A Privacy-Preserving Buyer-Seller Watermarking Protocol Based on Priced Oblivious Transfer

Alfredo Rial, Josep Balasch, and Bart Preneel, *Member, IEEE*

***Abstract*—Buyer-seller watermarking protocols allow copyright protection of digital goods. To protect privacy, some of those protocols provide buyers with anonymity. However, anonymous e-commerce protocols pose several disadvantages, like hindering customer management or requiring anonymous payment mechanisms. Additionally, no existing buyer-seller watermarking protocol provides fair exchange.**

We propose a novel approach for the design of privacy-preserving buyer-seller watermarking protocols. In our approach, the seller authenticates buyers but does not learn which items are purchased. Since buyers are not anonymous, customer management is eased and currently deployed methods of payment can be utilized.

We define an ideal functionality for privacy-preserving copyright protection protocols. To realize our functionality, a protocol must ensure that buyers pay the right price without disclosing the purchased item, and that sellers are able to identify buyers that released pirated copies. We construct a protocol based on priced oblivious transfer and on existing techniques for asymmetric watermark embedding. Furthermore, we implement and evaluate the efficiency of our protocol, and we explain how to extend it in order to achieve optimistic fair exchange.

***Index Terms*—Buyer-Seller Watermarking Protocol, Priced Oblivious Transfer, Fair Exchange.**

I. Introduction

The rapid development of communication networks has led to a situation that facilitates online e-commerce of digital goods. However, it also poses threats to copyright protection and to customers' privacy. On the one hand, distribution of illegal copies is eased, and thus mechanisms that allow the protection of intellectual property rights are needed. On the other hand, information about which items are bought can reveal sensitive data about the buyer. This information can easily be shared among service providers to create personal profiles. Consequently, privacy concerns discourage online e-commerce [1], and regulations to enforce privacy protection are being promulgated [2].

Previous work. Fingerprinting schemes deter people from illegally redistributing digital copies by enabling the seller of the data to identify the buyer. A scheme is said to be collusion-resistant [3] when it prevents a collusion of buyers up to a maximum size from producing non-traceable copies. In asymmetric fingerprinting schemes [4], the fingerprinted copy is only known to the buyer at the end of the purchase protocol. Thanks to this property, when the seller finds a redistributed copy, he can present it as a proof of buyer's misbehavior, and the buyer cannot claim that the copy was produced by the seller. In order to protect privacy, fingerprinting protocols that provide buyers with anonymity have been proposed [5].

Buyer-seller watermarking protocols [6] are asymmetric fingerprinting schemes in which the fingerprint is embedded by means of watermarking techniques. The basic idea is that each buyer obtains a slightly different copy of the digital content. Such difference, the watermark, does not harm the quality of the copy and cannot be removed by the buyer. Some buyer-seller watermarking protocols also provide buyers with anonymity [7], [8], [9].

As noted in [10], anonymous e-commerce protocols have several disadvantages. First, they hinder customer management. For example, the seller cannot give discounts to regular buyers or apply other loyalty marketing techniques. Second, they have to be used together with anonymous payment protocols (e.g. anonymous e-cash [11]), which makes it impossible the use of currently deployed payment protocols. Finally, they require the use of an underlying anonymous communication network, such as Tor [12]. It is well-known that achieving strong anonymity in such networks is a difficult goal [13]. Furthermore, some applications allow side-channel attacks against anonymity. For example, in location-based services, the service provider learns customer's location, and this information can be used to identify the a priori anonymous customer [14].

Additionally, e-commerce protocols are usually analyzed in order to prove their fairness [15]. Roughly speaking, fair exchange ensures that, at the end of the transaction, either the seller receives the payment and the buyer receives the purchased item, or both parties receive nothing. However, to the best of our knowledge, no fair buyer-seller watermarking protocol has been proposed.

Our contribution. We propose a different approach to provide privacy protection in buyer-seller watermarking protocols. In our approach, based on oblivious e-commerce protocols, buyers are authenticated by the seller, but the seller does not learn which items are purchased. This overcomes the disadvantages of anonymous purchase. Since buyers are authenticated, customer management is eased and currently deployed methods of payment can be utilized. As possible disadvantages, one can argue that the seller can find it difficult

This work was supported in part by the Research Council K.U.Leuven: GOA TENSE, and by the IAP Programme P6/26 BCRYPT of the Belgian State (Belgian Science Policy). The work of A. Rial was supported by the Research Foundation - Flanders (FWO). Josep Balasch is funded by a PhD grant within the covenant between K.U.Leuven and R.U.Nijmegen universities.

Alfredo Rial, Josep Balasch, and Bart Preneel are with IBBT and the COSIC group of Departement Elektrotechniek (ESAT), Katholieke Universiteit Leuven, Kasteelpark Arenberg 10, B-3001, Leuven, Belgium (phone: +32 (0)16 32 96 06, fax:+32 (0)16 32 19 69, e-mail: firstname.lastname@esat.kuleuven.be).

to learn which items are more demanded, e.g., to adapt his catalogue of products or to pay copyright owners. However, as noted in [10], this information can be obtained by other means, e.g., by conducting marketing researches.

We define formally privacy-preserving buyer-seller watermarking protocols, i.e., buyer-seller watermarking protocols in which the seller does not learn which items are purchased. We also provide a construction of such a protocol based on existing techniques for asymmetric watermark embedding and on priced oblivious transfer. (Priced oblivious transfer is the key building block of oblivious e-commerce protocols.) Finally, we explain how to extend our protocol to provide fair exchange.

Outline of the paper. In Section II we recall the definition of priced oblivious transfer and we define privacy-preserving buyer-seller watermarking protocols. We recall the definition of watermarking and of other cryptographic building blocks utilized in our construction in Section III. In Section IV we describe our construction, and we analyze its security in Section V. Additionally, we explain how to extend our construction to achieve fairness in Section VI, and we discuss its efficiency in Section VII. Finally, Section VIII draws a conclusion and discusses future work.

II. Definitions

A. Security Model

We define security following the ideal-world/real-world paradigm [16]. In the real world, a set of parties interact according to the protocol description in the presence of a real adversary $\mathcal{A}$, 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 $\mathcal{E}$. A protocol ψ is secure if there exists no environment $\mathcal{Z}$ that can distinguish whether it is interacting with adversary $\mathcal{A}$ and parties running protocol ψ or with the ideal process for carrying out the desired task, where ideal adversary $\mathcal{E}$ and dummy parties interact with an ideal functionality $\mathcal{F}_\psi$. More formally, we say that protocol ψ emulates the ideal process when, for any adversary $\mathcal{A}$, there exists a simulator $\mathcal{E}$ such that for all environments $\mathcal{Z}$, the ensembles $\mathrm{IDEAL}_{\mathcal{F}_\psi,\mathcal{E},\mathcal{Z}}$ and $\mathrm{REAL}_{\psi,\mathcal{A},\mathcal{Z}}$ are computationally indistinguishable. By applying the universal composition theorem [16], a protocol ψ that realizes functionality $\mathcal{F}_\psi$ remains secure even when it is composed along with an unbounded number of protocol instances controlled by the adversary. We refer to [16] for a description of how these ensembles are constructed. 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 the description of our ideal functionalities.

B. Priced Oblivious Transfer

POT [17] is a two-party protocol between a seller $\mathcal{S}$ and a buyer $\mathcal{B}$, where $\mathcal{S}$ sells a set of messages $m_1, \ldots, m_N$ with prices $p_1, \ldots, p_N$ to $\mathcal{B}$. At each purchase, $\mathcal{B}$ chooses $\tau \in \{1, \ldots, N\}$, gets m_τ and pays p_τ. $\mathcal{S}$ must learn neither τ nor p_τ, while $\mathcal{B}$ must not learn anything about the other messages. We recall the ideal functionality $\mathcal{F}_{\mathrm{POT}}$ for priced oblivious transfer in [10].

Functionality $\mathcal{F}_{\mathrm{POT}}$

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 seller $\mathcal{S}$ and buyers $\mathcal{B}_1, \ldots, \mathcal{B}_n$, $\mathcal{F}_{\mathrm{POT}}$ works as follows:

- On input a message $(\text{init}, m_1, p_1, \ldots, m_N, p_N)$ from $\mathcal{S}$, where each $m_i \in \{0,1\}^l$ and each $p \in [0, p_{max}]$, it stores $(m_1, p_1, \ldots, m_N, p_N)$ and sends $(\text{init}, p_1, \ldots, p_N)$ to each buyer $\mathcal{B}$.
- On input a message $(\text{deposit}, ac_i)$ from $\mathcal{B}_i$, where $ac_i \in [0, D_{max}]$, if a $(\text{init}, \ldots)$ message was not received before, then it does nothing. Otherwise, it stores $(\mathcal{B}_i, ac_i)$ and sends $(\text{deposit}, \mathcal{B}_i, ac_i)$ to $\mathcal{S}$.
- On input a message $(\text{request}, \tau)$ from $\mathcal{B}_i$, where $\tau \in [1, N]$, if message $(\text{init}, \ldots)$ was not received, $(\text{deposit}, ac_i)$ was not sent by $\mathcal{B}_i$ before or $ac_i - p_\tau < 0$, then it does nothing. Otherwise, it sends $(\text{request}, \mathcal{B}_i)$ to $\mathcal{S}$. If $\mathcal{S}$ is corrupted, $\mathcal{F}_{\mathrm{POT}}$ receives a bit $(\text{response}, b)$ from $\mathcal{E}$, and otherwise $\mathcal{F}_{\mathrm{POT}}$ sets $b = 1$. If $b = 0$, it sends $(\text{response}, \perp)$ to $\mathcal{B}_i$. If $b = 1$, it updates $ac_i = ac_i - p_\tau$ and sends $(\text{response}, m_\tau)$ to $\mathcal{B}_i$.

C. Privacy-Preserving Copyright Protection Protocol

We describe an ideal functionality $\mathcal{F}_{\mathrm{PCP}}$ that models the behavior and desirable properties of any privacy-preserving copyright protection protocol, i.e., a copyright protection protocol in which buyers do not disclose to the seller which items are bought. We consider a setting with three parties: a seller $\mathcal{S}$ that sells protected messages y; a set of buyers $\mathcal{B}_1, \ldots, \mathcal{B}_n$ that purchase protected messages from $\mathcal{S}$; and a judge $\mathcal{J}$ that decides whether a buyer is guilty of releasing pirated copies. $\mathcal{F}_{\mathrm{PCP}}$ is parameterized with a set of parties $\mathcal{P}$ that contains the aforementioned entities.

$\mathcal{F}_{\mathrm{PCP}}$ models the properties that the protocol should fulfill under three assumptions. First, the judge $\mathcal{J}$ is never corrupted by the ideal adversary $\mathcal{E}$. Second, parties can be corrupted statically, i.e., the ideal adversary $\mathcal{E}$ decides at the beginning of the protocol execution the set of parties it wishes to corrupt and cannot modify this set throughout the execution. Finally, $\mathcal{F}_{\mathrm{PCP}}$ assumes that uncorrupted buyers never release pirated copies.

Under those assumptions, $\mathcal{F}_{\mathrm{PCP}}$ requires that buyers, after making an initial deposit, purchase items from the seller by disclosing neither the item bought nor the amount of money paid. This requirement must hold both when the seller is honest and when he is corrupted.

In addition, when the seller is uncorrupted, buyers receive a unique protected message y at each purchase. This unique protected message, when released as a pirated copy, can be traced back to the corrupted buyer that released it. In our

construction, unique protected messages are computed by embedding different watermarks into the original content.

When the seller $\mathcal{S}$ is corrupted, $\mathcal{F}_{\text{PCP}}$ does not require the seller to send unique protected messages y. However, it requires that $\mathcal{S}$ is not able to frame uncorrupted buyers, who by assumption do not release pirated copies.

Below we describe formally $\mathcal{F}_{\text{PCP}}$. In Section V we prove that our privacy-preserving buyer-seller watermarking protocol *realizes* functionality $\mathcal{F}_{\text{PCP}}$. This means that our protocol fulfills the aforementioned properties.

Functionality $\mathcal{F}_{\text{PCP}}$

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 seller $\mathcal{S}$, a judge $\mathcal{J}$ and buyers $\mathcal{B}_1, \ldots, \mathcal{B}_n$, $\mathcal{F}_{\text{PCP}}$ works as follows:

- On input a message $(\text{init}, m_1, p_1, \ldots, m_N, p_N)$ from $\mathcal{S}$, it checks that each $m_i \in \{0,1\}^l$ and each $p \in [0, p_{max}]$. For each buyer $\mathcal{B}_i$, it computes a unique protected message y_{ij} from message m_j ($j \in [1, N]$). If $\mathcal{S}$ is corrupted, $\mathcal{F}_{\text{PCP}}$ receives $(\text{init}, \{\mathcal{B}_i, y_{i1}, \ldots, y_{iN}\}_{i=1}^n, p_1, \ldots, p_N)$ from the ideal adversary $\mathcal{E}$. $\mathcal{F}_{\text{PCP}}$ stores $(\{\mathcal{B}_i, y_{i1}, \ldots, y_{iN}\}_{i=1}^n, p_1, \ldots, p_N)$ and sends $(\text{init}, p_1, \ldots, p_N)$ to each buyer $\mathcal{B}$.
- On input a message $(\text{deposit}, ac_i)$ from $\mathcal{B}_i$, where $ac_i \in [0, D_{max}]$, if a $(\text{init}, \ldots)$ message was not received before, then it does nothing. Otherwise, it stores $(\mathcal{B}_i, ac_i, d)$, where $d = 1$, and sends $(\text{deposit}, \mathcal{B}_i, ac_i)$ to $\mathcal{S}$.
- On input a message $(\text{request}, \tau)$ from $\mathcal{B}_i$, where $\tau \in [1, N]$, if message $(\text{init}, \ldots)$ was not received, $(\text{deposit}, ac_i)$ was not sent by $\mathcal{B}_i$ before or $ac_i - p_\tau < 0$, then it does nothing. Otherwise, it sends $(\text{request}, \mathcal{B}_i)$ to $\mathcal{S}$. If $\mathcal{S}$ is corrupted, $\mathcal{F}_{\text{POT}}$ receives a bit $(\text{response}, b)$ from $\mathcal{E}$, and otherwise $\mathcal{F}_{\text{POT}}$ sets $b = 1$. If $b = 0$, it sends $(\text{response}, \perp)$ to $\mathcal{B}_i$. If $b = 1$, it updates $ac_i = ac_i - p_\tau$ and sends $(\text{response}, y_{i\tau})$ to $\mathcal{B}_i$.
- On input a message $(\text{release}, y)$ from the ideal adversary $\mathcal{E}$, $\mathcal{F}_{\text{PCP}}$ searches for y in the tuples $\{\mathcal{B}_i, y_{i1}, \ldots, y_{iN}\}_{i=1}^n$ and, if found in the tuple of $\mathcal{B}_i$, sets d to 0 in $(\mathcal{B}_i, ac_i, d)$. Otherwise $\mathcal{F}_{\text{PCP}}$ stores $(\mathcal{E}, y)$.
- On input a message (detect, y), if y belongs to a buyer $\mathcal{B}_i$ such that $d = 1$ in the tuple $(\mathcal{B}_i, ac_i, d)$, $\mathcal{F}_{\text{PCP}}$ sends $(\text{detresp}, \mathcal{B}_i, \textit{not guilty})$ to $\mathcal{J}$ and $\mathcal{S}$, and otherwise it sends $(\text{detresp}, \mathcal{B}_i, \textit{guilty})$ to $\mathcal{J}$ and $\mathcal{S}$. If y is stored in a tuple $(\mathcal{E}, \ldots)$, it sends $(\text{detresp}, \mathcal{E}, \textit{guilty})$ to $\mathcal{J}$ and $\mathcal{S}$.

Informally speaking, the requirements a protocol should fulfill in order to realize functionality $\mathcal{F}_{\text{PCP}}$ can be summarized as follows:

- **Correctness.** The protocol should terminate successfully whenever its parties are honest.
- **Traceability.** Upon finding a pirated copy, $\mathcal{S}$ should always be able to trace and identify the buyer $\mathcal{B}$ that released it.
- **Non-frameability.** An honest buyer cannot be found guilty of releasing pirated copies.
- **Direct Non-repudiation.** A guilty buyer cannot deny she released the pirated copy. Additionally, $\mathcal{S}$ can convince a third party that $\mathcal{B}$ is guilty without needing interaction with $\mathcal{B}$.
- **Privacy.** $\mathcal{S}$ does not learn which contents are purchased.
- **Anti-fraud.** $\mathcal{S}$ is assured that $\mathcal{B}$ pays the right price and that $\mathcal{B}$ cannot purchase when running out of funds.

D. Registration Authority

In Section V we prove that our buyer-seller watermarking protocol *realizes* functionality $\mathcal{F}_{\text{PCP}}$ in the $\mathcal{F}_{\text{REG}}$-hybrid model, where parties register their public keys at a trusted registration entity. Below we depict the ideal functionality $\mathcal{F}_{\text{REG}}$ given in [16]. $\mathcal{F}_{\text{REG}}$ is parameterized with a set of participants $\mathcal{P}$, which is restricted to contain the buyers $\mathcal{B}_1, \ldots, \mathcal{B}_n$, the seller $\mathcal{S}$ and the judge $\mathcal{J}$. $\mathcal{F}_{\text{REG}}$ can be implemented with a public key infrastructure.

Functionality $\mathcal{F}_{\text{REG}}$

Parameterized with a set of parties $\mathcal{P}$, $\mathcal{F}_{\text{REG}}$ works as follows:

- Upon receiving $(\text{register}, v)$ from party $P \in \mathcal{P}$, it records the value (P, v).
- Upon receiving $(\text{retrieve}, P)$ from party $P' \in \mathcal{P}$, if (P, v) is recorded then return $(\text{retrieve}, P, v)$ to P'. Otherwise send $(\text{retrieve}, P, \perp)$ to P'.

III. Technical Preliminaries

We recall the definition of watermarking and of other cryptographic building blocks utilized in our construction. Some subsections and notation utilized here and in the following sections is taken verbatim from [18].

A. Blind Watermarking

A blind and readable watermarking scheme [19] consists of a setup algorithm $\mathsf{WATsetup}$, a watermark embedding algorithm WATemb and a watermark detection algorithm WATdet. $\mathsf{WATsetup}$ outputs a secret watermarking key swk, a message space $\mathcal{M}$ and a watermark space $\mathcal{W}$. $\mathsf{WATemb}(swk, m, w)$, on input swk, message $m \in \mathcal{M}$, and watermark $w \in \mathcal{W}$, outputs a watermarked message y. The algorithm WATemb can be computed in the encrypted domain, where both w and the result y are encrypted with a public key of a public key encryption scheme. The algorithm $\mathsf{WATdet}(swk, y)$ outputs the watermark w embedded in y.

A secure watermarking scheme should be robust and collusion resistant. Let d be a distortion metric that quantifies the distortion suffered by a watermarked content y when it underwent signal processing operations such as compression, filtering, noise addition, desynchronization, cropping, insertions,

mosaicing, and collage. Let y' be a distorted content. The robustness property requires that under a distortion metric d and a distortion bound D, given swk output by $\mathsf{WATsetup}$ and y output by $\mathsf{WATemb}(swk, m, w)$, $\mathsf{WATdet}(swk, y')$ outputs w with overwhelming probability if $d(y, y') \leq D$.

The collusion resistance property [20] requires that a collusion of up to l parties cannot manipulate or remove the watermark from a watermarked content by comparing or composing their differently watermarked copies. This property can be formalized as follows:

Definition 1 (Collusion Resistant Watermarking)**.** *The collusion resistance property is defined through the following game between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$.*

- *Challenge. $\mathcal{C}$ runs* $\mathsf{WATsetup}$ *to get swk, picks random original content $m \in \mathcal{M}$, and, for $i = 1$ to l, picks random watermark $w_i \in \mathcal{W}$ and runs $y_i = \mathsf{WATemb}(swk, m, w_i)$. $\mathcal{C}$ sends $(y_1, \ldots, y_l)$ to $\mathcal{A}$.*
- *Response. $\mathcal{A}$ outputs watermarked content y'.*

$\mathcal{A}$ wins if there exists $i \in [1, l]$ such that $d(y_i, y') \leq D$ and $\mathsf{WATdet}(swk, y')$ outputs a watermark w' such that, for $i = 1$ to l, $w' \neq w_i$. A blind watermarking scheme is l collusion resistant if all p.p.t. adversaries $\mathcal{A}$ win the game above with negligible probability.

Current practical watermarking schemes do not provide collusion-resistance against any p.p.t. adversary. In Section V we assume that the watermarking scheme used to instantiate the protocol fulfills this definition, and thus we conclude that our protocol is secure against any p.p.t. adversary. When the protocol is instantiated with a concrete watermarking scheme, the security offered against malicious buyers is lowered to the security offered by the watermarking scheme.

B. Signature Schemes

A signature scheme consists of the algorithms Keygen, Sign and $\mathsf{VerifySig}$. Keygen outputs a secret key sk and a public key pk. $\mathsf{Sign}(sk, m)$ outputs a signature s of message m. $\mathsf{VerifySig}(pk, m, s)$ outputs accept if s is a valid signature of m and reject otherwise. A signature scheme must be correct and unforgeable [21]. Informally speaking, correctness implies that the $\mathsf{VerifySig}$ algorithm always accepts an honestly generated signature. Existential unforgeability means that no p.p.t. adversary should be able to output a message-signature pair (s, m) unless he has previously obtained a signature on m. We can employ any existentially unforgeable signature scheme to instantiate the signature scheme $(\mathsf{Keygen}, \mathsf{Sign}, \mathsf{VerifySig})$ employed by buyers in our construction in Section IV.

C. Homomorphic Encryption

A public key encryption scheme consists of the algorithms Keygen, Enc and Dec. Keygen outputs a public key pk and a secret key sk. Enc outputs a ciphertext c on input a public key pk and a message m. Dec outputs the message m on input the ciphertext c and the secret key sk. Roughly speaking, indistinguishability under chosen plaintext attack [22] (IND-CPA) guarantees that an adversary does not get any knowledge about m from c.

We employ a homomorphic public key encryption scheme that supports two operations. An operation $\odot$ that, on input two ciphertexts $\mathsf{Enc}(pk, x)$ and $\mathsf{Enc}(pk, y)$ that encrypt messages x and y, outputs a ciphertext $\mathsf{Enc}(pk, x + y) = \mathsf{Enc}(pk, x) \odot \mathsf{Enc}(pk, y)$ that encrypts the addition of the messages, and an operation $\otimes$ that, on input a message x and a ciphertext $\mathsf{Enc}(pk, y)$, outputs a ciphertext $\mathsf{Enc}(pk, xy) = x \otimes \mathsf{Enc}(pk, y)$ that encrypts the multiplication of the messages x and y. The homomorphic public key encryption scheme proposed by Paillier [23], and its generalization by Damgård and Jurik [24], support these operations, and therefore can be used to instantiate the encryption scheme $(\mathsf{BKeygen}, \mathsf{BEnc}, \mathsf{BDec})$ employed in Section IV.

In our construction in Section IV, we need a function that, on input a bit b and an encryption $\mathsf{Enc}(pk, b')$ of a bit b', computes the encryption $\mathsf{Enc}(pk, b \oplus b')$, where $\oplus$ denotes the exclusive or operation. This function can be computed as follows. If $b = 0$, output $\mathsf{Enc}(pk, b')$. If $b = 1$, output $\mathsf{Enc}(pk, b) \odot (-1 \otimes \mathsf{Enc}(pk, b'))$.

D. Zero-Knowledge Proofs of Knowledge

A zero-knowledge proof of knowledge [25] 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 the 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 transcript that cannot be distinguished from the interaction with a real prover.

To express a zero-knowledge proof of knowledge, we follow the notation introduced by Camenisch and Stadler [26]. For example, $\mathsf{PK}\{(x) : y = f(x)\}$ denotes a "*zero-knowledge proof of knowledge of secret input x such that $y = f(x)$*". Letters in the parenthesis, in this example x, denote the secret input, while y and the function f are also known to the verifier.

We employ a proof of knowledge $\mathsf{PK}\{(sk') : (pk', sk') \leftarrow \mathsf{BKeygen}(1^k) \wedge C \leftarrow \mathsf{Enc}(pk, sk')\}$, i.e., a proof that C is a correct encryption under pk of the secret key sk' related with public key pk', so that a party in possession of the secret key sk related with pk can recover sk' from C. The verifiable encryption schemes proposed by Camenisch et al. [27] and by Poupard and Stern [28], which are provided with such a proof of knowledge, can be employed to instantiate the encryption scheme $(\mathsf{JKeygen}, \mathsf{JEnc}, \mathsf{JDec})$ used in our construction in Section IV.

We also use a proof of knowledge of the statement $\mathsf{PK}\{(b) : c \leftarrow \mathsf{Enc}(pk, b) \wedge b \in \{0, 1\}\}$, i.e., a proof that the value b encrypted in ciphertext c under public key pk is a bit. Such a proof is described in [24].

IV. Privacy-Preserving Buyer-Seller Watermarking Protocol

A. Intuition Behind Our Construction

Our privacy-preserving buyer-seller watermarking (PBSW) protocol is based on priced oblivious transfer (POT). POT allows buyers to purchase messages from the seller without the seller learning which messages are bought (see Section II). Existing secure POT schemes follow an assisted decryption approach in which the interaction between a seller $\mathcal{S}$ and a buyer $\mathcal{B}$ is divided into an initialization phase and several purchase phases. In the initialization phase, $\mathcal{S}$ encrypts the messages to be sold and sends the ciphertexts to $\mathcal{B}$. In each purchase phase, $\mathcal{S}$ helps $\mathcal{B}$ to decrypt one of the ciphertexts via an interactive protocol.

To allow for payments, existing POT schemes employ a prepaid mechanism. In the initialization phase, $\mathcal{B}$ makes an initial deposit to $\mathcal{S}$, and, in each purchase phase, $\mathcal{B}$ debits the price of the message from the deposit and proves to $\mathcal{S}$ that the remaining deposit is non-negative. $\mathcal{S}$ is able to verify those facts by learning neither the price of the message nor the new value of the deposit. We note that, to achieve full privacy, the initial deposit should be higher than the price of the most expensive item. Additionally, it is possible for the buyer to hide when she is buying something by having the seller include a dummy item with price zero.

Our PBSW protocol combines POT with existing techniques for asymmetric watermark embedding. In particular, we use a simplified version of the buyer-seller watermarking protocol in [18], in which buyers are not provided with anonymity. This protocol employs homomorphic encryption to allow $\mathcal{S}$ and $\mathcal{B}$ to jointly compute an encryption (with the public key of $\mathcal{B}$) of the watermark to be embedded in the message, in such a way that none of the parties knows the watermark. This prevents both a malicious seller from releasing pirated copies with the same watermark in order to frame the buyer, and a malicious buyer that releases pirated copies from invoking the possibility of being framed by the seller.

Additionally, the protocol involves a judge $\mathcal{J}$ to resolve disputes between $\mathcal{S}$ and $\mathcal{B}$. To this end, $\mathcal{B}$ sends to $\mathcal{S}$ a key escrow ciphertext that encrypts her secret key with the public key of the judge $\mathcal{J}$. When $\mathcal{S}$ accuses $\mathcal{B}$ of releasing pirated copies, $\mathcal{S}$ sends this ciphertext to $\mathcal{J}$ so that $\mathcal{J}$ can recover $\mathcal{B}$'s secret key. With the secret key, $\mathcal{J}$ can check whether $\mathcal{B}$ is guilty or not guilty.

Our PBSW protocol consists of four phases: setup, initialization, purchase and arbitration. In the setup phase, $\mathcal{B}$ and $\mathcal{J}$ create key pairs and register their public keys with the registration authority. $\mathcal{S}$ computes a secret watermarking key.

In the initialization phase, $\mathcal{S}$ and $\mathcal{B}$ run an interactive protocol for asymmetric watermark embedding. As result, $\mathcal{S}$ obtains a set of N watermarks, encrypted with the public key of $\mathcal{B}$, to be embedded in each of the N messages. Each watermark is of the form $w = \phi||(W_{\mathcal{S}} \oplus W_{\mathcal{B}})$, where $W_{\mathcal{S}}$ and $W_{\mathcal{B}}$ are randomly chosen by $\mathcal{S}$ and $\mathcal{B}$ respectively, and ϕ is a random string, unique for each released content, chosen by $\mathcal{S}$. By using signal processing in the encrypted domain techniques, $\mathcal{S}$ embeds the watermarks and obtains N encrypted watermarked messages. These watermarked messages are given as input to the seller initialization algorithm of the POT scheme, which produces N ciphertexts to be sent to $\mathcal{B}$.[1] After that, the initialization phase follows that of the POT scheme.

The purchase phases follow those of the POT scheme, except that as result $\mathcal{B}$ obtains a watermarked message encrypted with her public key. $\mathcal{B}$ decrypts the result to obtain the watermarked message. In the arbitration phase, when $\mathcal{S}$ claims to have found a pirated copy, $\mathcal{S}$ employs the watermark detection algorithm to obtain the watermark $\phi||(W_{\mathcal{S}} \oplus W_{\mathcal{B}})$. $\mathcal{S}$ sends $\mathcal{J}$ the identity of the alleged pirate, i.e., the buyer $\mathcal{B}$ that received the content whose watermark included ϕ. $\mathcal{J}$ checks whether $\mathcal{B}$ is guilty or not guilty.

Our construction employs authenticated channels. In its description we employ a single watermarking key swk. If, in order for the watermarking scheme to be collusion resistant, a different watermarking key should be associated to each of the original contents, our construction can be modified to watermark each content with a different key.

B. A Syntax for POT Schemes

Our construction can be instantiated with any POT scheme that realizes $\mathcal{F}_{\mathrm{POT}}$, such as [10], [29]. Following the syntax for POT proposed in [29], a POT scheme consists of the following algorithms.

- A seller initialization algorithm $\mathrm{POT}\mathsf{InitS}(1^\kappa, m_1, p_1, \ldots, m_N, p_N, D_{max})$ that, on input the security parameter 1^κ, message-price pairs $(m_1, p_1, \ldots, m_N, p_N)$ and the maximum deposit D_{max}, outputs a secret key $sk_{\mathcal{S}}$, a public key $pk_{\mathcal{S}}$ and a set of N ciphertexts T.
- A buyer initialization algorithm $\mathrm{POT}\mathsf{InitB}(1^\kappa, pk_{\mathcal{S}}, T, ac_0)$ that, on input the public key $pk_{\mathcal{S}}$, the ciphertexts T and the buyer's deposit ac_0, outputs a payment message P and buyer's state information $D_0^{'}$.
- An algorithm $\mathrm{POT}\mathsf{GetDep}(sk_{\mathcal{S}}, P, D_{max})$ that, on input the secret key $sk_{\mathcal{S}}$, the payment message P and the maximum deposit D_{max}, outputs the buyer's deposit ac_0 and seller's state information D_0.
- An algorithm $\mathrm{POT}\mathsf{Req}(pk_{\mathcal{S}}, T, D_{i-1}^{'}, \tau)$ that, on input the public key $pk_{\mathcal{S}}$, the ciphertexts T, buyer's state information $D_{i-1}^{'}$ and selection value τ, computes the request Q for the ith transfer, trapdoor information $Q^{'}$ and updated buyer's state information $D_i^{'}$.
- A request verification algorithm $\mathrm{POT}\mathsf{VerReq}(pk_{\mathcal{S}}, D_{i-1}, Q)$ that, on input the public key $pk_{\mathcal{S}}$, seller's state information D_{i-1} and request Q, verifies the request and outputs either accept or reject.
- An algorithm $\mathrm{POT}\mathsf{Resp}(sk_{\mathcal{S}}, pk_{\mathcal{S}}, Q)$ that, on input secret key $sk_{\mathcal{S}}$, public key $pk_{\mathcal{S}}$ and request Q, outputs a response R and updated seller's state information D_i.

[1] If the encrypted watermarked messages do not belong to the message space of the POT scheme used, its message space can be modified. For example, in the POT scheme in [29], a ciphertext of message m consists of elements (A, B), where $B = e(g, A) \oplus m$ for a bilinear map $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_t$ and a generator g of $\mathbb{G}$. A hash function H can be used to map $e(g, A)$ to the key space of a secure symmetric key encryption scheme. The message is then encrypted via the symmetric key encryption scheme utilizing $H(e(g, A))$ as key.

- A response verification algorithm $\mathrm{POTVerResp}(pk_{\mathcal{S}}, R)$ that, on input the public key $pk_{\mathcal{S}}$ and the response R, verifies the response and outputs either accept or reject.
- An algorithm $\mathrm{POTComplete}(T, R, Q^{'})$ that, on input the ciphertexts T, the response R and trapdoor information $Q^{'}$, outputs the message m_{τ}.

C. Construction

In the following, $(\mathsf{JKeygen}, \mathsf{JEnc}, \mathsf{JDec})$ and $(\mathsf{BKeygen}, \mathsf{BEnc}, \mathsf{BDec})$ stand for the algorithms for key generation, encryption and decryption of the public key encryption schemes used by $\mathcal{J}$ and $\mathcal{B}$ respectively. They are described in Section III. Additionally, $(\mathsf{Keygen}, \mathsf{Sign}, \mathsf{VerifySig})$ are the algorithms for key generation, signing and signature verification of the signature scheme used by $\mathcal{B}$.

In the setup phase, each buyer $\mathcal{B}_i$ runs Keygen to obtain a key pair (sk_i, pk_i) and registers pk_i at $\mathcal{F}_{\text{REG}}$. The judge $\mathcal{J}$ runs his key generation algorithm JKeygen in order to generate a key pair $(sk_{\mathcal{J}}, pk_{\mathcal{J}})$ and registers $pk_{\mathcal{J}}$ at $\mathcal{F}_{\text{REG}}$. Every party can retrieve public keys of other parties by querying $\mathcal{F}_{\text{REG}}$. The seller $\mathcal{S}$ executes the watermarking setup algorithm WATsetup to obtain a secret watermarking key swk. The following phases are depicted in Figure 1 and described below.

Protocol PBSW

- **Initialization.** When $\mathcal{S}$ is activated with the message $(\text{init}, m_1, p_1, \ldots, m_N, p_N)$ and each buyer $\mathcal{B}_i$ is activated with $(\text{deposit}, ac_i)$, $\mathcal{B}_i$ and $\mathcal{S}$ run algorithms $\mathsf{SetupB}(1^{\kappa}, sk_i, pk_{\mathcal{J}})$ and $\mathsf{SetupV}(pk_i, pk_{\mathcal{J}}, swk, m_1, \ldots, m_N)$ respectively. As results, $\mathcal{B}$ obtains a key pair $(sk_{\mathcal{B}}{}', pk_{\mathcal{B}}{}')$. $\mathcal{S}$ obtains encrypted watermarked messages $(ct_1, \ldots, ct_j)$ and state information $info$ that allows the identification of $\mathcal{B}_i$ when she releases a pirated copy related with $(ct_1, \ldots, ct_j)$. Then $\mathcal{S}$ runs $\mathrm{POTInitS}(1^{\kappa}, ct_1, p_1, \ldots, ct_N, p_N, D_{max})$ to obtain a secret key $sk_{\mathcal{S}}$ and a public key $pk_{\mathcal{S}}$ of the POT scheme, and a set of N ciphertexts T. $\mathcal{S}$ sends $(pk_{\mathcal{S}}, T)$ to $\mathcal{B}$. $\mathcal{B}$ runs $(P, D_0^{'}) \leftarrow \mathrm{POTInitB}(1^{\kappa}, pk_{\mathcal{S}}, T, ac_0)$ and aborts if the output is reject. Otherwise, $\mathcal{B}$ sends the payment message (P) to $\mathcal{S}$ and pays an amount of ac_0 through an arbitrary payment channel. $\mathcal{S}$ runs $(D_0, ac_0) \leftarrow \mathrm{POTGetDep}(sk_{\mathcal{S}}, P, D_{max})$ and checks that ac_0 corresponds to the amount of money received. $\mathcal{S}$ adds $(\mathcal{B}_i, info)$ to a buyer's database DB_{info}, stores state information $U_0 = (sk_{\mathcal{S}}, pk_{\mathcal{S}}, T, D_0)$, and outputs $(\text{deposit}, ac_0)$. $\mathcal{B}_i$ stores state information $U_0^{'} = (pk_{\mathcal{S}}, T, D_0^{'})$.
- **Purchase.** When $\mathcal{B}_i$ is activated with $(\text{request}, \tau_i)$, $\mathcal{B}_i$ runs $\mathrm{POTReq}(pk_{\mathcal{S}}, T, D_{i-1}^{'}, \tau_i)$ to get a request Q, a trapdoor $Q^{'}$ and private state $D_i^{'}$. $\mathcal{B}$ sends (Q) and stores $(Q^{'}, D_i^{'})$. $\mathcal{S}$ runs $\mathrm{POTVerReq}(pk_{\mathcal{S}}, D_{i-1}, Q)$ and ignores the request if the output is reject. Otherwise $\mathcal{S}$ runs $\mathrm{POTResp}(sk_{\mathcal{S}}, pk_{\mathcal{S}}, Q)$ to obtain a response R and state D_i, and sends (R) to $\mathcal{B}$. $\mathcal{B}$ runs $\mathrm{POTVerResp}(pk_{\mathcal{S}}, R)$ and outputs $(\text{response}, \perp)$ if the output is reject. Otherwise $\mathcal{B}$ runs $\mathrm{POTComplete}(T, R, Q^{'})$ to obtain ct_{τ_i} and runs $\mathsf{Dec}(sk_{\mathcal{B}}{}', ct_{\tau_i})$ to get y_{τ_i}, i.e., the message m_{τ_i} watermarked. $\mathcal{S}$ stores state information $U_i = (sk_{\mathcal{S}}, pk_{\mathcal{S}}, T, D_i)$, and $\mathcal{B}$ stores state information $U_i^{'} = (pk_{\mathcal{S}}, T, D_i^{'})$ and outputs $(\text{response}, y_{\tau_i})$.
- **Arbitration.** When $\mathcal{S}$ is activated with (detect, y), $\mathcal{S}$ runs $\mathsf{Detect}(swk, y, \mathrm{DB}_{info})$ to obtain a framing message F and sends (F) to $\mathcal{J}$. $\mathcal{J}$ retrieves the public key pk_i of the buyer $\mathcal{B}_i$ in F and runs $\mathsf{Check}(pk_i, F, sk_{\mathcal{J}})$ to obtain a bit b. If $b = 0$, $\mathcal{J}$ sends $(\textit{not guilty})$ to $\mathcal{S}$ and outputs $(\text{detresp}, \textit{not guilty})$. Otherwise $\mathcal{J}$ sends $(\textit{guilty})$ to $\mathcal{S}$ and outputs $(\text{detresp}, \textit{guilty})$.

- $\mathsf{SetupB}(1^{\kappa}, sk_i, pk_{\mathcal{J}})$. Run $\mathsf{BKeygen}(1^{\kappa})$ to obtain a key pair $(sk_{\mathcal{B}}{}', pk_{\mathcal{B}}{}')$. Run $\mathsf{JEnc}(pk_{\mathcal{J}}, sk_{\mathcal{B}}{}')$ to get an encryption C of $sk_{\mathcal{B}}{}'$. Pick a random string $W_{\mathcal{B}} \leftarrow \{0,1\}^{l_2}$ and, for $i = 1$ to l_2, run $c_i = \mathsf{BEnc}(pk_{\mathcal{B}}{}', W_{\mathcal{B}i})$ to encrypt bitwise $W_{\mathcal{B}}$. Set a message $m = (pk_{\mathcal{B}}{}', (c_i)_{i=1}^{l_2}, C)$ and run $\mathsf{Sign}(sk_i, m)$ to compute a signature s_m. (If m does not belong to the message space of the group signature scheme, use a collision-resistant hash function H to compute a hash $H(m)$ that belongs to the message space and sign $H(m)$.) Send (m, s_m) to $\mathcal{S}$. As the prover, engage with $\mathcal{S}$ in the following interactive zero-knowledge proofs of knowledge: a proof $\pi_1 = \mathsf{PK}\{(sk_{\mathcal{B}}{}') : (pk_{\mathcal{B}}{}', sk_{\mathcal{B}}{}') \leftarrow \mathsf{BKeygen}(1^k) \wedge C \leftarrow \mathsf{JEnc}(pk_{\mathcal{J}}, sk_{\mathcal{B}}{}')\}$ that $(pk_{\mathcal{B}}{}', sk_{\mathcal{B}}{}')$ are correctly setup and that C is an encryption of $sk_{\mathcal{B}}{}'$ with $pk_{\mathcal{J}}$; for $i = 1$ to l_2, a proof $\pi_{2i} = \mathsf{PK}\{(W_{\mathcal{B}i}) : c_i \leftarrow \mathsf{BEnc}(pk_{\mathcal{B}}{}', W_{\mathcal{B}i}) \wedge W_{\mathcal{B}i} \in \{0,1\}\}$ that each c_i encrypts a bit. Output a key pair $(sk_{\mathcal{B}}{}', pk_{\mathcal{B}}{}')$.
- $\mathsf{SetupV}(pk_i, pk_{\mathcal{J}}, swk, m_1, \ldots, m_N)$. Receive tuple (m, s_m) and parse the message m as $(pk_{\mathcal{B}}{}', (c_i)_{i=1}^{l_2}, C)$. Run $\mathsf{VerifySig}(pk_i, m, s_m)$ and abort if the output is reject. As the verifier, engage in the execution of the interactive proofs π_1 and, for $i = 1$ to l_2, π_{2i}, and abort if any of them is not correct. For $j = 1$ to N, pick random $W_{\mathcal{S}j} \leftarrow \{0,1\}^{l_2}$ and, for $i = 1$ to l_2, run $\mathsf{BEnc}(pk_{\mathcal{B}}{}', W_{\mathcal{S}ji} \oplus W_{\mathcal{B}i})$. Pick random unique $\phi_j \leftarrow \{0,1\}^{l_1}$ and, for $i = 1$ to l_1, encrypt $\mathsf{BEnc}(pk_{\mathcal{B}}{}', \phi_{ji})$. Set the watermark to be embedded as $w_j = \phi_j || (W_{\mathcal{S}j} \oplus W_{\mathcal{B}})$, and let its bitwise encryption be $\mathsf{BEnc}(pk_{\mathcal{B}}{}', w_j)$. Perform the watermark embedding operation $\mathsf{WATemb}(swk, m_j, \mathsf{BEnc}(pk_{\mathcal{B}}{}', w_j))$ in the encrypted domain to obtain an encrypted watermarked message $ct_j = \mathsf{BEnc}(pk_{\mathcal{B}}{}', y_j)$. Output encrypted watermarked messages $(ct_1, \ldots, ct_N)$ and state information $info = \{m, s_m, (\phi_j, W_{\mathcal{S}j})_{j=1}^{N}\}$.
- $\mathsf{Detect}(swk, y, \mathrm{DB}_{info})$. Execute the watermark detection algorithm $\mathsf{WATdet}(swk, y)$ to obtain the watermark $w = \phi || x$, parse the entry $(\mathcal{B}_i, \phi, W_{\mathcal{S}}, m, s_m)$ of table DB_{info}, compute $W_{\mathcal{B}} = W_{\mathcal{S}} \oplus x$ and output $F = (W_{\mathcal{B}}, m, s_m, \mathcal{B}_i)$.
- $\mathsf{Check}(pk_i, F, sk_{\mathcal{J}})$. Parse F as $(W_{\mathcal{B}}, m, s_m, \mathcal{B}_i)$ and m as $(pk_{\mathcal{B}}{}', (c_i)_{i=1}^{l_2}, C)$. Run $\mathsf{VerifySig}(pk_i, m, s_m)$ and

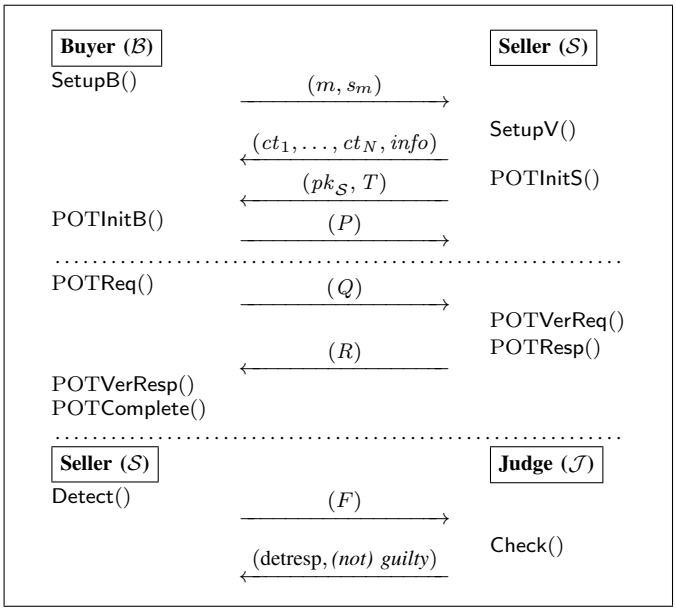


Fig. 1. Phases of PBSW protocol: initialization (top), purchase (middle), arbitration (bottom).

abort if the output is 0. Decrypt $\mathsf{JDec}(sk_{\mathcal{J}}, C)$ to obtain $sk_{\mathcal{B}}{}'$. For $i = 1$ to l_2, decrypt $\mathsf{BDec}(sk_{\mathcal{B}}{}', c_i)$ to obtain $W_{\mathcal{B}i}'$. Check whether $W_{\mathcal{B}}{}' = W_{\mathcal{B}}$. If it is the case, output $b = 1$ and otherwise $b = 0$.

V. Security Analysis

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

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 $\mathcal{F}_{\mathrm{PCP}}$ and environment $\mathcal{Z}$ in such a way that ensembles $\mathrm{IDEAL}_{\mathcal{F}_{\mathrm{PCP}},\mathcal{E},\mathcal{Z}}$ and $\mathrm{REAL}_{\mathrm{PCP},\mathcal{A},\mathcal{Z}}$ are computationally indistinguishable.

We prove security of our protocol under the assumption that it is instantiated with a secure POT scheme, i.e., one that realizes functionality $\mathcal{F}_{\mathrm{POT}}$ described in Section II. The security of the POT scheme implies the existence of a simulator $\mathcal{E}_{\mathrm{POT}}$ that interacts with $\mathcal{F}_{\mathrm{POT}}$ and $\mathcal{Z}$ in such a way that $\mathcal{Z}$ cannot distinguish the ideal world from the real world. Our simulator $\mathcal{E}$ employs simulator $\mathcal{E}_{\mathrm{POT}}$.

We analyze formally the security of our scheme when the seller and a subset of buyers are corrupted, and when (a subset of) buyers are corrupted. We denote by ν a negligible function.

A. Security Analysis When Seller Is Corrupted

Claim 1. *When the seller and a subset of the buyers are corrupted, the ensembles* $\mathrm{IDEAL}_{\mathcal{F}_{\mathrm{PCP}},\mathcal{E},\mathcal{Z}}$ *and* $\mathrm{REAL}_{\mathrm{PCP},\mathcal{A},\mathcal{Z}}$ *are computationally indistinguishable under the security of the* POT *scheme, the zero-knowledge property of the proofs of knowledge, the IND-CPA security of encryption schemes* (BKeygen, BEnc, BDec) *and* (JKeygen, JEnc, JDec)*, and the unforgeability of the signature scheme* (Keygen, Sign, VerifySig).

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

- **Game** 0: This game corresponds to the execution of the real-world protocol with a subset of honest buyers and an honest $\mathcal{J}$. Thus Pr [**Game** 0] $= 0$.
- **Game** 1: This game proceeds as **Game** 0, except that **Game** 1 aborts if $\mathcal{A}$ sends a message-signature pair (m, s_m) correct according to algorithm $\mathsf{VerifySig}(pk_i, m, s_m)$, where pk_i is the public key of an honest buyer and such that $\mathcal{A}$ did not obtain before a signature of message m for public key pk_i. The probability that **Game** 1 aborts is bounded by the following lemma:

  Lemma 1. *Under the existential unforgeability of the signature scheme* (Keygen, Sign, VerifySig)*,* $|Pr\ [\textbf{\textit{Game}}\ 1] - Pr\ [\textbf{\textit{Game}}\ 0]| = \nu_1$.

  Proof: We construct an algorithm $\mathcal{T}$ that, if there exists an adversary $\mathcal{A}$ that makes **Game** 1 abort with non-negligible probability ϵ, breaks the existential unforgeability of the signature scheme with non-negligible probability ϵ/n, where n denotes the number of honest buyers. Existential unforgeability is formally defined in [21] as a game between a challenger $\mathcal{C}$ and an adversary. First, $\mathcal{C}$ gives to the adversary a public key pk and access to a signing oracle $\mathsf{O}(sk, \cdot)$. Eventually, $\mathcal{A}$ outputs a message signature pair (m, s_m), and wins the game if $\mathsf{VerifySig}(pk, m, s_m)$ outputs accept and $\mathsf{O}(sk, \cdot)$ was not queried on input m.
  Algorithm $\mathcal{T}$ operates as follows. First, $\mathcal{T}$ receives pk from $\mathcal{C}$. $\mathcal{T}$ chooses an honest buyer $\mathcal{B}_i$, and, when $\mathcal{A}$ queries the public key of $\mathcal{B}_i$, $\mathcal{T}$ sends pk. To simulate a purchase request from $\mathcal{B}_i$, $\mathcal{T}$ queries $\mathsf{O}(sk, \cdot)$ on input the purchase message m to obtain s_m and sends (m, s_m) to $\mathcal{A}$. Eventually, $\mathcal{A}$ sends a pair $(m', s_{m'})$ such that **Game** 1 aborts. If $s_{m'}$ is not associated with public key pk, $\mathcal{T}$ fails. Otherwise, $\mathcal{T}$ submits $(m', s_{m'})$ to $\mathcal{C}$. ■
- **Game** 2: This game proceeds as **Game** 1, except that the proofs $\pi_1 = \mathsf{PK}\{(sk_{\mathcal{B}}{}') : (pk_{\mathcal{B}}{}', sk_{\mathcal{B}}{}') \leftarrow \mathsf{BKeygen}(1^k) \wedge C \leftarrow \mathsf{JEnc}(pk_{\mathcal{J}}, sk_{\mathcal{B}}{}')\}$ and $\{\pi_{2i} = \mathsf{PK}\{(W_{\mathcal{B}i}) : c_i \leftarrow \mathsf{BEnc}(pk_{\mathcal{B}}{}', W_{\mathcal{B}i}) \wedge W_{\mathcal{B}i} \in \{0,1\}\}\}_{i=1}^{l_2}$ are replaced by simulated proofs. Under the assumption that the proof system is zero-knowledge, |Pr [**Game** 2]−Pr [**Game** 1]| $= \nu_2$.
- **Game** 3: This game proceeds as **Game** 2, except that the ciphertext $C = \mathsf{JEnc}(pk_{\mathcal{J}}, sk_{\mathcal{B}}{}')$ included in the initialization message of buyers is replaced by a ciphertext that encrypts a random message. At this point, the proof of knowledge $\pi_1 = \mathsf{PK}\{(sk_{\mathcal{B}}{}') : (pk_{\mathcal{B}}{}', sk_{\mathcal{B}}{}') \leftarrow \mathsf{BKeygen}(1^k) \wedge C \leftarrow \mathsf{JEnc}(pk_{\mathcal{J}}, sk_{\mathcal{B}}{}')\}$ is a simulated proof of a false statement. The probability that $\mathcal{Z}$ distinguishes between **Game** 3 and **Game** 2 is bounded by the following lemma:

  Lemma 2. *Under the IND-CPA security of the encryption scheme that consists of algorithms* (JKeygen, JEnc,

JDec), $|Pr\ [\textbf{\textit{Game}}\ 3] - Pr\ [\textbf{\textit{Game}}\ 2]| = \nu_3$.

Proof: We construct an algorithm $\mathcal{T}$ that, given an environment $\mathcal{Z}$ that distinguishes **Game** 3 and **Game** 2 with non-negligible probability, breaks the IND-CPA security of the encryption scheme with non-negligible probability. Chosen plaintext security is formally defined through a game between a challenger $\mathcal{C}$ and an adversary [22]. First, $\mathcal{C}$ provides the adversary with a public key pk. The adversary sends two messages m_0 and m_1. $\mathcal{C}$ flips a coin b and sends $C = \mathsf{Enc}(pk, m_b)$ to the adversary. Finally, the adversary sends his guess b' and wins if $|\Pr[b = b'] - \frac{1}{2}|$ is non-negligible.
Let n be the number of honest buyers. We consider a sequence of hybrid games, where, in game-j, ciphertext C is replaced by the encryption of a random message in the initialization messages of buyers $\mathcal{B}_1, \ldots, \mathcal{B}_j$, while the remaining requests remain unchanged. Clearly, game-0 corresponds to **Game** 2 and game-n corresponds to **Game** 3. If $\mathcal{Z}$ distinguishes **Game** 3 and **Game** 2 with non-negligible probability ϵ, there must be an index j such that $\mathcal{Z}$ distinguishes game-j from game-$(j+1)$ with non-negligible probability ϵ/n.
Our algorithm $\mathcal{T}$ operates as follows. First, $\mathcal{T}$ receives the public key pk from $\mathcal{C}$. When $\mathcal{A}$ requests the public key of $\mathcal{J}$, $\mathcal{T}$ sends pk. Initialization messages of buyers $\mathcal{B}_1, \ldots, \mathcal{B}_j$ are computed following algorithm SetupB, except that C is replaced by the encryption of a random value and π_1 and $\{\pi_{2i}\}_{i=1}^{l_2}$ by simulated proofs. For $\mathcal{B}_{j+2}, \ldots, \mathcal{B}_n$, initialization messages are computed following algorithm SetupB. For $\mathcal{B}_{j+1}$, $\mathcal{T}$ computes $\mathsf{BKeygen}(1^\kappa)$ to obtain a key pair $(sk_{\mathcal{B}}{}', pk_{\mathcal{B}}{}')$, picks random m and submits $(sk_{\mathcal{B}}{}', m)$ to $\mathcal{C}$. $\mathcal{C}$ flips a coin b and returns $C = \mathsf{JEnc}(pk, m_b)$, and $\mathcal{T}$ uses C to compute the request. If $b = 0$, the distribution corresponds to game-j, and, if $b = 1$, it corresponds to game-$j+1$. $\mathcal{Z}$ outputs a bit b', which is forwarded by $\mathcal{T}$ to $\mathcal{C}$. ■

- **Game** 4: This game proceeds as **Game** 3, except that **Game** 4 aborts upon receiving an arbitration request $(W_{\mathcal{B}}, m, s_m, \mathcal{B}_i)$ where (m, s_m) was previously sent to $\mathcal{A}$, s_m is associated with the public key of an honest buyer $\mathcal{B}_i$ and $W_{\mathcal{B}}$ was the buyer's watermark sent by $\mathcal{B}_i$ in the initialization phase. The probability that $\mathcal{Z}$ distinguishes between **Game** 4 and **Game** 3 is bounded by the following lemma:

  Lemma 3. *Under the IND-CPA security of the encryption scheme that consists of algorithms* ($\mathsf{BKeygen}, \mathsf{BEnc}, \mathsf{BDec}$)*, $|Pr\ [\textbf{\textit{Game}}\ 4] - Pr\ [\textbf{\textit{Game}}\ 3]| = \nu_4$.*

  Proof: Let n be the number of honest buyers. We construct an algorithm $\mathcal{T}$ that, given an adversary $\mathcal{A}$ that makes **Game** 4 abort with non-negligible probability, breaks the chosen plaintext security of the encryption scheme with non-negligible probability ϵ/n.
  Algorithm $\mathcal{T}$ operates as follows. First, $\mathcal{T}$ receives a public key $pk_{\mathcal{B}}{}'$ from $\mathcal{C}$. To compute the initialization message m of $\mathcal{B}_1$, $\mathcal{T}$ follows **Game** 3 and uses $pk_{\mathcal{B}}{}'$ to encrypt bitwise $W_{\mathcal{B}}$ and obtain $(c_i)_{i=1}^{l_2}$. To encrypt the first $l_2 - 1$ bits, $\mathcal{T}$ encrypts bitwise $W_{\mathcal{B}}$ using $c_i = \mathsf{BEnc}(pk_{\mathcal{B}}{}', m_i)$. To encrypt the last bit, $\mathcal{T}$ sends $(0, 1)$ to $\mathcal{C}$ and receives back a ciphertext c, which is used to complete the bitwise encryption. To compute the initialization message of buyers $\mathcal{B}_2, \ldots, \mathcal{B}_n$, $\mathcal{T}$ follows **Game** 3. Eventually $\mathcal{A}$ sends an arbitration message $(W_{\mathcal{B}}, m, s_m, \mathcal{B}_i)$ that makes **Game** 4 abort. If $\mathcal{B}_i \neq \mathcal{B}_1$, $\mathcal{T}$ fails. Otherwise, if the last bit of $W_{\mathcal{B}}$ is 0, $\mathcal{T}$ sends $b' = 0$ to $\mathcal{C}$, and otherwise $b' = 1$ to $\mathcal{C}$. ■
- **Game** 5: This game proceeds as **Game** 4, except that **Game** 5 performs all the changes described in $\mathcal{E}_{\mathrm{POT}}$ for the case in which sender and a subset of buyers are corrupted. Under the security of the POT scheme, we have that $|\Pr\ [\textbf{Game}\ 5] - \Pr\ [\textbf{Game}\ 4]| = \nu_5$.

$\mathcal{E}$ performs all the changes described in **Game** 5, and forwards and receives messages from $\mathcal{F}_{\mathrm{PCP}}$ as described in our simulation below:

- **Setup.** When $\mathcal{A}$ sends a request $(\text{retrieve}, \mathcal{J})$, $\mathcal{E}$ runs $\mathsf{JKeygen}$ in order to generate a key pair $(sk_{\mathcal{J}}, pk_{\mathcal{J}})$ and sends $(\text{retrieve}, \mathcal{J}, pk_{\mathcal{J}})$ to $\mathcal{A}$. Similarly, when $\mathcal{A}$ sends a request $(\text{retrieve}, \mathcal{B}_i)$, where $\mathcal{B}_i$ is an uncorrupted buyer, $\mathcal{E}$ runs Keygen to generate (sk_i, pk_i) and returns $(\text{retrieve}, \mathcal{J}, pk_i)$ to $\mathcal{A}$. When $\mathcal{A}$ sends $(\text{register}, pk_j)$ to register the public key pk_j of a corrupted buyer $\mathcal{B}_j$, $\mathcal{E}$ stores pk_j.
- **Initialization.** $\mathcal{E}$ follows **Game** 5 to compute the initialization message (m, s_m) of each honest buyer $\mathcal{B}_i$ using $pk_{\mathcal{B}_i}'$ as encryption public key, and runs the subsequent simulated zero-knowledge proofs of knowledge π_1 and $\{\pi_{2i}\}_{i=1}^{l_2}$ with $\mathcal{A}$. Upon receiving $(pk_{\mathcal{S}}, T)$ from $\mathcal{A}$, $\mathcal{E}$ runs $\mathcal{E}_{\mathrm{POT}}$ on input $(pk_{\mathcal{S}}, T)$ and obtains $(\text{init}, ct_1, p_1, \ldots, ct_N, p_N)$. For $j = 1$ to N, $\mathcal{E}$ runs $\mathsf{Dec}(sk_{\mathcal{B}_i}', ct_j)$ to obtain y_{ij}. After obtaining $y_{i1}, \ldots, y_{iN}$ for all buyers $\mathcal{B}_i$, $\mathcal{E}$ sends $(\text{init}, \{\mathcal{B}_i, y_{i1}, \ldots, y_{iN}\}_{i=1}^{n}, p_1, \ldots, p_N)$ to $\mathcal{F}_{\mathrm{PCP}}$. Upon receiving $(\text{deposit}, \mathcal{B}_i, ac_i)$ from $\mathcal{F}_{\mathrm{PCP}}$, $\mathcal{B}_i$ runs $\mathcal{E}_{\mathrm{POT}}$ on input $(\text{deposit}, \mathcal{B}_i, ac_i)$ and forwards messages between $\mathcal{E}_{\mathrm{POT}}$ and $\mathcal{A}$.
- **Purchase.** Upon receiving $(\text{request}, \mathcal{B}_i)$ from $\mathcal{F}_{\mathrm{PCP}}$, $\mathcal{E}$ runs $\mathcal{E}_{\mathrm{POT}}$ on input $(\text{request}, \mathcal{B}_i)$ and forwards messages between $\mathcal{E}_{\mathrm{POT}}$ and $\mathcal{A}$. When $\mathcal{E}_{\mathrm{POT}}$ returns $(\text{response}, b)$, $\mathcal{E}$ sends $(\text{response}, b)$ to $\mathcal{F}_{\mathrm{PCP}}$.
- **Release.** Upon receiving a pirated copy y' from $\mathcal{A}$, $\mathcal{E}$ sends $(\text{release}, y')$ to $\mathcal{F}_{\mathrm{PCP}}$ and stores y' in a table T_{rel} of released copies.
- **Arbitration.** Upon receiving a framing message (F) from $\mathcal{A}$, $\mathcal{E}$ aborts if any of the conditions described up to **Game** 5 is fulfilled. (We showed that $\mathcal{E}$ aborts with negligible probability.) Those conditions prevent $\mathcal{A}$ from framing honest buyers, who by assumption do not release pirated copies. If $\mathcal{E}$ does not abort, $\mathcal{E}$ picks any pirated copy in T_{rel} and sends (detect, y) to $\mathcal{F}_{\mathrm{PCP}}$. Upon receiving $(\text{detresp}, \mathcal{E}, \textit{guilty})$, $\mathcal{E}$ sends $(\textit{guilty})$ to $\mathcal{A}$.

The distribution produced in **Game** 5 is identical to that of our simulation. By summation we have that $|\Pr\ [\textbf{Game}\ 5] \leq \nu_6$. ■

B. Security Analysis When Buyers Are Corrupted

Claim 2. *When only the buyers are corrupted, the ensembles* $\mathrm{IDEAL}_{\mathcal{F}_{\mathrm{PCP}},\mathcal{E},\mathcal{Z}}$ *and* $\mathrm{REAL}_{\mathrm{PCP},\mathcal{A},\mathcal{Z}}$ *are computationally indistinguishable under the security of the* POT *scheme, the unforgeability of the signature scheme* (Keygen, Sign, VerifySig) *and the collusion resistance of the watermarking scheme.*

Proof: We show by means of a series of hybrid games that the environment $\mathcal{Z}$ cannot distinguish between the real execution ensemble $\mathrm{REAL}_{\mathrm{PCP},\mathcal{A},\mathcal{Z}}$ and the simulated ensemble $\mathrm{IDEAL}_{\mathcal{F}_{\mathrm{PCP}},\mathcal{E},\mathcal{Z}}$ with non-negligible probability.

- **Game** 0: This game corresponds to the execution of the real-world protocol with honest $\mathcal{S}$ and $\mathcal{J}$. Therefore, Pr [**Game** 0] = 0.
- **Game** 1 : This game operates as **Game** 0, except that the string $w = \phi||(W_{\mathcal{S}} \oplus W_{\mathcal{B}})$ that is used to compute the watermark embedding is replaced by a random string. Since the strings ϕ and $W_{\mathcal{S}}$ are picked at random by the honest seller, w is a random string that leaks no information on $W_{\mathcal{B}}$. Therefore, |Pr [**Game** 1] − Pr [**Game** 0]| = 0.
- **Game** 2 : This game operates as **Game** 1, except that **Game** 2 aborts if $\mathcal{A}$ releases a watermarked content y whose watermark w does not equal that of any of the watermarked contents previously received by $\mathcal{A}$. The probability that **Game** 2 aborts is bounded by the following lemma:

  Lemma 4. *Under the assumption that the watermarking scheme is collusion resistant, |Pr* [***Game*** 2] − *Pr* [***Game*** 1]| = ν_1.

  Proof: We construct an algorithm $\mathcal{T}$ that, given an adversary $\mathcal{A}$ that makes **Game** 2 abort with non-negligible probability, breaks the collusion resistant property of the watermarking scheme with non-negligible probability. $\mathcal{T}$ interacts with the challenger $\mathcal{C}$ of the collusion resistant game described in Definition 1. First, $\mathcal{T}$ receives the challenge $(y_1, \ldots, y_l)$ from $\mathcal{C}$. $\mathcal{T}$ computes the public key $pk_{\mathcal{J}}$ of $\mathcal{J}$ and sends $pk_{\mathcal{J}}$ to $\mathcal{A}$ upon request. $\mathcal{T}$ employs $(y_1, \ldots, y_l)$ as the watermarked copy of messages m_1. (We assume that l is larger than the number of buyers n.) For other items, the watermarked copies are computed as usual. Eventually, $\mathcal{A}$ releases a pirated copy y' that corresponds to m_1 whose watermark does not equal any of the watermarks embedded in $(y_1, \ldots, y_l)$. $\mathcal{T}$ forwards y' to $\mathcal{C}$. ■
- **Game** 3: This game proceeds as **Game** 2, except that **Game** 3 performs all the changes described in $\mathcal{E}_{\mathrm{POT}}$ for the case in which the buyers are corrupted. Under the security of the POT scheme, we have that |Pr [**Game** 3] − Pr [**Game** 2]| = ν_2.

$\mathcal{E}$ performs all the changes described in **Game** 3, and forwards and receives messages from $\mathcal{F}_{\mathrm{PCP}}$ as described in our simulation below:

- **Setup.** When $\mathcal{A}$ sends a request $(\text{retrieve}, \mathcal{J})$, $\mathcal{E}$ runs JKeygen in order to generate a key pair $(sk_{\mathcal{J}}, pk_{\mathcal{J}})$ and sends $(\text{retrieve}, \mathcal{J}, pk_{\mathcal{J}})$ to $\mathcal{A}$. When $\mathcal{A}$ sends $(\text{register}, pk_j)$ to register the public key pk_j of a corrupted buyer $\mathcal{B}_j$, $\mathcal{E}$ stores pk_j.
- **Initialization.** $\mathcal{E}$ receives the message $(\text{init}, p_1, \ldots, p_N)$ and stores it. Upon receiving (m, s_m) from $\mathcal{A}$, where s_m is associated with the public key of buyer $\mathcal{B}_i$, $\mathcal{E}$ checks whether VerifySig outputs accept. As verifier, $\mathcal{E}$ executes the proofs π_1 and, for $i = 1$ to l_2, π_{2i}, and ignores the request if any of them fails. Otherwise $\mathcal{E}$ stores the tuple $(\mathcal{B}_i, m, s_m)$. $\mathcal{E}$ runs $\mathcal{E}_{\mathrm{POT}}$ on input $(\text{init}, p_1, \ldots, p_N)$ and forwards messages between $\mathcal{E}_{\mathrm{POT}}$ and $\mathcal{A}$. When $\mathcal{E}_{\mathrm{POT}}$ returns $(\text{deposit}, ac_i)$, $\mathcal{E}$ sends $(\text{deposit}, ac_i)$ to $\mathcal{F}_{\mathrm{PCP}}$.
- **Purchase.** Upon receiving a request Q from $\mathcal{A}$ on behalf of $\mathcal{B}_i$, $\mathcal{E}$ runs $\mathcal{E}_{\mathrm{POT}}$ on input Q. $\mathcal{E}_{\mathrm{POT}}$ returns $(\text{request}, \tau)$, and $\mathcal{E}$ sends $(\text{request}, \tau)$ to $\mathcal{F}_{\mathrm{PCP}}$. $\mathcal{F}_{\mathrm{PCP}}$ returns $(\text{response}, y_{i\tau})$. $\mathcal{E}$ picks the tuple $(\mathcal{B}_i, m, s_m)$, parses m as $(pk_{\mathcal{B}}{}', (c_i)_{i=1}^{l_2}, C)$ and encrypts $y_{i\tau}$ by running $ct_{i\tau} = \mathsf{Enc}(pk_{\mathcal{B}}{}', y_{i\tau})$. $\mathcal{E}$ runs $\mathcal{E}_{\mathrm{POT}}$ on input $(\text{response}, ct_{i\tau})$ and forwards messages between $\mathcal{E}_{\mathrm{POT}}$ and $\mathcal{A}$.
- **Release.** Upon receiving a pirated copy y' from $\mathcal{A}$, $\mathcal{E}$ sends $(\text{release}, y')$ to $\mathcal{F}_{\mathrm{PCP}}$.

The distribution produced in **Game** 3 is identical to that of our simulation. By summation we have that |Pr [**Game** 3] ≤ ν_3. ■

VI. Fair Privacy-Preserving BSW Protocol

Recently, a transformation that takes as input a secure POT scheme and turns it into an optimistic fair POT scheme has been proposed [29]. This transformation requires a neutral third party, an adjudicator, who is only involved in case of dispute between a seller and a buyer (hence the protocol is called optimistic). Other fair e-commerce protocols that do not protect privacy also require the involvement of a third party [30].

The transformation is based on the use of verifiably encrypted signatures (VES) [31]. Roughly speaking, a VES is a signature encrypted under the public key of the adjudicator that can be publicly verified, i.e., the verifier can check that the ciphertext contains a valid signature without the secret key of the adjudicator.

The transformation works as follows. The buyer computes a VES ω on her purchase request Q, and sends (Q, ω) to the seller. Upon receiving a correct response from seller, the buyer reveals a valid signature on her request. This signature can be used by the seller to prove that the buyer accepted the result and that a payment was done. If a malicious buyer does not reveal the signature, the adjudicator, upon being requested by the seller, can verify that the seller fulfilled his delivery obligations and, in that case, extract a valid signature from the VES ω. Similarly, if a malicious seller does not fulfill his delivery obligations, the adjudicator, upon being requested by the buyer, can tell the seller to fulfill them and, in the end, send the seller a valid signature. We refer to [29] for a detailed description. One of appealing properties of this transformation is that it adds very few overhead in terms of communication and computation.

Our PBSW protocol can also be extended to achieve fairness by applying this transformation to the POT scheme used as building block. In our protocol, the role of the adjudicator

can be played by the judge. Both judge and buyers have to compute a key pair as defined in the VES scheme used and register the public key at the registration authority.

VII. Efficiency

The efficiency of our construction depends on the efficiency of the building blocks used to instantiate it. Efficiency measurements for the asymmetric watermark embedding technique we employ (algorithms SetupB, SetupV, Detect, Check) can be found in [9], which describes and implements an instantiation based on the homomorphic public key encryption scheme due to Paillier [23]. In [9], images of size 512 x 512 pixels are employed as digital content offered by $\mathcal{S}$, whose size after embedding the watermark in the encrypted domain is $536,870,912$ bits when each DCT coefficient is encrypted, or $6,318,080$ bits when composite signal representation is used. In the following, we employ watermarked messages $(m_1, \ldots, m_N)$ of those sizes as input to the POT scheme.

To evaluate the performance of the whole PBSW protocol, we implement the POT scheme proposed in [29] in a workstation equipped with an IntelCore2Duo processor at 3 GHz and 4 Gbyte of RAM. All the functionalities are implemented in the C programming language. We use the PBC library[2] for elliptic curve and pairing operations. We select type A pairings constructed on the curve $y^2 = x^3 + x$ over the field F_q for a 512-bit prime $q = 3 \bmod 4$. For other cryptographic primitives we employ the OpenSSL library[3]. Specifically, we employ RIPEMD-160 [32] as hash function and AES [33] in counter mode as block cipher.

The efficiency of the POT scheme in [29] in terms of computation and communication depends on the selection of three parameters: the number of messages N offered by $\mathcal{S}$, the size of the watermarked messages, and the values u and l that define the maximum deposit allowed $D_{max} = u^l$.

The performance of the initialization phase (algorithms POTInitB and POTInitS) depends on the number N of messages and on the message size. Table I shows performance measurements when N is 100, 1000 and 10000, and when the message size is $536,870,912$ and $6,318,080$ bits. As can be seen, the computation and communication complexity of both algorithms grows linearly with N, because POTInitS computes N signatures and encrypts N messages, and POTInitB verifies N signatures. The overhead of the initial deposit sent from $\mathcal{B}$ to $\mathcal{S}$ is negligible.

The efficiency of the purchase phase does not depend on N, while the message size only affects algorithm POTComplete. However, it depends on the maximum deposit allowed $D_{max} = u^l$, because this parameter determines the efficiency of a range proof that is computed by $\mathcal{B}$ to show that her account is non-negative. Table II shows the measurements of the purchase phase for a message size of $6,318,080$ bits and N = 100. D_{max} is 100, 1000 and 10000 respectively, with fixed value u = 10, and l = 2, 3, 4 respectively. The value l influences the computation and communication cost during POTReq() and POTVerReq().[4]

As can be seen, our protocol consists of an expensive initialization phase, whose cost grows linearly with the amount of messages and the message size, and very cheap purchase phases. In contrast, anonymous buyer-seller watermarking protocols consist of a cheap initialization phase and expensive transfer phases. Therefore, our protocol is more convenient in resource constrained settings because the initialization phase needs to be run only once, and later on a lot of very cheap purchase phases can be carried out.

An alternative to our PBSW protocol, which also preserves buyer's privacy by hiding from vendors which items are bought, would employ private information retrieval (PIR) [34]. Basically, the idea would be as follows. In the initialization phase, when the seller $\mathcal{S}$ computes $(ct_1, \ldots, ct_N)$, for $j = 1$ to N, $\mathcal{S}$ picks randomly a key K_j for a secure symmetric key encryption scheme, and encrypts $R_j = \mathsf{Enc}(K_j, ct_j)$. Keys $(K_1, \ldots, K_N)$ are given as input messages to the POT protocol, and $(R_1, \ldots, R_N)$ are given as input messages of the PIR scheme. To purchase a message, a buyer $\mathcal{B}$ obtains the symmetric key K via the POT scheme, then obtains R via the PIR scheme, and finally decrypts R with K to obtains ct. After that, this alternative would follow the PBSW protocol.

Since the size of the keys K is smaller than the encrypted watermarked messages ct, the communication complexity of the initialization phase of the POT scheme decreases. However, after obtaining K, $\mathcal{B}$ has to perform a PIR query to obtain R. Most efficient computational PIR schemes achieve log-squared communication complexity in the database size [35], [36]. Therefore, if the number of purchases is expected to be small, this alternative would be more appealing. However, when the number of purchases is big, our PBSW protocol is more efficient since the communication complexity of purchase phases is constant. The exact threshold for the number of purchases in which our PBSW is more efficient depends on the concrete POT and PIR schemes used to instantiate both constructions.

VIII. Conclusion

We have proposed a privacy-preserving buyer-seller watermarking protocol, i.e., a protocol that allows copyright protection and in which buyers purchase from sellers without the seller learning the items they buy. Furthermore, we have also described how to extend the protocol to provide both buyers and sellers with optimistic fair exchange.

Further research needs to be conducted on the integration of privacy-preserving buyer-seller watermarking protocols with e-commerce and digital right management applications, as well as on the compliance of such applications with existing legislation. Another interesting goal is the design of lightweight privacy-preserving buyer-seller watermarking protocols suitable for resource-constrained devices.

[2]http://crypto.stanford.edu/pbc/

[3]http://www.openssl.org/

[4]The value u affects the efficiency of the initialization phase, but its influence is negligible when the message size is big. The most efficient choice in our case would be to fix $l = 1$.

TABLE I
IMPACT OF NUMBER OF MESSAGES AND MESSAGE SIZE ON THE PERFORMANCE OF THE POT INITIALIZATION PHASE

Algorithm	Message size (bytes)	N = 100 messages Time (seconds)	N = 100 messages Communication (Kbytes)	N = 1000 messages Time (seconds)	N = 1000 messages Communication (Kbytes)	N = 10000 messages Time (seconds)	N = 10000 messages Communication (Kbytes)
POTInitS()	789,760	1.52	77,139	14.73	771,377	147.92	7,713,752
POTInitB()		0.37	0.015	3.31	0.015	32.94	0.015
POTInitS()	67,108,864	91.89	6,553,614	923.73	65,536,127	9,209.55	655,361,252
POTInitB()		0.37	0.015	3.31	0.015	32.94	0.015

TABLE II
IMPACT OF MAXIMUM DEPOSIT ON THE PERFORMANCE OF THE POT PURCHASE PHASE

Algorithm	D_{max} = 100 Time (ms)	D_{max} = 100 Communication (bytes)	D_{max} = 1000 Time (ms)	D_{max} = 1000 Communication (bytes)	D_{max} = 10000 Time (ms)	D_{max} = 10000 Communication (bytes)
POTReq()	40	1,776	48	2,072	56	2,368
POTVerReq()	43	–	50	–	58	–
POTResp()	7.5	532	7.5	532	7.5	532
POTVerResp()	3.2	–	3.2	–	3.2	–
POTComplete()	11.7	–	11.7	–	11.7	–
Total Purchase	105.4	2,308	120.4	2,604	136.4	3,170

REFERENCES

[1] J. Tsai, S. Egelman, L. Cranor, and R. Acquisti, "The effect of online privacy information on purchasing behavior: An experimental study, working paper," June 2007.

[2] "Enforcing privacy promises: Section 5 of the ftc act," Federal Trade Commission Act, http://www.ftc.gov/privacy/privacyinitiatives/promises.html.

[3] D. Boneh and J. Shaw, "Collusion-secure fingerprinting for digital data (extended abstract)," in *CRYPTO*, ser. Lecture Notes in Computer Science, D. Coppersmith, Ed., vol. 963. Springer, 1995, pp. 452–465.

[4] B. Pittzmann and M. Schunter, "Asymmetric fingerprinting," in *Adv. in Cryptology - EUROCRYPT'96*, ser. LNCS 1070, 1996, pp. 84–95.

[5] B. Pfitzmann and M. Waidner, "Anonymous fingerprinting," in *EUROCRYPT*, 1997, pp. 88–102.

[6] N. D. Memon and P. W. Wong, "A buyer-seller watermarking protocol," *IEEE Transactions on Image Processing*, vol. 10, no. 4, pp. 643–649, 2001.

[7] H.-S. Ju, H.-J. Kim, D.-H. Lee, and J.-I. Lim, "An anonymous buyer-seller watermarking protocol with anonymity control," *Information Security and Cryptology*, pp. 421–432, Nov. 2002.

[8] C.-L. Lei, P.-L. Yu, P.-L. Tsai, and M.-H. Chan, "An efficient and anonymous buyer-seller watermarking protocol," *IEEE Trans. on Image Processing*, vol. 13, no. 12, pp. 1618–1626, 2004.

[9] M. Deng, T. Bianchi, A. Piva, and B. Preneel, "An efficient buyer-seller watermarking protocol based on composite signal representation," in *Proceedings of the 11th ACM workshop on Multimedia and security*. Princeton, New Jersey, USA: ACM New York, NY, USA, 2009, pp. 9–18.

[10] A. Rial, M. Kohlweiss, and B. Preneel, "Universally composable adaptive priced oblivious transfer," in *Pairing*, ser. Lecture Notes in Computer Science, H. Shacham and B. Waters, Eds., vol. 5671. Springer, 2009, pp. 231–247.

[11] D. Chaum, "Blind signatures for untraceable payments," in *CRYPTO*, 1982, pp. 199–203.

[12] R. Dingledine, N. Mathewson, and P. F. Syverson, "Tor: The second-generation onion router," in *USENIX Security Symposium*. USENIX, 2004, pp. 303–320.

[13] O. Berthold, H. Federrath, and M. Köhntopp, "Project "anonymity and unobservability in the internet"," in *CFP '00: Proceedings of the tenth conference on Computers, freedom and privacy*. New York, NY, USA: ACM, 2000, pp. 57–65.

[14] M. Gruteser and B. Hoh, "On the anonymity of periodic location samples," in *SPC*, ser. Lecture Notes in Computer Science, D. Hutter and M. Ullmann, Eds., vol. 3450. Springer, 2005, pp. 179–192.

[15] S. Kremer, *Formal Analysis of Optimistic Fair Exchange Protocols*. Université Libre de Bruxelles - Faculté des Sciences: Ph.D. Thesis, 2004.

[16] R. Canetti, "Universally composable security: A new paradigm for cryptographic protocols," in *FOCS*, 2001, pp. 136–145.

[17] W. Aiello, Y. Ishai, and O. Reingold, "Priced oblivious transfer: How to sell digital goods," in *EUROCRYPT*, ser. Lecture Notes in Computer Science, B. Pfitzmann, Ed., vol. 2045. Springer, 2001, pp. 119–135.

[18] A. Rial, M. Deng, T. Bianchi, A. Piva, and B. Preneel, "A provably secure anonymous buyer-seller watermarking protocol," *IEEE Trans. on Information Forensics and Security*.

[19] M. Barni and F. Bartolini, *Watermarking Systems Engineering: Enabling Digital Assets Security and Other Applications*, 1st ed. CRC Press, Boca Raton, February 2004.

[20] G. Tardos, "Optimal probabilistic fingerprint codes," *J. ACM*, vol. 55, no. 2, 2008.

[21] S. Goldwasser, S. Micali, and R. L. Rivest, "A digital signature scheme secure against adaptive chosen-message attacks," *SIAM J. Comput.*, vol. 17, no. 2, pp. 281–308, 1988.

[22] S. Goldwasser and S. Micali, "Probabilistic encryption," *J. Comput. Syst. Sci.*, vol. 28, no. 2, pp. 270–299, 1984.

[23] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *EUROCRYPT*, 1999, pp. 223–238.

[24] I. Damgård and M. Jurik, "A generalisation, a simplification and some applications of paillier's probabilistic public-key system," in *Public Key Cryptography*, ser. Lecture Notes in Computer Science, K. Kim, Ed., vol. 1992. Springer, 2001, pp. 119–136.

[25] M. Bellare and O. Goldreich, "On defining proofs of knowledge," in *CRYPTO '92*, E. F. Brickell, Ed., vol. 740. Springer-Verlag, 1992, pp. 390–420.

[26] J. Camenisch and M. Stadler, "Proof systems for general statements about discrete logarithms," Institute for Theoretical Computer Science, ETH Zürich, Tech. Rep. TR 260, Mar. 1997.

[27] J. Camenisch and V. Shoup, "Practical verifiable encryption and decryption of discrete logarithms," in *CRYPTO*, ser. Lecture Notes in Computer Science, D. Boneh, Ed., vol. 2729. Springer, 2003, pp. 126–144.

[28] G. Poupard and J. Stern, "Fair encryption of rsa keys," in *EUROCRYPT*, 2000, pp. 172–189.

[29] A. Rial and B. Preneel, "Optimistic fair priced oblivious transfer," in *AFRICACRYPT*, ser. Lecture Notes in Computer Science, D. J. Bernstein and T. Lange, Eds., vol. 6055. Springer, 2010, pp. 131–147.

[30] N. Asokan, V. Shoup, and M. Waidner, "Optimistic fair exchange of digital signatures (extended abstract)," in *EUROCRYPT*, 1998, pp. 591–606.

[31] D. Boneh, C. Gentry, B. Lynn, and H. Shacham, "Aggregate and verifiably encrypted signatures from bilinear maps," in *EUROCRYPT*, ser. Lecture Notes in Computer Science, E. Biham, Ed., vol. 2656. Springer, 2003, pp. 416–432.

[32] H. Dobbertin, A. Bosselaers, and B. Preneel, "RIPEMD-160: A strengthened version of RIPEMD," in *FSE*, 1996.

[33] NIST, *Advanced Encryption Standard (AES) (FIPS PUB 197)*, National Institute of Standards and Technology, 2001.

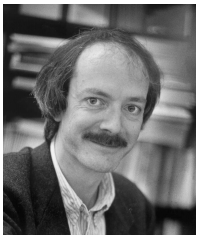
[34] E. Kushilevitz and R. Ostrovsky, "Replication is not needed: Single database, computationally-private information retrieval," in *FOCS*, 1997, pp. 364–373.

[35] H. Lipmaa, "An oblivious transfer protocol with log-squared communication," in *ISC*, ser. Lecture Notes in Computer Science, J. Zhou, J. Lopez, R. H. Deng, and F. Bao, Eds., vol. 3650. Springer, 2005, pp. 314–328.

[36] C. Gentry and Z. Ramzan, "Single-database private information retrieval with constant communication rate," in *ICALP*, ser. Lecture Notes in Computer Science, L. Caires, G. F. Italiano, L. Monteiro, C. Palamidessi, and M. Yung, Eds., vol. 3580. Springer, 2005, pp. 803–815.

Alfredo Rial received his master's degree in Telecommunication Engineering from the Universidade de Vigo, Spain, in 2008. Currently he is a Ph.D. candidate at Katholieke Universiteit Leuven, Belgium, under the supervision of Prof. Bart Preneel. He is funded by the Research Foundation - Flanders (FWO). His research interests include public key cryptography, cryptographic protocols design and privacy.

Josep Balasch received his master's degree in Telecommunication Engineering from the Universitat Politecnica de Catalunya, Barcelona, Spain, in 2008. Currently he is pursuing a Ph.D. degree at the Katholieke Universiteit Leuven, Belgium. His research interests include efficient implementations of cryptographic algorithms, embedded security and privacy-preserving systems.



Bart Preneel (Member, IEEE) received the M.S. degree in electrical engineering and the Ph.D. degree in applied sciences (cryptology) from the Katholieke Universiteit Leuven, Belgium, in 1987 and 1993, respectively. He is currently Full Professor with the Katholieke Universiteit Leuven. He was Visiting Professor at five universities in Europe and was a Research Fellow with the University of California at Berkeley. He has authored and coauthored more than 300 reviewed scientific publications and is the inventor of three patents. His main research interests are cryptography and information security. Prof. Preneel is President of the International Association for Cryptologic Research (IACR) and of the Leuven Security Excellence Consortium (L-SEC vzw.), an association of 60 companies and research institutions in the area of e-security.