next, in Section 3 we present our membership encryption scheme along with the security proof and efficiency comparison with existing scheme. After that, we demonstrate our priced oblivious transfer protocol constructed using our proposed membership encryption providing the complete security argument of the protocol in Section 4. Finally, Section 5 concludes the paper.

2 Preliminaries

A function ϵ is *negligible* if, for every integer c, there exists an integer K such that for all $k > K$, $|\epsilon(k)| < 1/k^c$. A problem is said to be *computationally hard* (or *intractable*) if there exists no probabilistic polynomial time ($\mathsf{p.p.t.}$) algorithm that solves it with non-negligible probability (in the size of the input or the security parameter).

2.1 Membership Encryption and Security Notions

Formally, a membership encryption $A \in \mathcal{P}(\mathbf{G})$ consists of the following four algorithms, where A is an attribute, $\mathbf{G}$ is a group attribute and $\mathcal{P}(\cdot)$ denotes some sort of privacy protection such that given $\mathcal{P}(\mathbf{G})$ it is hard to determine $\mathbf{G}$. Henceforth, we will refer $\mathcal{P}(\mathbf{G})$ a group token for $\mathbf{G}$.
MSetup: Taking as input a security parameter 1^λ and universe of attributes $\{A_1, A_2, \ldots, A_n\}$, a trusted authority generates and publishes the system parameter SP.
MGroupGen: The decryptor takes as input the system parameter SP and a group attribute $\mathbf{G} = \{A_{i_1}, A_{i_2}, ..., A_{i_k}\}$ $(1 \leq k \leq n)$, and determines the group token $\mathcal{P}(\mathbf{G})$ together with the secret key $\mathbf{S}$. It sends $\mathcal{P}(\mathbf{G})$ to the encryptor and keeps $\mathbf{S}$ secret to itself.
MEncrypt: On input the system parameter SP, an attribute A, a group token $\mathcal{P}(\mathbf{G})$ and a message M, the encryptor returns a ciphertext C. We define the ciphertext as $C \leftarrow \mathcal{ME}[A, \mathcal{P}(\mathbf{G}), M]$.
MDecrypt: Taking as input the attribute A, the group attribute $\mathbf{G}$, the secret key $\mathbf{S}$ and the ciphertext C, the decryptor retrieves the message M or a null string $\perp$. We define the decryption as $\{M, \perp\} \leftarrow$

$\mathcal{MD}[C, \mathbf{G}, \mathbf{S}]$.
The membership encryption is said to be correct if

$$\mathcal{MD}[\mathcal{ME}[A, \mathcal{P}(\mathbf{G}), M], \mathbf{G}, \mathbf{S}] = \begin{cases} M, \text{ when } A \in \mathbf{G} \\ \perp, \text{ when } A \notin \mathbf{G} \end{cases}$$

• **Security Model:** We adopt the security model of [18] to analyse the security of our membership encryption scheme.

Definition 1 (Message Security). *A membership encryption captures the message security if given a ciphertext generated with attribute A and group token $\mathcal{P}(\boldsymbol{G})$ for a group attribute $\boldsymbol{G}$, it is computationally hard to decrypt the ciphertext when either the decryptor does not have the secret key $\boldsymbol{S}$ of $\mathcal{P}(\boldsymbol{G})$ or the attribute A does not satisfy the membership, i.e., $A \notin \boldsymbol{G}$.*

More formally, message security is captured by the following two games.

Game 1: Indistinguishability against Secret Key

Setup: The challenger runs the MSetup algorithm to generate the system parameter SP and sends it to the adversary.

Phase 1: The adversary queries for group tokens and decryption which are answered by the challenger as follows:

- For a token query on group attribute $\mathbf{G}_i$ that is adaptively chosen by the adversary, the challenger responds by generating $(\mathcal{P}(\mathbf{G}_i), \mathbf{S}_i)$ and sending $\mathcal{P}(\mathbf{G}_i)$ to the adversary.
- For a decryption query on a ciphertext C_i for $(A, \mathcal{P}(\mathbf{G}_i))$ where $\mathcal{P}(\mathbf{G}_i)$ is generated by the challenger, the challenger returns $\perp$ to the adversary if $A \notin \mathbf{G}_i$; otherwise, the challenger responds by decrypting the ciphertext with $\mathbf{S}_i$, and sending the decryption result to the adversary.

Challenge: The adversary gives the challenger an attribute A^*, a group token $\mathcal{P}(\mathbf{G}^*)$ and two messages M_0, M_1, where $\mathcal{P}(\mathbf{G}^*)$ was generated in the query phase. The challenger responds by randomly choosing a coin $c \in \{0, 1\}$, generating a ciphertext $C^* \leftarrow \mathcal{ME}[A^*, \mathcal{P}(\mathbf{G}^*), M_c]$, and sending the challenge ciphertext to the adversary.

Phase 2: The adversary can continue the query as in Phase 1 except no decryption query on the challenge ciphertext C^* for $(A^*, \mathcal{P}(\mathbf{G}^*))$.

Win: The adversary outputs a guess c' of c and wins the game if $c' = c$.

We define the advantage of the adversary as $\mathsf{Adv}_{I_1} = |\mathsf{Pr}[c' = c] - 1/2|$.

Definition 2 (Security against Secret Key). *A membership encryption generated with a security parameter 1^λ is (t, q_k, q_d, ϵ)-secure against secret key if $\epsilon = \mathsf{Adv}_{I_1}$ is a negligible function of λ for all adversaries who make at most q_k token queries, q_d decryption queries and whose running time is at most t where t is polynomial in the security parameter. We call the membership encryption selectively secure* [5] *against secret key if the adversary outputs A^* before the setup of system parameters.*

Note 1. The security model presented above captures security against chosen ciphertext attack (CCA) which is the strongest security notion. If we do not allow any decryption query in **Phase 1** or **Phase 2**, the resulting model is relatively weaker and is known as chosen plaintext attack (CPA) security model.

Game 2: Indistinguishability against Membership

Setup: The challenger runs the MSetup algorithm to generate the system parameter SP and sends it to the adversary.

Challenge: The adversary gives the challenger an attribute A^*, a group token $\mathcal{P}(\mathbf{G}^*)$, group attribute $\mathbf{G}^*$, secret key $\mathbf{S}$ and two messages M_0, M_1.The challenger first verifies that $A^* \notin \mathcal{P}(\mathbf{G}^*)$ with $\mathbf{G}^*$ and $\mathbf{S}$. Then, the challenger responds by randomly choosing a coin $c \in \{0, 1\}$, generating a ciphertext $C^* \leftarrow \mathcal{ME}[A^*, \mathcal{P}(\mathbf{G}^*), M_c]$, and sending the challenge ciphertext to the adversary.

Win: The adversary outputs a guess c' of c and wins the game if $c' = c$.

The advantage of the adversary is defined as $\mathsf{Adv}_{I_2} = |\mathsf{Pr}[c' = c] - 1/2|$.

Definition 3 (Security against Membership). *A membership encryption generated with a security parameter 1^λ is (t, ϵ)-secure against membership if $\epsilon = \mathsf{Adv}_{I_2}$ is a negligible function of λ for all adversaries whose running time is at most t which is a polynomial in the security parameter. We call the membership encryption selectively secure* [5] *against membership if the adversary outputs A^* and $\boldsymbol{G}^*$ before the setup of system parameters.*

Definition 4 (Privacy). *A membership encryption preserves the privacy of group attributes if given a group token $\mathcal{P}(\boldsymbol{G})$ and two group attributes $\boldsymbol{G}_0 = \{A_{i_1}, A_{i_2}, \ldots, A_{i_{k_1}}\}$ and $\boldsymbol{G}_1 = \{A_{j_1}, A_{j_2}, \ldots, A_{j_{k_2}}\}$, it is computationally hard to decide whether $\boldsymbol{G} = \boldsymbol{G}_0$ or $\boldsymbol{G} = \boldsymbol{G}_1$.*

The notion of privacy is formally defined by the following game.

Game 3: Privacy

Setup: The challenger runs the MSetup algorithm to generate the system parameter SP and sends it to the adversary.

Challenge: The adversary gives the challenger two group attributes $\mathbf{G}_0 = \{A_{i_1}, A_{i_2}, \ldots, A_{i_{k_1}}\}$ and $\mathbf{G}_1 = \{A_{j_1}, A_{j_2}, \ldots, A_{j_{k_2}}\}$. The challenger responds by randomly choosing a coin $c \in \{0, 1\}$, generating $\mathcal{P}(\mathbf{G}_c)$ for $\mathbf{G}_c$ and sending $\mathcal{P}(\mathbf{G}_c)$ to the adversary.

Win: The adversary outputs a guess c' of c and wins the game if $c' = c$.

We define the advantage of adversary as $\mathsf{Adv}_P = |\mathsf{Pr}[c' = c] - 1/2|$.

Definition 5 (Privacy of Group Tokens). *A membership encryption generated with a security parameter 1^λ is said to (t, ϵ) preserves the privacy of group tokens if $\epsilon = \mathsf{Adv}_P$ is a negligible function of λ for all adversaries whose running time is at most t, where t is a polynomial in the security parameter. We say it unconditionally preserves the privacy of group tokens if $\epsilon = 0$ for any time t and SP is generated by the adversary.*

2.2 Priced Oblivious Transfer and Security Notion

A 1-out-of-n priced oblivious transfer (POT) is an "on-line" protocol between a buyer $\mathcal{B}$ and a vendor $\mathcal{V}$. This enables the buyer and vendor to engage in multiple transactions. Both the buyer and the vendor are allowed to store (short) state information between transactions. Below we describe the desired functionality of a 1-out-of-n priced oblivious transfer protocol following [1].

Main Protocol:

Initialization phase: At time $t = 0$, the buyer initializes its balance with a pre-payment to the vendor.

Transfer phase: At time $t > 0$, $(t = 1, 2, \ldots)$

- The vendor may choose a database $M = (M_1, \ldots, M_n)$ of n items for sale and some public information Pub concerning the identity of these items. Pub contains a price list $\mathbf{p} = (p_1, p_2, \ldots, p_n)$.
- The buyer then may decide to *buy* the i-th item, where $1 \leq i \leq n$. If the buyer's remaining balance is sufficiently large, i.e., the combined price of all items previously received plus the current price p_i does not exceed the initial deposit, the buyer receives M_i and obtains no information about the other items. Besides, the vendor will not be able to know which item is bought.

Subscription: The main motivation of a subscription is to allow efficient one-way communication from the vendor to the buyer. In the subscription setting we have the following operations:

- *Subscribe* to the i-th channel; by subscribing, the buyer indicates that it wishes to continue buying the i-th item until overriding the subscription with a new request. We assume that throughout the subscription, the buyer is charged the price p_i effective when initiating the subscription (even though $\mathbf{p}$ may change).
- *Unsubscribe*, i.e., terminate a previous "subscribe" request.
- *Do nothing*, i.e., maintain its default subscription (if any) or keep idle (otherwise).

• **Universally Composable Security Model:** We use the universally composable (UC) security framework [12] with static corruptions to prove security of our POT construction. In this framework, parties are modeled as probabilistic polynomial time interactive Turing machines (ITM). A protocol ψ is UC-secure if there exists no environment $\mathcal{Z}$ that can distinguish whether it is interacting with the adversary $\mathcal{A}$ and parties running the protocol ψ or with the ideal process for carrying out the desired task. In the ideal process, the ideal adversary $\mathcal{E}$ and dummy parties interact with an ideal functionality $\mathcal{F}_\psi$ which acts as a trusted ITM that carries out the desired task. More formally, we say that the protocol ψ emulates the ideal process when, for all environments $\mathcal{Z}$, the ensembles $\mathsf{IDEAL}_{\mathcal{F}_\psi, \mathcal{E}, \mathcal{Z}}$

)A,

with 1^λ as $\mathcal{Z}$'s security parameter together with x as $\mathcal{Z}$'s input. The identity of an ITM is determined at invocation time by the invoking instance and it is unchangeable. Each identity consists of two different fields: a session id (*sid*) and a party id (*pid*). For a particular instance of the protocol, *sid* is fixed for all the parties. This *sid* is determined by $\mathcal{Z}$.

As in [19], our construction operates in the $\mathcal{F}_{\mathsf{CRS}}$-hybrid plain model, where parties have access to an honestly-generated common reference string crs and authenticated channels. Further, following [16], [19], we assume the environment learns about the common reference string from the adversary, and thus the ideal world adversary can setup a string with "trapdoor information". This is at odds with the notion that the crs is a "global" entity. However, as mentioned in [16], there are strong impossibility results for UC-realising oblivious transfer in a setting where the crs is available to everyone (including the environment) and can no longer be crafted by the ideal world adversary.

Below we present the description of the ideal functionality $\mathcal{F}_{\mathsf{CRS}}$ for generating common reference string. $\mathcal{F}_{\mathsf{CRS}}$ is parameterized with a distribution D and a set of participants $\mathcal{P}$. For the POT scheme, $\mathcal{P}$ is restricted to contain only the buyer $\mathcal{B}$ and the vendor $\mathcal{V}$. We also describe an ideal functionality $\mathcal{F}_{\mathsf{POT}}$ for 1-out-of-n POT.

$\mathcal{F}_{\mathsf{CRS}}$: On input (sid, crs) from party P, it checks if $P \in \mathcal{P}$. If not, it aborts. Otherwise, if there is no value r recorded, it picks $r \leftarrow D$ and records r. It sends (sid, crs, r) to P.

$\mathcal{F}_{\mathsf{POT}}$: Parameterized with integers (n, l), a maximum price p_{max}, a deposit upper bound A, and running with a vendor $\mathcal{V}$ together with a buyer $\mathcal{B}$, $\mathcal{F}_{\mathsf{POT}}$ works as follows:

At time $t = 0$,

(a) On input a message $(sid, \mathcal{V})$ from $\mathcal{V}$, it sends *sid* to $\mathcal{B}$ and to the adversary.
(b) On input a message $(sid, \mathcal{B}, deposit)$, where $deposit \in [0, \ldots, A)$, it checks if a $(sid, \mathcal{V})$ message was received before. If not, it does nothing; otherwise, it stores *deposit* and sends $(sid, deposit)$ to $\mathcal{V}$.

At time $t > 0$,

(a) On input a message $(sid, \mathcal{V}, \mathsf{Pub})$ from $\mathcal{V}$, where Pub contains the identities of the messages chosen by $\mathcal{V}$ for sale and a price list $\{p_1, \ldots, p_n\}$ such that each $p_i \in [0, p_{max}]$, it does nothing if $(sid, \mathcal{V})$ and $(sid, \mathcal{B}, deposit)$ were not obtained previously. Otherwise, it stores $\{p_1 \ldots, p_n\}$ and sends (sid, Pub) to $\mathcal{B}$ and to the adversary.
(b) On input a message $(sid, \mathcal{B}, \sigma_t)$ from $\mathcal{B}$, it does nothing, if either $(sid, \mathcal{V})$, $(sid, \mathcal{B}, deposit)$ and $(sid, \mathcal{V}, \mathsf{Pub})$ were not received earlier or $deposit - p_{\sigma_t} < 0$. Otherwise, it sends $(sid, request)$ to $\mathcal{V}$ and receives $(sid, \{M_1, \ldots, M_n\})$ in response where M_i's are the messages whose identities were contained in Pub such that each $M_i \in \{0,1\}^l$. It updates $deposit = deposit - p_{\sigma_t}$ and hands (sid, M_{σ_t}) to $\mathcal{B}$ as well as $(sid, 1)$ to the adversary.

2.3 Bilinear Maps and Complexity Assumptions

Let $\mathbb{G}$ and $\mathbb{G}_T$ be multiplicative groups of prime order p. A bilinear map $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ must satisfy the following properties:

(a) *Bilinearity* : A map $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ is bilinear if $e(a^x, b^y) = e(a, b)^{xy}$ for all $a, b \in \mathbb{G}$ and $x, y \in \mathbb{Z}_p$;
(b) *Non-degeneracy*: For all generators $g \in \mathbb{G}$, $e(g, g)$ generates $\mathbb{G}_T$;
(c) *Efficiency*: There exists an efficient algorithm that outputs the pairing group setup $(p, \mathbb{G}, \mathbb{G}_T, e, g)$ and an efficient algorithm to compute $e(a, b)$ for any $a, b \in \mathbb{G}$.

Definition 6. [Hidden Strong DH (HSDH)]: *On input $(g, g^\alpha) \in \mathbb{G}^2, u \in \mathbb{G}$, and a set of tuples $(g^{1/(\alpha+c_i)}, g^{c_i}, u^{c_i})_{i=1}^l$ for random exponents $\alpha \in \mathbb{Z}_p^*$, $c_1, \ldots, c_l \in \mathbb{Z}_p$, the l-**HSDH** assumption holds if it is computationally hard to output a new tuple $(g^{1/(\alpha+c)}, g^c, u^c)$ for $c \in \mathbb{Z}_p$.*

Definition 7. [Triple DH (TDH)]: *On input $(g, g^x, g^y) \in \mathbb{G}^3$ and a set of tuples $(c_i, g^{1/(x+c_i)})_{i=1}^l$ for random exponents $x, y \in \mathbb{Z}_p^*$, $c_1, \ldots, c_l \in \mathbb{Z}_p$, the l-**TDH** assumption holds if it is computationally hard to output a tuple $(g^{\mu x}, g^{\mu y}, g^{\mu xy})$ for $\mu \in \mathbb{Z}_p^*$.*

Definition 8. [Decisional Linear (DLIN)]: *On input $(g, g^a, g^b, g^{ac}, g^{bd}, z) \in \mathbb{G}^6$ for random exponents $a, b \in \mathbb{Z}_p^*$, $c, d \in \mathbb{Z}_p$, the **DLIN** assumption holds if it computationally hard to decide whether $z = g^{c+d}$.*

The validity of the **HSDH** assumption in the generic group model is proven by Boyen and Waters [8] and that of the **DLIN** assumption by Boneh et al. [6]. The **TDH** assumption has been introduced by Belenkiy et al. [3]. We introduce two new assumptions, viz., the **Square Decisional Bilinear DH** assumption, which is a derived version of the well-known **Decisional Bilinear DH (DBDH)** assumption introduced by Boneh and Francklin [7], and an extended version of that, namely, the **Simultaneous Square Decisional Bilinear DH** assumption.

Definition 9. [Square Decisional Bilinear DH (SqDBDH)]: *On input* $(g, g^a, g^b, z) \in \mathbb{G}^3 \times \mathbb{G}_T$ *for random exponents* $a, b \in \mathbb{Z}_p^*$*, the* ***SqDBDH*** *assumption holds if it computationally hard to decide whether* $z = e(g,g)^{a^2 b}$.

Definition 10. [Simultaneous Square Decisional Bilinear DH (SimSqDBDH)]: *On input* $(g, g^a, g^b, z_1, z_2) \in \mathbb{G}^3 \times \mathbb{G}_T^2$ *for random exponents* $a, b \in \mathbb{Z}_p^*$*, the* ***SimSqDBDH*** *assumption holds if it is computationally hard to decide whether* $z_1 = e(g,g)^{a^2 b}$ *and* $z_2 = e(g,g)^{ab^2}$.

The **SqDBDH** problem and the **SimSqDBDH** problem in $(\mathbb{G}, \mathbb{G}_T, e)$ are no harder than the **DBDH** problem in $(\mathbb{G}, \mathbb{G}_T, e)$. However, in both cases, the converse is currently an open problem. Nonetheless, one can easily establish that the computational version of both of our new problems are exactly equivalent to the **Bilinear DH (BDH)** problem. We believe the **SqDBDH** and the **SimSqDBDH** assumptions hold in certain bilinear groups of prime order.

3 Our Membership Encryption

MSetup: On input a security parameter 1^λ and universe of attributes $\mathbb{A} = \{A_1, \ldots, A_n\} \subseteq \mathbb{Z}_p$, a trusted authority runs the MSetup algorithm that works as follows:

- Choose a pairing group $\mathbb{PG} = (p, \mathbb{G}, \mathbb{G}_T, e, g)$.
- Select distinct $z_1, \ldots, z_n \xleftarrow{\$} \mathbb{Z}_p^*$ such that for all $i \neq j$, $z_i A_i \not\equiv z_j A_j \bmod p$ or equivalently $g^{z_i A_i} \neq g^{z_j A_j}$, and compute $u_i = g^{z_i}$, $u_{i,j} = g^{z_i z_j}$, $i, j = 1, 2, \ldots, n$, $i \neq j$.
- Publish $\mathsf{SP} = (\mathbb{PG}, \{u_i\}_{i=1}^n, \{u_{i,j}\}_{\substack{i,j=1 \\ i \neq j}}^n)$.

MGroupGen: The decryptor takes as input a group attribute $\mathbf{G} = \{A_{i_1}, \ldots, A_{i_k}\} \subseteq \{A_1, \ldots, A_n\}$ for any $k \leq n$, the system parameter SP, and computes the group token $\mathcal{P}(\mathbf{G}) = (\prod_{l=1}^{k} u_{i_l}^{A_{i_l}}) g^r = w$ (say), where $r \xleftarrow{\$} \mathbb{Z}_p$ is the secret key of the group token. The decryptor sends $\mathcal{P}(\mathbf{G})$ to the encryptor and keeps r secret to itself.

MEncrypt: Taking as input an attribute $A_{i^*} \in \mathbb{A}$, a group token $\mathcal{P}(\mathbf{G}) = w$, a message $M \in \mathbb{G}_T$ and the system parameter SP, the encryptor prepares the ciphertext as follows:

- Choose $S \xleftarrow{\$} \mathbb{Z}_p$.
- Compute the ciphertext $C = (C_1, C_2) = (e(w/u_{i^*}^{A_{i^*}}, u_{i^*}^S) M, g^S)$ and send C to the decryptor.

MDecrypt: On input the ciphertext $C = (C_1, C_2)$, the secret key r, the attribute A_{i^*}, the group attribute $\mathbf{G} = \{A_{i_1}, \ldots, A_{i_k}\}$ and the system parameter SP, the decryptor proceeds as follows:

- Compute $\Lambda = (\prod_{\substack{l=1 \\ i_l \neq i^*}}^{k} u_{i^*,i_l}^{A_{i_l}}) u_{i^*}^r$.
- Retrieve the message as $M = C_1 / e(\Lambda, C_2)$.

• Correctness and Security Proof:

We have,

$$C_1/e(\Lambda, C_2) = Me(w/u_{i^*}^{A_{i^*}}, u_{i^*}^S)/e(\Lambda, g^S) = Me((\prod_{\substack{l=1 \\ i_l \neq i^*}}^{k} u_{i_l}^{A_{i_l}}) g^r, u_{i^*}^S)/e((\prod_{\substack{l=1 \\ i_l \neq i^*}}^{k} u_{i^*,i_l}^{A_{i_l}}) u_{i^*}^r, g^S)$$

$$= Me(g^\gamma, g^{z_{i^*} S})/e(g^\delta, g^S), \text{ where } \gamma = \sum_{\substack{l=1 \\ i_l \neq i^*}}^{k} z_{i_l} A_{i_l} + r \text{ and } \delta = z_{i^*} \sum_{\substack{l=1 \\ i_l \neq i^*}}^{k} z_{i_l} A_{i_l} + z_{i^*} r$$

$$= Me(g,g)^\eta / e(g,g)^\eta, \text{ where } \eta = z_{i^*} (\sum_{\substack{l=1 \\ i_l \neq i^*}}^{k} z_{i_l} A_{i_l} + r) S$$

$$= M.$$

Theorem 1 (Indistinguishability against Secret Key). *The above membership encryption scheme is* (t', q_k, ϵ')*-selectively secure against secret key under the assumption that the* ***SimSqDBDH*** *problem is* (t, ϵ)*-hard. Here,* $t' = t - O(q_k n t_e)$*,* q_k *is number of group token query made by the adversary and* $\epsilon' = \frac{\epsilon}{q_k}$*, where* t_e *denotes the average time of an exponentiation in* $\mathbb{G}$.

Proof. Suppose there exists an adversary who can (t', ϵ') break the membership encryption against secret key under the selective security model by a chosen-plaintext attack. We construct an algorithm $\mathcal{B}$ that solves the **SimSqDBDH** problem in time t with advantage ϵ. $\mathcal{B}$ interacts with the adversary as follows:

Initialization: Let $\mathbb{PG} = (p, \mathbb{G}, \mathbb{G}_T, e, g)$ be the pairing group and $\mathbb{A} = \{A_1, \ldots, A_n\}$ be the attribute universe. The adversary outputs the attribute A_{i^*} for challenge.
Setup: The algorithm $\mathcal{B}$ works as follows to simulate the system parameter.
Let $(g, g^a, g^b, e(g,g)^{c_1}, e(g,g)^{c_2})$ be the given instance of the **SimSqDBDH** problem.

- $\mathcal{B}$ sets $u_{i^*} = g^b$.
- For $i \neq i^*$, $\mathcal{B}$ randomly selects distinct $z_i \in \mathbb{Z}_p^*$ such that $g^{z_i A_i} \neq g^{z_j A_j} \neq g^{bA_{i^*}}$ for all $i \neq j \neq i^*$ and computes $u_i = g^{z_i}$.
- For $j = 1, 2, \ldots, n$,
  For $i = i^*$, $\mathcal{B}$ sets $u_{i^*,j} = (g^b)^{z_j}$, $j \neq i^*$and
  For $i \neq i^*$, $\mathcal{B}$ computes $u_{i,j} = g^{z_i z_j}$, $i \neq j$.

$\mathcal{B}$ sends the system parameter $\mathsf{SP} = (\mathbb{PG}, \{u_i\}_{i=1}^n, \{u_{i,j}\}_{\substack{i,j=1 \\ i \neq j}}^n)$ to the adversary.
Phase 1: The algorithm $\mathcal{B}$ randomly chooses k^* from $[1, q_k]$ and simulates the group tokens as follows:

- For a group token query on $\mathbf{G}_k$, if $k \neq k^*$, $\mathcal{B}$ runs the **MGroupGen** algorithm to generate $\mathcal{P}(\mathbf{G}_k)$.
- Otherwise, if $k = k^*$, $\mathcal{B}$ sets $\mathcal{P}(\mathbf{G}_{k^*}) = g^a$.

Challenge: The adversary returns $(\mathcal{P}(\mathbf{G}^*), M_0, M_1)$ for challenge. If $\mathbf{G}^* \neq \mathbf{G}_{k^*}$, $\mathcal{B}$ aborts; otherwise, $\mathcal{P}(\mathbf{G}^*) = g^a$. The algorithm $\mathcal{B}$ randomly chooses a coin $c \in \{0, 1\}$ and simulates the challenge ciphertext as follows:

$$C^* = (C_1^*, C_2^*) = (M_c e(g,g)^{c_1} e(g,g)^{-c_2 A_{i^*}}, g^a).$$

If $c_1 = a^2 b$ and $c_2 = ab^2$ then $C_1^* = M_c e(g,g)^{(a - bA_{i^*})ba}$ and hence C^* is a valid ciphertext. $\mathcal{B}$ sends the ciphertext C^* to the adversary.
Win: The adversary outputs $c' \in \{0, 1\}$. Then $\mathcal{B}$ outputs 1 if the adversary outputs $c' = c$, i.e., the adversary wins, and outputs 0 otherwise.

This completes the description of the simulation. If $c_1 = a^2 b$ and $c_2 = ab^2$, the challenge ciphertext is valid and the adversary will output $c' = c$ with probability $(\frac{1}{2} + \epsilon')$; otherwise, if c_1 and c_2 are random then the challenge ciphertext is also random and hence the adversary outputs $c' = c$ with probability $\frac{1}{2}$. The simulation is successful when $\mathbf{G}^* = \mathbf{G}_{k^*}$ which holds with probability $\frac{1}{q_k}$. Hence,

$$\begin{aligned}
\epsilon &= |\mathsf{Pr}[\mathcal{B}(g, g^a, g^b, e(g,g)^{a^2 b}, e(g,g)^{ab^2}) = 1] - \mathsf{Pr}[\mathcal{B}(g, g^a, g^b, e(g,g)^{c_1}, e(g,g)^{c_2}) = 1]| \\
&= |\mathsf{Pr}[\text{Adversary wins when } C^* \text{ is valid} \mid \text{the simulation is perfect }] \\
&\quad - \mathsf{Pr}[\text{Adversary wins when } C^* \text{ is random} \mid \text{the simulation is perfect }]| \\
&= \frac{|(\frac{1}{2} + \epsilon') - \frac{1}{2}|}{\frac{1}{q_k}} \implies \epsilon' = \frac{\epsilon}{q_k}.
\end{aligned}$$

In order to see the relationship between t and t', note that the simulation time is mainly dominated by the group token simulation and each group token requires $O(n)$ exponentiations in $\mathbb{G}$. □

Theorem 2 (Indistinguishability against Membership). *The membership encryption scheme introduced above is (t', ϵ')-selectively secure against membership under the assumption that the **SqDBDH** problem is (t, ϵ)-hard. Here, $t' = t - O(n^2 t_e)$ and $\epsilon' = \epsilon$, where t_e denotes the average time of an exponentiation in $\mathbb{G}$.*

Proof. Suppose there exist an adversary who can (t', ϵ') break the membership encryption against membership under selective security model. We construct an algorithm $\mathcal{B}$ that solves the **SqDBDH** problem in time t with advantage ϵ. $\mathcal{B}$ interacts with the adversary as follows:
Initialization: Let $\mathbb{PG} = (p, \mathbb{G}, \mathbb{G}_T, e, g)$ be the pairing group and $\mathbb{A} = \{A_1, \ldots, A_n\}$ be the attribute universe. The adversary outputs $(A_{i^*}, \mathbf{G}^*)$ for challenge where $A_{i^*} \notin \mathbf{G}^*$.
Setup: Let $(g^a, g^b, e(g,g)^{c_1})$ be the given instance of the **SqDBDH** problem. $\mathcal{B}$ generates the system parameter as follows:

- For $i = i^*$, $\mathcal{B}$ sets $u_{i^*} = g^a$.
- For $i \neq i^*$, $\mathcal{B}$ randomly chooses distinct $z_i \leftarrow \mathbb{Z}_p^*$ such that $g^{z_i A_i} \neq g^{z_j A_j} \neq g^{aA_{i^*}}$ for all $i \neq j \neq i^*$, and computes $u_i = g^{z_i}$.

– Also $\mathcal{B}$ sets $u_{i^*,j} = (g^a)^{z_j}, j \neq i^*$.
– For $i \neq i^*$, $\mathcal{B}$ computes $u_{i,j} = g^{z_i z_j}, i \neq j$.

$\mathcal{B}$ sends the system parameter $\mathsf{SP} = (\mathbb{PG}, \{u_i\}_{i=1}^n, \{u_{i,j}\}_{\substack{i,j=1 \\ i\neq j}}^n)$ to the adversary.

Challenge: The adversary returns $(\mathcal{P}(\mathbf{G}^*), \mathbf{S}^*, M_0, M_1)$ for challenge. Let the secret randomness or the secret key in computation of $\mathcal{P}(\mathbf{G}^*)$ as sent by the adversary be $\mathbf{S}^* = r$ and $\mathbf{G}^* = \{A_{i_1}, \ldots, A_{i_k}\}$. $\mathcal{B}$ randomly chooses a coin $c \in \{0,1\}$ and simulates the challenge ciphertext as follows:

$$C^* = (C_1^*, C_2^*) = (M_c e((\prod_{l=1}^{k} u_{i^*,i_l}^{A_{i_l}}) u_{i^*}^r, g^b) e(g,g)^{-c_1 A_{i^*}}, g^b).$$

If $c_1 = a^2 b$ then $C_1^* = M_c e(g,g)^\eta$, where $\eta = a(\sum_{l=1}^{k} z_{i_l} A_{i_l} + r - a A_{i^*})b$, which implies that C^* is a valid ciphertext on M_c for $(A_{i^*}, \mathcal{P}(\mathbf{G}^*))$. $\mathcal{B}$ sends the ciphertxet C^* to the adversary.
Win: The adversary outputs $c' \in \{0,1\}$, and the algorithm $\mathcal{B}$ outputs 1 if $c' = c$, i.e., the adversary wins, and outputs 0 otherwise.

This completes the description of our simulation. If $c_1 = a^2 b$, the challenge ciphertext is valid and the adversary will output $c' = c$ with probability $(\frac{1}{2} + \epsilon')$; otherwise, the challenge ciphertext is universally random and the adversary outputs $c' = c$ with probability $\frac{1}{2}$. Hence,

$$\begin{aligned}
\epsilon &= |\mathsf{Pr}[\mathcal{B}(g, g^a, g^b, e(g,g)^{a^2 b}) = 1] - \mathsf{Pr}[\mathcal{B}(g, g^a, g^b, e(g,g)^{c_1}) = 1]| \\
&= |\mathsf{Pr}[\text{adversary wins when the challenge ciphertext is valid}] \\
&\quad - |\mathsf{Pr}[\text{adversary wins when the challenge ciphertext is random}]| \\
&= |(\frac{1}{2} + \epsilon') - \frac{1}{2}| \implies \epsilon' = \epsilon.
\end{aligned}$$

In order to see the relationship between t and t', note that the simulation time is mainly dominated by the $u_{i,j}$ simulation each of which takes 1 exponentiation and there are $O(n^2)$ such $u_{i,j}$'s. □

Theorem 3 (Privacy). *$\mathcal{P}(\boldsymbol{G})$ unconditionally preserves the privacy of all attributes in $\boldsymbol{G}$.*

Proof. Let $\mathcal{P}(\mathbf{G}_0)$ be the group token generated from $\mathbf{G}_0 = \{A_{i_1}, \ldots, A_{i_{k_1}}\}$ and with secret randomness (or the secret key) r. Then $\mathcal{P}(\mathbf{G}_0) = g^\gamma$, where $\gamma = \sum_{l=1}^{k_1} z_{i_l} A_{i_l} + r$. Let $\mathbf{G}_1 = \{A_{j_1}, \ldots, A_{j_{k_2}}\}$. Since there exists $r' \in \mathbb{Z}_p$ such that, $\sum_{l=1}^{k_1} z_{i_l} A_{i_l} + r = \sum_{l=1}^{k_2} z_{j_l} A_{j_l} + r' (\text{mod } p)$, $\mathcal{P}(\mathbf{G}_0)$ can also be seen as a group token generated from $\mathbf{G}_1$ with secret randomness r'. Thus, given $\mathcal{P}(\mathbf{G}_c)$, $c \in \{0,1\}$ no adversary can distinguish whether $c = 0$ or $c = 1$ with positive advantage for any $\mathbf{G}_0$ and $\mathbf{G}_1$. □

Remark 1. As in [18], the membership encryption described above is secure against chosen-plaintext attack (CPA). Let $\mathcal{ME}[\sigma, r]$ be the ciphertext on $\sigma \in \mathbb{G}_T$ encrypted with the randomness $r \in \mathbb{Z}_p$. Using the Fujisaki-Okamoto approach [15] in the random oracle model, our scheme can also be extended to the security against chosen-ciphertext attack (CCA). Let $I : \mathbb{G}_T \to \{0,1\}^*$ be an efficiently computable injective map and $H_1 : \{0,1\}^* \to \mathbb{Z}_p$, $H_2 : \{0,1\}^* \to \{0,1\}^{l_m}$ be cryptographic hash functions, where $\{0,1\}^{l_m}$ denotes the new message space. If $\mathcal{ME}[\sigma, r]$ is secure against CPA then the membership encryption construction $\mathcal{ME}[\sigma, H_1(A, \mathcal{P}(\mathbf{G}), I(\sigma), M)],\ H_2(I(\sigma)) \oplus M$, for $(A, \mathcal{P}(\mathbf{G}))$, is secure against CCA.

• **Efficiency:** Table 1 presents the computation and communication complexities of our membership encryption in comparison with that of [18]. Our construction has significantly less cost than [18] both in terms of computation and communication. In Particular, the number of exponentiations is much smaller in our scheme.

Note 2. Our membership encryption scheme has the advantage that, unlike [18] our scheme does not involve the attributes explicitly in the setup parameter. As a result, the same setup can be used for performing the operation using different universe of attributes $\mathbb{A}$ with the only restriction that $|\mathbb{A}| = n$ and A_i's are such that for all $i \neq j$, $g^{z_i A_i} \neq g^{z_j A_j}$ holds. Precisely, assume that initially the setup parameter is constructed for some attribute universe $\mathbb{A} = \{A_1, \ldots, A_n\}$ and later we want to use the

Table 1: Comparison Summary

Membership encryption	Number of exponentiation	Number of pairing	Public key size	Group token size	Ciphertext size
[18]	$(n^2+3n+4k+10)$ in $\mathbb{G}$, 2 in $\mathbb{G}_T$	3	(n^2+3n+3) in $\mathbb{G}$, 1 in $\mathbb{G}_T$	3 in $\mathbb{G}$	2 in $\mathbb{G}$, 1 in $\mathbb{G}_T$
Ours	$((n^2+n)/2+2k+3)$ in $\mathbb{G}$	2	$((n^2+n)/2)$ in $\mathbb{G}$	1 in $\mathbb{G}$	1 in $\mathbb{G}$, 1 in $\mathbb{G}_T$

Here, k denotes the size of the group attribute.

same setup for a different attribute universe $\mathbb{A}' = \{A'_1, \ldots, A'_n\}$. Thus initially $g^{z_i A_i} \neq g^{z_j A_j}$ for all $i \neq j$ and now we require $g^{z_i A'_i} \neq g^{z_j A'_j}$ for all $i \neq j$. Note that if the condition is violated for some $i \neq j$, we can slightly modify A'_i or A'_j, e.g., set $A'_i = A'_i + 1$ etc., to overcome such violation. In this sense our scheme is more flexible than that of [18] which can be applied only with an universe of attributes fixed before generation of the system parameters. This property makes our scheme particularly suitable for application in POT where item prices may change with time.

Note 3. Our scheme, if applied with a constant universe of attributes, can be made much more simple and efficient. Note that, if $\mathbb{A}$ is fixed then we can set $u_i = g^{z_i A_i}$ and $u_{i,j} = g^{z_i A_i z_j A_j}, i \neq j$. In that case the group token $\mathcal{P}(\mathbf{G})$ will be $(\prod_{l=1}^{k} u_{i_l}) g^r$ $(= w)$, the ciphertext will be $C = (C_1, C_2) = (e(w/u_{i^*}, u_{i^*})^S M, g^S)$, and to decrypt C we need to compute $M = C_1 / e((\prod_{\substack{l=1 \\ i_l \neq i^*}}^{k} u_{i^*,i_l}) u_{i^*}^r, C_2)$. Thus if $\mathbb{A}$ is fixed for the entire operation, then the **MGroupGen**, **MEncrypt** and **MDecrypt** algorithms will each require only a single exponentiation resulting in a scheme with constant computation complexity. The security argument will be similar for the modified scheme. On the contrary, the scheme of [18] is applicable only with a fixed universe of attributes and has quite large computation complexity.

4 Our Priced Oblivious Transfer

In this section, we show how to construct an efficient 1-out-of-n priced oblivious transfer protocol (POT) from our membership encryption. Our 1-out-of-n POT scheme is inspired from the k-out-of-n POT scheme of [19]. To construct the POT we employ the Groth-Sahai proof system [17] for the **DLIN** instance (Appendix A.1), the P-Signature scheme [3] (Appendix A.2) and the range prove [19] (Appendix A.3) with our membership encryption. In our scheme, each transaction (a single 'buy' operation) requires two passes of communication: (1) a message from the buyer; (2) the vendor's reply. This is optimal since even without privacy the buyer still needs to specify the item he wants to retrieve and the vendor needs to send this item. Also our scheme *allows more than one items to have the same price.*

Protocol requirements: Our scheme is parameterized with integers (n, l) (for the number of messages and their length), p_{max} (the upper bound for the prices) and $A = d^a$ (the upper bound for the deposit). This scheme is built on a pairing group setup $(p, \mathbb{G}, \mathbb{G}_T, e, g)$ such that $p_{max} < A \pmod p$ holds. As in [1], [19], we develop a prepaid scheme, where in the initialization phase the buyer $\mathcal{B}$ pays an initial deposit ac_0 to the vendor $\mathcal{V}$ and in subsequent transfer phases this deposit is subtracted by the price p_{σ_t} of the message M_{σ_t} that is being bought. The message space is $\{0,1\}^l$, but we abuse notation and write M_i to denote the corresponding group element in $\mathbb{G}_T$ assuming the existance of some efficient and invertible mapping. Also the prices p_i and the deposit ac_0 are considered as elements of $\mathbb{Z}_p$.

Informally speaking, in our POT scheme, the buyer $\mathcal{B}$ communicates with the vendor $\mathcal{V}$ in an initialization phase to make a prepayment, followed by several transfer phases. In each transfer the vendor chooses a set of messages $\{M_1, \ldots, M_n\}$ with prices $\{p_1, \ldots, p_n\}$ for sale and publishes an information Pub regarding those messages. Pub contains specification of $\{p_1, \ldots, p_n\}$. Then the buyer chooses to buy a particular message M_{σ_t} and sends a request Q to $\mathcal{V}$. The vendor $\mathcal{V}$ then returns a response R to $\mathcal{B}$ from which $\mathcal{B}$ retrieves M_{σ_t}.

The POT scheme must ensure that $\mathcal{V}$ learns neither the price of the message being queried nor the new value of the account, while $\mathcal{B}$ pays the correct price for the message updating the balance honestly and that he has enough funds to buy it. To achieve this, in the initialization phase $\mathcal{B}$ sends the deposit ac_0 from which $\mathcal{V}$ prepares a commitment on it. In the t-th transfer, $\mathcal{B}$ sends a commitment to the new value of

the account ac_t and proves that (1) this value is correct, i.e., that $\mathsf{ac}_t = \mathsf{ac}_{t-1} - p_{\sigma_t}$ and that (2) it is non-negative. In order to allow for (1), we need to ensure that $\mathcal{B}$ uses the correct price. To accomplish this, $\mathcal{V}$ uses membership encryption to encrypt each message M_i with its proper price p_i and the group token sent by $\mathcal{B}$. To ensure that this group token is honestly computed, $\mathcal{B}$ proves non-interactively the possession of a P-Signature [3] provided by the trusted authority as part of the common reference string on the component of the common reference string that has been used in forming the group token. Then $\mathcal{V}$ sends all these computed ciphertext to $\mathcal{B}$. Now the property called 'indistinguishability against membership' of the membership encryption scheme ensures that $\mathcal{B}$ can decrypt the ciphertext C_{σ_t} to obtain M_{σ_t} if $\mathcal{B}$ has paid the correct price p_{σ_t}. To achieve (2), $\mathcal{V}$ computes in the initialization phase parameters of the range proof [19] namely $params_{Range}$ and hands them to $\mathcal{B}$. In each transfer phase, $\mathcal{B}$ proves that the new value of the account $\mathsf{ac}_t \in [0, \ldots, A)$. We formally describe our POT scheme below.

Initialization phase: At time $t = 0$, on input $(sid, \mathcal{V}, n)$ for the vendor $\mathcal{V}$ and $(sid, \mathcal{B}, n, \mathsf{ac}_0)$ for the buyer $\mathcal{B}$,

1. $\mathcal{V}$ queries $\mathcal{F}_{\mathsf{CRS}}$ with $(sid, \mathcal{V}, \mathcal{B}, n)$. $\mathcal{F}_{\mathsf{CRS}}$ generates crs by running $\mathsf{POTGenCRS}(1^\lambda, p_{max}, A, n)$, as discussed below, and sends (sid, crs) to $\mathcal{V}$.
   $\mathsf{POTGenCRS}(1^\lambda, p_{max}, A, n)$: Given security parameter 1^λ and the total number of messages n,
   - $\mathcal{F}_{\mathsf{CRS}}$ generates a Groth-Sahai reference string crs_{PoK} under DLIN instance for the pairing group setup $(p, \mathbb{G}, \mathbb{G}_T, e, g)$ such that $p_{max} < A \pmod p$ holds, i.e., $crs_{PoK} = (t_1, t_2, t_3)$, where $t_1 = (g^\alpha, 1, g), t_2 = (1, g^\beta, g), t_3 = (g^{r\alpha}, g^{s\beta}, g^{r+s}) = (y_1, y_2, y_3)$, say, where $\alpha, \beta \xleftarrow{\$} \mathbb{Z}_p^*$ and $r, s \xleftarrow{\$} \mathbb{Z}_p$.
   - $\mathcal{F}_{\mathsf{CRS}}$ picks distinct random $z_1, \ldots, z_n \in \mathbb{Z}_p^*$ and computes $u_i = g^{z_i}, u_{i,j} = g^{z_i z_j}$, $i, j = 1, 2, \ldots, n$, $i \neq j$. $\mathcal{F}_{\mathsf{CRS}}$ also computes $v_i = y_3^{z_i}, i = 1, 2, \ldots, n$.
   - $\mathcal{F}_{\mathsf{CRS}}$ chooses random $u \leftarrow \mathbb{G}$ and computes $q_i = u^{z_i}, i = 1, \ldots, n$.
   - $\mathcal{F}_{\mathsf{CRS}}$ runs $\mathbf{PKeyGen}(crs_{Sig})$ of the $\mathbf{P}$-Signature scheme, discussed in Appendix A.2, where $crs_{Sig} = (crs_{PoK}, u)$ to get a signing-verification key pair $(sk = (\gamma, \delta), pk = (g^\gamma, g^\delta))$, $\gamma, \delta \xleftarrow{\$} \mathbb{Z}_p$, and for all $i \in \{1, \ldots, n\}$, it computes $s_i = \mathbf{PSign}(crs_{Sig}, sk, z_i)$.
   - $\mathcal{F}_{\mathsf{CRS}}$ sets $crs = (crs_{PoK}, \{u_i\}_{i=1}^n, \{u_{i,j}\}_{\substack{i,j=1 \\ i \neq j}}^n, \{v_i\}_{i=1}^n, u, pk, \{q_i\}_{i=1}^n, \{s_i\}_{i=1}^n)$. We mention that v_i's help $\mathcal{B}$ to decrypt the ciphertext sent by $\mathcal{V}$ and (u_i, q_i, s_i) is used to construct a non-interactive proof of possession of the P-Signature s_i on z_i by $\mathcal{B}$ without knowing z_i, as explained in Appendix A.2, in the transfer phase.
2. $\mathcal{B}$ queries $\mathcal{F}_{\mathsf{CRS}}$ with $(sid, \mathcal{V}, \mathcal{B}, n)$. $\mathcal{F}_{\mathsf{CRS}}$ sends (sid, crs) to $\mathcal{B}$. This crs is the same as that generated by $\mathcal{F}_{\mathsf{CRS}}$ following the procedure $\mathsf{POTGenCRS}$.
3. $\mathcal{V}$ executes the following procedure $\mathsf{POTInitVendor}(crs, A)$ to obtain $params_{Range}$ and sends $(sid, params_{Range})$ to $\mathcal{B}$.
   $\mathsf{POTInitVendor}(crs, A)$: Taking input crs and A, $\mathcal{V}$ works as follows:
   - $\mathcal{V}$ parses the crs to obtain $crs_{Sig} = (crs_{PoK}, u)$.
   - $\mathcal{V}$ runs $\mathsf{RPInitVerifier}(crs_{Sig}, A)$, discussed in Appendix A.3, to obtain $params_{Range}$.
4. Upon receiving $(sid, params_{Range})$ from $\mathcal{V}$, the buyer $\mathcal{B}$ computes $(P, D_0^{(Priv)})$ by invoking $\mathsf{POTInitBuyer}$ on input $(crs, params_{Range}, \mathsf{ac}_0)$ as follows.
   $\mathsf{POTInitBuyer}(crs, params_{Range}, \mathsf{ac}_0)$: On input $params_{Range}$ and a deposit $\mathsf{ac}_0 \in [0, \ldots, A)$;
   - $\mathcal{B}$ parses crs to obtain $crs_{Sig} = (crs_{PoK}, u)$.
   - $\mathcal{B}$ runs $\mathsf{RPInitProver}(crs_{Sig}, params_{Range})$, discussed in Appendix A.3 , to verify $params_{Range}$.
   - If the above check fails, $\mathcal{B}$ outputs reject. Otherwise, $\mathcal{B}$ sets $P = \mathsf{ac}_0$ and $D_0^{(Priv)} = (\mathsf{ac}_0, open_{\mathsf{ac}_0} = (0, 0, 0))$.

   $\mathcal{B}$ aborts if the output is reject. Otherwise, $\mathcal{B}$ sends (sid, P) to $\mathcal{V}$ and keeps $D_0^{(Priv)}$ secret to itself. Note that, $\mathcal{B}$ also needs to pay an amount of ac_0 to $\mathcal{V}$ through an arbitrary payment channel.
5. After getting the initial deposit money, $\mathcal{V}$ runs the procedure $\mathsf{POTGetDeposit}(crs, P, A)$ described below to check that ac_0 corresponds to amount of money received.
   $\mathsf{POTGetDeposit}(crs, P, A)$: Receiving P from $\mathcal{B}$, $\mathcal{V}$ works as follows:
   - $\mathcal{V}$ checks that $\mathsf{ac}_0 \in [0, \ldots, A)$
   - $\mathcal{V}$ sets $D_0 = \mathsf{Commit}(g^{\mathsf{ac}_0}, open_{\mathsf{ac}_0} = (0, 0, 0)) = (1, 1, g^{\mathsf{ac}_0})$ as explained in Appendix A.1.
6. $\mathcal{V}$ stores state information $V_0 = (params_{Range}, D_0)$ and $\mathcal{B}$ stores state information $B_0 = (params_{Range}, D_0^{(Priv)})$.

Transfer phase: At time $t > 0$, $\mathcal{V}$ with state information V_{t-1} and input $(sid, \mathcal{V}, \{M_1, M_2, \ldots, M_n\}, \{p_1, p_2, \ldots, p_n\})$ and $\mathcal{B}$ with state information B_{t-1} and input $(sid, \mathcal{B}, \{p_1, p_2, \ldots, p_n\}, \sigma_t)$ interact as follows. Here prices p_i of messages M_i are such that, for all $i \neq j$, $g^{z_i p_i} \neq g^{z_j p_j}$. Note that this can always be done while selecting p_i's. For instance, if it is found that $g^{z_i p_i} = g^{z_j p_j}$ for some $i \neq j$, then one can choose $p_i = p_i + 1$ etc.

1. $\mathcal{B}$ invokes $\mathsf{POTRequest}(crs, params_{Range}, \{p_1, \ldots, p_n\}, D_{t-1}^{(Priv)}, \sigma_t)$ to set a request Q and to generate private state $(Q^{(Priv)}, D_t^{(Priv)})$ as detailed below. $\mathcal{B}$ sends $(sid,\ Q)$ to $\mathcal{V}$ and stores $(sid,\ Q^{(Priv)}, D_t^{(Priv)})$ as private information.
   $\mathsf{POTRequest}(crs, params_{Range}, \{p_1, \ldots, p_n\}, D_{t-1}^{(Priv)}, \sigma_t)$: Taking input set of prices $\{p_1, \ldots, p_n\}$ and a selection value $\sigma_t \in \{1, \ldots, n\}$, $\mathcal{B}$ proceeds as follows:
   - $\mathcal{B}$ parses crs to obtain $crs_{PoK},\ u,\ \{u_i\}_{i=1}^n,\ pk,\ \{q_i\}_{i=1}^n,\ \{s_i\}_{i=1}^n$.
   - $\mathcal{B}$ parses $D_{t-1}^{(Priv)}$ as $(\ \mathsf{ac}_{t-1},\ open_{\mathsf{ac}_{t-1}} = (l_{t-1,1}, l_{t-1,2}, l_{t-1,3}))$, where $(l_{t-1,1}, l_{t-1,2}, l_{t-1,3}) \in \mathbb{Z}_p^3$, and computes $D_{t-1} = \mathsf{Commit}(g^{\mathsf{ac}_{t-1}}, open_{\mathsf{ac}_{t-1}}) = (g^{\alpha(l_{t-1,1}+rl_{t-1,3})}, g^{\beta(l_{t-1,2}+sl_{t-1,3})}, g^{l_{t-1,1}+l_{t-1,2}+(r+s)l_{t-1,3}} g^{\mathsf{ac}_{t-1}})$.
   - $\mathcal{B}$ also picks fresh $open_{p_{\sigma_t}} = (a_1, a_2, a_3), open_{z_{\sigma_t}} = (b_1, b_2, b_3), open_{z_{\sigma_t} p_{\sigma_t}} = (r_1, r_2, r_3), open_{\mathsf{ac}_t} = (l_{t,1}, l_{t,2}, l_{t,3})$ randomly from $\mathbb{Z}_p^3$, where $\mathsf{ac}_t = \mathsf{ac}_{t-1} - p_{\sigma_t}$ and computes $D_t = \mathsf{Commit}(g^{\mathsf{ac}_t}, open_{\mathsf{ac}_t})$, $h_1 = \mathsf{Commit}(g^{p_{\sigma_t}}, open_{p_{\sigma_t}})$, $h_2 = \mathsf{Commit}(u_{\sigma_t}, open_{z_{\sigma_t}})$, $h_3 = \mathsf{Commit}(g^{z_{\sigma_t} p_{\sigma_t}}, open_{z_{\sigma_t} p_{\sigma_t}})$ as above for D_{t-1}.
   - $\mathcal{B}$ runs $\mathsf{PoKProve}$ on input crs_{PoK} to compute a witness indistinguishable proof pok_t following approaches discussed in Appendix A.1:

$$\mathsf{NIPK}\{(g^{\mathsf{ac}_t}, g^{\mathsf{ac}_{t-1}}, g^{p_{\sigma_t}}, u_{\sigma_t}, g^{z_{\sigma_t} p_{\sigma_t}}, q_{\sigma_t}, s_{\sigma_t}) : 0 \le \mathsf{ac}_t < A \wedge e(g, g^{\mathsf{ac}_{t-1}}) e(g^{-1}, g^{\mathsf{ac}_t}) e(g^{-1}, g^{p_{\sigma_t}}) = 1$$
$$\wedge\ e(g, g^{z_{\sigma_t} p_{\sigma_t}}) e(u_{\sigma_t}, g^{p_{\sigma_t}})^{-1} = 1\ \wedge\ \mathbf{PVerifySig}(pk, s_{\sigma_t}, z_{\sigma_t}) = \mathsf{accept}\ \wedge\ \mathsf{ac}_t \text{ in } D_t$$
$$\wedge\ \mathsf{ac}_{t-1} \text{ in } D_{t-1}\ \wedge\ p_{\sigma_t} \text{ in } h_1\ \wedge\ u_{\sigma_t} \text{ in } h_2\ \wedge\ g^{z_{\sigma_t} p_{\sigma_t}} \text{ in } h_3\} \quad (1)$$

     Note that pok_t includes range proof $\mathsf{NIPK}\{(g^{\mathsf{ac}_t}) : 0 \le \mathsf{ac}_t < A\} = \mathsf{NIPK}\{(g^{\mathsf{ac}_t}, \{g^{\alpha_j}, u^{\alpha_j}, s'_{\alpha_j}\}_{j=0}^{a-1}) : \{\mathbf{PVerifySig}(pk', s'_{\alpha_j}, \alpha_j) = \mathsf{accept}\}_{j=0}^{a-1}\ \wedge e(g, g^{\mathsf{ac}_t}) \prod_{j=0}^{a-1} e(g^{-d^j}, g^{\alpha_j}) = 1\ \wedge\ g^{\mathsf{ac}_t} \text{ in } D_t\}$, for ac_t committed in D_t, where pk' and $\{g^{\alpha_j}, u^{\alpha_j}, s'_{\alpha_j}\}_{j=0}^{a-1}$ are contained in $params_{Range}$ sent by $\mathcal{V}$ during the initialization phase and $\mathsf{ac}_t = \sum_{j=0}^{a-1} \alpha_j d^j$. Also pok_t contains a non-interactive proof of possession of the P-Signature s_{σ_t} on z_{σ_t} using u_{σ_t} and q_{σ_t} following equation 3 of Appendix A.2.
   - $\mathcal{B}$ sets $Q = (h_1, h_2, h_3, pok_t, D_t)$, $Q^{(Priv)} = (\sigma_t, open_{z_{\sigma_t} p_{\sigma_t}})$ and $D_t^{(Priv)} = (\mathsf{ac}_t, open_{\mathsf{ac}_t})$.
2. Upon receiving $(sid,\ Q)$ from $\mathcal{B}$, $\mathcal{V}$ executes $\mathsf{POTRespond}$ on input $(crs, \{M_1, \ldots, M_n\}, \{p_1, \ldots, p_n\}, D_{t-1}, Q)$ to obtain a response R and state D_t as explained below. $\mathcal{V}$ sends $(sid,\ R)$ to $\mathcal{B}$ and stores (sid, D_t) as private information.
   $\mathsf{POTRespond}(crs, \{M_1, \ldots, M_n\}, \{p_1, \ldots, p_n\}, parms_{Range}, D_{t-1}, Q)$: Taking input $params_{Range}$, a set of messages $\{M_1, \ldots, M_n\}$ with prices $\{p_1, \ldots, p_n\}$, private state D_{t-1} and a request Q, $\mathcal{V}$ works as follows:
   - $\mathcal{V}$ parses crs to obtain $(crs_{PK}, u, \{u_i\}_{i=1}^n, pk)$, Q to obtain $(h_1, h_2, h_3, pok_t, D_t)$.
   - $\mathcal{V}$ verifies pok_t by running $\mathsf{PoKVarify}$ on input crs_{PoK} and it aborts if the output is reject. For this verification, $\mathcal{V}$ uses the commitments $(h_1, h_2, h_3, D_{t-1}, D_t)$. For a clear insight regarding such verifications the reader is refer to Appendix A.1.
   - $\mathcal{V}$ parses $h_3 = (w_1, w_2, w_3) = (g^{\alpha(r_1+rr_3)}, g^{\beta(r_2+sr_3)}, g^{z_{\sigma_t} p_{\sigma_t}} g^{r_1+r_2+(r+s)r_3})$ where $open_{z_{\sigma_t} p_{\sigma_t}} = (r_1, r_2, r_3) \in \mathbb{Z}_p^3$ is not known to $\mathcal{V}$. Note that w_3 can be written as $w_3 = g^{z_{\sigma_t} p_{\sigma_t}} g^b$, where $b = r_1 + r_2 + (r+s)r_3 \in \mathbb{Z}_p$ is random since $open_{z_{\sigma_t} p_{\sigma_t}} = (r_1, r_2, r_3) \in \mathbb{Z}_p^3$ is random. Thus w_3 can be viewed as a group token $\mathcal{P}(\mathbf{G})$ for group attribute $\mathbf{G} = \{p_{\sigma_t}\}$ according to our membership encryption introduced in Section 3. However, in this case we are only able to guarantee computational privacy of the group token since the group token in this case is part of a commitment of the Groth-Sahai proof system which has computational witness indistinguishability, whereas, in the original membership encryption scheme, the privacy of group tokens is unconditional.
   - For $i = 1, \ldots, n$, $\mathcal{V}$ selects random $S_i \leftarrow \mathbb{Z}_p$ and computes $C_i = (C_i^{(1)}, C_i^{(2)}) = (e(w_3/u_i^{p_i}, u_i^{S_i}) M_i,\ g^{S_i})$. Note that, C_i is essentially the membership encryption of M_i using p_i and the group token w_3 for $\{p_{\sigma_t}\}$.
   - $\mathcal{V}$ sets $R = (C_1, \ldots, C_n)$.
3. $\mathcal{B}$, on receiving $(sid,\ R)$ from $\mathcal{V}$, runs the following procedure $\mathsf{POTComplete}(crs, R, Q^{(Priv)})$ to obtain M_{σ_t}.
   $\mathsf{POTComplete}(crs, R, Q^{(Priv)})$: Taking input R and private state $Q^{(Priv)}$,
   - $\mathcal{B}$ extracts $(crs_{PoK}, \{u_i\}_{i=1}^n, \{v_i\}_{i=1}^n, \{u_{i,j}\}_{i,j=1, i \ne j}^n)$ from crs, parses R as $(C_1, \ldots, C_n)$, $Q^{(Priv)}$ as $(\sigma_t, open_{z_{\sigma_t} p_{\sigma_t}})$ where $open_{z_{\sigma_t} p_{\sigma_t}} = (r_1, r_2, r_3)$ is known to $\mathcal{B}$.

– $\mathcal{B}$ parses C_{σ_t} as $(C_{\sigma_t}^{(1)}, C_{\sigma_t}^{(2)})$ and it retrieves the message $M_{\sigma_t} = C_{\sigma_t}^{(1)}/e(u_{\sigma_t}^{r_1+r_2} v_{\sigma_t}^{r_3}, C_{\sigma_t}^{(2)})$.

4. $\mathcal{V}$ stores state information $V_t = (params_{Range}, D_t)$ and $\mathcal{B}$ stores state information $B_t = (params_{range}, D_t^{(Priv)})$ and outputs (sid, M_{σ_t}).

Theorem 4. *The priced oblivious transfer protocol described above securely realizes $\mathcal{F}_{\mathsf{POT}}$ under $\{max\{n,d\}\}$-**HSDH**, $\{max\{n,d\}\}$-**TDH**, **DLIN** and **SqDBDH** assumptions, where n = no. of messages and $A = d^a$ is the upper bound of the buyer's account.*

We prove Theorem 4 in Appendix B.

Remark 2. Note that, in the above protocol $\mathcal{B}$ can store D_t instead of $D_t^{(Prive)}$ in each transfer phase. However, $D_t^{(Prive)}$ will be needed when we will fit our scheme in a subscription setting (Appendix C). In a subscription setting $\mathcal{B}$ needs to store his remaining balance after each transaction. In case $\mathcal{B}$ wishes to terminate his subscription before his legitimate period expires, he will have to use $D_t^{(Prive)}$ corresponding to the last transaction to open the respective D_t to a trusted third party or the court of law in the process of claiming his remaining balance from $\mathcal{V}$.

• **A note on efficiency:** The common reference string of our POT protocol consists of $n^2+6n+12$ group elements. Regarding the communication complexity of this scheme, we note that, in the initialization phase $\mathcal{V}$'s message contains $3d+2$ group elements, which is the size of $params_{Range}$, and that of $\mathcal{B}$ involves a single element of $\mathbb{Z}_p$. In each transfer phase $\mathcal{B}$'s request Q is composed of $30a+57$ group elements and $\mathcal{V}$'s response R has $2n$ group elements. For the computational complexity of our POT scheme, observe that the initialization phase requires n^2+6n+7 exponentiations for $\mathcal{F}_{\mathsf{CRS}}$, $3d+3$ exponentiations for $\mathcal{V}$ and d exponentiations along with $3d$ pairings for $\mathcal{B}$. Further, each transfer phase involves 2 exponentiations, 1 pairings plus the complexity of constructing the NIPK (1), which involves the cost of generating a range proof, a non-interactive proof of a P-Signature possession and Groth-Sahai non-interactive proof of knowledge for two additional pairing product equations, for $\mathcal{B}$ and $3n$ exponentiations, $n+87a+174$ pairings for $\mathcal{V}$. We note that $A = d^a$ is the upper bound of buyer's account.

Remark 3. Observe that, since in our POT construction the membership encryption discussed in Section 3 is applied for singleton group attributes $\{p_{\sigma_t}\}$, $u_{i,j}$'s are not required for decrypting the ciphertext C_{σ_t}. Thus we can omit $\{u_{i,j}\}_{i\neq j}$ from the crs resulting in further reduction in crs size an well as the number of exponentiations computed by $\mathcal{F}_{\mathsf{CRS}}$ by n^2. This modification will not affect the security argument.

5 Conclusion

In this paper we have constructed a new efficient, provably secure membership encryption scheme and have applied it to develop an efficient 1-out-of-n POT protocol. Privacy preserving membership proof and encryption are both simultaneously used in many complex cryptographic protocols. Hence, it is really beneficial to have more efficient scheme that combines both of them in a single framework, i.e., to design efficient and practical membership encryption scheme. Our proposed membership encryption has constant length group token and constant ciphertext size both of which is shorter than that of [18]. Also unlike [18], our scheme is flexible in the sense that the same setup can be used with different universe of attributes. This property is important for applications such as POT where item prices may change with time. Moreover, computational complexity is constant, when applied with a fixed universe of attributes, making our scheme significantly more efficient than that of [18] which has quite large complexity. Our developed POT protocol is secure under universally composable framework and thus, unlike the existing 1-out-of-n schemes [1], [21] available in the literature, preserves security when it is executed with multiple protocol instances that run concurrently in an adversarily controlled way.Further, the protocol is round optimal having constant computation and communication cost on the buyer's side and $O(n)$ complexity on the vendor's side, which is so far the best known for 1-out-of-n POT.

References

1. Aiello, B., Ishai, Y., Reingold, O.: Priced oblivious transfer: How to sell digital goods. In: Advances in CryptologyEUROCRYPT 2001, pp. 119–135. Springer (2001)
2. Au, M.H., Tsang, P.P., Susilo, W., Mu, Y.: Dynamic universal accumulators for ddh groups and their application to attribute-based anonymous credential systems. In: Topics in Cryptology–CT-RSA 2009, pp. 295–308. Springer (2009)

3. Belenkiy, M., Chase, M., Kohlweiss, M., Lysyanskaya, A.: P-signatures and noninteractive anonymous credentials. In: Theory of Cryptography, pp. 356–374. Springer (2008)
4. Blum, M., Feldman, P., Micali, S.: Non-interactive zero-knowledge and its applications. In: Proceedings of the twentieth annual ACM symposium on Theory of computing. pp. 103–112. ACM (1988)
5. Boneh, D., Boyen, X.: Efficient selective-id secure identity-based encryption without random oracles. In: Advances in Cryptology-EUROCRYPT 2004. pp. 223–238. Springer (2004)
6. Boneh, D., Boyen, X., Shacham, H.: Short group signatures. In: Advances in Cryptology–CRYPTO 2004. pp. 41–55. Springer (2004)
7. Boneh, D., Franklin, M.: Identity-based encryption from the weil pairing. In: Advances in CryptologyCRYPTO 2001. pp. 213–229. Springer (2001)
8. Boyen, X., Waters, B.: Full-domain subgroup hiding and constant-size group signatures. In: Public Key Cryptography–PKC 2007, pp. 1–15. Springer (2007)
9. Camenisch, J., Chaabouni, R., et al.: Efficient protocols for set membership and range proofs. In: Advances in Cryptology-ASIACRYPT 2008, pp. 234–252. Springer (2008)
10. Camenisch, J., Dubovitskaya, M., Neven, G.: Unlinkable priced oblivious transfer with rechargeable wallets. In: Financial Cryptography and Data Security, pp. 66–81. Springer (2010)
11. Camenisch, J., Kohlweiss, M., Soriente, C.: An accumulator based on bilinear maps and efficient revocation for anonymous credentials. In: Public Key Cryptography–PKC 2009, pp. 481–500. Springer (2009)
12. Canetti, R.: Universally composable security: A new paradigm for cryptographic protocols. In: Foundations of Computer Science, 2001. Proceedings. 42nd IEEE Symposium on. pp. 136–145. IEEE (2001)
13. Cramer, R., Damgård, I., Schoenmakers, B.: Proofs of partial knowledge and simplified design of witness hiding protocols. In: Advances in CryptologyCRYPTO94. pp. 174–187. Springer (1994)
14. Damgård, I., Nielsen, J.B., Orlandi, C.: Essentially optimal universally composable oblivious transfer. In: Information Security and Cryptology–ICISC 2008, pp. 318–335. Springer (2009)
15. Fujisaki, E., Okamoto, T.: Secure integration of asymmetric and symmetric encryption schemes. In: Advances in CryptologyCRYPTO99. pp. 537–554. Springer (1999)
16. Green, M., Hohenberger, S.: Universally composable adaptive oblivious transfer. In: Advances in Cryptology-ASIACRYPT 2008, pp. 179–197. Springer (2008)
17. Groth, J., Sahai, A.: Efficient non-interactive proof systems for bilinear groups. In: Advances in Cryptology–EUROCRYPT 2008, pp. 415–432. Springer (2008)
18. Guo, F., Mu, Y., Susilo, W., Varadharajan, V.: Membership encryption and its applications. In: Information Security and Privacy. pp. 219–234. Springer (2013)
19. Rial, A., Kohlweiss, M., Preneel, B.: Universally composable adaptive priced oblivious transfer. In: Pairing-Based Cryptography–Pairing 2009, pp. 231–247. Springer (2009)
20. Rial, A., Preneel, B.: Optimistic fair priced oblivious transfer. In: Progress in Cryptology–AFRICACRYPT 2010, pp. 131–147. Springer (2010)
21. Tobias, C.: Practical oblivious transfer protocols. In: Information Hiding. pp. 415–426. Springer (2003)

A Some Necessary Cryptographic Tools

A.1 Non-interactive Zero-Knowledge Proofs of Knowledge of [17]

Let R be an efficiently computable relation and $L = \{y : R(y, w) = \mathsf{accept} \text{ for some } w\}$ be an NP-language. For tuples $(y, w) \in R$, we call y the instance and w the witness. A non-interactive proof of knowledge system consists of algorithms PoKSetup, PoKProve and PoKVerify. PoKSetup(1^λ) outputs a common reference string crs_{PoK}, PoKProve(crs_{PoK}, y, w) computes a proof of knowledge pok of instance y by using witness w and PoKVerify(crs_{PoK}, y, pok) outputs accept if pok is correct.

Zero-knowledge captures the notion that a verifier learns nothing from the proof except the truth of the statement. *Witness-indistinguishability* is a weaker property that guarantees that the verifier learns nothing about the witness that was used in the proof. In either case, we will also require *soundness*, meaning that an adversarial prover cannot convince an honest verifier of a false statement, and *completeness*, meaning that all correctly computed proofs are accepted by the honest verification algorithm. See [4] for formal definitions.

In addition, a proof of knowledge needs to be extractable. *Extractability* means that there exists a polynomial time extractor (PoKExtractSetup,PoKExtract). Algorithm PoKExtractSetup(1^λ) generates parameters crs_{PoK} that are identically distributed to the ones generated by algorithm PoKSetup and an extraction trapdoor td_{ext}. PoKExtract$(crs_{PoK}, td_{ext}, y, pok)$ extracts the witness w with all but negligible probability when PoKVerify(crs_{PoK}, y, pok) outputs accept.

We recall the notion of *f-extractability* defined by Belenkiy et al. [3]. In an f-extractable proof system the extractor PoKExtract extracts a value z such that $z = f(w) \wedge (y, w) \in R$ for some w. If $f(\cdot)$ is the identity function, we get the usual notion of extractability.

Commitment schemes: A non-interactive commitment scheme consists of algorithms $\mathsf{ComSetup}$ and Commit. $\mathsf{ComSetup}(1^\lambda)$ generates the parameters of the commitment scheme $params_{Com}$. The algorithm $\mathsf{Commit}(params_{Com}, x, open)$ outputs a commitment C to x using auxiliary information $open$. A commitment C is opened by revealing $(x, open)$ and checking $\mathsf{Commit}(params_{Com}, x, open) = C$. A commitment scheme has a *hiding* property and a *binding* property. Informally speaking, the hiding property ensures that a commitment C to x does not reveal any information about x, whereas the binding property ensures that C cannot be opened to another value x'. When it is clear from the context, we omit the commitment parameters $params_{Com}$.

A notation for f-extractable non-interactive proofs of knowledge (NIPK): We are interested in NIPK about (unconditionally binding) commitments. By 'x in C' we denote that there exists $open$ such that $C = \mathsf{Commit}(params_{Com}, x, open)$. After Belenkiy et al. [3], we use the following notation to express an f-extractable NIPK pok, on the instance $(C_1, \ldots, C_m, \mathsf{Condition})$ with witness $(x_1, open_1, \ldots, x_m, open_m, s)$ that allows to extract all the witnesses except the openings of the commitments. Here $\mathsf{Condition}$ stands for a constraint on $crs,\ x_1, \ldots, x_m,\ s$.

$$\mathsf{NIPK}\{(x_1, \ldots, x_m, s) : \mathsf{Condition}(crs, x_1, \ldots, x_m, s) \wedge x_1 \text{ in } C_1 \wedge \ldots \wedge x_m \text{ in } C_m\} \tag{2}$$

The f-extractability of a NIPK ensures that, with overwhelming probability over the choice of crs, we can extract $(x_1, \ldots, x_m, s)$ from pok, when $\mathsf{PoKVerify}$ accepts, x_i is contained in commitment C_i, where $1 \leq i \leq m$, and $\mathsf{Condition}(crs, x_1, \ldots, x_m, s)$ is satisfied. To further abbreviate this notation, we omit crs when it is understood from the context.

Applying the notation to Groth-Sahai proofs: We now illustrate below equation (2) by applying the notation to Groth-Sahai proofs [17] which allow proving statements about pairing product equations. The pairing group setup $(p, \mathbb{G}, \mathbb{G}_T, e, g)$ is part of the common reference string crs_{PoK} output by $\mathsf{PoKSetup}(1^\lambda)$ and the instance consists of the constants $\{\mathcal{A}_i\}_{i=1}^m \in \mathbb{G},\ t_T \in \mathbb{G}_T, \{\gamma_{i,j}\}_{i,j=1}^m \in \mathbb{Z}_p$ of the pairing product equation: $\prod_{i=1}^{m} e(\mathcal{A}_i, \mathcal{Y}_i) \prod_{i=1}^{m}\prod_{j=1}^{m} e(\mathcal{Y}_i, \mathcal{Y}_j)^{\gamma_{i,j}} = t_T$. The prover knows $\{\mathcal{Y}_i\}_{i=1}^m$ satisfying this equation.

Internally, Groth-Sahai proofs prove relations between commitments. A homomorphism guarantees that the same relations also hold for the committed values. Normally, as the first step in creating the proof, the prover prepares commitments $\{C_i\}_{i=1}^m$ for all values $\mathcal{Y}_i$ in $\mathbb{G}$. Then, the instance, known to the prover and the verifier, is the pairing product equation alone (i.e., its constants).

In addition, it is possible to add pre-existing Groth-Sahai commitments $\{C_i\}_{i=1}^n, n \leq m$, to the instance for some of the $\mathcal{Y}_i$ values. The corresponding openings $open_{\mathcal{Y}_i}$ become part of the witness. The proof will be computed in the same way, except that for values with existing commitments no fresh commitments need to be computed. We will write $C_i = \mathsf{Commit}(\mathcal{Y}_i, open_{\mathcal{Y}_i})$ to create Groth-Sahai commitments. Note that here Commit uses parameters contained in the crs_{PoK} of the Groth-Sahai proof systems. This proof system generates f-extractable witness indistinguishable[1] NIPK pok of the form $\mathsf{NIPK}\{(\mathcal{Y}_1, \ldots, \mathcal{Y}_n, \mathcal{Y}_{n+1}, \ldots, \mathcal{Y}_m) : \prod_{i=1}^{m} e(\mathcal{A}_i, \mathcal{Y}_i) \prod_{i=1}^{m}\prod_{j=1}^{m} e(\mathcal{Y}_i, \mathcal{Y}_j)^{\gamma_{i,j}} = t_T \ \wedge \mathcal{Y}_1 \text{ in } C_1 \ \wedge \ldots \wedge \mathcal{Y}_m \text{ in } C_m\}$.

In order to construct NIPK for a system of pairing product equations, a separate proof is to be computed for each equation. In [17], Groth and Sahai have given three different instantiations of their proof system. (In fact, their proposed proof system also works in asymmetric pairing groups.) Groth-Sahai proofs are extractable, composable witness-indistinguishable and composable zero-knowledge (given certain conditions). For definitions of these notions the reader is referred to [17].

Out of the three instantiations presented in [17], we will consider the one based on the **DLIN** assumption. In this instantiation, the common reference string crs_{PoK} is of the form $crs_{PoK} = (t_1, t_2, t_3)$, where $t_1 = (g^\alpha, 1, g) = (x_1, x_2, x_3)$ (say), $t_2 = (1, g^\beta, g) = (y_1, y_2, y_3)$ (say), $t_3 = (g^{r\alpha}, g^{s\beta}, g^{r+s}) = (z_1, z_2, z_3)$ (say), $\alpha, \beta \xleftarrow{\$} \mathbb{Z}_p^*,\ r, s \xleftarrow{\$} \mathbb{Z}_p$, where $\xleftarrow{\$}$ stands for random selection. The commitment on a value $\mathcal{Y}_i \in \mathbb{G}$ is $C_i = \mathsf{Commit}(\mathcal{Y}_i, open_{\mathcal{Y}_i}) = \left(g^{\alpha(r_{i,1}+rr_{i,3})}, g^{\beta(r_{i,2}+sr_{i,3})}, g^{r_{i,1}+r_{i,2}+(r+s)r_{i,3}}\mathcal{Y}_i\right)$, where $open_{\mathcal{Y}_i} = (r_{i,1}, r_{i,2}, r_{i,3}) \in \mathbb{Z}_p^3$. For a linear equation $\prod_{i=1}^{m} e(\mathcal{A}_i, \mathcal{Y}_i) = t_T$, the proof pok has the form $pok = \left(\prod_{i=1}^{m} \mathcal{A}_i^{r_{i,1}}, \prod_{i=1}^{m} \mathcal{A}_i^{r_{i,2}}, \prod_{i=1}^{m} \mathcal{A}_i^{r_{i,3}}\right)$ and for a quadratic equation $\prod_{i=1}^{m} e(\mathcal{A}_i, \mathcal{Y}_i) \prod_{i=1}^{m}\prod_{j=1}^{m} e(\mathcal{Y}_i, \mathcal{Y}_j)^{\gamma_{i,j}} = t_T$, pok consists of nine group elements in $\mathbb{G}$ of similar type.

[1] Some classes of pairing product equations also admit zero-knowledge proofs.

The verification of the above proof is done by checking three and nine pairing product equalities in linear and quadratic case respectively. Below we present a small example which is used in our POT construction.

Example 1. Consider the linear equation $e(g, \mathcal{Y}_1)e(g^{-1}, \mathcal{Y}_2)e(g^{-1}, \mathcal{Y}_3) = 1$. Let, $C_1 = \mathsf{Commit}(\mathcal{Y}_1, open_{\mathcal{Y}_1}) = (u_1, u_2, u_3)$, $C_2 = \mathsf{Commit}(\mathcal{Y}_2, open_{\mathcal{Y}_2}) = (v_1, v_2, v_3)$, $C_3 = \mathsf{Commit}(\mathcal{Y}_3, open_{\mathcal{Y}_3}) = (w_1, w_2, w_3)$. Further suppose that the proof $pok = (\pi_1, \pi_2, \pi_3)$. All of C_i's and pok are of the form as described above. The verification equalities for the above linear equation are $e(g, u_i)e(g^{-1}, v_i)e(g^{-1}, w_i) = e(\pi_1, x_i)e(\pi_2, y_i)e(\pi_3, z_i)$ for $i = 1, 2, 3$.

A.2 P-Signature Scheme of [3]

P-Signatures introduced by Belenkiy et al. [3] are signatures equipped with a common reference string crs_{Sig} and a NIPK that allows proving possession of a signature of a committed message. Belenkiy et al. show in [3] how to use the Groth-Sahai proof system to build this proof. Since in their construction $M \in \mathbb{Z}_p$ and Groth-Sahai proofs prove knowledge of a witness in $\mathbb{G}$, they need to compute a bijection $F(M) \in \mathbb{G}$ and prove knowledge of $F(M)$. The P-Signature scheme is said to be F-unforgeable if no p.p.t. adversary can output $(F(M), s)$ without previously obtaining a signature on M.

Below we present the P-Signature scheme of [3]. This P-Signature scheme is employed in the range proof discussed in Appendix A.3 and in our POT scheme.

PSetup: Taking as input a security parameter 1^λ, a trusted authority runs the Groth-Sahai PoKSetup with input 1^λ to obtain crs_{PoK} for pairing groups $(p, \mathbb{G}, \mathbb{G}_T, e, g)$, picks random $u \in \mathbb{G}$, and publishes $crs_{Sig} = (crs_{PoK}, u)$.

PKeygen: On input crs_{Sig}, the signer picks a secret key $sk = (\alpha, \beta) \xleftarrow{\$} \mathbb{Z}_p$ and computes a public key $pk = (v, w) = (g^\alpha, g^\beta)$.

PSign: The signer takes as input $(crs_{Sig}, sk, M \in \mathbb{Z}_p)$, picks random $r \xleftarrow{\$} \mathbb{Z}_p / \{\frac{\alpha - M}{\beta}\}$ and computes $s = (s_1, s_2, s_3) = (g^{1/(\alpha+M+\beta r)}, w^r, u^r)$.

PVerifySig: On input (crs_{Sig}, pk, M, s), the verifier outputs accept when $e(s_1, vg^M s_2) = e(g, g),\ e(u, s_2) = e(s_3, w)$. Otherwise, it outputs reject.

Using Groth-Sahai proofs, a NIPK of such a signature is constructed as follows. This is a proof of a pairing product equation of the form

$$\mathsf{NIPK}\{(g^M, u^M, s_1, s_2, s_3) : e(s_1, vg^M s_2) = e(g, g) \ \wedge e(u, s_2) = e(s_3, w) \ \wedge \ e(u, g^M) = e(u^M, g)\}. \quad (3)$$

We abbreviate this expression as $\mathsf{NIPK}\{(g^M, u^M, s) : \mathsf{PVerifySig}(pk, s, M) = \mathsf{accept}\}$. We would like to highlight the fact that to construct this NIPK the knowledge of g^M and u^M is sufficient, no need to know M explicitly. This scheme is F-unforgeable ($F(M) = (g^M, u^M)$) under the **HSDH** and the **TDH** assumption.

A.3 Non-Interactive Range Proof of [19]

We use the efficient non-interactive range proof proposed by Rial et al. [19] to prove that a committed value $\sigma \in \mathbb{Z}_p$ lies in an interval $[0, d^a)$ by representing σ in base d and employing P-Signature of [3] discussed in Appendix A.2.

$\mathsf{RPSetup}(1^\lambda)$: Given a security parameter 1^λ, a trusted third party executes $\mathbf{PSetup}(1^\lambda)$ to generate $crs_{Sig} = (crs_{PoK}, u)$.

$\mathsf{RPInitVerifier}(crs_{Sig}, A)$: The verifier takes as input $A = d^a$, and runs $\mathbf{PKeygen}(crs_{Sig})$ to get (sk, pk). Then for all $i \in \mathbb{Z}_d$, it computes $S_i = \mathbf{PSign}(crs_{Sig}, sk, i)$. It outputs $params_{Range} = (pk, \{S_i\}_{i \in \mathbb{Z}_d})$.

$\mathsf{RPInitProver}(crs_{Sig}, params_{Range})$: The prover parses $params_{Range}$ as $(pk, \{S_i\}_{i \in \mathbb{Z}_d})$. It verifies the signatures by running $\mathbf{PVerifySig}(crs_{Sig}, pk, S_i, i)$, for all $i \in \mathbb{Z}_d$. If these verifications succeed, it outputs accept. Otherwise it outputs reject.

$\mathsf{RangeProve}(crs_{Sig}, params_{Range}, g, \sigma, open_\sigma)$: The prover computes the following proof for a commitment $C_\sigma = \mathsf{Commit}(g^\sigma, open_\sigma)$: $\mathsf{NIPK}\{(g^\sigma, \{g^{\sigma_j}, u^{\sigma_j}, S_{\sigma_j}\}_{j=0}^{a-1}) : \{\mathbf{PVerifySig}(pk, S_{\sigma_j}, \sigma_j) = \mathsf{accept}\}_{j=0}^{a-1} \wedge e(g, g^\sigma) \prod_{j=0}^{a-1} e(g^{-d^j}, g^{\sigma_j}) = 1 \ \wedge \ g^\sigma \text{ in } C_\sigma\}$. The short form $\mathsf{NIPK}\{(g^\sigma) : 0 \leq \sigma < A \wedge g^\sigma \text{ in } C_\sigma\}$ is used to refer to this proof. This proof is only witness indistinguishable. While this is sufficient for our application, it is possible to make the proof zero-knowledge using techniques described in [17].

B Proof of Theorem 4

In order to prove this theorem, we need to build a simulator $\mathcal{E}$ that invokes a copy of adversary $\mathcal{A}$ and interacts with $\mathcal{F}_{\mathsf{POT}}$ and environment $\mathcal{Z}$ in such a way that ensembles $\mathsf{IDEAL}_{\mathcal{F}_{\mathsf{POT}},\mathcal{E},\mathcal{Z}}$ and $\mathsf{REAL}_{\mathsf{POT},\mathcal{A},\mathcal{Z}}$ are computationally indistinguishable.

In our proof we make use of the fact that Groth-Sahai proofs are extractable and composable witness-indistinguishable. As we deal with static security, the adversary needs to chose which party to corrupt before the protocol starts. We can, therefore, address the case of the malicious vendor and the malicious buyer separately.

Case (I). Security against malicious vendor:

1. $\mathcal{E}$ runs algorithm $\mathsf{PoKSetup}$ to generate a Groth-Sahai reference string crs_{PoK} under **DLIN** instance for a pairing group setup $(p, \mathbb{G}, \mathbb{G}_T, e, g)$, where $p_{max} < A \pmod p$ holds. $\mathcal{E}$ picks random $u \leftarrow \mathbb{G}$, selects distinct random $z_1, \ldots, z_n \leftarrow \mathbb{Z}_p^*$ and computes $u_i = g^{z_i}, u_{i,j} = g^{z_i z_j}, v_i = y_3^{z_i}, q_i = u^{z_i}, i, j = 1, \ldots, n, i \neq j$, where, $crs_{PoK} = (t_1, t_2, t_3)$ and $t_3 = (y_1, y_2, y_3)$, say. $\mathcal{E}$ runs $\mathbf{PKeyGen}(crs_{Sig})$ to obtain (pk, sk) where $crs_{Sig} = (crs_{PoK}, u)$. $\mathcal{E}$ computes $s_i = \mathbf{PSign}(crs_{Sig}, sk, z_i), i = 1, \ldots, n$. $\mathcal{E}$ sets $crs = (crs_{PoK}, \{u_i\}_i, \{u_{i,j}\}_{i \neq j}, \{v_i\}_i, u, pk, \{q_i\}_i, \{s_i\}_i)$. When $\mathcal{F}_{\mathsf{CRS}}$ is queried, $\mathcal{E}$ returns (sid, crs).
2. At time $t = 0$, upon receiving $(sid, params_{Range})$ from $\mathcal{A}$, $\mathcal{E}$ checks $params_{Range}$ as described in $\mathsf{POTInitBuyer}$ and aborts when the check fails. Otherwise $\mathcal{E}$ sends $(sid, \mathcal{V})$ to $\mathcal{F}_{\mathsf{POT}}$.
3. Upon receiving (sid, ac_0) from $\mathcal{F}_{\mathsf{POT}}$, $\mathcal{E}$ computes $(P, D_0^{(Priv)})$ by running $\mathsf{POTInitBuyer}$. $\mathcal{E}$ sends (sid, P) to $\mathcal{A}$ and keeps $D_0^{(Priv)}$.
4. At time $t > 0$, on publication of (sid, Pub) by $\mathcal{A}$, $\mathcal{E}$ notes the set of item prices $\{p_1, \ldots, p_n\}$ from Pub and sends $(sid, \mathcal{V}, \mathsf{Pub})$ to $\mathcal{F}_{\mathsf{POT}}$. Upon receiving $(sid, request)$ from $\mathcal{F}_{\mathsf{POT}}$, $\mathcal{E}$ executes $\mathsf{POTRequest}(crs, params_{Range}, \{p_1, \ldots, p_n\}, D_{t-1}^{(Priv)}, \sigma_{min})$, where $\sigma_t = \sigma_{min}$ corresponds to the index of the message with the lowest price for that time, to obtain $(Q, Q^{(Priv)}, D_t^{(Priv)})$ and sends (sid, Q) to $\mathcal{A}$. Observe that, since we use the item with the lowest price, $\mathcal{A}$ never rejects on the basis of not having enough funds. Upon receiving the response (sid, R) from $\mathcal{A}$, $\mathcal{E}$ parses R *as* $(C_1, C_2, \ldots, C_n)$, each $C_i = (C_i^{(1)}, C_i^{(2)})$ and computes $M_i = C_i^{(1)} / e(w_3 / u_i^{p_i}, C_i^{(2)})^{z_i}$ using the secret values z_i known to itself, where $Q = (h_1, h_2, h_3, pok_t, D_t)$ and $h_3 = \mathsf{Commit}(g^{z_{\sigma_{min}} p_{\sigma_{min}}}, open_{z_{\sigma_{min}} p_{\sigma_{min}}}) = (w_1, w_2, w_3)$ are as defined in the POT protocol of Section 4, i.e., w_3 is the group token for the group attribute $\{p_{\sigma_{min}}\}$ according to the membership encryption discussed in Section 3. $\mathcal{E}$ inputs $(sid, \{M_1, \ldots, M_n\})$ to $\mathcal{F}_{\mathsf{POT}}$.

Claim (Buyer's security). When $\mathcal{V}$ is corrupted, the ensembles $\mathsf{IDEAL}_{\mathcal{F}_{\mathsf{POT}},\mathcal{E},\mathcal{Z}}$ and $\mathsf{REAL}_{\mathsf{POT},\mathcal{A},\mathcal{Z}}$ are computationally indistinguishable under the **DLIN** assumption.

Proof. We show by means of a series of hybrid games that environment $\mathcal{Z}$ cannot distinguish between the real execution ensembles $\mathsf{REAL}_{\mathsf{POT},\mathcal{A},\mathcal{Z}}$ and the simulated ensembles $\mathsf{IDEAL}_{\mathcal{F}_{\mathsf{POT}},\mathcal{E},\mathcal{Z}}$ with non-negligible probability. We denote by $\mathsf{Pr}[\mathbf{Game}\ \mathrm{i}]$ the probability that $\mathcal{Z}$ distinguishes between the ensembles of **Game** i and that of the real execution.

Game 0: This game corresponds to the execution of the real world protocol with an honest $\mathcal{B}$. Therefore, $\mathsf{Pr}[\mathbf{Game}\ 0\ ] = 0$.

Game 1: This game differs from the previous game in that at each time $t > 0$, the request $Q = (h_1, h_2, h_3, pok_t, D_t)$ is computed by executing $\mathsf{POTRequest}(crs, params_{Range}, \{p_1, \ldots, p_n\}, D_{t-1}^{(Priv)}, \sigma_{min})$, where σ_{min} corresponds to the message with the lowest price for that time. Since $(h_1, h_2, h_3, pok_t, D_t)$ are computationally witness indistinguishable under the **DLIN** assumption, Q cannot be distinguished from a request computed by using another selection value $\sigma_t \in \{1, \ldots, n\}$. Therefore, $|\mathsf{Pr}[\mathbf{Game}\ 1] - \mathsf{Pr}[\mathbf{Game}\ 0]| \leq \epsilon(\lambda)$, where $\epsilon(\lambda)$ is a negligible function of the security parameter λ.

$\mathcal{E}$ performs the changes mentioned in **Game** 1 but for $i = 1, \ldots, n$, $\mathcal{E}$ uses the ciphertexts sent by $\mathcal{A}$ to compute messages $M_i = C_i^{(1)} / e(w_3 / u_i^{p_i}, C_i^{(2)})^{z_i}$ using the secret values z_i, where $(w_1, w_2, w_3) = \mathsf{Commit}(g^{z_{\sigma_{min}} p_{\sigma_{min}}}, open_{z_{\sigma_{min}} p_{\sigma_{min}}})$ and sends $(si, \{M_1, \ldots, M_n\})$ to $\mathcal{F}_{\mathsf{POT}}$. The crs generated by $\mathcal{E}$ is identically distributed as the real crs. Also upon receiving $P = \mathsf{ac}_0$ from $\mathcal{F}_{\mathsf{POT}}$, $\mathcal{E}$ computes and sends P by following $\mathsf{POTInitBuyer}$ and stores private state $D_0^{(Priv)}$ exactly as in real protocol. At time $t > 0$, $\mathcal{E}$ notes $\{p_1, \ldots, p_n\}$ from Pub published by $\mathcal{A}$ and sends Pub to $\mathcal{F}_{\mathsf{POT}}$. Further, $\mathcal{E}$

computes a request for σ_{min} by using $D_{t-1}^{(Priv)}$ and stores private state information $D_t^{(Priv)}$. Note that the distribution produced in **Game** 1 is identical to our simulation of the ideal ensemble. Therefore, $|\mathsf{Pr}[\textbf{Game } 1]| \leq \epsilon(\lambda)$. □

Case(II). Security against malicious buyer:

1. $\mathcal{E}$ sets $crs = (crs_{PoK}, \{u_i\}_i, \{u_{i,j}\}_{i\neq j}, \{v_i\}_i, u, pk, \{q_i\}_i, \{s_i\}_i)$ exactly in the same way as in Step 1 of Case (I) except that to construct crs_{PoK}, $\mathcal{E}$ executes algorithm $\mathsf{PoKExtractSetup}$ to obtain an extraction trapdoor td_{ext} along with crs_{PoK}. $\mathcal{E}$ returns (sid, crs) when $\mathcal{F}_{\mathsf{CRS}}$ is queried.
2. Upon receiving sid from $\mathcal{F}_{\mathsf{POT}}$, $\mathcal{E}$ runs $\mathsf{POTInitVendor}(crs, A)$ to obtain $params_{Range}$. $\mathcal{E}$ sends $(sid, params_{Range})$ to $\mathcal{A}$.
3. Upon receiving $(sid, P = \mathsf{ac}_0)$, $\mathcal{E}$ runs $D_0 \leftarrow \mathsf{POTGetDeposit}(crs, P, A)$, sends $(sid, \mathcal{B}, \mathsf{ac}_0)$ to $\mathcal{F}_{\mathsf{POT}}$ and stores D_0.
4. At time $t > 0$, after getting (sid, Pub) from $\mathcal{F}_{\mathsf{POT}}$, $\mathcal{E}$ notes $\{p_1, \ldots, p_n\}$ from Pub and publishes (sid, Pub) to $\mathcal{A}$. upon receiving (sid, Q) from $\mathcal{A}$, $\mathcal{E}$ parses Q as $(h_1, h_2, h_3, pok_t, D_t)$. Note that pok_t is of the form $\mathsf{NIPK}\{w : \mathsf{Condition}\}$ where w is the witness consisting of $(g^{\mathsf{ac}_t}, g^{\mathsf{ac}_{t-1}}, g^{p_{\sigma_t}}, u_{\sigma_t}, g^{z_{\sigma_t} p_{\sigma_t}}, q_{\sigma_t}, s_{\sigma_t}, \{g^{\alpha_j}, u^{\alpha_j}, s'_{\alpha_j}\}_{j=0}^{a-1})$ and y is the instance consisting of the commitments on w that includes $(D_t, D_{t-1}, h_1, h_2, h_3)$ together with $\mathsf{Condition}(w)$. For the exact form of $\mathsf{Condition}(w)$ see the detail construction of pok_t in equation (1) of Section 4. $\mathcal{E}$ verifies the proof pok_t by running $\mathsf{PoKVerify}$ and utilizing the instance y. $\mathcal{E}$ aborts if verification fails, otherwise, $\mathcal{E}$ executes $\mathsf{PoKExtract}(crs_{PoK}, td_{ext}, y, pok_t)$ to extract the witness w. Then for $i = 1, \ldots, n$, $\mathcal{E}$ compares u_{σ_t} and $g^{p_{\sigma_t}}$ to all u_i's and g^{p_i}'s respectively to determine $\mathcal{A}$'s choice σ_t and to check whether $\mathcal{A}$ has really paid the correct price corresponding to his choice. $\mathcal{E}$ also compares the signatures $\{g^{\alpha_j}, u^{\alpha_j}, s'_{\alpha_j}\}_{j=0}^{a-1}$ in the extracted witness that corresponds to the range proof with each of the signatures that were sent to $\mathcal{A}$ in $params_{Range}$ and the signature $\{u_{\sigma_t}, q_{\sigma_t}, s_{\sigma_t}\}$ to all the signatures $\{u_i, q_i, s_i\}_{i=1}^{n}$ that were sent to $\mathcal{A}$ in crs (this is done in order to ensure that $\mathcal{A}$ did not compute a forgery) and aborts if in either case no match is found, i.e., a forgery is detected.
5. If $\mathcal{E}$ finds that $\mathcal{A}$ has actually paid the price p_{σ_t}, i.e., extracted u_{σ_t}, $g^{p_{\sigma_t}}$ match with some u_i and the corresponding g^{p_i}, $\mathcal{E}$ sends $(sid, \mathcal{B}, \sigma_t)$ to $\mathcal{F}_{\mathsf{POT}}$ in order to obtain M_{σ_t}. Now $\mathcal{E}$ computes C_{σ_t} exactly as $\mathcal{V}$ does in $\mathsf{POTRespond}$ of the real protocol. For $i \neq \sigma_t$, $\mathcal{E}$ randomly picks $C_i^{(1)} \in \mathbb{G}_T$ and $C_i^{(2)} \in \mathbb{G}$ and sets $C_i = (C_i^{(1)}, C_i^{(2)})$. On the other hand, if $\mathcal{E}$ finds that $\mathcal{A}$ has not paid the correct price, it simply chooses $C_i^{(1)} \in \mathbb{G}_T$ and $C_i^{(2)} \in \mathbb{G}$ and sets $C_i = (C_i^{(1)}, C_i^{(2)}), i = 1, \ldots, n$. $\mathcal{E}$ sends $R = (C_1, \ldots, C_n)$ to $\mathcal{A}$ and stores D_t.

Claim (Vendor's security). When $\mathcal{B}$ is corrupted, the ensembles $\mathsf{IDEAL}_{\mathcal{F}_{\mathsf{POT}},\mathcal{E},\mathcal{Z}}$ and $\mathsf{REAL}_{\mathsf{POT},\mathcal{A},\mathcal{Z}}$ are computationally indistinguishable under the $\{max\{n,d\}\}$-**HSDH**, $\{max\{n,d\}\}$-**TDH** and the **Square DBDH** assumptions, where n is the the number of messages and $A = d^a$ is the upper bound of the buyer's account.

Proof. We show by means of a series of hybrid games that the environment $\mathcal{Z}$ cannot distinguish between the real execution ensembles $\mathsf{REAL}_{\mathsf{POT},\mathcal{A},\mathcal{Z}}$ and the simulated ensembles $\mathsf{IDEAL}_{\mathcal{F}_{\mathsf{POT}},\mathcal{E},\mathcal{Z}}$ with non-negligible probability. We again denote by $\mathsf{Pr}[\textbf{Game } \mathrm{i}]$ the probability that $\mathcal{Z}$ distinguishes between the ensembles of **Game** i and that of the real execution.

Game 0: This game corresponds to the execution of the real-world protocol with an honest $\mathcal{V}$. Therefore, $\mathsf{Pr}[\textbf{Game } 0]$=0.

Game 1: This game follows **Game** 0, except that to set crs we run $\mathsf{PoKExtractSetup}$ to obtain crs_{PoK} and an extraction trapdoor td_{ext} for a bilinear group setup $(p, \mathbb{G}, \mathbb{G}_T, e, g)$, where $p_{max} < A \pmod p$ holds. Note that crs_{PoK} so computed is identically distributed to the output of $\mathsf{PoKSetup}$ and thus, $|\mathsf{Pr}[\textbf{Game } 1] - \mathsf{Pr}[\textbf{Game } 0]| = 0$.

Game 2: The difference between this game and the previous one consists in that, in each time $t > 0$, we extract the witness of pok_t by running $\mathsf{PoKExtract}(crs_{PoK}, td_{ext}, y, pok_t)$, where y is the instance of pok_t as defined in the simulation. Groth-Sahai proofs being perfectly extractable, we have $|\mathsf{Pr}[\textbf{Game } 2] - \mathsf{Pr}[\textbf{Game } 1]| = 0$.

Game 3: This game is identical to **Game** 2, except that **Game** 3 aborts if at least one of the extracted signatures $\{g^{\alpha_j}, u^{\alpha_j}, s'_{\alpha_j}\}_{j=0}^{a-1}$ that are employed in the range proof does not equal any of the signatures $\{(g^i, u^i, s'_i)\}_{i\in\mathbb{Z}_d}$, where the signatures s'_i are included in $params_{Range}$. This means that $\mathcal{A}$ computed a forgery of the P-Signature scheme. The probability that $\mathcal{Z}$ distinguishes between **Game** 2 and **Game** 3 is bounded by the following lemma.

Lemma 1. *If the d-**HSDH** and the d-**TDH** assumptions hold, then* $|\mathsf{Pr}[\textbf{\textit{Game}}\ 3] - \mathsf{Pr}[\textbf{\textit{Game}}\ 2]| \leq \epsilon_1(\lambda)$*, where* $\epsilon_1(\lambda)$ *is a negligible function of* λ*.*

Proof. We build a forger $\mathcal{D}$ that breaks the F-unforgeability of the P-Signature scheme with non-negligible probability. Given such a forger, it is shown in [3] how to construct an algorithm $\mathcal{E}$ that breaks either the d-**HSDH** assumption or the d-**TDH** assumption with non-negligible probability.
Given a buyer that causes **Game** 3 to abort with non-negligible probability, $\mathcal{D}$ works as follows:

1. $\mathcal{D}$ obtains the parameters of the P-Signature scheme $crs_{Sig} = (p, \mathbb{G}, \mathbb{G}_T, e, g, u)$ from $\mathcal{E}$, runs $\mathsf{PoKExtractSetup}$ to get (crs_{PoK}, td_{ext}) and computes $u_i = g^{z_i}, u_{i,j} = g^{z_i z_j}, v_i = y_3^{z_i}, q_i = u^{z_i}, i, j = 1, \ldots, n, i \neq j$,where, $z_i \xleftarrow{\$} \mathbb{Z}_p^*$ are distinct and $crs_{PoK} = (t_1, t_2, t_3), t_3 = (y_1, y_2, y_3)$. $\mathcal{D}$ runs $\mathbf{PKeyGen}(crs_{Sig})$, where $crs_{Sig} = (crs_{PoK}, u)$ to compute (pk, sk). Also $\mathcal{D}$ computes s_i=$\mathbf{PSign}(crs_{Sig}, pk, z_i), i = 1, \ldots, n$. $\mathcal{D}$ sets $crs = (crs_{PoK}, \{u_i\}_i, \{u_{i,j}\}_{i \neq j}, \{v_i\}_i, u, pk, \{q_i\}_i, \{s_i\}_i)$.
2. $\mathcal{D}$ obtains the public key of the signature scheme pk' from $\mathcal{E}$.
3. For each $i \in \mathbb{Z}_d$, $\mathcal{D}$ queries $\mathcal{E}$ to obtain a signature s'_i=$\mathbf{PSign}(crs_{Sig}, pk', i)$ on the message i. $\mathcal{D}$ uses these signatures to set $params_{Range} = (pk', \{s'_i\}_{i \in \mathbb{Z}_d})$.
4. Upon receiving a request $Q = (h_1, h_2, h_3, pok_t, D_t)$ from $\mathcal{B}$, $\mathcal{D}$ extracts the witnesses that corresponds to the signatures $\{g^{\alpha_j}, u^{\alpha_j}, s'^{\alpha_j}\}_{j=0}^{a-1}$ that are employed in the range proof. If there exist a signature $\{g^{\alpha_j}, u^{\alpha_j}, x'_{\alpha_j}\}$ that does not equal any of the signatures $\{(g^i, u^i, s'_i)\}$ in $params_{Range}$, then $\mathcal{D}$ outputs this tuple $\{g^{\alpha_j}, u^{\alpha_j}, s'_{\alpha_j}\}$ as a forgery. □

Game 4: This game is identical to **Game** 3, except that this game aborts if the extracted signature $\{u_{\sigma_t}, q_{\sigma_t}, s_{\sigma_t}\}$ does not equal any of the signatures $\{(u_i, q_i, s_i)\}_{i=1}^n$ that are included in the crs. Again this means that $\mathcal{A}$ has computed a forged P-Signature . The probability that $\mathcal{Z}$ distinguishes between **Game** 4 and **Game** 3 is bounded by the following lemma.

Lemma 2. *If the n-**HSDH** and the n -**TDH** assumptions hold, then* $|\mathsf{Pr}[\textbf{\textit{Game}}\ 4] - \mathsf{Pr}[\textbf{\textit{Game}}\ 3]| \leq \epsilon_2(\lambda)$*, where* $\epsilon_2(\lambda)$ *is a negligible function of* λ*.*

Proof. The proof of this lemma is similar to that of Lemma 1 except the following. In the previous case, the forger $\mathcal{D}$ queried $\mathcal{E}$ for signatures to form $params_{Range}$ and computed the signatures included in the crs himself, whereas, in this case $\mathcal{D}$ does the other way round, i.e., computes $params_{Range}$ itself and queries $\mathcal{E}$ for the signatures involved in crs. □

Game 5: This game is same as the previous one except that the response R is computed as explained in Step 5. of the simulation. The probability that $\mathcal{Z}$ distinguishes between **Game** 5 and **Game** 4 is bounded by the following lemma.

Lemma 3. *If the **SqDBDH** assumption holds then* $|\mathsf{Pr}[\textbf{\textit{Game}}\ 5] - \mathsf{Pr}[\textbf{\textit{Game}}\ 4]| \leq \epsilon_3(\lambda)$*, where* $\epsilon_3(\lambda)$ *is a negligible function of* λ*.*

Proof. We consider two cases separately- (a) proper price is paid and (b) proper price is not paid.

Case (a) It is verified that the proper price p_{σ_t} corresponding to chosen index σ_t is paid. In this case R contain C_{σ_t} which is a valid encryption of M_{σ_t} and random C_i's for $i \neq \sigma_t$.
Without loss of generality, we may assume in **Game** 5, $\sigma_t = 1$ (can be reached by renumbering). We define a series of intermediate distributions $\mathcal{D}_j$ for $1 \leq j \leq n$. In distribution $\mathcal{D}_j$ the value C_i is a valid ciphertext of the message $M_i (1 \leq i \leq j)$, as generated in **Game** 4 and the remaining C_i's, $j < i \leq n$, are random. Thus, for $1 \leq i \leq j$, $C_i = (e(w_3/u_i^{p_i}, u_i^{S_i}) M_i, g^{S_i})$, where $w_3 = \mathcal{P}(\mathbf{G})$ for $\mathbf{G} = \{p_1\}$ extracted from $Q = (h_1, h_2, h_3, pok_t, D_t)$ by parsing h_3, and for $j < i \leq n$, $C_i = (C_i^{(1)}, C_i^{(2)})$, where $C_i^{(1)} \xleftarrow{\$} \mathbb{G}_T$ and $C_i^{(2)} \xleftarrow{\$} \mathbb{G}$. Note that, we define distributions such that $\mathcal{D}_1$ occurs in **Game** 5 and $\mathcal{D}_n$ occurs in **Game** 4. Now if an environment $\mathcal{Z}$ can distinguish between $\mathcal{D}_1$ and $\mathcal{D}_n$, then $\mathcal{Z}$ must also be able to distinguish the distributions $\mathcal{D}_\mu$ and $\mathcal{D}_{\mu-1}$ for index $1 < \mu \leq n$, i.e., $\exists$ a polynomial $p(\cdot)$ such that, $|\mathsf{Pr}[\mathcal{Z}(\mathcal{D}_\mu) = 1] - \mathsf{Pr}[\mathcal{Z}(\mathcal{D}_{\mu-1}) = 1]| > \frac{1}{p(\lambda)}$.
We use environment $\mathcal{Z}$ to construct an algorithm $\mathcal{T}$ that solves the **SqDBDH** problem. Given an instance $(g, g^a, g^b, e(g,g)^c)$, $\mathcal{T}$ works as follows:

– Run $\mathsf{PoKExtractSetup}$ to generate crs_{PoK} and td_{ext} as follows. Pick $(\alpha, \beta) \xleftarrow{\$} \mathbb{Z}_p^*; (r, s) \xleftarrow{\$} \mathbb{Z}_p$, compute $t_1 = (g^\alpha, 1, g), t_2 = (1, g^\beta, g), t_3 = (g^{r\alpha}, g^{s\beta}, g^{r+s})$, generate $crs_{PoK} = (t_1, t_2, t_3)$ and

set the extraction trapdoor $td_{ext} = (\alpha, \beta)$. Choose uniformly the index μ, $1 < \mu \leq n$, of the distribution and set $crs = (crs_{PoK}, \{u_i\}_i, \{u_{i,j}\}_{i \neq j}, \{v_i\}_i, u, pk, \{q_i\}_i, \{s_i\}_i)$ where

$$u_i = \begin{cases} g^a, & i = \mu \\ g^{z_i}, & i \neq \mu \end{cases}, \qquad u_{i,j} = \begin{cases} (g^a)^{z_j}, & j \neq \mu \\ g^{z_i z_j}, & j \neq i \neq \mu \end{cases}, \qquad s_i = \begin{cases} (g^r, g^{r^{-1}}, g^{-r^{-1}}), & i = \mu \\ (g, g^{a-z_i}, g^{z_i - a}), & i \neq \mu \end{cases},$$

$v_i = u_i^{r+s}$, $u = g^\tau$, $q_i = u_i^\tau$ and $pk = ((g^a)^{-1}, g^{-\tau})$ so that implicitly $sk = (-a, -\tau)$, for $z_1, \ldots, z_{\mu-1}, z_{\mu+1}, \ldots, z_n, \tau, r \xleftarrow{\$} \mathbb{Z}_p^*$ such that z_i's are all distinct.
(Note that all the s_i's are valid P-Signatures on z_i's,where implicitly $z_\mu = a$.

- Compute $params_{Range}$ as usual following POTInitVendor.
- Select some balance ac_0, $0 \leq \mathsf{ac}_0 < A$, and perform POTInitBuyer to obtain P $(= \mathsf{ac}_0)$.
- Choose messages $\{M_1, \ldots, M_n\}$ with prices $\{p_1, \ldots, p_n\}$ respectively from the same distribution as in the POT protocol of Section 4 and compute a request Q for $\sigma_t = 1$ followingPOTRequest as in **Game** 4.
- Extract the witnesses from Q and compute the response $R = (C_1, \ldots, C_n)$ as follows:

$$C_i = \begin{cases} (e(w_3/u_i^{p_i}, u_i^{S_i}) M_i, g^{S_i}), \ S_i \xleftarrow{\$} \mathbb{Z}_p, & \text{for } 1 \leq i < \mu \\ (e(u_{\mu,1}^{p_1} v_\mu^{r_3} u_\mu^{r_1 + r_2}, g^b)(e(g,g)^c)^{-p_\mu} M_\mu, g^b), & \text{for } i = \mu \\ (C_i^{(1)}, C_i^{(2)}), \ C_i^{(1)} \xleftarrow{\$} \mathbb{G}_T, \ C_i^{(2)} \xleftarrow{\$} \mathbb{G}, & \text{for } \mu < i \leq n \end{cases}$$

  where, $Q = (h_1, h_2, h_3, pok_t, D_t)$, $h_3 = (w_1, w_2, w_3)$ and (r_1, r_2, r_3) is the randomness used in the commitment h_3 as explained in POTRequest.
- Give all outputs together with the messages $\{M_1, \ldots, M_n\}$ and prices $\{p_1, \ldots, p_n\}$ to $\mathcal{Z}$ and outputs whatever is the output of $\mathcal{Z}$.

If $c = a^2 b$, the resulting distribution equals $\mathcal{D}_\mu$, otherwise, it equals $\mathcal{D}_{\mu-1}$. It follows that,

$$|\mathsf{Pr}[\mathcal{T}(g, g^a, g^b, e(g,g)^{a^2 b}) = 1] - \mathsf{Pr}[\mathcal{T}(g, g^a, g^b, e(g,g)^c) = 1]| > \frac{1}{np(\lambda)}$$

.
The factor n comes from the choice of μ. Since n is a polynomial in λ, the fraction $\frac{1}{np(\lambda)}$ is non-negligible in λ which contradicts the **SqDBDH** assumption.

Case (b) This case corresponds to the situation when it is found that the proper price for σ_t is not paid. In this case the response R consists of all random C_i's. Again in this case, we may assume, without loss of generality, that $\sigma_t = 1$, i.e., the message M_1 is queried but the price paid is not the correct price p_1. We define $(n+1)$ distributions as above with $\mathcal{D}_0$ denoting the distribution where R contains all random C_i's. Clearly in this case $\mathcal{D}_0$ corresponds to the distribution in **Game** 5 and $\mathcal{D}_n$ that in **Game** 4. As above we can argue that, if the environment $\mathcal{Z}$ can distinguish between $\mathcal{D}_0$ and $\mathcal{D}_n$ then $\mathcal{Z}$ must also be able to distinguish between $\mathcal{D}_\mu$ and $\mathcal{D}_{\mu-1}, 0 < \mu \leq n$. We use $\mathcal{Z}$ to construct an algorithm $\mathcal{T}$ which solves the **SqDBDH** problem as Case (a) with the following modifications.

- $\mathcal{T}$ chooses $1 \leq \mu \leq n$, uniformly.
- While computing the request Q for $\sigma_t = 1$, $\mathcal{T}$ uses a different price $p \neq p_1$.
- $\mathcal{T}$ computes the C_i's as follows:

$$C_i = \begin{cases} (e(w_3/u_i^{p_i}, u_i^{S_i}) M_i, g^{S_i}), \ S_i \xleftarrow{\$} \mathbb{Z}_p, & \text{for } 1 \leq i < \mu \\ (C_i^{(1)}, C_i^{(2)}), \ C_i^{(1)} \xleftarrow{\$} \mathbb{G}_T, \ C_i^{(2)} \xleftarrow{\$} \mathbb{G}, & \text{for } \mu < i \leq n \end{cases}$$

$$C_\mu = \begin{cases} (e(u_{\mu,1}^{p} v_\mu^{r_3} u_\mu^{r_1 + r_2}, g^b)(e(g,g)^c)^{-p_\mu} M_\mu, g^b), & \text{for } 1 < \mu \leq n \\ (e(v_\mu^{r_3} u_\mu^{r_1 + r_2}, g^b)(e(g,g)^c)^{p - p_\mu} M_\mu, g^b), & \text{for } \mu = 1 \end{cases}$$

With the above modified construction we can show as in Case (a) that $\mathcal{T}$ can solve the **SqDBDH** problem with non-negligible probability, which is a contradiction to the **SqDBDH** assumption. Thus if the **SqDBDH** assumption holds, then there exist a negligible function $\epsilon_3(\lambda)$ such that $|\mathsf{Pr}[\textbf{Game } 5] - \mathsf{Pr}[\textbf{Game } 4]| \leq \epsilon_3(\lambda)$. □

$\mathcal{E}$ performs all the changes described in **Game** 5, but in the initialization phase, $\mathcal{E}$ runs $\mathsf{POTGetDeposit}$ and sends ac_0 to $\mathcal{F}_{\mathsf{POT}}$, and in each transfer phase $\mathcal{E}$ stores $\{p_1, \ldots, p_n\}$ and publishes Pub after getting Pub from $\mathcal{F}_{\mathsf{POT}}$; sends the choice σ_t to $\mathcal{F}_{\mathsf{POT}}$ to obtain M_{σ_t} provided the price p_{σ_t} is paid, in which case $\mathcal{E}$ uses M_{σ_t} to compute C_{σ_t}; otherwise $\mathcal{E}$ sets C_{σ_t} to be a random value in $\mathbb{G}_T \times \mathbb{G}$. For all other i, $\mathcal{E}$ sets C_i's as random elements in $\mathbb{G}_T \times \mathbb{G}$. The distribution produced in **Game** 5 is identical to our simulation of the ideal ensemble $\mathsf{IDEAL}_{\mathcal{F}_{\mathsf{POT}},\mathcal{E},\mathcal{Z}}$. Therefore, by summation we have that, $|\mathsf{Pr}[\textbf{Game } 5]| \leq \epsilon_1(\lambda) + \epsilon_2(\lambda) + \epsilon_3(\lambda) = \epsilon(\lambda)$. □

□

C Extending the Priced Oblivious Transfer of Section 4 to Subscription Setting

The motivation of a subscription is to allow efficient one-way communication from the vendor to the buyer. Here we will briefly sketch how our proposed POT can be modified to fit in a 'subscription' setting. Note that for subscription a buyer $\mathcal{B}$ is charged the same price p_i for the i-th message effective at time of subscription even if prices may change over time. To initialize the protocol first both the vendor $\mathcal{V}$ and the buyer $\mathcal{B}$ takes crs from $\mathcal{F}_{\mathsf{CRS}}$ and $\mathcal{V}$ sends $(sid, params_{Range})$ to $\mathcal{B}$ by executing $\mathsf{POTInitVendor}$ as in the POT of Section 4.

Subscribing: The buyer $\mathcal{B}$ computes $(P, D_0^{(Priv)})$ by running $\mathsf{POTInitBuyer}$ on input $(crs, params_{Range}, \mathsf{ac}_0)$, chooses an index σ to subscribe and a time period τ for which he wants to subscribe. Now $\mathcal{B}$ picks $open_{p_\sigma}, open_{z_\sigma}, open_{z_\sigma p_\sigma}, \{open_{ac_t}\}_{t=1}^{\tau}$ randomly from $\mathbb{Z}_p^3$ and computes $h_1 = \mathsf{Commit}(g^{p_\sigma}, open_{p_\sigma})$, $h_2 = \mathsf{Commit}(u_\sigma, open_{z_\sigma}), h_3 = \mathsf{Commit}(g^{z_\sigma p_\sigma}, open_{z_\sigma p_\sigma}), D_t = \mathsf{Commit}(g^{ac_t}, open_{ac_t}), t = 1, \ldots, \tau$. Also for $t = 1, \ldots, \tau$, $\mathcal{B}$ runs $\mathsf{PoKProve}$ introduced in Section A.1 on input crs_{PoK} to obtain non-interactive proof of knowledge pok_t:

$$\mathsf{NIPK}\{(g^{\mathsf{ac}_t}, g^{\mathsf{ac}_{t-1}}, g^{p_\sigma}, u_\sigma, g^{z_\sigma p_\sigma}, q_\sigma, s_\sigma) : 0 \leq \mathsf{ac}_t < A \ \wedge \ e(g, g^{\mathsf{ac}_{t-1}})e(g^{-1}, g^{\mathsf{ac}_t})e(g^{-1}, g^{p_\sigma}) = 1 \ \wedge$$
$$e(g, g^{z_\sigma p_\sigma})e(u_\sigma, g^{p_\sigma})^{-1} = 1 \ \wedge \ \textbf{PVerifySig}(pk, s_\sigma, z_\sigma) = \mathsf{accept} \ \wedge$$
$$\mathsf{ac}_t \text{ in } D_t \wedge \mathsf{ac}_{t-1} \text{ in } D_{t-1} \wedge p_\sigma \text{ in } h_1 \wedge u_\sigma \text{ in } h_2 \wedge g^{z_\sigma p_\sigma} \text{ in } h_3\}.$$

$\mathcal{B}$ sets $Q = (h_1, h_2, h_3, \{(pok_t, D_t)\}_{t=1}^{\tau})$, $Q^{(Priv)} = (\sigma, open_{z_\sigma p_\sigma})$ and $\{D_t^{(Priv)} = (\mathsf{ac}_t, open_{\mathsf{ac}_t})\}_{t=0}^{\tau}$.
$\mathcal{B}$ sends (sid, P, Q, τ) to $\mathcal{V}$ and stores $(Q^{(Priv)}, \{D_t^{(priv)}\}_{t=0}^{\tau})$.

The vendor $\mathcal{V}$ runs $D_0 \leftarrow \mathsf{POTGetDeposit}(crs, P, A)$ as in our POT protocol of Section 4. $\mathcal{V}$ sets $D = (Q, \tau)$ after checking whether it really has received τ number of (pok_t, D_t)'s.

Maintaining a subscription: At time $t > 0$ following a subscription, $\mathcal{V}$ runs $\mathsf{POTRespond}$ on input D_{t-1}, Q_t where $Q_t = (h_1, h_2, h_3, pok_t, D_t)$, sends response (sid, R_t) to $\mathcal{B}$ and sets $D = (h_1, h_2, h_3, \{pok_j\}_{j=t+1}^{\tau}, \{D_j\}_{j=t}^{\tau}, \tau - t)$.

Upon receiving (sid, R_t), $\mathcal{B}$ executes $\mathsf{POTComplete}(crs, R_t, Q^{(Priv)})$ to obtain $M_\sigma^{(t)}$. Also $\mathcal{B}$ updates $D_t^{(Priv)} = \{(\mathsf{ac}_j, open_{\mathsf{ac}_j})\}_{j=t}^{\tau}$.

Unsubscribing: After time τ (for which $\mathcal{B}$ has subscribed), $\mathcal{V}$ finds $D = (h_1, h_2, h_3, \phi, \{D_\tau\}, 0)$ and hence, $\mathcal{V}$ automatically unsubscribes $\mathcal{B}$. However, if $\mathcal{B}$ wants to unsubscribe after l transfers, where $l < \tau$, then $\mathcal{V}$ sends $(sid, \{D_t\}_{t=l}^{\tau})$ to $\mathcal{B}$. Here D_l contains $\mathcal{B}$'s remaining balance after l-th transaction. $\mathcal{B}$ can open the commitment D_l to any trusted third party or the court of law to claim his remaining balance from $\mathcal{V}$. Note that, since Groth-Sahai commitments are perfectly binding, $\mathcal{B}$ cannot open D_l to any value other than ac_l.