J. Math. Crypt. **3** (2009), 37–68

DOI 10.1515 / JMC.2009.003

# One-round secure comparison of integers

Ian F. Blake and Vladimir Kolesnikov

Communicated by Ronald Cramer

**Abstract.** We consider the problem of securely evaluating the Greater Than (GT) predicate and its extension – transferring one of two secrets, depending on the result of comparison. We generalize our solutions and show how to securely decide membership in the union of a set of intervals. We then consider the related problem of comparing two *encrypted* numbers. We show how to efficiently apply our solutions to practical settings, such as auctions with the semi-honest auctioneer, proxy selling, etc. All of our protocols are one round.

We propose new primitives, *Strong Conditional Oblivious Transfer* (SCOT) and *Conditional Encrypted Mapping* (CEM), which capture common security properties of one round protocols in a variety of settings, which may be of independent interest.

**Keywords.** Two millionaires, encrypted inputs, auctions, strong conditional oblivious transfer, conditional encrypted mapping.

**AMS classification.** 94A60.

## 1 Introduction

This work falls into the area of constructing efficient secure multi-party protocols for interesting functionalities. The more basic the functionality, the more common is its use, and the more significant is the value of any improvement of the corresponding protocol. We start with presenting the problems we investigate and their motivation.

The functionality we consider – Greater Than (GT) – is one of the most basic and commonly used. Numerous applications rely on efficient secure evaluation of GT. Consider the following scenarios.

*Secure distributed database mining.* The setting is as follows: several parties, each having a private database, wish to determine some properties of, or perform computations on, their *joint* database. Many interesting properties and computations, such as transaction classification or rule mining, involve evaluating a large number of instances of GT [21, 25]. Because of the large size of the databases, even a minor efficiency gain in computing GT results in significant performance improvements.

*Computing on intervals.* Consider the problem of determining whether a point belongs to the union of a set of intervals. (Sometimes we will refer to this problem as the membership in a set of intervals.) It arises in appointment scheduling, flexible timestamp verification, expression evaluation, biometrics, and many others. Certain kinds of set membership problems, as studied by Freedman, Nissim and Pinkas [16], can be represented succinctly as instances of problems we consider. For example, the problem of membership in a set consisting of all even integers on a large interval $(y, z)$

Preliminary versions of contributions composing this work previously appeared in [3, 4].

can be represented as a conjunction of two small instances of interval memberships ($S = \{x | x_0 < 1 \wedge x \in (y, z)\}$, where $x_0$ is the low bit of $x$). In such cases, using our solutions may have significant advantages over the general set intersection solution of [16].

*Auctions and Bargaining.* With the continuing expansion of the Internet, electronic commerce and especially online auctions continue to grow at an impressive pace. Many sellers also discover the appeal of flexible pricing. For example, sites such as `priceline.com` ask a buyer for a price he is willing to pay for a product, and the deal is committed to if that price is greater than a certain (secret) threshold.

In many such situations, it is vital to maintain the privacy of bids of the players. Indeed, revealing an item's worth can result in artificially high prices or low bids, specifically targeted for a particular buyer or seller. While a winning bid or a committed deal may necessarily reveal the cost of the transaction, it is desirable to keep all other information (e.g. unsuccessful bids) secret.

There has been a large amount of work dedicated to ensuring privacy and security of online auctions and haggling (e.g., [5, 9, 15, 28]). Our work complements, extends, and builds on it. We discuss the Private Selective Payments protocols of Di Crescenzo [9] and show how our improvements benefit this application.

*Server-aided transactions.* It is often beneficial to both sellers and buyers to employ a mutually semi-trusted server $S$ to assist them in their transaction. The use of such a server simplifies secure protocol design, allowing for more efficient protocols. It allows the seller to be offline most of the time, allowing $S$ to act on behalf of the seller in handling bid requests. Further, a reputable $S$ (such as eBay) may provide additional assurance of security to the potential buyer. However, since sellers and buyers wish to hide their inputs from $S$, the latter must perform operations, such as comparisons, on *encrypted* numbers. We propose a formalization of this setting, as well as new, more efficient GT protocols for it.

Other applications that rely on secure evaluation of GT might need to employ a proxy server $S$, as above; if so, our work improves their performance as well.

## 1.1 Our setting

All our protocols are one round. That is, the first party sends a message, receives a reply from the second party, and outputs the result of the computation. We will refer to the first party as the *receiver* $R$, and to the second party as the *sender* $S$.

We note that the one-round setting with computationally unbounded receiver is appealing. Numerous papers consider unconditional security against one or more parties, in particular, the receiver, e.g. [6, 7, 14, 20, 28]. Practical one-round computation with unbounded receiver currently seems to be hard to achieve. The best currently known general approach [8, 22, 23, 33] is quite inefficient and only works for $NC^1$ circuits. At the same time, if the receiver is assumed to be polytime bounded, we could use the very efficient Yao's garbled circuit approach [26, 28, 32, 34] at a cost linear with the size of the circuit. We remark that our solutions are in the more difficult setting (unbounded receiver), while achieving performance only slightly worse than the best known approach in the easier (polytime bounded receiver) setting.

When discussing the comparison of encrypted numbers in Section 5, we find it convenient and more general to introduce a semi-honest "crypto computer" server $S$. Upon the receipt of encrypted inputs, $S$ produces an output from which (only) the result of comparison can be obtained, given the decryption keys. We note that this setting is a natural generalization of the one-round two-party case, since the sender (of the two-party case) can simply run the algorithm of such a server. See the beginning of Section 5 for more discussion and justification.

## 1.2 Contributions and outline of the work

After discussing related work and preliminaries in Sections 2 and 3, we approach the problem of *one round* secure evaluation of GT in two settings. In Section 4, we consider the standard two-party setting, where the parties "know" their respective inputs. In contrast, in Section 5, we consider a more complicated setting, where the parties only possess *encryptions* of their inputs, but not plaintexts. We present our solutions as variants of Oblivious Transfer (OT) protocols, where one of the two secrets is transferred to the receiver, depending on the value of the GT predicate. We concentrate on the setting where the receiver is computationally unbounded. We give corresponding definitions, comparisons with previous work, and discuss practical applications, such as auctions.

*GT with Plaintext Inputs.* We start with a definitional discussion, and introduce Strong Conditional Oblivious Transfer (SCOT), which is a variant of OT noted in the previous paragraph. (We will write $Q$-SCOT to make explicit the basis predicate $Q$). In Section 4.2, we present our main tool – an efficient protocol for computing GT-SCOT built from a homomorphic encryption scheme. We exploit the structure of the GT predicate in an elegant and novel way to arrive at a solution that is more efficient and flexible than the best previously known (of Fischlin [15] and Di Crescenzo [9]) for our model with unbounded receiver. Additionally, our construction is the first to offer transfer of $c$-bit secrets, with $c \approx 1000$ for practical applications, at no extra cost, with *one* invocation of the protocol, as opposed to the necessary $c$ invocations of previous protocols. This results in additional significant efficiency gains.

In Section 4.3, we show how to use the bandwidth of our GT-SCOT solution and present protocols for efficiently computing SCOT based on the interval membership (which we call I-SCOT) and SCOT based on the membership in a union of $k$ intervals (which we call $k$-UI-SCOT). Because of their modularity, these protocols can also be constructed based on Fischlin's [15] and Di Crescenzo's [9] solutions at the efficiency loss described in the previous paragraph. Because they leak the private inputs of the sender, we do not know of an efficient way to extend solutions of [10] to compute these functionalities. We remark on how to use UI-SCOT to compute the conjunction or disjunction of the memberships in unions of intervals. We compare and summarize resource requirements of schemes of Fischlin, Di Crescenzo, Di Crescenzo et al., and ours in the Table in Section 4.4.1.

*GT with Encrypted Inputs.* In Section 5.1, we shift our attention to the setting where players only possess encryptions of their inputs. As justified in Section 1.1, we consider the setting where clients send their encrypted inputs to a semi-honest "crypto computer" $S$, who produces an output that can be decoded by the clients. To enable formal discussion of crucial parts of our protocols in a number of settings simultaneously, we extract what these settings have in common – the following requirements on the output of $S$: it allows efficient reconstruction of the value of the function, given the decryption key, and does not contain any other information. This allows to postpone the (easy but tedious) discussion of setting-specific clients' privacy requirements. We formalize (Definition 5.1) a special case of this notion, which we call *Conditional Encrypted Mapping* (CEM). In $Q$-CEM, $S$ has two secrets $s_0, s_1$, is given encryptions of two values $x, y$, and outputs a string that only allows reconstruction of $s_{Q(x,y)}$, where $Q$ is a fixed public predicate. We note that our statistical privacy requirement on the output of $S$ is strong, e.g., precluding Yao's garbled circuit-based solutions.

We propose two new, more efficient CEM protocols for the GT predicate (Section 5.1). Note that our protocol of Section 4 requires $S$ to know one of the compared numbers, and thus cannot be naturally cast as a CEM. We overcome this with a new tool – a randomized way to represent secrets to be transferred by $S$ (presented in Section 5.1.2). The cost of CEM protocols is comparable to SCOT protocols of Section 4. We believe this method may be used to improve efficiency of other constructions relying on homomorphic encryptions.

In Section 5.2, we show how our constructions result in new, more efficient, protocols for the examples of private selective payments of Di Crescenzo [9] and proxy selling. We discuss methods of protection against malicious behavior of parties (Section 5.2.1) and show that efficient CEM schemes exist for any $NC^1$ predicate (Section 5.1.6).

Section 5.3 summarizes and compares resource requirements of schemes based on the work of Di Crescenzo [9], Fischlin [15], Laur and Lipmaa [24] and ours. (It appears clearer to present comparisons with previous work separately for our GT-SCOT and GT-CEM protocols, because of the differences among the two problems.)

## 2 Related work

We discuss related work in both directions of our contributions – definition of SCOT and CEM, as well as concrete protocols for GT and auction-like functionalities.

### 2.1 Variants of SCOT and CEM

Several notions similar to SCOT and CEM were previously proposed.

The notion of Conditional Oblivious Transfer (COT) was introduced by Di Crescenzo, Ostrovsky and Rajagopalan [10] in the context of timed-release encryption. It is a variant of Oblivious Transfer (OT) [31]. Intuitively, in COT, the two participants, a receiver $R$ and a sender $S$, have private inputs $x$ and $y$ respectively, and share a public predicate $Q(\cdot,\cdot)$. $S$ has a secret $s$ he wishes (obliviously to himself) to transfer to $R$ iff $Q(x, y) = 1$. If $Q(x, y) = 0$, no information about $s$ is transferred to $R$. $R$'s private input and the value of the predicate remain computationally hidden from $S$.

A similar notion to COT, Conditional Disclosure of Secrets (CDS), was introduced by Gertner, Ishai, Kushilevitz and Malkin [18] in the context of multi-server Symmetrically Private Information Retrieval (SPIR). In their work, the receiver of the secret apriori knows the inputs of the (many) senders. The secret is unknown to the receiver and sent to him only if a predicate holds on the inputs.

Aiello, Ishai and Reingold [1] adapt CDS into the single server setting, where the (single) sender holds *encryptions* of parts (i.e. bits) of input. The receiver knows both the input and the decryption key. As in COT and CDS, the receiver does not know the secret; it is sent to him only if a predicate holds on the input.

Laur and Lipmaa [24] extend the study of CDS for the case of additive homomorphic encryptions and give generic constructions and specific protocols.

The lack of requirement of privacy of the value of $Q(x, y)$ and the sender's input often prevents the use of COT or CDS as a building block of other protocols. Di Crescenzo [9] informally described a stronger concept, Symmetrically-private COT, by additionally requiring that both parties' inputs $x, y$ remain private. Our SCOT notion is most similar to Symmetrically-private COT. We note that our work was performed independently of [9]. We give formal definitions of SCOT. Our discussions and justifications further contribute to the understanding of the concept.

Of the above notions, CEM is most similar to SCOT. In addition to capturing the property of computing on encrypted inputs, CEM satisfies stronger technical requirements, explicitly allowing reuse of generated encryption keys in multiple executions. It also has the feature of not specifying the precise security properties of the used encryptions, allowing for more flexibility and applicability (see Section 1.2 and 5 for more discussion).

### 2.2 Secure evaluation of GT

Research specifically addressing the GT problem is quite extensive. It was considered as a special case in the context of general secure function evaluation [2, 26, 28, 32, 35, 34]. This general solution is often impractical. However, because the circuit for computing GT is quite small, the general solution based on the natural circuit for computing GT is the best currently known one round approach in the standard model. As search for efficient solutions to special classes of problems in different models progressed, more efficient GT solutions implicitly appeared.

Let $n$ be the length in bits of compared numbers, and $N$ be the group size of the plaintext of the employed encryption scheme.

Naor and Nissim [27] presented a general approach to securely computing functions with low communication overhead. While the application of their solution to GT is quite efficient in the message length, it needs at least $O(\log n + \log \frac{1}{\epsilon})$ 1-out-of-$O(n)$ oblivious transfers and the same number of rounds, where $\epsilon$ is the tolerated probability of error.

Di Crescenzo [9] proposed a GT protocol, secure according to our SCOT definition. Its cost is $12n^2 + 8n^2 \log N$ modular multiplications and message complexity is $8n^2 \log N$ bits.

Fischlin's GT protocol [15] requires $6n\lambda + n\lambda \log N$ modular multiplications, where $2^{-\lambda}$ is the allowed error probability. Its message complexity (in bits) is $n \log N(\lambda + 1)$.

Fischlin extends this protocol (at the cost of approximately doubling the communication and computation costs) to satisfy our definition of GT-SCOT, with the exception of leaking the value of the predicate. We remark that this extension can be further extended to fully satisfy our definitions at the expense of further approximately doubling the communication and computation costs.

Laur and Lipmaa [24] recently proposed GT constructions, which can be viewed as modifications of our GT-SCOT protocol of Section 4. We note that the work of [24] addresses resistance to malicious adversaries.

More recently, Damgård et al. [11] initiated study of constant-round protocols for bit decomposition, a work further extended by Nishide and Ohta [29]. Both [11] and [29] present integer comparison protocols as applications of their techniques. These protocols are proposed for the multi-round and multi-party setting, which is significantly different from ours. We further note that the comparison protocol of [11] can also be viewed as a modification of our GT protocol.

## 2.3 Auctions and Private Selective Payments Protocols (PSPP)

PSPP, introduced by Di Crescenzo [9], solve the following practical problem. A server has a private message representing, say, a signed authorization, and wants to give it to one among several clients, according to some public criteria, evaluated on clients' private inputs. Di Crescenzo considers a natural instance of PSPP, where clients' inputs represent their auction bids and the highest bidding client obtains the authorization. He considers a setting with a semi-honest server and malicious clients.

Di Crescenzo designs his protocols in several phases. During *registration*, executed between each client and the server, the client's public/private key pair is established, and the server obtains the client's public key. Then the *selection* protocol is executed between all registered clients and the server, during which the selected client obtains the server's secret. Finally, in the *verification* phase, the selected client presents his claim – the obtained secret – and convinces the server that he indeed is the selected client. The registration and verification phases are designed using standard cryptographic tools; it is the selection phase that is the challenging and computationally expensive area. The main contribution of [9] is the novel maximum bidder selection protocols.

One of our contributions, GT-CEM constructions, can be used to replace the core – the selection protocols – of the PSPP of [9] (with corresponding natural modifications of the other two phases). Appropriately modified protocols of Fischlin [15] and Laur and Lipmaa [24] can be similarly used. We discuss more details and the resulting efficiency improvements of our protocols in Section 5.2 and 5.3.

We mention, but do not discuss in detail, the auction protocols for use in the settings, significantly different from ours. Naor, Pinkas and Sumner [28] use Yao's garbled circuit approach in the setting with a semi-honest mostly offline server, whose role is to ensure that the auctioneer does not cheat. Cachin [5] suggested a protocol for private bidding with the semi-honest server in the setting where the bidders additionally exchange messages between each other.

## 3 Notation, definitions and preliminaries

A function $\mu : \mathbb{N} \to \mathbb{R}$ is *negligible* if for every positive polynomial $p(\cdot)$ there exists an $N$, such that for all $n > N, \mu(n) < 1/p(n)$. A probability is *overwhelming* if it is negligibly different from 1. Statistical distance between distributions $X$ and $Y$ is defined as $\text{Dist}(X, Y) = 1/2 \sum_\alpha |Pr(X = \alpha) - Pr(Y = \alpha)|$. Statistical closeness of ensembles of random variables $X$ and $Y$ is denoted by $X \overset{s}{\equiv} Y$, and their computational indistinguishability is denoted by by $X \overset{c}{\equiv} Y$.

Informally, an encryption scheme $E = (\mathit{Gen}, \mathit{Enc}, \mathit{Dec})$ is *additively homomorphic*, if it is possible to compute an encryption of $x + y$ from encryptions of $x$ and $y$. $E$ is *probabilistic* if its encryption function randomly encrypts plaintext as one of many possible ciphertexts. There exist several schemes, satisfying these conditions, e.g. [30].

We denote a uniform sampling of an element $r$ of domain $D$ by $r \in_R D$.

By a two-party *functionality* we mean a possibly random process that maps two inputs to two outputs. We denote the *view* (i.e. its randomness, input and messages received) of a party $P$ executing a protocol $\Pi$ with a party $R$ on respective inputs $x$ and $y$ by $\text{VIEW}_P^\Pi(x, y)$. We note that $\text{VIEW}_P^\Pi(x, y)$ is a random variable over the random coins of $P$ and $R$.

Let $\nu$ denote the security parameter. Let $\lambda$ denote the correctness parameter, which specifies the allowed probability of error in the protocols. We stress that although our constructions and analysis are presented for fixed $\nu$ and $\lambda$, we have in mind their asymptotic notions. Therefore, for example, when talking about a view of a party $\text{VIEW}_P^\Pi(x, y)$, we mean an ensemble $\{\text{VIEW}_P^\Pi(x, y)\}_{\nu,\lambda}$ of views. We refer the reader to Goldreich [19] for background and in-depth discussion.

### 3.1 The Paillier encryption scheme

Our protocols (in Sections 4 and 5) require an additional property of the encryption scheme – large plaintext size, or *bandwidth*. The Paillier scheme [30] satisfies all our requirements, and we will instantiate our GT protocols with it. We present the scheme, but omit the number-theoretic justification.

Key generation: Let $N$ be an RSA modulus $N = pq$, where $p, q$ are large primes. Let $g$ be an integer of order $N\alpha$ in $\mathbb{Z}_{N^2}^*$, for some integer $\alpha$. The public key $pk = (N, g)$ and the secret key $sk = \lambda(N) = \text{lcm}((p - 1), (q - 1))$, where $\lambda(N)$ is Carmichael's lambda function.

Encryption: to encrypt $m \in \mathbb{Z}_N$, compute $\mathit{Enc}(m) = g^m r^N \bmod N^2$, where $r \in_R \mathbb{Z}_N^*$.

Decryption: to decrypt a ciphertext $c$, compute $m = \frac{L(c^{\lambda(N)} \bmod N^2)}{L(g^{\lambda(N)} \bmod N^2)} \bmod N$, where $L(u) = \frac{u-1}{N}$ takes as input an element from the set $S_N = \{u < N^2 | u = 1 \bmod N\}$.

Re-randomization: to re-randomize a ciphertext $c$, multiply it by a random encryption of 0, i.e. compute $cr^N \bmod N^2$, for $r \in_R \mathbb{Z}_N^*$.

The underlying security assumption is that the so-called composite residuosity class problem is intractable. This assumption is referred to as the Computational Composite Residuosity Assumption (CCRA). It is potentially stronger than the RSA assumption, as well as the quadratic residuosity assumption. We refer the interested reader to [30] for further details.

## 4 Secure evaluation of GT with plaintext inputs

We remind the reader of our setting. There are two semi-honest participants, who use randomness in their computation. A computationally unbounded receiver $R$ sends the first message; the polytime limited sender $S$ replies, and $R$ outputs the value of the evaluated function.

### 4.1 Strong Conditional Oblivious Transfer

We design our definition by restricting the notion of COT of [10]. Recall, in COT, the two participants, a receiver $R$ and a sender $S$, have private inputs $x$ and $y$ respectively, and share a public predicate $Q(\cdot, \cdot)$. $S$ has a secret $s$ he wishes (obliviously to himself) to transfer to $R$ iff $Q(x, y) = 1$. If $Q(x, y) = 0$, no information about $s$ is transferred to $R$. $R$'s private input and the value of the predicate remain computationally hidden from $S$.

#### 4.1.1 Our definitions

We start by describing several ways of strengthening the definition of COT, which increases modularity and widens the applicability of SCOT protocols. Our own construction for UI-SCOT, for example, requires its building blocks to have the proposed features.

First, while sufficient for the proposed timed-release encryption scheme, the definition of [10] lacks the requirement of secrecy of the sender's private input. We include this requirement in the new definition.

Secondly, we prefer the "1-out-of-2" approach. In our proposed setting, the sender possesses two secrets $s_0$ and $s_1$, and wishes (obliviously to himself) to send $s_1$ if $Q(x, y) = 1$, and to send $s_0$ otherwise. Unlike the COT "all-or-nothing" definition, this allows SCOT protocols to have the property of *not revealing $Q(x, y)$ to the receiver*. This proposal strengthens the notion since while a SCOT protocol can be trivially modified to satisfy COT definitions of [10], it is not hard to see that the opposite does not (efficiently) hold. (Clearly, because secure function evaluation can be based on OT [22], COT implies SCOT. This solution, however, is inefficient.) Further, note that it follows from our requirements that a $Q$-SCOT protocol can be trivially and efficiently modified into a $(\neg Q)$-SCOT protocol. This also does not hold for COT. We will use this property in our constructions later in this section.

The definition of [10] is presented in the common random string model, while ours is not. We do not see the need of including it, since we are able to construct useful (more) efficient COT protocols in the standard setting.

Finally, as a minor point, we only require statistical, as opposed to perfect, correctness and security against $R$, to allow for easier analysis of the protocols and wider applicability of the SCOT notion.

We now present our definition. Let sender $S$ and receiver $R$ be the participants of the protocol. We remark that, while our definitions do not impose any round complexity restrictions, our constructions will be one-round. Let $\nu$ be the security parameter and $\lambda$ be the correctness parameter, upper-bounding error probability by $O(2^{-\lambda})$. Let $D_I$ and $D_S$ be the respective domains of the parties' private inputs and sender's secrets. Let $d_I = |D_I|$ and $d_S = |D_S|$. We assume that both domains are known to both parties. Let $R$ have input $x \in D_I$, and $S$ has input $(y \in D_I, s_0, s_1 \in D_S)$. Let $Q : D_I \times D_I \to \{0, 1\}$ be a predicate. Consider the SCOT functionality, where $\emptyset$ denotes the empty string:

**Functionality 1.**

$$f_{Q-\text{SCOT}}(x,(y,s_0,s_1)) = \begin{cases} (s_1,\emptyset) & \text{if } Q(x,y)=1, \\ (s_0,\emptyset) & \text{otherwise} \end{cases}$$

There are several settings in which we can consider computing this functionality. Each of the two parties may be malicious or semi-honest and each party may or may not be computationally limited. Of course, in some of the combinations it is not possible to have nontrivial secure SCOT protocols, such as when both parties are computationally unlimited. We wish to give one definition that refers to all possible models and rely on existing definitions of secure computations in these models. We refer the reader to Goldreich [19] for in-depth presentations of definitions of security in these and other settings.

**Definition 4.1.** ($Q$-Strong Conditional Oblivious Transfer)
We say that a protocol $\Pi$ is a *$Q$-Strong Conditional Oblivious Transfer* protocol with respect to a given model, if it securely implements functionality $f_{Q-\text{SCOT}}$ (1) in the given model.

We note that this general definition covers the case when $Q$ is probabilistic.

One of the more practical and interesting settings is the model with the semi-honest unlimited receiver, semi-honest polytime sender and deterministic $Q$. We discuss our constructions in this model, and thus wish to make explicit the definition for this setting.

**Definition 4.2.** Let receiver $R$, sender $S$, their inputs $x$ and $y$, secrets $s_1$ and $s_0$, unary parameters $\nu$ and $\lambda$, and predicate $Q$ be as discussed above. We say that $\Pi$ is a *Strong Conditional Oblivious Transfer* protocol for predicate $Q$ in the semi-honest model with computationally unlimited receiver and polytime sender if

- *Transfer Validity.* With overwhelming probability in $\lambda$: If $Q(x, y) = 1$, $R$ obtains $s_1$, otherwise $R$ obtains $s_0$.
- *Security against R.* ($R$ obtains essentially no information other than the transferred secret) There exists an algorithm $\mathit{Sim}_R$ (called simulator), such that for any $x, y, s, s'$ from appropriate domains:

$$\text{if } Q(x,y) \text{ then } \{\mathit{Sim}_R(x,s)\}_\nu \overset{s}{\equiv} \{\text{VIEW}_R^\Pi(x,(y,s',s))\}_\nu$$
$$\text{if } \neg Q(x,y) \text{ then } \{\mathit{Sim}_R(x,s)\}_\nu \overset{s}{\equiv} \{\text{VIEW}_R^\Pi(x,(y,s,s'))\}_\nu$$

- *Security against $S$.* ($S$ gets no efficiently computable information about $x$) There exists an efficient simulator $\mathit{Sim}_S$, such that for any $x, (y, s_0, s_1)$ from appropriate domains:

$$\{\mathit{Sim}_S(y, s_0, s_1)\}_\nu \overset{c}{\equiv} \{\mathrm{VIEW}_S^\Pi(x, (y, s_0, s_1))\}_\nu.$$

As further justification, we wish to point out an interesting use of $Q$-SCOT protocols. When sufficiently long secrets are chosen randomly by $S$, upon completion of a $Q$-SCOT protocol, $R$ does not know either the value of $Q$, or the non-transferred secret. Thus this can be viewed as a convenient way to share the value of $Q$ among $R$ and $S$. Further, the secret that $R$ received may serve as a proof to $S$ of the value of $Q$. For example, $R$, by sending the received secret to $S$, convinces him of the value of $Q$, even if $R$ may have tried to cheat. This is not possible with COT, as $R$ is only able to provide such proof if $Q(x, y) = 1$.

## 4.2 The GT-SCOT protocol

Our constructions use semantically secure additively homomorphic encryption schemes with large message domains. For the ease and clarity of presentation and to enable resource analysis, we "instantiate" our protocols with the original Paillier scheme. We remark that the Paillier scheme has received much attention in the literature, and several variants, including an elliptic curve version [17], and improvements and extensions, e.g., the Damgård-Jurik cryptosystem [12, 13], have appeared. Using more efficient implementations further improves performance of our constructions.

Let $(\mathit{Gen}, \mathit{Enc}, \mathit{Dec})$ be the instance generation, encryption and decryption algorithms, respectively, of such a scheme. As in Definition 4.2, let $R$ and $S$ be the receiver and the sender with inputs $x$ and $y$ respectively and common parameters $\nu$ and $\lambda$. Let $x, y \in D_I$ and $s_0, s_1 \in D_S$. Let $d_S = |D_S|$ and, without loss of generality, $d_I = |D_I| = 2^n$.

Throughout this section, we will work with numbers which we will need to represent as binary vectors. For $x \in \mathbb{N}$, unless specified otherwise, $x_i$ will denote the $i^{\text{th}}$ most significant bit in the $n$-bit binary representation of $x$, including leading zeros, if applicable. Where it is clear from the context, by $x$ we mean the vector $\langle x_1, x_2, \ldots, x_n \rangle$ of bits, and by $\mathit{Enc}(x)$ we mean a vector $\langle \mathit{Enc}(x_1), \mathit{Enc}(x_2), \ldots, \mathit{Enc}(x_n) \rangle$. We will also write $\mathit{Enc}(x)$ instead of $\mathit{Enc}_{pk}(x)$, where $pk$ is clear from the context.

For the clarity of presentation, we describe the setup phase outside of the protocol. We stress that it is run as part of $R$'s first move, and in particular, *after* the parties' inputs $x$ and $(y, s_0, s_1)$ have been fixed.

**Setup Phase.** $R$ sets up the Paillier encryption scheme with group size $N = pq$ by running Gen and generating secret and public keys ($sk$ and $pk$). He chooses the number of bits in $N$ to be $\max\{\nu, |d_S| + \lambda\}$, where $|d_S|$ is the bit-length of the integer $d_S$.

We will view $D_S$ as a subset of $\mathbb{Z}_N$, and will perform operations on elements of $D_S$ modulo $N$.

**Observation 1.** We envision the following practical parameter choices for our GT protocols. First, choose $N$ and $\lambda$ to satisfy the security and correctness requirements of the encryption scheme. In practice, $\log N(\approx 1000) \gg \lambda(\approx 40 \dots 80)$, so we set $|d_S| = \log N - \lambda > 900$ bits of the bandwidth of the encryption scheme to be used for sending secrets. If $D_S$ needs to be much larger than that, it may be more practical to split it in blocks of size $|d_S|$ and run GT-SCOT several times. Choosing parameters in this manner also simplifies comparison of our results to others, and we follow this approach in Section 4.4.1.

**Observation 2.** There is a negligible (in $\lambda$) minority of elements of $D_S$ in the group of size $N$.

For our protocols, we are only interested in binary comparisons, i.e. one of $\{>, <, \leq, \geq\}$. We can trivially reduce $\{\geq, \leq\}$ to $\{>, <\}$. Indeed, for example, $x \leq y$ iff $x < y + 1$. Furthermore, in our GT-SCOT protocol (Construction 1 below), we assume that inputs $x$ and $y$ of $R$ and $S$ are not equal. This can be guaranteed, for instance, by mapping $x \mapsto 2x, y \mapsto 2y + 1$. Thus, for, example, when $R$ and $S$ wish to compute SCOT based on whether $x \leq y$, they will execute the GT-SCOT protocol on inputs $2x$ and $2y + 1$.

Similarly, we assume that $s_0 \neq s_1$. The case when $s_0 = s_1$ can be reduced to the $s_0 \neq s_1$ case by, for example, $S$ setting $y = \max\{D_I\}$ and $s_1 \in_R D_S \setminus \{s_0\}$, ensuring that $x < y$ and $s_0$ is always sent.

We now present the GT-SCOT construction. The intuition behind each step is presented immediately below, in the proof of the corresponding security theorem. Note that the arithmetic operations in the presentation of the construction are in the group of the plaintext domain of the encryption scheme.

**Construction 1.** (Computing functionality GT-SCOT)

(1) $R$ runs the setup phase, then encrypts each bit $x_i$ of $x$ with the generated $pk$ and sends $(pk, \mathit{Enc}(x_1), \dots, \mathit{Enc}(x_n))$ to $S$.

(2) $S$ computes the following, for each $i = 1, \dots, n$:

   a) *an encryption of the difference vector d, where* $d_i = x_i - y_i$.

   b) *an encryption of the flag vector f, where* $f_i = x_i \,\mathit{XOR}\, y_i = (x_i - y_i)^2 = x_i^2 - 2x_i y_i + y_i^2 = x_i - 2x_i y_i + y_i$.

   c) *an encryption of vector* $\gamma$, *where* $\gamma_0 = 0$ *and* $\gamma_i = 2\gamma_{i-1} + f_i$.

   d) *an encryption of vector* $\delta$, *where* $\delta_i = d_i + r_i(\gamma_i - 1)$, *where* $r_i \in_R \mathbb{Z}_N$.

   e) *a random encryption of vector* $\mu$, *where* $\mu_i = \frac{s_1 - s_0}{2}\delta_i + \frac{s_1 + s_0}{2}$

   *and sends a random permutation* $\pi(\mathit{Enc}(\mu))$ *to* $R$.

(3) $R$ obtains $\pi(\mathit{Enc}(\mu))$, decrypts it, and determines the output as follows: if $\mu$ contains a single $v \in D_S$, output $v$, otherwise abort.

**Theorem 4.3.** *The protocol of Construction 1 is a GT-SCOT protocol in the semi-honest model, assuming semantic security of the employed encryption scheme.*

*Proof.* We will first show that the protocol correctly computes the desired functionality. This part of the proof also provides the intuition of the construction.

It is easy to see that the homomorphic properties of the encryption scheme allow $S$ to perform all necessary operations. For example, step 2b is possible because $y_i$ are known to $S$.

Step 2a computes the (encryption of) the bit difference vector $d$.

Observe that the flag vector $f$, whose encryption is computed in Step 2b, is a $\{0,1\}$-vector, with the ones in positions where $x$ and $y$ differ. Furthermore, $\gamma$, whose encryption is computed in Step 2c, is a vector with the following structure: it starts with one or more zeros, then a one, then a sequence of non-ones. Moreover, with overwhelming probability the non-zero elements $(\gamma_i - 1)$ are not multiples of either $p$ or $q$, i.e. are in $\mathbb{Z}_N^*$. This is because the fraction of multiples of $p$ or $q$ in $\mathbb{Z}_N$ is negligible, and $p$ and $q$ are chosen randomly and independently of $x$ and $y$.

Let $\text{ind}_1$ be the (only) position where $\gamma_{\text{ind}_1} = 1$. This position is where $x$ and $y$ first differ, and thus $d_{\text{ind}_1}$ determines $\text{GT}(x,y)$. The transformation $(\gamma, d) \rightarrow \delta$ of Step 2d randomizes all coordinates of $\delta$, while setting $\delta_{\text{ind}_1}$ to the value of $d_{\text{ind}_1}$. Indeed, for $i \neq \text{ind}_1$, with overwhelming probability, $(\gamma_i - 1) \in \mathbb{Z}_N^*$. Thus, multiplying $(\gamma_i - 1)$ by $r_i \in_R \mathbb{Z}_N$ ensures that $\delta_i$ is statistically close to uniformly random in $\mathbb{Z}_N$.

With overwhelming probability, the transformation $(\delta, s_0, s_1) \rightarrow \mu$ of Step 2e is a permutation on $\mathbb{Z}_N$ that maps $-1 \mapsto s_0, 1 \mapsto s_1$. Indeed, it is not such a permutation only when $(s_1 - s_0)$ is a multiple of $p$ or $q$, an event that occurs with negligible probability, because $p$ and $q$ are are chosen randomly and independently of $s_1$ and $s_0$. This permutation preserves the randomness properties of all elements of the vector, and (as is easy to verify) performs the required mapping. Random re-encryption of Step 2e hides the information that may be contained in the randomness of the encryption. Finally, the random permutation $\pi(\mu)$ of Step 2 hides the index $\text{ind}_1$.

Consider the probability that there is not exactly one element of size $|d_S|$ in the vector decrypted by $R$. It easily follows from Observation 2 that this probability is negligible. Thus, with overwhelming probability, $R$ terminates and outputs the correct value.

Security of $R$ trivially holds because of the semantic security properties of the employed encryption scheme.

We now prove security of $S$ against computationally unbounded semi-honest $R$ by constructing a protocol view simulator $\mathit{Sim}_R(x,s)$, where $x$ is the input, and $s$ is the output of $R$. $\mathit{Sim}_R(x,s)$ has to generate a distribution statistically close to the view of $R$ in a real execution – $\text{VIEW}_R(x,(y,s_0,s_1)) = \{x, r, \mathit{Enc}(\pi(\mu))\}$, where $r$ is the randomness used by $R$ to generate $pk$ and $sk$ (of the setup phase) and the random encryptions of the first message, and $\pi(\mu)$ is defined in the protocol construction. $\mathit{Sim}_R(x,s)$ proceeds as follows. It first generates a random string $r'$ of appropriate length (to match $r$). It uses $r'$ to compute the keys $sk$ and $pk$ (including $N$). It then computes a candidate $\mu'$: for $i = 1..n$, pick random $\mu'_i \in_R \mathbb{Z}_N$. It then replaces a randomly chosen element of $\mu'$ with $s$, and outputs $\{x, r', \mathit{Enc}_{pk'}(\mu')\}$, where $\mathit{Enc}_{pk'}(\mu')$ is a vector of random encryptions of coordinates of $\mu'$ under $pk'$. Because of the previously presented arguments of the randomness of all elements of $\pi(\mu)$ (other than the one that carries the secret) and the randomness of re-encryption, it is easy to see that $\mathit{Sim}_R$ generates a distribution statistically close to the view of $R$. We note that the simulation is not perfect, since the transfer of the unintended secret is possible during the real execution, with negligible probability. □

We observe that a GT-SCOT protocol, such as presented above, immediately implies a solution to GT, in the semi-honest model. Indeed, running GT-SCOT with at least one of the secrets $s_i$ known to $R$ (say $s_1 = 1, s_0 = 0$), immediately yields the desired functionality. Moreover, for GT, the transformation of step 2e is unnecessary (while the re-randomization of the same step is still required).

### 4.2.1 Resource analysis

We evaluate the message and modular multiplication efficiency of our construction based on the use of the Paillier encryption scheme. As noted above, the use of more efficient schemes correspondingly improves the performance of our scheme. We note that we do not include the relatively small computational cost of key generation, to be consistent with the compared results of [10], [9] and [15]. Let $n$ be the length of inputs $x$ and $y$ in binary, and $N$ be the size of the plaintext domain of the Paillier scheme. Then the message complexity of Construction 1 is $l = 2n\log(N^2) = 4n\log N$ bits. Costs of some of the steps of Construction 1 depend on the actual values of operands. In our assessment, we will compute *expected* costs.

Let $w = w(y) \leq n$ be the weight (i.e. the number of ones) of the binary representation of $y$.

To encrypt each bit, $R$ computes $r^N$ and multiplies by $g^b$. We disregard the cost of single multiplication of $g^b$ by $r^N$ as a lower order term. Further, $r^N$ is computed by square and multiply, and requires expected $1.5\log N$ multiplications. Thus Step 1 takes approximately $1.5n\log N$ multiplications.

Observe that it is not necessary to perform expensive randomized encryption in the intermediate steps of $S$. Therefore, Step 2a takes $w \leq n$ multiplications. Indeed, no work is done if $y = 0$, and one multiplication if $y = 1$ (multiply by a precomputed $g^{-1}$.)

In Step 2b, if $y = 0$, no work needs to be done ($f_i = x_i$). If $y = 1$, $S$ needs to compute $-1 \cdot x_i + 1$. Addition of 1 takes one modular multiplication and is disregarded. Multiplication by $-1$ takes expected $1.5\log N$ modular multiplications. Thus Step 2b takes approximately $1.5w\log N \leq 1.5n\log N$ multiplications[1] [2].

Step 2c requires two multiplications for each $i$, that is, $2n$ total.

Step 2d requires one multiplication each for subtraction of 1 and addition of $d_i$, and $1.5\log N$ expected multiplications to multiply by $r_i$. Thus Step 2d takes approximately $1.5n\log N$ multiplications.

Step 2e requires up to expected $1.5\log N$ multiplications to multiply $\delta_i$ by $\frac{s_1-s_0}{2}$, and one multiplication to add $\frac{s_1+s_0}{2}$. (Computing *Enc*$(\frac{s_1+s_0}{2})$ takes up to expected $1.5\log N$

[1]The work of this step can be shifted to $R$ by having him also send encryptions of $-x_i$. More precisely, $R$ would need to additionally perform $1.5n\log N$ modular multiplications and send $2n\log N$ bits.

[2]Alternatively, multiplication by $-1$ under Paillier encryption can be achieved by inverting the encryption using Extended Euclidian Algorithm, which, depending on the implementation, may offer slight performance gains.

multiplications, but is done once per protocol execution, and, moreover, can be precomputed, so we ignore this cost.) Further, expected $1.5 \log N$ multiplications are needed to re-randomize each $\mu_i$. Thus Step 2e takes approximately $3n \log N$ multiplications. We note that if we do not perform the transformation of Step 2e (when, for example, computing GT), we only need $1.5n \log N$ multiplications.

Paillier decryption takes $2 \log N$ multiplications. However, we expect to perform only half of the decryptions before recovering the secret and halting. Therefore, the expected cost of decryption is $n \log N$.

Thus, in total, the protocol requires approximately $8.5n \log N$ modular multiplications ($7n \log N$ for GT). We stress that transferring up to $\log N - \lambda$ bit secrets requires the same resources. We observe that randomizing encryption and re-encryption multiplications can be precomputed once the encryption scheme is initialized, saving $3n \log N$ multiplications.

We compare the efficiency of our approach to that of Fischlin [15] and Di Crescenzo [9], using appropriate parameters. We first note that in practice, no known attack on the Paillier system is better than factoring the modulus $N$. Clearly, factoring-based attacks would also be effective against the Goldwasser-Micali (GM) scheme with the same modulus size. Thus, having already assumed CCRA (see Section 3.1), we also assume that the security of Paillier and GM schemes with the modulus of the same size are approximately the same. We note that our modular multiplications are four times slower, since we are working with modulus length twice that of the Goldwasser-Micali encryption scheme employed in [15] and [9]. The comparisons are summarized in the Table in Section 4.4.1.

## 4.3 SCOT for unions of intervals

In this section we present new efficient protocols for I-SCOT (SCOT based on the membership in an interval) and UI-SCOT (SCOT based on the membership in a union of intervals), both of which are generalizations of GT-SCOT. More specifically, in I-SCOT, the secret $s_1$ (resp. $s_0$) is transferred to $R$ if $R$'s input point $x$ belongs (resp. doesn't belong) to the interval that is the input of $S$. Similarly, in UI-SCOT, the secrets are transferred based on whether $R$'s input point $x$ belongs to the set of intervals that is the input of $S$. We will sometimes write $k$-UI-SCOT, to emphasize that $S$'s input to UI-SCOT has $k$ intervals.

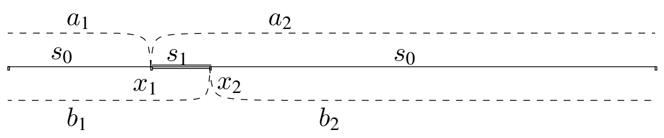
We build the I-SCOT and UI-SCOT protocols on our GT-SCOT solution. While other GT-SCOT approaches (such as based on Fischlin's protocol) are also suitable for these constructions, our solution is simpler and produces more efficient protocols in terms of both multiplication and communication complexity. In our constructions, we denote the instance of the $Q$-SCOT functionality with the secrets $s_0, s_1$ on parties' inputs $x, y$ by $Q$-SCOT $(s_1|s_0?Q(x,y))$.

In Section 4.3.1 we show how to reduce UI-SCOT to I-SCOT and I-SCOT to GT-SCOT. (We chose this modular way of presentation because it appears to be simpler.) We note that in the semi-honest model, secure reductions provide us with secure protocols when the underlying oracles are replaced by their secure implementations (see Goldreich [19] for the composition theorem.) Furthermore, the oracles' implementations may be composed in parallel, since the actions taken by the parties and the transcripts they see, can not change due to parallel (vs. sequential) composition in the semi-honest model. This, with our implementations, provides secure one round protocols for I-SCOT to UI-SCOT.

### 4.3.1 The UI-SCOT protocol

Let $x \in D_I$ be $R$'s input. In the I-SCOT setting, $S$'s input $x_1, x_2 \in D_I$ represents an interval $I$. $S$ wishes to transfer (obliviously to himself) to $R$ the secret $s_1$ (resp. $s_0$) if $x \in I$ (resp. $x \notin I$). Without loss of generality, we assume that the domain of secrets $D_S$ is an additive group $\mathbb{Z}^+_{d_S}$. We stress that we use GT-SCOT as black box, and, in particular, addition in $D_S$ is unrelated to the corresponding operation in the GT-SCOT implementation. In our discussion, all additions of secrets will be done in $D_S$, unless specified otherwise. As in the GT-SCOT discussion, we assume that $x \neq x_1, x \neq x_2$, implying that we consider open intervals. Non-open intervals are easily handled by corresponding amendments of inputs, as is done in the GT-SCOT discussion in Section 4.2.

The intuition of the reduction of I-SCOT to GT-SCOT is as follows, illustrated on the diagram below. Interval $I$ splits $D_I$ in three parts, and $S$ wishes to transfer $s_1$ "on the central part" ($I$) and $s_0$ "on the side parts" ($D_I \setminus I$). The idea is to represent these secrets as sums of independently random (i.e. random if taken separately) elements ($a_1, a_2, b_1, b_2 \in D_S$) which are to be transferred using GT-SCOT.



**Construction 2.** (Reducing I-SCOT to GT-SCOT)

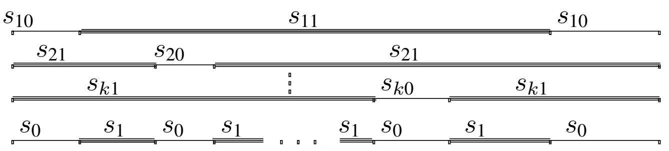
(1) $S$ randomly chooses $a_1 \in D_S$ and sets $b_1, a_2, b_2 \in D_S$ to satisfy $s_0 = a_1 + b_1 = a_2 + b_2$ and $s_1 = a_2 + b_1$
(2) (Reduction) $R$ and $S$ (in parallel) invoke oracles for GT-SCOT$(a_1|a_2?x < x_1)$ and GT-SCOT$(b_1|b_2?x < x_2)$.
(3) $R$ obtains $a', b' \in D_S$ from GT-SCOT oracle executions and outputs $a' + b'$.

**Theorem 4.4.** *The protocol of Construction 2 securely reduces functionality I-SCOT to GT-SCOT in the semi-honest model.*

*Proof.* The transfer validity property of this reduction trivially holds. Since $S$ does not receive any messages from $R$ or oracle executions, the reduction is secure against semi-honest $S$. We show how to construct $Sim_R$, simulating the following ensemble (view of $R$): VIEW$_R(x, (x_1, x_2, s_0, s_1)) = \{x, r_1, r_2\}$, where $r_1, r_2$ are the sent (via the GT-SCOT oracles) $a_i, b_j$. Let $s$ be the transferred secret. Then $Sim_R(x, s) = \{x, r'_1, r'_2\}$, where $r'_i$ are independently random elements of $D_S$ that sum up to $s$. Because, by construction, $r_1, r_2$ are also independently random with the same sum, $Sim_R$ perfectly simulates the view of $R$. □

We now reduce UI-SCOT of polynomially many intervals to I-SCOT. Here, $S$'s input represents a set of *disjoint* intervals $\{I_i = (x_{i1}, x_{i2} \in D_I)\}$, and the secrets $s_0, s_1 \in D_S$. $S$ wishes to transfer $s_1$ if $x \in \bigcup I_i$, and transfer $s_0$ otherwise. Let $k$ be the number of intervals in the set (to avoid leaking $k$ to $R$, $S$ can pad it to a known upper bound by adding empty intervals).

We represent $\bigcup I_i$ as the intersection of $k$ intervals as follows, which is illustrated on the diagram below. The bottom line represents the input set of intervals on the domain, and all other lines represent the constructed intervals that together correspond to this set. (All intervals are marked by bold lines.) The $s_i$ are the secrets to be transferred by the UI-SCOT construction, and the $s_{ij}$ are the intermediate secrets to be created by UI-SCOT and transferred by the existing I-SCOT protocol. Because the input intervals are disjoint, the cut out (thin, on the diagram) parts of the constructed intervals do not intersect, and thus any $x$ either belongs to all or to all but one constructed intervals.



To reduce UI-SCOT to I-SCOT, we choose $s_{ij} \in D_S$ based on the given $s_i$. Because of the above observation we only need to satisfy the following: $s_1 = \sum_i s_{i1}$ and $s_0 = (\sum_{i \neq j} s_{i1}) + s_{j0}, \forall j = 1, \ldots, k$. Observe that the second condition is equivalent to requiring $s_1 - s_0 = s_{j1} - s_{j0}, \forall j = 1, \ldots, k$.

**Construction 3.** (Reducing UI-SCOT to I-SCOT)

(1) $S$ chooses $s_{11}, \ldots, s_{(k-1)1} \in_R D_S$ and sets $s_{k1} = s_1 - \sum_{i=1,\ldots,k-1} s_{i1}$ and $s_{i0} = s_{i1} - (s_1 - s_0), i = 1, \ldots, k$.

(2) (Reduction) $S$ and $R$ (in parallel) invoke oracles for I-SCOT$(s_{i1}|s_{i0}?x \in I_i)$, for each $i = 1, \ldots, k$.

(3) $R$ obtains $a_1, \ldots, a_k \in D_S$ from $k$ oracle executions and outputs $\sum_i a_i$.

**Theorem 4.5.** *The protocol of Construction 3 securely reduces functionality UI-SCOT to I-SCOT in the semi-honest model.*

*Proof.* The transfer validity property of this reduction trivially holds. Since $S$ does not receive any messages from $R$ or oracle executions, the reduction is secure against semi-honest $S$. To show security against computationally unbounded $R$, we construct $Sim_R$ simulating the view of $R$ VIEW$_R(x, y) = \{x, r_1, \ldots, r_k\}$, where $r_1, \ldots, r_k$ are the oracle-sent elements of $D_S$ defined by step 1 of the construction. Let $s$ be the transferred secret. Then $Sim_R(x, s) = \{x, r'_1, \ldots, r'_k\}$, where $r'_i \in_R D_S$ with the restriction $s = \sum_i r'_i$. $Sim_R$ perfectly simulates view of $R$ because both ensembles are $(k-1)$-wise independent random elements of $D_S$ that sum up to the same value $s$. □

## 4.4 The $(\bigwedge_i Q_i(x_i, y_i))$-COT protocol.

We construct $\bigwedge_i Q_i(x_i, y_i))$-COT protocol using oracles for corresponding $Q_i$-SCOT. Let $R$ have input $x_1, \dots, x_n$, and $S$ have input $y_1, \dots, y_n$. $S$ wishes to send a secret $s$ to $R$ iff $\bigwedge_i (Q_i(x_i, y_i)) = 1$. The idea is to introduce "specialness" of $s$ as we did for GT-SCOT, by, for example, extending the domain of secrets $D_S$ to the group $D'_S = \mathbb{Z}^+_{d'_S}$, where $d'_S = |D'_S| \gg |D_S|$. Then $S$ represents $s \in D_S$ as a sum of random secrets $s_i \in_R D'_S$, and runs $Q_i$-SCOT$(s_i | r_i ? Q_i(x_i, y_i))$, where $r_i \in_R D'_S$. Indeed, if the conjunction holds, then only the $s_i$'s will be transferred, and they will sum up to $s \in D_S$. If any (or any number of) predicates do not hold, one (or more) $r_i$ will be transferred, which will randomize (in $D'_S$) the sum obtained by $R$.

**Construction 4.** (Reducing $(\bigwedge_i Q_i(x_i, y_i))$-COT to $Q_i$-SCOT)

(1) $S$ chooses $r_1, \dots, r_n, s_1, \dots, s_{n-1} \in_R D'_S$ and sets (in $D'_S$) $s_n = s - \sum_{i=1,\dots,n-1} s_i$.

(2) $R$ and $S$ in parallel invoke oracles for $Q_i$-SCOT$(s_i | r_i ? Q_i(x_i, y_i))$, $\forall i = 1, \dots, n$.

(3) $R$ obtains $a_1, \dots, a_n \in D'_S$ from the $Q_i$-SCOT oracle executions and sets $v = \sum_i a_i$. $R$ outputs $v$, if $v \in D_S$, and outputs a special symbol $\perp$ otherwise.

**Theorem 4.6.** *The protocol of Construction 4 securely reduces functionality $(\bigwedge_i Q_i(x_i, y_i))$-COT to $Q_i$-SCOT in the semi-honest model.*

*Proof.* Correctness of this reduction is easily verifiable. Since $S$ does not receive any messages from $R$ or oracle executions, the reduction is secure against semi-honest $S$. To show security against computationally unbounded $R$, we construct $Sim_R$ simulating the view of $R$ VIEW$_R(x, y) = \{x, t_1, \dots, t_n\}$, where $t_1, \dots, t_n$ are the oracle-sent elements of $D'_S$ defined by step 1 of the construction. Let $s$ be the output of $R$. If $s = \perp$, set $s \in_R D'_S$. Then $Sim_R = \{x, t'_1, \dots, t'_n\}$, where $t'_i \in_R D'_S$ with the restriction $s = \sum_i t'_i$. $Sim_R$ perfectly simulates view of $R$. If the output of $R$ was $\perp$, both ensembles are independent random elements of $D'_S$. Otherwise, if the output of $R$ was not $\perp$, both ensembles are $(n-1)$-wise independent random elements of $D'_S$ that sum up to the same value $s$. □

The above reduction, together with the UI-SCOT construction, gives a protocol for computing the conjunction of memberships in sets of intervals. Application of De Morgan law ($P \vee Q = \neg(\neg P \wedge \neg Q)$) additionally gives a protocol for computing the disjunction of memberships in sets of intervals.

**Corollary 4.7.** *There exists efficient one round protocols for computing conjunction and disjunction of memberships in sets of intervals, secure against computationally unlimited R.*

**Observation 3.** When instantiating underlying oracles, constructions of Section 4.3 and 4.4 require $R$ to send the same encrypted input as part of each such instance. It is easy to see that this saves on computation and communication if this step is performed only once and shared among the instances. Namely, in Construction 1, for each repeated encrypted input, $R$ saves $1.5n \log N$ multiplications, and $S$ saves $1.5n \log N$ multiplications (since $S$ now does not need to recompute $-1 \cdot x_i$ in Step 2b). Thus, each "subsequent" instance of Construction 1 costs only $5.5n \log N$ multiplications.

This optimization also helps with potential transition to the malicious model, since $R$ now would not need to prove the equality of his inputs to the oracle executions.

### 4.4.1 Resource analysis

We continue and expand the resource analysis of Section 4.2.1. Recall that $\lambda$ and $\nu$ are the correctness and security parameters, $n$ is the number of bits in the compared integers, and $N$ is the modulus of the employed encryption scheme. As discussed in Observation 1, we choose $\nu = \log N$ and $\lambda$ as in [15]. This determines the secrets domain $D_S$ to be of size (at most) $2^{\nu-\lambda}$. As noted in Section 4.2.1, we do not include the cost of key generation in any of the compared solutions.

It is easy to see that our $k$-UI-SCOT construction (Construction 3) makes $2k$ calls to the underlying $\lambda$-bit GT-SCOT oracle. As noted in Observation 3, the first call requires full $8.5n \log N$ multiplications and $4n \log N$ bits transferred, and each of $2k-1$ subsequent calls costs only $5.5n \log N$ multiplications and $2n \log N$ bits transferred.

In our comparison, we consider comparable multiplications, and take a factor of four penalty (reflected below), since multiplications mod$N^2$ are four times slower than multiplications mod$N$ used in [10, 9, 15]. Thus, when using our implementation of GT-SCOT, UI-SCOT requires sending $(2k-1) \cdot 2n \log N + 4n \log N = (4k+2)n \log N$ bits and performing about $4(2k \cdot 5.5 + 3)n \log N = (44k + 12)n \log N$ multiplications in a group of size $N$. Using $\lambda$-bit GT-SCOT oracle implementation based on Fischlin's and Di Crescenzo's GT results in almost full factor of $2k$ blowup in communication since server sends most of the traffic. The $2k$ factor blowup in the computation also seems necessary when using these schemes.

The following table summarizes the cost of comparable modular multiplications and communication of our protocol in relation to others.

| Protocol | GT predicate | | $c$-bit GT-SCOT, $c<\nu$-$\lambda$ | |
|---|---|---|---|---|
| | mod. mult. | comm. | mod. mult. | comm. |
| of [15] | $6n\lambda + n\lambda \log N$ | $\lambda n \log N$ | $24nc\lambda + 4nc\lambda \log N$ | $4nc\lambda \log N$ |
| of [10] | $8n + 4n \log N$ | $4n \log N$ | N/A | N/A |
| of [9] | $6n^2 + 4n^2 \log N$ | $4n^2 \log N$ | $12n^2c + 8n^2c \log N$ | $8n^2c \log N$ |
| our work | $28n \log N$ | $4n \log N$ | $34n \log N$ | $4n \log N$ |

| Protocol | $k$-UI-SCOT | |
|---|---|---|
| | mod. mult. | comm. |
| of [15] | $48kn\lambda^2 + 8kn\lambda^2 \log N$ | $8kn\lambda^2 \log N$ |
| of [10] | N/A | N/A |
| of [9] | $24kn^2c + 16kn^2c \log N$ | $16kn^2c \log N$ |
| our work | $(44k + 12)n \log N$ | $(4k + 2)n \log N$ |

We see no obvious way to transform the schemes of [10] to GT-SCOT, and thus do not include the corresponding resource calculations.

## 5 Secure evaluation of GT with encrypted inputs

We now consider the setting where players only possess *encryptions* of the compared integers. This restriction may arise, for example, due to composition of protocols, or when an additional player serves as a facilitator of the computation of the multiparty functionality $f$. This player – the semi-honest helping Server $S$ – is given the encrypted inputs to $f$; he produces some representation of the value of $f$. The value of $f$ can later be decoded from this representation using the private key of the employed encryption scheme. This scenario is appealing for its round efficiency and is widely applicable in practice. For example, it applies to auctions run by semi-honest servers. There, the server $S$ receives encryptions of parties' bids, and wants to commit to a deal (e.g. by sending a secret) with the winner.

The first step in designing secure protocols is making explicit the setting in which they are run and the necessary security requirements. This is a difficult task, especially since we would like our constructions to be applicable to a variety of settings. For example, the server $S$ may obtain encrypted inputs from parties $A$ and $B$ and let either $A$ or $B$ or a third party $C$ decode the output. The role of $S$ might be played by one of the parties (thus bringing us to the two-party setting). Protocols can use encryption schemes, which may or may not be re-initialized for each execution of the protocol. Players $A, B$ or $C$ may have different levels of trust.

Encompassing all these situations in one definition is difficult. We propose to extract and formalize what these definitions would have in common – requirements of correctness and privacy of the output of the semi-honest Server $S$. This modularity is convenient, since we can now model $S$ as a non-interactive algorithm. A variety of setting-specific requirements for hiding the input from the server can be later naturally defined and satisfied with appropriate use of encryption. In particular, protocols for all the settings mentioned in the previous paragraph can be derived from one for the setting with the helping server. It thus appears most natural and general to interpret and approach the problem of comparing encrypted numbers in the latter setting, and we present our work from this angle.

**Encrypted Mapping.** We model the Server $S$ as a polytime randomized mapping algorithm *Rmap*. *Rmap* takes as input the public key of the encryption scheme $E$, the (encrypted with $E$) input(s), and outputs some representation of the value of $f$. Of course, this output should be interpreted. We require existence of the polytime recovery procedure *Rec*, which takes the representation of the value of $f$ and the private key of $E$ and computes the intended output (this is the correctness condition). Further, we require that the randomized representation statistically hides all other information, ensuring privacy of arbitrary compositions of outputs of *Rmap* even against computationally unlimited attackers. We call the pair $(\textit{Rmap}, \textit{Rec})$ an Encrypted Mapping (EM). We formalize a variant of this notion in Definition 5.1 below.

We choose not to specify the requirements of security of encryption in the definition.

This allows a protocol designer to concentrate on the high-level combinatorial properties of EM and defer discussion of detailed setting-specific concerns. Such low-level concerns include considering whether some inputs to $S$ contain decryption keys (which would allow $S$ to learn more than he should) and considering malicious behaviour, such as providing invalid or substituted inputs. A protocol designer can now first describe the combinatorial *Rmap* and *Rec*, which would imply solutions to a variety of settings in the semi-honest model, assuming the semantic security of the employed encryption scheme. Afterwards, the protocols can be adapted to a variety of specific settings and modified to withstand certain malicious behaviours (e.g. using the conditional disclosure techniques of [1, 24]. See more in Section 5.2.1).

We wish to give a strong definition, so that the constructions can be used in a variety of settings. In particular, we want our construction to work with all (including adversarial) instantiations of used encryption schemes. In many popular encryption schemes (e.g. Paillier [30]) the plaintext domain $D_P$ varies with different instantiations. Many interesting functions $f$ are defined on fixed domains, independent of $D_P$. We handle this detail by ensuring that $D_P$ includes the domain of inputs to $f$ by appropriately modifying the family of encryptions to only include members with sufficiently large $D_P$. We note that a sufficiently large $D_P$ is usually implied by the semantic security requirement of the scheme.

We remark that we achieve a strong definition by quantifying over all valid inputs and randomness used by encryptions – i.e. over everything but the randomness used by *Rmap*. This, for example, ensures that the adversary does not benefit from knowing the randomness used for encrypting inputs to *Rmap*.

**Conditional Encrypted Mapping.** In this work, we are mainly interested in constructing the protocols for transferring a secret (e.g. a sale commitment or a rejection) depending on whether a certain predicate on two inputs (e.g. the bid is greater than the asking price) holds. We call the corresponding EM a Conditional Encrypted Mapping (CEM). We give a formal definition for this special case and note that a more general EM definition can be naturally constructed.

We define CEM with respect to an encryption scheme $E = (\mathit{Gen}, \mathit{Enc}, \mathit{Dec})$. Denote by $(sk, pk)$ a public/private key pair for $E$, and by $E_{pk}$ denote the initialized encryption scheme $E$. Let $D_{P_{pk}}$ denote the plaintext domain of $E_{pk}$ and $D_{R_{pk}}$ denote the domain of randomness used by $\mathit{Enc}_{pk}$. Denote by $\mathit{Enc}_{pk,\alpha}(x)$ the encryption of $x$ under $pk$ using randomness $\alpha$. Let $Q : D_Q \times D_Q \to \{0, 1\}$ be a deterministic predicate defined on a fixed domain. Recall, we only consider families of $E_{pk}$ where $D_Q \subset D_{P_{pk}}$. Let $\nu$ be the security parameter. We remark that in practice we are also interested in the correctness parameter $\lambda$. Security and correctness properties of Definition 5.1 are formulated with the notion of statistical closeness. Since $\nu$ and $\lambda$ are polynomially related, we, for simplicity, use only the parameter $\nu$. Let $D_S$ be the (fixed) domain of secrets. Note that even though for simplicity of presentation the domains $D_Q$ and $D_S$ are fixed, their elements representation is polynomially related to all other parameters. Further, in practice (and in our constructions), $D_Q$ and $D_S$ can grow with $\nu$ at no extra cost.

**Definition 5.1.** ($Q$-Conditional Encrypted Mapping) A *$Q$-Conditional Encrypted Mapping* ($Q$-CEM) is a pair of polytime algorithms $(\mathit{Rmap}, \mathit{Rec})$ (with implicitly defined domain of mappings $D_{M_{pk}}$), such that the following holds.

The probabilistic randomized mapping algorithm *Rmap* takes as input

$$(s_0, s_1, e_0, e_1, pk),$$

where $e_0, e_1$ are encryptions under $E_{pk}$, and $s_0, s_1 \in D_S$. *Rmap* outputs an element from $D_{M_{pk}}$. The deterministic recovery algorithm *Rec* takes as input secret key $sk$ and an element from $D_{M_{pk}}$ and outputs an element from the domain of secrets $D_S$ or a failure symbol $\perp$.

*Rmap* and *Rec* satisfy the following conditions:

- (correctness) $\forall (sk, pk) \leftarrow \mathit{Gen}(\nu), \forall s_0, s_1 \in D_S, \forall \alpha, \beta \in D_{R_{pk}}, \forall x, y \in D_Q$ : with overwhelming probability in $\nu$, taken over random inputs of *Rmap*:
  $\mathit{Rec}(\mathit{Rmap}(s_0, s_1, \mathit{Enc}_{pk,\alpha}(x), \mathit{Enc}_{pk,\beta}(y), pk), sk) = s_{Q(x,y)}$.
- (statistical privacy) There exists a simulator *Sim*, such that $\forall (sk, pk) \leftarrow \mathit{Gen}(\nu)$, $\forall s_0, s_1 \in D_S, \forall x, y \in D_Q, \forall \alpha, \beta \in D_{R_{pk}}$ : the statistical distance

  $$\mathrm{Dist}(\mathit{Sim}(s_{Q(x,y)}, pk), \mathit{Rmap}(s_0, s_1, \mathit{Enc}_{pk,\alpha}(x), \mathit{Enc}_{pk,\beta}(y), pk))$$

  is negligible in $\nu$.

Note, Definition 5.1 does not require $E$ to have any security properties. Thus, formally, inputs $e_0, e_1$ to *Rmap* are simply *encodings* of elements in $D_Q$ (and $Q$-CEM can be constructed unconditionally). In practice, however, we envision using a semantically secure $E$; thus we call $e_0, e_1$ encryptions. Jumping ahead, we note that in our GT constructions 5.1.3, the inputs $e_0, e_1$ to *Rmap* are bitwise encryptions of the clients' bids. Note that Definition 5.1 allows this interpretation, since encrypting $x$ bit-by-bit can be viewed as an encryption scheme itself.

Further, Definition 5.1 does not guarantee either correctness or privacy if $e_0$ or $e_1$ are not proper encryptions of elements of $D_Q$. This is sufficient in the semi-honest model; we discuss methods of handling malicious behaviour in Section 5.2.1.

## 5.1 The GT-CEM construction and protocols

The GT-CEM construction builds on the ideas of the GT-SCOT protocol of Construction 1. Recall, this protocol operates on (homomorphically encrypted) bits of the inputs. The main idea is to isolate the "important" position – the one where input bit strings first differ. This is done by applying a randomization procedure, which assigns a predetermined value to the "important" position, and simultaneously randomizes values in all other positions. (The rest is easily accomplished by applications of linear functions under encryption.)

Observe that the randomization procedure of Construction 1 requires $S$ to know the plaintext of his input. In Section 5.1.1, we discuss the necessary properties of the new randomization, which works with encryptions only. In Section 5.1.2, we present such a randomization procedure and in Section 5.1.3 we give a GT-CEM construction. We give an alternative randomization procedure in Section 5.1.4, which can be incorporated into our GT-CEM.

### 5.1.1 The intuition of GT-CEM and the formalization of the randomization requirements

Recall, we are given secrets $s_0, s_1$ and bitwise encryptions of inputs $x$ and $y$. We can compute an encryption of the bit difference vector $d$, where $d_i = x_i - y_i$. Elements of the difference vector $d$ assume one of $\{-1, 0, 1\}$. Let $j = \min_{d_i \neq 0} i$ be the index of the "important" position. Our goal is to isolate the value $d_j$ by computing an encryption of vector $\mu$, such that $\forall i \neq j, \mu_i \in_R D_{P_{pk}}$ and $\mu_j = d_j$. As in Construction 1, we can obtain such $\mu_i$ for $i \geq j$ by computing for $i = 1, \ldots, n$: $\mu_0 = 0; \mu_i = r_i \mu_{i-1} + d_i$, where $r_i \in_R D_{P_{pk}}$. Now vector $\mu$ is a vector of encryptions of (in order): one or more 0, either a 1 or a $-1$, one or more random elements of $D_{P_{pk}}$. We need to map the zeros of $\mu$ to random elements in $D_{P_{pk}}$, while preserving the properties of $\mu_i, i \geq j$. Our randomization maps $-1 \rightarrow s_0, 1 \rightarrow s_1$ (under encryption). At the same time, it maps 0 and random elements from $D_{P_{pk}}$ to random elements from $D_{P_{pk}}$. It is not hard to see (and we explicitly show it in Section 5.1.3) that such randomization naturally leads to a GT-CEM.

We believe that such randomization may be useful in other applications as well. Therefore, we formalize its requirements. We present the definition in a slightly more general way, by allowing arbitrary constants instead of $-1, 1$. Further natural extensions of this definition are possible.

Let $v_0, v_1 \in \mathbb{Z} \setminus \{0\}$ be fixed, and $v_0 \neq v_1$. Let $E, \nu, sk, pk, E_{pk}, D_{P_{pk}}, D_{R_{pk}}, D_S$ be as in Definition 5.1. Let $i \in \{0, 1\}$. We view $v_i$ as an element of $D_{P_{pk}}$ in the natural manner (i.e. as $v_i \bmod |D_{P_{pk}}|$). We note that even though this representation may vary with the choice of $pk$, $v_i$ is a constant. Further, we require $v_i \neq 0 \bmod |D_{P_{pk}}|$ and $v_0 \neq v_1 \bmod |D_{P_{pk}}|$.

**Definition 5.2.** ($(v_0, v_1)$-Randomizing Mapping)
A $(v_0, v_1)$-*Randomizing Mapping* (RM) is a pair of polytime algorithms (*Rmap*, *Rec*) (with implicitly defined domain of mappings $D_{M_{pk}}$), such that the following holds.

The probabilistic randomized mapping algorithm *Rmap* takes as input $(s_0, s_1, e, pk)$, where $e$ is an encryption under $E_{pk}$, and $s_0, s_1 \in D_S$. *Rmap* outputs an element from $D_{M_{pk}}$. The deterministic recovery algorithm *Rec* takes as input secret key $sk$ and an element from $D_{M_{pk}}$ and outputs an element from the domain of secrets $D_S$ or a failure symbol $\perp$.

*Rmap* and *Rec* satisfy the following conditions:

- (correctness) $\forall (sk, pk) \leftarrow Gen(\nu), \forall i \in \{0, 1\}, \forall s_0, s_1 \in D_S, \forall \alpha \in D_{R_{pk}}$, for $x \in_R D_{P_{pk}}$, with overwhelming probability in $\nu$:
  $Rec(Rmap(s_0, s_1, Enc_{pk,\alpha}(v_i), pk), sk) = s_i$,
  $Rec(Rmap(s_0, s_1, Enc_{pk,\alpha}(x), pk), sk) = \perp$,
  where the probability is taken over choices of $x$ and random inputs of *Rmap*.

- (statistical privacy at $v_0, v_1$) There exists a simulator *Sim*, such that $\forall(sk, pk) \leftarrow$ *Gen*$(\nu)$, $\forall s_0, s_1 \in D_S$, $\forall i \in \{0,1\}, \forall \alpha \in D_{R_{pk}}$ : the statistical distance
  $\text{Dist}(\mathit{Sim}(s_i, pk), \mathit{Rmap}(s_0, s_1, \mathit{Enc}_{pk,\alpha}(v_i), pk))$
  is negligible in $\nu$.
- (statistical privacy at 0 and at random elements of $D_{P_{pk}}$) There exists a simulator *Sim*$_0$, such that $\forall(sk, pk) \leftarrow \mathit{Gen}(\nu), \forall s_0, s_1 \in D_S, \forall \alpha \in D_{R_{pk}}$ : the statistical distances
  $\text{Dist}(\mathit{Sim}_0(pk), \mathit{Rmap}(s_0, s_1, \mathit{Enc}_{pk,\alpha}(0), pk))$ and
  $\text{Dist}(\mathit{Sim}_0(pk), \mathit{Rmap}(s_0, s_1, \mathit{Enc}_{pk,\alpha}(R), pk))$
  are negligible in $\nu$, where $R$ is uniform on $D_{P_{pk}}$.

It is easy to see that, with overwhelming probability, recovery of a mapping of 0 will result in the output of $\perp$. More specifically,

**Observation 4.** Let (*Rmap*, *Rec*) be a $(v_0, v_1)$-RM, as above. Then *Rmap* and *Rec* satisfy the following correctness condition: $\forall(sk, pk) \leftarrow \mathit{Gen}(\nu), \forall s_0, s_1 \in D_S, \forall \alpha \in D_{R_{pk}}$, with overwhelming probability in $\nu$:
$\mathit{Rec}(\mathit{Rmap}(s_0, s_1, \mathit{Enc}_{pk,\alpha}(0), pk), sk) = \perp$,
where the probability is taken over random inputs of *Rmap*.

Indeed, from the second privacy requirement of $(v_0, v_1)$-RM, it follows that the output of *Rmap* on 0 is distributed statistically close to the output of *Rmap* on a random element of $D_{P_{pk}}$. At the same time, the correctness requirement of $(v_0, v_1)$-RM guarantees that, with overwhelming probability in $\nu$,

$$\mathit{Rec}(\mathit{Rmap}(s_0, s_1, \mathit{Enc}_{pk,\alpha}(x), pk), sk) = \perp.$$

This implies Observation 4.

### 5.1.2 A space-efficient $(-1, 1)$-RM

We present a construction for $(-1, 1)$-RM, based on the Paillier encryption scheme [30], which we use to construct GT-CEM. Let $E$ be the Paillier scheme initialized as described in Definition 5.2. Let *Rmap* be given an encryption under $E_{pk}$. Our $(-1, 1)$-RM is space optimal – *Rmap* outputs a single encryption under $E_{pk}$.

At first glance, the requirements on *Rmap* are conflicting: we must satisfy three data points $((v_0, s_0), (v_1, s_1), (0, \mathit{random}))$ with a linear function (only linear functions can be applied under the homomorphic encryption). Our idea is for *Rmap* to produce encryptions not of secrets $s_i$, but of their *randomized encodings* $S_i$. We carefully randomize the encodings $S_i$, such that their linear combination of interest (i.e. the value that 0 is mapped to) is a random element in $D_{P_{pk}}$.

Let $f = ax + b$ be a linear mapping, such that $f(-1) = -a + b = S_0$ and $f(1) = a + b = S_1$. Then $b = (S_0 + S_1)/2$ and $a = S_1 - (S_0 + S_1)/2 = (S_1 - S_0)/2$. We want to ensure that $f(0) = b = (S_0 + S_1)/2$ is random, while, for $i \in \{0, 1\}$, $S_i$ encodes $s_i$ and contains no other information.

**Construction 5.** ($(-1,1)$-RM)

Let $\lambda$ and $\nu$ be the correctness and security parameters. Let the plaintext group of $E_{pk}$ be $D_{P_{pk}} = \mathbb{Z}_N$, where $N = pq$ is of bit size $n > \nu$. Let $k = \lfloor (n-1)/2 \rfloor$. Define the domain of secrets to be $D_S = D_{S_{pk}} = \{0,1\}^{k-\lambda}$, and the domain of mappings $D_{M_{pk}}$ to be the domain of encryptions under $E_{pk}$.

(1) *Rmap* on input $(s_0, s_1, e, pk)$ proceeds as follows.

Set $s'_i = s_i 0^\lambda$ (to help distinguish secrets from random strings). View $s'_0, s'_1$ as elements of $\mathbb{Z}_N$. Choose $R \in_R \mathbb{Z}_N$ and a bit $c \in_R \{0,1\}$. Let $r_1$ (resp. $r_0$) be the integer represented by $k$ lower (resp. remaining) bits of $R$, i.e. $R = r_0 2^k + r_1$.

Set $S_0, S_1$ as follows. If $c = 0$, then set $S_0 = r_0 2^k + s'_0$ and $S_1 = s'_1 2^k + r_1$. If $c = 1$, then set $S_0 = s'_0 2^k + r_1$ and $S_1 = r_0 2^k + s'_1$.

Compute $a = (S_1 - S_0)/2 \bmod N$ and $b = (S_0 + S_1)/2 \bmod N$ .

Finally, apply $f = ax + b$ to $e$ under the encryption and re-randomize the result, that is, choose $r' \in_R \mathbb{Z}_N^*$ and output $e^a g^b r'^N \bmod N^2$.

(2) *Rec* on input $(e', sk)$ proceeds as follows.

*Rec* computes $d = Dec_{sk}(e')$. Let $d_n, \ldots, d_1$ be the bit representation of $d$. Let $D_1 = d_{2k}, \ldots, d_k$ and $D_0 = d_k, \ldots, d_1$. For $i \in \{0,1\}$, if $D_i = s0^\lambda$, output $s$ and halt. Otherwise output $\perp$.

**Theorem 5.3.** *(Rmap, Rec) described in Construction 5 is a $(-1,1)$-RM.*

*Proof.* : We first show that the two correctness properties hold. It is easy to follow the construction of $S_i$ and observe that either its lower $k$ bits or the remaining bits contain the intended secret $s_i$. Further, the part of $S_i$ that does not represent the secret is random. Therefore the secret is easily distinguishable thanks to the added trailing zeros. Thus, the first correctness condition holds with overwhelming probability in $\lambda$. Further, $f$ applied by *Rmap* is a linear function, which is a permutation on $\mathbb{Z}_N$ with overwhelming probability in $\nu$. (Indeed $f = ax + b$ is not a permutation only if $a = (S_1 - S_0)/2$ is not invertible, which occurs with negligible probability.) Therefore, *Rmap*, evaluated on an encryption of a random element of $\mathbb{Z}_N$, produces a random encryption of a random element of $\mathbb{Z}_N$. It is now easy to see that *Rec* outputs $\perp$ on an encryption of a random element with overwhelming probability in $\lambda$.

The privacy at $v_0, v_1$ condition also holds. Indeed, given a secret $s \in D_S$, and $pk$, the required *Sim*$(s, pk)$ simulates the output of *Rmap*$(s_0, s_1, Enc_{pk,\alpha}(v_i), pk)$ as follows. Choose a random bit $c' \in_R \{0,1\}$ and a random $S' \in \mathbb{Z}_N$. If $c' = 0$ set the lower $k$ bits of $S'$ to be $s0^\lambda$. If $c' = 1$ set the the higher $n-k$ bits of $S'$ to be $s0^\lambda$. Return a random encryption of $S'$ under $pk$. It is easy to see that *Sim* satisfies the necessary conditions. Indeed, both *Sim* and *Rmap* output random encryptions of an element of $\mathbb{Z}_N$. In both cases equiprobably either the lower or higher half of the bits of the plaintext encode $s$.

The privacy at 0 and at random elements of $\mathbb{Z}_N$ holds for the following reasons. Firstly, as shown in the proof of correctness, *Rmap*, evaluated on encryptions of random elements of $\mathbb{Z}_N$, produces random encryptions of random elements of $\mathbb{Z}_N$. This is easy to simulate with only knowing $pk$, and we present the simple simulator below.

It remains to show that *Rmap* evaluated on an encryption of 0 does the same. Recall, *Rmap* applies $f$ to the input encryption. There are two cases.

If $c = 0$ then $f(0) = (S_0 + S_1)/2 = (r_0 2^k + s_0 + s_1 2^k + r_1)/2 = (r_0 2^k + r_1 + s_0 + s_1 2^k)/2 = (R + s_0 + s_1 2^k)/2$.

If $c = 1$ then $f(0) = (S_0 + S_1)/2 = (s_0 2^k + r_1 + r_0 2^k + s_1)/2 = (r_0 2^k + r_1 + s_1 + s_0 2^k)/2 = (R + s_1 + s_0 2^k)/2$.

In either case, $f(0)$ is random on $\mathbb{Z}_N$ due to the additive random term $R/2$.

*Sim*$_0$, required by Definition 5.2, proceeds as follows. On input $pk$, it outputs a random encryption of a randomly chosen element of $\mathbb{Z}_N$. The above argument shows that this simulator suffices. □

### 5.1.3 GT-CEM based on bitwise Paillier encryption of inputs

Let $n$ be the length of the compared numbers $x$ and $y$. We will use the Paillier encryption scheme $E$ to encrypt inputs to *Rmap* in the bitwise manner. That is, *Gen*$(\nu)$ is run, fixing $(sk, pk)$ and the instance $E_{pk}$. The inputs to *Rmap* are $(s_0, s_1, e_0, e_1, pk)$, where $e_0 = \langle Enc_{pk}(x_1), \ldots, Enc_{pk}(x_n)\rangle, e_1 = \langle Enc_{pk}(y_1), \ldots, Enc_{pk}(y_n)\rangle$, where $x_1$ and $y_1$ are the most significant bits. Let $(Rmap_1, Rec_1)$ be a $(-1,1)$-RM based on the Paillier encryption scheme (e.g. Construction 5), instantiated with $E_{pk}$. Let $D_{M_{pk1}}, D_{S_1}$ be the domains of mappings and secrets of $(Rmap_1, Rec_1)$.

**Construction 6.** (GT-CEM)

Let $\lambda$ and $\nu$ be the correctness and security parameters. Let the plaintext group of $E_{pk}$ be $D_{P_{pk}} = \mathbb{Z}_N$, where $N = pq$ is of bit size $|N| > \nu$. Define the domain of secrets $D_S$ of GT-CEM to be $D_{S_1}$ and the domain of mappings $D_{M_{pk}}$ of GT-CEM to be $D^n_{M_{pk1}}$, the set of $n$-tuples over $D_{M_{pk1}}$.

(1) *Rmap* on input $(s_0, s_1, e_0, e_1, pk)$ computes, for each $i = 1, \ldots, n$:

 a) *an encryption of the difference vector d, where* $d_i = x_i - y_i$.

 b) *an encryption of vector* $\gamma$, *s.t.* $\gamma_0 = 0$ *and* $\gamma_i = r_i\gamma_{i-1} + d_i$, *where* $r_i \in_R \mathbb{Z}_N$.

 c) *a randomized mapping vector* $\mu$, *where* $\mu_i = Rmap_1(s_0, s_1, Enc_{pk}(\gamma_i), pk)$.

 *Rmap* outputs a random permutation $\pi(\mu)$.

(2) *Rec* on input $(\mu'_1..\mu'_n, sk)$ proceeds as follows. For $i = 1, \ldots, n$:

 a) Let $z_i = Rec_1(\mu'_i, sk)$. If $z_i \neq \perp$, output $z_i$ and halt.

 Otherwise, if $\forall i = 1, \ldots, n, z_i = \perp$, output $\perp$.

**Theorem 5.4.** *Construction 6 is a GT-CEM, if* $(Rmap_1, Rec_1)$ *is a* $(-1,1)$*-RM based on* $E_{pk}$.

*Proof.* First, it is easy to see that the homomorphic properties of the encryption scheme allow *Rmap* and *Rec* to perform all necessary operations.

We now show that Construction 6 satisfies the correctness requirement. Let $j$ be the position where $x$ and $y$ first differ; thus $d_j$ determines GT$(x, y)$. Then $\gamma$ is a vector with the following structure: it starts with one or more zeros, then, in position $j$, a one or a minus one, then a sequence of elements statistically close to uniform in $\mathbb{Z}_N$. Indeed, for $i > j, \gamma_i$ deviates from being uniform on $\mathbb{Z}_N$ only if $\gamma_{i-1} \notin \mathbb{Z}_N^*$, which occurs with negligible probability. It is not hard to see that *Rec*, using $Rec_1$, will recover $s_{GT(x,y)}$, with overwhelming probability. This immediately follows from the structure of $\gamma$, the correctness property of Definition 5.2, and Observation 4.

We now show that the privacy condition holds as well. We construct simulator $Sim_{\text{CEM}}(s, pk)$, required by Definition 5.1, where $pk$ is the public key established in the setup phase and $s = s_{GT(x,y)}$. Recall, $Sim_{\text{CEM}}(s, pk)$ has to generate a distribution statistically close to the vector $Rmap(s_0, s_1, Enc_{pk,\alpha}(x), Enc_{pk,\beta}(y), pk)$. Because of the structure of $\gamma$, *Rmap*, while computing $\mu$ (Step 1c), will call one of $Rmap_1(s_0, s_1, Enc_{pk}(-1), pk)$ and $Rmap_1(s_0, s_1, Enc_{pk}(1), pk)$ exactly once. The remaining $n-1$ *Rmap*'s calls to $Rmap_1$ are either with argument $Enc_{pk}(0)$ or $Enc_{pk}(r)$, where $r \in_R D_{M_{pk1}}$. Recall that the result of the first type of call can be simulated by the simulator $Sim(s, pk)$ of $(-1, 1)$-RM. The second type of call can be simulated by the simulator $Sim_0(pk)$ of $(-1, 1)$-RM.

$Sim_{\text{CEM}}(pk, s)$ thus proceeds as follows, using $Sim_0$ and *Sim*. It runs $Sim_0(pk)$ $n-1$ times and $Sim(s, pk)$ once, obtaining a vector $z'$ of $n$ simulated mappings. $Sim_{\text{CEM}}(s, pk)$ outputs a random permutation $\pi'(z')$. From the above arguments it follows that $Sim_{\text{CEM}}(pk, s)$ statistically simulates the output of *Rmap*. □

### 5.1.4 A general $(v_0, v_1)$-RM construction

We informally present a construction for any two constants $v_0, v_1$. We note that it can be naturally generalized for a polynomial number of constants $v_1, \ldots, v_n$.

*Rmap* proceeds as follows. First, as in Construction 5, add trailing zeros to $s_0, s_1$ to distinguish them from random elements in $D_{P_{pk}}$. For $i = 1, 2$ do the following. Choose random linear functions $f_i = a_i x + b_i$ on the plaintext domain $D_{P_{pk}}$ of the underlying (Paillier) encryption, such that $f_i(v_i) = s_i$. Apply $f_i$ to the encrypted input, obtaining $Enc_{pk}(s_i)$ if $x = v_i$, or an encryption of a random value otherwise. Re-randomize and randomly permute the two obtained encryptions. It is easy to see that this sequence encodes at most a single secret $s_i$ and contains no other information. *Rec* decrypts the vector, recognizes the secret and outputs it with overwhelming probability.

This $(v_0, v_1)$-RM can be used with Construction 6, producing GT-CEM with slightly different performance properties. Because this $(v_0, v_1)$-RM uses larger domains of mappings $D_{M_{pk}}$ than Construction 5, the resulting GT-CEM is less efficient for transferring smaller secrets. When the transferred secrets are large, this $(v_0, v_1)$-RM performs better due to slightly smaller loss in bandwidth due to redundancy in secrets. See Table in Section 5.3 for detailed comparisons.

### 5.1.5 Resource analysis

We evaluate the message and modular multiplication efficiency of Construction 6, used with $(-1, 1)$-RM of Section 5.1.2 (which we refer to as CEM1) and of Section 5.1.4 (CEM2). This analysis in this section is similar to, but less detailed than that of GT-

SCOT of Section 4.2.1.

The generated encryption key is reused for a polynomial number of executions of our protocols, thus we do not count the relatively small computational cost of key generation. Let $n$ be the length of inputs $x$ and $y$ in base 2, and $N$ be the size of the plaintext domain of the Paillier scheme. Then the message complexity (the size of the output of *Rmap*) of CEM1 is $l_1 = n\log(N^2) = 2n\log N$ bits, and that of CEM2 is $l_2 = 2n\log(N^2) = 4n\log N$. We do not count the encrypted inputs $x, y$ for message complexity, since their length is usually small, and, in many settings, they are not sent to $S$, but computed by $S$.

To encrypt the $2n$ input bits, expected $3n\log N$ multiplications are required, using square-and-multiply. (The costs of encrypting $-1$ and 1 are the same, since $g^{-1}$ can be precomputed and reused throughout the protocol execution.) Step 1a of Construction 6 requires $n$ multiplications; inversions here are not necessary since *Rmap* may be given encryptions of $-y_i$ instead of $y_i$. Thus this cost can be neglected. Step 1b requires $1.5n\log N$ multiplications. Step 1c of CEM1 requires approximately $4.5n\log N$ multiplications (expected $3\log N + 1$ multiplications for application of the linear function $f$, and $1.5\log N$ to re-randomize the encryption). Step 1c of CEM2 requires double of that effort, $9n\log N$ multiplications.

*Rec* of CEM1 (resp. CEM2) costs an expected $n\log N$ (resp. $2n\log N$) multiplications. Indeed, each decryption costs $2\log N$ multiplication, but we expect to perform only half of the $n$ (resp. $2n$) of them before *Rec* recovers the secret and halts.

In total, CEM1 (resp. CEM2) requires no more than $\approx 10n\log N$ (resp. $\approx 15.5n\log N$) modular multiplications. Of those, $6n\log N$ (resp. $10.5n\log N$) are performed by the server, and $4n\log N$ (resp. $5n\log N$) are spent for encrypting inputs and reconstructing the output. Note that the encryption and re-encryption multiplications can be precomputed once the encryption scheme is initialized.

Our modular multiplications are four times slower than those of [9, 15], since they are performed $\bmod N^2$, while the Goldwasser-Micali multiplications (used in [9, 15]) are $\bmod N$.

One execution of CEM1 (resp. CEM2) allows transfers of secrets of size up to $(\log N)/2 - \lambda$ (resp. $\log N - \lambda$) for the same cost.

Care must be taken in choosing appropriate parameters for comparisons of our results with the performance of other schemes, in particular those based on the potentially weaker quadratic residuosity assumption ([9, 15]). Note that in practice no known attack on the Paillier system is better than factoring the modulus $N$. Clearly, factoring based attacks would also be effective against the GM scheme with the same modulus size. Thus we assume that the security of Paillier and GM schemes with the same size moduli is approximately the same.

The performance comparisons are summarized in the Table in Section 5.3.

### 5.1.6 CEM for any NC$^1$ predicate from homomorphic encryption

We note that it is possible to construct CEM for any NC$^1$ predicate $Q$, using, for example, an information-theoretic abstraction of Yao's garbled circuit [23]. The idea is to assign two specially constructed secrets to each input wire of the (polysize) formula representation of the NC$^1$ circuit. Here each secret corresponds to one of the two possible wire values. The secrets satisfy the following property: a set of secrets, one for each wire of the circuit, allows us to compute the value of the circuit on the corresponding input, and carries no other information.

It is easy to use the homomorphic encryption properties to allow *Rec* to reconstruct only one appropriate secret for each wire. Combined with the tools discussed in the previous paragraph, this implies CEM for any NC$^1$ predicate.

### 5.2 Protocol constructions from GT-CEM

As mentioned in the discussion of CEM in Section 5, natural protocol constructions immediately arise from CEM in the semi-honest model. We demonstrate this on a special case of PSPP of [9], where the server $S$ runs the auction with two bidders $C_0, C_1$. (Our solution can naturally accommodate more bidders, using, for example, the technique of Section 5.2 of [9].) As discussed in Section 2 and [9], in the initialization phase, each of the clients generates and publishes his public key $pk_i$ with $S$.

The main *selection* phase proceeds as follows. Each client $C_i$ sends to $S$ two encryptions of his input, with his own and with the other client's public keys (i.e. $S$ obtains $\mathit{Enc}_{pk_i}(x_i), \mathit{Enc}_{pk_{1-i}}(x_i)$) from $C_i$). $S$ applies GT-CEM twice (once under each key) and sends the outputs of *Rmap* to the corresponding $C_i$ for reconstruction. That is, $S$ sends $m_i = \mathit{Rmap}(s_0, s_1, \mathit{Enc}_{pk_i}(x_i), \mathit{Enc}_{pk_i}(x_{1-i}), pk_i)$ to each $C_i$, who then applies $\mathit{Rec}(sk_i, m_i)$ and obtains $s_1$ if his bid is greater and $s_0$ otherwise. (We note that the receipt of the non-winning $s_0$ is crucial to hide the rank of the bid of $C_i$ in auctions with more than two parties [9].)

It is easy to see that this protocol is secure in the semi-honest model. Indeed, by the definition of CEM, each $m_i$ contains only the intended secret and no other information. Further, it is not hard to see that computationally-bounded $S$ does not learn anything from seeing semantically secure encryptions of clients' bids (under a natural assumption that the secrets $s_0, s_1$ are a polytime computable function of the transcript of $S$'s view of execution of the auction and arbitrary information available prior to the key generation phase).

#### 5.2.1 Handling malicious behaviours

One of the main reasons for the introduction of the semi-honest facilitator is the simplification and efficiency improvement of protocols. In this discussion, we assume the presence of such semi-honest $S$ running *Rmap* and discuss methods of protection against malicious behaviour of other participants. We note that the CEM model is well suited for this task, since the malicious actions of parties are limited to improper input submission and reporting of the decoded output.

First, we observe that the free choice of secrets is a powerful tool. For example, when secrets are randomly chosen, they may serve as a proof of the value of $Q$ in the evaluated $Q$-CEM. Indeed, the recipient of $s_i$ is not able to claim $Q(x, y) = 1-i$, since he cannot obtain $s_{1-i}$. Further, for example, secrets can contain $S$'s signatures, proving the correctness of reconstruction to anyone.

A harder task is ensuring that malicious players do not gain from submitting contrived inputs to $S$. Firstly, zero-knowledge (ZK) techniques could be used to ensure players' compliance with the prescribed protocol. This is often computationally expensive and requires either a common random string or an extra round of interaction. There exist light-weight alternatives to ZK, such as conditional disclosures of Aiello, Ishai and Reingold [1] and Laur and Lipmaa [24]. Their idea, well suited for our setting, is to ensure that an improperly formed input will render useless the obtained output of *Rmap*. For example, suppose *Rmap* requires input encryption $e$ to be a Paillier encryption of a bit (i.e. that $\mathit{Dec}(e) \in \{0, 1\}$). We ensure that non-compliant inputs result in garbled output as follows. Let $s_0, s_1 \in D_S$ be inputs to *Rmap*. We choose a random $r \in_R D_S$ and run *Rmap* with secrets $s_0 \oplus r, s_1 \oplus r$. We now only need a CEM procedure that would transfer $r$ iff $\mathit{Dec}(e) \in \{0, 1\}$, which can be easily constructed.

### 5.2.2 Proxy selling with a secret reserve price

We sketch how to apply GT-CEM to an interesting variant of a proxy selling task, mentioned in the Introduction. Here, the seller wishes to be offline and delegate selling to the semi-trusted $S$. The seller initializes $E_{pk}$, publishes $pk$ and sends an encryption $\mathit{Enc}_{pk}(x)$ of his lowest acceptable price (i.e. reserve) to $S$, who later interacts with buyers as follows. On an encrypted offer $\mathit{Enc}_{pk}(y)$, $S$ replies with $\mathit{Rmap}(s_0, s_1, \mathit{Enc}_{pk}(y), \mathit{Enc}_{pk}(x), pk)$, where $s_1$ serves as $S$'s certification of the successful buyer (e.g. in a form of a signature), and $s_0$ is a non-winning (e.g. empty) secret. Thus, successful buyers obtain (an encryption of) the contract, which they later present to the seller.

Combining GT-CEM with the general CEM techniques based on secret representations, described in Section 5.1.6, allows us to obtain efficient CEM depending on several GT evaluations. This allows us to proxy sell not only based on a reserve price, but on a price range, delivery date ranges, etc.

## 5.3 Comparison with previous work

We continue the resource analysis of Section 5.1.5. Note that the protocols of [9, 15, 24] can be appropriately modified to be cast as GT-CEM. We summarize the cost of comparable modular multiplications and communication of evaluating GT-CEM based on [9, 15, 24] and our constructions CEM1 and CEM2 (i.e. Construction 5.1.3 instantiated with $(-1, 1)$-RM of Section 5.1.2 and 5.1.4 respectively).

Here $c$-bit secrets are transferred based on comparison of $n$-bit numbers. $\lambda$ and $\nu$ are the correctness and security parameters, and $N > 2^\nu$ is the modulus of the employed encryption scheme (GM for [9, 15] and Paillier for [24] and our work). We do not include the one-time cost of key generation. We measure communication as the size of the output of *Rmap*. We include the cost of encryption of inputs (and assign it to client).

Solutions of [9, 15] transfer one-bit secrets per execution, therefore $c$-bit secrets can be transferred at a factor $c$ cost increase. Our CEM1 (resp. CEM2) protocols transfer secrets of size $c < \nu/2 - \lambda$ (resp. $c < \nu - \lambda$) per execution. Today's common parameters

$\nu \approx 1000, \lambda \approx 40, \ldots, 80$ imply transfers of approximately 450 (resp. 950)-bit secrets per execution of CEM1 (resp. CEM2). For CEM of longer secrets, multiple execution is needed. Note the significant advantage of CEM1 for the most frequent case where the transfer of medium-size secrets is required.

**Costs and Comparisons.** GT-COT of [24] can be modified to obtain GT-CEM similar in cost to CEM2. Solution of Section 5.1 (in a more restricted setting, where one of the compared numbers is given in plaintext) carries approximately half of the cost of CEM2. Other costs and comparisons are summarized below. (The cost of (client-run) GM decryption, used in [15, 9], is not less than $\log N$ modular multiplications. For simplicity, we assume that it is $\log N$.) We note that the numbers for CEM1 and CEM2 reflect the factor of four penalty of performing multiplications $\bmod N^2$, vs. $\bmod N$ in previous solutions. We further note that for one execution of CEM1 and CEM2, $c$ is limited as follows: $c < \nu/2 - \lambda$ for CEM1, $c < \nu - \lambda$ for CEM2.

| Protocol | Comparable Modular Multiplications | | | Communication |
|---|---|---|---|---|
| | client | server | total | |
| of [15] | $4nc\lambda \log N$ | $24nc\lambda$ | $32nc\lambda + 4nc\lambda \log N$ | $4nc\lambda \log N$ |
| of [9] | $8n^2c \log N$ | $12n^2c$ | $12n^2c + 8n^2c \log N$ | $8n^2c \log N$ |
| CEM1 | $16n \log N$ | $24n \log N$ | $40n \log N$ | $2n \log N$ |
| CEM2 | $20n \log N$ | $42n \log N$ | $62n \log N$ | $4n \log N$ |

## 6 Conclusions

We considered the problem of securely evaluating the GT predicate in two important settings. We discussed several extensions and applications. In particular, we showed how to build secure auction protocols from our basic constructions. We gave corresponding general definitions, which may be of independent interest. Our protocols are elegant, simpler and more efficient than previously known.

## References

[1] William Aiello, Yuval Ishai, and Omer Reingold, *Priced Oblivious Transfer: How to Sell Digital Goods*. Advances in Cryptology – EUROCRYPT 2001, LNCS 2045, pp. 119–135. Springer-Verlag, 2001.

[2] Donald Beaver, Silvio Micali, and Phillip Rogaway, *The round complexity of secure protocols*. Proc. 22nd ACM Symp. on Theory of Computing, pp. 503–513, 1990.

[3] Ian F. Blake and Vladimir Kolesnikov, *Strong Conditional Oblivious Transfer and Computing on Intervals*. Advances in Cryptology – ASIACRYPT 2004, LNCS 3329, pp. 515–529. Springer-Verlag, 2004.

[4] ______, *Conditional Encrypted Mapping and Comparing Encrypted Numbers*. Financial Cryptography and Data Security, FC 2006, LNCS 4107, pp. 206–220. Springer-Verlag, 2006.

[5] Christian Cachin, *Efficient private bidding and auctions with an oblivious third party*. Proc. ACM CCS, pp. 120–127. ACM Press, 1999.

[6] Christian Cachin, Jan Camenisch, Joe Kilian, and Joy Müller, *One-Round Secure Computation and Secure Autonomous Mobile Agents*. ICALP '00: Proceedings of the 27th International Colloquium on Automata, Languages and Programming, pp. 512–523. Springer-Verlag, London, UK, 2000.

[7] David Chaum, Claude Crépeau, and Ivan Damgård, *Multiparty unconditionally secure protocols*. Advances in Cryptology – CRYPTO 87, LNCS 293, pp. 462–462. Springer-Verlag, 1988.

[8] Ronald Cramer, Serge Fehr, Yuval Ishai, and Eyal Kushilevitz, *Efficient Multi-party Computation over Rings.*. Advances in Cryptology – EUROCRYPT 2003, LNCS 2656, pp. 596–613. Springer-Verlag, 2003.

[9] Giovanni Di Crescenzo, *Private Selective Payment Protocols.*. Financial Cryptography and Data Security, FC 2000, LNCS 1962, pp. 72–89. Springer-Verlag, 2000.

[10] Giovanni Di Crescenzo, Rafail Ostrovsky, and Sivaramakrishnan Rajagopalan, *Conditional Oblivious Transfer and Time-Released Encryption*. Advances in Cryptology – CRYPTO 99, LNCS 1592, pp. 74–89. Springer-Verlag, 1999.

[11] Ivan Damgård, Matthias Fitzi, Eike Kiltz, Jesper Buus Nielsen, and Tomas Toft, *Unconditionally Secure Constant-Rounds Multi-party Computation for Equality, Comparison, Bits and Exponentiation*. Theory of Cryptography, TCC 2006, pp. 285–304, 2006.

[12] Ivan Damgård and Mats Jurik, *A Generalisation, a Simplification and Some Applications of Paillier's Probabilistic Public-Key System*. Public Key Cryptography, PKC 2001, pp. 119–136, 2001.

[13] Ivan Damgård, Mats Jurik, and Jesper Buus Nielsen, *A generalization of Paillier's public-key system with applications to electronic voting*, `http://www.daimi.au.dk/~ivan/GenPaillier_finaljour.ps`.

[14] Yvo Desmedt, *Unconditionally Secure Authentication Schemes and Practical and Theoretical Consequences*. Advances in Cryptology – CRYPTO 85, LNCS 218, pp. 42–55. Springer-Verlag, 1986.

[15] Marc Fischlin, *A Cost-Effective Pay-Per-Multiplication Comparison Method for Millionaires*. RSA Security 2001 Cryptographer's Track, LNCS 2020, pp. 457–471. Springer-Verlag, 2001.

[16] Michael J. Freedman, Kobbi Nissim, and Benny Pinkas, *Efficient Private Matching and Set Intersection*. Advances in Cryptology – EUROCRYPT 2004, LNCS 3027, pp. 1–19. Springer-Verlag, 2004.

[17] Steven D. Galbraith, *Elliptic Curve Paillier Schemes*, Journal of Cryptology 15 (2002), pp. 129–138.

[18] Yael Gertner, Yuval Ishai, Eyal Kushilevitz, and Tal Malkin, *Protecting data privacy in private information retrieval schemes*. Proc. 30th ACM Symp. on Theory of Computing, pp. 151–160. ACM Press, New York, NY, USA, 1998.

[19] Oded Goldreich, *Foundations of Cryptography*, 2: Basic Applications. Cambridge University Press, 2004.

[20] Shai Halevi, *Efficient Commitment Schemes with Bounded Sender and Unbounded Receiver*, Journal of Cryptology 12 (1999), pp. 77–89.

[21] Murat Kantarcioglu and Chris Clifton, *Privacy-preserving distributed mining of association rules on horizontally partitioned data*. ACM SIGMOD Workshop on Research Issues on Data Mining and Knowledge Discovery (DMKD'02), 2002.

[22] Joe Kilian, *Founding cryptography on oblivious transfer*. Proc. 20th ACM Symp. on Theory of Computing, pp. 20–31. ACM, Chicago, 1988.

[23] Vladimir Kolesnikov, *Gate Evaluation Secret Sharing and Secure One-Round Two-Party Computation*. Advances in Cryptology – ASIACRYPT 2005, LNCS 3788, pp. 136–155. Springer-Verlag, 2005.

[24] Sven Laur and Helger Lipmaa, *Additive Conditional Disclosure of Secrets And Applications*, Cryptology ePrint Archive, Report 2005/378, 2005, `http://eprint.iacr.org/`.

[25] Yehuda Lindell and Benny Pinkas, *Privacy Preserving Data Mining*. Advances in Cryptology – CRYPTO 2000, LNCS 1880, pp. 20–24. Springer-Verlag, 2000.

[26] ______, *A proof of Yao's protocol for secure two-party computation*, Cryptology ePrint Archive, Report 2004/175, 2004, `http://eprint.iacr.org/`.

[27] Moni Naor and Kobbi Nissim, *Communication preserving protocols for secure function evaluation*. Proc. 33rd ACM Symp. on Theory of Computing, pp. 590–599. ACM Press, New York, NY, USA, 2001.

[28] Moni Naor, Benny Pinkas, and Reuben Sumner, *Privacy Preserving Auctions and Mechanism Design*. 1st ACM Conf. on Electronic Commerce, pp. 129–139, 1999.

[29] Takashi Nishide and Kazuo Ohta, *Constant-Round Multiparty Computation for Interval Test, Equality Test, and Comparison*, IEICE Transactions 90-A (2007), pp. 960–968.

[30] Pascal Paillier, *Public-Key Cryptosystems Based on Composite Degree Residuosity Classes*. Advances in Cryptology – EUROCRYPT 99, LNCS 1592, pp. 223–238. Springer-Verlag, 1999.

[31] Michael O. Rabin, *How To Exchange Secrets with Oblivious Transfer*, Cryptology ePrint Archive, Report 2005/187, 2005, `http://eprint.iacr.org/`.

[32] Phillip Rogaway, *The round complexity of secure protocols*, Ph.D. thesis, MIT, 1991.

[33] Tomas Sander, Adam Young, and Moti Yung, *Non-Interactive CryptoComputing for $NC^1$*. Proc. 40th IEEE Symp. on Foundations of Comp. Science, pp. 554–566. IEEE, New York, 1999.

[34] Andrew C. Yao, *Protocols for Secure Computations*. Proc. 23rd IEEE Symp. on Foundations of Comp. Science, pp. 160–164. IEEE, Chicago, 1982.

[35] ______, *How to Generate and Exchange Secrets*. Proc. 27th IEEE Symp. on Foundations of Comp. Science, pp. 162–167. IEEE, Toronto, 1986.

Received 5 November, 2007; revised 5 January, 2009

**Author information**

Ian F. Blake, Department of Electrical and Computer Engineering, University of Toronto, Canada.
Email: `ifblake@comm.utoronto.ca`

Vladimir Kolesnikov, Bell Laboratories, Alcatel-Lucent, 600 Mountain Ave., Murray Hill, NJ 07974, USA.
Email: `kolesnikov@research.bell-labs.com`