Applications for e-ID Cards in Flanders

ADAPID Deliverable D13

Storage

A. Rial (Ed.), B. Van Alsenoy

November 2009
Executive Summary

This report, which presents the research conducted within ADAPID in the scope of the Storage Work Package, describes a privacy-preserving e-commerce application based on the priced oblivious transfer primitive. In a nutshell, buyers, after being authenticated via their e-ID cards, can purchase digital goods from vendors without disclosing the items being bought, while vendors still receive the right amount of money from buyers.

First, we motivate our problem and give an overview of the existing solutions that aim at providing privacy-preserving e-commerce. After that, we focus on the concept of priced oblivious transfer and compare our solution with existing schemes.

We then define security for POT and describe our priced oblivious transfer scheme. We prove that our scheme offers better security guarantees than previous ones, as well as several extra features. Furthermore, we show that the use of our scheme as a key building block in the deployment of a practical e-commerce application is feasible via the implementation of a demonstrator that can be efficiently run on standard personal computers.

Finally, we address the legal aspects concerning priced oblivious transfer. We analyze issues related with taxation and liability of vendors and buyers.
List of Contributions

Introduction  COSIC
Related Work  COSIC
Cryptographic Construction  COSIC
Implementation of the Demonstrator  COSIC
Legal aspects  COSIC, ICRI
Conclusions  COSIC, ICRI

Editor  Alfredo Rial (COSIC)
List of ADAPID Publications covered in this report


Contents

1 Introduction ........................................ 11
   1.1 Motivation ......................................... 11
   1.2 Privacy-preserving e-Commerce .................... 12
   1.3 Our Contribution ..................................... 13

2 Related Work ...................................... 15
   2.1 Overview on Security Models ......................... 15
   2.2 Oblivious Transfer .................................. 16
   2.3 Priced Oblivious Transfer ........................... 16

3 Cryptographic Construction ......................... 21
   3.1 Definitions ......................................... 21
   3.2 Technical Preliminaries ............................... 22
      3.2.1 Assumptions .................................... 22
      3.2.2 Commitment schemes ............................. 23
      3.2.3 Proofs of Knowledge ...................... 24
      3.2.4 Signature Schemes ............................. 25
      3.2.5 Public Key Encryption .......................... 26
   3.3 Intuition Behind Our Construction .................. 26
   3.4 Construction ......................................... 27
   3.5 Security Proof ....................................... 29
   3.6 Features and Extensions .............................. 34

4 Implementation of the Demonstrator ................. 37
   4.1 Tools .............................................. 37
   4.2 Architecture of the Demonstrator .................... 39
      4.2.1 Organization of the Source Code .............. 39
      4.2.2 Installation and Execution .................... 41
   4.3 The Graphical User Interface ....................... 42
      4.3.1 Initialization .................................. 42
      4.3.2 Transfer ....................................... 42
      4.3.3 Other functionalities ........................... 45
Chapter 1

Introduction

1.1 Motivation

Nowadays, more and more data is being created, processed and stored digitally, and thus secure data storage is indispensable for the development of the information society. Traditionally, computer security has focused on the design of data storage mechanisms that provide both confidentiality and integrity in order to ensure that data is neither accessed nor modified by unauthorized parties, as well as on the protection of the communications needed for remote access to the data storage. Another desirable property is message authentication, which provides assurance on the identity of the generator of data.

Furthermore, the progress of information technologies has provided efficient tools to collect and share data. When using electronic communication services, users reveal information that can be utilized to describe and individual. Among this information not only personal data like name, address or phone number should be included, but also every kind of data, e.g., the search terms used when browsing the web, that can be employed to characterize an individual.

However, commonly used designs usually overlook the preservation of the privacy of users that search and retrieve data, which is a major concern in applications such as e-health, e-commerce or location-based services. Namely, when users look for information in a database or web server, the data holder learns both their search terms and the records they download, which can reveal sensitive information such as their medical condition, their location, their political membership or their religion. This data disclosure leads to a situation where personal profiles can easily be created, and further, a privacy disrespectful data holder may consider to obtain benefits by selling them to unauthorized third parties.

From a legal point of view, privacy of personal data is of great impor-
CHAPTER 1. INTRODUCTION

tance [eu-95]. This applies not only to data useful to identify an individual, but also to information that describes her thoughts or her beliefs. The revelation of private data, besides being undesirable from users’ perspective, can also pose inconveniences and extra investments to service providers. Law enforcement requires sensitive personal data to be stored and processed in environments that provide additional security guarantees [org99], and failures that cause data loss receive intense media scrutiny and damage service providers’ reputation [ukd08, ukd07]. Consequently, it is of interest for service providers to deploy system architectures where the amount of location information that users need to disclose is minimized.

From a pragmatic perspective, transaction security and privacy concerns are among the main reasons that discourage the use of applications such as e-commerce. Although sometimes it is argued that users who claim to be worried about their privacy do not consistently take actions to protect it, recent research [TECA07] demonstrates that, when they are confronted to a prominent display of private information, they not only prefer vendors that offer better privacy protection but also are willing to pay higher prices to purchase from more privacy-friendly web sites. Consequently, secure data storage mechanisms that provide privacy-preserving protocols for remotely searching and retrieving data are fundamental.

1.2 Privacy-preserving e-Commerce

So far, the solutions proposed to develop privacy-enhancing e-commerce of digital goods can roughly be divided into two categories: those that hide the identity of the buyer from the vendor (anonymous purchase), and those that hide which goods are bought (oblivious purchase). Anonymous purchase [GA04, LOL04] usually employs anonymous e-cash [Cha82, CHL05, BCKL09] to construct systems where buyers can withdraw coins from a bank and spend them without revealing their identity. These systems have several shortcomings. First, they hinder customer management (e.g. the vendor cannot easily apply marketing techniques like giving discounts to regular buyers). Second, they do not allow for other methods of payment. Finally, strong anonymity is difficult to achieve and there exist several attacks to reduce it [BFK00].

Oblivious purchase is thus more appealing in scenarios where full anonymity cannot be obtained or when the disadvantages that anonymity causes are important. Oblivious purchase permits effective customer management and allows for every method of payment. Like for anonymous purchase [GA04, LOL04], it has also been shown how to integrate it into existing Digital Rights Management systems [SWH]. One can argue that, since the vendor does not know which items are sold, he can find it difficult to discover which products are more demanded. However, we note that this infor-
1.3. OUR CONTRIBUTION

Oblivious purchase employs the Priced Oblivious Transfer (POT) primitive \cite{AIR01}, which is a generalization of the well-known Oblivious Transfer (OT) \cite{Rab81} primitive intended to permit private purchases. OT is a two-party protocol between a sender $S$ and a receiver $R$, where $S$ offers a set of messages $m_1, \ldots, m_N$ to $R$. $R$ chooses selection values $\sigma_1, \ldots, \sigma_k \in \{1, \ldots, N\}$ and interacts with $S$ in such a way that $R$ learns $m_{\sigma_1}, \ldots, m_{\sigma_k}$ and nothing about the other messages, and $S$ does not learn anything about $\sigma_1, \ldots, \sigma_k$.

POT is a two-party protocol between a vendor $V$ and a buyer $B$, where $V$ sells a set of messages $m_1, \ldots, m_N$ with prices $p_1, \ldots, p_N$ to $B$. Besides the requirements that $V$ must not learn $\sigma_1, \ldots, \sigma_k$ and $B$ must not learn anything about the other messages, in POT $B$ must pay prices $p_{\sigma_1}, \ldots, p_{\sigma_k}$ without $V$ learning anything about the amount of money paid.

Both OT and POT admit an adaptive variant \cite{NP99b} ($OT_{k \times 1}^N, POT_{k \times 1}^N$) where, in transfer phase $i$, $R$ or $B$ may choose $\sigma_i$ after receiving $m_{\sigma_{i-1}}$. The adaptive variant is more suitable for constructing an oblivious database, enabling applications of OT such as medical record storage or location-based services \cite{NP99b, KFF07}, and the deployment of privacy-preserving e-commerce.

1.3 Our Contribution

We present a POT scheme that provides better security guarantees than existing ones. Namely, previous schemes can only be proven secure when executed in isolation, while ours provides security under sequential execution, i.e., when the protocol is run sequentially a number of times between a vendor and a buyer. (We also designed a scheme secure under concurrent execution in \cite{RKP09}, but that scheme is less efficient.) Furthermore, our scheme provides extra features, such as the possibility of charging different prices to different buyers for the same item, which allows the vendor to apply marketing techniques like making discounts to regular buyers.

We also implement a demonstrator of a privacy-preserving e-commerce application. The Belgian e-ID card is employed by buyers to authenticate themselves towards the vendor, and later on buyers interact with vendors according to our POT scheme in order to purchase items. We give a detailed description of the functionalities and of the structure of our demonstrator, as well as efficiency measurements of the most resource intensive procedures.

Finally, we address the analysis of the legal aspects concerning the deploying of priced oblivious transfer in an e-commerce application. So far, there has been no research on this field. Interestingly, POT poses some additional issues over traditional e-commerce. For instance, since the ven-
dor does not know which items are sold, how can he pay sale’s taxes? Our analysis focuses on taxation and liability.

**Outline of the report.** In Chapter 2 we describe previous priced oblivious transfer schemes and we compare them with our scheme. We depict our cryptographic scheme and prove its security in Chapter 3. In Chapter 4 we present our demonstrator. We address the legal aspects concerning the use of priced oblivious transfer as a building block included in an e-commerce application in Chapter ?? and we conclude in Chapter 6.
Chapter 2

Related Work

2.1 Overview on Security Models

The notion of security in OT and POT has been evolving as time goes by, from an honest-but-curious model to a half-simulation model, then to a full-simulation model and finally to universal composability.

In the honest-but-curious model all the parties behave honestly. Security guarantees that after running the protocol any curious party cannot learn anything else by analyzing the transcript.

In the half-simulation model [NP05] security holds separately for senders and receivers. Security for receivers implies that the sender’s view of the protocol when the receiver chooses $\sigma$ is indistinguishable from his view when she chooses $\sigma'$. However, security for senders involves a stronger notion following the ideal-world/real-world paradigm, by means of which it is possible to describe an ideal-world counterpart for every real-world malicious receiver such that the adversary in the real world gains no more information as counterpart in an ideal world in which OT is implemented by using a trusted third party.

In the full-simulation model [Can01], a protocol $\psi$ is secure if there exists no environment $Z$ that can distinguish whether it is interacting with adversary $A$ and parties running protocol $\psi$ or with the ideal process for carrying out the desired task, where ideal adversary $E$ and dummy parties interact with an ideal functionality $F_\psi$. More formally, we say that protocol $\psi$ emulates the ideal process when, for all adversaries $A$, there exists a simulator $E$ such that for all environments $Z$, the ensembles $\text{IDEAL}_{F_\psi,E,Z}$ and $\text{REAL}_{\psi,A,Z}$ are computationally indistinguishable. The universally composable security model can be defined similarly. The difference is that in the UC-model $Z$ and $A$ can communicate at any time during the execution of the protocol. Protocols that are proven UC-secure maintain their security even when they are run concurrently with an unbounded number of arbitrary
Chapter 2. Related Work

Protocol instances controlled by an adversary. However, the full-simulation model only guarantees security under sequential composition.

2.2 Oblivious Transfer

Introduced by Rabin [Rab81] in 1981, the concept of Oblivious Transfer was extended by Even, Goldreich and Lempel [EGL82], who defined $OT^2_1$, and afterwards by Brassard, Crépeau and Robert ([BCR86a], [BCR86b]), who proposed $OT^N_1$ and gave a construction using $N$ applications of a $OT^2_1$ protocol. In 1990 Bellare and Micali [BM89] showed practical implementations of $OT^2_1$ under the honest-but-curious model.


This notion was showed to admit practical attacks against receiver’s security [NP99b]. Therefore, Camenisch et al. [CNS07], as well as subsequent works [GH07], presented efficient adaptive OT schemes in the full-simulation model. However, these works are not UC-secure because they use black-box simulation with adversarial rewinding in their security proofs.

Recently, an adaptive UC-secure OT scheme was proposed [GH08]. They utilize the approach of assisted decryption used in [CNS07] [GH07], where $S$ sends to $R$ a collection of ciphertexts and in each transfer phase helps $R$ to decrypt one of them. As pointed out in [GH08], this approach allows for transfer phases with constant computational and communication complexity, and it is suitable to ensure that $S$ does not change the messages in each transfer phase, which are important properties for constructing an oblivious database. This is in contrast to the approach used in other non-adaptive UC-secure OT schemes [DNO08, Wag08], where, in each transfer phase, $R$ hands a set of keys to $S$, who sends back a collection of ciphertexts such that $R$ is able to decrypt only one of them.

2.3 Priced Oblivious Transfer

Despite this recent progress in OT, so far there are no efficient POT schemes whose security is proven within the UC security paradigm. The first POT scheme [AIR01], as well as subsequent works [Tom03], analyze security in the half-simulation model. In [DNO08] it is explained why these protocols fail even under sequential composition and a practical attack is shown.

The existing conditional oblivious transfer schemes [COR99, BK04], in which sender with input $x$ and receiver with input $y$ interact in such a
way that a transfer is completed only when \( q(x, y) = 1 \) for some public predicate \( q(\cdot, \cdot) \), are non-adaptive and employ the half-simulation model. On the other hand, security of both the non-adaptive [IK97, SSR08] and the adaptive [Her08] Generalized Oblivious Transfer schemes proposed so far, which can be instantiated as non-adaptive and adaptive POT schemes respectively, depends on the underlying OT scheme utilized to implement them, but we note that these solutions are rather inefficient. Finally, access control schemes for OT based on stateful anonymous credentials [CGH08] are not UC-secure either.

From now on we will show the main characteristics of the POT scheme proposed by [AIR01], in order to compare it with our scheme. In [AIR01] they show a basic scheme and after that they propose important efficiency improvements. Here we will only talk about their best solution, which can only be applied to sell keys.

In this solution they use three main tools: homomorphic encryption, a method for conditional disclosure of secrets described in [GIKM98] and a technique for symmetrically private information retrieval described in [NP99a] and in [NP99b]. It is assumed that each item has a different price (if it is not the case, this can be achieved by adequately scaling the prices and the deposit of the buyer).

In the initialization phase, the buyer sends the vendor an initial deposit and a public key of the homomorphic encryption scheme. Then the buyer conducts a proof of knowledge to ensure that she knows the corresponding secret key and that the public key is well-formed\(^1\). Finally, the vendor computes an encryption of the deposit under this public key.

In the initialization phase, the buyer sends the vendor an initial deposit and a public key of the homomorphic encryption scheme. Then the buyer conducts a proof of knowledge to ensure that she knows the corresponding secret key and that the public key is well-formed\(^1\). Finally, the vendor computes an encryption of the deposit under this public key.

Each transfer phase involves only two rounds of communication. In the \( i \)th transfer phase, the buyer sends a message composed of three elements: a separate encryption of the bits that form the current value of the account \( \{E_k(\text{account}_{i-1}), \ldots, E_k(\text{account}_0)\} \), a separate encryption of the bits that form the price \( \{E_k(p_{i-1}), \ldots, E_k(p_0)\} \) and an encryption \( E_k(u^i) \). The last encrypted value is used by the vendor to ensure that the buyer has behaved honestly in former transactions.

Upon receiving the message, the vendor creates an encryption of the price \( E_k(p) = \sum_j E_k(p_j)2^j \) and updates the value of the account \( E_k(\text{account}^i) = E_k(\text{account}^{i-1}) \oplus E_k(p) \), where \( \oplus \) is an operator that produces the ciphertext \( E_k(\text{account}^{i-1} - p) \). Then he discloses the values \( v^i \) and \( u^i \) under the condition \( (E_k(\text{account}^{i-1}) = \sum_j E_k(\text{account}_j)2^j) \land (0 \leq p \leq \text{account}^{i-1}) \land (u = u^{i-1}) \). We refer to [AIR01] in order to learn how this conditional disclosure

\(^1\)In order to avoid this proof the authors propose to use the ElGamal encryption scheme because then it is possible to perform a check to ensure the validity of the public key, although they note that in this case the vendor does not know if the buyer has the corresponding secret key.
is performed\footnote{For example, a conditional disclosure of the value $x$ under the condition $u = u_i^{-1}$ involves picking a random $\alpha$ and computing $E_k(\alpha(u - u_i^{-1}) + x)$. If the buyer knows the secret key and the condition is fulfilled, she can get the value $x$.} but we note that the condition $0 \leq p \leq \text{account}^{i-1}$ involves a monotone formula of size $O(l)$ which uses the separate encryption of bits sent by the buyer. The value $v^i$ is used to mask the item.

In addition, the vendor uses the symmetrically private information retrieval technique to disclose the item. Let $G_a$ be a group with a generator $g$ where the DDH assumption holds. The vendor has a sequence of pseudo-random items $(M^{p_0}, \ldots, M^{p_{n-1}})$, where each $M^{p_j} = g^{\prod_{j=0}^{l-1} s_j^i}$ and \{(s^0_0, s^0_1), \ldots, (s^l_{l-1}, s^l_{l-1})\} are keys. In order to let the buyer get the item, the vendor discloses $s_j^i r_j$ of each pair of keys, where $p_j$ is the $j$th bit of the price and $r_j$ is a random value. He also sends the value $g^{1/(r_0 \ldots r_{l-1})}$.

Finally, the buyer uses her secret key to obtain the values $s_j^i r_j$, $u^i$ and $v^i$. She uses the values $s_j^i r_j$ to raise $g^{1/(r_0 \ldots r_{l-1})}$ and obtain $M^{p_j}$. The final item is $M^{p_j} + v^i$. After that, the buyer can get the data encrypted under this key by using a PIR query. For further explanations we refer to \cite{AIR01}.

The fact that this scheme is only useful to sell keys is not an important limitation, because the more common application for Oblivious Transfer is also the provision of keys.

As can be seen, this scheme requires only two rounds of communication in each transfer phase. This is optimal since even without providing privacy the buyer needs to specify the item and the vendor has to send it. Both vendor and buyer have to send $O(l)$ encryptions. On the other hand, the buyer needs to perform $O(l)$ public key operations and at most $nl$ secret key operations, which can be a problem if the number of items is big\footnote{According to the authors, this can be improved by arranging the prices in a TRIE structure.}.

Our scheme follows a different approach, which consists in constructing a POT scheme by combining an oblivious payment (OP) scheme with any committed oblivious transfer (COT) scheme. The OP scheme will ensure that the buyer updates her account properly and that she pays the right price for the item she wants to obtain. A COT scheme is an OT scheme where the sender receives as input a commitment to the selection value.

In terms of amount of interaction we do not achieve the optimal solution, and in terms of computation and communication complexity the efficiency will depend mainly on the COT scheme selected. Apart from that, we note that the scheme due to \cite{AIR01} is secure in the half-simulation model, whereas the security of our scheme will depend on the COT scheme chosen and can thus be in the full-simulation model.

Another difference is that in their scheme the vendor does not know if the buyer misbehaves, i.e., the vendor cannot know whether she sends a price that does not fulfill $0 \leq p \leq \text{account}$, but if she misbehaves once then she
cannot get data from the vendor anymore. In our solution the vendor knows when a buyer misbehaves, but the buyer can still get data from the vendor in future transfers if after that she behaves properly. This is contrived for practical reasons, e.g., avoiding denial of service attacks.

In addition, we let different items have the same price. We also extend our scheme in order to allow the vendor to charge different prices for the same item depending on the buyer.
Chapter 3

Cryptographic Construction

We define formally priced oblivious transfer (POT) and security for POT in Section 3.1. After that, we introduce some cryptographic primitives used as building blocks in our scheme and the security assumptions under which our scheme is secure in Section 3.2. Later we give some intuition on how our scheme works in Section 3.3 and we depict it in detail in Section 3.4. Finally, we prove our scheme secure in Section 3.5 and we describe its features and possible extensions in Section 3.6.

3.1 Definitions

Adaptive k-out-of-N priced oblivious transfer \((POT^{N}_{k \times 1})\). It is a two-party protocol between a vendor \(V\) and a buyer \(B\). In the initialization phase, \(V\) receives messages \((m_1, \ldots, m_N)\) with prices \((p_1, \ldots, p_N)\) as input. \(B\) receives an initial deposit \(ac_0\) as input. \(B\) stores state information \(B_0\) and \(V\) stores state information \(V_0\) and outputs \(ac_0\). After that, \(V\) and \(B\) engage in up to \(k\) transfer phases. In the \(i\)th transfer, \(V\) gets state information \(V_{i-1}\) as input, and \(B\) gets state information \(B_{i-1}\) and selection value \(\sigma_i \in \{1, \ldots, N\}\). If \(ac_0 - \sum_{j \in S} p_{\sigma_j} \geq 0\), where \(S\) contains the indices of all transfers that ended successfully, then \(V\) stores state information \(V_i\) and \(B\) stores state information \(B_i\) and outputs \(m_{\sigma_i}\). Otherwise \(V\) stores \(V_i = V_{i-1}\) and \(B\) stores \(B_i = B_{i-1}\).

Ideal functionality \(\mathcal{F}_{POT}\). We define and prove security of our construction in the full-simulation model under static corruptions. Our construction operates in the plain model, where parties utilize authenticated channels. We describe an ideal functionality for POT \(\mathcal{F}_{POT}\) that extends the ideal functionality for OT given in [GH08].
$\mathcal{F}_{POT}$. Parameterized with integers $(N, l)$, a maximum price $p_{\text{max}}$, and a deposit upper bound $A$, and running with a vendor $V$ and a buyer $B$, $\mathcal{F}_{POT}$ works as follows:

- On input a message $(\text{sid}, \text{vendor}, m_1, \ldots, m_N, p_1, \ldots, p_N)$ from $V$, where each $m_i \in \{0, 1\}^l$ and each $p_i \in [0, p_{\text{max}}]$, it stores $(m_1, \ldots, m_N)$ and $(p_1, \ldots, p_N)$ and sends $(\text{sid}, p_1, \ldots, p_N)$ to $B$ and to the adversary.

- On input a message $(\text{sid}, \text{buyerdep}, \text{deposit})$, where $\text{deposit} \in [0, A]$, if a $(\text{sid}, \text{vendor}, \ldots)$ message was not received before, then it does nothing. Otherwise, it stores deposit and sends $(\text{sid}, \text{deposit})$ to $V$.

- On input a message $(\text{sid}, \text{buyerreq}, \sigma)$ from $B$, where $\sigma \in \{1, \ldots, N\}$, if messages $(\text{sid}, \text{vendor}, m_1, \ldots, m_N, p_1, \ldots, p_N)$ and $(\text{sid}, \text{buyerdep}, \text{deposit})$ were not received before or $\text{deposit} - p_\sigma < 0$, then it does nothing. Otherwise, it sends $(\text{sid}, \text{request})$ to $V$ and receives $(\text{sid}, b)$ in response. It hands $(\text{sid}, b)$ to the adversary. If $b = 0$, it sends $(\text{sid}, \bot)$ to $B$. If $b = 1$, it updates $\text{deposit} = \text{deposit} - p_\sigma$ and sends $(\text{sid}, m_\sigma)$ to $B$.

### 3.2 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 $G$ and $G_T$ be groups of prime order $p$. A map $e : G \times G \to G_T$ must satisfy the following properties:

- **Bilinearity.** A map $e : G \times G \to G_T$ is bilinear if $e(a^x, b^y) = e(a, b)^{xy}$;

- **Non-degeneracy.** For all generators $g \in G$, $e(g, g)$ generates $G_T$;

- **Efficiency.** There exists an efficient algorithm that outputs the pairing group setup $(p, G, G_T, e, g)$ and an efficient algorithm to compute $e(a, b)$ for any $a, b \in G$.

### 3.2.1 Assumptions

The security of our scheme relies on the Strong Diffie-Hellman (SDH) assumption [BB04], the Power Decisional Diffie-Hellman (PDDH) assumption [CNS07], the decision linear (DLIN) assumption [BBS04], and the strong RSA assumption [RSA78]:

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

- On input a message $(\text{sid}, \text{vendor}, m_1, \ldots, m_N, p_1, \ldots, p_N)$ from $V$, where each $m_i \in \{0, 1\}^l$ and each $p_i \in [0, p_{\text{max}}]$, it stores $(m_1, \ldots, m_N)$ and $(p_1, \ldots, p_N)$ and sends $(\text{sid}, p_1, \ldots, p_N)$ to $B$ and to the adversary.

- On input a message $(\text{sid}, \text{buyerdep}, \text{deposit})$, where $\text{deposit} \in [0, A]$, if a $(\text{sid}, \text{vendor}, \ldots)$ message was not received before, then it does nothing. Otherwise, it stores deposit and sends $(\text{sid}, \text{deposit})$ to $V$.

- On input a message $(\text{sid}, \text{buyerreq}, \sigma)$ from $B$, where $\sigma \in \{1, \ldots, N\}$, if messages $(\text{sid}, \text{vendor}, m_1, \ldots, m_N, p_1, \ldots, p_N)$ and $(\text{sid}, \text{buyerdep}, \text{deposit})$ were not received before or $\text{deposit} - p_\sigma < 0$, then it does nothing. Otherwise, it sends $(\text{sid}, \text{request})$ to $V$ and receives $(\text{sid}, b)$ in response. It hands $(\text{sid}, b)$ to the adversary. If $b = 0$, it sends $(\text{sid}, \bot)$ to $B$. If $b = 1$, it updates $\text{deposit} = \text{deposit} - p_\sigma$ and sends $(\text{sid}, m_\sigma)$ to $B$.

### 3.2 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 $G$ and $G_T$ be groups of prime order $p$. A map $e : G \times G \to G_T$ must satisfy the following properties:

- **Bilinearity.** A map $e : G \times G \to G_T$ is bilinear if $e(a^x, b^y) = e(a, b)^{xy}$;

- **Non-degeneracy.** For all generators $g \in G$, $e(g, g)$ generates $G_T$;

- **Efficiency.** There exists an efficient algorithm that outputs the pairing group setup $(p, G, G_T, e, g)$ and an efficient algorithm to compute $e(a, b)$ for any $a, b \in G$.

### 3.2.1 Assumptions

The security of our scheme relies on the Strong Diffie-Hellman (SDH) assumption [BB04], the Power Decisional Diffie-Hellman (PDDH) assumption [CNS07], the decision linear (DLIN) assumption [BBS04], and the strong RSA assumption [RSA78]:
3.2. TECHNICAL PRELIMINARIES

Definition 1 (l-SDH) Given \((g, g^x, \ldots, g^{x^l}) \in \mathbb{G}_l^{l+1}\), where \(x \in \mathbb{Z}_q\), output a pair \((c, g^{1/(x+c)})\), where \(c \in \mathbb{Z}_q\). Formally, the l-SDH assumption holds if there exists a negligible function \(\nu\) such that:

\[
\Pr[(q, \mathbb{G}, \mathbb{G}_T, g, e) \leftarrow \text{BilinearSetup}(1^k); x \leftarrow \mathbb{Z}_q; (z_1, z_2) \leftarrow A(g, g^x, \ldots, g^{x^l}) : (z_1, z_2) = (c, g^{1/(x+c)}) \land c \in \mathbb{Z}_q] < \nu(k).
\]

Definition 2 (l-PDDH) Given \((g, g^x, \ldots, g^{x^l}, H) \in \mathbb{G}_l^{l+1} \times \mathbb{G}_T\), where \(x \in \mathbb{Z}_q\), distinguish between the vector \((H^x, H^{x^2}, \ldots, H^{x^l}) \in \mathbb{G}_T^l\) and a random vector \(T \in \mathbb{G}_T^l\). Formally, the l-PDDH assumption holds if there exists a negligible function \(\nu\) such that:

\[
\Pr[(q, \mathbb{G}, \mathbb{G}_T, g, e) \leftarrow \text{BilinearSetup}(1^k); x \leftarrow \mathbb{Z}_q; H \leftarrow \mathbb{G}_T; T_0 \leftarrow (H^x, H^{x^2}, \ldots, H^{x^l}); T_1 \leftarrow \mathbb{G}_T^l; b \leftarrow \{0, 1\}; b' \leftarrow A(g, g^x, \ldots, g^{x^l}, H, T_b) : b = b'] < 1/2 + \nu(k).
\]

Definition 3 (DLIN) On input \((g, g^a, g^b, g^{ac}, g^{bd}, z) \in \mathbb{G}_6^3\) for random exponents \(a, b, c, d \in \mathbb{Z}_p\), the DLIN assumption holds if it is computationally hard to decide whether \(z = g^{c+d}\).

Definition 4 (Strong RSA) Given \(v \in \mathbb{Z}_n^*\), the flexible RSA problem consists in finding \(u \in \mathbb{Z}_n^*\) and \(2 < x < n\), where \(x\) is coprime to \(\phi(n)\), such that \(u^x = v \mod n\). Formally, the strong RSA assumption holds if there exists a negligible function \(\nu\) such that:

\[
\Pr[(n, \mathbb{Z}_n^*) \leftarrow \text{Setup}(1^k); v \leftarrow \mathbb{Z}_n^*; (z_1, z_2) \leftarrow A(v) : z_1^2 = v \mod n \land z_2 \in \{3, \ldots, n-1\} \land \gcd(z_2, \phi(n)) = 1] < \nu(k).
\]

3.2.2 Commitment schemes

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

We present here an instance of a class of commitment schemes due to
Damgård and Fujisaki [DF02], which is a correction and generalization of
the scheme proposed by Fujisaki and Okamoto [FO97]. It allows committing
to arbitrary size integers and is based on commutative groups that fulfill certain
properties, being the most important that computing the order of the group
is hard for the committer.

Damgård and Fujisaki stated the conditions that any group should fulfill
in order to be used in this scheme; they also said that the cyclic group of
quadratic residues modulo a special RSA modulus $n$, i.e., $QR_n$, can be used.
We use this group, which implies that the scheme is secure under the strong
RSA assumption. The construction works as follows:

**DGComSetup**($1^n$). This algorithm can be run by the verifier or by a trusted
third party. It outputs the parameters $params_{Com} = (n, g, h)$ of the
scheme, where $n$ is a special RSA modulus and $g$ and $h$ are generators
of $QR_n$ and $g = h^a$. The verifier provides a proof of knowledge that
$PK\{(\alpha) : g = h^\alpha\}$.

**DGCommit**($params_{Com}, x, open_x$). On input the parameters of the scheme,
an integer $x$ and randomness $open_x \in [0..\lfloor \frac{n}{4} \rfloor]$, it computes the com-
mitment $C = g^x h^{open_x} \mod n$. Here $\lfloor \frac{n}{4} \rfloor$ is a good upper bound for
the order of the group, as required in [DF02].

**DGOpen**($params_{Com}, C, x, open$). On input the commitment $C$ and values
$(x', open = (open_x, b))$, it outputs accept if $C = g^x h^{open_x} b$, where
$b^2 = 1$, and reject otherwise. An honest prover can always use
$b = 1$.

This commitment scheme is statistically hiding. It is computationally
binding if the strong RSA assumption holds.

### 3.2.3 Proofs of Knowledge

We use several existing results to prove statements about discrete loga-
rithms: (1) proof of knowledge of a discrete logarithm modulo a prime
[Sch91]; (2) proof of knowledge of the equality of some element in different
representations [CP93]; (3) proof of knowledge that a value lies in a given
interval $[A, B]$ [Lip03]; and (4) proof of the disjunction or conjunction of
any two of the previous [CDS94]. 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 [Dam99, Dam02].

When referring to the proofs above, we follow the notation introduced
by Camenisch and Stadler [CS97] for various proofs of knowledge of discrete
logarithms and proofs of the validity of statements about discrete logarithms.
3.2. TECHNICAL PRELIMINARIES

\[ PK\{(\alpha, \beta, \delta) : y = g^\alpha h^\beta \land \tilde{y} = \tilde{g}^\alpha \tilde{h}^\delta \land A \leq \alpha \leq B\} \] denotes a "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 \(G = \langle g \rangle\) and \(\tilde{G} = \langle \tilde{g} \rangle\) 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. There exists a knowledge extractor which can extract these quantities from a successful prover.

3.2.4 Signature Schemes

A signature scheme consists of the algorithms Keygen, Sign and VerifySig. Keygen outputs a secret key \(sk\) and a public key \(pk\). \(\text{Sign}(sk, m)\) outputs a signature \(s\) of message \(m\). \(\text{VerifySig}(pk, m, s)\) outputs accept if \(s\) is a valid signature of \(m\) and reject otherwise. (This definition can be extended to support multi-block messages \(\vec{m} = \{m_1, \ldots, m_n\}\).) A signature scheme must be correct and unforgeable \([GMR88]\). Informally speaking, correctness implies that the \(\text{VerifySig}\) algorithm always accepts an honestly generated signature. 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 employ the signature scheme in \([BB04]\).

\(\text{Keygen}(1^k)\) computes a bilinear map setup \((p, G, G_T, e, g)\), picks a secret key \(sk = x \leftarrow \mathbb{Z}_p\) and computes a public key \(pk = (g, y = g^x)\).

\(\text{Sign}(pk, sk, m)\) computes \(s = g^{1/(x+m)}\).

\(\text{VerifySig}(pk, m, s)\) outputs accept when \(e(s, y \cdot g^m) = e(g, g)\). Otherwise, it outputs reject.

We also use a multi-block signature scheme proposed in \([CDN09]\), which extends the one described above.

\(\text{MBKeygen}(1^k)\) computes a bilinear map setup \((p, G, G_T, e, g)\), picks a secret key \(sk = (x_m, x_1, \ldots, x_l) \leftarrow \mathbb{Z}_p\) and computes a public key \(pk = (g, y_m = g^{x_m}, y_1 = g^{x_1}, \ldots, y_l = g^{x_l})\).

\(\text{MBSign}(pk, sk, (m, c_1, \ldots, c_l))\) computes \(s = g^{1/(x_m+m+x_1c_1+\ldots+x_lc_l)}\).

\(\text{MBVerifySig}(pk, m, s)\) outputs accept when \(e(s, y_m \cdot g^m \cdot y_1^{c_1} \cdot \ldots \cdot y_l^{c_l}) = e(g, g)\). Otherwise, it outputs reject.
3.2.5 Public Key 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 $sk$. Semantic security guarantees that an adversary does not get any knowledge about $m$ from $c$.

We employ an encryption scheme secure under the decision linear assumption.

\[\text{DLKeygen}(1^k)\text{ computes a bilinear map setup } (p, G, G_T, e, g), \text{ chooses a generator } h, \text{ picks a secret key } sk = (x_1, x_2) \in \mathbb{Z}_p, \text{ and computes } (g_1, g_2) = (h^{1/x_1}, h^{1/x_2}). \text{ The public key is } pk = (g_1, g_2, h).\]

\[\text{DLEnc}(pk, m)\text{ pick random } (r, s) \in \mathbb{Z}_p \text{ and computes } c = (c_1, c_2, c_3) = (g_1^r, g_2^s, m \cdot h^{r+s}).\]

\[\text{DLDec}(sk, c)\text{ outputs } m = c_3/(c_1 c_2^x).\]

3.3 Intuition Behind Our Construction

The main idea behind our priced oblivious transfer (POT) scheme is to combine a committed oblivious transfer (COT) scheme and a so called oblivious payment (OP) scheme. A COT scheme is basically an OT scheme where the sender receives a commitment to the selection value of the message of interest for the receiver, such that the receiver has to prove that she requests the message whose selection value was used to compute the commitment.

We design a COT scheme by extending the adaptive OT scheme in [CNS07] and thus we also employ the assisted decryption approach.

An OP scheme must ensure that the buyer can pay to the vendor in such a way that the vendor learns nothing about the amount of money that she has paid and nothing about the item she wants to obtain. At the same time, the vendor is guaranteed that the buyer pays the right price for the corresponding item. Our OP scheme follows the approach in [AIR01] of building a prepaid mechanism where $B$ makes an initial deposit to $V$. $B$ also commits to the deposit. In each transfer phase, the buyer commits to the selection value of the message of her interest and to its corresponding price, and proves that they form a correct message-price pair and that $\text{deposit} - p_r \geq 0$. For the latter, the range proof proposed in [Lip03] is used. Then the vendor can use the commitment to the price to update the commitment to the deposit (by using the homomorphic property of the commitment scheme) and the commitment to the selection value as input for the COT scheme. The scheme described in Section 3.4 does not use a commitment to the selection value since there the price identifies the item. We show how
to construct a scheme that allows several items to have the same price in Section 3.6.

The following description of the scheme does not differentiate which operations belong to the COT scheme and which ones to the OP scheme. However, the implementation of the demonstrator depicted in Chapter 4 employs different functions for each scheme in order to allow for modularity.

3.4 Construction

We begin with a high level description of the priced oblivious transfer scheme. The vendor $V$ and the buyer $B$ interact in the initialization phase and in several transfer phases. Details on the algorithms can be found below. We recall that the scheme is parameterized with integers $(N, l)$ for the number of messages and their length, an upper bound $p_{\text{max}}$ for the prices and an upper bound $A$ for the deposit.

Initialization phase. On input the message $(\text{sid}, \text{vendor}, m_1, \ldots, m_N, p_1, \ldots, p_N)$ for the vendor and $(\text{sid}, \text{buyerdep}, ac_0)$ for the buyer (that fulfill the restrictions imposed by the parameters of the scheme):

1. $V$ runs $\text{POTInitVendor}(1^k, m_1, \ldots, m_N, p_1, \ldots, p_N, A)$ in order to obtain a database commitment $T$ and a secret key $sk$, and sends $(\text{sid}, T)$ to $B$.
2. $V$, as prover, and $B$, as verifier, run the interactive zero-knowledge proof of knowledge $\Pi_1$.
3. $B$ computes $(P, D_0^{(\text{priv})}) \leftarrow \text{POTInitBuyer}(\text{crs}, T, ac_0)$. $B$ aborts if the output is $\text{reject}$. Otherwise, $B$ sends $(\text{sid}, P)$ to $V$. ($B$ also needs to pay an amount of $ac_0$ to $V$ through an arbitrary payment channel.)
4. (Upon receiving the money) $V$ runs $(D_0, ac_0) \leftarrow \text{POTGetDeposit}(P, sk, A)$ and checks that $ac_0$ corresponds to the amount of money received.

$V$ stores state information $V_0 = (T, sk, D_0)$ and outputs $(\text{sid}, ac_0)$, and $B$ stores state information $B_0 = (T, D_0^{(\text{priv})})$.

Transfer phase. In the $i$th transfer, $V$ with state information $V_{i-1}$ and input $(\text{sid}, \text{vendor}, b)$ and $B$ with state information $B_{i-1}$ and input $(\text{sid}, \text{buyerreq}, \sigma_i)$ interact as follows:

1. $B$ runs $\text{POTRequest}(T, D_{i-1}^{(\text{priv})}, \sigma_i)$ to get a request $Q$ and private state $(Q^{(\text{priv})}, D^p_{i-1}^{(\text{priv})})$. $B$ sends $(\text{sid}, Q)$ to $V$ and stores $(\text{sid}, Q^{(\text{priv})}, D^p_{i}^{(\text{priv})})$. 
2. \( \mathcal{V} \) obtains \((sid, Q)\). Then \( \mathcal{B} \), as prover, and \( \mathcal{V} \), as verifier, run the interactive zero-knowledge proof of knowledge \( \Pi_2 \).

3. If \( b = 0 \), \( \mathcal{V} \) sends \((sid, \bot)\) to \( \mathcal{B} \). Otherwise \( \mathcal{V} \) runs \( \text{POTRespond}(T, sk, Q) \) to obtain a response \( R \) and state \( D_i \). \( \mathcal{V} \) sends \((sid, R)\) to \( \mathcal{B} \).

4. \( \mathcal{V} \), as prover, and \( \mathcal{B} \), as verifier, run the interactive zero-knowledge proof of knowledge \( \Pi_3 \).

5. \( \mathcal{B} \) receives \((sid, R)\) and runs \( \text{POTComplete}(\sigma_i, T, R, Q^{(\text{priv})}) \) to obtain \( m_{\sigma_i} \).

\( \mathcal{V} \) stores state information \( V_i = (T, sk, D_i) \), and \( \mathcal{B} \) stores state information \( B_i = (T, D_i^{(\text{priv})}) \) and outputs \((sid, m_{\sigma_i})\).

\begin{align*}
\text{POTInitVendor}(1^k, m_1, \ldots, m_N, p_1, \ldots, p_N, A). & \text{ On input the messages } (m_1, \ldots, m_N) \text{ with prices } (p_1, \ldots, p_N): \\
& \text{1. Compute a bilinear map setup } (p, \mathcal{G}, \mathcal{G}_T, e, g). \text{ Pick a random generator } h \in \mathcal{G} \text{ and compute } H = e(g, h). \\
& \text{2. Run } \text{DGComSetup}(1^k) \text{ to obtain } \text{params}_{\text{Comm}} = (n, g, h) \text{ and also } \text{DLKeygen}(1^k) \text{ to obtain } (pk_{\text{enc}}, sk_{\text{enc}}). \\
& \text{3. Run } \text{Keygen}(1^k) \text{ to obtain a public key } pk \text{ and a secret key } sk \text{ for } \\
& \text{the signature scheme.} \\
& \text{4. For } i = 1 \text{ to } N, \text{ compute } A_i = \text{Sign}(pk, sk, p_i) \text{ and } B_i = e(h, A_i) \cdot m_i. \text{ (As mentioned in } \text{CNS07}, \text{ alternatively one can compute } \\
& B_i = H(e(h, A_i)) \oplus m_i, \text{ where } H \text{ extracts a random pad from } B_i. \text{) Set } C_i = (A_i, B_i, p_i). \\
& \text{5. Set } sk = (sk_{\text{enc}}, h) \text{ and } T = (\text{params}_{\text{Comm}}, pk_{\text{enc}}, g, H, pk, C_1, \ldots, C_N). \text{ Output } (T, sk). \\
\end{align*}

\( \Pi_1 \). It is an interactive zero-knowledge proof of knowledge described by
\[ PK\{ (h, \alpha) : H = e(g, h) \land g = h^\alpha \} \]

\begin{align*}
\text{POTInitBuyer}(\text{crs}, T, ac_0). & \text{ On input a database commitment } T \text{ and a deposit } ac_0 \in [0, A): \\
& \text{1. Parse } T \text{ as } (\text{params}_{\text{Comm}}, g, H, pk, C_1, \ldots, C_N). \\
& \text{2. For } i = 1 \text{ to } N, \text{ parse } C_i \text{ as } (A_i, B_i, p_i) \text{ and check whether } \\
& \text{VerifySig}(pk, p_i, A_i) \text{ outputs } \text{accept}. \\
& \text{3. Run } \text{DLEnc}(pk_{\text{enc}}, ac_0) \text{ to obtain an encryption } c \text{ and set } P = c \text{ and } \\
& D_0^{(\text{priv})} = (ac_0, \text{open}_{ac_0} = 0). \text{ Output } (P, D_0^{(\text{priv})}). \\
\text{POTGetDeposit}(P, sk, A). & \text{ It works as follows:} \\
& \text{1. Parse } P \text{ as } c. \\
\end{align*}
2. Compute $\text{DLDec}(sk_{enc}, c)$ to obtain $ac_0$ and check that $ac_0 \in [0, A]$.

3. Compute $D_0 = \text{DGCommit}(\text{params}_{Com}, ac_0, 0)$. Output $(D_0, ac_0)$.

**POTRequest**$(T, D_{i-1}^{(priv)}, \sigma)$. On input a database commitment $T$ and a selection value $\sigma \in \{1, \ldots, N\}$, it works as follows:

1. Parse $T$ as $(\text{params}_{\text{Com}}, g, H, pk, C_1, \ldots, C_N)$ and $C_\sigma$ as $(A_\sigma, B_\sigma, p_\sigma)$.
2. Pick random $(\text{open}_p, \text{open}_{ac_i})$, calculate $ac_i = ac_{i-1} - p_\sigma$ and run $D_p = \text{DGCommit}(\text{params}_{\text{Com}}, p_\sigma, \text{open}_p)$ and $D_i = \text{DGCommit}(\text{params}_{\text{Com}}, ac_i, \text{open}_{ac_i})$.
3. Pick random $v \in \mathbb{Z}_p$ and compute $V = A_\sigma^v$.
4. Output $Q = (V, D_p, D_i)$, $Q^{(priv)} = (v, \text{open}_p)$ and $D_i^{(priv)} = (ac_i, \text{open}_{ac_i})$. Output $(Q, Q^{(priv)}, D_i^{(priv)})$.

$\Pi_2$. It is a zero-knowledge proof of knowledge given by $PK\{(p_\sigma, \text{open}_p, ac_i, \text{open}_{ac_i}, ac_{i-1}, \text{open}_{ac_{i-1}}, v, \alpha) : e(V, g) = e(V, g)^{−p_\sigma} e(g, g)^v \land D_p = \text{DGCommit}(\text{params}_{\text{Com}}, p_\sigma, \text{open}_p) \land D_i = \text{DGCommit}(\text{params}_{\text{Com}}, ac_i, \text{open}_{ac_i}) \land D_{i-1}/D_i D_p = h^\alpha \land ac_{i-1} \geq 0\}$.

**POTRespond**$(T, sk, Q)$. On input a database commitment $T$, a secret key $sk$ and a request $Q$, it works as follows:

1. Parse $Q$ as $(V, D_p, D_i)$.
2. Compute $R = e(h, V)$. Output $R$ and $D_i$.

$\Pi_3$. It is an interactive zero-knowledge proof of knowledge described by $PK\{(h) : H = e(g, h) \land R = e(V, h)\}$.

**POTComplete**$(\sigma, T, R, Q^{(priv)})$. On input a database commitment $T$, a response $R$ and private state $Q^{(priv)}$:

1. Parse $T$ as $(\text{params}_{\text{Com}}, g, H, pk, C_1, \ldots, C_N)$, $C_\sigma$ as $(A_\sigma, B_\sigma, p_\sigma)$ and $Q^{(priv)} = (v, \text{open}_p)$.
2. Compute $m_\sigma = B_\sigma/(R^{1/v})$. Output $m_\sigma$.

### 3.5 Security Proof

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

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 $\mathcal{F}_{\text{POT}}$ and environment $\mathcal{Z}$ in such a way that ensembles $\text{IDEAL}_{\mathcal{F}_{\text{POT}}, \mathcal{E}, \mathcal{Z}}$ and $\text{REAL}_{\mathcal{POT}, \mathcal{A}, \mathcal{Z}}$ are computationally indistinguishable.
Simulation of buyer’s security. In this case only the vendor \( V \) is corrupted.

1. Upon receiving message \((sid, T)\) from \( A \), \( E \) stores \((sid, T)\).
2. When running \( \Pi_1 \) with \( A \), \( E \) uses the knowledge extractor to obtain witness \( h \). \( E \) parses \( T \) as \((\text{params}_{\text{Com}}, \text{pk}_{\text{enc}}, g, \text{pk}, C_1, \ldots, C_N)\) and, for \( i = 1 \) to \( N \), parses \( C_i \) as \((A_i, B_i, p_i)\), runs \( \text{VerifySig}(\text{pk}, A_i) \) and computes \( m_i = B_i/e(h, A_i) \). \( E \) sends \((sid, \text{vendor}, m_1, \ldots, m_N, p_1, \ldots, p_N)\) to \( \mathcal{FPOT} \).
3. Upon receiving \((sid, \text{buyerdep}, \text{deposit})\) from \( \mathcal{FPOT} \), \( E \) runs \( P = \text{DLEnc}(\text{pk}_{\text{enc}}, \text{deposit}) \), stores \( D_{0}^{(\text{priv})} = (\text{deposit}, 0) \) and sends \((sid, P)\) to \( A \).
4. Upon receiving \((sid, \text{request})\) from \( \mathcal{FPOT} \), \( E \) selects \( \sigma \) that corresponds to the item with the lowest price \( p_{\sigma} \) and runs \( \text{POTRequest}(T, D_{i-1}^{(\text{priv})}, \sigma) \) to obtain \( Q, Q^{(\text{priv})} \) and \( D_{i}^{(\text{priv})} \). \( E \) sends \((sid, Q)\) to \( A \).
5. \( E \) runs \( \Pi_2 \) with \( A \).
6. Upon receiving \((sid, R)\) from \( A \), \( E \) stores \((sid, R)\).
7. \( E \) runs \( \Pi_3 \) with \( A \). If the proof is correct, \( E \) sends \((sid, 1)\) to \( \mathcal{FPOT} \). Otherwise \( E \) sends \((sid, 0)\) to \( \mathcal{FPOT} \).

**Theorem 2** When only the vendor \( V \) is corrupted, the distribution ensembles \( \text{IDEAL}_{\mathcal{FPOT}, E, Z} \) and \( \text{REAL}_{\text{POT}, A, Z} \) are unconditionally indistinguishable.

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

**Game 0:** This game corresponds to the execution of the real-world protocol with an honest \( B \). Therefore, \( \Pr[\text{Game } 0] = 0 \).

**Game 1:** This game proceeds as \( \text{Game } 0 \), except that we use the knowledge extractor to obtain \( h \) from \( \Pi_1 \). If extraction fails, \( E \) aborts. The probability that extraction fails is given by the knowledge error of \( \Pi_1 \), which is negligible. Therefore \( |\Pr[\text{Game } 1] - \Pr[\text{Game } 0]| = \nu_1 \).

**Game 2:** This game proceeds as \( \text{Game } 1 \), except that buyer’s deposit message \((sid, P)\), where \( P = \text{DLEnc}(\text{pk}_{\text{enc}}, ac_0) \), is replaced by another valid encryption of the same value under the same public key. Therefore \( |\Pr[\text{Game } 2] - \Pr[\text{Game } 1]| = 0 \).
3.5. SECURITY PROOF

**Game 3:** This game proceeds as **Game 2**, except that buyer’s requests are replaced by other valid requests that use a fixed \( \sigma \), which corresponds to the lowest price. Therefore \( Q = (V, D_p, D_i) \) is computed as follows: \( V = A'_v \) for a random \( v \); \( D_p = \text{DGCommit}(\text{params}_\text{Com}, p_p, \text{open}_p) \); and \( D_i = \text{DGCommit}(\text{params}_\text{Com}, ac_i, \text{open}_ac_i) \) for \( ac_i = ac_{i-1} - p_g \). We note that \( V \) completely hides \( A_i \) and that \( D_p \) and \( D_i \) completely hide \( p \) and \( ac_i \) because Damgård-Fujisaki commitments are information theoretically hiding. Moreover, the proof of knowledge \( \Pi_2 \) is perfect zero-knowledge and therefore does not leak any information about the witness. Therefore, \( |\Pr[\text{Game 3}] - \Pr[\text{Game 2}]| = 0 \).

\( \mathcal{E} \) performs all the changes described in **Game 3**, and forwards and receives messages from \( \mathcal{F}_{\text{POT}} \) as described in our simulation. The distribution produced in **Game 3** is identical to that of our simulation. Therefore, by summation we have that \( |\Pr[\text{Game 3}] \leq \nu_2(\kappa) \).

**Simulation of vendor’s security.** In this case only the buyer \( B \) is corrupted.

1. Upon receiving \((\text{sid}, p_1, \ldots, p_N)\) from \( \mathcal{F}_{\text{POT}} \), for \( i = 1 \) to \( N \), \( \mathcal{E} \) chooses random messages \( m'_i \rightarrow \mathcal{G}_T \) and then runs \( \text{POTInitVendor}(1^k, m'_1, \ldots, m'_N, p_1, \ldots, p_N, A) \) to obtain \( T \) and a secret key \( sk \). \( \mathcal{E} \) sends \((\text{sid}, T)\) to \( A \).

2. \( \mathcal{E} \) runs \( \Pi_1 \) with \( A \).

3. Upon receiving \((\text{sid}, P)\) from \( A \), \( \mathcal{E} \) runs \( \text{POTGetDeposit}(P, sk, A) \) to obtain \((D_0, ac_0)\) and sends \((\text{sid}, \text{buyerdep}, ac_0)\) to \( \mathcal{F}_{\text{POT}} \).

4. Upon receiving \((\text{sid}, Q)\), \( \mathcal{E} \) stores \((\text{sid}, Q)\).

5. When running \( \Pi_2 \) with \( A \), \( \mathcal{E} \) uses the knowledge extractor to obtain witness \((p_\sigma, \text{open}_p, ac_i, \text{open}_ac_i, ac_{i-1}, \text{open}_ac_{i-1}, v, \alpha)\). \( \mathcal{E} \) parses \( Q \) as \((V, D_p, D_i)\) and computes \( A' = V^{1/v} \). \( \mathcal{E} \) parses \( T \) as \((\text{params}_\text{Com}, pk_{\text{enc}}; g, H, pk, C_1, \ldots, C_N)\) and, for \( i = 1 \) to \( N \), \( C_i \) as \((A_i, B_i, p_i)\). If \( A' \) is not one of the signatures \( A_i \) that were previously sent to \( A \), \( \mathcal{E} \) aborts. Otherwise \( \mathcal{E} \) uses \( A_i \) to obtain the selection value \( \sigma \). Then \( \mathcal{E} \) checks that the extracted \( ac_i \) is correct, i.e., that it equals \( ac_{i-1} - p_\sigma \), and aborts if it is not the case. Otherwise \( \mathcal{E} \) sends \((\text{sid}, \text{buyerreq}, \sigma)\) to \( \mathcal{F}_{\text{POT}} \).

6. Upon receiving \((\text{sid}, m_\sigma)\) from \( \mathcal{F}_{\text{POT}} \), \( \mathcal{E} \) parses \( T \) as \((\text{params}_\text{Com}, pk_{\text{enc}}; g, H, pk, C_1, \ldots, C_N)\) and \( C_\sigma \) as \((A_\sigma, B_\sigma, p_\sigma)\). \( \mathcal{E} \) sets \( R = (B_\sigma/m_\sigma)^v \) and sends \((\text{sid}, R)\) to \( A \).
7. \( \mathcal{E} \) runs \( \Pi_3 \) with \( A \). Since \( R \) was not correctly computed, \( \mathcal{E} \) uses the simulator of the zero-knowledge proof of knowledge to compute a simulated proof of a false statement.

**Theorem 3** When only the buyer \( B \) is corrupted, the distribution ensembles \( \text{IDEAL}_{\text{OT}, \mathcal{E}, Z} \) and \( \text{REAL}_{\text{OT}, A, Z} \) are computationally indistinguishable under the \((N+1)\text{-SDH}\) assumption, the \((N+1)\text{-PDDH}\) assumption and the strong RSA assumption.

**Proof.**

**Game 0:** This game corresponds to the execution of the real-world protocol with an honest \( V \). Therefore, \( \Pr[\text{Game 0}] = 0 \).

**Game 1:** This game is identical to **Game 0**, except that we use the knowledge extractor of \( \Pi_2 \) to obtain witness \((p_\sigma, \text{open}_p, ac_i, \text{open}_{ac_i}, ac_{i-1}, \text{open}_{ac_{i-1}}, v, \alpha)\). If extraction fails, \( \mathcal{E} \) aborts. The probability that extraction fails is given by the knowledge error of \( \Pi_2 \), which is negligible. Therefore \( |\Pr[\text{Game 1}] - \Pr[\text{Game 0}]| = \nu_1 \).

**Game 2:** This game is identical to **Game 1**, except that \( \mathcal{E} \) aborts when the value \( V^{1/v} \) is not one of the signatures that were previously sent to \( A \). The signature scheme is weekly unforgeable under SDH assumption. The probability that \( Z \) distinguishes between **Game 2** and **Game 1** is bounded by the following lemma:

**Lemma 1** If the \((N+1)\text{-SDH}\) assumption holds, \( |\Pr[\text{Game 2}] - \Pr[\text{Game 1}]| = \nu_2 \).

**Game 3:** This game is identical to **Game 2**, except that \( \mathcal{E} \) aborts when the value \( ac_i \) does not equal \( ac_{i-1} - p_\sigma \). This means that \( A \) was able to break the binding property of Damgård-Fujisaki commitments, which holds under the strong RSA assumption. Therefore, the probability that \( Z \) distinguishes between **Game 3** and **Game 2** is bounded by the following theorem:

**Theorem 4** If the strong RSA assumption holds, \( |\Pr[\text{Game 3}] - \Pr[\text{Game 2}]| = \nu_3 \).

**Proof.** We build an algorithm \( D \) that breaks the binding property of the commitment scheme with non-negligible probability. Given such an algorithm, Damgård and Fujisaki show how to break the strong RSA assumption.
3.5. SECURITY PROOF

Given an adversary $\mathcal{A}$ that causes Game 3 to abort with non-negligible probability, $\mathcal{D}$ works as follows:

1. Upon receiving the parameters $\text{params}_{\text{Com}}$ of the commitment scheme from the challenger, $\mathcal{D}$ uses them to compute the database commitment $T$ and sends $T$ to $\mathcal{A}$.

2. After extracting witness $(p_\sigma, \text{open}_p, ac'_{i,1}, \text{open}_{ac'_{i,1}}, v, \alpha)$ from $\Pi_2$, where $ac'_{i,1}$ does not equal $ac_{i-1} - p_\sigma$, $\mathcal{D}$, given commitment $C = g^{ac_{i-1}-p_\sigma}h^{\text{open}_{ac_{i-1}}}$, is able to open $C$ to the values $(ac_i, \text{open}_{ac_i})$, where $ac_i = ac_{i-1} - p_\sigma$ and $\text{open}_{ac_i} = \text{open}_{ac_{i-1}} - \text{open}_p - \alpha$ (recall $h^\alpha = g^{ac_{i-1}-p_\sigma-ac'_{i,1}\text{open}_{ac_{i-1}}-\text{open}_p-\text{open}_{ac_i}}$).

3. $\mathcal{D}$ sends $(C, ac_i, \text{open}_{ac_i}, ac'_{i,1}, \text{open}'_{ac_{i,1}})$ to the challenger.

**Game 4:** This game is identical to Game 3, except that the database commitment $T$ is replaced by another valid database $T'$ computed on input the right messages and prices. Since both have identical distribution, $\Pr[\text{Game 4}] - \Pr[\text{Game 3}] = 0$.

**Game 5:** This game is identical to Game 4, except that now the database commitment $T'$ is computed by using random messages. At this point $\Pi_3$ is replaced by a simulated proof of a false statement. The probability that $Z$ distinguishes between Game 5 and Game 4 is bounded by the following theorem:

**Theorem 5** If the $(N+1)$-PDDH assumption holds, $|\Pr[\text{Game 5}] - \Pr[\text{Game 4}]| = \nu_4$.

**Proof.** Given an adversary $\mathcal{A}$ that distinguishes between Game 5 and Game 4 with non-negligible probability, we construct an algorithm $\mathcal{D}$ that breaks the $(N+1)$-PDDH assumption as follows. $\mathcal{D}$ receives the vector $(u, u^2, \ldots, u^{N+1}, V) \in \mathbb{G}^{N+2} \times \mathbb{G}_T$, where $x \in \mathbb{Z}_q$, and the vector $(T_1, \ldots, T_{N+1}) \in \mathbb{G}_T^{N+1}$. Let $T_0 = V$ and $f$ be the polynomial defined as $f(X) = \prod_{i=1}^{N} (X + p_i) = \sum_{i=0}^{N} c_i X^i$. Then $\mathcal{D}$ sets $g \leftarrow u^f(x) = \prod_{i=0}^{N} (u^x)^{c_i}$ and $y \leftarrow g^x = \prod_{i=0}^{N} (u^{x+i})^{c_i}$. If $f_i$ is the polynomial defined by $f_i(x) = f(X)/(X + p_i) = \sum_{j=0}^{N-1} a_{i,j} x^j$, then $\mathcal{D}$ can also compute the values $A_i = g^{1/(x+p_i)}$ as $A_i \leftarrow \prod_{j=0}^{N-1} (u^x)^{c_{i,j}}$. $\mathcal{D}$ then sets $H \leftarrow V^{f(x)} = \prod_{i=0}^{N} T_i^{c_i}$, and computes $B_i$ as $H^{1/(x+p_i)}$ as $B_i \leftarrow \prod_{j=0}^{N-1} T_i^{c_{i,j}}$, and continues the simulation. One can check that if $T_i = V^{x^i}$ then the simulation corresponds to Game 4, while if $(T_1, \ldots, T_{N+1})$ is random then the simulation corresponds to Game 5. Therefore, if $\mathcal{A}$ can distinguish between both, then $\mathcal{D}$ can break the $(N+1)$-PDDH assumption.
Simulation when none of the parties is corrupted. After receiving \((\text{sid}, p_1, \ldots, p_N)\) and \(k\) messages of the form \((\text{sid}, b)\), \(\mathcal{E}\) creates a simulated transcript by running copies of honest \(\mathcal{V}\) and \(\mathcal{B}\). \(\mathcal{V}\) is run on input random messages \((m'_1, \ldots, m'_N)\) and prices \((p_1, \ldots, p_N)\) while \(\mathcal{B}\) is run on input an account \(\text{ac}_0\) such that \(\text{ac}_0 > p_{\sigma_{\min}} k\), where \(\sigma_{\min}\) denotes the item with the lowest price. In the \(i\)th transfer phase, \(\mathcal{B}\) receives as input \(\sigma_{\min}\). If \(b_i = 0\) then \(\mathcal{V}\) sends an invalid response \((\text{sid}, \bot)\). Otherwise \(\mathcal{V}\) sends a valid response.

**Theorem 6** When none of the parties is corrupted, then \(\text{IDEAL}_{\text{FO}\text{T},\mathcal{E},\mathcal{Z}}\) and \(\text{REAL}_{\text{OT},\mathcal{A},\mathcal{Z}}\) are computationally indistinguishable under the \((N + 1)\)-PDDH and DLIN assumptions.

We do not provide a formal proof of this theorem. In the initialization phase, \(\mathcal{V}\)’s message consists of a random database, which we demonstrated that is indistinguishable from a real database under the \((N + 1)\)-PDDH assumption. The buyer’s message consists of the encryption of a different account value, which is secure under the DLIN assumption. In the transfer phase, the request message is replaced by a valid request message for message \(\sigma_{\min}\). The proof of knowledge is perfect zero-knowledge and Damg˚ard-Fujisaki commitments are information theoretically hiding. Therefore the request messages cannot be distinguished by environment \(\mathcal{Z}\). The response message involves a proof of knowledge that is perfect zero-knowledge as well.

Simulation when \(\mathcal{V}\) and \(\mathcal{B}\) are corrupted. In this case \(\mathcal{E}\) knows the inputs to \(\mathcal{B}\) and \(\mathcal{V}\) and so \(\mathcal{E}\) can simulate by computing the real messages that are sent by the two parties.

**Theorem 7** When both \(\mathcal{V}\) and \(\mathcal{B}\) are corrupted, then \(\text{IDEAL}_{\text{FO}\text{T},\mathcal{E},\mathcal{Z}}\) and \(\text{REAL}_{\text{OT},\mathcal{A},\mathcal{Z}}\) are indistinguishable.

We omit a formal proof of this theorem.

### 3.6 Features and Extensions

This scheme can be extended in order to offer extra features over previous ones [AIR01]. Namely, by using the signature scheme that signs blocks of messages, \(\mathcal{V}\) can compute a database commitment where each \(A_i = \text{MBSign}(pk, sk, \langle i, p_i \rangle)\). The rest of the scheme is conveniently modified as described in [CDN09]. This permits that several messages have the same price, which was not possible in the scheme proposed in [AIR01].

The scheme also allows the vendor to charge different prices for the same message to different buyers, which can be used to apply marketing techniques like making discounts to regular or underage buyers. This can be
done by recomputing the signatures included in the ciphertexts on different prices depending on the particular buyer. In order to allow for precomputed databases, \( \mathcal{V} \) can assign buyers to \( \ell \) different groups and associate to each group \( j \in \{1, \ldots, \ell\} \) a different price for each message \( m_i \) by signing \( A_{ij} = \text{MBSign}(pk, sk, (i, p_{ij})) \).
Chapter 4

Implementation of the Demonstrator

We describe here the implementation of a demonstrator that uses as a main building block the priced oblivious transfer (POT) scheme depicted in Chapter 3. This implementation separates the operations that correspond to the COT scheme from those related with the OP scheme in order to allow for modularity, i.e., someone interested in utilizing only the implementation of the COT scheme is able to do that straightforwardly.

We focus on the description of the functions that form the core of the OP and of the COT schemes, which are divided into two libraries, one for the vendor and one for the buyer. We also describe in detail how to use the buyer’s application. Other parts of the implementation, such as a client for the buyer and a server for the vendor, an interface to communicate the application with the buyer’s client, and an interface to communicate with the socket’s layer, which were made mainly to test the library, are explained briefly. However, this description can be complemented with the comments that are given in the source code.

To start with, we talk about the tools that were used to construct the implementation. After that, we describe its general structure and we give the necessary instructions to install and run it. The graphical user interface is depicted in Section 4.3. We focus on the OP scheme in Section 4.4.2 and in Section 4.4.3 we describe the implementation of the COT scheme.

4.1 Tools

Three main tools were used to build the implementation: the GNU Multiple Precision arithmetic (GMP) library, the Pairing-Based Cryptography (PBC) library and the Belgian electronic identity (eID) card middleware and
software development toolkit. The OP scheme needs only the GMP library, whereas the COT scheme needs both. The buyer’s application accesses the Belgian eID card via the middleware.

The GMP library is a free library for precision arithmetic that operates with integers, rational numbers and floating point numbers. According to its authors, it has a number of advantages: it is faster than any other library for operating with large numbers, there is no limit to the precision except the one implied by the hardware resources, etc. In addition, it has been designed specially for its use in cryptography and internet security applications. For further explanations about the library, we refer to its website\footnote{http://gmplib.org/}, where it can be downloaded and a manual and information about its different releases are provided.

In our implementation, we use only the functions that operate with integers and some number theory algorithms, like the Miller-Rabin algorithm to test probabilistically whether a number is prime. The release that we use is the Version 4.2.2.

On the other hand, we use the PBC library to compute bilinear maps, which is needed for the COT scheme. This is a recent library that was constructed by utilizing the GMP library and whose main purpose is the computation of pairings. It offers a pairing computation time between eleven milliseconds and several hundreds of milliseconds, depending on the parameters used to define the pairing. For further explanations we refer to its website\footnote{http://crypto.stanford.edu/pbc/}, where it can be downloaded along with a manual.

We mainly use the routines for initializing and computing pairings, but also some arithmetic functions. The release that we utilize is the Version 0.4.12.

The Belgian eID card SDK allows accessing the Belgian eID card by providing API’s for C++, Java and dotNet languages. It needs that the Belgian eID middleware be installed in order to work. Both of them can be obtained through the Belgian eID web site\footnote{http://eid.belgium.be/nl/} which also contains information about which card readers are compatible with the Belgian eID card. In our implementation, we employ the MW and SDK version 3.5 designed for Linux Debian 4.

Since GMP and PBC use C as programming language, we have used C to build the implementation of the COT and OP schemes. The GUI is implemented in Java, and thus we employ the eID SDK 3.5 for that language.
4.2. ARCHITECTURE OF THE DEMONSTRATOR

In this section we give a general overview on how this implementation is structured. We describe its different parts and the content of each file of source code in Section 4.2.1. After that, we explain how to install and run it in Section 4.2.2. Finally, in Section 4.4.1 we show the different types of functions that are part of the libraries of the buyer and of the vendor and the nomenclature used in both libraries.

4.2.1 Organization of the Source Code

As mentioned at the beginning of the chapter, the implementation, apart from the libraries for the buyer and for the vendor, has several parts that help us to check their correctness and that can be utilized to build practical applications. Figure 4.1 In the following we relate the different parts along with the name of their corresponding files and a description of their content.

Vendor’s library. It contains the functions that are needed to implement the algorithms (POTInitVendor, POTGetDeposit, POTRespond). The file “VendorLibrary.c” contains the body of the functions and “VendorLibrary.h” contains their respective heads. See Section 4.4.2 and Section 4.4.3 for more details.
Buyer’s library. It contains the functions that are needed to implement the algorithms (POTInitBuyer, POTRequest, POTComplete). The file “BuyerLibrary.c” contains the body of the functions and “BuyerLibrary.h” contains their respective heads. See Section 4.4.2 and Section 4.4.3 for more details.

Client and Server. They are a client for the buyer, in file “BuyerMain.c”, and a server for the vendor, in file “VendorMain.c”. Both use the libraries mentioned above to build a POT scheme.

Application. It contains a graphical user interface that can be used by a buyer to purchase digital goods. It provides the initialization and transfer functionalities. It employs the Java programming language, and it communicates with the client of the buyer through an interface that employs the C programming language. To invoke functions of this interface, the Java Native Interface is utilized. The file “POT.java” contains the main class. In the folder “window”, the files “Ventana.java”, “Dialogo.java” and “DialogoOK.java” contain the classes that form part of the package “window”, which is imported by “POT.java”. In addition, the file “Ventana.java” imports the package “be.belgium.eid.*”, which contains all the classes defined by the eID card SDK, and it also loads two dynamic libraries: “libbeidlibJava_Wrapper.so”, which is provided by the SDK, and “libinterfaceVent.so”, which contains all the functions provided by the interface for the application.

Interface for the transport layer. It contains functions that mask the C functions to create and use TCP sockets. These functions are used by the libraries to communicate buyer and vendor. The file “communication.c” contains the body and “communication.h” the heads.

Interface for the application. It contains functions which mask the C functions to create and use datagram sockets. They are used by the application and the buyer’s library to communicate with each other. The files “interface.c” and “interface.h” contain the body and the heads of the functions that are employed by the buyer’s library. The files “interfaceVent.c” and “window_Ventana.h” contain the body and the heads of the functions used by the buyer’s application.

Hash function. The files “md5.c” and “md5.h” contain the implementation of the hash function md5 (RFC 1321). It is used to extract a random pad from the element in the target group in the COT scheme (see Section 3.4). The concrete implementation that was utilized is a free and widely used one that can be found in several websites.\footnote{For example, http://www.fourmilab.ch/md5/}
4.2. ARCHITECTURE OF THE DEMONSTRATOR

Pairing Generation. It contains an application that creates a file with pairing parameters of Type A (see Section 4.5.2.1). The file “TypeA-PairingGeneration.c” contains this application.

In addition, the file “types.h” contains the types of data used in both libraries along with macros that define the security parameters and other constants.

4.2.2 Installation and Execution

In order to install all this source code, first it is necessary to install the GMP library, the PBC library and the eID MW and SDK. After that, by means of the “Makefile” file, which uses gcc as compiler, it is possible to install all the components but the buyer’s application and corresponding interface by typing “make vendor” and “make buyer” respectively. The vendor needs his library and the interface for the transport layer, the hash function and the file “types.h”. The buyer needs her library and the interface for the transport layer, the interface for the application (files “interface.c” and “interface.h”), the hash function and the file “types.h”. The application to create pairing parameters can also be compiled with “make paramgen”.

In order to install the buyer’s application and interface, one should first compile the application GUI:

```
javac window/Dialogo.java
javac window/DialogoOK.java
javac window/Ventana.java
javac POT.java
```

Then it is necessary to create the header file “window.Ventana.h” for the native functions that are used by class “Ventana” and store it in folder window:

```
javah -jni -classpath [Path to window’s parent directory] window.Ventana
cp window.Ventana.h window
```

Finally, it is necessary to create the library “libinterfaceVent.so”, which should be stored in folder window:

```
gcc -c -fPIC interfaceVent.c -o interfaceVent.o
gcc -shared -Wl,-soname,libinterfaceVent.so.1 -o libinterfaceVent.so interfaceVent.o
```

To execute the vendor, the buyer and the application of the buyer it is necessary to type the following commands:

```
vendor -fm messages -fp prices [-p port] < params
buyer -f POT.dg [-p port] < params
java -Djava.library.path=window POT
```
CHAPTER 4. IMPLEMENTATION OF THE DEMONSTRATOR

messages is a file that contains the messages that the vendor offers to the buyer. Each line contains one message, a blank and the price of the message. The $i$th file contains the message corresponding to the $i$th item. prices is a file that contains the price of the $i$th item in the $i$th file. port is the number of the listening port that the vendor uses to receive connection queries from buyers. socket is the name of the file of the socket that is used to communicate the client of the buyer with the application of the buyer. “POT.dg” is the name of the file used as a default option by the application, and the file should be removed after the protocol execution. Finally, params is the name of a file that contains the pairing parameters. This file can be created by typing:

```
./paramgen [-f params] [-r rbits] [-q qbits]
```

$rbits$ is the bit-length of the group order and $qbits$ is the bit-length of the base field (see Section 4.5.2.1).

4.3 The Graphical User Interface

In this section, we explain how to use the graphical user interface to carry out initializations and transfers. We need that the vendor’s server and the buyer’s client are already being executed, and that a card reader with a Belgian eID card has been plugged in.

4.3.1 Initialization

Figure 4.2 shows the main window of the application at an initial stage. In the toolbar, there are three menus: File, Edit and Help. In order to start the initialization phase, the buyer should choose Edit and then click on Send Deposit. If the card is not detected, then the error message in Figure 4.3 is displayed. Otherwise the buyer is asked to introduce the amount of money she wants to deposit (see Figure 4.4).

As a result, if the deposit was successfully sent, the buyer receives a confirmation message that is displayed on the left side panel, and the prices of all the messages that are offered by the vendor on the right side list, as can be seen in Figure 4.5. Otherwise the buyer receives an error message that is displayed on the left side panel.

4.3.2 Transfer

In order to carry out a purchase, the buyer should choose Edit and then click on Purchase Item. If the initialization phase was not completed successfully
4.3. THE GRAPHICAL USER INTERFACE

Figure 4.2: Initial Window

Figure 4.3: Error Dialog: No Card Detected
CHAPTER 4. IMPLEMENTATION OF THE DEMONSTRATOR

Figure 4.4: Input Deposit

Figure 4.5: Initialization Complete
4.3. THE GRAPHICAL USER INTERFACE

Figure 4.6: Error Dialog: Initialization Incomplete

before, then the error dialog in Figure 4.6 is displayed. Otherwise the buyer is asked to introduce the index of the message of her choice (see Figure 4.7). The buyer can also select the message index by clicking on the list of messages before the beginning of the purchase phase, as can be seen in Figure 4.8. In order to hide from the vendor when a purchase is carried out, the buyer can select a dummy item with index and price 0.

After selecting the index, the buyer gets an error message if the price of the corresponding message is greater than her current deposit (see Figure 4.9). Otherwise the buyer receives the message from the vendor (see Figure 4.10). Finally, the main window is updated with a confirmation message on the left side panel and with the message on the right side list, as can be seen in Figure 4.11. The buyer can check her current deposit by selecting the Edit menu and clicking on View Deposit (see Figure 4.12).

4.3.3 Other functionalities

The File menu allows closing the application. The Help Menu gives contact information of its author (see Figure 4.13).

4.3.4 Extensions

In order to use the demonstrator as a building block of an e-commerce website, several additional tools should be used. First of all, we note that the
CHAPTER 4. IMPLEMENTATION OF THE DEMONSTRATOR

Figure 4.7: Input Choice

Figure 4.8: Item Choice
4.3. THE GRAPHICAL USER INTERFACE

Figure 4.9: Error Dialog: Not Enough Funds

Figure 4.10: Purchase Result
CHAPTER 4. IMPLEMENTATION OF THE DEMONSTRATOR

Figure 4.11: Purchase Complete

Figure 4.12: Deposit View
deposit that the buyer sends to the vendor at the initialization phase is not a real payment. This payment should be carried out by employing other existing payment methods, and the vendor should check that the amount of money paid equals the deposit.

In addition, we note that the provided application can also be run as an applet. However, applets are subject to additional security restrictions which do not allow invoking native functions. In order to solve this problem, the applet must be signed.\footnote{http://java.sun.com/developer/onlineTraining/Programming/JDCBook/signed.html}

Finally, to authenticate the user towards the vendor, an additional applet for authentication purposes should be included in the vendor’s web site. After authenticating the user via this additional applet, our applet ensures that a malicious buyer does not remove the card from the card reader and that she does not plug in a different card.

4.4 Implementation Details

4.4.1 Function Classes and Nomenclature

Both the library of the vendor and the library of the buyer use the same notation in order to facilitate the comprehension of the purpose of a function\footnote{http://code.google.com/p/eid-applet/}.
or of a type of data. In the following, \((V_l, B_l)\) denote the vendor and buyer initialization algorithms of the OP scheme, and \((V_T, B_T)\) denote the vendor and buyer transfer algorithms of the COT scheme. Similarly, \((S_l, R_l)\) denote the sender and receiver initialization algorithms of the COT scheme, and \((S_T, R_T)\) denote the sender and receiver transfer algorithms of the COT scheme. There are three generic types of functions in the libraries:

**Main Functions.** These are the functions called by the client of the buyer and the server of the vendor. They offer an interface that implements the algorithms \((V_l, B_l, V_T, B_T)\) of the OP scheme and the algorithms \((S_l, R_l, S_T, R_T)\) of the COT scheme. The functions \(BI, VT\) and \(BT\) of the OP scheme and \(ST\) and \(RT\) of the COT scheme implement their respective algorithms, whereas algorithms \(V_l, S_l\) and \(R_l\) are split into two functions, \(Setup_V\) and \(VI\) for \(V_l\), \(Setup_S\) and \(SI\) for \(S_l\), and \(Setup_R\) and \(RI\) for \(R_l\). The setup functions contain the parts of the initialization algorithms that need to be computed only once.

**I/O Functions.** These are functions used to exchange information between the buyer and the vendor. The functions used to send information are referred to as \(send\_message\_\ast\), whereas the ones utilized to receive information are called \(receive\_message\_\ast\). \(\ast\) indicates the Main Function that calls a particular function. For example, \(receive\_message\_VT\) is called by the Main Function \(VT\), and its counterpart, \(send\_message\_BT\), is called by \(BT\).

**PoK Functions.** These functions are used to run a proof of knowledge and are named by the tag \(PK\_\ast\), where \(\ast\) has the same meaning as before. After an I/O Function pair \(receive\_message\_-send\_message\_\ast\), a PoK function pair is run, in which the party who sends data proves knowledge of it according to the description of the OP and of the COT schemes.

Each \(PK\_\ast\) Function calls four functions that represent the four steps of a proof. First, one out of the pair \(send\_commitment\_\ast-receive\_commitment\_\ast\), then \(receive\_challenge-send\_challenge\), after that \(send\_response\_\ast-receive\_response\_\ast\) and finally \(receive\_verification-send\_verification\). A prover executes the functions on the left of the hyphen and a verifier the ones on the right. Here \(\ast\) stands for the name of the \(PK\_\ast\) Function that invokes the function. Note that the functions for exchanging the challenge and the result of the verification are the same for all proofs.

All the functions return an integer that indicates an error when it is negative. This error is not related with the result of the computation, i.e., it indicates a failure in the execution of the function. For example, the fact
that a prover does not succeed is indicated through a parameter that is output by a PoK Function, not by the returned value.

The general structure of a Main Function consists in invoking first an I/O function and after that a PoK Function. If the proof succeeds, it proceeds with the next I/O-PoK function pair, and otherwise it returns an error. We should note that these functions use the functions of the interface with the transport layer. However, it is possible to change the body of the functions of the interface in order to use a different transport layer.

Apart from that, there are some auxiliary functions that implement some specific algorithms, like, for example, a function to output Sophie Germain primes that is used during the vendor’s setup. These functions will be detailed in Section 4.4.2 and in Section 4.4.3.

On the other hand, the types of data that were created for both libraries are structures referred to as $s_\ast$, where $\ast$ is a description of the content of the structure. The corresponding variables are called $\ast$. If there is more than one variable of the same type, a figure is concatenated to $\ast$.

The main idea of this design is that the only functions that the server and the client need to call in order to construct a POT scheme are the Main Functions. Nevertheless, it is possible to access directly the other kinds of functions, which can be used to implement a different POT scheme.

Most of the parameters that are given as input to the functions of both libraries are pointers to structures or to arrays of structures. We should note that these functions do not reserve memory for these variables, and thus the calling functions are the ones that have to do it.

In order to build a server for the vendor that handles multiple buyers the usual approach of creating threads should be taken. Each thread will call one of the Main Functions, and thus an initialization phase or a subphase of a transfer phase will be completely managed by the thread.

### 4.4.2 Implementations Details of the OP Scheme

In this section we explain the implementation of the OP scheme. We begin by talking about the data structures and after that we depict the functions that are needed to implement the setup, the initialization and the transfer phase for both the vendor and the buyer. Finally, in Section 4.5.1.1 we talk about the security parameters of the scheme and in Section 4.5.1.2 we discuss its efficiency.

As a general remark, we note that, in order to let different items have the same price, we scale the value of the account by multiplying it with a value contained in the macro $scale$ of the file “types.h”. Then the prices of the items become $scale \cdot p_i - i$, where $p_i$ is the price of the item and $i$ is its corresponding selection value.
4.4.2.1 Data Structures

All the types defined for the implementation of the OP scheme are structures. In the following we show the name of the type and its function:

\textbf{s\_commitment\_DF}. It contains the values $\text{params}_{\text{comm}} = (n, g, h)$, i.e., the public parameters of the Damgård-Fujisaki commitment scheme.

\textbf{s\_secret\_key\_V}. It contains the secret key $s_k = (p, q)$, i.e., the two primes such that $n = pq$.

\textbf{s\_alpha}. It contains the value $\alpha$ such that $g = h^\alpha$.

\textbf{s\_price}. It contains the price of an item.

\textbf{s\_payment}. It contains the commitments $C_{\text{account}_i}$ and $C_{\text{price}_i}$ that the buyer sends to the vendor at the $i$th transfer phase.

\textbf{s\_payment\_buyer}. It contains the values $\text{account}_i$, $\text{open}_{\text{account}_i}$, $\text{price}$ and $\text{open}_{\text{price}}$ that were used to compute the commitments mentioned above.

\textbf{s\_item}. It contains the commitment $C_{\sigma_i}$ that the buyer sends to the vendor at the $i$th transfer phase.

\textbf{s\_item\_receiver}. It contains the values $\sigma_i$ and $\text{open}_{\sigma_i}$ that were used to compute the commitment mentioned above.

\textbf{s\_commitment\_FS}. It contains the commitments $C_{w_1}, C_{w_2}, C_{w_3}, C_{w_4}$ to the four values $w_1, w_2, w_3, w_4$ such that $\text{account}_i = w_1^2 + w_2^2 + w_3^2 + w_4^2$. They are used to prove that the account is non-negative by using the proof in [Lip03].

\textbf{s\_FS}. It contains the four values $w_1, w_2, w_3, w_4$ and the values $\text{open}_{w_1}$, $\text{open}_{w_2}$, $\text{open}_{w_3}$, $\text{open}_{w_4}$ that were used to compute the commitments mentioned above.

\textbf{s\_commitmentPK}. It contains one commitment that is sent in the first step of a proof of knowledge.

\textbf{s\_randomnessPK}. It contains one random value that was used to compute a commitment in the first step of a proof of knowledge.

\textbf{s\_challengePK}. It contains the challenge that is sent in the second step of a proof of knowledge.

\textbf{s\_responsePK}. It contains one response that is sent in the third step of a proof of knowledge.
For each value, each structure stores two fields: one of type `unsigned char[]` and the other one of type `unsigned int`. The former contains the value that we want to store. It is an array and not an integer because there we store the output of the Export functions of the GMP library. These functions also output the size of the element in bytes, which is stored in the latter field. Therefore, when we want to use a value contained in this structure we need to use the Import function that is provided by the GMP library in order to get back the integer.

The last structure, `s_responsePK`, contains a field “negative”, which is used to indicate whether the number is negative or not. This is necessary because the import/export functions of the GMP library use the absolute value. The other structures do not include this field because they never store a negative value.

### 4.4.2.2 Setup

The setup of the OP scheme includes all the parts of the initialization phase that can be precomputed, i.e., that do not need to be computed anew for each initialization phase. It takes place only in the vendor’s side.

The functions that are run in this phase are the Main Function `Setup_V` and the auxiliary functions `setup_commit_DF`, `sophie germain_prime_generation` and `get_prices`. The function `Setup_V`, on input the name of the file that contains the prices of the items in `*file_prices`, calls first the function `setup_commit_DF`, whose behavior is described in detail below. This function runs the function `sophie germain_prime_generation` and outputs the parameters \((n, g, h)\) of the Damgård-Fujisaki commitment scheme in `*commitment_DF`, the values \(p, q\) such that \(n = pq\) in `*secret_key` and the value \(\alpha\) such that \(g = h^\alpha\) in `*alph`.

Second, `Setup_V` calls `get_prices` on input `*file_prices`. This function reads the prices from the file and outputs an array of prices in `*price` and the number of elements in `*number_prices`. In order to have protection against big files, the macro `size_comm` limits the maximum number of lines that can be read.

Finally, `Setup_V` outputs `*commitment_DF`, `*secret_key`, `*alph`, `*price` and `*number_prices`.

**Setup of the Damgård-Fujisaki Commitment Scheme** The setup of the Damgård-Fujisaki commitment scheme (see Section 3.2) corresponds to the first step of the initialization algorithm of the OP scheme described in Section 3.4. Recall that this `SetupCommit(1^k)` algorithm, on input a security parameter \(k\), outputs a special RSA modulus \(n\) of bit-length given by \(k\) and two generators \(g, h\) of the group of quadratic residues \(QR_n\).
First Step. The first step consists in generating two Sophie Germain primes $p', q'$ and is implemented in function `sophie_germain_prime_generation`. For this purpose, a procedure described in [CS00] has been followed:

1. Generate a random odd number in the interval $2^{\frac{k}{2}} - 1 \leq p' < 2^{\frac{k}{2}}$.
2. Test if either $p'$ or $p = 2p' + 1$ are divisible by any primes up to some bound $B$. If so, go back to Step 1.
3. Test if 2 is a Miller-Rabin witness to the compositeness of $p'$. This involves the following steps:
   (a) Express $p' - 1$ as $2^s d$.
   (b) Check if $2^d = 1 \mod p'$.
   (c) For $i = 0$ to $s - 1$, check if $2^{2d^i} = p' - 1 \mod p'$.

   If none of the checks succeed, then 2 is a witness and so go back to Step 1.
4. Test if $2^p' = \pm 1 \mod p$. If not, go back to Step 1.
5. Apply the Miller-Rabin algorithm as many times as necessary with randomly selected bases in order to ensure an error probability lower than $\epsilon$.

This procedure is applied twice for the generation of both $p'$ and $q'$.

In Step 2, we use $B = \left(\frac{k}{2}\right)^4$. According to [CS00], the probability that a random number passes this test is approximately $0.416/(\log B)^2$ when $k$ tends to infinity.

On the other hand, according to [CS00] it is desirable to have an error probability $\epsilon < 2^{-80}$. Let $p_{k/2,t}$ be the probability that a composite $p'$ passes the Miller-Rabin algorithm when applying it $t$ times. The formula that relates this probability with the error probability is:

$$p_{k/2,t} \leq \frac{\epsilon}{\epsilon + 2(k/2)}$$

In [DLP93], there is a table that gives the number of times $t$ depending on $k/2$ and the probability $p_{k/2,t}$. For example, for $k/2 = 512$, we need $p \approx 2^{-90}$. Then, following that table we see that $t = 6$ gives a probability $p_{k/2,t} \leq 2^{-91}$ for $k/2 = 500$ and $p_{k/2,t} \leq 2^{-96}$ for $k/2 = 550$.

Therefore, we decided to use, with $k/2 = 512$, a number of times $t = 6$ to ensure a negligible error probability. For further explanations we refer to [CS00].
4.4. IMPLEMENTATION DETAILS

Second Step. The second step, which is implemented in function \texttt{setup\_commit\_DF}, consists in generating \((n, g, h)\) such that \(n = pq\), where \(p = 2p' + 1\) and \(q = 2q' + 1\), and \(g, h \in QR_n\) such that \(g \in \langle h \rangle\).

In order to generate \(h \in QR_n\), we need to pick a random number \(h^*\) in \(\mathbb{Z}_n^*\) and check whether it belongs both to the set of quadratic residues modulo \(p (QR_p)\) and to the set of quadratic residues modulo \(q (QR_q)\). This is done by checking both if \(h^{p'} = 1 \mod p\) and if \(h^{q'} = 1 \mod q\). If both checks succeed, then \(h^*\) is a quadratic residue and we set \(h = h^*\).

To compute \(g\), we recall that in \(QR_n\) almost all the elements \(\phi(p'q')\) are generators. Thus, we pick a random \(\alpha\) such that \(1 < \alpha < |QR_n|\), where \(|QR_n| = p'q'\), and we compute \(g = h^\alpha \mod n\). Finally, we set \(g = g^*\).

4.4.2.3 Initialization Phase

The implementation of the initialization of the OP scheme contains all the parts of the \((V, B)\) algorithms that need interaction between buyer and vendor. First, the vendor sends the buyer the parameters of the Damgård-Fujsisaki commitment scheme and proves knowledge of the value \(\alpha\). After that, the buyer sends him the deposit and a commitment to it, and also proves that the commitment is correctly formed.

First we describe the functions of the vendor and after that we depict the functions of the buyer.

Vendor Initialization Phase. The functions that are run in this phase on the vendor’s side are the Main Function \(VI\), the I/O Functions \texttt{receive\_message\_VI} and \texttt{send\_message\_VI} and the PoK Functions \texttt{PK\_VI\_1} and \texttt{PK\_VI\_2}, along with the functions that implement these proofs of knowledge.

The function \(VI\) is run on input the parameters \((n, g, h)\) of the Damgård-Fujsisaki commitment scheme in \(*commitment\_DF\), the value \(\alpha\) in \(*alph\), the array of the prices of the items in \(*price\) and the number of elements of the array in \(*number\_prices\). First, \(VI\) runs \texttt{send\_message\_VI} on input \(*commitment\_DF\), \(*price\) and \(*number\_prices\) in order to send to the receiver the parameters of the commitment scheme and the prices of the items.

After that, it calls \texttt{PK\_VI\_1} on input \(*commitment\_DF\) and \(*alph\) in order to prove to the buyer that \(g = h^\alpha\). For this purpose, \(PK\_VI\_1\) invokes \texttt{send\_commitment\_PK\_VI\_1} to send a value \(t = h^\alpha\). After receiving the challenge \(c\) by means of \texttt{receive\_challenge}, it runs \texttt{send\_response\_VI\_1} in order to send a response \(s = r - ca\). Finally, it calls \texttt{receive\_verification} to get the result, which is output by \(PK\_VI\_1\) in result.

If the result is not zero, \(VI\) returns and outputs \(*result\). Otherwise it starts the second step by calling \texttt{receive\_message\_VI} in order to receive the deposit in \(*deposit\) and a commitment \(C_{account}\) to the deposit in \(*pay-\)
CHAPTER 4. IMPLEMENTATION OF THE DEMONSTRATOR

ment. Then, in order to check that this commitment was correctly computed by the buyer, VI invokes PK_VI_2 on input *commitment_DF, *payment and *deposit. After receiving the commitment t by means of receive_commitment_PK_VI_2, PK_VI_2 runs send_challenge in order to send a challenge c, which is output in *challenge. It runs receive_response_PK_VI_2 in order to receive the response s and check whether \( t = h^{s}(C_{\text{account}}/g^{\text{deposit}})^c \). Finally it sends the result by means of send_verification, and it outputs the result in *result.

In the end, VI outputs the deposit in *deposit, C_{\text{account}} in *payment and the result in *result.

**Buyer Initialization Phase.** The functions that are run in this phase in the buyer’s side are the Main Function BI, the I/O Functions receive_message_BI and send_message_BI and the PoK Functions PK_BI_1 and PK_BI_2, along with the functions that implement these proofs of knowledge.

The function BI receives as input the deposit in deposit. First of all, BI invokes receive_message_BI in order to get the parameters \((n, g, h)\) in *commitment_DF, the array of prices of the items in price and the number of elements in the array in *number_prices. After that, BI invokes PK_BI_1 in order to verify if \((n, g, h)\) were computed correctly by the vendor. For this purpose, first PK_BI_1 calls receive_commitment_PK_BI_1 in order to get the commitment \( t = h^r \) in *commitment, and then it sends the challenge c by invoking send_challenge. After that, it receives the response \( s = r - c_{\alpha} \) and verifies if \( t = h^s g^c \) by means of receive_response_PK_BI_1. It sends the result that is output by this function to the vendor by using send_verification. Finally, PK_BI_1 outputs the result in *result. If it is 0, BI proceeds with the second step; otherwise it returns and outputs *result.

The second step begins when BI calls send_message_BI and gives in deposit the deposit account_0 as input in order to send both this value and a commitment to the deposit \( C_{\text{account}} \) to the vendor. The value account_0 and the randomness of the commitment open_account_0 are output in *payment_buyer. Afterwards, it invokes PK_BI_2 in order to prove that the commitment was computed correctly. First PK_BI_2 uses send_commitment_PK_BI_2 to send a commitment \( t = h^r \), and then it calls receive_challenge in order to receive a challenge c. After that, it invokes send_response_PK_BI_2 in order to compute \( s = r - c_{\text{open account}} \) and send s to the vendor, and it receives the result of the verification by running receive_verification. It outputs the result in *result. Finally, BI outputs account_0 and the randomness of the commitment open_account_0 in *payment_buyer, and the result in *result.
4.4. IMPLEMENTATION DETAILS

4.4.2.4 Transfer Phase

The implementation of the transfer phase of the OP scheme contains everything necessary to implement the algorithms \((V_T, B_T)\). First we describe the functions of the vendor and after that we depict the functions of the buyer.

**Vendor Transfer Phase.** The functions that are run in this phase in the vendor’s side are the Main Function \(VT\), the I/O Function \(receive\_message\_VT\) and the PoK Function \(PK\_VT\), along with the functions that implement this proof of knowledge.

First of all, \(VT\), which receives as input the parameters \((n, g, h)\) in \(*commitment\_DF\) and the commitment to the account \(C_{\text{account}_{i-1}}\) in \(*payment\), invokes \(receive\_message\_VT\) in order to get the commitment \(C_{\text{account}_i}\) to the new value of the account and the commitment to the price \(C_{\text{price}_i}\) both in \(*new\_payment\), and also a commitment to the item \(\sigma_i\) in \(*item\). In addition, it receives in \(*commitment\_FS\) commitments \((C_{w_1}, C_{w_2}, C_{w_3}, C_{w_4})\) to four values \((w_1, w_2, w_3, w_4)\) such that \(\text{account}_i = w_1^1 + w_2^2 + w_3^3 + w_4^4\).

After that, \(VT\) runs \(PK\_VT\) to verify that the new commitment to the account was computed correctly, i.e., that the value \(\text{account}_i\) is not lower than zero and that it equals \(\text{account}_i = \text{account}_{i-1} - p\). First, \(PK\_VT\) calls \(receive\_commitment\_PK\_VT\) to obtain an array of seven commitments in \(*commitment\). These commitments are of the form \(t_{w_1} = g^{w_1}g^{\text{open}_{w_1}}\) and similar for \(t_{w_2}, t_{w_3}\) and \(t_{w_4}\), \(t_{\text{account}_i} = g^{\text{account}_i}h^{\text{open}_{\text{account}_i}}\), \(t_{\text{notneg}} = C_{w_1}C_{w_2}C_{w_3}C_{w_4}h^{\alpha}\), where \(\alpha = \text{open}_{\text{account}_i} - \text{open}_{w_1}w_1 - \text{open}_{w_2}w_2 - \text{open}_{w_3}w_3 - \text{open}_{w_4}w_4\), and \(t_\beta = h^{\beta}\), where \(\beta = \text{open}_{\text{account}_{i-1}} - \text{open}_{\text{account}_i} - \text{open}_{\text{price}_i}\). These commitments are mentioned in the same order as they appear in the array.

Second, \(PK\_VT\) runs \(send\_challenge\) in order to send a challenge \(c\) to the buyer. Then it invokes \(receive\_response\_PK\_VT\) in order to receive the twelve responses of the buyer and verify them (see the next subsection for the order of the responses in the array and for their computation).

In the input of this function, \(*commitment\) has to be the array of seven structures that was output by \(receive\_commitment\_PK\_VT\), whereas the other parameters are pointers to one structure as usual. It verifies whether \(t_{w_1} = g^{\text{account}_1}h^{\text{open}_{w_1}}C_{w_1}^c\) and it performs a similar check for \(t_{w_2}, t_{w_3}\) and \(t_{w_4}\).

It also verifies that \(t_{\text{account}_i} = g^{\text{account}_i}h^{\text{open}_{\text{account}_i}}C_{\text{account}_i}^c\), that \(t_{\text{notneg}} = C_{w_1}C_{w_2}C_{w_3}C_{w_4}h^{\alpha}\), and \(t_\beta = h^{\beta}\) \((C_{\text{account}_{i-1}}/C_{\text{account}_i})^c\).

---

7In the scheme implemented, the commitment to the price and the commitment to the item have the same value. Different variables are used in order to facilitate the extension of this implementation to construct the extended scheme. In order to ensure that the commitment to the price is used as commitment to the item in the transfer phase of the COT scheme, this function copies \(C_{\text{price}_i}\) in \(C_{\text{account}_i}\).
Finally, if all the checks are true, \(\text{receive\_response\_PK\_VT}\) sets \(*\text{result}\) to 0 and outputs it. After that, \(\text{PK\_VT}\) runs \(\text{send\_verification}\) to send the result to the buyer and outputs \(*\text{result}\). Finally, \(\text{VT}\) outputs \(*\text{result}\).

**Buyer Transfer Phase.** The functions that are run in this phase in the vendor’s side are the Main Function \(\text{BT}\), the I/O Function \(\text{send\_message\_BT}\) and the PoK Function \(\text{PK\_BT}\), along with the functions that implement this proof of knowledge.

First of all, \(\text{BT}\) receives as input the selection value in \(\text{choice}\), the array of the prices of the items in \(\text{price}\), the number of elements of the array in \(\text{number\_prices}\), the parameters \((n, g, h)\) in \(\text{commitment\_DF}\) and the value of the account in the last transfer phase \(\text{account}_{i-1}\) and the random value \(\text{open\_account}_{i-1}\) that was used to compute the last commitment to the account both in \(\text{payment\_buyer}\).

\(\text{BT}\) invokes \(\text{send\_message\_BT}\) with the same inputs. If the selection value is such that \(0 \leq \sigma_i < \text{number\_prices}\) and the new account \(\text{account}_i \geq 0\), this function computes the new account value \(\text{account}_i = \text{account}_{i-1} - p\sigma_i\), a commitment \(C_{\text{account}_i} = g^{\text{account}_i} h^{\text{open\_account}_i}\) to it, a commitment to the price \(C_{p\sigma_i} = g^{p\sigma_i} h^{\text{open\_psi}}\), a commitment to the item \(C_{\sigma_i} = g^{\sigma_i} h^{\text{open\_si}}\) and four commitments \((C_{w_1}, C_{w_2}, C_{w_3}, C_{w_4})\) to four values \((w_1, w_2, w_3, w_4)\) such that \(\text{account}_i = w_1^2 + w_2^2 + w_3^2 + w_4^2\), and \(*\text{result}\) is set to 0. Otherwise it sets \(*\text{result}\) to another value and returns.

The values \((w_1, w_2, w_3, w_4)\) are computed by means of the Rabin-Shallit algorithm. We have used an improved version proposed in [Lip03]. It works as follows:

**Step 1** Find \(t\) and \(k \geq 0\) such that \(\text{account}_i = 2^t(2^k + 1)\).

**Step 2** Pick random values \(w_1 \leq \sqrt{\text{account}_i}\) and \(w_2 \leq \sqrt{\text{account}_i - w_1^2}\) where \(w_1\) is even and \(w_2\) is odd, and compute \(p = \text{account}_i - w_1^2 - w_2^2\). Now \(p = 1 \mod 4\). Hoping that \(p\) is prime, try to express \(p\) as \(w_3^2 + w_4^2\).

1. Find a solution to the equation \(u^2 = -1 \mod p\). If there is no solution for this \(p\), go back to the beginning of Step 2.
2. Apply the Euclidean algorithm to \((p, u)\) and take the first two remainders that are less than \(\sqrt{p}\) to be \(w_3\) and \(w_4\) (if \(u \leq \sqrt{p}\), use \(u\) as the first remainder).
3. If \(p \neq w_3^2 + w_4^2\) then \(p\) was not prime and thus go back to the beginning of Step 2. If not, return \((w_1, w_2, w_3, w_4)\) as a representation.

**Step 3** If \(t\) is odd but not 1, find a representation for \(2(2^k + 1)\) following Step 2. After that, return \((sw_1, sw_2, sw_3, sw_4)\), where \(s = 2^{(t-1)/2}\).

*In the simple scheme these two commitments are the same.*
4.4. IMPLEMENTATION DETAILS

Step 4 If \( t \) is even find a representation for \( 2(2k+1) \) following Step 2. After that, if \( w_3 \) is even, exchange \( w_2 \) and \( w_3 \). Otherwise exchange \( w_2 \) and \( w_4 \). Then return \((s(w_1+w_2), s(w_1-w_2), s(w_3+w_4), s(w_3-w_4))\), where \( s = 2^{t/2-1} \).

The function compute_representation implements Step 2. It receives as input a number of the form \( 2(2k+1) \) in *number and outputs in *value1, *value2, *value3 and *value4 a representation \((w_1, w_2, w_3, w_4)\).

The function algorithm_Rabin_Shallit, with the same input and output as compute_representation, implements Step 1, Step 3 and Step 4 and calls compute_representation in order to implement Step 2. It is called by the function send_message_BT.

When all the commitments are computed, send_message_BT sends the vendor \( C_{\text{account}} \), \( C_{\text{pa}} \), \( C_{\sigma} \), and the four commitments \((C_{w_1}, C_{w_2}, C_{w_3}, C_{w_4})\). Apart from *result, it returns the values \( account_i \), \( open_{account_i} \), \( p_\sigma \), and \( open_{pa} \) in *new_payment buyer and also the last two values in *item. It also returns the commitments \((C_{w_1}, C_{w_2}, C_{w_3}, C_{w_4})\) in *commitment_FS and the values \((w_1, w_2, w_3, w_4)\) and the random values \((open_{w_1}, open_{w_2}, open_{w_3}, open_{w_4})\) that were used to compute those commitments in *FS.

After that, if *result is not 0 BT returns. Otherwise it runs PK_VT in order to prove that the commitment \( C_{\text{account}} \), was computed correctly and that \( account_i \geq 0 \) and \( account_i = account_{i-1} - p_\sigma \). For this purpose, PK_VT calls send_commitment_PK_BT, which sends the vendor seven commitments as described in the previous subsection. It receives as input the values \((n, g, h)\) and the commitments \((C_{w_1}, C_{w_2}, C_{w_3}, C_{w_4})\) that were output by send_message_BT in *commitment_FS. It outputs in *randomness an array of twelve structures that contains the twelve random values that were used to compute the seven commitments. The first position contains \( r_{w_1} \), the second \( r_{open_{w_1}} \) and the next positions follow the same structure for \( w_2 \), \( w_3 \) and \( w_4 \). The ninth position contains \( r_{account_i} \), the tenth \( r_{open_{account_i}} \), the eleventh \( r_{\alpha} \) and the twelfth \( r_{\beta} \).

Then PK_BT invokes receive_challenge to get the challenge \( c \) from the vendor. Afterwards, it calls send_response_PK_BT in order to send twelve responses of the form \( s = r - cx \), where \( x \) stands for the secret key. It receives as input the challenge \( c \) in *challenge, the array of twelve random values in *randomness, the values \((w_1, w_2, w_3, w_4)\) and \((open_{w_1}, open_{w_2}, open_{w_3}, open_{w_4})\) in *FS, the value \( open_{account_{i-1}} \) in *payment_buyer and the values \( account_i \), \( open_{account_i} \), and \( open_{pa} \) in *new_payment_buyer. The responses are stored in the array of responses that is sent to the vendor in the same position that was used to store the random value corresponding to each response in the array of random values.

Finally, PK_VT runs receive_verification and outputs the result in *result. Then, BT outputs *result.
CHAPTER 4. IMPLEMENTATION OF THE DEMONSTRATOR

4.4.3 Implementations Details of the COT Scheme

In this section we explain the implementation of the COT scheme that is described in Section 3.4. We begin by talking about the data structures and after that we depict the functions that are needed to implement the setup, the initialization and the transfer phase for both the sender and the receiver. Finally, in Section 4.5.2.1 we talk about the security parameters of the scheme and in Section 4.5.2.2 we discuss its efficiency.

The functions of the implementation of the COT scheme are of the same types and follow the same structure as the functions of the OP scheme. The proofs of knowledge that are run in this scheme are depicted in [CNS07].

4.4.3.1 Data Structures

As for the OP scheme, all the data types that are defined for the COT scheme are structures. In the following we show the name of the type and its function:

s_public_key. It contains the values \((g, y, H)\) that are generated during the setup of the COT scheme.

s_secret_key. It contains the value \(h\) such that \(H = e(g, h)\).

s_commitment. It contains the values \((A_p, B_p)\), i.e., the commitments for the \(i\)th item.

s_parameter_RT. It contains the values \((v, V)\) computed by the receiver at the beginning of each transfer phase.

s_parameter_ST. It contains the value \(V\) that the receiver sends to the sender.

s_key. It contains the value \(W\) that the sender sends to the receiver at the end of each transfer phase.

In addition, the structures \(s_{item}, s_{item\_receiver}, s_{commitment\_DF}, s_{commitmentPK}, s_{randomnessPK}, s_{challengePK}\) and \(s_{responsePK}\), which are used in the implementation of the OP scheme (see Section 4.4.2.1), are also used in this scheme. For each value the structure stores an array of type \texttt{unsigned char[]} for the value and the length of the value in a field of type \texttt{unsigned int} (see Section 4.4.2.1).

4.4.3.2 Setup

The setup of the COT scheme includes all the parts of the initialization phase that can be precomputed, i.e., that do not need to be computed anew.
for each initialization phase. It takes place both in the sender’s side and in
the receiver’s side.

The function that is run in this phase in the sender’s side is the Main
Function setup_S. It receives as input the pairing parameters in a file
descriptor stdin and the messages that are offered in a file file_messages
(this parameter indicates the name of the file).

Although the PBC library provides functions that generate pairing pa-
rameters, we note that these functions are different for each type of pairings.
However, the function that initializes pairings works for all the types. There-
fore, in order to build a scheme that can be used with all these types there
were two approaches: one that consisted in implementing all the possible
generations and choose one through a parameter given as input, and the
other that consisted in giving a file with the parameters as input. We chose
the latter, which means that the generation of the parameters (of any type)
should be done before executing the implementation. Another reason is that
the library does not provide a way to export the parameters and, since the
receiver needs to have the same parameters as the sender, the most natu-
ral approach is that the sender publishes the parameters in a file and the
receiver also takes this file as input.

The file with the messages contains in each line the message and the
price. These elements are separated by a blank.

With this input, first setup_S initializes the global variable pairing of
type pairing_t. Then it computes the secret values (x, h) and the parameters
(y, g, H). After that, it reads the file and, for each line, it computes a
pair of commitments (Ap, Bp), where the position of the line in the file
indicates the index i of the commitment, but the price is used to compute
Ap = g^(1/(x+p)). For the computation of Bp, it computes the md5 hash of
e(h,Ap) and after that it xores the hash with the message, according to the
solution explained in Section 3.4. In order to have protection against big
files, the macro size_comm limits the maximum number of lines that can be
read.

Finally, it outputs the number of commitments that have been created in
*number-commitments, an array of commitments of size given by *num-
ber-commitments in *commitments, in which the commitment for the ith
item is stored in the ith position of the array, the parameters (y, g, H) in
*pk and the secret value h in *sk.

In the receiver’s side, the setup consists in running the Main Function
setup_R, which, on input the name of a file that describes the pairing pa-
rameters, initializes the global variable pairing.

---

9Note that this scheme uses symmetric pairings.

10Recall that we need to introduce the price because in the simple OP scheme the item
is identified by its price.
4.4.3.3 Initialization Phase

The implementation of the initialization of the COT scheme contains all the parts of the \((S_I, R_I)\) algorithms that need interaction between sender and receiver. First we describe the functions of the sender and after that we depict the functions of the receiver.

**Sender Initialization Phase.** The functions that are run in this phase in the sender’s side are the Main Function \(S_I\), the I/O Function \(send\_message\_SI\) and the PoK Function \(PK\_SI\), along with the functions that implement this proof of knowledge.

First of all, \(S_I\) runs \(send\_message\_SI\) on input the array of commitments in \(*commitment\), the number of commitments in \(*number\_commitments\) and the parameters \((y, g, H)\) in \(*public\_key\). This function sends the receiver these three sets of values.

After that, \(S_I\) calls \(PK\_SI\) on input \((y, g, H)\) in \(*public\_key\) and \(h\) in \(*secret\_key\) in order to prove knowledge of \(h\). First, \(PK\_SI\) runs \(send\_commitment\_PK\_SI\) to compute a commitment \(a = e(r, g)\) and send it to the verifier. It outputs the random value \(r\) in \(*randomness\).

Upon receiving the challenge \(c\) by means of \(receive\_challenge\), it invokes \(send\_response\_PK\_SI\) on input the challenge \(c\) in \(*challenge\), the random value \(r\) in \(*randomness\) and the secret value \(h\) in \(*secret\_key\). This function computes the response \(z = rh^{-c}\) and sends it to the receiver. Then, \(PK\_SI\) calls \(receive\_verification\) to obtain the result and outputs it in \(*result\). Finally, \(S_I\) also outputs result in \(*result\).

**Receiver Initialization Phase.** The functions that are run in this phase in the receiver’s side are the Main Function \(R_I\), the I/O Function \(receive\_message\_RI\) and the PoK Function \(PK\_RI\), along with the functions that implement this proof of knowledge.

First of all, \(R_I\) runs \(receive\_message\_RI\) in order to obtain an array of commitments of the form \((A_{p_i}, B_{p_i})\) in \(*commitments\), the size of the array in \(*number\_commitments\) and the parameters \((y, g, H)\) in \(*public\_key\).

After that, \(R_I\) calls \(PK\_RI\) to verify that the vendor has knowledge of a value \(h\) such that \(H = e(h, g)\). For this purpose, \(PK\_RI\) first calls \(receive\_commitment\_PK\_RI\) to get the commitment \(a = e(r, g)\) in \(*commitment\). Then it sends the challenge \(c\) by means of \(send\_challenge\), and afterwards it calls \(send\_response\_PK\_RI\) in order to receive and verify the response sent by the sender. This function checks whether \(a = e(z, g)H^{-c}\) and, if it is the case, sets \(*result\) to 0. Otherwise, it sets \(*result\) to a different value.

Finally, \(R_I\) outputs the result in \(*result\), the array of commitments of the form \((A_{p_i}, B_{p_i})\) in \(*commitments\), the size of the array in \(*number\_commitments\) and the parameters \((y, g, H)\) in \(*public\_key\).
4.4. IMPLEMENTATION DETAILS

4.4.3.4 Transfer Phase

The implementation of the transfer phase of the COT scheme contains everything necessary to implement the algorithms \((S_T, R_T)\). First we describe the functions of the sender and after that we depict the functions of the receiver.

**Sender Transfer Phase.** The functions that are run in this phase in the sender’s side are the Main Function \(ST\), the I/O Functions receive_message\_ST and send_message\_ST and the PoK Functions PK\_ST\_V and PK\_ST\_W, along with the functions that implement these proofs of knowledge.

First of all, \(ST\) receives as input the values \((n, g, h)\) in *commitment\_DF*, \((y, g, H)\) in *public\_key*, \(h\) in *secret\_key* and the commitment to the item\(^{[11]}\) \(C_{\sigma_i}\) in *item*. Then it calls receive_message\_ST in order to get the value \(V\) in *parameter\_ST*.

After that, \(ST\) runs PK\_ST\_V with the same inputs as \(ST\) plus parameter\_ST in order to verify that the receiver knows a value \(v\) such that \(V = (A_{p_{\sigma_i}})^v\), that she is requesting an item such that\(^{[12]}\) \(\sigma_i \in \{0, \ldots, N - 1\}\) and that this value is the same as the one used to compute the commitment \(C_{\sigma_i}\). For this purpose, PK\_ST\_V runs receive\_commitment\_PK\_ST\_V in order to get commitments \(a = e(V, g)^{-z_1} e(g, g)^{z_2} e(V, y)^c\) and \(t = g^{z_1} h^{r_3}\) in *commitment1* and *commitment2* respectively.

After sending the challenge \(c\) by means of send\_challenge, PK\_ST\_V invokes receive\_response\_PK\_ST\_V on input \((n, g, h)\) in *commitment\_DF*, \((y, g, H)\) in *public\_key*, \(h\) in *secret\_key*, \(V\) in *parameter\_ST, \(a\) in *commitment1*, \(t\) in *commitment2*, \(c\) in *challenge* and \(C_{\sigma_i}\) in *item*. This function receives the responses \(z_1, z_2, s\) (see the next subsection) and verifies whether \(a \overset{?}{=} e(V, g)^{-z_1} e(g, g)^{z_2} e(V, y)^c\) and whether \(t \overset{?}{=} g^{z_1} h^{r_3}\). If it is the case it sets *result* to 0, and otherwise to another value. It outputs *result*. Finally, PK\_ST\_V runs send\_verification in order to send the result to the receiver and outputs the result in *result*.

Then \(ST\) returns and outputs result in *result* if it is not zero; otherwise it proceeds with the second step. In the latter case, it calls send\_message\_ST on input \(h\) in *secret\_key* and the value \(V\) in *parameter\_ST* in order to compute a value \(W = e(h, V)\) and send it to the receiver. send\_message\_ST outputs \(W\) in *key*.

Afterwards, \(ST\) invokes PK\_ST\_W on input \((y, g, H)\) in *public\_key*, \(h\) in *secret\_key*, \(W\) in *key* and \(V\) in *parameter\_ST*. This function proves that

\(^{[11]}\)We use here \(C_{\sigma_i}\) instead of \(C_{p_{\sigma_i}}\) because the behavior is the same when the price identifies the item and when it is not the case.

\(^{[12]}\)The items are indexed from 0 to N-1 in order to facilitate the access to the arrays of commitments. When using prices instead of selection values, it is verified that the receiver uses a price that the sender utilized to compute the commitments.
Commitment $W$ was correctly computed. For this purpose, it first runs send_commitment_{PK,ST, W} in order to compute two commitments $a_1 = e(r,g)$ and $a_2 = e(r,V)$ and send them in this order to the receiver. It outputs the random value $r$ in *randomness.

After receiving the challenge $c$ by means of receive_challenge, PK,ST, W calls send_response_{PK,ST, W} on input $h$ in *secret_key, the challenge in *challenge and the random value $r$ in *randomness. This function computes the response $z = rh^{-c}$ and sends it to the receiver.

Finally, it receives the result by means of receive_verification and it outputs it in *result. Then ST outputs *result.

**Receiver Transfer Phase.** The functions that are run in this phase in the receiver’s side are the Main Function RT, the I/O Functions send_message_{RT} and receive_message_{RT} and the PoK Functions PK,RT,V and PK,RT,W, along with the functions that implement these proofs of knowledge. The auxiliary function obtain_item is also used.

First of all, RT receives as input in *item_receiver the item $\sigma_i$ and the random value open_{\sigma_i} that were used to compute the commitment $C_{\sigma_i}$ that is used as input of ST. It also receives the values $(y,g,H)$ in *public_key, $(n,g,h)$ in *commitment_DF and the values $(A_{p_{\sigma_i}}, B_{p_{\sigma_i}})$ in *commitment (it does not receive the whole array of commitments).

Then RT calls send_message_{RT} on input $(A_{p_{\sigma_i}}, B_{p_{\sigma_i}})$ in *commitment in order to pick a random $v$, compute the value $V = A_{p_{\sigma_i}}^v$ and send it to the sender. This function outputs $(v,V)$ in *parameter_{RT}.

After that, RT calls PK,RT,V in order to prove that it knows $v$, that the selection value is such that $\sigma_i \in \{0, \ldots, N-1\}$ and that $C_{\sigma_i}$ is a commitment to this selection value. For this purpose, PK,RT,V calls send_commitment_{PK,RT,V} on input $(y,g,H)$ in *public_key, $(n,g,h)$ in *commitment_DF and $(v,V)$ in *parameter_{RT}. This function picks random values $r_1, r_2, r_3$, computes the commitments $a = e(V,g)^{-r_1}e(g,g)^{r_2}$ and $t = g^{r_1}h^{r_3}$ and sends them in this order to the sender. It outputs, $r_1$, $r_2$ and $r_3$ in *randomness1, *randomness2 and *randomness3 respectively.

Upon receiving the challenge $c$ by means of receive_challenge, it invokes send_response_{PK,RT,V} on input the challenge in *challenge, the abovementioned random values, the values $(v,V)$ in *parameter_{RT} and $\sigma_i$ and open_{\sigma_i} in *item_receiver. This function computes the responses $z_1 = r_1 - \sigma_i c$, $z_2 = r_2 - vc$ and $s = r_3 - open_{\sigma_i} c$ and sends them in this order to the sender.

Finally, PK,RT,V receives the result by means of receive_verification and outputs it in *result. Then RT returns if result is not zero. Otherwise it proceeds with the second step.

RT runs receive_message_{RT} in order to receive the value $W$ in *key. After that, it runs PK,RT,W in order to check that the value $W$ was computed correctly. For this purpose, PK,RT,W calls receive_commitment_{PK,RT,W}
in order to get commitments $a_1$ and $a_2$ in commitment1 and commitment2 respectively. Then it invokes send\_challenge to send the challenge $c$.

Afterwards, PK\_RT\_W calls receive\_response\_PK\_RT\_W on input $(y, g, H)$ in *public_key, W in *key, $(v, V)$ in *parameter\_RT and the challenge and the commitments mentioned above. This function receives the response $z$ from the sender and verifies whether $a_1 = e(z, g)H^c$ and $a_2 = e(z, V)W^c$. If it is the case, it sets the result to zero; otherwise it sets the result to another value. It outputs the result in *result.

Then PK\_RT\_W sends the result by means of send\_verification and outputs it. If it is not zero, RT returns and outputs it in *result. Otherwise, it calls obtain\_item, on input $(A_{p_{s_1}}, B_{p_{s_1}})$ in *commitment, $(v, V)$ in *parameter\_RT and W in *key. This function computes the operation $M_{s_1} = B_{p_{s_1}} \oplus \text{hash}$, where hash is the result of applying md5 to $W^{1/v}$. Finally, it outputs $M_{s_1}$ in the string *item and result in *result.

4.5 Efficiency Measurement

4.5.1 Efficiency Measurement for the OP Scheme

4.5.1.1 Security Parameters

There are three security parameters in this scheme: the bit-length $k$ of the special RSA modulus $n$, the bit-length $k'$ of the challenge $c \in \{0, 1\}^{k'}$ that is used in the proofs of knowledge and the bit-length $k''$ that is used to define the interval from which the random values $r \in ] - n^{2^{-2+k'+k''}} .. n^{2^{-2+k'+k''}} [$ that are picked for running proofs of knowledge are taken.

These values are defined by three macros, leng\_rsa, leng\_challenge and param\_indistin, in the file “types.h”. The current values of these macros are 1024, 80 and 80 respectively.

4.5.1.2 Efficiency

The most resource-intensive functions of the OP scheme are setup\_commit\_DF, which is run in the setup of the vendor, and algorithm\_Rabin\_Shallit, which is run in the transfer phase in the buyer’s side. Therefore, we have measured the efficiency of these functions by calculating the average running time after twenty executions.

The running time of setup\_commit\_DF is probabilistic and depends on the value of the macro leng\_rsa, which gives the bit-length of the special RSA modulus $n$. The average running time and the variance are depicted in Table 4.1 in which we can see that the running time of this function grows polynomially with the length of the security parameter.

The running time of algorithm\_Rabin\_Shallit is also probabilistic and depends on the size of the number for which a representation needs to be
computed. In our case this number is the product of the account of the buyer in each phase and the bound used to build a scheme in which different items can have the same price. This bound is given by the macro scale. In Table 4.2 we show the average running time and the variance for numbers whose order of magnitude ranges from 3 to 9.

As can be seen, the variance of the running time of these functions is big. This is specially undesirable for algorithm_Rabin_Shallit, because this function is run in every transfer phase. Therefore it is advisable not to multiply the prices by a large scale, although this would mean that prices that in theory are the same in practice are a little bit different.

### 4.5.2 Efficiency Measurements for the COT Scheme

#### 4.5.2.1 Security Parameters

In order to implement pairings, the PBC library offers several pairing types that have different security and efficiency properties. We have chosen the Type A pairing because this is the most efficient one.

The PBC library has one function that generates Type A pairings. As input of this function it is necessary to specify the bit-length of the group order in \( rbits \) and the bit-length of the base field order in \( qbits \).

Therefore, we have to select these bit-lengths in order to ensure the
4.5. EFFICIENCY MEASUREMENT

<table>
<thead>
<tr>
<th>Number of items</th>
<th>Security Level</th>
<th>$i(s)$</th>
<th>$S^2(s^2)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>AES-80</td>
<td>6.212601</td>
<td>0.007440</td>
</tr>
<tr>
<td>100</td>
<td>AES-128</td>
<td>73.409019</td>
<td>0.224027</td>
</tr>
<tr>
<td>200</td>
<td>AES-80</td>
<td>12.320795</td>
<td>0.003043</td>
</tr>
<tr>
<td>200</td>
<td>AES-128</td>
<td>145.644363</td>
<td>0.260003</td>
</tr>
</tbody>
</table>

Table 4.3: Running time of setup$_S$

security of the scheme. According to [KM05], to achieve a level of security equivalent to AES-80 we need that $r bits = 160$ and $q bits = 1024$, and to achieve a level equivalent to AES-128 we need that $r bits = 256$ and $q bits = 3072$.

By means of the application TypeAPairingGeneration it is possible to obtain a file that specifies new pairing parameters for the bit-lengths $r bits$ and $q bits$ that are given as input. In addition, the files paramsOT80 and paramsOT128 contain Type A pairing parameters that provide a level of security equivalent to AES-80 and AES-128 respectively.

4.5.2.2 Efficiency

The most resource-intensive function of the COT scheme is setup$_S$. We have measured its efficiency by calculating the average running time after twenty executions.

The running time of setup$_S$ is almost deterministic and depends on the number of items that the vendor wants to sell and on the parameters that are used to compute pairings. The average running time and the variance are depicted in Table 4.3, in which we can see that the running time of this function grows linearly with the number of commitments, and that the pairing parameters that are needed to achieve a security level equivalent to AES-128 lead to a pairing computation time far larger than the ones that offer a security level equivalent to AES-80 (see the previous subsection).
Chapter 5

Legal Aspects
5.1 Introduction

The preceding sections described in detail the cryptographic construction and implementation of a priced oblivious transfer (POT) scheme. While this protocol clearly has the potential of offering enhanced privacy properties, at the same time it raises questions as to whether or not it may hinder compliance in other areas of law. The purpose of this chapter is to investigate the extent to which the proposed scheme raises issues of compliance in areas of law other than data protection, such as e-commerce regulations or tax law.

For purposes of clarity, we reiterate the main qualities of the POT scheme:

1. The vendor does not learn which item was requested/purchased by the buyer;
2. The vendor does not learn the exact price of the item that was requested/purchased by the buyer;
3. The vendor only learns that a customer has in fact purchased a particular item, and that the appropriate amount was paid;
4. The buyer only obtains access to those items he or she has paid for (no more, no less).

In short, the vendor does not learn any information with regards to the individual items that are requested by a specific buyer. The vendor only knows that the buyer has placed an order, and that he (the vendor) has received appropriate payment.

The legal analysis in this chapter shall be structured as follows. First, we will analyze the potential legal obstacles emanating from consumer information obligations, such as those stipulated in the e-Commerce Directive. Secondly, we will look at the question of fiscal compliance, in particular with regards to value added tax (VAT) legislation. In the third instance, we will look at potential ‘misbehavior scenarios’ to determine whether the proposed scheme provides a sufficient ‘audit trail’ to safeguard the rights and interests of both parties in the event of a dispute.

To help guide our legal analysis, we have defined a more concrete setting in which the priced oblivious transfer scheme might be implemented. Our further investigation is premised upon the following elements:
- the ‘vendor’ is a service provider who operates a website offering newspaper and magazine articles in pdf format;
- the vendor only offers these digital items to private consumers (B2C transactions);
- the vendor is established in Belgium;
- the customers of the vendor have to purchase a certain amount of funds (‘credits’) before they are able to place an order for any particular item and
- once a buyer has acquired these funds, the credits can only be spent by placing orders for particular items from the vendor’s catalogue.
5.2 Consumer information obligations

Obligations to provide consumers with particular information have become an increasingly central element of European consumer protection legislation.\(^1\) Multiple European Directives have introduced informational obligations with the aim of augmenting and harmonizing the level of consumer protection within the internal market.\(^2\) The scope of the information obligation, the amount of information and the moment at which it must be provided, varies among the different regulatory instruments. Generally speaking, two main categories information obligations can be distinguished: pre-contractual and contractual information obligations.\(^3\) Our interest primarily goes to the second category of information obligations. These information obligations require the service provider to provide consumers with a confirmation, typically in writing or on a durable medium, concerning certain details of their transaction.\(^4\) Seeing as the ‘vendor’ in a priced oblivious transfer scheme does not learn any information about the order that is placed by the consumer, the question arises whether or not the former will be able to comply with these information obligations. In the following subsections, we will look at the information obligations of two regulatory instruments, namely the E-Commerce Directive and the Distance Contract Directive, to assist in making this determination.

5.2.1 E-Commerce Directive

Directive 2000/31/EC\(^5\) was adopted with the aim of establishing a clear and general framework for certain legal aspects of electronic commerce within the internal market.\(^6\) This Directive, commonly referred to as the ‘E-Commerce Directive’, has mainly two objectives. In the first instance, it sought to remove certain legal obstacles which were seen as hampering the development of electronic commerce within internal market.\(^7\) In the second instance, it simultaneously aimed to provide legal certainty and ensure consumer confidence towards electronic commerce.\(^8\) The following aspects of electronic commerce have been regulated by Directive 2000/31/EC\(^9\):
- information requirements;
- on-line advertising;
- electronic contracting and
- liability of intermediaries.

---

2 Ibid, p. 102.
3 Ibid, p. 102.
6 Recital (7) of Directive 2000/31/EC.
7 Recital (5) of Directive 2000/31/EC.
8 Recital (7) of Directive 2000/31/EC.
The E-Commerce Directive governs the provisioning of ‘information society services’ within the internal market.\(^{10}\) Information society services are defined as ‘\textit{any service normally provided for remuneration, at a distance, by electronic means and at the individual request of a recipient of services}’.\(^{11}\) Information society services span a wide range of economic activities that take place on-line: selling goods on-line, offering on-line information or commercial communications, providing tools allowing for search, access and retrieval of data … .\(^{12}\) A ‘vendor’ who offers pdf articles against remuneration on-line, such as the one described in our use case, can be qualified as a provider of information society services.\(^{13}\)

Directive 2000/31/EC contains several information obligations for providers of information society services:
- a general information obligation (art. 5);
- information to be provided in commercial communications (art. 6);
- information to be provided in relation to the conclusion of the contract as well as contract terms and conditions and
- information obligations relating to order placement (art. 11).

It is particularly the last set of information obligations that is relevant to our current analysis. Art. 11 of the E-Commerce Directive stipulates that where the recipient of the service places his order through technological means, the service provider has to acknowledge the receipt of the recipient’s order without undue delay and by electronic means. This provision appears as if it might present an obstacle for the implementation of a priced oblivious transfer scheme. If the vendor does not learn any information about the items that the buyer ordered, one would assume that he will be unable to provide the buyer with an acknowledgement of his order, and thus be unable to comply with the requirement of art. 11. Over the following paragraphs, we will further assess the extent to which this requirement may form a legal barrier towards the implementation of a priced oblivious transfer scheme.

It is noteworthy that recital (34) of Directive 2000/31/EC also specifies that ‘the acknowledgement of receipt by a service provider may take the form of the on-line provision of the service paid for’. This phrase in the recital suggests that the requirement for an acknowledgement of receipt of the order contained in art. 11 can be interpreted with some flexibility. One could argue that if service provisioning occurs immediately after the order has been placed, then this provisioning (fulfilling of the order) provides implicit (yet sufficient) acknowledgement of receipt of the order. However, when reading recital (34) in its entirety and in combination with the subsequent recitals, it would appear that this phrase merely provides a clarifying statement as to the flexibility Member States have in implementing this particular aspect of the Directive. This reading implies that Member States would actually need to incorporate a provision to such an extent in their

\(^{10}\) Art. 1 Directive 2000/31/EC.
\(^{11}\) See art. 2 Directive 2000/31/EC; referring to Directive 98/34/EC as amended by Directive 98/48/EC.
\(^{12}\) Recital (18) Directive 2000/31/EC.
\(^{13}\) For a detailed analysis of the constitutive elements of information society services see M. Schaub, o.c., p. 27-31.
national legislation when implementing the Directive, or that the national implementation should at a minimum allow for such an interpretation. In absence of such a provision, information society service providers would still be required to send the recipient of the service an explicit confirmation of their order, even where service provisioning takes place on-line immediately after the order has been placed.

The Belgian Law implementing the E-Commerce Directive does not contain any reference to the fact that the acknowledgement of receipt of an order might take place implicitly through on-line provision of the requested service. Art. 10 only states that ‘where the recipient of an information society service places an order by electronic means, the following principles must be taken into account:

1° the service provider shall confirm as soon as possible, by electronic means, the receipt of the order of the recipient;
2° the proof of receipt must mention, among other things, a summary of the order that was placed;
3° the order and the proof of receipt shall be deemed to be received when the parties to whom they are addressed are able to access them.’

This implementation does not seem to allow for an interpretation which suggests that explicit confirmation is no longer deemed necessary where service provisioning takes place on-line immediately after the order has been placed, particularly as it requires the service provider to send or make a available a ‘proof’ of receipt of an order. In this regard, it is interesting to note the subtle linguistic differences which exist among the English and Dutch version of the Directive. While the English version of the E-Commerce Directive mentions ‘an acknowledgement of receipt’, the Dutch version makes reference to a ‘proof of receipt’ (‘ontvangstbewijs’). Similar to the Dutch version, the French text makes reference to ‘l’accusé de réception’, which means ‘proof of receipt’. The reason these subtle textual difference have practical relevance is the following. In the event that the information society service provider merely needs to ‘acknowledge’ the receipt of the order by electronic means, one could argue that it would be sufficient to merely display such confirmation in a web page (e.g., as a final message after the customer has made his order final). On the other hand, if the provider of the information society service is to provide the customer with an actual ‘proof of receipt’, this language would suggest that this confirmation should be delivered or made available in way that more readily ensures a certain durability (e.g., a separate email containing a pdf attachment with confirmation). From a consumer protection perspective, it seems preferable to require service providers to provide their customers with a durable confirmation of their orders, so that the confirmation might assist the consumer in case of

14 This interpretation would of course also entail that a pre-existing provision to that extent would not be seen as conflicting with the Directive.
15 Law of 11 March 2003 concerning certain legal aspects of information society services, B.S. 17 March 2003. The reader should note that it does not automatically follow that because the service provider is established in Belgium, as in our scenario, that Belgian Law shall be applicable to every consumer contract, even if the consumer resides outside of Belgium. This is a matter of international private law which is beyond the scope of our current analysis. For more information see e.g. L.E. Gillies, Electronic Commerce and International Private Law: A Study of Electronic Consumer Contracts, Burlington (England), 2008, 261p.
a dispute (e.g., in case the product or service turned out to be not what was actually ordered, or if the items received were defective). Because the confirmation provided on a webpage is ephemeral in nature, it is not considered to provide the same benefit, from a consumer protection perspective, as a durable confirmation.\footnote{See also infra the discussion of recital (13) of Directive 97/7/EC (section 2.2).}

For purposes of completeness, we note that the second paragraph of art. 11 also requires that the service provider make available to the recipient of the service ‘appropriate, effective and accessible technical means allowing him to identify and correct input errors, prior to the placing of the order’. This requirement entails that the interface to the buyer must allow him to review the items he has selected for purchase prior to actual order placement.

5.2.2 Distance Contract Directive

The Distance Contract Directive\footnote{Directive 97/7/EC of the European Parliament and of the Council of 20 May 1997 on the protection of consumers in respect of distance contracts, O.J. 4 June 1997, L 144/19–27. This Directive is often also referred to as the ‘Distant Selling Directive’.} covers contracts concerning goods or services concluded at a distance between a consumer and a supplier.\footnote{M. Schaub, o.c., p. 79.} Although this Directive does not specifically target e-commerce, it is, as a rule, integrally applicable to B2C e-commerce contracts, seeing as e-commerce most often involves the conclusion of a contract at a distance.\footnote{Ibid, p. 79.}

Art. 2 of Directive 97/7/EC defines a ‘distance contract’ as ‘any contract concerning goods or services concluded between a supplier and a consumer under an organized distance sales or service-provision scheme run by the supplier, who, for the purpose of the contract, makes exclusive use of one or more means of distance communication up to and including the moment at which the contract is concluded’. The contract that is concluded between the ‘vendor’ and the ‘buyer’, such as the one described in our use case, qualifies as a distance contract for purposes of Directive 97/7/EC.

Art. 4 of the Distance Contract Directive specifies that, prior to the conclusion of any distance contract, the consumer shall be provided with the following information:

(a) the identity of the supplier and, in the case of contracts requiring payment in advance, his address;
(b) the main characteristics of the goods or services;
(c) the price of the goods or services including all taxes;
(d) delivery costs, where appropriate;
(e) the arrangements for payment, delivery or performance;
(f) the existence of a right of withdrawal, except in the cases referred to in Article 6 (3);
(g) the cost of using the means of distance communication, where it is calculated other than at the basic rate;
(h) the period for which the offer or the price remains valid and
where appropriate, the minimum duration of the contract in the case of contracts for 
the supply of products or services to be performed permanently or recurrently.

This prior notice obligation is supplemented by the obligation to provide the consumer with confirmation of the information referred to in art. 4 (1) (a) to (f). Article 5, 1 of Directive 99/7/EC specifies that the consumer must receive this confirmation in writing or on another durable medium, and ‘at the latest at the time of delivery […] unless the information has already been given to the consumer prior to conclusion of the contract in writing or on another durable medium available and accessible to him’. These obligations appear to present a similar obstacle for the implementation of a priced oblivious transfer scheme as the information obligations under the E-Commerce Directive. The requirement of confirmation of ‘the main characteristics of the goods or services’ could be read as requiring the supplier to provide written confirmation (or confirmation in another durable medium) of the exact products that were ordered. Due to the specification that the confirmation must be ‘provided in a written or in another durable medium’, Directive 99/7/EC in fact requires a confirmation in an instrument which the customer can store in a way that makes it accessible for future reference.\(^{20}\) In this regard, recital (13) of Directive 97/7/EC specifies that ‘Whereas information disseminated by certain electronic technologies is often ephemeral in nature insofar as it is not received on a permanent medium; whereas the consumer must therefore receive written notice in good time of the information necessary for proper performance of the contract.’ This recital additionally suggests that one of the objectives of the confirmation requirement may be to provide the consumer with a starting point to substantiate later claims in case the supplier does not adequately perform his obligations under the distance contract (e.g., non-conforming delivery).\(^{21}\)

Alternatively, one could argue that the distance contract between the supplier and the consumer is concluded at the moment where the buyer consents to the general terms and conditions of the supplier and makes his initial purchase of funds (credits). The actual delivery of pdf articles pursuant to later requests of the buyer could be seen as merely constituting the subsequent performance of the initial distance contract between the vendor and the buyer. Under this approach, one could argue that the supplier would satisfy articles 4 and 5 by providing the buyer with confirmation of their initial contract, a generic description of the main characteristics of the items contained in the supplier’s catalogue, the arrangements for payment, delivery, performance, etc. However, seeing as the initial contract will not contain any details about the exact items the supplier is to provide to the buyer at a later time, it is more likely that the subsequent order and delivery of individual items must be qualified as separate distance contracts. The initial agreement, which is concluded at the moment that the buyer makes a purchase of credits, can be seen as a type of ‘framework agreement’ that stipulates the terms and conditions


\(^{21}\) Compare also *supra*; section 2.1 (when discussing the obligation of the service provider to acknowledge the recipients order under the E-Commerce Directive). See also M. Demoulin, ‘La notion de “support durable” dans les contrats à distance: une contrefaçon de l’écrit?’, *Revue européenne de droit de la consommation*, 2000, vol. 4, p. 366.
which shall govern the later contracts which are concluded at the moment of order placements and fulfillment.\textsuperscript{22}

\section*{5.2.3 Conclusion}

The requirement to acknowledge the receipt of the customer’s order as contained in the E-Commerce Directive will in many cases present an obstacle to the implementation of a priced oblivious transfer scheme, if the scheme cannot be extended to provide the consumer with a ‘durable confirmation’ of his order. Only where the applicable legislation stipulates that the confirmation may take the form of on-line provision of the order, will this not be the case. The service provider operating the website must in any event ensure compliance with the other information requirements contained in this Directive, such as the obligation to provide the consumer with appropriate, effective and accessible technical means which allow him to identify and correct input errors prior to order placement.

Whether the confirmation obligation contained in art. 5 of the Distance Contract Directive presents a legal obstacle for the implementation of the priced oblivious transfer scheme depends largely on whether the orders for particular pdf articles, which take place after the initial purchase of funds by the customer, are also considered to be distance contracts by themselves. We have argued that because the initial contract between buyer and vendor will not contain any details as to the exact items to be provided, it is more likely that the subsequent order and delivery of individual items must be seen as separate distance contracts. This entails the requirement of separate confirmation, in writing or another durable medium, of the order that was placed.

From a consumer protection perspective, it appears desirable to require suppliers to provide consumers with a durable confirmation of individual orders to assist the consumer in case of dispute (see also \textit{infra}; Section 5.4).

\section*{5.3 Requirements under VAT legislation}

Value added tax is a consumption tax which is levied based on the value that is added to a particular product or in the performance of a particular service. In principle, the financial burden of the tax is borne by the final consumers of the product or service, as the tax is continuously ‘carried over’ throughout the production- and distribution chain, until it finds its ‘final resting place’ with the consumer.\textsuperscript{23} As we described earlier, the priced oblivious transfer scheme enhances consumer privacy by ensuring that the ‘vendor’ does not learn which items its customers order. This entails that it is impossible for the former


to maintain records about how the pre-paid credits of its customers have been spent or how many times a particular item has been downloaded.

The purpose of this section is to analyze whether the scenario outlined in the introduction poses any specific legal issues with regards to VAT legislation. A preliminary remark which needs to be made is that for fiscal purposes, the provisioning of digital content against remuneration is not considered to be a ‘sale of goods’, but instead is qualified as an ‘electronically supplied service’. This distinction is important because different VAT rules apply depending on whether the object of the transaction is qualified as being either a ‘good’ or a ‘service’. While it may seem counter-intuitive, this means that products will be treated differently depending on whether they are being delivered in a digital or non-digital format. For example, when a printed copy of a magazine is purchased at the newspaper shop around the corner, this transaction shall be qualified as a sale of goods. However, when this same magazine is downloaded in pdf format, even though it has exactly the same content, this transaction shall be treated as an (electronically supplied) service under VAT legislation. For this reason, we shall from here on refer to the entity offering the digital content as the ‘service provider’ (rather than the ‘vendor’).

Our analysis shall be structured as follows. First, we will look at the scope of the invoice obligation and at the level of detail that is required of invoices. Secondly, we will look at the mandatory elements of the periodic VAT return. Thirdly, we will analyze the moment of chargeability of tax and how the basis of taxation is determined. Finally, we will look at the main accounting and registration obligations to make a determination as to whether these requirements stand in the way of the implementation of a priced oblivious transfer scheme.

Although the legal framework concerning VAT has been harmonized to a large extent by European Directives, there are still certain aspects which may be specific to individual Member States. Seeing as our use case concerns a service provider which is established in Belgium, our legal analysis will be supplemented by reference to Belgian legislation.

5.3.1 Invoice obligation

In principle, every vendor or service provider is required to draft an invoice for every transaction which gives rise to VAT. Such an invoice must typically contain at least the following elements:

26 With regard to the place of taxation of electronically supplied services see also D.M. Parrilli, l.c., p. 8 et seq.
- the date upon which it was issued as well as the sequence number in the book of outgoing invoices;
- identification of the taxable entity (name, address and VAT-number);
- identification of the co-contractor (name, address and VAT-number);
- the information that is necessary to determine the taxable act and the applicable VAT tariff for the provided goods or services (in particular their common name, quantity and object of the supplied goods or services). 28

The last element appears to be prohibitive towards the offering of services under a priced oblivious transfer scheme, seeing as VAT legislation requires service providers to draft an invoice which specifies the name, objects and quantity of the delivered services. An important exemption exists, however, for B2C transactions: a taxable entity generally does not need to issue an invoice when it provides services or goods to natural persons who intend them for private use. 29 Since the use case described in the introduction specified that the newspaper or journal articles are only offered to consumers who intend them for private use, the invoice obligation does not apply here.

5.3.2 Elements of the periodic VAT return

In principle, every service provider taxable under Belgian VAT legislation must file a periodic VAT return on either a monthly or quarterly basis. 30 The main elements of this periodic declaration are:
- sums received for the services rendered;
- the applicable VAT rate(s) (6-12-21%);
- deductions or revisions which the entity submitting the declaration may claim and
- the total of sum of VAT which is due. 31

When reviewing these elements, it would appear that this formality does not by itself impede implementation of a priced oblivious transfer scheme, provided of course that all of the offered services are subject to the same rate. If the offered services are subject to different rates, this would create a need for clear differentiation as to the items requested by consumers, at least as to allow identification of the applicable rate.

5.3.3 Chargeability of tax and the taxable amount

Two additional elements are important for our analysis: the taxable amount and the moment of chargeability of the VAT. The taxable amount refers to the amount of income that is taxable (also referred to as the ‘basis of taxation’). For purposes of Belgian VAT,

28 See Art. 5 of Royal Decree no. 1 of 29 December 1992 concerning the settlement of value added tax [Koninklijk besluit met betrekking tot de regeling voor de voldoening van de belasting over de toegevoegde waarde], B.S. 31 December 1992 (hereafter: ‘Royal Decree no. 1’).
30 See art. 53 Belgian VAT Code. A Royal Decree may specify instances in which the periodicity of the invoice obligation shall be extended to six months or a year (see art. 53(8) Belgian VAT Code).
31 See art. 53, § 1, 2° Belgian VAT Code. For the model form for electronic VAT returns see http://fiscus.fgov.be/interfaoifnl/Btw-aangifte/inleiding.htm
the taxable amount in respect to services consists of the consideration provided by the consumer in order to receive the service.\textsuperscript{32} Applied to our current use case, this would mean that in principle the basis of taxation for a particular order is determined by the amount of credits the consumer spends to obtain this item.

Equally relevant to our analysis, however, is the moment of chargeability of the tax, i.e. the moment upon which the VAT becomes due.\textsuperscript{33} In principle, VAT only becomes due at the moment that the chargeable event (taxable act) is completed, in other words when the goods are delivered or the services are performed.\textsuperscript{34} Where the service provider issues an invoice or receives payment prior to the completion of the service, the VAT may become due at an earlier time. The Belgian VAT Code specifies that for service providers who benefit from an exemption to the invoice obligation (cf. supra; 3.1), the VAT shall in fact become due at the moment of the receipt of any payment, regardless of whether such payment occurs prior to or after the moment that the service is or has been provided.\textsuperscript{35}

Under the priced oblivious transfer scheme described earlier in this deliverable, the service provider does not learn how many credits its customer will be spending when placing a request for an item, nor the amount of the remaining funds. This could present an issue because the service provider is thereby unable to calculate the VAT for a particular transaction because the basis of taxation is ordinarily determined by the amount of credits that are being spent (cf. supra). However, because the scheme also requires all customers to purchase credits (‘funds’) in advance in order to obtain items, the situation may be different. One could argue that the purchase of these credits in fact amounts to an advanced payment for the service, which would entail that the VAT would become due as of that moment. In other words, the VAT would have to be calculated at the moment that the credits are purchased (and on that amount), rather than at the moment of expenditure. In the following paragraphs, we shall analyze two decisions which help clarify the issue of the moment chargeability of tax and the application of VAT to prepaid credits.

In 2004, the AOIF (‘Adminstratie van Ondernemings- en Inkomensfiscaliteit’) issued a circular concerning the VAT system that governs prepaid phone cards.\textsuperscript{36} Initially, the purchase of these cards was considered to be an advanced payment towards the provisioning of a telecommunications service. Consequently, VAT became due at the

\textsuperscript{32} Art. 26 Belgian VAT Code. See also R. Wuytjens, ‘Maatstaf van Heffing’, in P. Vandendriessche and W. Devroe (ed.), Handboek Belasting Toegevoegde Waarde, Brugge, De Keure, 2007, 175 et seq.. Art. 11 of (Sixth) Council Directive 77/388/EEC similarly provides that the taxable amount for services shall be “everything which constitutes the consideration which has been or is to be obtained by the supplier from […] the customer or a third party for such supplies”.

\textsuperscript{33} Art. 10, 1 (b) of Directive 77/388/EEC stipulates that VAT is considered ‘chargeable’ “when the tax authority becomes entitled under the law at a given moment to claim the tax from the person liable to pay, notwithstanding that the time of payment may be deferred.”

\textsuperscript{34} Art. 22, § 2 Belgian VAT Code (art. 10, 2 Directive 77/388/EEC).


The moment of sale of such cards pursuant to art. 22, § 2 of the Belgian VAT Code. The actual expenditure of the credits that these phone cards represented was not taxed, seeing as the taxation had already taken place at the moment of their sale. However, over time the commercial offering in the telecommunications sector has become far more extensive than that of phone services alone (e.g. games, logos, ringtones, competitions). Due to the fact that the majority of prepaid cards can now be used to acquire an increasing variety of services, and sometimes even to obtain tangible goods, the AOIF held that they were to be considered equivalent to an instrument of electronic payment as of January 2005.

Because the purchase of prepaid phone cards is no longer seen as falling within the scope of art. 22, § 2 Belgian VAT Code, the VAT does not become due until the moment of expenditure. The basis of taxation in this case is equal to the value of the received service, which is determined by the amount of credits that is deducted from the prepaid card.

The issue of the moment chargeability of tax and the application of VAT on prepayments was also addressed by the European Court of Justice (ECJ) in BUPA Hospitals. In this case, the Court considered that in order for the VAT to become chargeable to payments on account, "all the relevant information concerning the chargeable event, namely the future delivery or future performance, must already be known and therefore, in particular, [...] when payment on account is made the goods or services must be precisely identified."

The ECJ continued its reasoning by reiterating that it is the supply of services which are subject to VAT, rather than the payments which are made by way of consideration for such supplies. Therefore, according to the Court, "payments on account of supplies of goods or services that have not yet been clearly identified cannot be subject to VAT." It concluded that "[...] prepayments of the kind at issue in the main proceedings whereby lump sums are paid for goods referred to in general terms in a list which may be altered at any time by agreement between the buyer and the seller and from which the buyer may possibly select articles, on the basis of an agreement which he may unilaterally rescile from at any time, thereupon recovering the unused balance of the prepayments, do not fall within the scope of the second subparagraph of Article 10(2) of Sixth Council Directive 77/388/EEC [...]"

Both the circular of the AOIF and the BUPA decision provide useful guidance for determining whether the purchase of credits by consumers in our use case should be qualified as an advanced payment (‘payment on account’) for VAT purposes.

---

37 Ibid, consideration 2-4.
38 Ibid, consideration 7-9.
40 European Court of Justice, BUPA Hospitals Ltd, Goldsborough Developments Ltd v. Commissioners of Customs & Excise, C-419/02 available at http://curia.europa.eu.
41 ECJ, C-419/02, consideration 48.
42 Ibid, consideration 50.
When examining the reasoning of the AOIF, one could infer that the determining factor in deciding that the purchase of prepaid phone cards should no longer be seen as an ‘advanced payment’, was because the credits they represent could be spent in a number of ways. To put it differently, because the services that could be obtained in exchange for use of the card’s value were no longer limited to one or more clearly defined services, the phone card had come to resemble a generic instrument of payment, which, as a rule, does not give rise to VAT. The extent to which the manners of expenditure are defined or restricted, therefore, seems to be key in deciding whether the acquisition of credits precipitates the chargeability of VAT.

In *BUPA*, the ECJ also placed apparent emphasis on the fact that the goods and services in dispute were not clearly defined. The critical factors were that (1) the prepayments related to goods that were ‘referred to in general terms in a list’ could be altered upon mutual agreement, and (2) the fact that the buyer retained the ability to unilaterally withdraw and recover the unused balance of the prepayments.

Applying these factors to our use case, it would seem that the chargeability of tax will, in the first instance, be determined by the variety of (electronically supplied) services that are being offered under the priced oblivious transfer scheme. The wider the range of ESS being offered, the less future performance shall be considered ‘clearly defined’. The terms and conditions of the agreement between the customer and service provider will also be determinative in ascertaining when the VAT becomes chargeable. If, upon cancellation, the customer is entitled to reimbursement for the credits he has previously purchased, chargeability of VAT shall likely be deferred until the moment of expenditure. On the other hand, if the purchased credits are not reimbursed upon cancellation and the variety of ESS being offered is limited, one might still argue that chargeability arises at the moment of purchase of credits.

### 5.3.4 Accounting and registration obligations

Every service provider subject to VAT must keep accounts in sufficient detail to permit application of the value added tax and inspection by the tax authority (requirement of verifiability).

Under the Belgian VAT Code, the accounting system of the taxable person must comprise the following three components:

- a book of incoming invoices;
- a book of outgoing invoices and
- a journal of receipts.

It is particularly the last component which is relevant to our further analysis. The journal of receipts details the earnings received by the taxable entity for which it is not required

43 See art. 22, § 3, 7°-9° Belgian VAT Code. See also K. Martens, ‘Waardebbonen, geschenkbonnen, betaalbonnen, … Wat met de btw?’, *Fiscale Wenken*, 2009, n.09/17-01, available at [www.monKEY.be](http://www.monKEY.be) (discussing the VAT scheme applicable to store gift certificates).


45 Art. 14, § 2 Royal Decree no. 1.
to draft an invoice (and has not done so). On a daily basis, the total amount of earnings must be registered in this journal (per tariff). The entries in the journal of receipts must be supported by documents in proof (e.g. order forms, cash register receipts, VAT-voucher), each of which must be dated.

The taxable entity is obliged to organize its accounting system in a manner that is compliant with the applicable legislation, which includes, but is not limited to, the Law of 17 July 1975 concerning the accountancy and annual financial statements of enterprises. This law provides that ‘every accounting system must be maintained by means of a system of books and accounts which are kept in accordance with the customary rules of bookkeeping by double entry’.

The accounting and registration obligations described in the previous paragraphs create certain obstacles for the implementation of a priced oblivious transfer scheme. Although the journal of receipts must only be updated on a daily basis, the service provider will be unable to provide an overview of the actual receipts in credits for each day. If the fiscal authorities accept that the chargeability of tax and the basis of taxation are determined by the purchase of credits rather than their expenditure, this could alleviate the problem to some extent. Even under a priced oblivious transfer scheme, the service provider should not have any difficulties in maintaining detailed records as to the credits (funds) that are purchased by its customers. However, it could also be argued that, independent of the moment of chargeability and the basis of taxation, the requirement of verifiability (cf. supra) demands that the service provider maintains documents in proof not only with regards to the purchase of funds, but also as to how the credits are actually spent. After all, absence of such records would substantially reduce the means with which the tax authorities might verify whether the reported purchases of funds correspond with the amount and value of the services which were actually delivered. In addition, one might argue that a ‘regular’ system of accounts, which abides by the customary rules of bookkeeping by double entry, would detail both the purchase of the credits as well as their expenditure.

5.3.5 Conclusion

The preceding sections have shown that current VAT legislation is likely to pose a legal barrier to the implementation of a priced oblivious transfer scheme in the scenario

---

46 Art. 14, § 2, 3° Royal Decree no. 1. In principle the taxable entity must maintain a journal of receipts for every place of business that is part of its operation.
47 Transactions exceeding the value of 250 Euros must be listed separately, unless document in support is sufficiently detailed as to both the price and the type of the supplied goods. Art. 15, § 4 Royal Decree no. 1.
48 Documents in support must be kept for a minimum of 5 years depending on the case. See art. 15, § 2 Royal Decree no. 1 and art. 60, § 1 and 4 of the VAT Code. See also Administrative Decision E.T. 112.577 of 8 November 2007 concerning the maintenance of the journal of receipts and the journal for centralisation by means of an ICT system, consideration 9.
49 B.S. 4 September 1975. Legal accounting obligations can for instance also be found in corporate law and legislation on direct taxation.
50 Art. 4 of the Law of 17 July 1975 concerning the accountancy and annual financial statements of enterprises.
described at the beginning of this section. Particularly the provisions relating to chargeability of tax, the basis of taxation, and accounting and registration requirements seem to preclude a scheme which obfuscates the details of individual transactions between service providers and their customers. Because the application of these provisions can still be debated (e.g., with regard to the moment of chargeability and the level of detail required under the principle of verifiability), a service provider considering implementation of such a scheme should in any event request a ruling from the tax authorities prior to deployment.

5.4 Audit trail

When vendors or service providers engage in commerce, there are a number of legal provisions that will govern the contractual relationship between them and their customers. In principle, the parties to a contract are free to determine the content of their agreement and the legal effects it will create. However, in absence of clear contractual specifications to the contrary, the respective rights and obligations of the parties are determined mainly by the principles of contract law and the provisions of the Civil Code which relate to the type of contract in dispute. In some cases, there may also be mandatory legislation that can restrict the contractual freedom of the parties (e.g., in the case of consumer contracts).

In the event of a dispute between the vendor and the buyer, it is, in principle, the plaintiff who bears the burden of substantiating his or her claim (actori incumbit probatio). However, the defendant who subsequently invokes a particular exception or objection is in turn obligated to provide the evidence thereof (reus excipiendo fit actor). For example, in case of non-payment, it will, in principle, be the vendor who will have to demonstrate that there is an obligation upon the buyer to do so. In case the buyer claims to have paid, or invokes another exception, he will have to provide the evidence to substantiate these exceptions.

51 An important preliminary matter in this regard concerns the question of the qualification of the contract. In our use case, the object of the contract is the provisioning of digital content in return for remuneration. In the previous section, we have elaborated that, for purposes of fiscal legislation, this transaction is qualified as a ‘service’ rather than as a ‘sale of goods’ (cf. supra; section 3). As for other areas of law, there is still a certain degree of debate and legal uncertainty. Several authors argue that it is unfair to treat what is essentially the same item differently, depending on the way it is formatted or delivered. This argument is regularly made in relation the consumer protection afforded by Directive 1999/44EC of 25 May 1999 on certain aspects of the sale of consumer goods and associated guarantees (O.J., 7 July 1999, L 171/12-16), where the scope is clearly limited to tangible movable items. As for civil law, one could argue that, at least under Belgian law, the provisions of the Civil Code which relate to the sale of goods are equally applicable to tangible and non-tangible items. (See also F. Gilio, ‘Digital Rights Management on Consumer Goods – A struggle for balance in view of Belgian and European legislation’, European Consumer Law Journal, 2006, issue 4, p. 276.)

The purpose of this section is to identify which elements of the interaction between the vendor and buyer (e.g. placing of an order or making payment) may need to be demonstrable in case of a dispute, taking into account the obligations each of them takes on. Because the priced oblivious transfer scheme is aimed at obfuscating certain substantive elements of a particular transaction, the question arises whether there will in fact remain an ‘audit trail’ which is sufficient to safeguard the rights and interests of both the buyer and the vendor. The required audit trail properties will be derived by looking at ‘misbehavior scenarios’ which might occur and an analysis of which evidence the wronged party might need in order to vindicate his or her rights. We start by looking at potential misbehavior by the vendor (‘cheating vendor scenarios’), and subsequently at potential misbehavior by the buyer (‘cheating buyer scenarios’). The two basic obligations of the vendor and the buyer (i.e., conforming delivery and payment, respectively) form the focal point of this analysis.

Two preliminary remarks still need to be made. In the priced oblivious transfer scheme described above, the vendor ‘delivers’ an item by providing the buyer with the necessary cryptographic key which will enable him (decrypt the key which enables him) to retrieve the requested item. Thus, delivery consists of making available to the purchaser the necessary means to retrieve the item he requested, which, from a legal point of view, is seen as a valid form of delivery.53 A second remark concerns the method of payment. As elaborated earlier, the buyer must first purchase a certain amount of funds (credits) which enable him to subsequently buy the items in which he is interested. Payment will thus consist in the removal of the necessary credits from the buyer’s account. Seeing as the buyer will only be provided with the means to obtain an item upon a finding of sufficiency of funds by the vendor, we will not investigate non-payment as a possible ground for dispute any further.

5.4.1 Cheating vendor scenarios

We have identified the following ways in which the vendor might act in a manner that is detrimental to the rights or interests of the buyer:

1. The vendor charges the account of the buyer without delivering the requested item.

2. The vendor has delivered the requested item, but it is defective in some way (e.g. it is not a complete article, or it is a different one than requested) (non-conforming delivery).

3. The vendor overcharges the account of the buyer or charges the account for no reason.

4. The vendor refuses a request for particular items made by the buyer based on insufficiency of funds, whereas the remaining funds should ordinarily still be sufficient.

When looking at these possible modes of misbehavior, the following audit trail requirements can be derived:

1. The buyer must be able to show non-conformity of product (e.g., the item is different from the one the buyer ordered). This requirement implies that the buyer must be in a position to prove his request after the fact (‘proof of order’).

2. The buyer must be able to demonstrate payment for the item he requested (‘proof of purchase’).

3. The scheme should enable the buyer to see the balance of credits and how the balance diminishes after each order. In other words, it should enable him to verify how much he has been charged and to establish sufficiency of funds.

If a buyer wishes to seek redress for the fact that he did not receive the product he ordered (non-conforming delivery), he must, of course, be willing to surrender his ‘transactional privacy’ with regards to his actual order (but he would only have to do so in instances of non-conforming delivery).

5.4.2 Cheating buyer scenarios

We have identified the following ways in which the buyer might act in a manner that is detrimental to the rights or interests of the vendor:

1. The buyer claims not to have received any item after placing an order and seeing the corresponding credits removed from his account, whereas he has, in fact, been provided with the item in question.

2. The buyer acknowledges having obtained an item subsequent to his request, but claims that this is not the item that he ordered (non-conformity), whereas he did receive the exact item for which he placed an order.

3. The buyer receives a particular item pursuant to his request and is charged for it, but claims that he has not ordered any item at all (repudiation of order).

4. The buyer tampers with the item that was received in order to assert non-conformity.

When looking at these possible modes of misbehavior, the following audit trail requirements can be derived:
1. The vendor must be able to demonstrate non-payment (in case of the priced oblivious transfer scheme: insufficiency of funds)

2. The vendor must be able to demonstrate delivery.

3. The vendor must be able to demonstrate conformity of delivery with what was ordered.

4. The vendor should be able to establish whether delivered items have been tampered with (authenticity and integrity protection).

5.4.3 Conclusion

Under the priced oblivious transfer scheme, the vendor does not learn any details about the items the buyer is requesting. In order to establish conformity of delivery, the vendor must be able to show that the item that was delivered is, in fact, in accordance with the request submitted by the buyer. As elaborated earlier, delivery of the correct item is premised entirely upon correct computation of the final response by the vendor. One could argue that if the vendor can demonstrate that he computed his responses properly, he did, in fact, provide the buyer with the means to obtain the requested item, thus satisfying his obligation to deliver. However, if the client-side application\textsuperscript{54} made an error, but this application was provided by the vendor, this argument would lose relevance. Similar considerations apply to disputes relating to overcharging of the buyer’s account.

Even if the audit trail requirements to safeguard the interests of the buyer could be met (e.g., proof of order, proof of purchase, proof of how credits have accumulated and decreased), it would appear that the audit trail properties which would be needed to safeguard the interests of the vendor would defeat the purpose of the priced oblivious transfer scheme. After all, if the vendor has the ability to prove that the delivered items are in accordance with request made by the buyer, he would effectively have the ability to learn the details of every order the buyer has placed. In our conclusion we will outline potential extensions to the current scheme which might resolve these issues.

\textsuperscript{54} The client-side application is the application which allows the buyer to select the items he wishes to order and to add additional credits.
Chapter 6

Conclusion and Future Work

We have presented the research conducted by the ADAPID partners in the scope of the Storage Work Package, which consists of the design and implementation of a privacy-preserving e-commerce application that employs the Belgian eID card to authenticate buyers towards vendors. Our application, based on priced oblivious transfer, ensures that vendors do not learn which items are bought, while they are guaranteed that buyers pay the right prices.

As argued, privacy preserving e-commerce is convenient both for buyers, who preserve their privacy, and vendors, who can address more easily the legal obligations related with the management of customers’ personal data and who obtain an added value over vendors that employ conventional e-commerce applications. We have also compared our solution with existing priced oblivious transfer schemes and with other privacy-preserving primitives for e-commerce and we have shown that it offers some extra features for vendors, such as efficient customer management and the possibility of applying marketing techniques.

We have also studied certain legal issues which might arise during the deployment of our application in a practical e-commerce scenario. Specifically, we have analyzed consumer information obligations, requirements under VAT legislation, and audit trail requirements. During this analysis we have identified several aspects, such as the duty of providing an acknowledgement of order and the keeping of records, whose fulfillment seems difficult in an e-commerce application based on priced oblivious transfer.

We do however believe that the proposed scheme can be extended in a way which allows both compliance with the legal framework as well as preservation of the privacy-enhancing properties of the proposed priced oblivious transfer scheme. In order to achieve this, several extensions will be necessary. In first instance, we envisage the use of remote integrity checking tools \cite{AKM08}, i.e., tools that allow a software provider to check whether a remote entity is running a non-tampered copy of its software. By means of these tools, the vendor can provide the buyer with a dedicated application and,
at any purchase, ensure remotely that the buyer is using this application.

Once the vendor is able to ensure that he is executing the protocol with a trusted application, he can provide the application with extra functionalities in order to fulfill the legal requirements, such as the generation of acknowledgements of orders and documents in proof to support of accounting and registration obligations. For the former, the application could for instance request from the vendor a blind signature \cite{Cha82} on the buyer’s order, which would constitute a valid acknowledgement of the order. For the latter we propose a solution based on group signatures \cite{CvH91}, i.e., signature schemes that allow signing on behalf of a group, in such a way that the verifier cannot identify which entity in the group that produced the signature. The vendor can set up a group signature scheme and provide each application with a signing key. The application keeps track of all the items bought during each time period, and at the end of the period computes a signature on the identifiers of the items, and sends this signature to the vendor through an anonymous channel (so that the vendor cannot learn who bought those items). The vendor can verify the signature and use the identifiers of the items to produce the records. In case of dispute, e.g., when the legal authority suspects that the vendor produces signatures himself, a trusted party that holds the opening key can open a signature to learn who produced it. Further research will be needed to determine with greater certainty whether these extensions do in fact resolve all of the current legal barriers towards implementation of priced oblivious transfer schemes in e-commerce applications.

Finally, we note that our scheme employs novel cryptographic primitives, such as bilinear maps, which are not widely used in existing applications. Therefore, it is also necessary to integrate the resources required to implement our scheme into tools that are currently used by applications and web programmers. This would facilitate the transfer of knowledge and the adoption and implementation of novel solutions.
Bibliography


BIBLIOGRAPHY


Institute for Theoretical Computer Science, ETH Zürich, March 1997.


