Conditional Encrypted Mapping and Comparing Encrypted Numbers

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

[1] Dept. ECE, University of Toronto, Canada, ifblake@comm.utoronto.ca
[2] Dept. Comp. Sci., University of Toronto, Canada, vlad@cs.utoronto.ca

Abstract. We consider the problem of comparing two encrypted numbers and its extension – transferring one of the two secrets, depending on the result of comparison. We show how to efficiently apply our solutions to practical settings, such as auctions with the semi-honest auctioneer, proxy selling, etc. We propose a new primitive, *Conditional Encrypted Mapping*, which captures common security properties of one round protocols in a variety of settings, which may be of independent interest.

Keywords: Two Millionaires with encrypted inputs, auctions, private selective payments, conditional encrypted mapping.

1 Introduction

In this paper we study secure evaluation of the Greater Than (GT) predicate. It is one of the most basic and widely used functionalities. It plays an especially important role in secure financial transactions and database mining applications.

Auctions and Bargaining. With the 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 highly desirable to keep all other information (e.g. unsuccessful bids) secret.

There has been a large stream of work dedicated to ensuring privacy and security of online auctions and haggling (e.g.,[3, 5, 6, 14]). Our work complements, extends, and builds on it. We discuss the Private Selective Payments protocols of Di Crescenzo [5] and show how our improvements benefit this application.

The need for comparing encrypted numbers. 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 work with, e.g. compare, *encrypted* numbers. We propose a formalization of this setting, as well as new, more efficient GT protocols for it.

Other applications. We mention other interesting applications that benefit from efficient secure evaluation of GT. These applications might need to employ a proxy server S, as above; if so, our work improves their performance as well.

In Distributed Database Mining, 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 very large number of instances of GT [10, 13]. Our improvements also apply to solving interval membership problems (reduced to GT in [2]). The immediate uses lie in appointment scheduling, flexible timestamp verification, biometrics, etc. Certain kinds of set intersection problems, as studied in [7, 9], can be represented succinctly as GT instances, resulting in more efficient solutions using our constructions.

1.1 Our Contributions, Setting and Outline of the Work

We approach several practical problems (auctions, proxy selling, GT) in a variety of settings, concentrating on a setting with a semi-honest helping server.

We are interested in one-round protocols, where clients send their encrypted inputs to a "cypto computer" S, who produces an output that can be decoded by the clients. Such scenarios arise in a variety of practical settings. 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 the reconstruction of the value of the function, and does not contain any other information. This allows to postpone the (easy but tedious) discussion of setting-specific clients' privacy requirements. We formalize (Def. 1) a special case of this notion, which we call *Conditional Encrypted Mapping* (CEM). Here, S has two secrets s_0, s_1, is given encryptions of two values x, y, and outputs something that allows (only) 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 very strong, e.g., precluding Yao's garbled circuit-based solutions.

We propose two new, more efficient CEM protocols for the GT predicate (Sect. 4). We use ideas of the recent protocol of Blake and Kolesnikov [2]. Their protocol 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 Sect. 4.3). The cost of our solution is comparable to that of [2]. We believe this method may be used to improve efficiency of other constructions relying on homomorphic encryptions.

In Sect. 5, we show how our constructions result in new, more efficient, protocols for the examples of private selective payments of Di Crescenzo [5] and proxy selling. We discuss methods of protection against malicious behavior of parties. We mention that efficient CEM schemes exist for any NC^1 predicate (Sect. 4.7).

In Sect. 6 we summarize and compare resource requirements of schemes based on the work of Di Crescenzo [5], Fischlin [6], Laur and Lipmaa [12] and ours.

2 Related Work

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

Variants of CEM. Several notions similar to CEM were previously proposed.

The notion of Conditional Oblivious Transfer (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) [16]. 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 [8] 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. Again, the receiver does not know the secret; it is sent to him only if a predicate holds on the input.

Laur and Lipmaa [12] extend the study of CDS for the case of additive homomorphic encryptions, give generic constructions and specific protocols (GT).

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 [5] described a stronger concept, Symmetrically-private COT, by additionally requiring that both parties' inputs x, y remain private. Later, Blake and Kolesnikov [2], independently proposed and formalized essentially the same notion, which they call Strong COT. Of the above, CEM is most similar to this notion. We note that CEM is a stronger notion, explicitly allowing reuse of generated encryption keys in multiple executions. We also have the feature of not specifying the precise security properties of the used encryptions, allowing for more flexibility and applicability (see Sect. 1.1 and 3 for more discussion).

Auctions and Private Selective Payments Protocols (PSPP). PSPP, introduced by Di Crescenzo [5], 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 the server's and clients' private inputs. Client's inputs may represent their auction bids, and a server's input may be a lowest acceptable price or a required signature. Di Crescenzo considers a natural instance of PSPP, where the highest bidding client obtains the authorization. He considers a setting with a helping 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 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 computationally expensive area. The main contribution of [5] is the novel maximum bidder selection protocols.

Our main contribution, GT-CEM constructions, can be used to replace the core – the selection protocols – of the PSPP of [5] (with corresponding natural modifications of the other two phases). Appropriately modified protocols of Fischlin [6] and Laur and Lipmaa [12] can be similarly used. We discuss more details and the resulting efficiency improvements of our protocols in Sect. 5 and 6.

We mention, but do not discuss in detail the auction protocols for use in the settings, significantly different from ours. Naor, Pinkas and Sumner [14] 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 [3] suggested a protocol for private bidding with the semi-honest server in the setting where the bidders additionally exchange messages between each other.

2.1 Notation, Definitions and Preliminaries

A function $\mu : N \mapsto R$ is *negligible* if for every positive polynomial $p(\cdot)$ there exists an N, such that for all $n > N, \mu(n) < 1/p(n)$. A probability is *overwhelming* if it is negligibly different from 1. Statistical distance between distributions X and Y is defined as $Dist(X, Y) = 1/2 \sum_{\alpha} |Pr(X = \alpha) - Pr(Y = \alpha)|$.

Informally, an encryption scheme $E = (Gen, Enc, Dec)$ 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. Many such schemes (e.g. Paillier [15]) exist.

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

3 Conditional Encrypted Mapping

We consider the setting where one of the players is a facilitator of the computation of the multiparty functionality f. This player – the 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 with semi-honest servers. There, the server S is given encryptions of parties' bids, and he 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. 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 very 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 defined and satisfied with appropriate use of encryption.

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 $(Rmap, Rec)$ an Encrypted Mapping (EM). We formalize a variant of this notion in Def. 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, 12]. See more in Sect. 5.1).

We wish to give a very strong definition, so that the constructions can be used in a variety of settings. In particular, we want our construction to work with all instantiations of used encryption schemes. In many popular encryption schemes (e.g. Paillier [15]) 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 very 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 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 $E = (Gen, Enc, Dec)$. 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 Enc_{pk}. Denote by $Enc_{pk,\alpha}(x)$ the encryption of x under pk using randomness α. Let $Q : D_Q \times D_Q \mapsto \{0, 1\}$ 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 ν be the security parameter[3]. Let D_S be the (fixed) domain of secrets[4].

Definition 1. *(Q-Conditional Encrypted Mapping) A Q -* Conditional Encrypted Mapping *(Q-CEM) 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_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)* $\forall (sk, pk) \leftarrow Gen(\nu), \forall s_0, s_1 \in D_S, \forall \alpha, \beta \in D_{R_{pk}}, \forall x, y \in D_Q$: *with overwhelming probability in ν, taken over random inputs of Rmap:* $Rec(Rmap(s_0, s_1, Enc_{pk,\alpha}(x), Enc_{pk,\beta}(y), pk), sk) = s_{Q(x,y)}$.
- *(statistical privacy)* $\exists$ *Sim, s.t.* $\forall (sk, pk) \leftarrow Gen(\nu), \forall s_0, s_1 \in D_S, \forall x, y \in D_Q, \forall \alpha, \beta \in D_{R_{pk}}$: *the statistical distance* $Dist[Sim(s_{Q(x,y)}, pk), Rmap(s_0, s_1, Enc_{pk,\alpha}(x), Enc_{pk,\beta}(y), pk)]$ *is negligible in ν.*

Formally, the inputs e_0, e_1 to *Rmap* are *encodings* of elements in D_Q and need not have any privacy guarantees. Jumping ahead, we note that in our GT constructions 4.4, the inputs e_0, e_1 to *Rmap* are bitwise encryptions of the clients' bids. Note that Def. 1 allows this interpretation, since encrypting x bit-by-bit can be viewed as an encryption scheme itself.

Further, we do 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 Sect. 5.1.

[3] In practice, we are also interested in the correctness parameter λ. Security and correctness properties of Def. 1 are formulated with the notion of statistical closeness. Since ν and λ are polynomially related, we, for simplicity, use only the parameter ν.

[4] 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 ν at no extra cost.

4 The GT-CEM Construction and Protocols

Our construction builds on the ideas of the GT protocol of Blake and Kolesnikov [2]. Their protocol can be cast as a variant of GT-CEM, where one of the inputs is given in plaintext. We present their main idea and observe that a part of their protocol – the randomization procedure – requires S to know his input. In Sect. 4.2, we discuss the necessary properties of our new randomization, which works with encryptions only. In Sect. 4.3, we present such a randomization procedure and in Sect. 4.4 we give a GT-CEM construction. We give an alternative randomization procedure in Sect. 4.5, which can be incorporated into our GT-CEM.

4.1 The GT Protocol of [2]

We give a brief overview of the protocol and direct the reader to [2] for more details. Recall, there are two players, a receiver R with input x and a sender S with input y, s_0, s_1. S needs to send R the secret $s_{GT(x,y)}$.

The protocol operates on (homomorphically encrypted) bits of the inputs. The idea is to isolate the "important" position – the one where input bit strings first differ – by mapping it to a predetermined value and simultaneously randomizing values in all other positions. The rest is easily accomplished by applications of linear functions. In this work, we pay special attention to and improve the isolating randomization procedure.

In [2], the Receiver R sends bitwise additively homomorphic encryption of his input $x = \langle x_1, ..., x_n \rangle$ to the Sender S. For each bit position i, S computes (an encryption of) $f_i = x_i \oplus y_i$, i.e. whether $x_i = y_i$ (this requires the knowledge of y_i; knowing $Enc(y_i)$ is not sufficient). It is easy to see that $GT(x, y) = x_j$, where $j = min_{f_i \neq 0} i$. S's randomization procedure crucially relies on the fact that $f_j = 1$. Our randomization relies on the (encrypted) difference vector $d_i = x_i - y_i$, the "important element" of which may be (an encryption of) one of $\{-1, 1\}$.

4.2 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 μ, such that $\forall i \neq j, \mu_i \in_R D_{P_{pk}}$ and $\mu_j = d_j$. As in [2], we can obtain such μ_i for $i \geq j$ by computing for $i = 1..n$: $\mu_0 = 0; \mu_i = r_i \mu_{i-1} + d_i$, where $r_i \in_R D_{P_{pk}}$. Now vector μ 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 μ 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 Sect. 4.4) 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 $v_0, v_1 \in \mathbb{Z}\backslash\{0\}$ 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 Def. 1. Let $i \in \{0, 1\}$. We view v_i as an element of $D_{P_{pk}}$ in the natural manner (i.e. as $v_i \mod |D_{P_{pk}}|$). We note that even though this representation may vary with the choice of pk, v_i is a constant. Further, we require $v_i \neq 0 \mod |D_{P_{pk}}|$ and $v_0 \neq v_1 \mod |D_{P_{pk}}|$.

Definition 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* ν*:*
  $Rec(Rmap(s_0, s_1, Enc_{pk,\alpha}(v_i), pk), sk) = s_i$
  $Rec(Rmap(s_0, s_1, Enc_{pk,\alpha}(x), pk), sk) = \perp$,
  where the probability is taken over choices of x and random inputs of Rmap.
- *(statistical privacy at* v_0, v_1*)* $\exists Sim$*, s.t.* $\forall (sk, pk) \leftarrow Gen(\nu), \forall s_0, s_1 \in D_S$, $\forall i \in \{0, 1\}, \forall \alpha \in D_{R_{pk}}$ *: the statistical distance* $Dist[Sim(s_i, pk), Rmap(s_0, s_1, Enc_{pk,\alpha}(v_i), pk)]$ *is negligible in* ν*.*
- *(statistical privacy at* 0 *and at random elements of* $D_{P_{pk}}$*)* $\exists Sim_0$*, such that* $\forall (sk, pk) \leftarrow Gen(\nu), \forall s_0, s_1 \in D_S, \forall \alpha \in D_{R_{pk}}$ *: the statistical distances* $Dist[Sim_0(pk), Rmap(s_0, s_1, Enc_{pk,\alpha}(0), pk)]$ *and* $Dist[Sim_0(pk), Rmap(s_0, s_1, Enc_{pk,\alpha}(R), pk)]$ *are negligible in* ν*, where R is uniform on* $D_{P_{pk}}$*.*

4.3 A space-efficient $(-1, 1)$-RM

We present a construction for $(-1, 1)$-RM, based on the Paillier encryption scheme [15], which we use to construct GT-CEM. Let E be the Paillier scheme initialized as described in Def. 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 $((v_0, s_0), (v_1, s_1), (0, random))$ with a linear function (only linear functions can be applied under the homomorphic encryption). Our idea is for *Rmap* to produce not encryptions 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 1 *($(-1, 1)$-RM)*

Let λ and ν 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}.

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

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 $e^a g^b r'^N \mod N^2$.

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

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

Proof. (Sketch): 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 λ. Further, f applied by *Rmap* is a linear function, which is a permutation on $\mathbb{Z}_N$ with overwhelming probability in ν. (Indeed $f = ax + b$ is not a permutation only if $a = (S_1 - S_0)/2$ is not invertible.) 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 easy to see that *Rec* outputs $\perp$ on an encryption of a random element with overwhelming probability in λ.

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.

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

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

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

4.4 GT-CEM Based on Bitwise Paillier Encryption of Inputs

Let n be the length of the compared numbers. 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 $e_0 = \langle Enc_{pk}(x_1), ..., Enc_{pk}(x_n)\rangle, e_1 = \langle Enc_{pk}(y_1), ..., Enc_{pk}(y_n)\rangle$, where x_1 and y_1 are the most significant bits. The sender additionally has the secrets $s_0, s_1 \in D_S$ as inputs. Let $(Rmap_1, Rec_1)$ be a $(-1, 1)$-RM based on the Paillier encryption scheme (e.g. Constr. 1), instantiated with E_{pk}. Let $D_{M_{pk1}}, D_{S_1}$ be the domains of mappings and secrets of $(Rmap_1, Rec_1)$.

Construction 2 *(GT-CEM)*

Let λ and ν 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 $D_S = D_{S_1}$ and the domain of mappings $D_{M_{pk}} = D^n_{M_{pk1}}$.

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

1. *an encryption of the difference vector d, where $d_i = x_i - y_i$.*
2. *an encryption of vector γ, s.t. $\gamma_0 = 0$ and $\gamma_i = r_i\gamma_{i-1} + d_i$, where $r_i \in_R \mathbb{Z}_N$.*
3. *a randomized mapping vector μ, where $\mu_i = Rmap_1(s_0, s_1, Enc_{pk}(\gamma_i))$.*

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

Rec on input $(\mu'_1..\mu'_n, sk)$ proceeds as follows. For $i = 1..n$, let $z_i = Rec_1(\mu'_i, sk)$. If $z_i \neq \perp$, output z_i and halt. Otherwise, if $\forall i = 1..n, z_i = \perp$, output $\perp$.

Theorem 2. *Construction 2 is a GT-CEM.*

Proof. (Sketch) We will first show that Construction 2 satisfies the correctness requirement. It is easy to see that the homomorphic properties of the encryption scheme allow *Rmap* and *Rec* to perform all necessary operations.

Let j be the position where x and y first differ; thus d_j determines $\mathrm{GT}(x, y)$. With overwhelming probability, γ is a vector with the following structure: it starts with zero or more zeros, then, in position j, a one or a minus one, then a sequence of random elements in $\mathbb{Z}_N$. It is not hard to see that, by the correctness and privacy properties of $(-1, 1)$-RM, *Rec*, using Rec_1, will recover $s_{GT(x,y)}$.

We now show that the privacy condition holds as well. We construct simulator $\mathrm{Sim}_{GT}(s, pk)$, where pk is the public key established in the setup phase and

$s = s_{Q(x,y)}$. $\mathrm{Sim}_{GT}(s, pk)$ has to generate a distribution statistically close to the output of *Rmap*. $\mathrm{Sim}_{GT}(pk, s)$ proceeds as follows, using the simulators Sim_0 and Sim, required by $(-1, 1)$-RM. It runs $\mathrm{Sim}_0(pk)$ $n-1$ times and $\mathrm{Sim}(s, pk)$ once, obtaining a vector z' of n simulated mappings. $\mathrm{Sim}_R(s, pk)$ outputs a random permutation $\pi'(z')$. It is easy to see that $\mathrm{Sim}_{GT}(pk, s)$ statistically simulates the output of *Rmap*, due to properties of Sim_0 and Sim. □

4.5 A General (v_0, v_1)-RM Construction

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

Rmap proceeds as follows. First, as in Construction 1, 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 2, 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 1, 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 Sect. 6 for detailed comparisons.

4.6 Resource Analysis

We evaluate the message and modular multiplication efficiency of Construction 2, used with $(-1, 1)$-RM of Sect. 4.3 (which we refer to as CEM1) and of Sect. 4.5 (CEM2). 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, $2n \log N$ multiplications are required. Step 1 of Construction 2 requires n multiplications, and step 2 requires $(\log N + 1)n$ multiplications. Step 3 of CEM1 requires $(3 \log N + 2)n$ multiplications ($2 \log N + 1$ multiplications for application of the linear function f, and $\log N$ to re-randomize the encryption). Similarly, step 3 of CEM2 requires $(6 \log N + 4)n$ multiplications.

Rec of CEM1 (resp. CEM2) costs $2n \log N$ (resp. $4n \log N$) multiplications (We expect to perform half of them before *Rec* recovers the secret and halts).

In total, CEM1 (resp. CEM2) requires no more than $\approx 8n \log N$ (resp. $\approx 13n \log N$) modular multiplications. Of those, $4n \log N$ (resp. $7n \log N$) are performed by *Rmap*, and $4n \log N$ (resp. $6n \log N$) 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 [5, 6], since they are performed $\mod N^2$, while the Goldwasser-Micali (GM) multiplications (used in [5, 6]) are $\mod N$.

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 ([5, 6]). 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 Sect. 6.

4.7 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 [11]. The idea is to assign two specially constructed secrets to each input wire of the (poly-size) 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 Protocol Constructions from GT-CEM

As mentioned in the discussion of CEM in Sect. 3, natural protocol constructions immediately arise from CEM in the semi-honest model. We demonstrate this on a special case of PSPP of [5], where the server S runs the auction with two bidders C_0, C_1. (Our solution can naturally accomodate more bidders, using, e.g., technique of Sect. 5.2 of [5].) As discussed in Sect. 2 and [5], 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 $Enc_{pk_i}(x_i), Enc_{pk_{1-i}}(x_i)$) 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 $m_i = Rmap(s_0, s_1, Enc_{pk_i}(x_i), Enc_{pk_i}(x_{1-i}), pk_i)$

to each C_i, who then applies $Rec(sk_i, m_i)$ 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 [5].)

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).

5.1 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 [12]. 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 $Dec(e) \in \{0, 1\}$). 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 $Dec(e) \in \{0, 1\}$, which can be easily constructed.

5.2 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 $Enc_{pk}(x)$ of his lowest acceptable price (i.e. reserve) to S, who later interacts with buyers as follows. On an encrypted offer $Enc_{pk}(y)$, S replies with $Rmap(s_0, s_1, Enc_{pk}(y), Enc_{pk}(x), pk)$, 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 sect. 4.7, allows us to obtain very 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.

6 Comparison with Previous Work

We continue the resource analysis of Sect. 4.6. Note that the protocols of [5, 6, 12] 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 [5, 6, 12] and our constructions CEM1 and CEM2 (i.e. Construction 4.4 instantiated with $(-1, 1)$-RM of Sect. 4.3 and 4.5 respectively).

Here c-bit secrets are transferred based on comparison of n-bit numbers. λ and ν are the correctness and security parameters, and $N > 2^\nu$ is the modulus of the employed encryption scheme (GM for [5, 6] and Paillier for [12] 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*.

Solutions of [5, 6] 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 $\nu \approx 1000, \lambda \approx 40..80$ 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 [12] can be modified to obtain GT-CEM similar in cost to CEM2. Solution of [2] (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 [6, 5], is not less than $\log N$ modular multplications. For simplicity, we assume that it is $\log N$.)

Protocol	Comparable Modular Multiplications			Communication	Comment
	client	server	total		
of [6]	$4nc\lambda \log N$	$24nc\lambda$	$32nc\lambda + 4nc\lambda \log N$	$4nc\lambda \log N$	
of [5]	$8n^2c \log N$	$12n^2c$	$12n^2c + 8n^2c \log N$	$8n^2c \log N$	
CEM1	$16n \log N$	$16n \log N$	$32n \log N$	$2n \log N$	$c < \nu/2 - \lambda$
CEM2	$24n \log N$	$28n \log N$	$52n \log N$	$4n \log N$	$c < \nu - \lambda$

Acknowledgements. The authors are very grateful to Charles Rackoff for many technical comments and suggestions on this paper. We also thank Sven Laur and Helger Lipmaa for discussions of CDS and their related work [12], Marc Fischlin for clarifications of the costs associated with his scheme [6], and the anonymous reviewers of FC 2006 for helpful comments and suggestions.

References

1. William Aiello, Yuval Ishai, and Omer Reingold. Priced oblivious transfer: How to sell digital goods. In *Proc. EUROCRYPT 2001*, pages 119–135, 2001.
2. Ian F. Blake and Vladimir Kolesnikov. Strong conditional oblivious transfer and computing on intervals. In *Advances in Cryptology - ASIACRYPT 2004*, volume 3329 of *Lecture Notes in Computer Science*, pages 515–529. Springer, 2004.
3. Christian Cachin. Efficient private bidding and auctions with an oblivious third party. In *Proceedings of the 6th ACM Conference on Computer and Communications Security*, pages 120–127. ACM Press, 1999.
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. Giovanni Di Crescenzo. Private selective payment protocols. In *Financial Cryptography*, pages 72–89, 2000.
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. Yael Gertner, Yuval Ishai, Eyal Kushilevitz, and Tal Malkin. Protecting data privacy in private information retrieval schemes. In *STOC '98: Proceedings of the thirtieth annual ACM symposium on Theory of computing*, pages 151–160, New York, NY, USA, 1998. ACM Press.
9. Bart Goethals, Sven Laur, Helger Lipmaa, and Taneli Mielikäinen. On Secure Scalar Product Computation for Privacy-Preserving Data Mining. In Choonsik Park and Seongtaek Chee, editors, *The 7th Annual International Conference in Information Security and Cryptology (ICISC 2004)*, volume 3506, pages 104–120, December 2–3, 2004.
10. 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.
11. Vladimir Kolesnikov. Gate evaluation secret sharing and secure one-round two-party computation. In *Advances in Cryptology - ASIACRYPT 2005*, volume 3788 of *Lecture Notes in Computer Science*, pages 136–155. Springer, 2005.
12. Sven Laur and Helger Lipmaa. Additive conditional disclosure of secrets and applications. Cryptology ePrint Archive, Report 2005/378, 2005. `http://eprint.iacr.org/`.
13. 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.
14. Moni Naor, Benny Pinkas, and Reuben Sumner. Privacy preserving auctions and mechanism design. In *1st ACM Conf. on Electronic Commerce*, pages 129–139, 1999.
15. 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.
16. M. Rabin. How to exchange secrets by oblivious transfer. Technical Report TR-81, Harvard Aiken Computation Laboratory, 1981.