How to Combine Homomorphic Encryption and Garbled Circuits

Improved Circuits and Computing the Minimum Distance Efficiently

Vladimir Kolesnikov[1], Ahmad-Reza Sadeghi[2], and Thomas Schneider[2]

[1] Bell Laboratories, 600 Mountain Ave. Murray Hill, NJ 07974, USA
`kolesnikov@research.bell-labs.com`
[2] Horst Görtz Institute for IT-Security, Ruhr-University Bochum, Germany
{`ahmad.sadeghi`,`thomas.schneider`}`@trust.rub.de`*

Abstract. We show how two existing paradigms for two-party secure function evaluation (SFE) in the semi-honest model can be combined securely and efficiently – those based on additively homomorphic encryption (HE) with those based on garbled circuits (GC) and vice versa. Additionally, we propose new GC constructions for addition, subtraction, multiplication, and comparison functions. Our circuits are approximately two times smaller (in terms of garbled tables) than previous constructions. This implies corresponding computation and communication improvements in SFE of functions using the above building blocks (including the protocols for combining HE and GC).
Based on these results, we present efficient constant-round protocols for secure integer comparison, and the related problems of minimum selection and minimum distance, which are crucial building blocks of many cryptographic schemes such as privacy preserving biometric authentication (e.g., face recognition, fingerprint matching, etc).

Keywords: Secure Computation, Homomorphic Encryption, Garbled Circuit, Millionaires Problem, Minimum Selection, Minimum Distance

1 Introduction

We are motivated by secure function evaluation (SFE) of integer comparison, and related problems such as biometric authentication. We propose new, more efficient SFE protocols for these functions. More specifically, we propose improved constructions for addition, subtraction, multiplication, and comparison functions, and demonstrate their advantages on the example of our motivating applications.

Comparison is a widely used basic primitive. In particular, it plays an especially important role in financial transactions, biometric authentication, database mining applications, etc.

* Supported by EU FP6 project SPEED, EU FP7 project CACE and ECRYPT II.

Biometric authentication. Widespread adoption of biometric authentication (e.g., fingerprint or face recognition) is causing strong concerns of privacy violations. Indeed, improper use of biometric information has far more implications than "simple" collection of personal information. Adoption of privacy-preserving biometric authentication is highly desired and will benefit the users and the administrators of the systems alike. Because biometric images are never scanned perfectly, the identity of the user is determined by proximity of the scanned and stored biometrics. It is natural, therefore, that threshold comparisons are frequently employed in such identification systems. In particular, privacy-preserving face-recognition based on the Eigenfaces algorithm [TP91b,TP91a] relies on securely computing the closest point, which requires secure comparison. Further, in some multi-user systems, it may be desired to simply find the closest match in the database. In such systems, secure comparison would be also extensively used.

Garbled Circuit (GC) and state of the art for secure comparison and related algorithms. Starting with the original paper of Yao [Yao82], secure comparison, also referred to as the "two Millionaires problem", has attracted much attention [Yao86,Fis01,MNPS04,Kol05]. A variety of techniques are employed in the many proposed solutions – homomorphic encryption, evaluation of branching programs, Garbled Circuit (GC).

Today, in the standard computational setting, the most efficient protocol is the simple evaluation of the generic GC. Indeed, the size of the comparison circuit is quite small (linear in the size of the inputs), and its secure evaluation is rather efficient (linear number of Oblivious Transfers (OT) and evaluations of a cryptographic hash function, such as SHA-256).

Most popular alternative solutions are based on homomorphic encryptions. For comparison, they offer a similar complexity compared to GC, as they still must perform a linear (in the input) number of public key operations by both players. However, GC offers more flexible and cheap programming possibilities, due to its low cost of manipulation of boolean values. In contrast, homomorphic encryptions are not suitable, e.g., for branching based on the encrypted value which can be achieved only with much more expensive techniques than GC).

In sum, GC approach is a clear choice for integer comparison, its extensions, such as auctions, simple integer manipulations (addition and even multiplications which have circuits of quadratic size) and a variety of other problems that have small circuit representation.

Homomorphic Encryption (HE). Nevertheless, in some cases it is beneficial to decrease the communication complexity at the cost of an increased round complexity by performing additions or multiplications using additively homomorphic encryption (and an additional round for multiplication). This has been used in many protocols, such as [OPB07,PCB$^+$08,JP09] - just to name a view.

To achieve highly efficient protocols that benefit from the flexibility of GC based protocols and the reduced communication complexity of HE based protocols for additions and multiplications, both approaches for secure two-party computation can be combined.

Our contributions. As justified above, our work focusses on GC as well as its combination with HE.

We advance the state of the art of SFE for addition, subtraction, multiplication and comparison functions, by constructing their more efficient GC representations. We work in the semi-honest model.More specifically, our optimizations take advantage of the recently proposed method of GC construction [KS08], where XOR gates are evaluated essentially for free (one XOR operation on keys, and no garbled table entries to generate or transfer). We show how to compute comparison and other basic functions with circuits consisting mostly of XOR gates. This results in reduction of the size of GC (i.e., the size of garbled tables) by approximately half (see Table 1 for detailed comparison). We note that the method of [KS08] (and thus our work) requires the use of a weak form of Random Oracle, namely of correlation-robust functions [IKNP03].

After that, we show how protocols based on HE and GC can be combined by providing efficient conversion protocols between homomorphically encrypted inputs/outputs and inputs/outputs into a GC.

As an additional contribution, we then follow through, and discuss in detail GC-based constructions for the Millionaires problem, and minimum Hamming- or Euclidian distance. In addition to establishing a new performance baseline for these problems, we aim to promote GC as a very efficient solution, and prevent its frequent unfair dismissal as an "impractical generic approach".

Related work. SFE (and in particular GC), and secure comparison has received much attention in the literature, all of which we cannot possibly include here. In this section we summarize relevant work to give the reader a perspective on our results. We discuss additional related work (on which we improve) in the individual sections of the paper.

Circuit-Based Secure Function Evaluation. GC technique of SFE was introduced by Yao [Yao86], with a formal proof of security (in the semi-honest model) given in [LP04]. Extensions of Yao's garbled circuit protocol to security against covert players were given in [AHL05,GMS08], and against malicious players in [JS07,LP07,NO09]. Our constructions rely on the recent work of Kolesnikov and Schneider [KS08]. In [KS08], a GC technique is proposed, where XOR gates can be evaluated "for free", i.e., with negligible computation and communication costs. The authors of [KS08] present improved circuit constructions for multiplexer, addition and (in-)equality testing. Our main contribution – the building block constructions further improve their proposals (e.g., the addition is improved by a factor of two, and we propose the more general functionality of comparison).

Secure Two-Party Comparison. The first secure two-party comparison protocol was proposed in [Yao82], and today GC is the most efficient solution to this problem. Homomorphic encryption is another popular tool for comparison of

ℓ-bit strings. The protocol of Fischlin [Fis01] uses the Goldwasser-Micali XOR-homomorphic encryption scheme [GM84] and has communication complexity $\ell T(\kappa+1)$, where κ is a statistical correctness parameter (e.g., $\kappa = 40$). The comparison protocol of [BK04] uses bitwise Paillier encryption and has communication complexity $4\ell T$, where T is an asymmetric security parameter (e.g., size of an RSA modulus). This protocol was improved in [DGK07,DGK08b,DGK08a] to communication complexity $2\ell T$ by using a new homomorphic encryption scheme with smaller ciphertext size. These two-party protocols were extended to comparisons in the multi-party setting with logarithmic and linear round complexity in [GSV07].

Minimum Selection. A two-party protocol for finding k-Nearest Neighbors was given in [SKK06], and improved from quadratic to linear communication complexity in [QA08]. Our protocol for finding the nearest neighbor is a more efficient protocol for the special case $k = 1$. A simple protocol to select the minimum of homomorphically encrypted values based on the multiplicative hiding assumption was given in [Ker08] in the context of privacy-preserving benchmarking. However, multiplicative blinding reveals some information about the magnitude of the blinded value. Our minimum selection protocol can be used as a provably secure replacement of this protocol. Finally, we note that in our minimum Hamming distance protocol we use several steps of the Hamming distance protocol of [JP09].

2 Preliminaries

In this section, we summarize our conventions and setting in §2.1 and cryptographic tools used in our constructions: oblivious transfer (OT) in §2.3, garbled circuits (GC) with free XOR in §2.4, and additively homomorphic encryption (HE) in §2.2.

Reader familiar with the prerequisites may safely skip to §3.

2.1 Parameters, Notation and Model

We denote symmetric security parameter by t and the asymmetric security parameter, i.e., bitlength of RSA moduli, by T. Recommended parameters for short-term security (until 2010) are for example $t = 80$ and $T = 1024$ [GQ09]. The bitlength of a garbled value is $t' := t+1$ (cf. §2.4 for details). The statistical correctness parameter is denoted with κ (the probability of a protocol failure is bounded by $2^{-\kappa}$) and the statistical security parameter with σ. In practice, one can choose $\kappa = \sigma = 80$. The bitlength of protocol inputs is denoted with ℓ and the number of inputs with n. We write $\mathbf{x}^{\ell}$ to denote ℓ-bit value x.

We work in the semi-honest model. We note that the method of [KS08] (and thus our work) requires the use of a weak form of Random Oracle, namely of correlation-robust functions [IKNP03].

2.2 Homomorphic Encryption (HE)

Some of our constructions make black-box usage of a semantically secure homomorphic encryption scheme with plaintext space $(P, +, 0)$, ciphertext space $(C, *, 1)$, and probabilistic polynomial-time algorithms $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$.

An *additively homomorphic encryption* scheme allows addition under encryption as it satisfies $\forall x, y \in P : \mathsf{Dec}(\mathsf{Enc}(x) * \mathsf{Enc}(y)) = x + y$. It can be instantiated with a variety of cryptosystems including [Pai99,DJ01], or the cryptosystem of [DGK07,DGK08b,DGK08a] which is restricted to small plaintext space P – just to name a few.

For the sake of completeness we mention, that the cryptosystem of [BGN05] allows for an arbitrary number of additions and one multiplication and *fully homomorphic encryption* schemes allow to evaluate an arbitrary number of additions and multiplications on ciphertexts. Possible candidates are the cryptosystem of [AS08] (size of ciphertexts grows exponentially in the number of multiplications) or the recently proposed scheme without such a restriction [Gen09]. However, the size of ciphertexts in these schemes is substantially larger than that of the purely additively homomorphic schemes.

2.3 Oblivious Transfer (OT)

Parallel 1-out-of-2 Oblivious Transfer of m ℓ-bit strings, denoted as OT_ℓ^m, is a two-party protocol run between a chooser $\mathcal{C}$ and a sender $\mathcal{S}$. For $i = 1, \ldots, m$, $\mathcal{S}$ inputs a pair of ℓ-bit strings $s_i^0, s_i^1 \in \{0,1\}^\ell$ and $\mathcal{C}$ inputs m choice bits $b_i \in \{0,1\}$. At the end of the protocol, $\mathcal{C}$ learns the chosen strings $s_i^{b_i}$, but nothing about the unchosen strings $s_i^{1-b_i}$ whereas $\mathcal{S}$ learns nothing about the choices b_i.

Efficient OT protocols. We use OT_ℓ^m as a black-box primitive which can be instantiated efficiently with different protocols [NP01,AIR01,Lip03,IKNP03]. For example the protocol of [AIR01] implemented over a suitably chosen elliptic curve has communication complexity $m(6(2t+1)) + (2t+1) \sim 12mt$ bits and is secure against malicious $\mathcal{C}$ and semi-honest $\mathcal{S}$ in the standard model as described in §A. Similarly, the protocol of [NP01] implemented over a suitably chosen elliptic curve has communication complexity $m(2(2t+1)+2\ell)$ bits and is secure against malicious $\mathcal{C}$ and semi-honest $\mathcal{S}$ in the random oracle model. Both protocols require $\mathcal{O}(m)$ scalar point multiplications.

Extending OT efficiently. The extensions of [IKNP03] can be used to efficiently reduce the number of computationally expensive public-key operations of OT_ℓ^m to be independent of m. Their transformation for semi-honest receiver reduces OT_ℓ^m to OT_t^t and a small additional overhead: one additional message, $2m(\ell+t)$ bits additional communication, and $\mathcal{O}(m)$ invocations of a correlation robust hash function ($2m$ for $\mathcal{S}$ and m for $\mathcal{C}$) which is substantially cheaper than $\mathcal{O}(m)$ asymmetric operations. Also a slightly less efficient extension for malicious receiver is given in their paper.

2.4 Garbled Circuits (GC)

The most efficient method for secure evaluation of a boolean circuit C for computationally bounded players is Yao's garbled circuit approach [Yao86,LP04]. We briefly summarize the main ideas of this protocol in the following. The circuit *constructor* (server $\mathcal{S}$) creates a *garbled circuit* $\widetilde{C}$ with algorithm $\mathsf{CreateGC}$: for each wire W_i of the circuit, he randomly chooses two garbled values $\widetilde{w}_i^0, \widetilde{w}_i^1$, where $\widetilde{w}_i^j$ is the *garbled value* of W_i's value j. (Note: $\widetilde{w}_i^j$ does not reveal j.) Further, for each gate G_i, $\mathcal{S}$ creates a *garbled table* $\widetilde{T}_i$ with the following property: given a set of garbled values of G_i's inputs, $\widetilde{T}_i$ allows to recover the garbled value of the corresponding G_i's output, but nothing else. $\mathcal{S}$ sends these garbled tables, called *garbled circuit* $\widetilde{C}$ to the *evaluator* (client $\mathcal{C}$). Additionally, $\mathcal{C}$ obliviously obtains the *garbled inputs* $\widetilde{w}_i$ corresponding to inputs of both parties (details on how this can be done later in §4). Now, $\mathcal{C}$ can evaluate the garbled circuit $\widetilde{C}$ on the garbled inputs with algorithm EvalGC to obtain the *garbled outputs* simply by evaluating the garbled circuit gate by gate, using the garbled tables $\widetilde{T}_i$. Finally, $\mathcal{C}$ translates the garbled outputs into output values given for the respective players (details below in §4). Correctness of GC follows from the method of how garbled tables $\widetilde{T}_i$ are constructed.

Improved Garbled Circuit with free XOR [KS08]. An efficient method for creating garbled circuits which allows "free" evaluation of XOR gates was presented in [KS08]. More specifically, a garbled XOR gate has no garbled table (*no communication*) and its evaluation consists of XOR-ing its garbled input values (*negligible computation*). The other gates, called *non-XOR gates*, are evaluated as in Yao's GC construction [Yao86] with a *point-and-permute technique* (as used in [MNPS04]) to speed up the implementation of the GC protocol: the garbled values $\widetilde{w}_i = \langle k_i, \pi_i \rangle \in \{0,1\}^{t'}$ consist of a symmetric key $k_i \in \{0,1\}^t$ and a random permutation bit $\pi_i \in \{0,1\}$ and hence have length $t' = t+1$ bits. The permutation bit π_i is used to select the right table entry for decryption with the t-bit key k_i (recall, t is the symmetric security parameter). The encryption is done with the symmetric encryption function $\mathsf{Enc}^s_{k_1,\ldots,k_d}(m) = m \oplus H(k_1||\ldots||k_d||s)$, where d is the number of inputs of the gate, s is a unique identifier used once, and H is a suitably chosen cryptographic hash function. Hence, creation of the garbled table of a non-XOR d-input gate requires 2^d invocations of H and its evaluation needs one invocation, while XOR gates are "for free".

The main observation of [KS08] is, that the constructor $\mathcal{S}$ chooses a global key difference $\Delta \in_R \{0,1\}^t$ which remains unknown to evaluator $\mathcal{C}$ and relates the garbled values as $k_i^0 = k_i^1 \oplus \Delta$. (This technique was subsequently extended in the LEGO paper [NO09] which allows to compose garbled circuits dynamically with security against malicious circuit constructor). Clearly, the usage of such garbled values allows for *free evaluation of XOR gates* with input wires W_1, W_2 and output wire W_3 by computing $\widetilde{w}_3 = \widetilde{w}_1 \oplus \widetilde{w}_2$ (no communication and negligible computation). However, using related garbled values requires that the hash function H used to create the garbled tables of non-XOR gates has to be modeled to be correlation robust (as defined in [IKNP03]) which is stronger than modeling H as a key-derivation function (standard model) but weaker than modeling H as a random-oracle (ROM). In practice, H can be chosen from the SHA-2 family.

3 Improved Building Blocks for GC

In this section we present improved circuit constructions for several frequently used primitives, such as integer addition, subtraction and multiplication (§3.2), comparison (§3.3), and selection of the minimum value and index (§3.4). As summarized in Table 1, our improved circuit constructions are smaller than previous solutions by 33% to 50% (depending on the functionality) when used with the GC of [KS08]. This reduction in size immediately translates into a corresponding improvement in communication and computation complexity of any GC protocol built from these blocks. The efficiency improvements are achieved by modifying the underlying circuits, i.e., by carefully replacing larger (more costly) non-XOR gates (e.g., a 3-input gate) with smaller non-XOR gates (e.g., a 2-input gate) and (free) XOR gates.

Table 1. Size of efficient circuit constructions for operations on ℓ-bit values and computing the minimum value and index of n ℓ-bit values (in table entries).

Circuit	Standard GC	[KS08]	This Work (Improvement)
Multiplexer (§3.1)	8ℓ	4ℓ	
Addition/Subtraction (§3.2)	16ℓ	8ℓ	4ℓ (50%)
Multiplication (§3.2)	$20\ell^2 - 16\ell$	$12\ell^2 - 8\ell$	$8\ell^2 - 4\ell$ (33%)
Equality Test (§3.3)	8ℓ	4ℓ	
Comparison (§3.3)	8ℓ		4ℓ (50%)
Minimum Value + Index (§3.4)	$\approx 15\ell n$ [NPS99]		$8\ell(n-1) + 4(n+1)$ (47%)

3.1 Improved Multiplexer (MUX) from [KS08]

In our constructions we use ℓ-bit multiplexer circuits MUX which select their output $\mathbf{z}^\ell$ to be one of the ℓ-bit inputs $\mathbf{x}^\ell$ and $\mathbf{y}^\ell$, depending on the selection bit c. For MUX, we use the improved construction of [KS08] summarized in the following:

An ℓ-bit multiplexer circuit MUX selects its output $\mathbf{z}^\ell$ to be its left ℓ-bit input $\mathbf{x}^\ell$ if the input selection bit c is 0, respectively its right ℓ-bit input $\mathbf{y}^\ell$ otherwise. As shown in Fig. 1, the block can be composed from ℓ parallel Y blocks that are 1-bit multiplexers. The Y gates have three inputs x_i, y_i, c and one output z_i. They could be instantiated with a 3-to-1 gate of size $2^3 = 8$ table entries. According to [KS08], Y blocks can be instantiated more efficiently, as shown in Fig. 2: this instantiation needs only a 2-input AND gate of size $2^2 = 4$ table entries and two free XOR gates resulting in an overall improvement by a factor of two (as 4 instead of 8 table entries need to be garbled and transferred in the GC protocol). Overall, the efficient ℓ-bit multiplexer construction has size $\left|\mathsf{MUX}^\ell\right| = \ell \cdot |\mathsf{Y}| = 4\ell$ table entries.

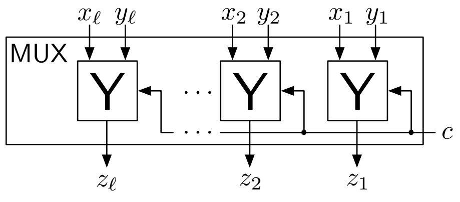

Fig. 1. Multiplexer Circuit

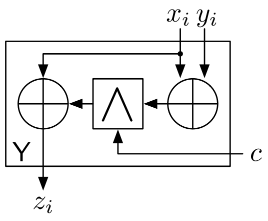

Fig. 2. Improved Y Block [KS08]

In the following we describe our new improved circuit constructions.

3.2 Improved Integer Addition, Subtraction and Multiplication

Addition circuits (ADD) to add two unsigned integer values $\mathbf{x}^\ell, \mathbf{y}^\ell$ can be efficiently composed from a chain of 1-bit adders ($+$), often called full-adders, as shown in Fig. 3. (The first 1-bit adder has constant input $c_1 = 0$ and can be replaced by a smaller half-adder). Each 1-bit adder has as inputs the carry-in bit c_i from the previous 1-bit adder and the two input bits x_i, y_i. The outputs are the carry-out bit $c_{i+1} = (x_i \wedge y_i) \vee (x_i \wedge c_i) \vee (y_i \wedge c_i)$ and the sum bit $s_i = x_i \oplus y_i \oplus c_i$ (the latter can be computed "for free" [KS08]). Our improved construction of a 1-bit adder shown in Fig. 4 computes the carry-out bit as $c_{i+1} = c_i \oplus ((x_i \oplus c_i) \wedge (y_i \oplus c_i))$. Overall, our construction for a 1-bit adder consists of four free XOR gates and a single 2-input AND gate which has size $2^2 = 4$ table entries. The overall size of our improved addition circuit is $\left|\mathsf{ADD}^\ell\right| = \ell \cdot |+| = 4\ell$ table entries.

Subtraction in two's complement representation is defined as $\mathbf{x}^\ell - \mathbf{y}^\ell = \mathbf{x}^\ell + \bar{\mathbf{y}}^\ell + 1$. Hence, a subtraction circuit (SUB) can be constructed analogously to the addition circuit from 1-bit subtractors ($-$) as shown in Fig. 5. Each 1-bit subtractor computes the carry-out bit $c_{i+1} = (x_i \wedge \bar{y}_i) \vee (x_i \wedge c_i) \vee (\bar{y}_i \wedge c_i)$ and the difference bit $d_i = x_i \oplus \bar{y}_i \oplus c_i$. We instantiate the 1-bit subtractor efficiently as shown in Fig. 6 to compute $c_{i+1} = x_i \oplus ((x_i \oplus c_i) \wedge (y_i \oplus c_i))$ with the same size as the 1-bit adder.

Multiplication circuits (MUL) to multiply two ℓ-bit integers $\mathbf{x}^\ell, \mathbf{y}^\ell$ can be constructed according to the "school method" for multiplication, i.e., adding up the bitwise multiplications of y_i and $\mathbf{x}^\ell$ shifted corresponding to the position: $\boldsymbol{x}^\ell \cdot \boldsymbol{y}^\ell = \sum_{i=1}^{\ell} 2^{i-1}(y_i \cdot \boldsymbol{x}^\ell)$. This circuit is composed from ℓ^2 of 1-bit multipliers (2-input AND gates) and $(\ell-1)$ of ℓ-bit adders. Using our improved implementation

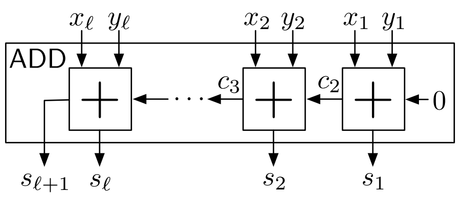

Fig. 3. Addition Circuit (ADD)

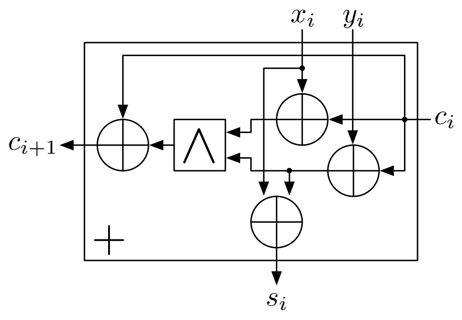

Fig. 4. Improved 1-bit Adder (+)

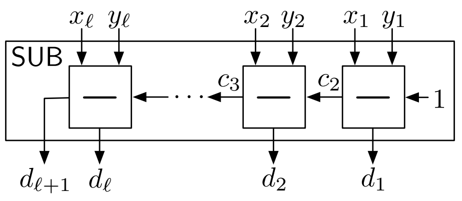

Fig. 5. Subtraction Circuit (SUB)

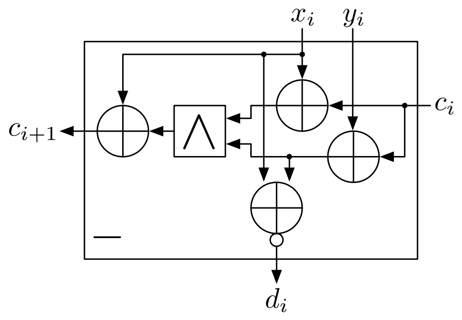

Fig. 6. Improved 1-bit Subtractor (−)

for adders, the size of the multiplication circuit is improved to $4\ell^2 + 4\ell(\ell - 1) = 8\ell^2 - 4\ell$ table entries.

3.3 Improved Integer Comparison

We present improved circuit constructions for comparison of two ℓ-bit integers $\mathbf{x}^\ell$ and $\mathbf{y}^\ell$, i.e.,

$$z = \left[\mathbf{x}^\ell > \mathbf{y}^\ell\right] := \begin{cases} 1 & \text{if } \mathbf{x}^\ell > \mathbf{y}^\ell, \\ 0 & \text{else.} \end{cases}$$

Note that this functionality is more general than checking equality of ℓ-bit integers $\mathbf{x}^\ell$ and $\mathbf{y}^\ell$, i.e., $z = \left[\mathbf{x}^\ell = \mathbf{y}^\ell\right]$, for which an improved construction was given in [KS08].

As shown in Fig. 7, a comparison circuit (CMP) can be composed from ℓ sequential 1-bit comparators (>). (The first 1-bit comparator has constant input $c_1 = 0$ and can be replaced by a smaller gate). Our improved instantiation for a 1-bit comparator shown in Fig. 8 uses one 2-input AND gate with 4 table entries and three free XOR gates. Note, this improved bit comparator is exactly the improved bit subtractor shown in Fig. 6 restricted to the carry output: $\left[\mathbf{x}^\ell > \mathbf{y}^\ell\right] \Leftrightarrow \left[\mathbf{x}^\ell - \mathbf{y}^\ell - 1 \geq 0\right]$ which coincides with an underflow in the corresponding subtraction denoted by subtractor's most significant output bit $d_{\ell+1}$. The size of our improved comparison circuit is $\left|\mathsf{CMP}^{\ell}\right| = \ell \cdot |{>}| = 4\ell$ table entries.

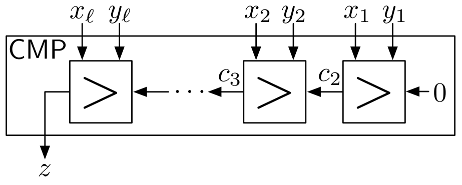

Fig. 7. Comparison Circuit (CMP)

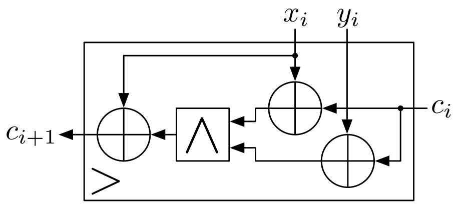

Fig. 8. Improved 1-bit Comparator ($>$)

Improved comparison circuits for $\left[\mathbf{x}^\ell < \mathbf{y}^\ell\right]$, $\left[\mathbf{x}^\ell \geq \mathbf{y}^\ell\right]$, or $\left[\mathbf{x}^\ell \leq \mathbf{y}^\ell\right]$ can be obtained from the improved circuit for $\left[\mathbf{x}^\ell > \mathbf{y}^\ell\right]$ by interchanging $\mathbf{x}^\ell$ with $\mathbf{y}^\ell$ and/or setting the initial carry to $c_1 = 1$.

3.4 Improved Minimum Value and Minimum Index

Finally, we show how the improved blocks presented above can be combined to obtain an improved minimum circuit (MIN) which selects the minimum value $\mathbf{m}^\ell$ and minimum index i of a list of n ℓ-bit values $\mathbf{x}_0^\ell, \ldots, \mathbf{x}_{n-1}^\ell$, i.e., $\forall j \in \{0, \ldots, n-1\} : (\mathbf{m}^\ell < \mathbf{x}_\mathbf{j}^\ell) \vee (\mathbf{m}^\ell = \mathbf{x}_\mathbf{j}^\ell \wedge i \leq j)$. E.g., for the list $3, 2, 5, 2$ the outputs would be $\mathbf{m}^\ell = 2$ and $i = 1$ as the leftmost minimum value of 2 is at position 1. W.l.o.g. we assume that n is a power of two, so the minimum index can be represented with $\log n$ bits.

Performance improvement of MIN mainly comes from the improved building blocks for integer comparison. We shave off an additive factor by carefully arranging a tournament-style circuit so that some of the index wires can be reused and eliminated. That is, at depth d of the resulting tree we keep track of the ℓ-bit minimum value $\mathbf{m}^\ell$ of the sub-tree containing 2^d values but store and propagate only the d least significant bits $\mathbf{i}_d^d$ of the minimum index.

More specifically, the minimum value and minimum index are selected pairwise in a tournament-like way using a tree of minimum blocks (min) as shown in Fig. 9. As shown in Fig. 10, each minimum block at depth d gets as inputs the minimum ℓ-bit values $\mathbf{m}_{d,L}^\ell$ and $\mathbf{m}_{d,R}^\ell$ of its left and right subtrees T_L, T_R and the d least significant bits of their minimum indices $\mathbf{i}_{d,L}^d$ and $\mathbf{i}_{d,R}^d$, and outputs the minimum ℓ-bit value $\mathbf{m}_{d+1}^\ell$ and $(d+1)$-bit minimum index $\mathbf{i}_{d+1}^{d+1}$ of the tree. First, the two minimum values are compared with a comparison circuit (cf. §3.3). If the minimum value of T_L is bigger than that of T_R (in this case, the comparison circuit outputs value 1), $\mathbf{m}_{d+1}^\ell$ is chosen to be the value of T_R with an ℓ-bit multiplexer block (cf. §3.1). In this case, the minimum index $\mathbf{i}_{d+1}^{d+1}$ is set to 1 concatenated with the minimum index of T_R using another d-bit multiplexer. Alternatively, if the comparison yields 0, the minimum value of T_L and the value 0 concatenated with the minimum index of T_L are output. Overall, the size of the efficient minimum circuit is $\left|\mathsf{MIN}^{\ell,n}\right| = (n-1)\cdot(\left|\mathsf{CMP}^{\ell}\right| + \left|\mathsf{MUX}^{\ell}\right|) + \sum_{j=1}^{\log n} \frac{n}{2^j}\left|\mathsf{MUX}^{j-1}\right| = 8\ell(n-1) + 4n\sum_{j=1}^{\log n}\frac{j-1}{2^j} < 8\ell(n-1) + 4n(1+\frac{1}{n}) = 8\ell(n-1) + 4(n+1)$.

Our method of putting the minimum blocks together in a tree (cf. Fig. 9) is non-trivial: If the minimum blocks would have been arranged sequentially (according to the standard selection algorithm to find the minimum), the size of the circuit would have been $(n-1)\cdot(\left|\mathsf{CMP}^{\ell}\right| + \left|\mathsf{MUX}^{\ell}\right| + \left|\mathsf{MUX}^{\log n}\right|) = 8\ell(n-1) + 4(n-1)\log n$ table entries which is less efficient than the tree.

In previous work [NPS99], a circuit for computing first-price auctions (which is functionally equivalent to computing the maximum value and index) with a size of approximately $15\ell n$ table entries is mentioned over which our explicit construction improves by a factor of approximately $\frac{15}{8}$.

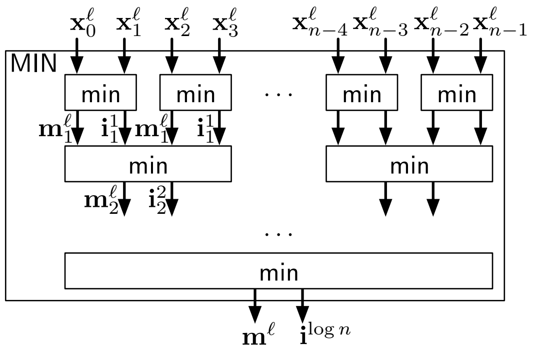

Fig. 9. Minimum Circuit (MIN)

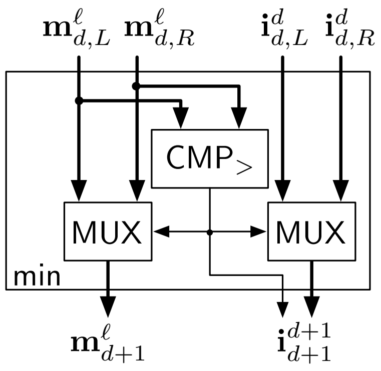

Fig. 10. Minimum Block (min)

4 Input/Output Conversion Protocols

In secure two-party computation protocols executed between circuit constructor $\mathcal{S}$ and circuit evaluator $\mathcal{C}$, each of the inputs and outputs of the securely computed functionality can be given in different forms depending on the application scenario: privately known to one party (§4.1), or homomorphically encrypted under the public key of the other party (§4.2). These inputs can be converted from different forms to garbled inputs given to $\mathcal{C}$. Afterwards, $\mathcal{C}$ evaluates the garbled circuit, obtains the garbled outputs, and converts them into outputs in the needed form.

The resulting communication complexities of these input and output conversion protocols for semi-honest parties are summarized in Table 2 and a detailed description of these known techniques is given next.

Table 2. Communication complexity for converting ℓ-bit inputs/outputs in different forms to inputs/outputs of a garbled circuit (parameters defined in §2.1, HE: Homomorphically Encrypted).

	Input	Output
Private $\mathcal{S}$ (§4.1)	$\ell t'$ bits	ℓ bits
Private $\mathcal{C}$ (§4.1)	$\mathrm{OT}^{\ell}_{t'}$	ℓ bits
HE (§4.2)	1 ciphertext + $5\ell t'$ bits + $\mathrm{OT}^{\ell}_{t'}$	1 ciphertext + $(\ell+\sigma)(5t'+1)$ bits

4.1 Private Inputs and Outputs

Private $\mathcal{S}$ Input: Inputs privately known to the circuit constructor $\mathcal{S}$ are easiest to deal with. For each of these inputs i, $\mathcal{S}$ sends the garbled value $\widetilde{w}_i^{v_i}$ corresponding to the plain value v_i to evaluator $\mathcal{C}$.

As described in [PSS09], in case of semi-honest constructor (i.e., with no cut-and-choose), the inputs of $\mathcal{S}$ can also be securely incorporated into the garbled circuit. This optimization avoids to transfer any additional data for $\mathcal{S}$'s private inputs and the size of the GC can be reduced as well. However, in many applications it is beneficial even in the semi-honest scenario to separate conversion of the inputs from creation of the garbled circuit, as this allows $\mathcal{S}$ to create the garbled circuit in an offline pre-computation phase already before its private inputs are known.

Private $\mathcal{C}$ Input: For private inputs w_i of the evaluator $\mathcal{C}$, both parties execute an OT protocol for each input bit in which constructor $\mathcal{S}$ inputs the two garbled t'-bit values $\widetilde{w}_i^0, \widetilde{w}_i^1$ and $\mathcal{C}$ inputs its plain value v_i to obtain $\widetilde{w}_i^{v_i}$ as output. For ℓ input bits, the OTs can be executed in a parallel $\mathrm{OT}^{\ell}_{t'}$ protocol which can efficiently be extended to OT^{t}_{t} as described in 2.3.

Private $\mathcal{S}$ Output: If the output of the functionality is a private output w_i of the evaluator $\mathcal{C}$, constructor $\mathcal{S}$ provides $\mathcal{C}$ with the output decryption table for w_i, i.e., the permutation bit π_i chosen when creating the garbled value $\widetilde{w}_i^0 = \langle k_i^0, \pi_i \rangle$.

Private $\mathcal{C}$ Output: For private outputs w_i of the constructor $\mathcal{S}$, evaluator $\mathcal{C}$ does not get an output decryption table but sends the obtained permutation bit π_i of the obtained garbled value $\widetilde{w}_i = \langle k_i, \pi_i \rangle$ back to $\mathcal{S}$ who can deduce the corresponding plain value from this. Clearly, this works only if $\mathcal{C}$ is semi-honest as otherwise he could easily flip the output bit. This can be prevented by requiring $\mathcal{C}$ to send the output key k_i instead.

4.2 Homomorphically Encrypted Inputs and Outputs

In the scenario of secure two-party computation based on homomorphic encryption, one party, say client $\mathcal{C}$, generates a key-pair of the homomorphic encryption scheme and sends the public key and its inputs encrypted under the public key to $\mathcal{S}$. Afterwards, $\mathcal{S}$ can perform operations on the ciphertexts which result in corresponding operations on the encrypted plaintext data (using the homomorphic property of the cryptosystem). In order to compute operations that are not compatible with the homomorphic property (e.g., multiplication of two ciphertexts encrypted with an additively homomorphic encryption scheme), additional communication rounds must be performed. In the following we show how computing on homomorphically encrypted data can be combined with a garbled circuit in order to efficiently evaluate non-linear functions, such as comparison, minimum search, or other arbitrary functionalities in a constant number of rounds.

Homomorphically Encrypted Input: If $\mathcal{S}$ holds an ℓ-bit value $[\![\mathbf{x}^\ell]\!]$, additively homomorphically encrypted under $\mathcal{C}$'s public key, this value can be converted into a garbled value $\widetilde{\mathbf{x}}^\ell$ output to $\mathcal{C}$ as follows: $\mathcal{S}$ chooses a random value r from the plaintext space P and adds this to the encrypted value: $[\![y]\!] = [\![\mathbf{x}^\ell + r]\!]$. $\mathcal{S}$ sends $[\![y]\!]$ to $\mathcal{C}$ who decrypts into y. Afterwards, both parties evaluate a garbled circuit which takes off the additive blinding: the private input of $\mathcal{S}$ into this garbled circuit are the ℓ least significant bits of r, $\mathbf{r}^\ell = r \mod 2^\ell$, and $\mathcal{C}$ inputs the ℓ least significant bits of y, $\mathbf{y}^\ell = y \mod 2^\ell$. The garbled circuit is an ℓ-bit subtraction circuit (cf. §3.2) which recovers the plaintext value from the blinded value as $\widetilde{x}^\ell = \widetilde{\mathbf{y}}^\ell - \widetilde{\mathbf{r}}^\ell$. This conversion protocol from additively homomorphically encrypted values into garbled values was used in [BPSW07,JP09]. A detailed proof of the protocol and further efficiency improvements, that can be achieved by packing together multiple values under encryption and converting this packed value at once, can be found in [BFK$^+$09].

Homomorphically Encrypted Output: A garbled ℓ-bit output $\widetilde{\mathbf{x}}^\ell$, held by $\mathcal{C}$ after having evaluated the garbled circuit, can be converted back into a homorphic encryption $[\![\mathbf{x}^\ell]\!]$ held by $\mathcal{S}$ as follows: $\mathcal{S}$ chooses a random $(\ell + \sigma)$-bit value $\mathbf{r}^{\ell+\sigma} \in_R \{0,1\}^{\ell+\sigma}$, where σ is a statistical security parameter, e.g., $\sigma = 80$. Now, a garbled $(\ell + \sigma)$-bit addition circuit (cf. §3.2) is evaluated which computes $\widetilde{\mathbf{y}}^{\ell+\sigma} = \widetilde{\mathbf{x}}^\ell + \widetilde{\mathbf{r}}^{\ell+\sigma}$: $\mathcal{C}$ inputs $\widetilde{\mathbf{x}}^\ell$ into this circuit and $\mathcal{S}$ provides $\mathcal{C}$ with the garbled value $\widetilde{\mathbf{r}}^{\ell+\sigma}$ together with an output decryption table for $\mathbf{y}^{\ell+\sigma}$. After evaluation of GC, $\mathcal{C}$ obtains $\mathbf{y}^{\ell+\sigma}$, encrypts it under his public key of the additively homomorphic cryptosystem and sends the ciphertext $[\![\mathbf{y}^{\ell+\sigma}]\!]$ to $\mathcal{S}$. $\mathcal{S}$ can subtract the blinding value $\mathbf{r}^{\ell+\sigma}$ under encryption and gets $[\![\mathbf{x}^\ell]\!]$. This output conversion protocol can be proven secure analogous to the proof of the conversion protocol for homomorphically encrypted inputs [BFK$^+$09].

5 Applications

We now describe how our efficient circuit constructions (§3) can be applied to improve previous solutions for several applications. We note that constructions of this section largely are not novel and may be folklore knowledge. We explicate them for concreteness, and use them to demonstrate the usefulness of our building blocks and to arrive at performance estimates to form a baseline for future protocols.

5.1 Integer Comparison (Millionaires Problem)

The "Millionaires problem" was introduced by Yao in [Yao82] as motivation for secure compuation: two millionaires want to securely compare their respective private input values (e.g., their amount of money) without revealing more information than the outcome of the comparison to the other party. More specifically, client $\mathcal{C}$ holds a private ℓ-bit value x^ℓ and server $\mathcal{S}$ holds a private ℓ-bit value y^ℓ. The output bit $z = [x^\ell > y^\ell]$ should be revealed to both.

We obtain an efficient solution for the Millionaires problem simply by evaluating the comparison circuit of §3.3 with the GC protocol of [KS08] and an efficient OT protocol. Our protocol, when executed without precomputation has asymptotic communication complexity $5\ell t + \mathrm{OT}^\ell_t$ bit, where t is the symmetric security parameter (cf. §2.1).

In many practical application scenarios it is beneficial to shift as much of the computation and communication cost of a protocol into a setup (precomputation) phase, which is executed before the parties' inputs are known, while the parties' workload is low. In the following we apply a folklore technique, which demonstrates that GC protocols are ideally suited for precomputation as (in contrast to many protocols based on homomorphic encryption) almost their entire cost can be shifted into the setup phase.

Millionaires with setup. GC protocols allow to move all expensive operations (i.e., computationally expensive OT and creation of GC, as well as the transfer of GC which dominates the communication complexity) into the setup phase. The idea is to create and transfer the garbled circuit in the setup phase, and precompute the parallel OTs [Bea95]: this can be done by executing the parallel OT protocol on randomly chosen (by $\mathcal{C}$ and $\mathcal{S}$) values of corresponding sizes (instead of private inputs of $\mathcal{C}$ and pairs of garbled input values of $\mathcal{S}$). Then, in the online phase, $\mathcal{C}$ uses its randomly chosen value to mask his private inputs, and sends them to $\mathcal{S}$. $\mathcal{S}$ replies with encryptions of wire's garbled inputs using his random values from the setup phase (which garbled input is masked with which random value is determined by $\mathcal{C}$'s message) . Finally, $\mathcal{C}$ can use the masks he received from the OT protocol in the setup phase to exactly decrypt the correct garbled input value.

More specifically, the **setup phase** works as follows: for $i = 1, \ldots, \ell$, $\mathcal{C}$ chooses random bits $r_i \in_R \{0,1\}$ and $\mathcal{S}$ chooses random masks $m_i^0, m_i^1 \in_R \{0,1\}^{t'}$ (recall, $t' = t + 1$ is the bitlength of garbled values). Both parties run a $\mathrm{OT}^\ell_{t'}$ protocol on these randomly chosen values, where $\mathcal{S}$ inputs the pairs m_i^0, m_i^1 and $\mathcal{C}$ inputs r_i and $\mathcal{C}$ obliviously obtains the mask $m_i = m_i^{r_i}$. Additionally, $\mathcal{S}$ creates a garbled circuit $\widetilde{C}$ with garbled inputs $\widetilde{x}_i^0, \widetilde{x}_i^1$ and $\widetilde{y}_i^0, \widetilde{y}_i^1$ and sends

$\widetilde{C}$ together with the output decryption table to $\mathcal{C}$. This message has the size $4\ell t' + 1 \sim 4\ell t$ bits. Overall, the setup phase has a slightly smaller communication complexity than the Millionaires protocol without setup described above.

In the **online phase**, $\mathcal{S}$ sends the garbled values $\widetilde{\mathbf{y}}^\ell$ corresponding to his input $\mathbf{y}^\ell$ to $\mathcal{C}$ and the online part of the OT protocol is executed: for each $i = 1, \ldots, \ell$, $\mathcal{C}$ masks its input bits x_i with r_i as $X_i = x_i \oplus r_i$ and sends these masked bits to $\mathcal{S}$. $\mathcal{S}$ responds with the masked pair of t'-bit strings $\langle M_i^0, M_i^1 \rangle = \langle m_i^0 \oplus \widetilde{x}_i^0, m_i^1 \oplus \widetilde{x}_i^1 \rangle$ if $X_i = 0$ or $\langle M_i^0, M_i^1 \rangle = \langle m_i^0 \oplus \widetilde{x}_i^1, m_i^1 \oplus \widetilde{x}_i^0 \rangle$ otherwise. $\mathcal{C}$ obtains $\langle M_i^0, M_i^1 \rangle$ and decrypts $\widetilde{x}_i = M_i^{r_i} \oplus m_i$. Using the garbled inputs $\widetilde{\mathbf{x}}^\ell, \widetilde{\mathbf{y}}^\ell$, $\mathcal{C}$ evaluates the garbled circuit $\widetilde{C}$, obtains the result from the output decryption table and sends it back to $\mathcal{S}$. Overall, in the online phase $\ell t' + 2\ell t' + 1 \sim 3\ell t$ bits are sent.

Cost Evaluation. *Computation Complexity.* As our improved GC for integer comparison consists of no more than ℓ non-XOR 2-to-1 gates (cf. comparison circuit in §3.3), $\mathcal{C}$ needs to invoke the underlying cryptographic hash-function (e.g., SHA-256 for $t = 128$ bit symmetric security) exactly ℓ times to evaluate the GC (cf. §2.4). All other operations are negligible (XOR of t-bit strings). Hence, the computational complexity of the online phase of our protocol is negligible as compared to that of protocols based on homomorphic encryption. Even with an additional setup phase, those protocols need to invoke a few modular operations for each input bit which are usually by several orders of magnitude more expensive than the evaluation of a cryptographic hash function used in our protocols. Further the computational complexity of the setup phase in our protocol is more efficient than in protocols based on homomorphic encryption when using efficient OT protocols implemented over elliptic curves and efficient extensions of OT for a large number of inputs (cf. §2.3).

Communication Complexity. Table 3 shows that also the communication complexity of our protocol is much lower than that of previous protocols which are based on homomorphic encryption. As underlying $\mathrm{OT}_{t'}^{\ell}$ protocol we use the protocol of [AIR01] implemented over a suitably chosen elliptic curve and using point compression described in §A. This protocol has asymptotic communication complexity $12\ell t$ bits and is secure in the standard model. (Using the protocol of [NP01] which is secure in the random oracle model would result in communication complexity $6\ell t$ bits and much lower computation complexity.) The chosen values for the security parameters correspond to standard recommendations for short-term (upto 2010, $t = 80$, $T = 1024$), medium-term (upto 2030, $t = 112$, $T = 2048$) and long-term security (after 2030, $t = 128$, $T = 3072$) [GQ09].

We summarize these findings in Table 3.

5.2 Minimum Distance

Finally, we give an efficient protocol for secure computation of the *minimum distance* (or *nearest neighbor*) between a private query point Q, held by client $\mathcal{C}$, and an ordered list of private points $P_0, \ldots, P_{n-1}$ (called *database*), held by server

Table 3. Asymptotic communication complexity of comparison protocols on ℓ-bit values, $\ell = 16, \kappa = 40$ (parameters defined in §2.1).

Security	Previous Work			This Work		
Level	[Fis01]	[BK04]	[DGK07]	Setup Phase	Online Phase	Total
asymptotic	$(\kappa+1)\ell T$	$4\ell T$	$2\ell T$	$4\ell t + 12\ell t$	$3\ell t$	$19\ell t$
short-term	82 kByte	8 kByte	4 kByte	$0.6 + 1.9$ kByte	0.5 kByte	3.0 kByte
medium-term	164 kByte	16 kByte	8 kByte	$0.9 + 2.6$ kByte	0.7 kByte	4.2 kByte
long-term	246 kByte	24 kByte	12 kByte	$1.0 + 3.0$ kByte	0.8 kByte	4.8 kByte

$\mathcal{S}$. The protocol consists of two sub-protocols: The first sub-protocol computes for $i = 1, \ldots, n$ the encrypted distance $[\![\delta_i]\!]$ of the query point Q to each point P_i in the database, using a suitably chosen homomorphic encryption scheme, and outputs these encrypted distances to $\mathcal{S}$. The second sub-protocol securely selects the minimum value and index of these encrypted distances and outputs the minimum distance δ_{min} and minimum index i_{min} to $\mathcal{C}$.

Distance Computation. We sketch the sub-protocols for securely computing the encrypted distance $[\![\delta_i]\!]$ between the query point Q and the points P_i in the database in the following.

Hamming Distance. The Hamming distance between two points $P = (p_1, \ldots, p_m)$ and $Q = (q_1, \ldots, q_m)$ with $p_j, q_j \in \{0,1\}$ is defined as $d_H(P,Q) := \sum_{j=1}^{m} p_j \oplus q_j = \sum_{i=1}^{m} (1-p_j)q_j + p_j(1-q_j)$. Using an additively homomorphic encryption scheme, the Hamming distance can then be computed as follows: $\mathcal{C}$ generates a public-key pk and a corresponding secret-key of an additively homomorphic cryptosystem and sends (the verifiably correct) pk and bitwise homomorphic encryptions of Q, $[\![q_1]\!], \ldots, [\![q_m]\!]$, to $\mathcal{S}$. As computing the Hamming distance is a linear operation, S can compute the encrypted Hamming distance to each point $P = P_i$ in its database as $[\![\delta_i]\!] = [\![d_H(P,Q)]\!]$ from $[\![q_j]\!]$ and p_j using standard techniques as proposed in [JP09].

Euclidean Distance. The Euclidean distance can be seen as an extension of the Hamming distance from 1-bit coordinates to ℓ-bit coordinates, i.e., for $j = 1, \ldots, m : p_j, q_j \in \{0,1\}^\ell$. The Euclidean distance is then defined as $d_E(P,Q) := \sqrt{\sum_{j=1}^{m}(p_j - q_j)^2}$. As the Euclidean distance is not negative, it is sufficient to compute the square of the Euclidean distance instead, in order to find the minimum (or maximum) Euclidean distance: $d_E(P,Q)^2 = \sum_{j=1}^{m}(p_j - q_j)^2$. The encryption of the square of the Euclidean distance $[\![\delta_i^2]\!] = [\![d_E(P_i,Q)^2]\!]$ can be computed analogously to the protocol for the Hamming distance by using additively homomorphic encryption which allows for at least one multiplication (cf. §2.2). Alternatively, additively homomorphic encryption together with an additional round for squaring can be used as proposed in [EFG$^+$09].

Minimum Selection. After having securely computed the homomorphically encrypted distances $[\![\delta_i]\!]$ held by $\mathcal{S}$, the minimum and minimum index of these values can be selected by converting these homomorphically encrypted values to garbled values as described in §4.2 and securely evaluating the minimum circuit of §3.4. The asymptotic communication complexity of this minimum selection protocol is $13\ell nt$ bits for the garbled circuits (when GCs are pre-computed), n homomorphic ciphertexts, and $\mathrm{OT}^{n\ell}_{t'}$. The number of homomorphic ciphertexts can be further reduced using packing (§4.2), and the number of OTs can be reduced to a constant number of OTs (§2.3). As for the other application scenarios described before, all expensive operations can be moved into a setup phase and the entire protocol has a constant number of rounds.

References

[AHL05] L. v. Ahn, N. J. Hopper, and J. Langford. Covert two-party computation. In *ACM Symposium on Theory of Computing (STOC'05)*, pages 513–522. ACM, 2005.

[AIR01] W. Aiello, Y. Ishai, and O. Reingold. Priced oblivious transfer: How to sell digital goods. In *Advances in Cryptology – EUROCRYPT'01*, volume 2045 of *LNCS*, pages 119–135. Springer, 2001.

[AS08] F. Armknecht and A.-R. Sadeghi. A new approach for algebraically homomorphic encryption. Cryptology ePrint Archive, Report 2008/422, 2008.

[Bea95] D. Beaver. Precomputing oblivious transfer. In *Advances in Cryptology – CRYPTO'95*, volume 963 of *LNCS*, pages 97–109. Springer, 1995.

[BFK$^+$09] M. Barni, P. Failla, V. Kolesnikov, R. Lazzeretti, A.-R. Sadeghi, and T. Schneider. Secure evaluation of private linear branching programs with medical applications. In *14th European Symposium on Research in Computer Security (ESORICS'09)*, LNCS. Springer, 2009. Full version available at http://eprint.iacr.org/2009/195.

[BGN05] D. Boneh, E.-J. Goh, and K. Nissim. Evaluating 2-dnf formulas on ciphertexts. In *Theory of Cryptography (TCC'05)*, volume 3378 of *LNCS*, pages 325–341. Springer, 2005.

[BK04] I. F. Blake and V. Kolesnikov. Strong conditional oblivious transfer and computing on intervals. In *Advances in Cryptology – ASIACRYPT'04*, volume 3329 of *LNCS*, pages 515–529. Springer, 2004.

[BPSW07] J. Brickell, D. E. Porter, V. Shmatikov, and E. Witchel. Privacy-preserving remote diagnostics. In *ACM Conference on Computer and Communications Security (CCS'07)*, pages 498–507. ACM, 2007.

[Bro05] D. R. L. Brown. Certicom proposal to revise SEC 1: Elliptic curve cryptography, version 1.0. Technical report, Certicom Research, 2005. Available from http://www.secg.org.

[CF05] H. Cohen and G. Frey, editors. *Handbook of elliptic and hyperelliptic curve cryptography.* CRC Press, 2005.

[DGK07] I. Damgård, M. Geisler, and M. Krøigaard. Efficient and secure comparison for on-line auctions. In *Australasian Conference on Information Security and Privacy (ACISP'07)*, volume 4586 of *LNCS*, pages 416–430. Springer, 2007.

[DGK08a] I. Damgård, M. Geisler, and M. Krøigaard. A correction to "efficient and secure comparison for on-line auctions". Cryptology ePrint Archive, Report 2008/321, 2008.

[DGK08b] I. Damgård, M. Geisler, and M. Krøigaard. Homomorphic encryption and secure comparison. *Journal of Applied Cryptology*, 1(1):22–31, 2008.

[DJ01] I. Damgård and M. Jurik. A generalisation, a simplification and some applications of paillier's probabilistic public-key system. In *Public-Key Cryptography (PKC'01)*, LNCS, pages 119–136. Springer, 2001.

[EFG$^+$09] Z. Erkin, M. Franz, J. Guajardo, Katzenbeisser, I. Lagendijk, and T. Toft. Privacy-preserving face recognition. In *Privacy Enhancing Technologies (PET'09)*, volume 5672 of *LNCS*, pages 235–253. Springer, 2009.

[FIP00] FIPS 186-2, Digital Signature Standard. Federal Information Processing Standards Publication 186-2, 2000. Available from http://csrc.nist.gov/.

[Fis01] M. Fischlin. A cost-effective pay-per-multiplication comparison method for millionaires. In *Cryptographer's Track at RSA Conference (CT-RSA'01)*, volume 2020 of *LNCS*, pages 457–472. Springer, 2001.

[Gen09] C. Gentry. Fully homomorphic encryption using ideal lattices. In *ACM Symposium on Theory of Computing (STOC'09)*, pages 169–178. ACM, 2009.

[GM84] S. Goldwasser and S. Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28(2):270–299, 1984.

[GMS08] V. Goyal, P. Mohassel, and A. Smith. Efficient two party and multi party computation against covert adversaries. In *Advances in Cryptology – EUROCRYPT'08*, volume 4965 of *LNCS*, pages 289–306. Springer, 2008.

[GQ09] D. Giry and J.-J. Quisquater. Cryptographic key length recommendation, March 2009. http://keylength.com.

[GSV07] J. A. Garay, B. Schoenmakers, and J. Villegas. Practical and secure solutions for integer comparison. In *Public Key Cryptography (PKC'07)*, volume 4450 of *LNCS*, pages 330–342. Springer, 2007.

[IKNP03] Y. Ishai, J. Kilian, K. Nissim, and E. Petrank. Extending oblivious transfers efficiently. In *Advances in Cryptology – CRYPTO'03*, volume 2729 of *LNCS*. Springer, 2003.

[JP09] A. Jarrous and B. Pinkas. Secure hamming distance based computation and its applications. In *Applied Cryptography and Network Security (ACNS'09)*, volume 5536 of *LNCS*, pages 107–124. Springer, 2009.

[JS07] S. Jarecki and V. Shmatikov. Efficient two-party secure computation on committed inputs. In *Advances in Cryptology – EUROCRYPT'07*, volume 4515 of *LNCS*, pages 97–114. Springer, 2007.

[Ker08] F. Kerschbaum. Practical privacy-preserving benchmarking. *International Information Security Conference (SEC'08)*, 278:17–31, 2008.

[Kob87] N. Koblitz. Elliptic curve cryptosystems. *Mathematics of Computation*, 48(177):203–209, 1987.

[Kol05] V. Kolesnikov. Gate evaluation secret sharing and secure one-round two-party computation. In *Advances in Cryptology – ASIACRYPT'05*, volume 3788 of *LNCS*, pages 136–155. Springer, 2005.

[KS08] V. Kolesnikov and T. Schneider. Improved garbled circuit: Free XOR gates and applications. In *International Colloquium on Automata, Languages and Programming (ICALP'08)*, volume 5126 of *LNCS*, pages 486–498. Springer, 2008.

[Lip03] H. Lipmaa. Verifiable homomorphic oblivious transfer and private equality test. In *Advances in Cryptology – ASIACRYPT'03*, volume 2894 of *LNCS*. Springer, 2003.

[LP04] Y. Lindell and B. Pinkas. A proof of Yao's protocol for secure two-party computation. ECCC Report TR04-063, Electronic Colloquium on Computational Complexity (ECCC), 2004.

[LP07] Y. Lindell and B. Pinkas. An efficient protocol for secure two-party computation in the presence of malicious adversaries. In *Advances in Cryptology – EUROCRYPT'07*, volume 4515 of *LNCS*, pages 52–78. Springer, 2007.

[MNPS04] D. Malkhi, N. Nisan, B. Pinkas, and Y. Sella. Fairplay — a secure two-party computation system. In *USENIX*, 2004.

[NO09] J. B. Nielsen and C. Orlandi. Lego for two-party secure computation. In *Theory of Cryptography (TCC'09)*, volume 5444 of *LNCS*, pages 368–386. Springer, 2009.

[NP01] M. Naor and B. Pinkas. Efficient oblivious transfer protocols. In *ACM-SIAM Symposium On Discrete Algorithms (SODA'01)*, pages 448–457. Society for Industrial and Applied Mathematics, 2001.

[NPS99] M. Naor, B. Pinkas, and R. Sumner. Privacy preserving auctions and mechanism design. In *ACM Conference on Electronic Commerce*, pages 129–139, 1999.

[OPB07] C. Orlandi, A. Piva, and M. Barni. Oblivious neural network computing via homomorphic encryption. *European Journal of Information Systems (EURASIP)*, 2007(1):1–10, 2007.

[Pai99] P. Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *Advances in Cryptology – EUROCRYPT'99*, volume 1592 of *LNCS*, pages 223–238. Springer, 1999.

[PCB$^+$08] A. Piva, M. Caini, T. Bianchi, C. Orlandi, and M. Barni. Enhancing privacy in remote data classification. *New Approaches for Security, Privacy and Trust in Complex Environments (SEC'08)*, 2008.

[PSS09] A. Paus, A.-R. Sadeghi, and T. Schneider. Practical secure evaluation of semi-private functions. In *Applied Cryptography and Network Security (ACNS'09)*, volume 5536 of *LNCS*, pages 89–106. Springer, 2009.

[QA08] Y. Qi and M. J. Atallah. Efficient privacy-preserving k-nearest neighbor search. In *International Conference on Distributed Computing Systems (ICDCS'08)*, pages 311–319. IEEE, 2008.

[SEC00a] Standards for efficient cryptography, SEC 1: Elliptic curve cryptography. Technical report, Certicom Research, 2000. Available from http://www.secg.org.

[SEC00b] Standards for efficient cryptography, SEC 2: Recommended elliptic curve domain parameters. Technical report, Certicom Research, 2000. Available from http://www.secg.org.

[Sho04] Victor Shoup. Sequences of games: a tool for taming complexity in security proofs. Cryptology ePrint Archive, Report 2004/332, 2004.

[SKK06] M. Shaneck, Y. Kim, and V. Kumar. Privacy preserving nearest neighbor search. In *International Conference on Data Mining - Workshops (ICDMW'06)*, pages 541–545. IEEE, 2006.

[TP91a] M. Turk and A. Pentland. Eigenfaces for recognition. *Journal of Cognitive Neuroscience*, 3(1):71–86, 1991.

[TP91b] M. A. Turk and A. P. Pentland. Face recognition using eigenfaces. In *IEEE Computer Vision and Pattern Recognition (CVPR'91)*, pages 586–591. IEEE, 1991.

[TY98] Yiannis Tsiounis and Moti Yung. On the security of elgamal based encryption. In *Public-Key Cryptography (PKC'98)*, volume 1431 of *LNCS*, pages 117–134. Springer, 1998.

[Yao82] A. C. Yao. Protocols for secure computations. In *Symposium on Foundations of Computer Science (SFCS'82)*, pages 160–164. IEEE, 1982.
[Yao86] A. C. Yao. How to generate and exchange secrets. In *IEEE Symposium on Foundations of Computer Science (FOCS'86)*, pages 162–167. IEEE, 1986.

A Efficient OT protocol of [AIR01] over elliptic curves

In the following we describe the efficient two-move OT protocol of [AIR01] instantiated with elliptic curves as proposed in [Lip03] in detail. The protocol of [AIR01] is based on a homomorphic, semantically secure cryptosystem with *verifiability* property, i.e., the validity of a public key *pk* and the validity of a ciphertext c with respect to a valid *pk* can efficiently be checked. In general, e.g., for the Paillier scheme [Pai99], additional Zero-Knowledge Proofs-of-Knowledge (ZK-PoK) are needed to prove these properties. As outlined in [AIR01], the multiplicatively homomorphic ElGamal scheme enjoys verifiability without such ZK-PoKs, as it can easily be checked whether a given element g has prime order q by checking $g^q \overset{?}{=} 1$.

EC-ElGamal. For efficiency reasons, ElGamal can be implemented over a suitably chosen elliptic curve (EC-ElGamal) which results in the protocol shown in Protocol 1. Recall, an elliptic curve over a prime field $\mathbb{F}_p$ is parametrized by the six-tuple $T = (p, a, b, P, q, h)$, where p is a $2t$-bit prime (t is the symmetric security parameter), $a, b \in \mathbb{F}_p$ specify the equation $E : y^2 \equiv x^3 + ax + b \mod p$, $P = (x_P, y_P) \in E(\mathbb{F}_p)$ is the base point with large prime order q ($2t$ bit in the curves we choose) and $h = \#E(\mathbb{F}_p)/q$ is a small cofactor. O denotes the point at infinity. Each point can be represented with $2t + 1$ bits using point compression which is computationally more expensive than the uncompressed representation which has size $4t$ bits [SEC00a].

Point embedding. We assume, that the strings s^0, s^1 are short enough to be embedded into points on the elliptic curve. A probabilistic algorithm for embedding a string into the x-coordinate of a point was proposed in [Kob87]: let κ be a statistical correctness parameter (e.g., $\kappa = 10$ is proposed in [Kob87]). Then, assuming $0 \leq s < q/2^\kappa - 1$, we try to append κ bits to s until we obtain an x, $2^\kappa s \leq x < 2^\kappa(s+1) < q$, such that $f(x) = x^3 + ax + b$ is a square in $\mathbb{F}_p$ (this can efficiently be tested by computing the Legendre-Symbol and testing if $\left(\frac{x}{p}\right) \overset{?}{=} 1$). Now, s is embedded into the point $S = (x, \sqrt{f(x)}) \in E(\mathbb{F}_p)$. Obviously, S can be decoded back into string s by dropping the last κ bits of its x-coordinate.

Choice of Elliptic Curves. An implementation could use the curves *secp160*, *secp224r1*, resp. *secp256r1* from the SECG standard [SEC00b,SEC00a,Bro05] which corresponds to symmetric security levels of t = 80, 112, resp. 128 bit. For these curves, the DDH assumption is assumed to hold as they are chosen verifiably at random [SEC00b,SEC00a] compliant with the recommendations in many international standards such as the Digital Signature Standard of NIST

[FIP00]. Additionally, these curves allow efficient checking whether a point G is a scalar multiple of the base point P, as they have cofactor $h = 1$ [SEC00b], which implies that each point $G \neq O$ that lies on the curve, i.e., satisfies E, is a scalar multiple of P [CF05].

Protocol 1 OT_ℓ^1 protocol of [AIR01] instantiated with EC-ElGamal

```
Input C: bit $b \in \{0,1\}$
Input S: pair of strings $s^0, s^1 \in \{0,1\}^\ell$
Output C: string $s^b$
Output S: $success \in \{\perp, \top\}$
1: Setup phase: $\mathcal{C}$ generates EC-ElGamal keypair (chooses secret key $s \in_R \mathbb{Z}_q$ and
   computes public key $Q = [s]P$) and sends public key $Q$ to $\mathcal{S}$.
2: Setup phase: $\mathcal{S}$ verifies that $Q \overset{?}{\in} E$ and aborts with $success = \perp$ otherwise.
3: $\mathcal{C}$ encrypts $B = O$ or $B = P$ with EC-ElGamal and public key $Q$ depending on $b$:
   choose $r \in_R \mathbb{Z}_q$, compute $C_1 = [r]P$, $C_2 = [r]Q = [r+s]P$, if $b = 1$ then $C_2 =$
   $C_2 + P$.
4: $\mathcal{C}$ sends $C_1, C_2$ to $\mathcal{S}$.
5: $\mathcal{S}$ verifies that $C_1, C_2 \overset{?}{\in} E$ and aborts with $success = \perp$ otherwise.
6: $\mathcal{S}$ maps $s^0$ to point $S_0$ and $s^1$ to point $S_1$.
7: $\mathcal{S}$ computes $(C_1^0, C_2^0)$ as conditional disclosure of $S_0$ conditioned on $B = O$:
   choose $r_0, s_0 \in_R \mathbb{Z}_q$, compute $C_1^0 = [s_0]C_1 + [r_0]P$, $C_2^0 = [s_0]C_2 + [r_0]Q + S_0$.
8: $\mathcal{S}$ computes $(C_1^1, C_2^1)$ as conditional disclosure of $S_1$ conditioned on $B = P$:
   choose $r_1, s_1 \in_R \mathbb{Z}_q$, compute $C_1^1 = [s_1]C_1 + [r_1]P$, $C_2^1 = [s_1](C_2 - P) + [r_1]Q + S_1$.
9: $\mathcal{S}$ sends $C_1^0, C_2^0, C_1^1, C_2^1$ to $\mathcal{C}$ and outputs $success = \top$ to $\mathcal{S}$.
10: $\mathcal{C}$ decrypts $S_b = C_2^b - [s]C_1^b$ and maps this point to the string $s^b$ which is output
    to $\mathcal{C}$.
```

Communication Complexity. The asymptotic communication complexity of the OT_ℓ^1 protocol given in Protocol 1 is $12t$ bits, as overall 6 points of size $2t+1$ bits each are sent. The corresponding parallel OT_ℓ^m protocol can be easily obtained by running this protocol m times in parallel (with same setup phase) and has asymptotic communication complexity $12mt$ bits.

Theorem 1 (Security). *The* OT_ℓ^1 *protocol given in Protocol 1 is secure against malicious* $\mathcal{C}$ *and semi-honest* $\mathcal{S}$ *provided the DDH assumption holds for the underlying elliptic curve.*

Proof. The proof of theorem 1 follows directly from the proof for the OT protocol in [AIR01, Sect. 5.1] and the proof that the semantic security of ElGamal is equivalent to the Decision Diffie-Hellman (DDH) assumption in the underlying group [TY98].

Hashed EC-ElGamal. Instead of using EC-ElGamal, the semantically secure Hashed EC-ElGamal encryption scheme [Sho04] can be used. In hashed EC-ElGamal, the message s is not embedded into a point S on the elliptic curve. Instead, a random point R on the elliptic curve is encrypted from which a symmetric encryption key $H_\kappa(R)$ for encryption of s is derived using an entropy smoothing hash function H_κ. As no point embedding is needed in hashed EC-ElGamal, this reduces the computational complexity of $\mathcal{S}$. However, the communication complexity is slightly increased by $|s|$ bits as the ciphertext additionally contains the symmetric encryption of the message s.