Bandwidth-optimized Secure Two-Party Computation of Minima

Jan Henrik Ziegeldorf, Jens Hiller, Martin Henze,
Hanno Wirtz, and Klaus Wehrle

Communication and Distributed Systems (COMSYS),
RWTH Aachen University, Aachen, Germany
{ziegeldorf, hiller, henze, wirtz, wehrle}@comsys.rwth-aachen.de

Abstract. Secure Two-Party Computation (STC) allows two mutually untrusting parties to securely evaluate a function on their private inputs. While tremendous progress has been made towards reducing processing overheads, STC still incurs significant communication overhead that is in fact prohibitive when no high-speed network connection is available, e.g., when applications are run over a cellular network. In this paper, we consider the fundamental problem of securely computing a minimum and its argument, which is a basic building block in a wide range of applications that have been proposed as STCs, e.g., Nearest Neighbor Search, Auctions, and Biometric Matchings. We first comprehensively analyze and compare the communication overhead of implementations of the three major STC concepts, i.e., Yao's Garbled Circuits, the Goldreich-Micali-Wigderson protocol, and Homomorphic Encryption. We then propose an algorithm for securely computing minima in the semi-honest model that, compared to current state-of-the-art, reduces communication overheads by 18 % to 98 %. Lower communication overheads result in faster runtimes in constrained networks and lower direct costs for users.

1 Introduction

The increasing collection of sensitive user data provided by mobile devices at cloud services, e.g., in genetic testing [9], gives rise to significant privacy concerns. However, performing all necessary computations exclusively on the mobile device, to preserve the user's privacy, is infeasible as this could disclose business secrets of the service provider. In this scenario, Secure Two-Party Computation (STC) presents a generic solution to reconcile these conflicting privacy interests.

The performance of STC has been thoroughly investigated in a static setting with a high-speed LAN connection. In this setting, processing overheads are the main performance bottleneck and tremendous improvements, both practical and theoretical, have been made in this regard, mainly focussing on Yao's Garbled Circuits approach [35]. STC in more constrained environments, e.g., between mobile devices that interact spontaneously, has only recently received interest [4, 5, 7, 9, 10, 18]. Still, these works assume a network with low latency and high throughput [5,7,10,18], prior interaction for pre-computations [5,9,18],

additional hardware or third parties [5, 10], or consider only specialized applications [4, 7]. In contrast, we strive to enable STCs in a purely ad-hoc manner between mobile devices and/or a cloud service over constrained, e.g., cellular, networks. In this setting, we argue for bandwidth consumption as a primary optimization goal. First, high-bandwidth STCs may quickly deplete users' data volume, inducing capped bandwidths or significant costs for subsequent communication. Second, less available bandwidth incurs significant transmission overheads which then constitute the main performance bottleneck. We show that this bottleneck eventually dominates the processing time, as also noted in [4, 7, 37].

In this work, we hence set out to analyze the bandwidth usage of the three major STC concepts Yao's Garbled Circuits (GCs) [35], the Goldreich-Micali-Wigderson (GMW) Protocol [16], and Homomorphic Encryption (HE). We focus on the fundamental STC problem of computing a minimum and its argument, since this problem i) has widely been considered in all three STC concepts and ii) is central to many STC applications, e.g., Nearest Neighbor search [30], Auctions [22], and Biometric Matching [17]. Our contributions are the following:

Thorough Analysis. We exactly quantify the communication complexity for state-of-the-art constructions and implementations of the (arg)min problem in GC, GMW, and HE. In constrained environments, we argue that the induced communication overheads quickly render mobile STCs infeasible.

Bandwidth-optimized (arg)min. We propose an HE-based, bandwidth-optimized (arg)min algorithm that reduces the communication overhead by 18 % to 98 % compared to state-of-the-art approaches. In constrained networks with bandwidths ranging from $\leq$1 MBit/s (e.g., Bluetooth, 3G) to $\leq$12-50 MBit/s (e.g., LTE), this translates to lower costs for end-users and even affords faster runtimes than bandwidth-heavy algorithms. We demonstrate the feasibility and performance of our protocol along a prototype implementation.

We discuss background on STC concepts and related work in §2. In §3, we analyze the communication complexity of previous (arg)min protocols. We present our improved protocol in §4 and its evaluation in §5. §6 concludes this paper. Note that all results also directly apply to the symmetric (arg)max problem.

2 Background

STC allows two mutually distrusting parties, i.e., a *client* $\mathcal{C}$ with private input x and a *server* $\mathcal{S}$ with private input y, to compute a known functionality $\mathcal{F}(x, y)$ without anyone learning the private inputs. Three predominant concepts for STC exist, partly building on Oblivious Transfer: Yao's Garbled Circuits, the Goldreich-Micali-Wigderson protocol, and Homomorphic Encryption.

Oblivious Transfer (OT). In the most general form of Oblivious Transfer, i.e., 1-out-of-n-OT_l^m, $\mathcal{S}$ holds m distinct n tuples of l-bit strings and $\mathcal{C}$ chooses exactly one string from each n tuple, while $\mathcal{S}$ learns nothing about $\mathcal{C}$'s choices. Formally, $\mathcal{S}$ holds $(s_{11}, ..., s_{1n}), ..., (s_{m1}, ..., s_{mn})$ with $s_{ij} \in \{0,1\}^l$ and $\mathcal{C}$ holds m choices $r_1, ..., r_m \in \{1...n\}$ and obtains the strings $s_{ir_i, 1 \leq i \leq m}$ while $\mathcal{S}$ has no output. 1-out-of-n-OT_l^m can be efficiently instantiated by first reducing it to

$m \log_2(n)$ invocations of 1-out-of-2-OT [27, 28] and then reducing the resulting large number of long l bit OTs to a small number of short t bit *Base OTs*, i.e., 1-out-of-2-OT$_t^t$ [1, 20] (*OT Extension*). The communication overhead for 1-out-of-n-OT$_l^m$ then amounts to $mnl + 3m\log(n)t$ bit, with symmetric security parameter t (e.g., AES key length). Base OTs can be implemented at the costs of an additional $2t^2+tT$ bit, where T is the asymmetric security level (e.g., bitlength of an RSA modulus) [27, 28]. The overhead for base OTs is often neglected in related work as it amortizes over a large number of OT Extensions. To further increase the communication efficiency of OT, custom OT variants exist. For GCs, e.g., general 1-out-2-OT can be replaced by *correlated* 1-out-of-2-OT [11], where $\mathcal{S}$'s inputs are correlated, reducing communication overhead to $m(t + l)$ bit per OT [1]. Similarly, in GMW, *random* 1-out-2-OT [11] obtains inputs randomly from a correlation-robust one-way function, reducing communication overhead to mt bits per OT [1].

Garbled Circuits (GC). Yao's Garbled Circuits [35] require to represent (automatically with special compilers) the desired functionality $\mathcal{F}(\cdot)$ as a Boolean circuit. After compiling the Boolean circuit, $\mathcal{S}$ *garbles* the circuit by encrypting and permuting the truth table entries for each circuit gate. Then $\mathcal{S}$ sends the garbled circuit $\tilde{\mathcal{F}}(\cdot)$ and its own garbled inputs $\tilde{y}$ to $\mathcal{C}$. $\mathcal{C}$ obtains its own garbled inputs $\tilde{x}$ via correlated OT from $\mathcal{S}$, with parameter $m = |x|$, the total bitlength of $\mathcal{C}$'s input. Finally, $\mathcal{C}$ evaluates $\tilde{\mathcal{F}}(\tilde{x}, \tilde{y})$ by decrypting the garbled circuit gate by gate to obtain the result. The communication overhead of GCs is almost completely due to the transmission of the garbled circuit and of the garbled inputs (via OT). It is thus critical to construct *size-efficient* circuits and to minimize inputs. The size of a circuit is usually measured in the number of Non-XOR gates, since XOR gates cause virtually no overheads due to the "free-XOR" optimization [24].

Goldreich-Micali-Wigderson (GMW). The GMW protocol [16], similarly to Yao's protocol, securely evaluates Boolean circuits. However, instead of garbling the circuit, it is evaluated jointly by $\mathcal{C}$ and $\mathcal{S}$ using an XOR-based 2-out-of-2 secret sharing scheme, i.e., Boolean sharings. The GMW protocol allows local evaluation of XOR gates while AND gates require interaction between $\mathcal{C}$ and $\mathcal{S}$, i.e., an exchange of 2 bit and one random OT per gate. Thus, while Yao's protocol has constant round complexity, the round complexity of GMW corresponds to the multiplicative depth of the circuit, i.e., the maximum number of AND gates on any path through the boolean circuit. Besides reducing the size of circuits to reduce the communication overhead, *depth-efficient* circuits are crucial to minimize communication rounds.

Homomorphic Encryption (HE). HE-based STC protocols allow to compute specific arithmetic operations under encryption, e.g., the Paillier cryptosystem [29] allows addition. Because Fully Homomorphic Encryption schemes currently still cause prohibitive overheads, multiplication for Paillier or addition for ElGamal is more efficiently realized using interactive protocols where $\mathcal{C}$ helps $\mathcal{S}$ to perform the respective operation. Using secure addition and multiplication, $\mathcal{C}$ and $\mathcal{S}$ can evaluate a representation of $\mathcal{F}$ as an arithmetic circuit. Then, $\mathcal{S}$ eval-

uates $\mathcal{F}$ on $\mathcal{C}$'s encrypted input $[x]$ and its own input y (square brackets denote encryption throughout this paper, i.e., $[x]$ is an encryption of x). $\mathcal{S}$ performs some operations locally, e.g., addition and scalar multiplication, while other operations, e.g., multiplication or comparison on ciphertexts, require interaction with $\mathcal{C}$. The overhead of HE-based STC is then due to interaction and public key operations.

2.1 Related Work

Different general purpose frameworks have been proposed, e.g., a port of the FastGC framework [19] in [18] and a port of the Fairplay framework [26] in [7], as well as protocols for specialized functionalities [4, 9]. While addressing mobile devices, these mostly assume a high-bandwidth network connection and consider processing and memory requirements as the main optimization goals: [25, 33] reduce the memory overhead of GC-based STC by more efficient circuit representations. To reduce processing overheads, [3,36] propose efficient garbling schemes, [5, 6] outsource GC from mobile devices to the cloud, and [10] use a hardware security token. In this, communication overhead as an optimization goal has received only passive or analytical attention. Notably, [4,6] only briefly argue that bandwidth efficiency is critical to minimize direct costs for users, energy consumption, and overall protocol runtime. Similar, [7, 37] observe that communication overhead can dominate the runtime of mobile STC but do not propose direct improvements. In this paper, we act on these observations by considering bandwidth consumption as the primary optimization goal.

3 Analysis of Efficient Secure Argmin Protocols

In this section, we analyze the problem of securely computing the minimum and its argument. We first introduce our problem definitions, security model, and parameters before we proceed to analyze the most efficient (arg)min protocols based on GCs §3.1, GMW §3.2, and HE §3.3.

Problem definition. Given a set of n arguments $X = (x_0, ..., x_{n-1})$ and n corresponding function values $Y = (y_0 = f(x_0), ..., y_{n-1} = f(x_{n-1}))$, the task is to find x^*, resp., $f(x^*)$ s.t. $f(x^*) \leq f(x_i)$ $\forall 1 \leq i \leq n$. We refer to x^* as *argmin* and to $y^* = f(x^*)$ as *min*. We assume that the inputs X and Y are already available in garbled (GC), secret-shared (GMW), or encrypted (HE) form and the output should be protected accordingly. This represents the usual case where the (arg)min algorithm is used as a building block within another secure computation, e.g., a nearest neighbor search which requires to derive certain distances before finding their argmin [17, 30]. In Appendix A, we briefly discuss a second version where $\mathcal{C}$ and $\mathcal{S}$ each hold half of the inputs and the (arg)min should be obtained in clear and only by $\mathcal{C}$.

Adversary model. We assume a semi-honest and computationally bounded adversary. A semi-honest adversary, other than the stronger malicious one, does not deviate from the protocol but may try to learn (private) information from

Protocol	Communication Overhead	Rounds
* Kolesnikov'09 [22] (GC)	$2l(n-1)2t + (n+1)2t$	$\mathcal{O}(1)$
Huang'11 [17] (GC)	$2l(n-1)2t + nlt + (n-1)2t$	$\mathcal{O}(1)$
Demmler'15 [11] (GC)	$3l(n-1)2t$	$\mathcal{O}(1)$
* Schneider'13 [31] (GMW)	$(n-1)(4l - \lceil \log_2(l) \rceil - 2 + \lceil \log_2(n) \rceil)(2t+2)$	$\mathcal{O}(\log_2(n) \log_2(l))$
Demmler'15 [11] (GMW)	$(n-1)(5l - \lceil \log_2(l) \rceil - 2)(2t+2)$	$\mathcal{O}(\log_2(n) \log_2(l))$
Erkin'09 [12] (HE)	$(n-1)(2l + 8 + 10/C)T$	$\mathcal{O}(\log_2(n))$
BOMA (HE+OT)	$(n-1)(4 + 6/C)T + 2nT/C + n(l+\sigma) + 3\log_2(n)t$	$\mathcal{O}(\log_2(n))$

Table 1: Communication complexity [bit] and rounds for the argmin problem in related work and in our improved protocol.

the protocol transcript. The semi-honest model, though more restrictive than the malicious, is widely used as it enables efficient secure computations and often serves as a stepping stone towards security against malicious adversaries.

Parameter definitions. We denote the symmetric security level by t and the asymmetric one by T and set $t = \{80, 112, 128\}, T = \{1024, 2048, 3072\}$ for legacy security until 2010, medium security until 2030, and long-term security beyond 2030, according to the NIST recommendations [2]. We set the statistical security parameter σ to 40 bit as, e.g., proposed in [11, 14, 17]. Finally, we vary the bitlength of the inputs $l \in \{32, 64, 128\}$ which represents a subset of frequent choices in the related literature [6, 10–12].

Overview. In the following, we analyze the most efficient (arg)min protocols for each of the three major STC concepts. The respective communication and round complexity is summarized in Table 1. Approaches marked with an asterisk restrict the argmin to $\{0, ..., n-1\}$. This restriction allows for efficiency improvements but does not fully meet our problem definition, as it would require further computation to realize the full range of applications. E.g., in biometric access control applications, the argmin is not only the index of a user but her full profile including access rights [17]. Nevertheless, we include them in our analysis since they indicate lower bounds. The last row in Table 1 shows the complexity for our improved protocol which we present and analyze in §4.

3.1 Garbled Circuits (GC)

The communication complexity of GC-based (arg)min protocols is dominated by i) the overhead for the input transfers (via OT), ii) the size of the garbled circuit, and iii) the chosen garbling scheme. We neglect all other minor communication overheads, e.g., for the establishment of a network connection. The overheads i) for transferring inputs only occur in the second version of our problem definition (cf. Appendix A). To quantify the overheads for ii), we analyze the most efficient circuit constructions and their respective sizes below. Regarding iii), we use the recent "Half Gates" garbling scheme [36] which allows to garble Non-XOR gates using only two wire keys, i.e., $2t$ bits. As proven in [36], "Half Gates" is currently optimal, i.e., its communication overhead of two keys per Non-XOR

gate constitutes a lower bound. The combination of the most efficient circuit construction with an optimal garbling scheme then yields state-of-the-art lower bounds on the communication complexity for GC-based (arg)min protocols.

Kolesnikov et al. present the most widely used (arg)min circuit construction in [22]. They select the min in a pairwise tournament tree fashion and construct the argmin while traversing down the tree. The circuit has a size of $2l(M-1)$ gates for finding the min and $n+1$ gates for constructing the argmin [22], resulting in a communication overhead of $(2l(n-1)+(n+1))2t$ bit. Notably, this construction limits the argmin to $\{0,...,n-1\}$.

Huang et al. [17] propose a different construction to overcome the limitation of the argmin value space, building on the observation that encoding complex data structures directly into the circuit is expensive. Hence, while using the same circuit of size $2l(n-1)$ for finding the min as [24], they replace the argmin functionality with a custom backtracking protocol. This protocol exchanges a fully encrypted backtracking tree from which exactly one path can be decrypted using the wire keys obtained during the evaluating of the garbled min circuit. In [17] $\mathcal{C}$ is allowed to recover the argmin in clear. However, to use their construction as a building block within another secure computation, the argmin must be garbled. This results in an overhead of nlt for encrypting n l-bit argmin values in the tree's leaves and $(n-1)2t$ bit for the two wire keys in each of the $n-1$ inner nodes of the backtracking tree.

Finally, Demmler et al. present the *ABY* framework for STCs [11] which implements the min circuit proposed in [22] with $2l(n-1)$ gates but does not supply the argmin. We trivially extend their implementation by adding l `MUX` gates per comparison which, analogous to the min, propagates the l bit argmin to obtain a second construction that fulfills our general problem definition (§3). A single 1 bit `MUX` gate can be realized using one Non-XOR gate, hence transmitting the complete circuit requires $3l(n-1)2t$ bit of communication.

3.2 Goldreich-Micali-Wigderson (GMW)

Schneider et al. present *depth-optimized* circuit constructions for GMW [31]. Their (arg)min circuit is based on the construction by Kolesnikov et al. [22] but replaces the size-optimized l-bit comparators with depth-optimized comparators that consists of about two times more gates but allows a logarithmic instead of linear depth in l. The circuit has $(n-1)(4l-\lceil\log_2(l)\rceil-2+\lceil\log_2(n)\rceil)$ gates and a depth of $\mathcal{O}(\log_2(n)\log_2(l))$ [31]. To evaluate a single gate, $\mathcal{C}$ and $\mathcal{S}$ exchange 2 bit and engage in one random OT at the costs of $2t$ bit communication overhead (§2, [11]). Since this construction directly bases on Kolesnikov's [22], it has the same limitations regarding the argmin value space.

The *ABY* framework presented by Demmler et al. [11] also implements GMW together with the depth-optimized minimum circuit presented in [31] but without the argmin logic. Again, we extend the implementation using `MUX` gates to relay the argmin along with the computation of the min. In contrast to Schneider's proposed circuit, this construction fulfills our problem definition. The circuit has $(n-1)(5l-\lceil\log_2(l)\rceil-2)$ gates and a depth of $\mathcal{O}(\log_2(n)\log_2(l))$.

3.3 Homomorphic Encryption (HE)

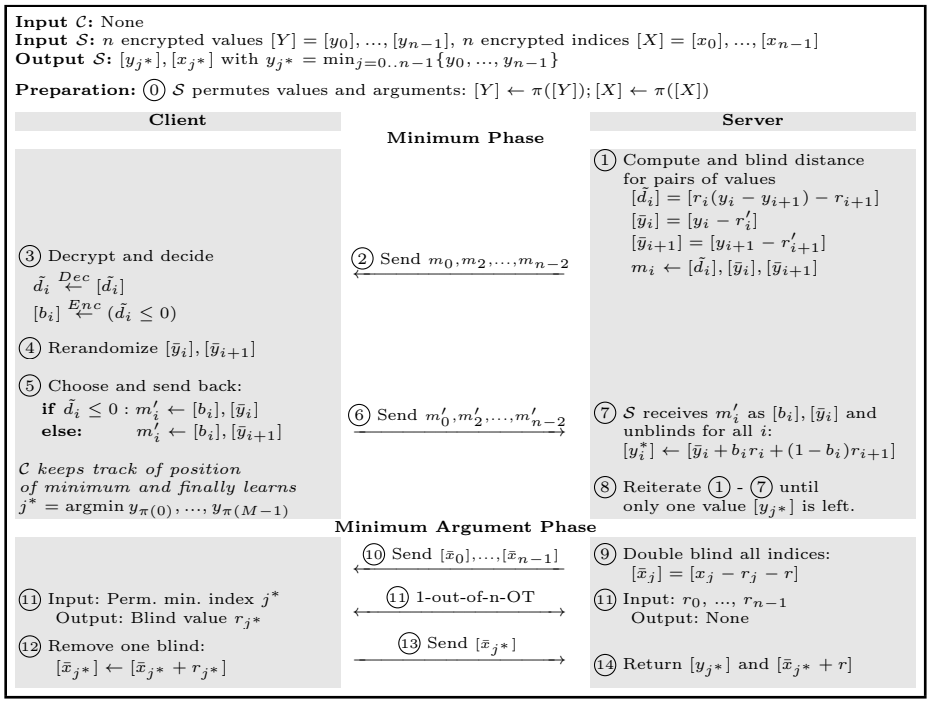
Most HE-based (arg)min constructions are based on the well-known DGK comparison protocol [8], which allows to compare two Paillier-encrypted integers $[a]$ and $[b]$ and obtain the result as an encrypted bit $[a \leq b]$. It has been used in a variety of applications, e.g., face recognition [12], recommender systems [14], bioinformatics [15], and clustering [13]. The most efficient construction that fulfills our problem definition has been proposed by Erkin et al. [12]. The authors arrange comparisons of values in a pairwise tournament fashion and propagate the min and argmin by multiplying with the encrypted comparison bit, i.e., $[\min(a,b)] = [a \leq b] * [a-b] + [b]$ and $[\text{argmin}(a,b)] = [a \leq b] * [id_a - id_b] + [id_b]$. Each DGK comparison needs to exchange $2l$ DGK ciphertexts and 3 Paillier ciphertexts. To propagate the min and argmin through one comparison, two ciphertext multiplications are required, causing transmission overheads of 6 Paillier ciphertexts [23]. This amounts to a communication complexity of $(n-1)(2l+8+10)T$ bit. While Erkin et al. [12] do not use ciphertext packing in their original work, applying packing to the Paillier ciphertexts exchanged from the server to the client reduces the communication complexity to $(n-1)(2l+8+10/C)T$ bit. Here, $C = \lfloor T/(l+\sigma+2)) \rfloor$ is the compression rate achieved by packing multiple ciphertexts into one, as detailed in [14].

4 Bandwidth-optimized Min and Argmin

To address the significant problems arising from high communication overheads in mobile STC, we propose BOMA, a bandwidth-optimized protocol for the secure (arg)min computation on homomorphically encrypted inputs. We emphasize that by using efficient conversions between encrypted, shared, and garbled values [11], BOMA can also be used as a building block to improve (mobile) STC frameworks, e.g., [11,19]. Following, we first present an efficient comparison protocol and based on this propose an efficient (arg)min protocol.

Efficient secure comparison: As an important building block, we use Kerschbaum's multi-party comparison protocol [21]. The core idea is to compute the encrypted distance $[d] = [a-b]$ between two encrypted inputs $[a]$ and $[b]$ and let all participants multiplicatively blind $[d]$ while preserving its sign before decrypting it and deciding the comparison. We adapt this multi-party protocol to the two-party setting as follows: $\mathcal{S}$ selects two large random numbers r_1 and $r_2 \in \{0,1\}^{l+\sigma}$ with $r_1 > r_2$ and computes $[\tilde{d}] = [r_1 \cdot (a-b) - r_2]$ under encryption. Note that the multiplicative blinding preserves the sign, i.e., $\tilde{d} \leq 0 \Leftrightarrow a \leq b$. $\mathcal{S}$ then sends the blinded encrypted distance $[\tilde{d}]$ to $\mathcal{C}$ who decrypts it and sends back the encrypted comparison bit $[a \leq b]$ to $\mathcal{S}$. To prevent $\mathcal{C}$ from learning the real outcome of the comparison, $\mathcal{S}$ chooses at random to compare $a \leq b$ or $b \leq a$ and flips the received comparison bit $[a \leq b]$ accordingly by computing $[1 - (a \leq b)]$ under encryption. Our two-party version of Kerschbaum's comparison exchanges only 2 ciphertexts, i.e., $4T$ bit, between $\mathcal{C}$ and $\mathcal{S}$.

Efficient secure (arg)min: As a first step, we replace the DGK comparison protocol in Erkin's (arg)min algorithm [12] with our two-party version of



Protocol 1: Bandwidth-optimized (arg)min protocol: The min and its permuted position is calculated. Then, $\mathcal{C}$ and $\mathcal{S}$ run an OT protocol to unblind the argmin.

Kerschbaum's comparison protocol. By simply using a more efficient comparison protocol, we reduce the communication complexity to $(M-1)(6+10/C)T$ bit, i.e., save $(M-1)(2l+2)T$ bit (cf. Table 1). This represents a significant reduction by 90 % to 97 % depending on the chosen security level t and bitlength l.

We now improve this construction with respect to both processing and communication overhead. In the first phase, the *minimum phase*, we determine the min $[y^*] = [f(x^*)]$ in pairwise comparisons arranged in a tournament fashion as before in [12]. However, we significantly reduce processing overheads and save $2\log_2(n)$ rounds by interleaving comparison and selection steps, thereby shaving off the costly ciphertext multiplications. Our second significant improvement is due to the observation that $\mathcal{C}$ learns the position of the min as a byproduct of this phase. With this information, we construct an efficient OT-based protocol for the second phase, the *minimum argument phase*, in which $\mathcal{C}$ helps $\mathcal{S}$ to obtain an encryption of the argmin $[x^*]$. In the following, we describe the two phases of our BOMA protocol, as depicted in Protocol 1, in detail.

Minimum phase: At the beginning, $\mathcal{S}$ holds the encrypted values $[y_0], ..., [y_{n-1}]$ and corresponding arguments $[x_0], ..., [x_{n-1}]$ and applies the same permutation π to both (0). This permutation prevents $\mathcal{C}$ from learning side knowledge, but has no effect on the outcome of the computation. For reasons of simplicity, we

thus leave π out in the following notation. At the core of ① - ⑦ are the batched pairwise comparisons according to our two-party version of Kerschbaum's comparison protocol. In ①, $\mathcal{S}$ computes the distances $[d_i]$ over the $n/2$ pairs of values $[y_i]$ and $[y_{i+1}], i = 0, 2, .., n-2$, and blinds the distances as well as the values. $\mathcal{S}$ then sends the blinded distances together with the blinded values $[y_i]$ and $[y_{i+1}]$ to $\mathcal{C}$ ②. $\mathcal{C}$ decrypts the distances and encrypts the binary result of the comparison ③, re-randomizes $[\bar{y}_i]$ and $[\bar{y}_{i+1}]$ ④, and chooses the smaller element ⑤. Finally, $\mathcal{C}$ sends back the binary result of this comparison together with only the smaller element of each comparison to $\mathcal{S}$ ⑥. Note that $\mathcal{S}$ cannot distinguish which elements were received due to the re-randomization and IND-CPA property of the cryptosystem. After unblinding the received values in ⑦, $\mathcal{S}$ now holds encryptions of the $n/2$ smaller values of the previous comparisons and repeats ① to ⑦ $\lceil \log_2(n) - 1 \rceil$ times until only the min $[y^*]$ remains ⑧. Our way of interleaving comparison and selection steps not only makes ciphertext packing more efficient, but significantly reduces processing costs and saves two communication rounds per level of the comparison tree, i.e., $2 \cdot \log_2(n)$ rounds in total, as compared to the construction of Erkin et al. [12] which implements selection steps via costly ciphertext multiplication.

Minimum argument phase: We further significantly improve the communication overhead by the observation that during the minimum phase it is easy for $\mathcal{C}$ to keep track of the position of the outcomes of the comparisons and thereby obtain the position j^* of the min in the *permuted* vector Y. This knowledge is not a security violation since the *permuted* position does not disclose any information to $\mathcal{C}$. In the minimum argument phase, $\mathcal{C}$ now helps $\mathcal{S}$ to obtain an encryption of the argmin $[x_{j^*}]$. $\mathcal{S}$ first blinds all encrypted arguments $[x_j], j = 1...M$ individually by subtracting random values $r_j \in \{0,1\}^{l+\sigma}$ and a second time with a single random value $r \in \{0,1\}^{l+\sigma}$ ⑨. $\mathcal{S}$ packs the double blinded arguments and sends them to $\mathcal{C}$ ⑩. $\mathcal{C}$ and $\mathcal{S}$ subsequently engage in 1-out-of-n-$\text{OT}^1_{l+\sigma}$ after which $\mathcal{C}$ obtains r_{j^*}, the distinct blind of the argmin, without $\mathcal{S}$ learning j^* ⑪. Then, $\mathcal{C}$ removes this blind from $[\bar{x}_{j^*}]$ by adding r_{j^*} (which automatically re-randomizes the value) ⑫ and sends the value (still blinded by r) to $\mathcal{S}$ ⑬. Finally, $\mathcal{S}$ removes the second blind by adding r and obtains an encryption of the argmin $[x_{j*}]$ ⑭.

Communication complexity. The minimum phase of BOMA costs only $(n-1)(4+6/C)T$ bit. By applying bandwidth-efficient OT protocols (§2), the minimum argument phase costs $n2T/C$ bit for transferring the blinded arguments and $n(l+\sigma) + 3\log_2(n)t$ bit for the 1-out-of-n-$\text{OT}^1_{l+\sigma}$ of the blind r_{j^*}.

4.1 Security Discussion

We show that our proposed protocol is secure in the semi-honest adversary model based on the security of the employed OT and comparison primitives. In particular, we show that i) $\mathcal{C}$ learns nothing and ii) $\mathcal{S}$ only learns an encryption of the min and argmin but nothing else.

Security against $\mathcal{C}$. From the messages received in ②, $\mathcal{C}$ learns nothing about the values y_i from $\bar{y}_i$ since they are additively blinded with random numbers $r'_i \in \{0,1\}^{l+\sigma}$. Furthermore, $\mathcal{C}$ learns nothing from $\tilde{d}_i$ about the distance d_i

between y_i and y_{i+1} due to the multiplicative and additive blinding. From the messages received in ⑩, $\mathcal{C}$ learns nothing since the arguments x_j are additively blinded using $r_j, r \in \{0,1\}^{l+\sigma}$. The security of ⑪ directly follows from the security proofs of the employed base OT protocol [1, 20, 27, 28]. After the OT has finished ⑪, $\mathcal{C}$ learns $x_j - r$ which however is still additively blinded. Finally, $\mathcal{C}$ learns the position of the minimum. However, due to the random permutation π applied by $\mathcal{S}$ in ⓪ this knowledge is useless to $\mathcal{C}$ as long as $\mathcal{S}$ keeps π secret.

Note that we use *statistical blinding*, i.e., with low probability $\sim 1/2^{l+\sigma}$ $\mathcal{C}$ learns a small amount of information about the magnitude of the blinded values. We can achieve perfect security against $\mathcal{C}$ by choosing $\sigma = T$ and substituting Kerschbaum's statistically secure protocol [21] with a perfectly secure protocol, e.g., [34]. However, this significantly increases the communication overhead.

Security against $\mathcal{S}$. From the messages received in ⑥, $\mathcal{S}$ learns the encrypted comparison bit and the encrypted smaller element of the comparison. Due to the IND-CPA property of the employed Paillier cryptosystem and the applied re-randomization, $\mathcal{S}$ can neither decide whether $[b_i]$ is an encryption of 0 or 1 nor distinguish $[\bar{y}_i]$ from $[\bar{y}_{i+1}]$. Again, $\mathcal{S}$ learns nothing from the OT in ⑪ due to the security of the employed OT primitives [1, 20, 27, 28]. Finally, $\mathcal{S}$ receives $[\bar{x}_{j^*}]$ which it cannot distinguish from the other arguments $[\bar{x}_{i \neq j^*}]$ due to the IND-CPA property of the cryptosystem. Since $\mathcal{S}$ can always try to break encryption to learn the inputs, we can only achieve computational security against $\mathcal{S}$.

5 Evaluation

We first compare the communication overhead of existing circuit and protocol constructions against an implementation of our optimized protocol BOMA in §5.1. Since BOMA trades increased local processing for a significant reduction in communication overhead, we evaluate processing overheads in §5.2 and show that BOMA achieves superior performance under constrained network speeds.

5.1 Quantitative Communication Overhead Analysis

Table 2 shows the communication overhead in MiB of each algorithm over increasing input lengths of $l = 32, 64, 128$ bit in each of the three security levels. We derive the results for Koleschnikov'09 [22], Huang'11 [17], and Schneider'13 [31] based on their theoretical complexities (cf. Table 1), since we could not obtain an actual implementation. Kolesnikov'09 and Schneider'13, the approaches marked with an asterisk, only realize the constrained argmin functionality and solely indicate lower bounds. For ABY-YAO'15 and ABY-GMW'15, we obtain the listed results using the C++ ABY framework [11], which we extended with the missing argmin circuits (cf. §3.2). We implement a prototype of BOMA (our own protocol), in Python 2.7. Additionally, we re-implement the (arg)min algorithm by Erkin et al. in our framework, since the available implementation in SeComLib [32] provides neither network support nor ciphertext packing.

n = 1000 Elements	short			medium			long		
	32	**64**	**128**	**32**	**64**	**128**	**32**	**64**	**128**
*** Koles'09**	1.24	2.46	4.90	1.73	3.44	6.86	1.98	3.93	7.84
(GC)	193%	370%	629%	154%	299%	554%	123%	240%	456%
Huang'11	1.54	3.07	6.12	2.16	4.30	8.56	2.47	4.91	9.79
(GC)	240%	462%	786%	192%	373%	692%	153%	299%	569%
ABY-YAO'15	1.87	3.73	7.45	2.61	5.21	10.41	2.98	5.95	11.90
(GC)	291%	562%	957%	232%	452%	840%	185%	363%	693%
*** Schneider'13**	2.53	4.98	9.90	3.53	6.94	13.81	4.03	7.93	15.76
(GMW)	394%	749%	1271%	314%	603%	1115%	250%	483%	917%
ABY-GMW'15	3.07	6.25	12.63	4.26	8.68	17.53	4.86	9.90	19.99
(GMW)	478%	941%	1622%	379%	754%	1416%	302%	603%	1163%
Erkin'09	9.37	17.60	34.13	18.14	34.18	66.31	26.92	50.77	98.49
(HE)	1460%	2650%	4383%	1614%	2967%	5355%	1672%	3094%	5730%
BOMA	0.64	0.66	0.78	1.12	1.15	1.24	1.61	1.64	1.72
(HE+OT)	100%	100%	100%	100%	100%	100%	100%	100%	100%

Table 2: Communication overhead [MiB] for varying security levels and input sizes. Gray rows denote theoretical estimates, all other values are measured. Approaches marked with an asterisk realize only a constrained argmin functionality.

For all available implementations, we initially compare the measured and theoretical communication overhead to analyze i) the accuracy of our theoretical complexity estimates and ii) the realization of these complexities in the actual implementations. We find that the measured overhead exceeds our estimate by at most 0.5 % for ABY-YAO, 1.5 % for ABY-GMW, 2 % for BOMA (our own protocol), and by less than 6 % for our re-implementation of Erkin's protocol. This deviation stems from the fact that our theoretical complexity estimates do not consider a decrease in packing efficiency when only few ciphertexts are left at the last levels of the comparison tree. Our way of interleaving comparison and selection (§4) greatly reduces this effect compared to Erkin's protocol design.

Table 2 then shows the communication overhead comparison of existing approaches and our (arg)min protocol (BOMA). We base our evaluation of existing approaches on the best available, i.e., most efficient, constructions of circuits, garbling schemes, and oblivious transfer primitives. BOMA achieves a significant reduction in communication overhead over all settings and even in comparison to the constrained argmin circuits by Kolesnikov'09 (GC) or Schneider'13 (GMW). Specifically, BOMA achieves the largest reductions for small security levels and high bitlengths of the input. With increasing security levels, the relative improvement, in comparison to GC- and GMW-based approaches that rely on symmetric crypto, decreases while still outperforming said approaches. Conversely, larger input bitsizes benefit BOMA. While packing efficiency for communication from $\mathcal{S}$ to $\mathcal{C}$ only degrades slightly, communication from $\mathcal{C}$ to $\mathcal{S}$, which cannot be packed, remains the same since a single l bit value always fits into one ciphertext for $l \leq T$. In contrast, the communication overhead in GC and GMW scales linearly with l. Finally, BOMA outperforms Erkin's HE-based protocol by one to two orders of magnitude in every setting.

n = 1000 Elements		short			medium			long		
		32	64	128	32	64	128	32	64	128
ABY-YAO	1 Mbit/s	16.70	33.50	67.02	23.35	46.71	93.64	26.71	53.46	107.21
	2 Mbit/s	8.44	16.90	33.75	11.70	23.54	46.85	13.48	26.81	53.51
	5 Mbit/s	3.38	6.92	13.86	4.81	9.68	19.21	5.32	10.91	21.78
	10 Mbit/s	1.76	3.72	7.19	2.56	5.13	9.92	2.81	5.68	11.17
ABY-GMW	1 Mbit/s	27.61	56.15	113.45	38.28	77.95	157.46	43.54	88.79	179.21
	2 Mbit/s	13.87	28.14	56.69	19.08	38.94	78.55	21.76	44.39	89.51
	5 Mbit/s	5.74	11.56	23.10	7.86	15.84	31.79	8.95	18.03	36.21
	10 Mbit/s	3.12	6.09	11.90	4.17	8.25	16.33	4.72	9.40	18.50
BOMA	1 Mbit/s	8.96	9.41	11.00	22.74	23.88	27.20	42.86	45.59	51.57
	2 Mbit/s	6.65	7.09	8.28	18.03	19.20	22.38	36.34	38.84	44.33
	5 Mbit/s	5.33	5.72	6.74	15.34	16.54	19.39	32.46	34.92	40.23
	10 Mbit/s	4.88	5.34	6.24	14.43	15.69	18.41	31.29	33.68	38.99

Table 3: Protocol runtimes [s] for varying security levels, input lengths and bandwidths for state-of-the-art GC-, GMW- and HE-based argmin protocols.

5.2 Performance Evaluation

We measure the runtime for ABY-YAO, ABY-GMW, Erkin'09, and BOMA for varying bandwidth and latency in a local setup between a desktop client (Intel i7, 8×2.93 GHz, 4 GB RAM) and a server (Intel Xeon, 16×2.6 GHz, 32 GB RAM) connected through a middlebox running OpenWRT. We choose a desktop instead of a mobile client to maintain comparability as no ABY implementation for Android or iOS exists. While ABY is fully threaded and thus employs all cores on the client device, we deliberately do not parallelize the client-side functionality in our BOMA implementation to emulate processing resources comparable to those of a mobile device, e.g., a smart phone. Since all overheads scale linearly in the number of elements n, we fix $n = 1000$. This is sufficiently large to eliminate small scale effects but also maintains short runtimes allowing for a repeated, thorough evaluation, i.e., averaging all results over 30 runs.

Network Bandwidth. We first vary the bandwidth between 1 Mbit/s and 10 Mbit/s using `netem` on the middlebox. Table 3 gives an overview of the resulting runtimes. For short-term security and at a bandwidth of 10 Mbit/s, BOMA performs in the same order of magnitude as ABY-YAO and ABY-GMW. Reducing to 1 Mbit/s, the runtime of BOMA doubles while the runtimes of ABY-YAO and ABY-GMW increase by roughly one order of magnitude. Communication overhead clearly dominates the runtime in these approaches and BOMA hence outperforms them. As indicated by the theoretical complexity, the approach by Erkin et al. [12] is by orders of magnitude slower. For $l = 32$ and short term security, the algorithm required almost 1 min, even already without bandwidth constraints. Due to this prohibitive runtime, we waived further measurements.

Increasing the security level to medium impacts BOMA more than ABY-YAO or ABY-GMW, due to the use of asymmetric crypto. Here, BOMA roughly matches the runtime of ABY-YAO for $l = 32, 64, 128$ at bandwidths between

1 Mbit/s and 5 Mbit/s and ABY-GMW between 2 Mbit/s and 10 Mbit/s. Still, we observe that BOMA outperforms these approaches in constrained networks (1 Mbit/s to 2 Mbit/s) and/or higher input sizes $l = 64, 128$.

Increasing to long term security, we observe that processing overheads begin to dominate the performance of BOMA, i.e., the relative difference of the performance at speeds of 10 Mbit/s and 1 Mbit/s is much smaller than for shorter security levels. Contrarily, for ABY-GMW and ABY-YAO, which exhibit very low processing overheads, bandwidth restrictions continue to dominate the overall protocol runtime. In comparison, BOMA still outperforms ABY-YAO up to bandwidths of 2 Mbit/s and ABY-GMW up to 5 Mbit/s for $l = 128$ bit inputs. For $l = 32, 64$ and larger bandwidths, BOMA's performance is slower but in general lies within the same order of magnitude as ABY-YAO and ABY-GMW.

Network Latency. Latency has the biggest impact on GMW-based protocols which have a high round complexity of $\mathcal{O}(\log_2(n) \log_2(l))$. Assuming, e.g., a relatively high latency of 200 ms, this adds 10 s to 14 s to the overall runtime for computing the argmin over $n = 1000$ and $l = 32, 128$ bit values, respectively. Under the same assumptions, BOMA's runtime increases only by approximately 2 s due to its lower round complexity of $\mathcal{O}(\log_2(n))$. GC-based protocols experience nearly no increase in runtime due to their constant round complexity. Hence, settings with higher network latencies favor BOMA over GMW-based approaches while the overhead compared to GC-based approaches is almost negligible.

In summary, the evaluation results support our initial design goals of supporting STC in mobile environments where network bandwidths are neither constant nor comparable to fixed, high-performance settings. Specifically, the improved performance of BOMA under the reduced network speeds of 3G or LTE networks, i.e., 1 Mbit/s – 10 Mbit/s as found in typical current urban scenarios, highlights the suitability of our protocol for spontaneous interaction and applications.

6 Conclusion

In this paper, we address the important problem of securely computing the (arg)min in a mobile ad-hoc setting where protocol participants had no prior interaction. Our analysis of the most efficient GC-, GMW-, and HE-based solutions reveals significant communication overheads that can be prohibitive in constrained, e.g., cellular, networks in terms of direct costs, energy consumption, and overall protocol runtime. We hence propose BOMA, a novel (arg)min protocol based on HE and OT that reduces communication overheads by orders of magnitude compared to related work (by 18 % – 98 %). Specifically, our approach trades communication overhead for local processing. Our quantitative evaluation shows a better performance by our protocol in many settings compared to state-of-the-art GC- and GMW-based solutions, e.g., in constrained networks operating at speeds below 1 Mbit/s to 10 Mbit/s, depending on the chosen security level and bitlength of the inputs. Using efficient conversion between encrypted, shared, and garbled values [11] our protocol can be used as a valuable building block for efficient mobile STCs in GC or GMW-based frameworks [11, 19].

Protocol	Communication Overhead	Rounds
* Kolesnikov'09 [22] (GC)	$2l(n-1)2t + (n+1)2t + {}^3/_2 nlt$	$\mathcal{O}(1)$
Huang'11 [17] (GC)	$2l(n-1)2t + n \cdot \max\{l,t\} + (n-1)2t + {}^3/_2 nlt$	$\mathcal{O}(1)$
Demmler'15 [11] (GC)	$3l(n-1)2t + 3nlt$	$\mathcal{O}(1)$
* Schneider'13 [31] (GMW)	$(n-1)(4l - \lceil \log_2(l) \rceil - 2 + \lceil \log_2(n) \rceil)(2t+2) + nl$	$\mathcal{O}(\log_2(n)\log_2(l))$
Demmler'15 [11] (GMW)	$(n-1)(5l - \lceil \log_2(l) \rceil - 2)(2t+2) + 2nl$	$\mathcal{O}(\log_2(n)\log_2(l))$
Erkin'09 [12] (HE)	$(n-1)(2l + 8 + 10/C)T + nT$	$\mathcal{O}(\log_2(n))$
BOMA (HE+OT)	$(n-1)(4 + 6/C)T + 2nT/C + n(l+\sigma) + 3\log_2(n)t + nt$	$\mathcal{O}(\log_2(n))$

Table 4: Communication complexity [bit] and rounds for the second problem definition, i.e., min and argmin computation with shared inputs.

A Min and Argmin with Shared Inputs

In §3, we define the secure min/argmin problem as a building block within another secure computation. This definition neglects those parts of the overheads which are due to sharing inputs between client and server. We thus shortly discuss a second version of the problem where $\mathcal{C}$ and $\mathcal{S}$ each hold half of the inputs and the client obtains the output.

For GCs, $\mathcal{S}$ sends its garbled inputs to $\mathcal{C}$, amounting to nlt bit of communication. $\mathcal{C}$ obtains its own inputs via correlated OT from $\mathcal{S}$ at a cost of $2nlt$ bit. These overheads are halved for Kolesnikov'09 [22] where the arguments are implicit and not part of the inputs as well as for Huang'11 where the arguments are encrypted in the backtracking tree. Sharing inputs in GMW-based approaches requires only $2nl$ bit, i.e., 1 bit per input bit. Again, this overhead is halved for the restricted (arg)min circuit of Schneider'13 [31]. For both Erkin's protocol and ours, only $\mathcal{C}$'s inputs need to be sent to $\mathcal{S}$ in encrypted form, requiring $2nT$ bit of communication. We summarize the overall complexity in Table 4.

We implement and evaluate this second (arg)min problem. As before, we observe an implementation overhead of at most 3 % compared to the complexities in Table 4. Only the measurements for ABY-YAO significantly deviate by 14 % and 27 % coupled with a large standard deviation in the send traffic. Since this renders the measurements incomparable, we present only a comparison of the theoretical communication overhead in Table 5. The results are qualitatively very similar to the results for our initial problem definition (Table 2 in §5.1). Furthermore, the processing required for sharing the inputs is very low in all approaches. Thus, for our second problem definition, we expect qualitatively very similar results to those presented in §5.2.

References

1. Asharov, G., Lindell, Y., Schneider, T., Zohner, M.: More Efficient Oblivious Transfer and Extensions for Faster Secure Computation. In: ACM CCS. ACM (2013)
2. Barker, E., Barker, W., Burr, W., Polk, W., Smid, M.: Nist special publication 800-57. NIST Special Publication 800(57), 1–142 (2007)

n = 1000 Elements	short			medium			long		
	32	**64**	**128**	**32**	**64**	**128**	**32**	**64**	**128**
*** Koles'09**	1.85	3.68	7.34	2.59	5.15	10.28	2.96	5.89	11.75
(GC)	225%	425%	792%	164%	320%	612%	127%	249%	483%
Huang'11	1.71	3.38	6.75	2.39	4.74	9.44	2.73	5.41	10.78
(GC)	207%	391%	727%	152%	294%	562%	117%	229%	443%
ABY-YAO'15	2.75	5.49	10.98	3.84	7.69	15.37	4.39	8.79	17.57
(GC)	334%	634%	1184%	244%	477%	915%	189%	372%	722%
*** Schneider'13**	2.53	4.99	9.93	3.53	6.96	13.84	4.03	7.94	15.79
(GMW)	308%	577%	1071%	224%	432%	824%	173%	336%	649%
ABY-GMW'15	2.96	6.03	12.20	4.13	8.41	17.01	4.71	9.60	19.42
(GMW)	360%	697%	1316%	262%	522%	1013%	202%	406%	798%
Erkin'09	9.12	16.97	32.64	18.14	33.79	65.08	27.16	50.62	97.52
(HE)	1108%	1960%	3520%	1152%	2096%	3875%	1167%	2141%	4008%
BOMA	0.82	0.87	0.93	1.58	1.61	1.68	2.33	2.36	2.43
(HE+OT)	100%	100%	100%	100%	100%	100%	100%	100%	100%

Table 5: Communication overhead [MiB] for varying security levels and input sizes. All numbers are theoretical estimates.

3. Bellare, M., Hoang, V.T., Keelveedhi, S., Rogaway, P.: Efficient garbling from a fixed-key blockcipher. In: IEEE SP. pp. 478–492. IEEE (2013)
4. Carter, H., Amrutkar, C., Dacosta, I., Traynor, P.: For your phone only: custom protocols for efficient secure function evaluation on mobile devices. SCN 7(7), 1165–1176 (2014)
5. Carter, H., Lever, C., Traynor, P.: Whitewash: Outsourcing garbled circuit generation for mobile devices. In: ACSAC. pp. 266–275. ACM (2014)
6. Carter, H., Mood, B., Traynor, P., Butler, K.: Secure Outsourced Garbled Circuit Evaluation for Mobile Devices. In: USENIX Security. USENIX (2013)
7. Costantino, G., Martinelli, F., Santi, P., Amoruso, D.: An implementation of secure two-party computation for smartphones with application to privacy-preserving interest-cast. In: PST. pp. 9–16 (2012)
8. Damgard, I., Geisler, M., Kroigard, M.: Homomorphic encryption and secure comparison. Int. J. Applied Cryptography 1(1), 22–31 (2008)
9. De Cristofaro, E., Faber, S., Gasti, P., Tsudik, G.: Genodroid: Are Privacy-preserving Genomic Tests Ready for Prime Time? In: ACM WPES. ACM (2012)
10. Demmler, D., Schneider, T., Zohner, M.: Ad-hoc secure two-party computation on mobile devices using hardware tokens. In: USENIX Security (2014)
11. Demmler, D., Schneider, T., Zohner, M.: ABY – A Framework for Efficient Mixed-Protocol Secure Two-Party Computation. In: NDSS (2015)
12. Erkin, Z., Franz, M., Guajardo, J., Katzenbeisser, S., Lagendijk, I., Toft, T.: Privacy-preserving face recognition. In: Goldberg, I., Atallah, M. (eds.) PETS, LNCS, vol. 5672, pp. 235–253. Springer (2009)
13. Erkin, Z., Veugen, T., Toft, T., Lagendijk, R.L.: Privacy-preserving user clustering in a social network. In: IEEE WIFS. pp. 96–100. IEEE (2009)
14. Erkin, Z., Veugen, T., Toft, T., Lagendijk, R.L.: Generating Private Recommendations Efficiently Using Homomorphic Encryption and Data Packing. IEEE Trans. Inf. Forensics Security 7(3), 1053–1066 (2012)
15. Franz, M., Deiseroth, B., Hamacher, K., Jha, S., Katzenbeisser, S., Schröder, H.: Towards secure bioinformatics services. In: Danezis, G. (ed.) Financial Cryptography and Data Security, LNCS, vol. 7035, pp. 276–283. Springer (2012)

16. Goldreich, O., Micali, S., Wigderson, A.: How to Play ANY Mental Game. In: ACM STOC. pp. 218–229. ACM (1987)
17. Huang, Y., Evans, D., Katz, J., Malka, L.: Efficient Privacy-Preserving Biometric Identification. In: NDSS (2011)
18. Huang, Y., Chapman, P., Evans, D.: Privacy-Preserving Applications on Smartphones. In: USENIX HotSec. USENIX (2011)
19. Huang, Y., Evans, D., Katz, J., Malka, L.: Faster Secure Two-party Computation Using Garbled Circuits. In: USENIX Security. USENIX (2011)
20. Ishai, Y., Kilian, J., Nissim, K., Petrank, E.: Extending oblivious transfers efficiently. In: Boneh, D. (ed.) CRYPTO 2003, LNCS, vol. 2729, pp. 145–161. Springer (2003)
21. Kerschbaum, F., Biswas, D., de Hoogh, S.: Performance comparison of secure comparison protocols. In: DEXA. pp. 133–136. IEEE (2009)
22. Kolesnikov, V., Sadeghi, A.R., Schneider, T.: Improved garbled circuit building blocks and applications to auctions and computing minima. In: Garay, J., Miyaji, A., Otsuka, A. (eds.) CANS 2009, LNCS, vol. 5888, pp. 1–20. Springer (2009)
23. Kolesnikov, V., Sadeghi, A.R., Schneider, T.: From Dust to Dawn: Practically Efficient Two-Party Secure Function Evaluation Protocols and their Modular Design. IACR Cryptology ePrint Archive (2010)
24. Kolesnikov, V., Schneider, T.: Improved garbled circuit: Free xor gates and applications. In: Aceto, L., Damgård, I., Goldberg, L., Halldórsson, M., Ingólfsdóttir, A., Walukiewicz, I. (eds.) Automata, Languages and Programming, LNCS, vol. 5126, pp. 486–498. Springer (2008)
25. Kreuter, B., Shelat, A., Mood, B., Butler, K.R.: PCF: A Portable Circuit Format for Scalable Two-Party Secure Computation. In: USENIX Security. USENIX (2013)
26. Malkhi, D., Nisan, N., Pinkas, B., Sella, Y.: Fairplay – a Secure Two-party Computation System. In: USENIX Security. USENIX (2004)
27. Naor, M., Pinkas, B.: Efficient Oblivious Transfer Protocols. In: SODA. pp. 448–457. SIAM (2001)
28. Naor, M., Pinkas, B.: Computationally Secure Oblivious Transfer. Journal of Cryptology 18(1), 1–35 (2005)
29. Paillier, P.: Public-key cryptosystems based on composite degree residuosity classes. In: Stern, J. (ed.) EUROCRYPT '99, LNCS, vol. 1592, pp. 223–238. Springer (1999)
30. Rane, S., Boufounos, P.: Privacy-Preserving Nearest Neighbor Methods. IEEE Signal Process. Mag. 30(2) (2013)
31. Schneider, T., Zohner, M.: Gmw vs. yao? efficient secure two-party computation with low depth circuits. In: Sadeghi, A.R. (ed.) Financial Cryptography and Data Security, LNCS, vol. 7859, pp. 275–292. Springer (2013)
32. Secomlib, `http://cybersecurity.tudelft.nl/content/secomlib`
33. Songhori, E.M., Hussain, S.U., Sadeghi, A.R., Schneider, T., Koushanfar, F.: TinyGarble: Highly Compressed and Scalable Sequential Garbled Circuits. In: IEEE SP. IEEE (2015)
34. Veugen, T.: Improving the dgk comparison protocol. In: IEEE WIFS. IEEE (2012)
35. Yao, A.: How to generate and exchange secrets. In: FOCS. pp. 62–167. IEEE (1986)
36. Zahur, S., Rosulek, M., Evans, D.: Two halves make a whole. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015, LNCS, vol. 9057, pp. 220–250. Springer (2015)
37. Ziegeldorf, J.H., Metzke, J., Henze, M., Wehrle, K.: Choose Wisely: A Comparison of Secure Two-Party Computation Frameworks. In: IEEE SPW. IEEE (2015)