Strong Conditional Oblivious Transfer and Computing on Intervals

Ian F. Blake[1] and Vladimir Kolesnikov[2]

[1] Dept. Elec. and Comp. Eng, University of Toronto, Toronto, ON, M5S 3G4, Canada, `ifblake@comm.utoronto.ca`

[2] Dept. Comp. Sci., University of Toronto, Toronto, ON, M5S 3G4, Canada, `vlad@cs.utoronto.ca`

Abstract. We consider the problem of securely computing the Greater Than (GT) predicate and its generalization – securely determining membership in a union of intervals. We approach these problems from the point of view of Q-Conditional Oblivious Transfer (Q-COT), introduced by Di Crescenzo, Ostrovsky and Rajagopalan [4]. Q-COT is an oblivious transfer that occurs *iff* predicate Q evaluates to `true` on the parties' inputs. We are working in the semi-honest model with *computationally unbounded* receiver.

In this paper, we propose: (i) a stronger, simple and intuitive definition of COT, which we call *strong* COT, or Q-SCOT. (ii) A simpler and more efficient one-round protocol for securely computing GT and GT-SCOT. (iii) A simple and efficient modular construction reducing SCOT based on membership in a union of intervals (UI-SCOT) to GT-SCOT, producing an efficient one-round UI-SCOT.

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 base functionality we consider - Greater Than (GT) - is one of the most basic and commonly used. Secure evaluation of GT is also one of the most famous and well-researched problems in cryptography. There exist a vast number of applications relying on it, such as auction systems or price negotiations.

Another typical example would be 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 [12, 14]. Because of the large size of the databases, even a minor efficiency gain in computing GT results in significant performance improvements.

Other functionalities – memberships in a set of intervals and their conjunctions and disjunctions – are less studied, but nevertheless are very useful. Their immediate uses lie in appointment scheduling, flexible timestamp verification, expression evaluation, in the areas of computational geometry, biometrics, and many others. Certain kinds of set membership problems, as studied by Freedman, Nissim and Pinkas [7], 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) 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 [7].

The setting with computationally unbounded receiver (Alice) is very appealing, both for oblivious transfer and general computations. Numerous papers consider unconditional security against one or more parties, in particular, the receiver, e.g. [2, 3, 5, 11, 17]. Practical one-round computation with unbounded first party (Alice) currently seems to be hard to achieve. The best known general approach [21] offers only polynomial efficiency and only for computing NC^1 circuits. At the same time, if Alice is bounded, we could use very efficient Yao's garbled circuit approach ([15, 17, 20, 22]) at the cost linear with the size of the circuit. We solve the posed problems in the difficult setting (unbounded Alice), while achieving performance only slightly worse than the best known approach in the easier (bounded Alice) setting.

1.1 Our Contributions and Outline of the Work

After presenting preliminary definitions and constructions in Sect. 1.2, we start with a discussion of Conditional Oblivious Transfer (COT) (Sect. 2). We wish to strengthen the current definition of [4] in several respects. Firstly, we observe that the definition of [4] does not require the privacy of the sender's private input. Secondly, we propose and justify the "1-out-of-2" Q-COT, where the receiver obtains one of two possible secret messages depending on Q, but without learning the value of Q. This is opposed to the "all-or-nothing" approach of [4] where the receiver receives either a message or nothing, which necessarily reveals the value of Q. Our approach significantly adds to the flexibility of COT functionalities and allows for more powerful compositions of COT protocols. We propose a definition of *strong* conditional oblivious transfer (SCOT) that incorporates the above observations and some other (minor) points.

Then, in Sect. 3, we discuss previous work on the GT problem and 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 a novel way to arrive at a solution that is more efficient and flexible than the best previously known (of Fischlin [6]) for our model with unbounded Alice. 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 Fischlin's protocol. This results in additional significant efficiency gains.

Then, in Sect. 4, we show how to use the bandwidth of our GT-COT solution and present protocols for efficiently computing SCOT based on the interval membership (I-SCOT) and SCOT based on the membership in a union of k intervals (k-UI-SCOT). Because of their modularity, these protocols can also be constructed based on Fischlin's solution 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 [4] 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. Finally, we compare and summarize resource requirements of schemes of Fischlin, Di Crescenzo et al., and ours in the Table in Sect. 4.2.

1.2 Definitions and Preliminaries

We start by introducing the necessary terminology and notation, and refer the reader to Goldreich [9] for in-depth discussion. We are working in a setting with two semi-honest participants, who use randomness in their computation. 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 Π with a party R on respective inputs x and y by $\mathrm{VIEW}_P^{\Pi}(x, y)$. We note that $\mathrm{VIEW}_P^{\Pi}(x, y)$ is a random variable over the random coins of P and R.

We stress that although our constructions and analysis are presented for a fixed security and correctness[3] parameters ν and λ, we have in mind their asymptotic notions. Therefore, for example, when talking about a view of a party $\mathrm{VIEW}_P^{\Pi}(x, y)$, we mean an ensemble $\{\mathrm{VIEW}_P^{\Pi}(x, y)\}_{\nu,\lambda}$ of views.

We denote statistical closeness of ensembles of random variables X and Y by $X \overset{s}{\equiv} Y$ and their computational indistinguishability by $X \overset{c}{\equiv} Y$. We say a function $\mu : N \mapsto R$ is *negligible* if for every positive polynomial $p(\cdot)$ there exists an N, such that for all $n > N, \mu(n) < 1/p(n)$. We say a probability is *overwhelming* if it is negligibly different from 1.

Homomorphic Encryption. Our constructions use semantically secure public key probabilistic *additive homomorphic encryption*. Informally, a scheme is probabilistic (or *randomized*), if its encryption function uses randomness to encrypt a plaintext as one of many possible ciphertexts. It allows *re-randomization* if a random encryption of a plaintext can be computed from its ciphertext and the public key. In our work, we will rely on the unlinkability of encryptions of the same message. An encryption scheme (G, E, D) is *homomorphic*, if for some operations $\oplus$ and $\otimes$ (defined on possibly different domains), it holds that $D(E(x \oplus y)) = D(E(x) \otimes E(y))$. A scheme is called additively (multiplicatively) homomorphic if it is homomorphic with respect to the corresponding operation (e.g. additive scheme allows to compute $E(x + y)$ from $E(x)$ and $E(y)$). Many of the commonly used schemes are homomorphic. For example, the ElGamal

[3] Correctness parameter specifies the allowed probability of error in the protocols.

scheme is multiplicatively homomorphic, and Goldwasser-Micali [10] and Paillier [18] schemes are additively homomorphic. Unfortunately, it is not known whether there exists a scheme that is algebraically (i.e. both additively and multiplicatively) homomorphic. We note that an additively homomorphic scheme allows multiplication by a known constant, i.e. computing $E(cx)$ from $E(x)$ and c, via repeated addition.

The Paillier Cryptosystem. Our protocols require an additional property of the encryption scheme: the large plaintext size, or *bandwidth*. The Paillier scheme [18] satisfies all our requirements, and we will instantiate all our protocols with it. We present it for completeness, 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 $N\alpha$ modulo N^2, for some integer α. The public key $pk = (N, g)$ and the secret key $sk = \lambda(N) = \text{lcm}((p-1),(q-1))$, where $\lambda(N)$ is the Carmichael's lambda function.

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

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

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

The underlying security assumption is that the so-called composite residuosity class problem is intractable (called the CCRA assumption). It it potentially stronger than the RSA assumption, as well as the quadratic residuosity assumption, used in [6]. We refer the interested reader to [18] for further details.

2 Strong Conditional Oblivious Transfer

The notion of COT was introduced by Di Crescenzo, Ostrovsky and Rajagopalan [4] in the context of timed-release encryption. It is a variant of Oblivious Transfer (OT) introduced by Rabin [19]. 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.

2.1 Our Definitions

We start by describing several ways of strengthening the existing definition with the goal of increasing modularity and widening 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 [4] lacks the requirement of secrecy of the sender's private input. We would like the new definition to include this requirement.

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 definition since while a SCOT protocol can be trivially modified to satisfy COT definitions of [4], the opposite does not (efficiently) hold[4]. Further, note that it follows from our requirements that a Q-SCOT protocol can be trivially modified into a $(\neg Q)$-SCOT protocol. This also does not hold for COT. We will use this important property in our constructions later in the paper.

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. Let ν be the security parameter and λ be the correctness parameter, upperbounding error probability by $O(2^{-\lambda})$. Let D_I and D_S be the respective domains of 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 $Q : D_I \times D_I \mapsto \{0, 1\}$ be a predicate. Consider the SCOT functionality:

Functionality 1

$$f_{Q-\mathrm{SCOT}}(x, (y, s_0, s_1)) = \begin{cases} (s_1, \textit{empty string}) & \textit{if } Q(x, y) = 1, \\ (s_0, \textit{empty string}) & \textit{otherwise} \end{cases} \tag{1}$$

There are many models 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[5]. 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 [9] for in-depth presentations of definitions of security in many interesting models.

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

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

[4] Clearly, because secure multi-party computation can be based on OT (Kilian [13]), COT implies SCOT. This solution, however, is inefficient.

[5] 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.

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 explicate the definition for this setting.

Definition 2. *Let receiver R, sender S, their inputs x and y, secrets s_1 and s_0, unary parameters ν and λ, and predicate Q be as discussed above. We say that Π 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 λ: 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 a simulator* Sim_R*, such that for any x, y, s, s' from appropriate domains:*

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

– Security against S. *(S gets no efficiently computable information about x) There exists an efficient simulator Sim_S, such that for any $x, (y, s_0, s_1)$ from appropriate domains:*

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

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. This is not possible with COT, as R is only able to provide such proof if $Q(x,y) = 1$.

3 The GT-SCOT Protocol

Research specifically addressing the GT problem is quite extensive. It was considered (as a special case) in the context of general secure multi-party computation [1, 15, 17, 20, 23, 22], whose solution is now well-known and celebrated. This general approach is impractical. However, because the circuit for computing GT is quite small, it is the best currently known one-round solution in the model with the computationally bounded Alice. As people searched for efficient solutions to special classes of problems in different models, more efficient GT solutions implicitly appeared. Naor and Nissim [16] 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 ϵ is the tolerated probability of error. Sander, Young and Yung [21] showed that all functionalities in NC^1 (including GT) can be computed by a one-round polytime protocol. Their solution is secure against unbounded Alice. Unfortunately, when used with the natural shallow GT circuit[6] (which seems to be optimal for their approach), it requires at least n^4 modular multiplications and $n^4 \log N$ communication (where n is the input size, and N is the GM modulus used).

Finally, in 2001, Fischlin [6] proposed a solution that significantly reduced the number of modular multiplications, while also reducing the message size and maintaining the minimal one-round efficiency. This is the best previously known solution to the GT problem in the model with unbounded Alice. The number of modular multiplications required to complete his protocol is $8n\lambda$, where $2^{-\lambda}$ is the allowed error probability. The message complexity (in bits) is $n \log N(\lambda+1)$. Fischlin also 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.

3.1 Our Construction

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 recently, and several variants, including an elliptic curve version [8], have appeared. Using more efficient implementations may further improve our results.

Let $(\mathrm{Gen}, \mathrm{Enc}, \mathrm{Dec})$ be the instance generation, encryption and decryption algorithms, respectively, of such a scheme. As in Definition 2, let R and S be the receiver and the sender with inputs x and y respectively and common parameters ν and λ. 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^{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 may mean the vector $< x_1, x_2, ..., x_n >$, and by $\mathrm{Enc}(x)$ we mean a vector $< \mathrm{Enc}(x_1), \mathrm{Enc}(x_2), ..., \mathrm{Enc}(x_n) >$. We will also write $\mathrm{Enc}(x)$ instead of $\mathrm{Enc}_{pk}(x)$, 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.

[6] The circuit based on the formula used by Fischlin's protocol [6].

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 $\max\{\nu, |d_S| + \lambda\}$.

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 λ to satisfy the security and correctness requirements of the encryption scheme. In practice,* $\log N (\approx 1000) \gg \lambda(\approx 40..80)$, *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 Sect. 4.2.*

Observation 2 *There is a negligible (in λ) 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 $\{>, <\}$. Furthermore, we assume that $x \neq y$. This can be enforced by mapping, for instance, $x \mapsto 2x, y \mapsto 2y + 1$. The mapping can be done entirely by S. 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.

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 $(pk, Enc(x_1), ..., Enc(x_n))$ to S.*
2. *S computes the following, for each $i = 1..n$:*
   (a) an encryption of the difference vector d, where $d_i = x_i - y_i$.
   (b) an encryption of the flag vector f, where $f_i = x_i$ XOR $y_i = (x_i - y_i)^2 = x_i - 2x_i y_i + y_i$.
   (c) an encryption of vector γ, where $\gamma_0 = 0$ and $\gamma_i = 2\gamma_{i-1} + f_i$.
   (d) an encryption of vector δ, where $\delta_i = d_i + r_i(\gamma_i - 1)$, where $r_i \in_R \mathbb{Z}_N$.
   (e) a random encryption of vector μ, where $\mu_i = \frac{s_1 - s_0}{2}\delta_i + \frac{s_1 + s_0}{2}$
   and sends a random permutation $\pi(Enc(\mu))$ to R.
3. *R obtains $\pi(Enc(\mu))$, decrypts it, and determines the output as follows: if μ contains a single $v \in D_S$, output v, otherwise abort.*

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

Proof. (sketch): We will now show that the protocol correctly computes the desired functionality. It is easy to see that the homomorphic properties of the encryption scheme allow S to perform all necessary operations. In particular, step 2b is possible because y_i are known to S.

Observe that the flag vector f is a $\{0,1\}$-vector, with the ones in positions where x and y differ. Furthermore, γ is a vector with the following structure: it starts with zero 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 Z_N is negligible, and p and q are chosen randomly and independently of x and y.

Let ind_1 be the (only) position where $\gamma_{\text{ind}_1} = 1$. This position is where x and y first differ, and thus d_{ind_1} determines $\text{GT}(x,y)$. The transformation $(\gamma, d) \rightarrow \delta$ of step 2d randomizes all coordinates of δ, while setting δ_{ind_1} to the value of d_{ind_1}. Because, with overwhelming probability, $(\gamma_i - 1) \in \mathbb{Z}_N^*$, multiplying it by $r_i \in_R \mathbb{Z}_N$ randomizes δ perfectly 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, the 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 mapping we are looking for. The random re-encryption step 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 of the determining d_i.

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

Security of R (against the semi-honest S) trivially holds because of the semantic security properties of the employed encryption scheme.

We now prove security of S against an unlimited semi-honest R by constructing a protocol view simulator $\text{Sim}_R(x, s)$, where x is the input, and s is the output of the protocol. $\text{Sim}_R(x, s)$ has to generate a distribution statistically close to the view of R in a real execution - $\text{VIEW}_R(x, (y, s_0, s_1)) = \{x, r, \text{Enc}(\pi(\mu))\}$, 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. $\text{Sim}_R(x, s)$ 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 μ': for $i = 1..n$, pick random $\mu'_i \in_R \mathbb{Z}_N$. It then replaces a random element of μ' with the received s, and outputs $\{x, r', \text{Enc}_{pk'}(\mu')\}$, where $\text{Enc}_{pk'}(\mu')$ is a vector of random encryptions of coordinates of μ' under the 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 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 other secret is possible during the real execution, with negligible probability. □

We observe that a GT-SCOT protocol, such as presented above, immediately implies 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 $s_1 = 1$), 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).

3.2 Resource Analysis

We evaluate the message and modular multiplication efficiency of our construction based on the use of Paillier encryption scheme. We note that we do not include the relatively small computational cost of key generation, to be consistent with the compared results of [4] and [6]. Let n be the length of inputs x and y in binary, N-the size of the plaintext domain of the Paillier scheme. Then message complexity of Construction 1 is $l = 2n\log(N^2) = 4n\log N$ bits.

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, $\log N$ multiplications are required. Observe that it is not necessary to perform expensive randomized encryption in the intermediate steps of S. This allows us to make do with only w multiplications for each of the steps 2a, 2b, $2n$ - for step 2c, and $(\log N + 2)n$ - for step 2d, and $(|s_i| + \log N)n \leq 2n\log N$ - for step 2e of the protocol. We note that if we do not perform the transformation of step 2e (when, for example, computing GT), we only need $n\log N$ multiplications for the last step.

Decryption takes $2n\log N$ multiplications. Thus, in total, the protocol requires no more than $(5n+1)\log N + 6n$ modular multiplications ($(4n+1)\log N + 6n$ for GT). We stress that transferring up to $\log N - \lambda$ bit secrets requires the same resources. We observe that the encryption and re-encryption multiplications can be precomputed once the encryption scheme is initialized.

We now compare the efficiency of our approach to that of Fischlin [6], 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 GM scheme with the same modulus size. Thus, having already assumed CCRA (see Sect. 1.2), we also assume that the security of Paillier and GM schemes with the modulus of the same size are approximately the same.

Compared with [6], our scheme offers a factor of $\lambda/4$ improvement in message complexity: ($(4n\log N)$ vs $(n\log N(\lambda+1))$ bits). We pay higher cost in the number of modular multiplications: ($(4n+1)\log N + 6n$) vs ($6n\lambda$). Additionally, our multiplications are four times slower, since we are working with modulus length twice that of the Goldwasser-Micali encryption scheme employed in [6]. These comparisons are summarized in the Table in Sect. 4.2.

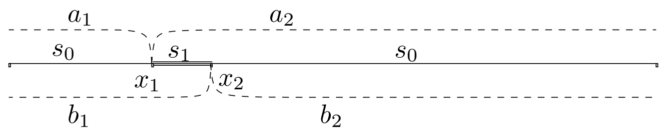
4 SCOT for Unions of Intervals

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. We build these 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 Sect. 4.1 we show how to reduce UI-SCOT to I-SCOT and I-SCOT to GT-SCOT. In our model, secure reductions provide us with secure protocols when the underlying oracles are replaced by their secure implementations (see Goldreich [9] for the composition theorem.) Furthermore, in our model the oracles' implementations may be run in parallel, which, with our implementations, provides secure one-round protocols for I-SCOT to UI-SCOT.

4.1 The UI-SCOT protocol

Without loss of generality, we assume that the domain of secrets D_S is an additive group[7] $\mathbb{Z}^+_{d_S}$. All additions of secrets will be done in D_S, unless specified otherwise. In the I-SCOT setting, S's input $x_1, x_2 \in D_I$ represents an interval I, and s_1 (resp. s_0) are to be obliviously transferred if $x \in I$ (resp. $x \notin I$), for R's input $x \in D_I$. The following diagram illustrates the idea of the reduction of I-SCOT to GT-SCOT:



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" ($D_I \backslash I$). 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)*

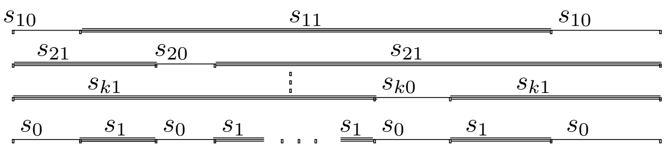
1. *S randomly chooses $a_1 \in D_S$ and sets $b_1, a_2, b_2 \in D_S$ to satisfy $s_0 = a_1 + b_1 = a_2 + b_2$ and $s_1 = a_2 + b_1$*
2. *-Reduction: R and S (in parallel) invoke oracles for GT-SCOT$(a_1|a_2?x < x_1)$ and GT-SCOT$(b_1|b_2?x < x_2)$.*
3. *R obtains $a', b' \in D_S$ from GT-SCOT oracle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

[7] 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.

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

Proof. (sketch): 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): $\mathrm{VIEW}_R(x,(x_1,x_2,s_0,s_1)) = \{x,r_1,r_2\}$, where r_1,r_2 are the sent (via the GT-SCOT oracles) a_i,b_j. Let s be the transferred secret. Then $\mathrm{Sim}_R(x,s) = \{x,r'_1,r'_2\}$, 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 view of R. □

We now wish to 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 one "regular" and $k-1$ "cutout" intervals as illustrated on the following diagram.



The bottom line represents the input set of intervals on the domain, and all other lines represent the constructed (by S) intervals that together correspond to this set. 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 need to 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 $s_0 = (\sum_{i \neq j} s_{i1}) + s_{j0}, \forall j = 1..k$ Observe that the second condition is equivalent to requiring $s_1 - s_0 = s_{j1} - s_{j0}, \forall j = 1..k$.

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

1. *S chooses $s_{11}, ..., s_{(k-1)1} \in_R D_S$ and sets $s_{k1} = s_1 - \sum_{i=1..k-1} s_{i1}$ and $s_{i0} = s_{i1} - (s_1 - s_0), i = 1..k$.*
2. *-Reduction: S and R (in parallel) invoke oracles for I-SCOT$(s_{i1}|s_{i0}?x \in I_i)$, for each $i = 1..k$.*
3. *R obtains $a_1, ..., a_k \in D_S$ from k oracle executions and outputs $\sum_i a_i$.*

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

Proof. (sketch): 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 view of R $\mathrm{VIEW}_R(x,y) = \{x, r_1, ..., r_k\}$, where $r_1, ..., r_k$ are the oracle sent elements of D_S defined by step 1 of the construction. Let s be the transferred secret. Then $\mathrm{Sim}_R(x,s) = \{x, r'_1, ..., r'_k\}$, 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 numbers that sum up to the same value s. □

The $(\bigwedge_i Q_i(x_i, y_i))$-COT Protocol. We now build $\bigwedge_i Q_i(x_i, y_i))$-COT (in the sense of [4]) using oracles for corresponding Q_i-SCOT. R now has input $x_1, ..., x_n$, and S has $y_1, ..., y_n$. 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 like we did for GT-SCOT, by, for example, extending the domain of secrets D_S to 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 $r_1, ..., r_n, s_1, ..., s_{n-1} \in_R D'_S$ and sets (in D'_S) $s_n = s - \sum_{i=1..n-1} s_i$.*
2. *R and S in parallel invoke oracles for Q_i-SCOT$(s_i|r_i?Q_i(x_i, y_i))$, $\forall i = 1..n$.*
3. *R obtains $a_1, ..., a_n \in D'_S$ from the Q_i-SCOT oracle executions and sets $v = \sum_i a_i$. R outputs v, if $v \in D_S$, and outputs $\perp$ otherwise.*

Theorem 4. *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: The simple proof is very similar to the previous ones and is omitted. □

Corollary 1. *There exists (via construction 4 and DeMorgan laws) efficient one-round protocols for computing conjunction and disjunction of memberships in sets of intervals, secure against computationally unlimited R.*

4.2 Resource Analysis

We continue and expand the resource analysis of Sect. 3.2. Recall that λ and ν are the correctness and security parameters. As discussed in Observation 1, we choose $\nu = \log N$ and λ as in [6]. This determines the secrets domain D_S to be of size $2^{\nu-\lambda}$. As noted in Sect. 3.2, we do not include the cost of key generation in any of the compared solutions.

It is easy to see that Construction 3 makes $2k$ calls to the underlying λ-bit GT-COT oracle. Thus, when using our implementation of GT-SCOT, UI-SCOT requires sending $8kn \log N$ bits and performing about $40kn \log N$ multiplications in group of size N. Using λ-bit GT-SCOT oracle implementation based on Fischlin'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 this scheme.

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, $c<\nu$-λ		k-UI-SCOT	
	mod. mult.	comm.	mod. mult.	comm.	mod. mult.	comm.
of [6]	$8n\lambda$	$\lambda n \log N$	$32nc\lambda$	$4nc\lambda \log N$	$64kn\lambda^2$	$8kn\lambda^2 \log N$
of [4]	$8n$	$4n \log N$	N/A	N/A	N/A	N/A
our work	$16n \log N$	$4n \log N$	$20n \log N$	$4n \log N$	$40kn \log N$	$8kn \log N$

We see no obvious way to transform the schemes of [4] to GT-SCOT, and thus do not include the corresponding resource calculations.

5 Conclusions and Future Work

We presented simple, intuitive and stronger definitions for Q-SCOT. We presented a flexible and efficient scheme for securely computing the GT predicate and GT-SCOT, in the semi-honest setting with unbounded receiver. We then showed simple modular reductions from UI-SCOT to GT-SCOT. In addition to the presented results, we noticed that natural efficient variants of our protocols are resilient to several natural attacks by malicious receivers. Devising versions of our protocols secure in the malicious model is an interesting aspect of further consideration.

Acknowledgments. The second author would like to thank Travis Gagie, Steven Myers, and especially Charles Rackoff for many insightful discussions. He also thanks Marc Fischlin and Pascal Paillier for useful comments relating to their schemes used in this paper.

References

1. D. Beaver, S. Micali, and P. Rogaway. The round complexity of secure protocols. In *Proc. 22nd ACM Symp. on Theory of Computing*, pages 503–513, 1990.
2. Christian Cachin, Jan Camenisch, Joe Kilian, and Joy Muller. One-round secure computation and secure autonomous mobile agents. In *Proceedings of the 27th International Colloquium on Automata, Languages and Programming*, 2000.
3. David Chaum, Claude Crépeau, and Ivan Damgård. Multiparty unconditionally secure protocols. In *Proc. CRYPTO 87*, pages 462–462. Springer-Verlag, 1988. Lecture Notes in Computer Science, vol. 293.
4. G. Di Crescenzo, R. Ostrovsky, and S. Rajagopalan. Conditional oblivious transfer and time-released encryption. In *Proc. CRYPTO 99*, pages 74–89. Springer-Verlag, 1999. Lecture Notes in Computer Science, vol. 1592.

5. Yvo Desmedt. Unconditionally secure authentication schemes and practical and theoretical consequences. In *Proc. CRYPTO 85*, pages 42–55. Springer, 1986. Lecture Notes in Computer Science, vol. 218.
6. Marc Fischlin. A cost-effective pay-per-multiplication comparison method for millionaires. In *RSA Security 2001 Cryptographer's Track*, pages 457–471. Springer-Verlag, 2001. Lecture Notes in Computer Science, vol. 2020.
7. Michael J. Freedman, Kobbi Nissim, and Benny Pinkas. Efficient private matching and set intersection. In *Proc. EUROCRYPT 2004*, pages 1–19. Springer-Verlag, 2004. Lecture Notes in Computer Science, vol. 3027.
8. Steven D. Galbraith. Elliptic curve paillier schemes. *Journal of Cryptology*, 15(2):129–138, 2002.
9. Oded Goldreich. *Foundations of Cryptography: Volume 2, Basic Applications.* Cambridge University Press, 2004.
10. S. Goldwasser and S. Micali. Probabilistic encryption and how to play mental poker keeping secret all partial information. In *Proc. 14th ACM Symp. on Theory of Computing*, pages 365–377, San Francisco, 1982. ACM.
11. Shai Halevi. Efficient commitment schemes with bounded sender and unbounded receiver. *Journal of Cryptology: the journal of the International Association for Cryptologic Research*, 12(2):77–89, 1999.
12. M. Kantarcioglu and C. Clifton. Privacy-preserving distributed mining of association rules on horizontally partitioned data. In *ACM SIGMOD Workshop on Research Issues on Data Mining and Knowledge Discovery (DMKD'02)*, 2002.
13. J. Kilian. Founding cryptography on oblivious transfer. In *Proc. 20th ACM Symp. on Theory of Computing*, pages 20–31, Chicago, 1988. ACM.
14. Yehuda Lindell and Benny Pinkas. Privacy preserving data mining. In *Proc. CRYPTO 00*, pages 20–24. Springer-Verlag, 2000. Lecture Notes in Computer Science, vol. 1880.
15. Yehuda Lindell and Benny Pinkas. A proof of yao's protocol for secure two-party computation. Cryptology ePrint Archive, Report 2004/175, 2004. `http://eprint.iacr.org/`.
16. Moni Naor and Kobbi Nissim. Communication preserving protocols for secure function evaluation. In *Proceedings of the thirty-third annual ACM symposium on Theory of computing*, pages 590–599. ACM Press, 2001.
17. Moni Naor, Benny Pinkas, and Reuben Sumner. Privacy preserving auctions and mechanism design. In *1st ACM Conf. on Electronic Commerce*, pages 129–139, 1999.
18. Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *Proc. EUROCRYPT 99*, pages 223–238. Springer-Verlag, 1999. Lecture Notes in Computer Science, vol. 1592.
19. M. Rabin. How to exchange secrets by oblivious transfer. Technical Report TR-81, Harvard Aiken Computation Laboratory, 1981.
20. Phillip Rogaway. *The round complexity of secure protocols.* PhD thesis, MIT, 1991.
21. Tomas Sander, Adam Young, and Moti Yung. Non-interactive cryptocomputing for NC^1. In *Proceedings 40th IEEE Symposium on Foundations of Computer Science*, pages 554–566, New York, 1999. IEEE.
22. A. C. Yao. Protocols for secure computations. In *Proc. 23rd IEEE Symp. on Foundations of Comp. Science*, pages 160–164, Chicago, 1982. IEEE.
23. A. C. Yao. How to generate and exchange secrets. In *Proc. 27th IEEE Symp. on Foundations of Comp. Science*, pages 162–167, Toronto, 1986. IEEE.