Priced Oblivious Transfer: How to Sell Digital Goods

Bill Aiello* Yuval Ishai† Omer Reingold‡

July 25, 2001

Abstract

We consider the question of protecting the privacy of customers buying digital goods. More specifically, our goal is to allow a buyer to purchase digital goods from a vendor without letting the vendor learn *what*, and to the extent possible also *when* and *how much*, it is buying. We propose solutions which allow the buyer, after making an initial deposit, to engage in an unlimited number of *priced oblivious-transfer* transactions, satisfying the following requirements: As long as the buyer's balance contains sufficient funds, it will successfully retrieve the selected item and its balance will be debited by the item's price. However, the buyer should be unable to retrieve an item whose cost exceeds its remaining balance. The vendor should learn nothing except what must inevitably be learned, namely, the amount of interaction and the initial deposit amount. In particular, the vendor should be unable to learn what the buyer's current balance is or when it actually runs out of its funds.

The technical tools we develop, in the process of solving this problem, seem to be of independent interest. In particular, we present the first one-round (two-pass) protocol for oblivious transfer that does not rely on the random oracle model. (A very similar protocol was independently proposed by Naor and Pinkas [21].) This protocol is a special case of a more general "conditional disclosure" methodology, which extends a previous technique from [11] and adapts it to the 2-party setting.

*AT&T Labs – Research, 180 Park Ave., Florham Park, NJ 07932. E-mail: `aiello@research.att.com`.

†DIMACS and AT&T Labs – Research. E-mail: `yuval@dimacs.rutgers.edu`.

‡AT&T Labs – Research, 180 Park Ave., Florham Park, NJ 07932. E-mail: `omer@research.att.com`.

1 Introduction

Consider a scenario where a buyer wishes to purchase digital goods from a vendor without disclosing *what* it is buying, or even *when* exactly it is buying. For instance, the buyer may wish to subscribe to a pay-per-view service, where different costs are associated with different channels, or get an up-to-date information on its stock portfolio. In both cases buyers may wish to hide from vendors what items they are buying, or even whether at a given moment they are buying anything at all.

In the realm of physical goods, it is inherently impossible to hide from the vendor what, when, and how much it is selling. Being bounded to a limited inventory, the vendor must keep track of how many items of each kind it has in stock. However, unlike physical goods, digital goods are typically of unlimited supply. The purpose of this paper is to exploit the difference between the physical and the digital worlds in order to obtain privacy of buyers in the following electronic commerce scenario. Assume that a buyer first deposits a pre-payment at the hands of a vendor.[1] The buyer should then be able to engage in a virtually unlimited number of interactions with the vendor in order to obtain digital goods (also referred to as *items*) at a total cost which does not exceed its initial deposit amount. After spending all of its initial credit, the buyer should be unable to obtain any additional items before depositing an additional pre-payment. This paper provides efficient ways to implement this, rather standard, e-commerce task with the added requirement of *maintaining the buyer's privacy.* That is, the vendor should learn nothing except what must inevitably be learned: the amount of interaction and the initial deposit amount (which imply upper bounds on the quantity and total price of all information obtained by the buyer). In particular, the vendor should be unable to learn what the buyer's current balance is or when it actually runs out of its funds.

Traditional approaches for protecting the privacy of buyers, such as anonymous digital payments (e.g., [7, 8]), do not address the problem of hiding which goods are being bought and when. This information, possibly combined with additional information from other sources (such as traffic analysis), may facilitate attacks on the privacy of individual buyers.[2] Moreover, strong anonymity is not only difficult to implement and prone to various types of attacks [2], but in some contexts it is also undesirable [26]. We stress that our solutions do not require anonymity of buyers and do not attempt to achieve this property. On the contrary, our work provides an *alternative* approach for protecting individual buyers engaging in e-commerce activities, which promises a different (and in a sense stronger) type of security. This approach is most beneficial when anonymity is undesirable, insufficient, or difficult to achieve.

Priced Oblivious Transfer. The well-known *oblivious transfer* primitive [25, 10, 4, 15] provides a partial solution to our problem. Oblivious transfer allows the buyer to retrieve one of several items held by the vendor without revealing to the vendor the identity of the item being retrieved. If all items are identically priced, then the buyer's initial deposit determines the number of items it is entitled to obtain. Thus, in this case the vendor may allow the buyer to retrieve just the right number of items by using multiple invocations of oblivious transfer. However, this solution is not applicable in the realistic scenario where the items are not identically priced. Moreover, coping with differently priced items may be highly beneficial even in the case that all "real" items have the same price. By adding a single dummy item with price 0, the buyer has the option of "buying" this item an arbitrary number of times for the sole purpose of hiding when it is buying *real* items. This added privacy feature is impossible to achieve with a standard use of oblivious transfer, unless the buyer is willing to pay for all the dummy items it retrieves.

Obtaining a complete solution to our problem requires a more general protocol that we call *priced*

[1] By having the buyer pay to a third party, the vendor may be initialized with an *encryption* of the buyer's deposit and therefore not even learn the deposit amount.

[2] One may argue that without any such information, the vendor can hardly optimize the offered goods. However, marketing-related information can still be *voluntarily* provided to the vendor by potential buyers.

oblivious transfer. Assume that at the beginning of each phase of interaction the vendor holds an encryption of the buyer's current balance. A phase of interaction (also referred to as a *transaction*) should allow the buyer to *privately* retrieve a *single* item. This in itself is an oblivious transfer protocol. However, in this case we have the following additional requirements: (1) The buyer can only retrieve an item if its current balance is larger than the item's price; (2) The price of the item the buyer retrieves should be decreased from the buyer's (encrypted) balance.

Broadcast Encryption. A prime motivating example for priced oblivious transfer is as follows. A vendor is broadcasting n different data streams. The data streams may be video, audio, or text and the content may be news, entertainment, technical and professional information, etc. To accomplish private buying in this setting, the vendor encrypts each of the n streams with a different key. The buyer and vendor then engage in a priced oblivious transfer protocol where the keys are the items being transferred. The buyer is then able to decrypt the data stream that it paid for, but as it does not have knowledge of the other keys, it is unable to gain access to the content of the other data streams.

Subscriptions. An important extension to enabling the purchase of a single digital good per transaction is to allow subscriptions. In a subscription scenario, the vendor changes the database periodically. Denote the ith data item at time j as x^i_j. The sequence of the ith data items over time, x^i_0, $x^i_1, \ldots$, is called the ith *channel* or channel stream. For example, a channel may be a daily financial white paper or a daily decryption key for a broadcast stream as above. In this setting the buyer is allowed to *subscribe* to a channel. As with a single data item from a static database, the channel to which a buyer subscribes should remain private. While the buyer is subscribed to a channel, it receives the sequence of data items of the channel and its balance is deducted by the appropriate amount each time period of the channel. The buyer remains subscribed to the channel until it explicitly *unsubscribes* or until its balance becomes negative. It is clear that the operation of subscribing to a channel can be simulated by repeated operations of purchasing an item. The issue however is one of efficiency and in particular it is a question of the communication pattern: While buying inherently requires some non-trivial interaction, maintaining a subscription should ideally require only efficient one-way communication from the vendor to the buyer. Allowing an efficient subscription implementation (with one-way communication) seems to be vital in many of the applications we have in mind. We therefore extend our solutions to handle this additional requirement.

A NOTE CONCERNING EFFICIENCY. The main goals of this work are to put forward a new problem, establish a "practical feasibility" result for this problem, and in the process develop some useful general tools. We do not attempt at minor optimizations which would complicate the presentation. Our solution should be mainly viewed as a feasible framework which may be the basis for further optimizations.

Additional Contributions. Several ingredients of our construction seem to be of independent interest. In particular, we obtain the first implementation of a 1-round oblivious transfer protocol satisfying a "reasonable" security definition and provably secure under a "reasonable" security assumption. The security of our protocol can be based on the decisional Diffie-Hellman (DDH) assumption. A similar protocol has been independently obtained by Naor and Pinkas [21]. The oblivious transfer protocol follows from a more general *conditional disclosure* methodology, which can be used in some contexts as a light-weight alternative to zero-knowledge proofs. In this we extend an "information-theoretic" technique from [11] (see Section 2.3) and adapts it to the 2-party setting. In the course of addressing the case of subscriptions, we propose efficient solutions for the problem of privately retrieving a chosen *prefix* of a long stream of information.

Related Work. General techniques for secure 2-party computation [28, 13] may be used to solve our problem. However, similarly to most other works in this area, our goal is to use the specific structure of the problem at hand for providing far more efficient solutions than those obtained via general techniques.

The current work has been greatly inspired by previous works on *specific* secure computation tasks such as private information retrieval (PIR) and oblivious transfer. In Section 2.3 we describe some relevant techniques from these works which we rely on or extend. A restricted "off-line" variant of our problem may be viewed as a special case of a *generalized oblivious transfer* primitive studied in [14]. In a distributed multi-vendor setting, an off-line variant of our problem has been considered in [11]. Adapting the solutions from [14, 11] to our setting would result in very inefficient protocols. We stress that unlike the PIR-related context of [11], where the main concern is that of minimizing the asymptotic complexity as a function of the number of data items, most aspects of our problem are equally interesting even when the number of items is as small as 2.

Organization. The remainder of the paper is organized as follows. In Section 2 we specify the problem and its security requirements, and review the tools we will use. In Section 3 we describe our basic protocol and its properties. We also discuss some efficiency improvements. In Section 4 we discuss an extension to the subscription scenario. Finally, in Section 5 we present the one-round OT protocol which is a special case of our methodology.

2 Preliminaries

2.1 Problem Specification

As discussed in the introduction, our goal is to construct an "on-line" protocol between a buyer $\mathcal{B}$ and a vendor $\mathcal{V}$ which allows the buyer and the vendor to engage in multiple transactions. Both the buyer and the vendor are allowed to store a (short) state information between transactions. Before specifying the security aspects of the protocol, we will first describe its desired functionality.

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

Main Protocol: At time t, $t = 1, 2, \ldots$

- The vendor may choose a database $\mathbf{x} = (x^0, x^1, \ldots, x^{n-1})$ of n items for sale and some public information $\mathbf{P}$ concerning the identity of these items. $\mathbf{P}$ contains a price list $\mathbf{p} = (p^0, p^1, \ldots, p^{n-1})$. By convention, x^0 is a dummy item with $p^0 = 0$.
- The buyer may then decide either to:
  - *Buy* the i-th item, where $0 \leq i < n$; if the buyer's remaining balance is sufficiently large (i.e., the combined price of all items previously received and the current price p_i does not exceed the initial deposit), the buyer receives x_i.
  - *Subscribe* to the i-th channel; by subscribing, the buyer indicates that it wishes to continue buying the i-th item until overriding the subscription with a new request. We assume that throughout the subscription, the buyer is charged the price p^i effective when initiating the subscription (even though $\mathbf{p}$ may change).
  - *Unsubscribe*, i.e., terminate a previous "subscribe" request.
  - *Do nothing*, i.e., maintain its default subscription if such exists, and otherwise keep idle.

Underlying the above specification is the efficiency-motivated assumption that the type of request made by the buyer may determine the interaction pattern. If it weren't for this assumption, a single "buy" request would have been sufficient.

Finally, in our presentation we make the simplifying assumption that at any given transaction the buyer retrieves at most one item (either via a 'buy' or 'subscribe' action). Our solutions can be efficiently extended to allow multiple retrievals from the same database, e.g., using techniques from [20].

2.2 Security Requirements

Efficiency considerations dictate some compromises we make in comparison to full-fledged simulation-based definitions for secure computation (e.g., those of [6, 12]). Nonetheless, our solutions are provably secure under standard security assumptions. Our formal security requirements, which are only sketched below, can be found in Appendix A.

Both the buyer $\mathcal{B}$ and the vendor $\mathcal{V}$ are modeled by efficient randomized algorithms, and are initially given a security parameter 1^κ as a common input. We assume that subsequent "inputs" are dynamically chosen by $\mathcal{B}, \mathcal{V}$ as the protocol proceeds. The protocol is divided into transactions, as described in Section 2.1, and is assumed to terminate after a polynomial number of transactions, depending only on κ. An *honest* buyer is restricted to choose items such that their total price does not exceed the initial deposit amount $b^{(0)}$.

We first address a default scenario which only allows the buyer to issue "buy" requests. A protocol $(\mathcal{B}, \mathcal{V})$ as above is considered *secure* if it satisfies the following requirements:

Correctness. If both $\mathcal{B}$ and $\mathcal{V}$ are honest, then $\mathcal{B}$ outputs the correct item x^i at the end of each transaction.

Buyer's security. A malicious vendor should not learn the choices made by an honest buyer. More formally, the view of any efficient (and possibly malicious) $\mathcal{V}^*$ in the interaction $(\mathcal{B}, \mathcal{V}^*)(1^\kappa)$ can be efficiently simulated. We note that this requirement is weaker than that of general security definitions in that it does not address the effect $\mathcal{V}^*$ may have on the output of $\mathcal{B}$. In particular, $\mathcal{V}^*$ does not need to "know" a database x which is effectively determined by its strategy in a given transaction. This is consistent with other definitions of related primitives (such as PIR, see Section 2.3, or even some definitions of oblivious transfer).

Vendor's security. A malicious buyer should not obtain more information than what its initial deposit allows. This is formalized by requiring that the interaction of $\mathcal{B}^*$ with an honest vendor $\mathcal{V}$ could be efficiently simulated in the natural idealized model.

Our security definitions for the general case, where the buyer may take any of the four actions, are more subtle. In a nutshell, the vendor's security requirement remains unchanged, and can be defined as above. The buyer's security in this setting may also be defined similarly to the above. However, such a definition will only be satisfied when the buyer's action type is oblivious to the received items, i.e. depends only on public data (yet its specific selections i may also depends on received items). The reader is referred to Appendix A for a more detailed discussion.

Finally, while we do not explicitly address issues of robustness or recovery from faults, our protocols can be extended in a straightforward manner to deal with these issues.

2.3 Tools

Homomorphic Encryption. Our constructions rely on the widely used tool of *homomorphic encryption*. Loosely speaking, an encryption scheme is said to be homomorphic if: (1) The plaintexts are taken from a group $(H, +)$; (2) From encryptions of group elements h_1, h_2 it is possible to *efficiently* compute a *random* encryption of $h_1 + h_2$. A useful consequence is that given an encryption of a group element h and an integer c in binary representation, one can *efficiently* compute a random encryption of $c \cdot h$. This is done in a similar fashion to the repeated squaring procedure for modular exponentiation.

In what follows H will always be a group of a (large) prime order Q. It is important to note that by "+" we denote an abstract group operation. Hence, our notation applies both in a case where $H = Z_Q$ is an *additive* group, and where $H \subset Z_P^*$ is a *multiplicative* group. A useful example of a multiplicative homomorphic encryption is the *El-Gamal* scheme. (We refer the reader to, e.g., [22] for relevant definitions.) In this case, H is a subgroup of Z_P^*, where Q is a prime of length κ that divides $P - 1$.

We prefer an additive notation over a multiplicative one due to its more intuitive nature in our context. However, our protocols can be instantiated with both types of encryption. We note that *all* of our constructions can be based on the El-Gamal encryption (whose security is equivalent to the DDH assumption, cf. [22]) and most on any other homomorphic encryption scheme candidate, e.g. [18, 23, 24]. An additional property enjoyed by the El-Gamal encryption, which explains the above distinction, is discussed below.

Verifiability. It is sometimes required to verify the validity of a public key k and the validity of a ciphertext c relative to a valid k. Luckily, the latter verification task is typically easy, and we can therefore assume it as part of our default requirements. However, in most encryption schemes the validity of the public key itself is difficult to verify. To this end a special zero-knowledge proof procedure may be employed during the initialization stage of our protocols. This step, however, is not always needed. A useful added feature of the El-Gamal scheme is that its public keys are easily verifiable: to verify that (P, Q, g, h) constitutes a valid public key, it is enough to verify that P, Q are primes, Q divides $P - 1$, and $g^Q \equiv h^Q \equiv 1 \pmod{Q}$.

See Appendix B for a more formal treatment and additional details on homomorphic encryption schemes.

PIR. A Private Information Retrieval (PIR) protocol [9] allows a user to retrieve a selected item from a database while hiding the identity of this item from the server holding the database. PIR only requires the protection of the user, and makes no requirement on the privacy of the database. Thus, a naive solution to the PIR problem is to send the entire database to the user. When the database is large, this solution is very expensive in terms of communication. The main goal of PIR-related research has been to minimize the communication complexity of PIR, which is measured by default as the cost of retrieving one out of n bits. Assuming either a general homomorphic encryption [16, 17, 27] or a stronger number theoretic assumption [5], the asymptotic communication complexity of PIR can be made very small. In practice, however, the naive solution may still be preferable due to the high computational cost of sublinear-communication PIR. Thus, when we use PIR as a building block in our protocols, one should always keep in mind that the naive solution can be used for its implementation, especially when the number of items is small.

Naor-Pinkas Pseudo-Random Sequence. A variant of PIR where the user is restricted to learn no more than a single data item has been referred to in the literature as *symmetrically* private information retrieval (SPIR) [11].[3] In [19] (followed by [20]), Naor and Pinkas suggested the following reduction from SPIR to PIR. Suppose that there is an efficient method allowing the user to retrieve exactly one out of n *pseudo-random* items $(r^0, \ldots, r^{n-1})$ chosen by the server. Then, SPIR can be solved by applying such a procedure and concurrently applying PIR on the database $(x^0 \oplus r^0, \ldots, x^{n-1} \oplus r^{n-1})$. The pseudo-random sequence $(r^0, \ldots, r^{n-1})$ is created in the following way. Represent i as a length-ℓ binary string (in this case, $\ell = \log n$). Let $(s_1^0, s_1^1), (s_2^0, s_2^1), \ldots, (s_\ell^0, s_\ell^1)$ be ℓ pairs of independent keys to a pseudo-random function f, and define $r^i = \oplus_{j=1}^{\ell} f_{s_j}(i)$ where $s_j = s_j^{i_j}$. By letting the user choose *one* key from each pair (s_j^0, s_j^1), the user can learn any selected r^i, but no more than one r^i. A SPIR protocol constructed via the above method keeps all but a single data item x^i semantically secure from the user. More precisely, it is possible to simulate the view of a user, whose $\log n$ selections define an index i, based on x^i alone (up to *computational* indistinguishability).

[3]This problem is very similar to $\binom{n}{1}$-OT (1-out-of-n oblivious transfer), except for the focus on sublinear communication and somewhat relaxed security requirements.

We extend the above construction in two ways. First, we observe that the role of i can be replaced by longer indices, possibly from an exponentially large domain (say, prices in our case); the only requirement is that the number of different *items*, or "useful" indices, is feasible. Second, by utilizing a more sophisticated key-selection policy, we devise an efficient protocol allowing the user to privately retrieve a chosen *prefix* of a long pseudo-random sequence. This application is discussed in Section 4.

Conditional Disclosure of Secrets. Motivated by the problem of constructing efficient SPIR protocols in the multi-server setting, Gertner et al. [11] suggested the following *conditional disclosure* primitive. An input string y to a public Boolean predicate C is *partitioned* among k servers, such that no server knows the entire string y. In addition, one of the servers holds a secret s. The goal of the servers is to each send a single message to a user, who knows y, such that the user will learn s if $C(y) = 1$ and otherwise will learn no information on s. To make this possible, the servers have a common random input r which is unknown to the user. In [11], the problem is reduced to *generalized secret-sharing*. It is shown that the communication complexity of conditional disclosure as above is linear in the span program size of C (and in particular in the *formula size* of C). If the user is allowed to "help" the servers by secret-sharing a *witness* to the validity of $F(y)$ between them (without letting individual servers learn additional information on y), the communication can be made linear in the *circuit size* of C. Moreover, these solutions were efficiently extended to the non-Boolean case, where y is a string over a large field, and the condition C tests whether y satisfies some linear equation over F (or more complicated predicates over such atomic conditions).

A main ingredient of our protocol is an almost exact adaptation of the above conditional disclosure scenario to the single-server setting. In our setting, y will always be viewed as a vector over a large field $F = Z_Q$. Instead of partitioning $y = (y_1, \dots, y_m)$ among several servers, a single server holds a public key k, the *encryptions* $E_k(y_1), \dots, E_k(y_m)$, and a secret $s \in F$. The user holds both y and the secret key corresponding to k. An important observation regarding the solutions to the multi-server conditional disclosure problem mentioned above is that the joint messages sent by the servers may be expressed as a random *linear* function of (y, s), where the distribution of this linear function depends only on C. Therefore, if the encryption scheme E is homomorphic, the server may compute an encryption of these messages from $E_k(y)$. Instead of formulating our solutions in a general complexity-theoretic terminology, we will solve the required instances along the way in an intuitive way.

3 Solving The Problem

In this section we describe our solutions for the priced oblivious transfer problem. For the sake of presentation, we develop our solutions gradually and improve their efficiency along the way. In particular, the only operation we consider at first is 'buy'. We deal with subscription operations in Section 4.

Establishing a Public-Key Meta Structure. As described in introduction, during the entire run of our protocol the vendor will maintain an encryption of the buyer's current balance (using the public key of the buyer). Let E, D and G be the encryption, decryption and key-generation algorithms respectively. In the *initialization phase* of the protocol (time 0), the buyer applies G to sample a public-key, secret-key pair (k, sk) and sends the public-key k to the vendor. The vendor needs to verify that k is indeed a valid public-key and that the buyer knows a private-key sk that corresponds to k. Therefore, the buyer also proves in zero-knowledge that it knows an input of G that generates the public key k.[4] Finally, the vendor sets the current balance $b^{(0)}$ to the initial deposit of the buyer

[4] Suppose that the encryption scheme enjoys the *verifiability* property discussed in Section 2.3. In this case, if the vendor is willing to settle on a somewhat weaker notion of security, it can verify the validity of k on its own instead of letting the buyer prove this validity as above. We use this fact in our one-round oblivious transfer protocol (where we cannot afford the additional rounds required for the zero-knowledge proof). See Section 5 for details.

and creates an encryption $E_k(b^{(0)})$ of the balance.

The first challenge in designing our protocol is that, at each transaction, the vendor needs to update the encrypted balance $E_k(b)$ by some value p *without knowing either* b *or* p. It should not be surprising that in order to do so it is useful to let E be a *homomorphic encryption*. Recall that we assume that the plaintexts are taken from a group G_Q of order Q, where Q is a prime of length κ. Under our additive notation, it is convenient to view G_Q as the *field* $F = Z_Q$ (see Section 2.3 and Appendix B for more details).

Representations. We assume for simplicity that the length of each data item x^i is smaller than the security parameter κ. Even if this is not the case, our problem can be reduced to that of selling *keys* which encrypt the actual data. We take $B = 2^\ell$ to be an upper bound on the initial balance, where $B < Q$. This allows to represent prices and balances as elements of F by identifying (in the natural way) each integer i in the interval $[B-Q, B-1]$ with the corresponding element of F. Thus, the elements $0, 1, \ldots, B-1 \in F$ will be referred to as non-negative, and $B, \ldots, Q-1$ as negative. In all of our protocols we will view a positive balance as being valid, and a negative balance as being invalid. If the buyer's balance is negative, it should not be allowed to learn any additional information.

3.1 Basic Solution

We present a solution where each transaction (here, a single 'buy' operation) requires two passes of communications: (1) A message from the buyer; (2) The vendor's reply. This is optimal since even without privacy the buyer still needs to specify the item it wants to retrieve and the vendor needs to send this item.

Assume without loss of generality that all item prices are distinct. (This assumption can be easily dispensed with at a moderate efficiency cost, e.g. by replacing each price p^i by $B'p^i - i$ for a sufficiently large B', and scaling the initial deposit by a factor of B'.) The most essential part of the *buyer's message* is an encryption $E_k(p)$ where p is the price of the item it wants to retrieve. The vendor needs to perform two operations: (1) Update the balance; (2) Send back (in some encrypted form) the item x^i such that $p = p^i$.

Updating the Balance. Since the vendor has an encryption of the current balance $E_k(b)$ and it received an encryption $E_k(p)$ of the retrieved item's price, it seems that updating the balance is not a problem. Simply create an encryption $E_k(b-p)$ of the new balance using the homomorphism of E. However, we should be careful: By setting p to be negative (e.g. $b - B + 1$), the buyer can arbitrarily increase its balance (this is of course undesirable, regardless of whether in this specific transaction the buyer gains any information).

One way to prevent the buyer from cheating in this manner is to require it to prove in a zero-knowledge fashion that $0 \le p \le b$. Such a solution requires more passes of interaction than desired. A better solution *in this respect* is for the buyer to use *non-interactive* zero-knowledge proofs of this claim (for that the buyer and vendor can agree upon a random string in the initialization phase of the protocol). However, non-interactive zero-knowledge proofs are usually very inefficient and we therefore give in Section 3.3 an alternative (more efficient) solution. Jumping ahead, the vendor in the revised protocol will not try to *verify* that p is in the right range but will rather make sure that any such violation on the part of the buyer will *cripple all future interactions*. We note that [3] gives an efficient zero-knowledge proof to a related problem, of proving that a *committed* number lies in a an interval. However, the problem we solve (and hence our machinery for solving it) is easier.

Sending an Item. We now assume that $0 \le p \le b$ and that the balance was updated by the vendor. All that is left is for the vendor to "send" an item x^i such that $p = p^i$ (if such an item exists).

The vendor's message is composed of n (parallel) messages $m^0, m^1, \ldots, m^{n-1}$. For every j, the message m^j allows the buyer to compute x^j in case $p = p^j$ and gives the buyer no information if $p \neq p^j$. Note that for a fixed j, what we have is in a sense an instance of *conditional disclosure in a computational setting*. The vendor wants to disclose the value x^j conditioned on $p = p^j$. For this simple condition (equality) the solution is very simple: For every j, the vendor uniformly samples $\alpha^j \in Z_Q$ and sets m^j to be a (random) encryption $E(\beta^j)$ of $\beta^j = \alpha^j(p - p^j) + x^j$. It is immediate that $\beta^j = x^j$ in case $p = p^j$ and is random in Z_Q if $p \neq p^j$ (therefore, in this case the buyer gets no information on x^j in an information theoretic sense).

Adapting the conditional disclosure methodology of [11] to the computational setting is one of the main tools of our solution. In addition to the example above, it is used extensively in Sections 3.3 and 4.

3.2 Reducing the Communication

The protocol of Section 3.1 has the disadvantage that the vendor's message is of linear length as a function of n (the number of items). This in itself is a non-trivial task and for some applications may be sufficient. We now give a simple method for reducing the communication. In Section 3.5 we discuss an alternative method for reducing the communication (and computation) which is superior in most settings of the parameters, but is slightly more involved.

The main observation for reducing the communication is simple: If the buyer wants to retrieve item x^i then the only part of the vendor's message it needs is the value m^i (in fact, the rest of the message is useless). Therefore, instead of getting the entire sequence, $m^0, m^1, \ldots, m^{n-1}$, the buyer can just retrieve m^i using a PIR protocol (where we view the vendor's message as a database of n records). Note that in this case PIR is sufficient since security is preserved even if the buyer learns the entire sequence. To implement this solution, the buyer should add to its message a PIR query for the index i and the vendor should send back a corresponding PIR reply using the database $m^0, m^1, \ldots, m^{n-1}$ which it can compute.

3.3 Avoiding Zero-Knowledge Proofs

In the protocol of Section 3.1, the buyer sends an encryption $E_k(p)$ and proves in zero-knowledge that $0 \leq p \leq b$. This was important for two reasons: (1) To prevent the buyer from learning x^i with $p^i > b$ *in the current transaction*; (2) To prevent the buyer from increasing its balance (in order to gain additional information *in future transactions*). However, as discussed above, both interactive and non-interactive zero-knowledge proofs are not efficient enough for our needs, and are in a sense an overkill. We now show how to replace zero-knowledge proofs with *conditional disclosures*. In these solutions, the vendor will not be able to *detect* a value p that is outside of the range $[0, b]$. Nevertheless, each such violation will prevent the buyer from learning any additional information.

The idea is simple. At the t-th transaction, the vendor will sample a random mask v^t and a random receipt u^t. The vendor will disclose v^t and u^t under the condition that $0 \leq p \leq b$. The value v^t will be used to mask the interaction in the current transaction (i.e. instead of retrieving x^i the buyer will retrieve $x^i + v^t$). The value u^t will be used as a receipt for future interaction – knowing u^t implies that the buyer behaved correctly until now.

A naive way to use the receipt u^t is to require the buyer to send it at the beginning of the next transaction. As it turns out, this solution may compromise the privacy of the buyer against a malicious vendor. We therefore use a chaining technique: at the t-th transaction the buyer will also send an encryption $E_k(u)$. The vendor will disclose v^t and u^t under the condition $(0 \leq p \leq b) \wedge (u = u^{t-1})$. We note that other methods of chaining are possible in this scenario. However, we find this particular solution appealing, both from a conceptual point of view and because it allows to maintain *statistical* vendor's security.

One may view this kind of chaining as an ongoing proof of the buyer that it behaves correctly, where the proof never gets to its conclusion (i.e. convincing the vendor). This kind of a technique may be useful in other scenarios.

It remains to show how to perform the more involved conditional disclosure needed here. We already saw how to perform conditional disclosure for equality. This also implies a recursive way to perform conditional disclosure under any condition that can be described as a monotone formula where the leaves are equalities: Assume we know how to perform conditional disclosure under the conditions A_1 and A_2. To perform conditional disclosure of x under $(A_1 \vee A_2)$, just perform two independent conditional disclosures of x — One under A_1 and the other under A_2. To perform conditional disclosure of x under $(A_1 \wedge A_2)$, sample a random mask r, disclose r under A_1 and $x + r$ under A_2.

To perform a conditional disclosure under the condition $(0 \leq p \leq b) \wedge (u = u^{t-1})$ it is enough to describe the condition $0 \leq p \leq b$ by a small monotone formula as above. For this purpose we will need some help from the buyer. Recall that $B = 2^\ell$ is an upper bound on a valid balance. In its message, the buyer will send separate encryptions of the bits $b_{\ell-1}, \ldots, b_0$ and $p_{\ell-1}, \ldots, p_0$ where $b_{\ell-1} \ldots b_0$ is supposed to be the binary representation of the current balance b and $p_{\ell-1} \ldots p_0$ defines the price p (i.e. $p = \sum_j p_j 2^j$). Note that the vendor can create an encryption of p from the encryptions of the bits p_i. The condition $0 \leq p \leq b$ is implied by the conjunction of the following conditions: (1) $b = \sum_j b_j 2^j$; (2) $b_{\ell-1}, \ldots, b_0$ and $p_{\ell-1}, \ldots, p_0$ are all bits; (3) $p \leq b$ when p and b are viewed as integers. It is well known (and rather simple) that (3) can be represented as a monotone formula of size $O(\ell)$ with leaves that are equalities (in the bits $b_{\ell-1}, \ldots, b_0, p_{\ell-1}, \ldots, p_0$ and the constants 0 and 1). We may therefore conclude that $0 \leq p \leq b$ can also be represented as such a monotone formula of size $O(\ell)$.

3.4 Putting the Pieces Together

The ideas presented so far already combine into a protocol that satisfies the specification of Section 2.1, has the desired communication pattern, and is relatively efficient. This protocol is still not the most efficient we propose (significant improvements are described in Section 3.5) and it does not handle subscriptions (which are dealt with in Section 4). Nevertheless, since most of the ideas already appear in this solution, we now give a short summary of the protocol and informally discuss its properties.

3.4.1 The Protocol

Initialization. The buyer applies the key generator G to sample a public-key, secret-key pair (k, sk) and sends the public-key k to the vendor. The buyer also proves in zero-knowledge that it knows an input of G that generates the public key k. The vendor creates an encryption $E_k(b^{(0)})$ of the initial balance $b^{(0)}$. Finally, both set u^0 to be some predefined string (e.g. the all zero string).

Buyer (time $t > 0$). The buyer's message is composed of (1) $E_k(u)$ (u is supposed to be u^{t-1}); (2) $E_k(b_{\ell-1}), \ldots, E_k(b_0)$ and $E_k(p_{\ell-1}), \ldots, E_k(p_0)$, where $b_{\ell-1} \ldots b_0$ is supposed to be the binary representation of the current balance $b^{(t-1)}$ and $p_{\ell-1} \ldots p_0$ the binary representation of the price p^i; (3) A PIR query q for the index i.

Vendor. The vendor computes an encryption of $p = \sum_j p_j 2^j$ and creates an encryption of the new balance $b^{(t)} = b^{(t-1)} - p$. It samples two keys v^t and u^t uniformly at random in F and discloses both under the condition $(b = \sum_j b_j 2^j) \wedge (0 \leq p \leq b) \wedge (u = u^{t-1})$. For every j, the vendor computes m^j which is the conditional disclosure of $x^j + v^t$ under the condition $p^j = p$. Finally, the vendor answers with the PIR answer to the query q for the database $(m^0, \ldots, m^{n-1})$.

Buyer's output. The buyer retrieves m^i and computes x^i (which is its output for this transaction). In addition, the buyer recovers and stores u^t for future interaction and also remembers the new balance.

3.4.2 Properties

Correctness. For honest buyer and vendor is straightforward.

Buyer's security. Follows from the semantic security of E since all the (even malicious) vendor sees at each transaction is a fixed number of encryptions. That is, a simulator for the view of $\mathcal{V}^*$ may first simulate the initialization stage, and then produce an appropriate number of encryptions for each transaction. Note that a malicious vendor can indeed create dependencies between the buyer's inputs and outputs (as discussed in Section 2.1).

Vendor's security. For any buyer $\mathcal{B}^*$, even malicious *and unbounded*, there exists an efficient simulator $\mathcal{S}$, with black-box access to $\mathcal{B}^*$, that produces an output which is *statistically close* to the view of $\mathcal{B}^*$. The simulator invokes $\mathcal{B}^*$ and simulates its conversation with $\mathcal{V}$. The first step is to extract (using the zero-knowledge extractor) the secret-key sk that corresponds to k. Given this information, the rest of the simulation is fairly straightforward. The only point that needs arguing is that starting at the first time t for which the condition $(b = \sum_j b_j 2^j) \wedge (0 \le p \le b) \wedge (u = u^{t-1})$ is violated it will be violated at all subsequent transactions (with overwhelming probability).

Efficiency. Excluding the PIR protocol, the buyer performs $O(\ell)$ public-key operations and its message consists of $O(\ell)$ encryptions. The vendor however is much less efficient — it performs $O(n)$ public-key operations (to create the messages m^j). The vendor's message consists of a PIR reply for the database containing the strings m^j. This in itself already seems optimal: Any solution to our problem will in particular give a PIR protocol for the database $x^0, x^1, \ldots, x^{n-1}$. Therefore, we cannot expect to have communication which is smaller than that of a PIR protocol. However, here the strings m^j can be significantly longer than the strings x^j, which may result in a communication blowout. In Section 3.5 we show how to achieve savings in both the communication and work on the part of the vendor.

3.5 Additional Improvements

We now describe a modification of the protocol of Section 3.4 that typically improves its performance. The alternative approach is especially natural in the case where the vendor only sells *keys* encrypting the data, and the encrypted data is accessed by other means (e.g., via broadcast, or a PIR protocol). We assume that these keys are refreshed at each transaction (in particular, we would not like the buyer to get all values of x^i after buying it once) and describe the modification in this setting.

The keys that the vendor will sell are a carefully chosen subsequence of the Naor-Pinkas pseudo-random sequence (see Section 2.3). Let ℓ be as above (i.e. the length of the binary representation of prices). Let $(s_0^0, s_0^1), (s_1^0, s_1^1), \ldots, (s_{\ell-1}^0, s_{\ell-1}^1)$ be ℓ pairs of independent keys to a pseudo-random function f, and let $\{k^z\}_{z\in\{0,1\}^\ell}$ be the Naor-Pinkas sequence that is generated by these ℓ key pairs.

The idea is the following. Let the j-th *key* that the vendor sells be the element of the Naor-Pinkas sequence *indexed by the price* p^j *of this key* (i.e. the element k^{p^j}). This slightly unusual choice (the more natural choice seems to be taking the j-th key to simply be k^j) is the main observation of the revised protocol. To make it even more compatible with our solution, we let the j-th key at time t be $k^{p^j} + v^t$ (recall that the sequence $\{k^z\}_{z\in\{0,1\}^\ell}$ is refreshed at each transaction). We can now consider the following adjustment in the protocol.

The buyer sends almost the same message as before (there is no need to send the PIR query). Recall that as part of its message, the buyer sends encryptions of the bits of the price $E_k(p_{\ell-1}), \ldots, E_k(p_0)$. The vendor updates the balance and discloses v^t and u^t as before. In addition, for every $0 \le j < \ell$ and $\sigma \in \{0,1\}$, the vendor discloses s_j^σ conditioned on $p_j = \sigma$. Recall that given the ℓ keys $\langle s_{\ell-1}^{p_{\ell-1}} \ldots s_0^{p_0} \rangle$, the buyer can compute k^p whereas the rest of the sequence (i.e. k^z for $z \neq p$) remains pseudo-random. This implies the security of the protocol.

As for efficiency, we have that the $O(n+\ell)$ public-key operations of the previous protocol are reduced to $O(\ell)$ public-key operations, plus at most $n\ell$ private-key operations.[5] In terms of communication, both the buyer and the vendor need only to send $O(\ell)$ encryptions. In case the encrypted data is accessed by a PIR protocol (and not via broadcast), the costs of this protocol (both in terms of computation and communication) should be added to the above count. It is important to note though that here the PIR protocol is invoked on a database of optimal size (i.e. the same size as the database $x^0, \ldots, x^{n-1}$).

3.5.1 The Multi-Buyer Setting

The security of the above solution may be compromised in a scenario where the vendor interacts with several buyers and uses the same pseudo-random key sequence k^{p^j} for all buyers. For instance, a collusion of two malicious buyers can recover the entire sequence by having one buyer learn $s_0^0, \ldots, s_{\ell-1}^0$ and the other learn $s_0^1, \ldots, s_{\ell-1}^1$. One way around this problem is to use an independent key sequence for each buyer. However, for efficiency reasons it may be desirable for the vendor to distribute the same keys k^{p^j} to all buyers. In such a case, techniques from [20] can be used to efficiently resolve the security concern addressed above. For completeness we describe a specific implementation from [20] which relies on the Decisional Diffie-Hellman (DDH) assumption.

Let $(G_g, \cdot)$ be a group with a generator g for which the DDH assumption holds. In the modified protocol, the vendor lets $s_0^0, s_0^1, \ldots, s_{\ell-1}^0, s_{\ell-1}^1$ be independent and uniformly random indices between 1 and $|G_g|$, and for each price $p = p_{\ell-1} \circ \cdots \circ p_0$ computes $k^p = g^{\Pi_{j=0}^{\ell-1} s_j^{p_j}}$. In a transaction with a buyer B selecting price p, the vendor independently picks random indices $r_0^B, \ldots, r_{\ell-1}^B$, and uses each bit p_j as a selector between $s_j^0 \cdot r_j^B$ and $s_j^1 \cdot r_j^B$. In addition, the vendor sends the single group element $g^{1/(r_0^B r_1^B \cdots r_{\ell-1}^B)}$. Buyer B can now recover k^p by raising the latter group element to the (mod-$|G_g|$) product of the ℓ selected indices. The efficiency of this solution can be improved (without compromising its security) by setting $s_j^0 = 1$ for all j. We refer the reader to [20] for a security proof and for alternative solutions.

4 Subscription

Recall that our motivation for letting the buyer issue a "subscribe" request is to allow efficient one-way communication from the vendor to the buyer. The protocols from the previous section are clearly inadequate for this purpose. Indeed, a two-way interaction between the buyer and the vendor and a significant amount of computation are required during each transaction.

We would like to obtain a subscription protocol which allows the buyer to learn a sufficiently long (but not too long) *prefix* of the sequence of keys encrypting future information in the i-th channel. Upon unsubscription, a fresh set of keys will be used to render the unused part of the prefix irrelevant. We should also make sure that the buyer is debited appropriately. Finally, we would like the protocol to be very efficient, both for initializing a subscription and (even more so) for maintaining it.

We first describe, in Section 4.1, a simple solution achieving the above goals. A more efficient solution will be discussed In Section 4.2.

4.1 A Simple Solution

We will denote by L an upper bound on the length of a subscription, measured by the number of transactions. For notational convenience we let $L = B = 2^\ell$. However, a smaller L can be used to

[5] The number of private-key operations may be improved by arranging the prices in a TRIE structure.

enhance efficiency. We will need to assume that $|F|$ is larger than $B + L \cdot 2^\ell$ (where here 2^ℓ is an upper bound on a price p that even a dishonest $\mathcal{B}^*$ will be able to encrypt). This assumption, which automatically holds for most reasonable choices of parameters, serves to prevent a long subscription from causing a balance wraparound.

Our solution to the subscription problem proceeds as follows.

Subscribing. As in the previous protocol, $\mathcal{B}$ will send to $\mathcal{V}$ encryptions of the bits of $p = p^i$ and b (ℓ bits each). In addition, $\mathcal{B}$ will hold a value τ, $0 \leq \tau \leq L$, which is assumed to be a length of a prefix of the i-th channel it is *entitled* to buy, and will send encryptions of the ℓ bits of τ. An honest buyer can let $\tau = \lfloor b/p \rfloor$, regardless of the intended subscription length. The vendor will disclose a random key v and a random receipt u to $\mathcal{B}$ subject to the conditions that: (1) the bit representations of p, b, τ are valid; and (2) $(0 \leq)\tau \cdot p \leq b$. These conditions ensure that buyer is entitled to receive a prefix of length τ from a channel of price p. An efficient implementation of the latter disclosure, which requires some additional help from $\mathcal{B}$, will be described later. As before, the key v will be used to mask all of the received data in the current subscription, and the receipt u to cripple future transactions in a case of cheating. Finally, using the methods of the previous section, a key k^p encrypting the channel indexed by p (if any) should be disclosed to $\mathcal{B}$. We note that, as before, a dishonest buyer will never benefit from choosing a nonexistent price p.

Maintaining a subscription. Once the value of τ is guaranteed to be valid, maintaining a subscription is easy. As before, the entire contents of the i-th channel is masked (or encrypted) with $k^{p_i} + v$. However, in the t-th transaction following a subscription (where the initialization is considered the first transaction) it is additionally masked with a key v_t, which is disclosed subject to the condition $t \leq \tau$. Note that realizing this additional condition does not require any further help from $\mathcal{B}$, since τ does not change and the value t is public. Also note that here there are no conditional disclosures involving b.

Unsubscribing. If $\mathcal{B}$ unsubscribes from the channel after T billing periods, $\mathcal{V}$ deducts from its balance the amount $T \cdot p$ (note that this can be done efficiently from the public value T and the encrypted values of p, b). After L transactions, $\mathcal{V}$ automatically unsubscribes $\mathcal{B}$. By the abovementioned assumption on the size of F, we are guaranteed that a "negative" balance will not be interpreted as a positive one in the transaction subsequent to the unsubscription. If the buyer's balance turns negative (by failing to unsubscribe before depleting its balance), all its future transactions will automatically be crippled.

It remains to describe the implementation of the second conditional disclosure in the subscription procedure described above, namely a disclosure subject to the condition $\tau \cdot p \leq b$. The fact that the underlying field F is large allows to obtain much greater efficiency than that obtained by emulating a Boolean multiplication circuit. The disclosure procedure proceeds as follows. $\mathcal{B}$ will provide, as additional help, encryptions of $a_{\ell-1} = (\tau_{\ell-1}2^{\ell-1}) \cdot p, \ldots, a_0 = (\tau_0 2^0) \cdot p$. If $\mathcal{B}$ acts honestly, these should sum up to the product $\tau \cdot p$. To guarantee that each a_j is valid, observe that $\mathcal{V}$ can compute the two possible valid values of a_j, and disclose to $\mathcal{B}$ a mask subject to the condition that a_j is indeed consistent with the value of τ_j. That is, the j-th conjunct in the condition is of the form: $(\tau_j = 0 \wedge a_j = 0) \vee (\tau_j = 1 \wedge a_j = 2^j p)$. Finally, using the methods of the previous sections, an additional mask will be disclosed subject to the condition that $\sum a_j \leq b$ (note that an encryption of $\sum a_j$ can be computed by the vendor alone using the homomorphism). As in the previous section, this conditional disclosure will require $\mathcal{B}$ to send the encrypted bit representation of the sum. This concludes the description of the conditional disclosure procedure, and thus of the entire subscription protocol.

Efficiency. Both initializing a subscription and each subsequent transaction require $O(\ell)$ exponentiations, with communication consisting of $O(\ell)$ encryptions. In comparison to the implementation of a "buy" operation from the previous section, initializing a subscription is more expensive and maintaining it is significantly cheaper.

4.2 Efficiency Improvement

In a typical case where subscriptions are more frequently maintained than initialized, it is important to optimize the efficiency of the procedure for maintaining a subscription. In particular, it is desirable to avoid exponentiations altogether.

In the following we describe an implementation which achieves the above goal. Similarly to the protocol from Section 4.1, when initializing a subscription $\mathcal{B}$ will send to $\mathcal{V}$ encryptions of the bits of $p = p^i$, b, and τ. (As before, τ is supposed to be a length of a prefix from the i-th channel $\mathcal{B}$ is entitled to receive, and may be set to $\lfloor b/p \rfloor$). As before, the vendor will disclose a random key v and a random receipt u to $\mathcal{B}$ subject to the conditions that: (1) the bit representations of p, b, τ are valid; and (2) $(0 \leq)\tau \cdot p \leq b$, i.e. the buyer is entitled to receive a prefix of length τ of the stream. The key v will be used to mask the current subscription, and the receipt u to cripple future transactions in case of cheating.

In the core of the optimized solution is an efficient subprotocol, performed during the subscription initialization stage, which allows $\mathcal{B}$ to effectively learn a prefix of length τ from a pseudo-random key sequence of length L. This subprotocol extends [19] (see Section 2.3) and may be of independent interest.

For the description of the prefix retrieval subprotocol, it is convenient to associate with each integer $0 \leq t < L$ the subset $T(t) \subseteq \{0, \ldots, \ell - 1\}$ corresponding to its binary representation. That is, $j \in T(t)$ iff the coefficient of 2^j in the binary representation of t is equal to 1. From now on we will freely interchange between integers and their corresponding sets. For any set T, the *cone* defined by T, denoted C_T, is the set of t's corresponding to all *subsets* of T. That is,

$$C_T \stackrel{\text{def}}{=} \{0 \leq t < L \ : \ T(t) \subseteq T\}.$$

In addition, for any $0 \leq t < L$, we will let $P_t \stackrel{\text{def}}{=} \{0, 1, \ldots, t-1\}$ denote the *t-prefix* of $\{0, 1, \ldots, L-1\}$. An important observation is that every prefix P_t can be expressed as the union of at most ℓ cones. Indeed,

$$P_t = \bigcup_{j : t_j = 1} C_{T(t,j)}$$

where $T(t,j) \stackrel{\text{def}}{=} \{0, 1, 2, \ldots, j-1\} \cup \{j+1 \leq j' < l \ : \ t_{j'} = 1\}$. This can be verified by inspection: It is clear that if $T(t') \in C_{T(t)}$ then $t' < t$ (since in the most significant bit in which t' differs from t we must have $t_j = 1$ and $t'_j = 0$). Conversely, for any $t' < t$ there is a bit j such that $t'_j = 0, t_j = 1$, and $t'_{j'} = t_{j'}$ for all $j' > j$ (j is the most significant bit in which t, t' differ). For such j, $t' \in C_{T(t,j)}$.

We turn to describe a *cone retrieval* protocol allowing an *honest* $\mathcal{B}$ to retrieve a set of pseudo-random keys indexed by a selected cone C_T while learning no information (in the computational sense) on the remaining keys, and while keeping T private from $\mathcal{V}$. To retrieve the cone C_T, where $T = T(c)$:

- $\mathcal{B}$ sends encryptions of the ℓ bits of c.
- $\mathcal{V}$ prepares ℓ keys $s_0, \ldots, s_{\ell-1}$, and discloses each key s_j under the condition $c_j = 1$. The sequence $\mathbf{k} = (k_0, \ldots, k_{L-1})$ will be defined by

$$k_m = \bigoplus_{j : m_j = 1} f_{s_j}(m).$$

(Note that the key k_0 is identically 0, and will thus always be disclosed. However, the remaining keys of $\mathbf{k}$ are pseudo-random.) To see that this protocol indeed satisfies the required properties for an honest $\mathcal{B}$, observe that: (1) For any $T' \subseteq T(c)$, where $T' = T(c')$, $\mathcal{B}$ can recover $k_{c'}$ from the keys s_j it received; (2) If the ℓ functions f_{s_j} were replaced by ℓ perfectly random functions, then the pads belonging to the set T would be completely independent from the keys k_m whose indices are not in C_T.

It is important to note that the sequence $\mathbf{k}$ does not have to be pre-computed or stored. At time t, the t-th entry of this sequence can be very efficiently computed "on the fly" both by $\mathcal{B}$ (if the t-th entry is in the cone) and by $\mathcal{V}$.

Putting the Pieces Together. We now describe how to combine the ingredients we have so far: (1) a skeleton for a subscription initialization protocol in which $\mathcal{B}$ wants to buy a prefix of length τ; (2) the above cone retrieval protocol; and (3) the observation that the prefix P_τ can be expressed as the union of at most ℓ cones, whose structure is simple to define in terms of the binary representation of τ. Recall that the vendor is holding encryptions of the bits of b, p, τ, and needs to disclose: (1) keys opening the τ-prefix of a channel having price p; and (2) a pair of keys (a "mask" and a "receipt") for ensuring that if τ is invalid, then the current and future transactions will be crippled. The second disclosure can be handled as in Section 4.1. In the following we focus on the first disclosure, which affects both the initialization of a subscription and its maintenance.

First, by the methods of the previous sections, the vendor will disclose, based on the encryptions of p, a single key which is used to encrypt the p-th channel. This handles the item selection aspect of the problem and allows us to focus on the more involved time dimension.

We will not be able to *directly* disclose the τ-th prefix of a pseudo-random key sequence (unless a certain relaxing assumption is made, see below). Instead, $\mathcal{V}$ will disclose to $\mathcal{B}$ an appropriate selection of at most ℓ cones, determined by the bit representation of τ, where each cone will use an independent key sequences $\mathbf{k}^j$. The key for the t-th transaction will be separately masked with each of the ℓ keys corresponding to time t in the ℓ sequences $\mathbf{k}^j$. Note that to maintain a subscription, $\mathcal{V}$ will have to compute all of these ℓ keys at each time t, and $\mathcal{B}$ at most a single key. To implement the retrieval of the appropriate collection of cones, the right length-ℓ representation of each cone has to be supplied to the vendor in an encrypted form. To this end, we observe that the decomposition of a prefix into cones has the following structure: if the j-th bit of τ is 0, then an "empty" cone can be disclosed (without $\mathcal{V}$ knowing that this is the case). Otherwise, the bit representation of the cone is identical to τ in its $j-1$ leftmost entries, equal to 0 in the j-th entry, and equal to 1 in the rest. Therefore, the disclosure of the cones (with keys $s_0, \ldots, s_{\ell-1}$) can be implemented quite efficiently: to disclose the j-th cone, $\mathcal{V}$ has to disclose each key $s_{j'}$ where $j' < j$ subject to the condition that $\tau_j = 1$ and each key $s_{j'}$ with an index $j' > j$ subject to the condition that $(\tau_j = 1 \wedge \tau_{j'} = 1)$.

Efficiency. Altogether, the communication and work in the initialization stage of the above solution are quadratic in ℓ, and those of each future transaction are linear in ℓ. However, an important advantage over the simpler solution of Section 4.1 is that maintaining a subscription only requires private-key operations (specifically, $O(\ell)$ such operations per transaction). Thus, the current solution is preferable to the previous one when the length of a subscription is expected to be large (say, larger than ℓ).

We note that while we use the same parameter ℓ to denote possibly different quantities, the value of ℓ here is typically very small: it is the logarithm of a public upper bound on the number of future billing periods before the subscription is automatically terminated. However, in a case that this overhead is too large, the following relaxation may provide a much more efficient alternative. Suppose that the buyer knows in advance that it will unsubscribe before 2^j transactions, where $2^j p \leq b$. Then, by letting $\tau = 2^j$, a much simpler prefix retrieval procedure can be employed. Namely, the single cone defined by the set $\{0, 1, \ldots, j\}$ is precisely the prefix P_{2^j}. This allows to reduce the cost of the above initialization procedure to be linear in ℓ, and the communication in a future transaction to consist of a single key sent to $\mathcal{B}$.

5 One-Round Oblivious Transfer

Oblivious Transfer (OT) [25, 10, 4] may be viewed as the simplest atomic building block for general secure computation [15]. OT is a 2-party protocol between Alice and Bob. In its most common variant, also known as $\binom{2}{1}$-OT, Alice holds a selection bit b and Bob holds a pair of secrets x^0, x^1. At the end of the protocol, Alice should output x^b and learn no information on x^{1-b}, and Bob should output and learn nothing.

As a special case of our general methodology, we obtain an efficient 1-round OT protocol which satisfies a *reasonable* security definition. In contrast to a previous protocol of [1], whose security relies on the use of a random oracle, the security of our protocol can be based on a standard computational assumption (DDH). A similar protocol (and definition) has been independently proposed by Naor and Pinkas [21].

Our $\binom{2}{1}$-OT protocol naturally extends into a more general $\binom{n}{1}$-OT protocol (where Alice retrieves one of n secrets held by Bob). We therefore directly describe our solution in this setting.

5.1 $\binom{n}{1}$-OT Protocol

Each transaction of a priced oblivious transfer protocol trivially implies an OT protocol. However, in our one-round implementations of such a transaction we assumed an *initialization phase*, which is not part of the setting in a stand-alone OT protocol. In fact, one part of the initialization phase will also be part of our OT protocol: Alice still needs to sample a public-key, secret-key pair (k, sk) and send the public-key k to Bob. Moreover, Bob still needs to verify that k is valid. However, in this case Alice cannot prove that k is valid (there is just not enough interaction). We therefore assume that the underlying homomorphic encryption scheme enjoys the *verifiability* property discussed in Section 2.3, as is the case for the El-Gamal scheme. For such an encryption scheme, Bob can verify on its own that k has a corresponding secret key sk (although Alice may not know this key). We can now define our basic $\binom{n}{1}$-OT protocol:

Alice invokes G to sample a public-key, secret-key pair (k, sk). She then sends to Bob the public-key k and a random encryption $c = E_k(i)$ of i.

Bob verifies that k is a valid public key and c is a valid encryption. In such a case, for every $j \in [n]$, Bob computes m^j which is the conditional disclosure of x^j conditioned on $j = i$ (i.e. m^j is a random encryption of $\alpha^j(i-j) + x^j$ for a uniformly distributed element α^j of F). Bob sends $m^0, \ldots, m^{n-1}$.

Alice decrypts $m_i = E(x^i)$ and outputs x^i.

Security. Various definitions of security for OT have been proposed. The most widely accepted are those relying on a general framework for defining secure two-party computation (cf., [6, 12]). We are unable to obtain this level of security while preserving the minimal number of rounds in our protocol. In a nutshell, the security definition satisfied by the above protocol relaxes the simulation-based definition of [6, 12] in two ways. First, the simulator for Alice is allowed to be computationally unbounded (yet its simulation quality is perfect or statistical rather than computational). This may be interpreted as saying that Bob's security is *purely* information theoretic. Second, the simulator for Bob should simulate Bob's view alone, without considering its correlation with Alice's output. In particular, we do not require that a cheating Bob *knows* the input to which Alice's selection effectively applies. We feel however that the notion of security we achieve is perfectly suitable for OT, either as a standalone application, or in more general "information-retrieval" contexts such as the one studied in this work. Next we analyze the security of the above protocol.

The view of a possibly cheating Bob only contains a random public-key and a (random) encryption. Therefore, the semantic security of E implies that this view can be simulated. The view of a possibly cheating Alice (even an unbounded one) can be perfectly emulated by an *unbounded* simulator. The simulator first computes the private key sk that corresponds to k (if such a key does not exist, Bob would refuse to interact with Alice). Note that this requires the simulator to be unbounded. Now there exists at most a single i for which $c = E_k(i)$. If such an i exists the simulator queries for x^i and defines m^i to be a random encryption of x^i. For all other j, the simulator defines m^j to be a random encryption of a random element. It is easy to verify that this is a perfect simulation.[6]

Efficiency and Improvements. Alice's work consists of sampling a key and a constant number of public-key operations. Bob performs $O(n)$ public-key operations and its message contains n encryptions. However, the improvements in efficiency that are described in Sections 3.2 and 3.5 apply also in the contexts of the OT protocol. First, Alice can just retrieve m^i (which is the only interesting value for her) using a PIR protocol (which can be carried out without adding rounds). In addition, both the communication and Bob's work can be reduced by applying the method described in Sections 3.5 (that uses the Naor-Pinkas pseudo-random sequence).

Acknowledgments. We wish to thank Avi Rubin for discussions which initiated this work, and Tal Malkin, Moni Naor, and Benny Pinkas for helpful discussions and pointers. We also thank the anonymous referees for their comments.

References

[1] M. Bellare and S. Micali. Non-interactive oblivious transfer and applications. CRYPTO '89, pp. 547-557, 1989.

[2] O. Berthold, H. Federrath, and M. Kohntopp. Project "Anonymity and Unobservability in the Internet". Workshop on Freedom and Privacy by Design / Conference on Freedom and Privacy, pp. 57-65, 2000.

[3] F. Boudot. Efficient proofs that a committed number lies in an interval. EUROCRYPT 2000, pp. 431-444. Springer-Verlag, 2000.

[4] G. Brassard, C. Crépeau, and J.-M. Robert. All-or-nothing disclosure of secrets. CRYPTO '86, pp. 234-238, 1987.

[5] C. Cachin, S. Micali, and M. Stadler. Computationally private information retrieval with poly-logarithmic communication. EUROCRYPT '99, pp. 402-414, 1999.

[6] R. Canetti. Security and composition of multiparty cryptographic protocols. *J. of Cryptology*, 13(1), 2000.

[7] D. Chaum. Security without identification: Transaction systems to make big brother obsolete. *Communications of the ACM*, 28(10):1030-1044, 1985.

[8] D. Chaum, A. Fiat, and M. Naor. Untraceable electronic cash. CRYPTO '88, pp. 319-327

[6] In this we assume that the verification process of E is never wrong (as is the case for the El-Gamal encryption). We also assume that the secrets are group elements (otherwise the simulated view can only be made statistically close to the actual view).

[9] B. Chor, O. Goldreich, E. Kushilevitz, and M. Sudan. Private information retrieval. FOCS '95, pp. 41-51, 1995.

[10] S. Even, O. Goldreich, and A. Lempel. A randomized protocol for signing contracts. *C. ACM*, 28:637-647, 1985.

[11] Y. Gertner, Y. Ishai, E. Kushilevitz, and T. Malkin. Protecting data privacy in private information retrieval schemes. *JCSS*, 60(3):592-629, 2000. Preliminary version in STOC '98.

[12] O. Goldreich. Secure multi-party computation. Available at http://philby.ucsb.edu/cryptolib/BOOKS, February 1999.

[13] O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game. STOC '87, pp. 218-229, 1987.

[14] Y. Ishai and E. Kushilevitz. Private simultaneous messages protocols with applications. STOC '97, pp. 174-183, 1997.

[15] J. Kilian. Basing cryptography on oblivious transfer. STOC '98, pp. 20-31, 1988.

[16] E. Kushilevitz and R. Ostrovsky. Replication is not needed: Single database, computationally-private information retrieval. FOCS '97, pp. 364-373, 1997.

[17] E. Mann. Private access to distributed information. Master's thesis, Technion - Israel Institute of Technology, Haifa, 1998.

[18] D. Naccache and J. Stern. A new public key cryptosystem. EUROCRYPT '97, pp. 27-36, 1997.

[19] M. Naor and B. Pinkas. Oblivious transfer and polynomial evaluation. STOC '99, pp. 245-254.

[20] M. Naor and B. Pinkas. Oblivious transfer with adaptive queries. CRYPTO '99, pp. 573-590.

[21] M. Naor and B. Pinkas. Efficient oblivious transfer protocols. SODA 2001.

[22] M. Naor and O. Reingold. Number theoretic constructions of efficient pseudo-random functions. FOCS '97.

[23] T. Okamoto and S. Uchiyama. A new public key cryptosystem as secure as factoring. EUROCRYPT '98, pp. 308-318, 1998.

[24] P. Pallier. Public-key cryptosystems based on composite degree residuosity classes. EUROCRYPT '99, pp. 223-238, 1999.

[25] M. O. Rabin. How to exchange secrets by oblivious transfer. Technical Report TR-81, Harvard Aiken Computation Laboratory, 1981.

[26] S. von Solms and D. Naccache. On blind signatures and perfect crimes. *Computers and Security*, 11(6):581-583,1992.

[27] J. P. Stern. A new and efficient all-or-nothing disclosure of secrets protocol. ASIACRYPT '98, pp. 357-371, 1998.

[28] A. C. Yao. How to generate and exchange secrets. FOCS '86, pp. 162-167, 1986.

A Security Requirements

We first address the scenario of a buyer who only issues "buy" requests. This should be viewed as a conservative default scenario, which we later extend to allow enhanced efficiency. A protocol $(\mathcal{B}, \mathcal{V})$ in this setting proceeds as follows. A security parameter 1^κ is given as public input. We assume that subsequent "inputs" are dynamically chosen by $\mathcal{B}, \mathcal{V}$ as the protocol proceeds. This simplifies our notation and eliminates some extra quantification. Initially, $\mathcal{B}$ chooses an initial deposit amount $b^{(0)}$ as a public input, and the parties engage in some initialization sub-protocol. The remainder of the protocol is divided into transactions, where at the beginning of each transaction: (1) $\mathcal{V}$ may choose a private database $\mathbf{x}$ such that $x^0 = 0$ and a public input $\mathbf{P}$ containing a price list $\mathbf{p}$ such that $p^0 = 0$; (2) $\mathcal{B}$ may subsequently choose the index of an item i, possibly depending on $\mathbf{P}$. At the end of each transaction, $\mathcal{B}$ produces a local output. We assume that the protocol terminates after a polynomial number of transactions, depending only on κ.

The expected (honest) behavior of the parties is defined as follows: An *honest* vendor's choices are restricted to be non-adaptive, i.e. may depend on the transaction number but not on previous history, whereas the choice of an honest buyer may depend on its outputs in previous transactions and on all public inputs (including $b^{(0)}$). Moreover, an honest $\mathcal{B}$ is restricted to choose i such that the total prices of all p^i up to (and including) the current transaction does not exceed the initial deposit amount $b^{(0)}$.

A protocol $(\mathcal{B}, \mathcal{V})$ as above is considered *secure* if it satisfies the following requirements:

Correctness. If both $\mathcal{B}$ and $\mathcal{V}$ are honest, then $\mathcal{B}$ outputs the correct item x^i at the end of each transaction.

Buyer's security. A malicious vendor should not learn the choices made by an honest buyer. Formally, we make the following requirement. For every efficient (and possibly malicious) $\mathcal{V}^*$, there exists an efficient simulator $\mathcal{S}_{\mathcal{V}^*}$, such that for every honest $\mathcal{B}$,[7] the following distributions are computationally indistinguishable: (1) The output of $\mathcal{S}_{\mathcal{V}^*}(1^\kappa, b^{(0)})$, where $b^{(0)}$ is as generated by $\mathcal{B}(1^\kappa)$; (2) The view of $\mathcal{V}^*$ in the interaction $(\mathcal{B}, \mathcal{V}^*)(1^\kappa)$. We note that this requirement is weaker than that of general security definitions in that it does not address the effect $\mathcal{V}^*$ may have on the output of $\mathcal{B}$. In particular, $\mathcal{V}^*$ does not need to "know" a database x which is effectively determined by its strategy in a given transaction. This is consistent with other definitions of related primitives (such as PIR, see Section 2.3, or even some definitions of oblivious transfer).

Vendor's security. A malicious buyer should not obtain more information than what its initial deposit allows. This is formalized by comparing the interaction of $\mathcal{B}^*$ with an honest vendor $\mathcal{V}$ to an "idealized" interaction of a simulator $\mathcal{S}$ with the inputs produced by $\mathcal{V}$. In the idealized interaction, $\mathcal{S}$ is free to adaptively access a single information item at each transaction, as long as the total cost of these items is bounded by an initial deposit amount. Specifically, $\mathcal{S}$ first runs $\mathcal{B}^*$ to produce a deposit amount $b^{(0)}$. Subsequently, at each transaction t, $\mathcal{S}$ explicitly requests for an item i and receives x^i in return. The database $\mathbf{x}$ and its public information $\mathbf{P}$ are chosen according to the (non-adaptive) strategy of $\mathcal{V}$, and the choices of i by $\mathcal{S}$ may be adaptive. If at any stage the price p^i is greater than the remaining balance of $\mathcal{S}$, its request and all following requests will be denied. At the end of the interaction, $\mathcal{S}$ outputs $b^{(0)}$ and an arbitrary function of its entire view (where this view consists of public information and at most a single item from each $\mathbf{x}$).

The vendor's security requirement may now be defined as follows. For every efficient (possibly malicious) $\mathcal{B}^*$, there exists an efficient simulator $\mathcal{S}_{\mathcal{B}^*}$, such that for every honest $\mathcal{V}$, the output of $(\mathcal{S}_{\mathcal{B}^*}, \mathcal{V})(1^\kappa)$ in an idealized interaction as described above is computationally indistinguishable from the output of $(\mathcal{B}^*, \mathcal{V})(1^\kappa)$.

[7] The quantification here is over different strategies for choosing the inputs, replacing a standard quantification over inputs.

Some variants of our construction satisfy stronger versions of this definition. One of our main variants satisfies the above definition even for an unbounded $\mathcal{B}^*$, and with computational indistinguishability replaced by *statistical* indistinguishability. Moreover, in our construction of a 1-round oblivious transfer protocol, this is further strengthened to *perfect* emulation.

We now address the more general case of a buyer who may take any of the four actions specified in Section 2.1. The above protocol framework is augmented by letting $\mathcal{B}$ choose an additional *public* input at each transaction, specifying the *type* of action it takes. Note that since the sole purpose of the different action types is to reduce the amount of interaction and work performed by both parties, we cannot expect to keep the action private. The vendor's security requirement remains unchanged, and can be defined as above. The buyer's security in this setting is inherently problematic, even in an idealized model. If the buyer's action type depends on the received data (say, it quits a subscription to a channel after its content has been corrupted), then the vendor may infer information on the identity of the selected item. Thus, even in an idealized model, a buyer cannot be completely safe unless its selection of action type depends only on the transaction number t. Yet, the above definition of buyer's security can be modified to capture the requirement that whatever information $\mathcal{V}^*$ learns in its interaction with $\mathcal{B}$, it could have also inferred from its (private and public) inputs and from $\mathcal{B}$'s public inputs alone.

Our constructions fall short from meeting the above requirement. This is a consequence of the fact that we do not guarantee that $\mathcal{V}^*$ "knows" which database $\mathbf{x}$ it is effectively offering at a given transaction. However, our solutions do provably meet the following reasonable relaxations. First, if the buyer's action type is *oblivious* to the received items, i.e. depends only on public data (yet its specific selections i may also depend on received items), then our general solutions satisfy the above buyer's security requirement. Second, this requirement is also satisfied if the vendor is *honest but curious*, i.e. if it honestly follows the protocol but tries to infer information on the buyer's requests from its view.

While a vendor may infer a significant information on the requests of a buyer who adaptively chooses its action *type*, we argue that the security offered by our solutions is very reasonable even in this case. First, a malicious vendor trying to violate the security of a specific user will be unlikely to disrupt service or modify the transmitted information (which may be authenticated). In doing so, the vendor's faultiness is detected, and it may therefore suffer reputation loss. Second, and perhaps more importantly, it is always up to the *buyer* to decide how much of its security it is willing to compromise by adaptively choosing the action types. We stress again that, at a cost of efficiency loss, an extra-cautious buyer may completely avoid the above concerns by using only "buy" operations.

Finally, while we do not explicitly address issues of robustness or recovery from faults, these can be handled in a straightforward manner. Again, our underlying assumption is that it is less important (and in a sense futile) to try to protect buyers from those faults which they can detect, namely from the vendor failing to deliver the goods. Given that a fault has occurred on the vendor's part, it is up to the buyer to decide whether to engage in an arbitration procedure, possibly compromising some of its privacy, or to continue as if nothing has happened.

B Homomorphic Encryption

In the following we provide some more details on homomorphic schemes and the precise way in which our relevant notation should be interpreted.

A public-key cryptosystem consists of a key generation algorithm G, an encryption algorithm E and a decryption algorithm D. $G(1^\kappa)$ outputs a pair (k, sk), consisting of a *public* key and a *secret* key, respectively. The following discussion applies to any *valid* pair of keys (k, sk) possibly produced by G. The issue of validity will be discussed later.

The public key k determines a homomorphism group $(H, +)$ of a prime order Q, along with a generator g and a representation of each $h \in H$ by a string of length κ. The representation is such that the group operation can be efficiently computed. A plaintext $h \in H$ is encrypted by invoking $E(k, h)$ (where h is encoded using its representation). A ciphertext c is decrypted by invoking $D(sk, c)$. By $E_k(h)$ we may denote, depending on the context, either: (1) The output distribution of $E(k, h)$ (also referred to as a *random encryption of* h); or (2) A *specific* ciphertext encrypting h. We only consider error-free variants of encryption. That is, for any distinct $h_1, h_2 \in H$, the supports of $E(k, h_1)$ and $E(k, h_2)$ must be disjoint. Finally, for the cryptosystem to be homomorphic it must satisfy the following: from any pair of ciphertexts $E_k(h_1)$ and $E_k(h_2)$, one may efficiently compute (using k) a *random* encryption $E_k(h_1 + h_2)$.

While encryption as defined above is applied to strings representing elements of H, we will actually need to encrypt two potentially different types of objects: elements of Z_Q, and *arbitrary* strings of some fixed length. In the simple case where the representation of H is equal to the natural (binary) representation of Z_Q, both problems are easy: a string σ of an appropriate length is encrypted by directly applying E, and an index $j \in Z_Q$ is encrypted by applying E to its binary representation.

We now explain the more general convention, which is needed for "multiplicative" schemes such as El-Gamal. To this end we temporarily depart from our default notation and use multiplicative group notation. An integer j (or element $j \in Z_Q$) is encrypted by applying $E(k, g^j)$ (where g is the specified generator of H). This allows efficient encryption, but does not necessarily allow an efficient reconstruction of j from the decryption of $E(k, g^j)$. As it turns out, this extra property is not needed for our purposes.[8]

In the case of encrypting a string, efficient reconstruction will be important. If we are guaranteed that all strings we need to encrypt are representations of elements from H, then we can encrypt by directly applying E to the string. It will be conceptually convenient to assume that this is always the case. However, even if this is not the case, the problem can be solved using hashing: to encrypt a string x, apply E to a random element $r \in H$, and attach to this encryption the exclusive-or of x with the hashed value of r. We refer the reader to [22] for more details.

[8]In our constructions, the buyer knows the values of all encrypted *integers* j.