# Optimistic Fair Priced Oblivious Transfer

Alfredo Rial and Bart Preneel

IBBT and Katholieke Universiteit Leuven, ESAT/COSIC
Kasteelpark Arenberg 10, B-3001 Leuven-Heverlee, Belgium
firstname.lastname@esat.kuleuven.be

**Abstract.** Priced oblivious transfer (POT) is a two-party protocol between a vendor and a buyer in which the buyer purchases digital goods without the vendor learning what is bought. Although privacy properties are guaranteed, current schemes do not offer fair exchange. A malicious vendor can, e.g., prevent the buyer from retrieving the goods after receiving the payment, and a malicious buyer can also accuse an honest vendor of misbehavior without the vendor being able to prove this untrue. In order to address these problems, we define the concept of optimistic fair priced oblivious transfer and propose a generic construction that extends secure POT schemes to realize this functionality. Our construction, based on verifiably encrypted signatures, employs a neutral adjudicator that is only involved in case of dispute, and shows that disputes can be resolved without the buyer losing her privacy, i.e., the buyer does not need to disclose which digital goods she is interested in. We show that our construction can be instantiated with an existing universally composable POT scheme, and furthermore we propose a novel full-simulation secure POT scheme that is much more efficient.

**Keywords:** Priced oblivious transfer, verifiably encrypted signatures, fair exchange.

## 1 Introduction

The protection of privacy in e-commerce is necessary both from a marketing perspective, since privacy concerns discourage buyers from online purchasing [1,2], as well as from a legal perspective, since different authorities have promulgated regulations to enforce the fulfillment of privacy policies and the confidentiality of buyer's personal data [3]. Additionally, buyers, feeling themselves in an unfavorable position at the payment phase, worry that malicious vendors can, e.g., deliver inappropriate or defective goods later on, and thus e-commerce protocols are normally analyzed in order to prove their fairness [4].

Priced oblivious transfer (POT) is a two-party protocol that provides privacy in e-commerce of digital goods by hiding from the vendor which items are bought. More formally, a vendor $\mathcal{V}$ sells a set of messages $m_1, \ldots, m_N$ with prices $p_1, \ldots, p_N$ to a buyer $\mathcal{B}$. At each purchase, $\mathcal{B}$ chooses $\tau \in \{1, \ldots, N\}$, gets $m_\tau$ and pays $p_\tau$ without $\mathcal{V}$ learning any information.

D.J. Bernstein and T. Lange (Eds.): AFRICACRYPT 2010, LNCS 6055, pp. 131–147, 2010.
© Springer-Verlag Berlin Heidelberg 2010

Existing POT schemes [5,6,7] employ a prepaid mechanism where, initially, $\mathcal{B}$ makes a deposit, and, at each purchase, subtracts $p_\tau$ from the deposit with $\mathcal{V}$ learning neither $p_\tau$ nor the new value of the deposit. Therefore, $\mathcal{B}$ has to trust that $\mathcal{V}$, after receiving the payment, will deliver the requested goods. Furthermore, a malicious $\mathcal{V}$ can claim that $\mathcal{B}$ ran out of funds, and it is impossible for $\mathcal{B}$ to prove this untrue. On the other hand, a malicious $\mathcal{B}$ can claim that $\mathcal{V}$ is not fulfilling his delivery obligations or that her current funds are larger, and again $\mathcal{V}$ cannot prove the contrary. We propose the concept of optimistic fair POT (OFPOT) as a countermeasure.

*Previous Work.* Fair e-commerce is often seen as a particular case of fair exchange. Early work on fair exchange [8,9] follows the approach of dividing the items to be exchanged into small pieces and exchanging them piece by piece. As noted in [10], the resulting protocol is unfair in the exchange of the last piece.

More recent work proposes the involvement of a neutral third party [11], which can be split up into several entities to avoid dependence on the reliability of a single entity [12]. When the third party is only involved in case of dispute, the protocol is called optimistic. Some of these protocols are based on verifiably encrypted signatures [13,14,15]. In a nutshell, $\mathcal{B}$ sends $\mathcal{V}$ a verifiably encrypted signature in her request, and, after $\mathcal{V}$ fulfills his delivery obligations, reveals a valid signature. If $\mathcal{B}$ does not reveal the signature, $\mathcal{V}$ sends the verifiably encrypted signature to the third party, which returns a valid signature after $\mathcal{V}$ demonstrates that he has fulfilled his obligations.

To the best of our knowledge, there were no previous attempts to design a fair e-commerce protocol based on POT. Existing privacy-preserving fair e-commerce protocols [16] provide buyers with anonymity, i.e., they hide from vendors the identity of buyers. As noted in [5,7], anonymous purchase has some disadvantages, like hindering customer management or making the use of currently deployed online payment methods impossible.

*Our Contribution.* We define an ideal functionality for OFPOT and we propose a construction that, taking a secure POT scheme as a building block, turns it into an OFPOT scheme. Our definition and construction involve a neutral third party, an adjudicator $\mathcal{A}$, which is only active in case of dispute.

Our construction is based on the use of verifiably encrypted signatures and, to some extent, resembles non-privacy preserving fair e-commerce protocols based on them. Nevertheless, we note that, since in these protocols privacy is not protected, it is trivial for $\mathcal{V}$ to show to $\mathcal{A}$ that he has fulfilled his obligations, and for $\mathcal{A}$ to verify this fact, because $\mathcal{B}$ discloses which item she requests. One of the main contributions of our work is to show that $\mathcal{A}$ can handle complaints from $\mathcal{V}$ and $\mathcal{B}$ by learning neither the list of items offered by $\mathcal{V}$ nor which ones are requested by $\mathcal{B}$.

To be proven full-simulation secure, our construction has to be instantiated with a full-simulation secure POT scheme. We show that the universally composable scheme in [7] is suitable for such an instantiation. However, the schemes in [5,6] are only half-simulation secure (vendor's security definition follows the

ideal-world/real-world paradigm, but buyer's security definition is an indistinguishability argument [17]), and thus they are not suitable. Since the scheme in [7] is rather inefficient, we propose a novel and efficient full-simulation secure POT scheme based on the oblivious transfer scheme in [18].

*Outline of the Paper.* In Section 2 we recall the ideal functionality for POT given in [7] and we describe our novel POT scheme. In Section 3 we define an ideal functionality for OFPOT and we depict our OFPOT scheme. Finally, Section 4 draws a conclusion and discusses future work.

## 2 Efficient Priced Oblivious Transfer

### 2.1 Technical Preliminaries

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

*Bilinear maps.* Let $\mathbb{G}$ and $\mathbb{G}_t$ be groups of prime order $p$. A map $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_t$ must satisfy bilinearity, i.e., $e(g^x, g^y) = e(g, g)^{xy}$; non-degeneracy, i.e., for all generators $g \in \mathbb{G}$, $e(g, g)$ generates $\mathbb{G}_t$; and efficiency, i.e., there exists an efficient algorithm $\mathsf{BMGen}(1^\kappa)$ 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 $a, b \in \mathbb{G}$.

**Security Assumptions.** Let $(p, \mathbb{G}, \mathbb{G}_t, e, g)$ be a pairing group setup. The Strong Diffie-Hellman assumption ($l$-SDH) [19] states that, on input $(g, g^x, \ldots, g^{x^l}) \in \mathbb{G}^{l+1}$ for $x \leftarrow \mathbb{Z}_p$, it is hard to output a pair $(c, g^{1/(x+c)})$ for $c \leftarrow \mathbb{Z}_p$. The Power Decisional Diffie-Hellman assumption ($l$-PDDH) [18] states that, on input $(g, g^x, \ldots, g^{x^l}, H) \in \mathbb{G}^{l+1} \times \mathbb{G}_t$ for random $H$ and $x \leftarrow \mathbb{Z}_p$, it is hard to distinguish between the vector $(H^x, H^{x^2}, \ldots, H^{x^l}) \in \mathbb{G}_t^l$ and a random vector $T \in \mathbb{G}_t^l$. As shown in [20], the $l$-PDDH assumption is implied by the $l$-BDHE assumption [21].

**Signature Schemes.** A signature scheme consists of the algorithms $(\mathsf{Kg}, \mathsf{Sign}, \mathsf{Vf})$. $\mathsf{Kg}(1^\kappa)$ outputs a key pair $(sk, pk)$. $\mathsf{Sign}(sk, m)$ outputs a signature $\sigma$ on message $m$. $\mathsf{Vf}(pk, \sigma, m)$ outputs $\mathsf{accept}$ if $\sigma$ is a valid signature on $m$ and $\mathsf{reject}$ otherwise. A signature scheme must be correct and unforgeable [22]. Informally speaking, correctness implies that $\mathsf{Vf}$ always accepts a valid signature. Unforgeability means that no p.p.t adversary should be able to output a message-signature pair $(\sigma, m)$ unless he has previously obtained a signature on $m$.

We employ the weakly secure signature scheme in [19], which utilizes the SDH assumption. This scheme is existentially unforgeable under a weak chosen message attack, where the adversary submits all signature queries before seeing the public key. The setup consists of the pairing group setup $(p, \mathbb{G}, \mathbb{G}_t, e, g)$.

$\mathsf{WKg}(1^\kappa)$ picks random $x \leftarrow \mathbb{Z}_p$ and outputs a key pair $(sk, pk) = (x, g^x)$. $\mathsf{WSign}(sk, m)$ computes $\sigma = g^{1/(x+m)}$. $\mathsf{WVf}(pk, \sigma, m)$ outputs $\mathsf{accept}$ if $e(\sigma, pk \cdot g^m) = e(g, g)$.

**Commitment schemes.** A non-interactive commitment scheme consists of the algorithms $(\mathsf{ComS}, \mathsf{Com}, \mathsf{Open})$. $\mathsf{ComS}(1^\kappa)$ generates the parameters of the commitment scheme $par_{com}$. $\mathsf{Com}(par_{com}, x, open)$ outputs a commitment $C$ to $x$ using auxiliary information $open$. A commitment is opened by revealing $(x, open)$ and checking whether $\mathsf{Open}(par_{com}, C, x, open)$ outputs $\mathsf{accept}$. 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 $par_{com}$.

We employ the commitment scheme proposed by Pedersen [23]. $\mathsf{PEComS}(1^\kappa)$ picks random generators $g, h$ of a group $\mathbb{G}$ of prime order $p$ and outputs $par_{com} = (g, h)$. $\mathsf{PECom}(par_{com}, x, open_x)$ outputs $C = g^x h^{open_x}$ on input $x \in \mathbb{Z}_p$ and randomness $open_x \in \mathbb{Z}_p$. $\mathsf{PEOpen}(par_{com}, C, x', open'_x)$ outputs $\mathsf{accept}$ if $C = g^{x'} h^{open'_x}$. This scheme is information theoretically hiding and computationally binding under the discrete logarithm assumption.

**Proofs of Knowledge.** A zero-knowledge proof of knowledge [24] is a two-party protocol between a prover and a verifier. The prover proves to the verifier knowledge of some secret input that fulfills some statement without disclosing this input to the verifier. The protocol should fulfill two properties. First, it should be a proof of knowledge, i.e., a prover without knowledge of the secret input convinces the verifier with negligible probability. More technically, there exists a knowledge extractor that extracts the secret input from a successful prover with all but negligible probability. Second, it should be zero-knowledge, i.e., the verifier does not learn any information about the secret input. More technically, for all possible verifiers there exists a simulator that, without knowledge of the secret input, yields a distribution that cannot be distinguished from the interaction with a real prover.

We use several existing results to prove statements about discrete logarithms: (1) proof of knowledge of a discrete logarithm modulo a prime [25]; (2) proof of knowledge of the equality of some element in different representations [26]; (3) proof of knowledge that a value $\alpha$ lies in a given interval $[0, A)$ [27]; and (4) proof of the disjunction or conjunction of any two of the previous [28]. These results are often given in the form of $\Sigma$-protocols but they can be turned into zero-knowledge protocols using efficient zero-knowledge compilers [29,30], and they can be turned into non-interactive proofs in the random oracle model via the Fiat-Shamir heuristic [31].

When referring to the proofs above, we follow the notation introduced by Camenisch and Stadler [32] for various proofs of knowledge of discrete logarithms and proofs of the validity of statements about discrete logarithms. $\mathsf{NIPK}\{(\alpha, \beta, \delta) : y = g^\alpha h^\beta \wedge \tilde{y} = \tilde{g}^\alpha \tilde{h}^\delta \wedge A \leq \alpha \leq B\}$ denotes a "*non-interactive*

*zero-knowledge proof of knowledge of integers $\alpha$, $\beta$, and $\delta$ such that $y = g^{\alpha}h^{\beta}$, $\tilde{y} = \tilde{g}^{\alpha}\tilde{h}^{\delta}$ and $A \leq \alpha \leq B$ holds*", where $y, g, h, \tilde{y}, \tilde{g}$, and $\tilde{h}$ are elements of some groups $G = \langle g \rangle = \langle h \rangle$ and $\tilde{G} = \langle \tilde{g} \rangle = \langle \tilde{h} \rangle$ that have the same order. (Note that some elements in the representation of $y$ and $\tilde{y}$ are equal.) The convention is that letters in the parenthesis, in this example $\alpha$, $\beta$, and $\delta$, denote quantities whose knowledge is being proven, while all other values are known to the verifier.

We employ the range proof proposed in [27] to prove that a value $\alpha$ lies in an interval $[0, d^a)$. The proof has an initialization phase where the verifier provides the prover with signatures that allow the prover to write the value $\alpha$ in base-$d$. First, the verifier runs $\mathsf{InitVer}(1^{\kappa}, D_{max})$, which runs $\mathsf{WKg}(1^{\kappa})$ to get a key pair $(sk, pk)$ and computes signatures $A_i = \mathsf{WSign}(sk, i)$ on $d$-ary digits, i.e., $i \in \mathbb{Z}_d$. It also runs $\mathsf{PEComS}(1^{\kappa})$ to get $par_{com}$ and sets $par_{ran} = (pk, \{A_i\}_{i \in \mathbb{Z}_d}, par_{com})$. The verifier sends $par_{ran}$ to the prover. The prover runs $\mathsf{InitP}(par_{ran})$ to verify the signatures by running, for $i \in \mathbb{Z}_d$, $\mathsf{WVf}(pk, A_i, i)$. After this initialization phase, prover and verifier can run multiple proofs. The verifier receives as input a commitment $C$, and the prover values $(\alpha, open_{\alpha})$ such that $C = \mathsf{PEComS}(par_{com}, \alpha, open_{\alpha})$. The prover runs $\mathsf{RProve}(par_{ran}, \alpha, open_{\alpha})$, which picks $v_j \leftarrow \mathbb{Z}_p$ and computes $V_j = A_{\alpha_j}^{v_j}$ for every $j \in \mathbb{Z}_a$ such that $\alpha = \sum_{j \in \mathbb{Z}_a} \alpha_j d^j$, and a proof $pok = \mathsf{NIPK}\{(open_{\alpha}, \{\alpha_j, v_j\}_{j \in \mathbb{Z}_a}) : C = h^{open_{\alpha}} \prod_{j \in \mathbb{Z}_a} (g^{d^j})^{\alpha_j} \wedge V_j = g^{v_j/(sk+\alpha_j)}\}$. (We abbreviate it as $\mathsf{NIPK}\{(\alpha, open_{\alpha}) : 0 \leq \alpha < d^a \wedge C = \mathsf{PEComS}(par_{com}, \alpha, open_{\alpha})\}$.) The prover sends $(\{V_j\}_{j \in \mathbb{Z}_a}, pok)$ to the verifier.

**Public Key Encryption.** A public key encryption scheme consists of the algorithms $(\mathsf{Kg}, \mathsf{Enc}, \mathsf{Dec})$. $\mathsf{Kg}(1^{\kappa})$ outputs a key pair $(sk, pk)$. $\mathsf{Enc}(pk, m)$ outputs a ciphertext $ct$ on input $pk$ and a message $m$. $\mathsf{Dec}(sk, ct)$ outputs the message $m$.

### 2.2 Definition

We define security following the ideal-world/real-world paradigm [33]. In the real world, a set of parties interact according to the protocol description in the presence of a real adversary $\mathcal{E}$, while in the ideal world dummy parties interact with an ideal functionality that carries out the desired task in the presence of an ideal adversary $\mathcal{S}$. A protocol $\psi$ is secure if there exists no environment $\mathcal{Z}$ that can distinguish whether it is interacting with adversary $\mathcal{E}$ and parties running protocol $\psi$ or with the ideal process for carrying out the desired task, where ideal adversary $\mathcal{S}$ and dummy parties interact with an ideal functionality $\mathcal{F}_{\psi}$. More formally, we say that protocol $\psi$ emulates the ideal process when, for any adversary $\mathcal{E}$, there exists a simulator $\mathcal{S}$ such that for all environments $\mathcal{Z}$, the ensembles $\mathrm{IDEAL}_{\mathcal{F}_{\psi},\mathcal{S},\mathcal{Z}}$ and $\mathrm{REAL}_{\psi,\mathcal{E},\mathcal{Z}}$ are computationally indistinguishable. We refer to [33] for a description of how these ensembles are constructed.

We recall the ideal functionality $\mathcal{F}_{\mathrm{POT}}$ for priced oblivious transfer in [7]. Every functionality and every protocol invocation should be instantiated with a unique session-ID that distinguishes it from other instantiations. For the sake of ease of notation, we omit session-IDs from our description.

**Functionality $\mathcal{F}_{\text{POT}}$**

Parameterized with the number of messages $N$, the message length $l$, the maximum price $p_{max}$, and the maximum deposit $D_{max}$, and running with a vendor $\mathcal{V}$ and a buyer $\mathcal{B}$, $\mathcal{F}_{\text{POT}}$ works as follows:

- On input a message $(\text{init}, m_1, p_1, \ldots, m_N, p_N)$ from $\mathcal{V}$, where each $m_i \in \{0,1\}^l$ and each $p_i \in [0, p_{max}]$, it stores $(m_1, p_1, \ldots, m_N, p_N)$ and sends $(\text{init}, p_1, \ldots, p_N)$ to $\mathcal{B}$.
- On input a message $(\text{deposit}, ac)$, where $ac \in [0, D_{max})$, if a $(\text{init}, \ldots)$ message was not received before, then it does nothing. Otherwise, it stores $ac$ and sends $(\text{deposit}, ac)$ to $\mathcal{V}$.
- On input a message $(\text{request}, \tau)$ from $\mathcal{B}$, where $\tau \in \{1, \ldots, N\}$, if either messages $(\text{init}, m_1, p_1, \ldots, m_N, p_N)$ and $(\text{deposit}, ac)$ were not received before or $ac - p_\tau < 0$, then it does nothing. Otherwise, it sends (request) to $\mathcal{V}$ and receives $(\text{response}, b)$ in response. If $b = 0$, it sends $(\text{response}, \perp)$ to $\mathcal{B}$. If $b = 1$, it updates $ac = ac - p_\tau$ and sends $(\text{response}, m_\tau)$ to $\mathcal{B}$.

### 2.3 Intuition Behind Our Construction

Our POT scheme is based on the adaptive oblivious transfer scheme in [18]. It is an assisted decryption scheme in which there is an initialization phase and several purchase phases. At the initialization phase, $\mathcal{V}$ sends $\mathcal{B}$ a collection of ciphertexts that encrypt the messages $m_1, \ldots, m_N$ to be sold, and at each purchase $\mathcal{V}$ helps $\mathcal{B}$ to decrypt one of them. As noted in [34], this design approach leads to purchase phases with constant communication and computation complexity, and ensures that $\mathcal{V}$ cannot modify the messages after the initialization phase.

Each ciphertext $C_i$ consists of a unique price[1] $p_i$, a signature on the price $A_i$ and an encryption $B_i = e(h, A_i) \cdot m_i$, where $h$ is the secret key of $\mathcal{V}$. To compute a request for $m_\tau$, $\mathcal{B}$ sends $\mathcal{V}$ a blinded value $V = A_\tau^v$ together with a zero-knowledge proof that $\mathcal{B}$ possesses a valid signature $A_\tau$ and that $V$ is correctly computed. $\mathcal{V}$ computes a blinded decryption $W = e(h, V)$ and proves that the secret key $h$ was used to compute $W$. Finally, $\mathcal{B}$ unblinds $W$ to decrypt the message $m_\tau$ via $B_\tau/(W^{1/v})$.

To allow for oblivious payments, our POT scheme follows the prepaid mechanism by [7]. At the initialization phase, $\mathcal{B}$ makes a deposit $ac_0$ and sends a commitment $D_0$ to the deposit and its opening to $\mathcal{V}$. At purchase phase $i$, $\mathcal{B}$ sends a commitment $D_i$ to the new value of the account $ac_i$ along with a zero-knowledge proof that $ac_i = ac_{i-1} - p_\tau$ and that $ac_i \in [0, D_{max})$, to prove that the account is correctly updated and that $\mathcal{B}$ has enough funds to buy $m_\tau$. For the latter, we employ the range proof recently proposed in [27].

[1] If several messages have the same price, unique prices can be obtained by adequately scaling prices and accounts [5].

We prove that our scheme realizes $\mathcal{F}_{\mathrm{POT}}$ under the SDH and PDDH assumptions (used in [18]), and under the assumption the binding and hiding properties of the commitment scheme hold. In the description of the scheme given below, we compute non-interactive proofs via Fiat-Shamir heuristic [31] and thus the construction is secure in the random oracle model [35]. This is convenient when using the scheme as a building block of the OFPOT scheme depicted in Section 3, because it allows $\mathcal{A}$ to verify requests and responses non-interactively. Nevertheless, for other uses of the scheme interactive proofs of knowledge can be employed, yielding a construction in the standard model.

### 2.4 Description of the Scheme

We begin with a high level description of the POT scheme. Details on the algorithms can be found below.

**POT**

**Initialization phase.** On input $(\text{init}, m_1, p_1, \ldots, m_N, p_N)$, in which each price is unique, $\mathcal{V}$ runs $\mathsf{POTInitV}(1^\kappa, m_1, p_1, \ldots, m_N, p_N, D_{max})$ in order to obtain a database commitment $T$ and a key pair $(pk_{\mathcal{V}}, sk_{\mathcal{V}})$, and sends $(pk_{\mathcal{V}}, T)$ to $\mathcal{B}$. On input $(\text{deposit}, ac_0)$, $\mathcal{B}$ runs $(P, D_0^{'}) \leftarrow \mathsf{POTInitB}(1^\kappa, pk_{\mathcal{V}}, T, ac_0)$ and aborts if the output is reject. Otherwise, $\mathcal{B}$ sends the payment message $(P)$ to $\mathcal{V}$ and pays an amount of $ac_0$ through an arbitrary payment channel. $\mathcal{V}$ runs $(D_0, ac_0) \leftarrow \mathsf{POTGetDep}(sk_{\mathcal{V}}, P, D_{max})$ and checks that $ac_0$ corresponds to the amount of money received. $\mathcal{V}$ stores state information $V_0 = (T, sk_{\mathcal{V}}, pk_{\mathcal{V}}, D_0)$ and outputs $(\text{deposit}, ac_0)$, and $\mathcal{B}$ stores state information $B_0 = (pk_{\mathcal{V}}, T, D_0^{'})$.

**Transfer phase.** On input $(\text{request}, \tau_i)$, $\mathcal{B}$ runs $\mathsf{POTReq}(pk_{\mathcal{V}}, T, D_{i-1}^{'}, \tau_i)$ to get a request $Q$ and private state $(Q^{'}, D_i^{'})$. $\mathcal{B}$ sends $(Q)$ and stores $(Q^{'}, D_i^{'})$. On input $(\text{response}, b)$, if $b = 0$ $\mathcal{V}$ sends $(\perp)$ to $\mathcal{B}$. If $b = 1$, $\mathcal{V}$ runs $\mathsf{POTVerReq}(pk_{\mathcal{V}}, D_{i-1}, Q)$ and ignores the request if the output is reject. Otherwise $\mathcal{V}$ runs $\mathsf{POTResp}(pk_{\mathcal{V}}, sk_{\mathcal{V}}, Q)$ to obtain a response $R$ and state $D_i$, and sends $(R)$ to $\mathcal{B}$. $\mathcal{B}$ runs $\mathsf{POTVerResp}(pk_{\mathcal{V}}, R)$ and outputs $(\text{response}, \perp)$ it the output is reject. Otherwise $\mathcal{B}$ runs $\mathsf{POTComplete}(T, R, Q^{'})$ to obtain $m_{\tau_i}$. $\mathcal{V}$ stores state information $V_i = (sk_{\mathcal{V}}, T, pk_{\mathcal{V}}, D_i)$, and $\mathcal{B}$ stores state information $B_i = (T, pk_{\mathcal{V}}, D_i^{'})$ and outputs $(\text{response}, m_{\tau_i})$.

$\mathsf{POTInitV}(1^\kappa, m_1, p_1, \ldots, m_N, p_N, D_{max})$ computes a pairing group setup $\Phi = (p, \mathbb{G}, \mathbb{G}_t, e, g)$, picks a random generator $h \in \mathbb{G}$ and sets $H = e(g, h)$. It runs $\mathsf{InitVer}(1^\kappa, D_{max})$ to obtain $par_{ran}$ (which includes parameters of the

commitment scheme $par_{com}$) and $\mathsf{Kg}(1^\kappa)$ to obtain $(sk_{enc}, pk_{enc})$.[2] It also runs $\mathsf{WKg}(1^\kappa)$ to obtain a key pair $(sk, pk)$ for the signature scheme, and, for $i = 1$ to $N$, it computes $A_i = \mathsf{WSign}(sk, p_i)$ and $B_i = e(h, A_i) \cdot m_i$. It sets $C_i = (A_i, B_i, p_i)$. It computes a non-interactive proof $\pi_1 = \mathsf{NIPK}\{(h) : H = e(g, h)\}$. It outputs $sk_{\mathcal{V}} = (sk_{enc}, h)$, $pk_{\mathcal{V}} = (\Phi, H, par_{ran}, pk, pk_{enc}, \pi_1)$ and $T = (C_1, \ldots, C_N)$.

$\mathsf{POTInitB}(1^\kappa, pk_{\mathcal{V}}, T, ac_0)$ parses, for $i = 1$ to $N$, $C_i$ as $(A_i, B_i, p_i)$, and checks whether $\mathsf{WVf}(pk, A_i, p_i)$ outputs $\mathsf{accept}$. It also verifies $\pi_1$ and checks $par_{ran}$ via $\mathsf{InitP}(par_{ran})$. Then it runs $\mathsf{Enc}(pk_{enc}, ac_0)$ to obtain an encryption $ct$ and computes a commitment $D_0 = \mathsf{PECom}(par_{com}, ac_0, open_{ac_0})$ for random $open_{ac_0}$.[3] It sets $P = (ct, open_{ac_0})$ and $D_0^{'} = (ac_0, open_{ac_0}, D_0)$. It outputs $(P, D_0^{'})$.

$\mathsf{POTGetDep}(sk_{\mathcal{V}}, P, D_{max})$ runs $\mathsf{Dec}(sk_{enc}, ct)$ to obtain $ac_0$ and checks that $ac_0 \in [0, D_{max})$. It computes $D_0 = \mathsf{PECom}(par_{com}, ac_0, open_{ac_0})$ and outputs $(D_0, ac_0)$.

$\mathsf{POTReq}(pk_{\mathcal{V}}, T, D_{i-1}^{'}, \tau)$ calculates $ac_i = ac_{i-1} - p_\tau$, picks random $(open_p, open_{ac_i}) \leftarrow \mathbb{Z}_p$, and computes $D_p = \mathsf{PECom}(par_{com}, p_\tau, open_p)$ and $D_i = \mathsf{PECom}(par_{com}, ac_i, open_{ac_i})$. It picks random $v \in \mathbb{Z}_p$ and computes $V = A_\tau^v$. It computes a non-interactive proof[4] $\pi_2$:

$$
\begin{aligned}
&\mathsf{NIPK}\{(p_\tau, open_p, ac_i, open_{ac_i}, v, \alpha) : \\
&\quad e(V, pk) = e(V, g)^{-p_\tau} e(g, g)^v \wedge && (1)\\
&\quad D_p = \mathsf{PECom}(par_{com}, p_\tau, open_p) \wedge && (2)\\
&\quad D_i = \mathsf{PECom}(par_{com}, ac_i, open_{ac_i}) \wedge && (3)\\
&\quad D_{i-1}/(D_i D_p) = \tilde{h}^\alpha \wedge && (4)\\
&\quad 0 \leq ac_i < D_{max}\}. && (5)
\end{aligned}
$$

Equation 1 proves that $V = A_\tau^v$ and that $A_\tau$ is a signature computed by $\mathcal{V}$. Equations 2 and 3 prove knowledge of the committed price $p_\tau$ and $ac_i$ respectively, where $p_\tau$ is the value signed in $A_\tau$. Equation 4 proves that $ac_i = ac_{i-1} - p_\tau$, and Equation 5 proves that $ac_i$ is non-negative. The algorithm outputs $Q = (V, D_p, D_i, \pi_2)$, $Q^{'} = (v, \tau)$ and $D_i^{'} = (ac_i, open_{ac_i}, D_i)$.

$\mathsf{POTVerReq}(pk_{\mathcal{V}}, D_{i-1}, Q)$ verifies $\pi_2$ and outputs either $\mathsf{accept}$ or $\mathsf{reject}$.

$\mathsf{POTResp}(pk_{\mathcal{V}}, sk_{\mathcal{V}}, Q)$ parses $Q$ as $(V, D_p, D_i, \pi_2)$ and computes $W = e(h, V)$ and a proof $\pi_3 = \mathsf{NIPK}\{(h) : H = e(g, h) \wedge W = e(V, h)\}$ that $W$ is computed correctly by using the request $V$ and the secret key $h$. It outputs $R = (W, \pi_3)$ and $D_i$.

$\mathsf{POTVerResp}(pk_{\mathcal{V}}, R)$ verifies $\pi_3$ and outputs either $\mathsf{accept}$ or $\mathsf{reject}$.

$\mathsf{POTComplete}(T, R, Q^{'})$ parses $Q^{'}$ as $(v, \tau)$, $R$ as $(W, \pi_3)$ and $C_\tau$ as $(A_\tau, B_\tau, p_\tau)$. It outputs $m_\tau = B_\tau/(W^{1/v})$.

[2] We can employ any IND-CPA secure encryption scheme.

[3] Although it would be possible to use a fixed value $open_{ac_0}$, this allows $\mathcal{B}$ to hide her deposit from the adjudicator in the OFPOT scheme.

[4] $\tilde{h}$ is included in the parameters of the commitment scheme $par_{com} = (\tilde{g}, \tilde{h})$.

**Theorem 1.** *This POT scheme securely realizes* $\mathcal{F}_{\mathrm{POT}}$.

We prove this theorem in the full version [36]. The proof follows the proof of the oblivious transfer scheme in [18] and of the priced oblivious transfer scheme in [7]. Buyer security holds under the assumption that the commitment scheme is hiding and under the extractability and zero-knowledge properties of proofs of knowledge. Vendor security holds under the $(N+1, d+1)$-SDH assumption, under the $(N+1)$-PDDH assumption and under the binding property of the commitment scheme. Security is proven under static corruptions.

As for efficiency, we note that the communication and computation cost of our protocol is roughly that of the OT scheme in [18], plus the overhead introduced by the range proof in [27]. We refer to [34] for a comparison of the efficiency of several OT schemes, and to [27] for efficiency measurements of the range proof.

## 3 An Optimistic Fair POT Scheme

### 3.1 Technical Preliminaries

**Verifiably Encrypted Signatures.** A verifiably encrypted signature (VES) scheme consists of algorithms $(\mathsf{Kg}, \mathsf{AdjKg}, \mathsf{Sign}, \mathsf{Vf}, \mathsf{Create}, \mathsf{VesVf}, \mathsf{Adj})$. Algorithm $\mathsf{Kg}(1^\kappa)$ outputs a key pair $(sk, pk)$. $\mathsf{AdjKg}(1^\kappa)$ outputs a key pair $(ask, apk)$ for the adjudicator. $\mathsf{Sign}(sk, m)$ computes a signature $\sigma$ under $sk$ on a message $m$. $\mathsf{Vf}(pk, \sigma, m)$ outputs $\mathsf{accept}$ if $\sigma$ is a valid signature on $m$ under $pk$. $\mathsf{Create}(sk, apk, m)$ outputs a verifiably encrypted signature $\omega$ on input the secret key $sk$, the adjudicator's public key $apk$ and the message $m$. $\mathsf{VesVf}(pk, apk, \omega, m)$ outputs $\mathsf{accept}$ if $\omega$ is a valid verifiably encrypted signature on input the public key $pk$, the adjudicator's public key $apk$ and the message $m$. $\mathsf{Adj}(pk, ask, apk, \omega, m)$ outputs an ordinary signature $\sigma$ on $m$ if $\omega$ is valid.

A VES scheme is complete if, for all key pairs $(sk, pk)$ computed by $\mathsf{Kg}(1^\kappa)$ and for all key pairs $(ask, apk)$ computed by $\mathsf{AdjKg}(1^\kappa)$, it holds that $\mathsf{VesVf}(pk, apk, \mathsf{Create}(sk, apk, m), m)$ outputs $\mathsf{accept}$ and algorithm $\mathsf{Vf}(pk, \mathsf{Adj}(pk, ask, apk, \mathsf{Create}(sk, apk, m), m), m)$ outputs $\mathsf{accept}$.

Security for verifiably encrypted signatures [13,15] is defined via four properties: unforgeability, opacity, extractability and abuse-freeness. Roughly speaking, unforgeability ensures that it is intractable to compute a verifiably encrypted signature on behalf of another user. Opacity ensures that nobody but the adjudicator and the signer can obtain an ordinary signature from a verifiably encrypted signature. Extractability means that the adjudicator is able to extract an ordinary signature from a valid verifiably encrypted signature with all but negligible probability. Abuse-freeness prevents an adversary that colludes with the adjudicator from forging verifiably encrypted signatures.

### 3.2 Definition

Previous work on defining optimistic fair exchange in the ideal-world/real-world paradigm only covers the case where buyer's privacy is not protected [37]. We

define an ideal functionality $\mathcal{F}_{\text{OFPOT}}$ for OFPOT that extends $\mathcal{F}_{\text{POT}}$. It describes the functionality provided by our construction when the adjudicator $\mathcal{A}$ is neutral, and we prove security under this restriction. (We explain more intuitively the functionality provided by $\mathcal{F}_{\text{OFPOT}}$ in Section 3.3.) We also recall the description of a functionality $\mathcal{F}_{\text{CRS}}$ given in [38], which is used when the POT scheme used to instantiate the OFPOT scheme operates in the $\mathcal{F}_{\text{CRS}}$-hybrid model, where parties have access to an honestly-generated common reference string.

**Functionality $\mathcal{F}_{\text{CRS}}$**

Parameterized with a distribution $D$ and a set of participants $\mathcal{P}$, on input (crs) from party $P$, if $P \notin \mathcal{P}$ it aborts. Otherwise, if there is no value $r$ recorded, it picks $r \leftarrow D$ and records $r$. It sends $(\text{crs}, r)$ to $P$.

**Functionality $\mathcal{F}_{\text{OFPOT}}$**

Parameterized with $(N, l, p_{max}, D_{max})$, and running with a vendor $\mathcal{V}$, a buyer $\mathcal{B}$ and an adjudicator $\mathcal{A}$, $\mathcal{F}_{\text{OFPOT}}$ works as follows:

- On input a message $(\text{init}, m_1, p_1, \ldots, m_N, p_N)$ from $\mathcal{V}$, it behaves as $\mathcal{F}_{\text{POT}}$.
- On input a message $(\text{deposit}, ac_0)$ from $\mathcal{B}$, it behaves as $\mathcal{F}_{\text{POT}}$.
- On input a message $(\text{request}, i, \tau)$ from $\mathcal{B}$, where $\tau \in \{1, \ldots, N\}$, if messages $(\text{init}, \ldots)$ or $(\text{deposit}, \ldots)$ were not received before, $i$ does not equal the number of the transfer $t$, $ac_i - p_\tau < 0$, or $dispute_i = 1$, then it does nothing. Otherwise, it sends $(\text{request}, i)$ to $\mathcal{V}$ and receives $(\text{response}, i, b)$ in response. It stores $(i, \tau, b)$. If $b = 0$, it sends $(\text{response}, i, \perp)$ to $\mathcal{B}$. If $b = 1$, it sends $(\text{response}, i, m_\tau)$ to $\mathcal{B}$. $\mathcal{B}$ returns a message $(\text{complete}, i, c)$, which is handed to $\mathcal{V}$. If $c = 1$, it sets $ac_i = ac_{i-1} - p_\tau$ and updates the number of transfers $t = t + 1$. If $c = 0$, it sets $dispute_i = 1$.
- On input a message $(\text{complain}_{\mathcal{V}}, i)$ from $\mathcal{V}$, if $dispute_i = 0$ or $t \neq i$, then it sends $(\text{compInv}_{\mathcal{V}})$ to $\mathcal{V}$ and $(\text{complain}_{\mathcal{V}}, i, \text{inv})$ to $\mathcal{A}$. Otherwise it sets $ac_i = ac_{i-1} - p_\tau$, updates the number of transfers $t = t + 1$ and sends $(\text{compSolved}_{\mathcal{V}})$ to $\mathcal{V}$, $(\text{compResp}_{\mathcal{V}}, i, m_\tau)$ to $\mathcal{B}$ and $(\text{complain}_{\mathcal{V}}, i, \text{solved})$ to $\mathcal{A}$.
- On input a message $(\text{complain}_{\mathcal{B}}, i, \tau)$ from $\mathcal{B}$, if $t \neq i$ or if there exists a tuple $(i, \tau, b)$ and $\tau \neq \tau'$, then it sends $(\text{compInv}_{\mathcal{B}})$ to $\mathcal{B}$ and $(\text{complain}_{\mathcal{B}}, i, \text{inv})$ to $\mathcal{A}$. Otherwise it sends $(\text{complain}_{\mathcal{B}}, i)$ to $\mathcal{V}$. Upon receiving $(\text{collaborate}, i, b)$ from $\mathcal{V}$, if $b = 0$ it sends $(\text{guilty}_{\mathcal{V}})$ to $\mathcal{V}$ and $\mathcal{B}$ and $(\text{complain}_{\mathcal{B}}, i, \text{guilty}_{\mathcal{V}})$ to $\mathcal{A}$. If $b = 1$ it sets $ac_i = ac_{i-1} - p_\tau$, updates the number of transfers $t = t + 1$, sends $(\text{compResp}_{\mathcal{B}}, i, m_\tau)$ to $\mathcal{B}$, $(\text{compSolved}_{\mathcal{B}})$ to $\mathcal{V}$ and $(\text{complain}_{\mathcal{B}}, i, \text{solved})$ to $\mathcal{A}$.

### 3.3 Intuition Behind Our Construction

Our OFPOT scheme extends a full-simulation secure POT scheme with a fair exchange protocol based on verifiably encrypted signatures. In a nutshell, at a purchase phase, $\mathcal{B}$ computes her request following the POT scheme and a verifiably encrypted signature that signs the request. Then $\mathcal{V}$ computes a response following the POT scheme. If $\mathcal{B}$ is satisfied, $\mathcal{B}$ reveals a valid signature on her request, and otherwise $\mathcal{B}$ complains. $\mathcal{V}$ also complains when he computed his response honestly and $\mathcal{B}$ did not reveal a valid signature.

More concretely, the initialization phase follows that of the POT scheme. Additionally, $\mathcal{B}$ signs the payment message she sends to $\mathcal{V}$, $\mathcal{A}$ hands its public key to $\mathcal{V}$ and $\mathcal{B}$, and $\mathcal{A}$ receives the public keys of $\mathcal{V}$ and $\mathcal{B}$. At purchase phase $i$, $\mathcal{B}$ computes a request $Q_i$ following algorithm $\mathsf{POTReq}$ of the POT scheme, and computes a verifiably encrypted signature $\omega_i$ on $H(i, Q_i)$, where $H$ is a collision-resistant hash function. $\mathcal{V}$ rejects the request if $i$ is not the valid purchase index, i.e., if $\mathcal{V}$ did not receive a signature $\sigma_i$ on $H(i-1, Q_{i-1})$ or if $\mathcal{V}$ already received a signature on $H(i, Q'_i)$. Otherwise $\mathcal{V}$ computes a response $R$ following algorithm $\mathsf{POTResp}$ of the POT scheme. Finally, $\mathcal{B}$ obtains the message $m_\tau$ by running algorithm $\mathsf{POTComplete}$ and reveals to $\mathcal{V}$ a signature $\sigma_i$ on $H(i, Q_i)$.[5]

When $\mathcal{B}$ does not reveal $\sigma_i$, $\mathcal{V}$ complains by sending to $\mathcal{A}$ the request $(i, Q_i, \omega_i)$ and the response $R$. $\mathcal{A}$ verifies both request and response, sends $R$ to $\mathcal{B}$, extracts $\sigma_i$ from $\omega_i$ and sends $\sigma_i$ to $\mathcal{V}$. Verification by $\mathcal{A}$ can be done without knowing the choice $\tau$ of $\mathcal{B}$. We note that a malicious $\mathcal{V}$ cannot produce fake requests on behalf of $\mathcal{B}$ thanks to the unforgeability property of the VES scheme. Moreover, if $\mathcal{B}$ did not reveal $\sigma_i$ because the response $R'$ from $\mathcal{V}$ was not correct, after $\mathcal{V}$ complains $\mathcal{A}$ ensures that $\mathcal{B}$ receives a valid response.

When the response sent by $\mathcal{V}$ is not correct, $\mathcal{B}$ complains by sending to $\mathcal{A}$ a request $(i, Q_i, \omega_i)$.[6] $\mathcal{A}$ tells $\mathcal{V}$ to send $(j, Q_j, \sigma_j)$ for $j = i - 1$ in order to check if the complaint is valid. (When $i = 1$, $\mathcal{V}$ should send the signature on the payment message.) If $\mathcal{V}$ sends it for $j \geq i$, then $\mathcal{A}$ states that the complaint is invalid because this means that $\mathcal{B}$ has already shown conformity for purchase $i$, and if $j < i - 1$, then it asks $\mathcal{B}$ to send a request with index $j$.[7] If $\mathcal{V}$ does not reveal any signature, $\mathcal{A}$ finds $\mathcal{V}$ guilty. (We note that $\mathcal{V}$ cannot produce signatures on behalf of $\mathcal{B}$ thanks to the opacity property of the VES scheme.) Otherwise, $\mathcal{A}$ verifies the request from $\mathcal{B}$ and sends it to $\mathcal{V}$. If $\mathcal{V}$ returns a different tuple $(i, Q'_i, \omega'_i)$, then $\mathcal{A}$ states that the complaint is invalid because this may mean that a malicious $\mathcal{B}$ already obtained the message requested in $Q'_i$, and is trying to obtain a different message. Otherwise, if $\mathcal{V}$ returns a valid response $R$, $\mathcal{A}$ forwards $R$ to $\mathcal{B}$ and reveals $\sigma_i$ to $\mathcal{V}$. If not, $\mathcal{A}$ finds $\mathcal{V}$ guilty.

[5] In practical implementations, this message can be sent together with the next request in order to save communication rounds.

[6] We note that $\mathcal{B}$ can complain without previously interacting with $\mathcal{V}$, but this does not give a malicious $\mathcal{B}$ any advantage over $\mathcal{V}$.

[7] We note that $\mathcal{V}$ should always send the larger index $j$ for which he possesses a signature, because otherwise he would be claiming that some purchases did not happen, thus increasing the account of $\mathcal{B}$.

We prove that our scheme realizes $\mathcal{F}_{\text{OFPOT}}$ if it is instantiated with a full-simulation secure POT scheme (which realizes $\mathcal{F}_{\text{POT}}$) and under the assumptions that the VES scheme is unforgeable, opaque and extractable, and that $\mathcal{A}$ is neutral. The abuse-freeness property of the VES scheme gives $\mathcal{B}$ some protection when $\mathcal{V}$ and $\mathcal{A}$ collude: they cannot produce fake requests. However, once $\mathcal{B}$ sends a request, a malicious $\mathcal{A}$ can reveal $\sigma_i$ to $\mathcal{V}$ without sending $R$ to $\mathcal{B}$. (Nevertheless, privacy of $\mathcal{B}$ is still preserved.) Similarly, when $\mathcal{B}$ and $\mathcal{A}$ collude, $\mathcal{V}$ is not protected.

When our construction is instantiated with the universally composable POT scheme in [7] (which we summarize in the full version [36]), we obtain a UC secure construction in the common reference string model. (In the description given below, we include the common reference string as input of the algorithms, but it should be removed when unnecessary.) When instantiated with the POT scheme proposed in Section 2, we obtain a full-simulation secure and more efficient scheme.

Finally, we point out that our scheme does not ensure that all the buyers receive the same messages, or that the messages that $\mathcal{V}$ sells fulfill the expectations of $\mathcal{B}$. Nevertheless, there already exist countermeasures which employ a third party against these problems [39], and $\mathcal{A}$ can apply them. Basically, these countermeasures ensure that all the buyers obtain the same messages from $\mathcal{V}$. We can achieve this by ensuring that they obtain the same database commitment $T$.

### 3.4 Description of the Scheme

We begin with a high level description of the OFPOT scheme. Details on the algorithms can be found below. We recall that the scheme is parameterized with $(N, l, p_{max}, D_{max})$.

**OFPOT**

**Initialization phase.** On input $(\text{init}, m_1, p_1, \ldots, m_N, p_N)$, $\mathcal{V}$ queries $\mathcal{F}_{\text{CRS}}$ with (crs), which runs $\mathsf{POTGenCRS}(1^\kappa, p_{max}, D_{max})$ and returns (crs, $crs$). $\mathcal{B}$ and $\mathcal{A}$ also query $\mathcal{F}_{\text{CRS}}$ with (crs), which returns (crs, $crs$). $\mathcal{V}$ runs $\mathsf{POTInitV}(crs, m_1, p_1, \ldots, m_N, p_N, D_{max})$ to get a database commitment $T$ and a key pair $(sk_{\mathcal{V}}, pk_{\mathcal{V}})$, and sends $(pk_{\mathcal{V}}, T)$ to $\mathcal{B}$. On input (deposit, $ac_0$), $\mathcal{B}$ computes $(P, D_0^{'}) \leftarrow \mathsf{POTInitB}(crs, pk_{\mathcal{V}}, T, ac_0)$ to obtain a payment message $P$ and the opening of the commitment to the account $D_0^{'}$, and aborts if the output is reject. Otherwise, $\mathcal{B}$ runs $(sk_{\mathcal{B}}, pk_{\mathcal{B}}, \sigma_0) \leftarrow \mathsf{OFInitB}(crs, D_0^{'})$ to obtain a key pair $(sk_{\mathcal{B}}, pk_{\mathcal{B}})$ and a signature $\sigma_0$ on $D_0$, sends $(P, pk_{\mathcal{B}}, \sigma_0)$ and pays $ac_0$ to $\mathcal{V}$ through an arbitrary payment channel. $\mathcal{V}$ runs $(D_0, ac_0) \leftarrow \mathsf{POTGetDep}(crs, sk_{\mathcal{V}}, P, D_{max})$ to obtain the commitment $D_0$ and checks that $ac_0$ equals the amount of money paid. $\mathcal{V}$ runs $\mathsf{OFInitV}(pk_{\mathcal{B}}, \sigma_0, D_0)$

to check $\sigma_0$ and aborts if the output is reject. $\mathcal{A}$ runs $(ask, apk) \leftarrow \mathsf{OFInitA}(crs)$ and sends $apk$ to $\mathcal{V}$ and $\mathcal{B}$. $\mathcal{V}$ and $\mathcal{B}$ retrieve $apk$. $\mathcal{V}$ ($\mathcal{B}$) sends $(pk_{\mathcal{V}}, pk_{\mathcal{B}})$ $((pk'_{\mathcal{V}}, pk'_{\mathcal{B}}))$ to $\mathcal{A}$, $\mathcal{A}$ checks equality of the public keys and stores $(pk_{\mathcal{V}}, pk_{\mathcal{B}}, ask)$. (Alternatively, $\mathcal{A}$, $\mathcal{V}$ and $\mathcal{B}$ can use a PKI.) $\mathcal{V}$ stores state information $V_0 = (sk_{\mathcal{V}}, T, pk_{\mathcal{V}}, pk_{\mathcal{B}}, apk, D_0, \sigma_0)$ and outputs $(\text{deposit}, ac_0)$, and $\mathcal{B}$ stores state information $B_0 = (sk_{\mathcal{B}}, T, pk_{\mathcal{B}}, pk_{\mathcal{V}}, apk, D_0^{'})$ and outputs $(\text{init}, p_1, \ldots, p_N)$.

**Transfer phase.** On input $(\text{request}, i, \tau)$, $\mathcal{B}$ does nothing if $\mathcal{B}$ has previously received $(\text{request}, i, \ldots)$. Otherwise $\mathcal{B}$ runs $(Q_i, Q_i^{'}, D_i^{'}) \leftarrow \mathsf{POTReq}(crs, pk_{\mathcal{V}}, T, D_{i-1}^{'}, \tau)$ to compute a request message $Q_i$, trapdoor information $Q_i^{'}$ and opening $D_i^{'}$, and $\omega_i \leftarrow \mathsf{OFReq}(sk_{\mathcal{B}}, apk, i, Q_i)$ to obtain a verifiably encrypted signature $\omega_i$ on $Q_i$. $\mathcal{B}$ sends the request $(i, Q_i, \omega_i)$ and stores $(Q_i^{'}, D_i^{'})$. Upon receiving $(i, Q_i, \omega_i)$, if $i$ does not equal the number of the transfer $t$, $\mathcal{V}$ does nothing. Otherwise, on input $(\text{response}, b)$, if $b = 0$ $\mathcal{V}$ sends $(\perp)$ to $\mathcal{B}$. If $b = 1$ $\mathcal{V}$ runs $\mathsf{POTVerReq}(crs, pk_{\mathcal{V}}, D_{i-1}, Q_i)$ and $\mathsf{OFVerReq}(pk_{\mathcal{B}}, apk, \omega_i, i, Q_i)$. If any of the two algorithms outputs reject, $\mathcal{V}$ rejects the request. Otherwise $\mathcal{V}$ runs $(R) \leftarrow \mathsf{POTResp}(crs, pk_{\mathcal{V}}, sk_{\mathcal{V}}, Q_i)$, sends the response $(R)$ and keeps state $(i, Q_i, \omega_i)$. If the output of $\mathsf{POTVerResp}(crs, pk_{\mathcal{V}}, R)$ is reject, $\mathcal{B}$ outputs $(\text{response}, i, \perp)$ and proceeds with the complaint phase. Otherwise, $\mathcal{B}$ runs $(m_\tau) \leftarrow \mathsf{POTComplete}(crs, T, R, Q_i^{'})$ and outputs $(\text{response}, i, m_\tau)$. On input $(\text{complete}, c, i)$, if $c = 0$, $\mathcal{B}$ sends $(\perp)$, and, if $c = 1$, $\mathcal{B}$ runs $\sigma_i \leftarrow \mathsf{OFComplete}(sk_{\mathcal{B}}, i, Q_i)$ and sends $(\sigma_i)$. $\mathcal{B}$ stores state information $B_i = (sk_{\mathcal{B}}, T, pk_{\mathcal{B}}, pk_{\mathcal{V}}, apk, D_i^{'})$. $\mathcal{V}$ runs $\mathsf{OFVerComp}(pk_{\mathcal{B}}, \sigma_i, i, Q_i)$ and, if the output is reject, $\mathcal{V}$ outputs $(\text{complete}, i, 0)$ and proceeds with the complaint phase. Otherwise $\mathcal{V}$ stores state information $V_i = (sk_{\mathcal{V}}, T, pk_{\mathcal{V}}, pk_{\mathcal{B}}, apk, Q_i, \sigma_i)$, increments the number of the transfer $t = t + 1$ and outputs $(\text{complete}, i, 1)$.

**Vendor complaint.** On input the tuple $(\text{complain}_{\mathcal{V}}, i)$, if no valid request $(i, Q_i, \omega_i)$ was previously received, $\mathcal{V}$ does nothing. Otherwise $\mathcal{V}$ runs $\mathsf{POTResp}(crs, pk_{\mathcal{V}}, sk_{\mathcal{V}}, Q_i)$ and sends the request-response pair $((i, Q_i, \omega_i), (R))$ to $\mathcal{A}$, along with the signature of the previous transfer $(i-1, Q_{i-1}, \sigma_{i-1})$ $((0, D_0, \sigma_0)$ when $i = 1)$. $\mathcal{A}$ parses $Q_{i-1}$ to obtain $D_{i-1}$ and runs $\mathsf{POTVerReq}(crs, pk_{\mathcal{V}}, D_{i-1}, Q_i)$, $\mathsf{OFVerReq}(pk_{\mathcal{B}}, apk, \omega_i, i, Q_i)$, $\mathsf{OFVerComp}(pk_{\mathcal{B}}, \sigma_{i-1}, i-1, Q_{i-1})$ $(\mathsf{OFInitV}(pk_{\mathcal{B}}, \sigma_0, D_0)$ when $i = 1)$ and $\mathsf{POTVerResp}(crs, pk_{\mathcal{V}}, R)$ and, if any of them outputs reject, sets $\sigma_i = \perp$ and $R = \perp$ and outputs $(\text{complain}_{\mathcal{V}}, i, \text{inv})$. Otherwise $\mathcal{A}$ runs $\sigma_i \leftarrow \mathsf{OFAdj}(pk_{\mathcal{B}}, ask, apk, \omega_i, i, Q_i)$. $\mathcal{A}$ sends $\sigma_i$ to $\mathcal{V}$ and $(i, R)$ to $\mathcal{B}$ and outputs $(\text{complain}_{\mathcal{V}}, i, \text{solved})$. $\mathcal{V}$ runs $\mathsf{OFVerComp}(pk_{\mathcal{B}}, \sigma_i, i, Q_i)$ and outputs $(\text{compSolved}_{\mathcal{V}})$ if the output is accept. $\mathcal{B}$ runs $(m_\tau) \leftarrow \mathsf{POTComplete}(crs, T, R, Q_i^{'})$, where $Q_i^{'}$ corresponds to the request for transfer $i$, and outputs $(\text{compResp}_{\mathcal{V}}, i, m_\tau)$.

**Buyer complaint.** On input $(\text{complain}_{\mathcal{B}}, i, \tau)$, if a request $(i, Q_i, \omega_i)$ was previously computed $\mathcal{B}$ sends it to $\mathcal{A}$. Otherwise, $\mathcal{B}$ sends to $\mathcal{A}$ a new re-

quest computed by running $\mathsf{POTReq}$ and $\mathsf{OFReq}$. $\mathcal{A}$ sends $(i)$ to $\mathcal{V}$, which returns $(j, Q_j, \sigma_j)$ ($(0, D_0, \sigma_0)$ if $i = 1$). Then $\mathcal{A}$ proceeds as follows:

- If $\sigma_j$ is invalid, $\mathcal{A}$ sends $(\text{guilty}_{\mathcal{V}})$ to $\mathcal{V}$ and $\mathcal{B}$ and outputs $(\text{complain}_{\mathcal{B}}, i, \text{guilty}_{\mathcal{V}})$.
- If $\sigma_j$ is valid but $j \geq i$, $\mathcal{A}$ sends $(\text{compInv}_{\mathcal{B}})$ to $\mathcal{B}$ and outputs $(\text{complain}_{\mathcal{B}}, i, \text{inv})$.
- If $\sigma_j$ is valid and $j = i - 1$, $\mathcal{A}$ parses $Q_{i-1}$ to obtain $D_{i-1}$, runs $\mathsf{POTVerReq}(crs, pk_{\mathcal{V}}, D_{i-1}, Q_i)$ and, if the output is $\mathsf{reject}$, sends $(\text{compInv}_{\mathcal{B}})$ to $\mathcal{B}$ and outputs $(\text{complain}_{\mathcal{B}}, i, \text{inv})$. $\mathcal{V}$ (possibly) sends another request $(i, Q_i', \omega_i')$. $\mathcal{A}$ verifies it and, if it is correct, but $(i, Q_i', \omega_i') \neq (i, Q_i, \omega_i)$, $\mathcal{A}$ sends $(\text{compInv}_{\mathcal{B}})$ to $\mathcal{B}$ and outputs $(\text{complain}_{\mathcal{B}}, i, \text{inv})$. Otherwise $\mathcal{A}$ sends $(i, Q_i, \omega_i)$ to $\mathcal{V}$. $\mathcal{V}$ verifies the request, runs $(R) \leftarrow \mathsf{POTResp}(crs, T, sk_{\mathcal{V}}, Q_i)$ and sends $(R)$ to $\mathcal{A}$. $\mathcal{A}$ verifies the response via $\mathsf{POTVerResp}(crs, pk_{\mathcal{V}}, R)$ and, if it is not valid, $\mathcal{A}$ sends $(\text{guilty}_{\mathcal{V}})$ to $\mathcal{B}$ and $\mathcal{V}$ and outputs $(\text{complain}_{\mathcal{B}}, i, \text{guilty}_{\mathcal{V}})$. If it is valid, $\mathcal{A}$ sends $R$ to $\mathcal{B}$ and $\sigma_i \leftarrow \mathsf{OFAdj}(pk_{\mathcal{B}}, ask, apk, \omega_i, i, Q_i)$ to $\mathcal{V}$, and outputs $(\text{complain}_{\mathcal{B}}, i, \text{solved})$. $\mathcal{B}$ retrieves $m_\tau$ via $\mathsf{POTComplete}$ and outputs $(\text{compResp}_{\mathcal{B}}, i, m_\tau)$, while $\mathcal{V}$ outputs $\text{compSolved}_{\mathcal{B}}$.

Algorithms POT* correspond to those of the POT schemes (see Section 2.4 and the full version [36]). Algorithms OF* are defined as follows:

$\mathsf{OFInitB}(crs, D_0')$ chooses a collision resistant hash function $H : \{0,1\}^* \rightarrow \mathbb{Z}_p$ and runs $(sk, pk) \leftarrow \mathsf{Kg}(1^\kappa)$ of the VES scheme for the pairing group setup contained in $crs$. Then it parses $D_0'$ to obtain the commitment to the deposit $D_0$ and signs $\sigma_0 \leftarrow \mathsf{Sign}(sk, H(0, D_0))$. It outputs $sk_{\mathcal{B}} = sk$, $pk_{\mathcal{B}} = (H, pk)$ and $\sigma_0$.

$\mathsf{OFInitV}(pk_{\mathcal{B}}, \sigma_0, D_0)$ outputs the result of $\mathsf{Vf}(pk, \sigma_0, H(0, D_0))$.

$\mathsf{OFInitA}(crs)$ runs $\mathsf{AdjKg}(1^\kappa)$ and outputs a key pair $(ask, apk)$ for the pairing group setup contained in $crs$.

$\mathsf{OFReq}(sk_{\mathcal{B}}, apk, i, Q_i)$ outputs a verifiably encrypted signature $\omega_i \leftarrow \mathsf{Create}(sk_{\mathcal{B}}, apk, H(i, Q_i))$.

$\mathsf{OFVerReq}(pk_{\mathcal{B}}, apk, \omega_i, i, Q_i)$ outputs the result of $\mathsf{VesVf}(pk, apk, \omega_i, H(i, Q_i))$.

$\mathsf{OFComplete}(sk_{\mathcal{B}}, i, Q_i)$ outputs $\sigma_i \leftarrow \mathsf{Sign}(sk_{\mathcal{B}}, H(i, Q_i))$.

$\mathsf{OFVerComp}(pk_{\mathcal{B}}, \sigma_i, i, Q_i)$ outputs $\mathsf{Vf}(pk, \sigma_i, H(i, Q_i))$.

$\mathsf{OFAdj}(pk_{\mathcal{B}}, ask, apk, \omega_i, i, Q_i)$ outputs $\mathsf{Adj}(pk, ask, apk, \omega, H(i, Q_i))$.

**Theorem 2.** *This OFPOT scheme securely realizes* $\mathcal{F}_{\text{OFPOT}}$.

We prove this theorem in the full version [36]. We prove security under static corruptions and under the assumption that $\mathcal{A}$ is neutral. If the $\mathcal{A}$ colludes either with $\mathcal{V}$ or with $\mathcal{B}$, the privacy properties of the underlying POT scheme still hold, but fairness does not hold anymore. Buyer security holds under the assumption

that buyer's security in the POT scheme holds, under the unforgeability and opacity properties of the VES scheme, and under the assumption that $H$ is collision-resistant. Vendor security holds under the assumption that vendor's security in the POT scheme holds and under the extractability of the VES scheme. We note that the proof that buyer's privacy also holds with respect to $\mathcal{A}$ follows the proof given with respect to $\mathcal{V}$, because the view of $\mathcal{A}$ and $\mathcal{V}$ with respect to $\mathcal{B}$ is equivalent.

As for efficiency, we note that the overhead in terms of computation and communication cost introduced by the VES scheme are small compared to the cost of the POT scheme. Therefore, a secure POT can efficiently be extended to provide optimistic fair exchange.

## 4 Conclusion

Our contribution is twofold. First, we have designed a full-simulation secure POT scheme that is more efficient than previous work. Second, we have proposed a generic construction that provides any secure POT scheme with optimistic fair exchange.

We leave open the definition of a more general ideal functionality for fair privacy-preserving e-commerce protocols. Another interesting open problem is the design of a fair POT scheme that is abuse-free in the sense of [40]. Further research also needs to be conducted to show how to integrate e-commerce protocols based on POT with digital rights management systems, and to analyze the compliance of such e-commerce protocols with e-commerce legislation.

## Acknowledgements

This work was supported in part by the IAP Programme P6/26 BCRYPT of the Belgian State. Alfredo Rial is funded by the Research Foundation - Flanders (FWO).

## References

1. Ackerman, M.S., Cranor, L.F., Reagle, J.: Privacy in e-commerce: examining user scenarios and privacy preferences. In: ACM Conference on Electronic Commerce, pp. 1–8 (1999)
2. Tsai, J., Egelman, S., Cranor, L., Acquisti, R.: The effect of online privacy information on purchasing behavior: An experimental study (June 2007) (working paper)
3. Enforcing privacy promises: Section 5 of the ftc act. Federal Trade Commission Act, `http://www.ftc.gov/privacy/privacyinitiatives/promises.html`
4. Kremer, S.: Formal analysis of optimistic fair exchange protocols (2004)
5. Aiello, W., Ishai, Y., Reingold, O.: Priced oblivious transfer: How to sell digital goods. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 119–135. Springer, Heidelberg (2001)

6. Tobias, C.: Practical oblivious transfer protocols. In: Petitcolas, F.A.P. (ed.) IH 2002. LNCS, vol. 2578, pp. 415–426. Springer, Heidelberg (2003)
7. Rial, A., Kohlweiss, M., Preneel, B.: Universally composable adaptive priced oblivious transfer. In: [42], pp. 231–247
8. Even, S., Goldreich, O., Lempel, A.: A randomized protocol for signing contracts. Commun. ACM 28(6), 637–647 (1985)
9. Goldreich, O.: A simple protocol for signing contracts. In: CRYPTO, pp. 133–136 (1983)
10. Bao, F., Deng, R.H., Mao, W.: Efficient and practical fair exchange protocols with off-line ttp. In: IEEE Symposium on Security and Privacy, pp. 77–85. IEEE Computer Society, Los Alamitos (1998)
11. Asokan, N., Shoup, V., Waidner, M.: Optimistic fair exchange of digital signatures (extended abstract). In: Nyberg, K. (ed.) EUROCRYPT 1998. LNCS, vol. 1403, pp. 591–606. Springer, Heidelberg (1998)
12. Avoine, G., Vaudenay, S.: Optimistic fair exchange based on publicly verifiable secret sharing. In: Wang, H., Pieprzyk, J., Varadharajan, V. (eds.) ACISP 2004. LNCS, vol. 3108, pp. 74–85. Springer, Heidelberg (2004)
13. Boneh, D., Gentry, C., Lynn, B., Shacham, H.: Aggregate and verifiably encrypted signatures from bilinear maps. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 416–432. Springer, Heidelberg (2003)
14. Lu, S., Ostrovsky, R., Sahai, A., Shacham, H., Waters, B.: Sequential aggregate signatures and multisignatures without random oracles. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 465–485. Springer, Heidelberg (2006)
15. Rückert, M., Schröder, D.: Security of verifiably encrypted signatures and a construction without random oracles. In: [42], pp. 17–34
16. Ray, I., Ray, I.: An anomymous fair exchange e-commerce protocol. In: IPDPS, p. 172. IEEE Computer Society, Los Alamitos (2001)
17. Naor, M., Pinkas, B.: Computationally secure oblivious transfer. J. Cryptology 18(1), 1–35 (2005)
18. Camenisch, J., Neven, G., Shelat, A.: Simulatable adaptive oblivious transfer. In: Naor, M. (ed.) EUROCRYPT 2007. LNCS, vol. 4515, pp. 573–590. Springer, Heidelberg (2007)
19. Boneh, D., Boyen, X.: Short signatures without random oracles. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 56–73. Springer, Heidelberg (2004)
20. Camenisch, J., Dubovitskaya, M., Neven, G.: Oblivious transfer with access control. In: Al-Shaer, E., Jha, S., Keromytis, A.D. (eds.) ACM Conference on Computer and Communications Security, pp. 131–140. ACM, New York (2009)
21. Boneh, D., Boyen, X., Goh, E.J.: Hierarchical identity based encryption with constant size ciphertext. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 440–456. Springer, Heidelberg (2005)
22. Goldwasser, S., Micali, S., Rivest, R.L.: A digital signature scheme secure against adaptive chosen-message attacks. SIAM J. Comput. 17(2), 281–308 (1988)
23. Pedersen, T.P.: Non-interactive and information-theoretic secure verifiable secret sharing. In: Feigenbaum, J. (ed.) CRYPTO 1991. LNCS, vol. 576, pp. 129–140. Springer, Heidelberg (1992)
24. Bellare, M., Goldreich, O.: On defining proofs of knowledge. In: Brickell, E.F. (ed.) CRYPTO 1992. LNCS, vol. 740, pp. 390–420. Springer, Heidelberg (1993)
25. Schnorr, C.P.: Efficient signature generation for smart cards. Journal of Cryptology 4(3), 239–252 (1991)

26. Chaum, D., Pedersen, T.P.: Wallet databases with observers. In: Brickell, E.F. (ed.) CRYPTO 1992. LNCS, vol. 740, pp. 89–105. Springer, Heidelberg (1993)
27. Camenisch, J., Chaabouni, R., Shelat, A.: Efficient protocols for set membership and range proofs. In: [41], pp. 234–252
28. Cramer, R., Damgård, I., Schoenmakers, B.: Proofs of partial knowledge and simplified design of witness hiding protocols. In: Desmedt, Y.G. (ed.) CRYPTO 1994. LNCS, vol. 839, pp. 174–187. Springer, Heidelberg (1994)
29. Damgård, I.: Concurrent zero-knowledge is easy in practice. Available online at Theory of Cryptography Library (June 1999)
30. Damgård, I.: On $\sigma$-protocols (2002), `http://www.daimi.au.dk/~ivan/Sigma.ps`
31. Fiat, A., Shamir, A.: How to prove yourself: Practical solutions to identification and signature problems. In: Odlyzko, A.M. (ed.) CRYPTO 1986. LNCS, vol. 263, pp. 186–194. Springer, Heidelberg (1987)
32. Camenisch, J., Stadler, M.: Proof systems for general statements about discrete logarithms. Technical Report TR 260, Institute for Theoretical Computer Science, ETH Zürich (March 1997)
33. Canetti, R.: Universally composable security: A new paradigm for cryptographic protocols. In: Proc. 42nd IEEE Symposium on Foundations of Computer Science (FOCS), pp. 136–145 (2001)
34. Green, M., Hohenberger, S.: Universally composable adaptive oblivious transfer. In: [41], pp. 179–197
35. Bellare, M., Rogaway, P.: Random oracles are practical: A paradigm for designing efficient protocols. In: First ACM Conference on Computer and Communication Security, Association for Computing Machinery, pp. 62–73 (1993)
36. Rial, A., Preneel, B.: Optimistic fair priced oblivious transfer (2009), `http://www.cosic.esat.kuleuven.be/publications/article-1428.pdf`
37. Okada, Y., Manabe, Y., Okamoto, T.: An optimistic fair exchange protocol and its security in the universal composability framework. IJACT 1(1), 70–77 (2008)
38. Canetti, R.: Obtaining universally compoable security: Towards the bare bones of trust. In: Kurosawa, K. (ed.) ASIACRYPT 2007. LNCS, vol. 4833, pp. 88–112. Springer, Heidelberg (2007)
39. Herranz, J.: Restricted adaptive oblivious transfer. Cryptology ePrint Archive, Report 2008/182 (2008), `http://eprint.iacr.org/`
40. Garay, J., Jakobsson, M., MacKenzie, P.: Abuse-free optimistic contract signing. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 449–466. Springer, Heidelberg (1999)
41. Pieprzyk, J. (ed.): ASIACRYPT 2008. LNCS, vol. 5350. Springer, Heidelberg (2008)
42. Shacham, H., Waters, B. (eds.): Pairing 2009. LNCS, vol. 5671. Springer, Heidelberg (2009)