New Protocols for Secure Equality Test and Comparison

Geoffroy Couteau(✉)

Karsruhe Institute of Technology, Karlsruhe, Germany
geoffroy.couteau@kit.edu

Abstract. Protocols for securely comparing private values are among the most fundamental building blocks of multiparty computation. introduced by Yao under the name millionaire's problem, they have found numerous applications in a variety of privacy-preserving protocols; however, due to their inherent non-arithmetic structure, existing construction often remain an important bottleneck in large-scale secure protocols.

In this work, we introduce new protocols for securely computing the greater-than and the equality predicate between two parties. Our protocols rely solely on the existence of oblivious transfer, and are UC-secure against passive adversaries. Furthermore, our protocols are well suited for use in large-scale secure computation protocols, where secure comparisons (SC) and equality tests (ET) are commonly used as basic routines: they perform particularly well in an amortized setting, and can be preprocessed efficiently (they enjoy an extremely efficient, information-theoretic online phase). We perform a detailed comparison of our protocols to the state of the art, showing that they improve over the most practical existing solutions regarding both communication and computation, while matching the asymptotic efficiency of the best theoretical constructions.

Keywords: Two-party computation · Equality test
Secure comparison · Oblivious transfer

1 Introduction

Multiparty Computation (MPC) addresses the challenge of performing computation over sensitive data without compromising its privacy. In the past decades, several general-purpose solutions to this problem have been designed, starting with the seminal works of Yao [53] and Goldreich et al. [27]. Among the large variety of problems related to MPC that have been considered, the *secure comparison* problem, in which the players wish to find out whether $x \geq y$ for given x, y without disclosing them, is probably the one that received the most attention. Indeed, in addition to being the first MPC problem ever considered (introduced

Part of this work was made while the author was at École Normale Supérieure de Paris, France.

B. Preneel and F. Vercauteren (Eds.): ACNS 2018, LNCS 10892, pp. 303–320, 2018.
https://doi.org/10.1007/978-3-319-93387-0_16

in [53] under the name of millionaire's problem), it has proven to be a fundamental primitive in a considerable number of important applications of multiparty computation. Examples include auctions, signal processing, database queries, machine learning and statistical analysis, biometric authentication, combinatorial problems, or computation on rational numbers. Secure comparison is at the heart of any task involving sorting data, finding a minimum value, solving any optimization problem, or even in tasks as basic as evaluating the predicate of a while loop, among countless other examples. The related task of *secure equality test*, known as the socialist millionaires' problem, in which the players wish to find out whether $x = y$ for given x, y without disclosing them, enjoys comparably many applications.

Two-party and multiparty computation seem now at the edge of becoming practical, with increasing evidence that they are no more beyond the reach of the computational power of today's computers. However, secure equality tests and comparisons appear to be a major bottleneck in secure algorithms that use them as a basic routines. Various implementations of secure algorithms unanimously lead to the conclusion that secure comparison is the most computationally involved primitive, being up to two orders of magnitude slower than, e.g., secure multiplication. Hence, we believe that designing improved protocols for these tasks is an important road toward making multiparty computation truly practical.

In this work, we consider secure equality test and comparison on inputs secretely shared between the parties, with output shared between the parties as well. This is the natural setting of large-scale computation, where inputs and outputs cannot always be disclosed to the parties. Our new two-party protocols compare very favorably to state-of-the-art solutions. In particular, our protocols are well suited for large scale secure computation protocols using secure comparison as a basic routine. Our protocols are secure in the universal composability framework of Canetti [11], which ensures that security is preserved under general composition. As this is the model used in most practical applications, we focus on the passive adversarial model, in which players are assumed to follow the specifications of the protocol. We leave as open the interesting question of extending our protocols to handle malicious adversaries, while preserving (as much as possible) their efficiency.

1.1 State of the Art for Secure Equality Test and Comparison

To avoid unnecessary details in the presentation, we assume some basic knowledge on classical cryptographic primitives, such as garbled circuits, oblivious transfers and cryptosystems. Preliminaries on oblivious transfers are given in the full version of this work [15]. In the following, we let ℓ denote an input length, and κ denote a security parameter. As secure protocols for equality tests and comparisons were commonly built together in the literature, the state of the art for both remains essentially the same, hence we unify the presentation.

– *From Garbled Circuits.* The first category regroups protocols following the garbled circuit approach of Yao [53]. The protocols of [37], which were later

improved in [36,55], are amongst the most communication-efficient protocols for secure equality test or comparison. The protocols of [36] proceed by letting the first player garble a circuit containing ℓ comparison gates (resp. $\ell - 1$ equality test gates), which amounts to ℓ AND gates with the free-xor trick (resp. $\ell - 1$ AND gates). In a setting where several instances of the protocols will be invoked, oblivious transfer extensions [33] can be used for an arbitrary number of executions, using a constant number of public key operations and only cheap symmetric operations for each invocation of the secure protocol, making them very efficient.

– *From Homomorphic Encryption.* Solutions to the millionaire problem from homomorphic-encryption originated in [7]. The most efficient method in this category, to our knowledge, is [20], which uses an ad hoc cryptosystem. This protocol was corrected in [21], and improved in [51]. The protocol communicates 4ℓ ciphertexts (in the version that outputs shares of the result) and is often regarded as one of the most computationally efficient. The more recent construction of [25] relies on the flexibility of lattice based cryptosystems to design a secure comparison protocol. Using a degree-8 somewhat homomorphic encryption scheme and ciphertext packing techniques, the (amortized) bit complexity of their protocol is $\tilde{O}(\ell+\kappa)$. Although asymptotically efficient, this method is expected to remain less efficient than alternative methods using simpler primitives for any realistic parameters.
– *From the Arithmetic Black Box Model.* The third category consists of protocols built on top of an arithmetic black box [17] (ABB), which is an ideal reactive functionality for performing securely basic operations (such as additions and multiplications) over secret values loaded in the ABB. The ABB itself can be implemented from various primitives, such as oblivious transfer [23,45] or additively homomorphic encryption (most articles advocate the Paillier scheme [44]). Protocols in this category vary greatly in structure. Most protocols [12,19,43] involve $\tilde{O}(\ell)$ private multiplications, each typically requiring $O(1)$ operations over a field of size $O(\ell + \kappa)$, resulting in an overall $\tilde{O}(\ell(\ell + \kappa))$ bit complexity. The protocols of Toft [50], and Toft and Lipmaa [40], use only a sublinear (in ℓ) number of invocations to the cryptographic primitive; however, the total bit complexity remains superlinear in ℓ. For large values of ℓ ($\kappa^2/\ell = o(1)$), the protocol of [54] enjoys an optimal $O(\ell)$ communication complexity; however, the constants involved are quite large: it reduces to $84\lambda + 96$ bit oblivious transfer and 6 ℓ-bit secure multiplications for a $1/2^\lambda$ error probability, and becomes competitive with e.g. [36] only for inputs of at least 500 bits (assuming a $1/2^{40}$ error probability).
– *From Generic Two-Party Computation.* Generic two-party computation (2PC) techniques can be used to securely compute functions represented as boolean circuits. An elegant logarithmic-depth boolean circuit, computing simultaneously the greater-than and the equality predicates, was suggested in [24]. It uses a natural recursive formula, and has $3\ell - \log \ell - 2$ AND gates. This circuit can be evaluated using $6\ell - 2\log \ell - 4$ oblivious transfers on bits, which can be precomputed and amortized using oblivious transfer extensions. In the amortized setting, we found this approach to be (by far) the most

efficient in terms of communication and computation; however, it is more interactive than the garbled circuit approach, which still enjoy efficient communication and computation.

In this paper, we will compare our protocols to the two most efficient alternatives in the amortized setting, namely, the garbled circuit approach, and the generic 2PC approach (which is more interactive, but has lower communication and computation). For fairness of the comparison, we will apply all optimizations that we apply to our protocols to these alternatives, when it is relevant.

1.2 Our Contribution

In this work, we construct new protocols for secure equality tests and comparisons which improve over the best state-of-the-art protocols. Our protocols are secure in the universal composability framework, assuming only an oblivious transfer. Using oblivious transfer extensions allows to confine all public-key operations to a one-time setup phase. The online phase of our protocols enjoys information theoretic security, and is optimal regarding both communication and computation: $O(\ell)$ bits are communicated, and $O(\ell)$ binary operations are performed, with small constants. Regarding overall complexity, our protocols match the best existing constructions in terms of asymptotic efficiency (and have in particular an optimal $O(\ell)$ complexity for large values of ℓ, see Table 1), and outperform the most efficient constructions for practical parameters, by 70% to 80% for equality test, and by 20% to 40% for secure comparison. Our protocols have non-constant round complexity: $O(\log^* \kappa)$ rounds for equality test (2 to 4 online rounds in practice), and $O(\log \log \ell)$ rounds for comparison (2 to 10 online rounds). Our secure comparison protocol relies on a new technique to (non-interactively) reduce comparison of values shared between the players to comparison of values held by each players, which might be of independent interest. Due to space restriction, we only focus on our new protocols for equality tests here; our protocols for secure comparison are described in the full version of this work [15].

Further Contributions of the Full Version. In addition to detailed security proofs, the full version of our work [15] contains further contributions, including a new simple method which reduces by 25% the communication of the Naor-Pinkas oblivious transfer protocol [41] when the size of the transmitted strings is lower than $\kappa/2$, and a variant of our equality test protocol in a batch settings (where many equality tests are performed "by blocks"), which uses additively homomorphic encryption to further improve the communication of our equality test protocol by up to 50%.

1.3 Our Method

The high level intuition of our approach is an observation that was already made in previous works [40,50]: to compare two strings, it suffices to divide

them in equal length blocks, and compare the first block on which they differ. Therefore, a protocol for (obliviously) finding this block can be used to reduce the secure comparison problem on large strings to the secure comparison problem on smaller strings. One can then recursively apply this size-reduction protocol, until the strings to be compared are small enough, and compute the final result using a second protocol tailored to secure comparison on small strings. However, this intuition was typically implemented in previous work using heavy public-key primitives, such as homomorphic encryption. In this work, we show how this strategy can be implemented using exclusively oblivious transfers on small strings.

To implement the size-reduction protocol, we rely on a protocol to obliviously determine whether two strings are equal. Therefore, a first step toward realizing a secure comparison protocol is to design a protocol for testing equality between two strings, which outputs shares (modulo 2) of a bit which is 1 if and only if the strings are equal. Keeping this approach in mind, we start by designing an equality test protocol which is based solely on oblivious transfer. Recall that in an oblivious transfer protocol, one party (the sender) inputs a pair (s_0, s_1), while the other party (the receiver) inputs a bit b; the receiver receives s_b as output and learns nothing about s_{1-b}, while the sender learns nothing about b. Our protocol relies on a classical observation: two strings are equal if and only if their Hamming distance is zero. More specifically, our protocols proceed as follows:

Equality Test. Consider two inputs (x, y), of length ℓ. We denote $(x_i, y_i)_{i \leq \ell}$ their bits. The parties execute ℓ parallel oblivious transfers over $\mathbb{Z}_{\ell+1}$, where the first player input pairs $(a_i + x_i \bmod \ell + 1, a_i + 1 - x_i \bmod \ell + 1)$ (a_i is a random mask over $\mathbb{Z}_{\ell+1}$), and the second party input his secret bits y_i; let b_i be his output ($b_i = a_i + x_i \oplus y_i \bmod \ell + 1$, where $\oplus$ is the exclusive or). Observe that $x' \leftarrow \sum_i a_i \bmod \ell + 1$ and $y' \leftarrow \sum_i b_i \bmod \ell + 1$ are equal if and only if the Hamming distance between x and y is 0, if and only if $x = y$. Note that (x', y') are of length $\log(\ell + 1)$.

The players repeatedly invoke the above method, starting from (x', y'), to shrink the input size while preserving equality, until they end up with string of length at most (say) 3 bits (it takes about $O(\log^* \ell)$ invocations of the protocol, where the first invocation dominates the communication cost). The players then perform a straightforward equality test on these small strings, using oblivious transfers to evaluate an explicit exponential-size formula for equality checking on the small entries.

The core feature of this compression method is that it can be almost entirely preprocessed: by executing the compression protocol on random inputs (r, s) in a preprocessing phase (and storing the masks generated), the players can reconstruct the output of the protocol on input (x, y) simply by exchanging $x \oplus r$ and $s \oplus y$ in the online phase. Therefore, the communication of the entire equality test protocol can be made as low as a few dozens to a few hundreds of bits in the online phase. Furthermore, in the preprocessing phase, the protocol

involves only oblivious transfers on very small entries (each entry has size at most $\log \ell$ bits), for which particularly efficient constructions exist [35].

Secure Comparison. We now describe our solution to the secure comparison problem. This protocol has a structure somewhat comparable to the previous one, but is more involved. The parties break their inputs (x, y) in $\sqrt{\ell}$ blocks of length $\sqrt{\ell}$ each. In the first part of the protocol, the parties will construct $\sqrt{\ell}$ shares of bits, which are all equal to 0 except for the ith bit, where i is the index of the first block on which x differs from y. This step relies on parallel invocations to the equality test functionality, and on oblivious transfers. Then, using these bit-shares and oblivious transfers, the players compute shares of the first block on which x differs from y.

At this point, we cannot directly repeat the above method recursively, as this method takes inputs *known to the parties*, while the output values are only shared between the parties. However, under a condition on the size of the group on which the shares are computed, we prove a lemma which shows that the parties can *non-interactively* reduce the problem of securely comparing shared value to the problem of securely comparing known values, using only local computations on their shares. From that point, the parties can apply the compression protocol again (for $O(\log \log \ell)$ rounds), until they obtain very small values, and use (similarly as before) a straightforward protocol based on an explicit exponential-size formula for comparison. Alternatively, to reduce the interactivity, the compression protocol can be executed a fixed (constant) number of times, before applying, e.g., a garbled-circuit-based protocol or a generic $\mathsf{2PC}$ protocol on the reduced-size inputs.

This protocol involves $O(\sqrt{\ell})$ equality tests and oblivious transfers on small strings, both of which can be efficiently preprocessed. This leads to a secure comparison protocol that communicates about a thousand bits in the online phase, for 64-bit inputs.

1.4 Comparison with Existing Works

For Secure Comparisons. We provide Table 1 a detailed comparison between the state of the art, our logarithmic-round protocol SC_1, and its constant-round variants SC_2 and SC_3. We evaluate efficiency in an amortized setting and ignore one-time setup costs. We considered two methods based on garbled circuit, the protocol of [36] and the same protocol enhanced with the method of [3] to optimize the online communication. We also considered the solution based on the DGK cryptosystem [20,21,51], the protocol of [40], the probabilistically correct protocol of [54], and generic $\mathsf{2PC}$ applied to the protocol of [24]. Note that [40,54] are described with respect to an arithmetic black box, hence their cost depends on how the ABB is implemented. For [40], which requires an ABB over large order fields, we considered a Paillier based instantiation, as advocated by the authors. For [54], which involves (mainly) an ABB over $\mathbb{F}_2$, we considered the same optimizations than in our protocols, implementing the ABB with oblivious transfers on bits.

As illustrated in Table 1, our protocols improve over existing protocols (asymptotically) regarding both communication and computation. This comes at the cost of a non-constant $O(\log\log\ell)$ interactivity (or $O(c\cdot\log^*\kappa)$ in the constant-round setting). In particular, for large values of ℓ (and for any value of ℓ in the online phase), our protocols enjoy an optimal $O(\ell)$ communication and computation complexity. The hidden constants are small, making our protocols more efficient than the state of the art for any practical parameter. For values of ℓ between 4 and 128, the protocols of [24,36] (which enjoy tiny constants) outperforms all other existing protocols regarding communication and computation. We therefore focus on these protocols as a basis for comparison with our protocols in our concrete efficiency estimations.

Equality Tests. The state of the art given Table 1 remains essentially the same for equality tests. Indeed, all the papers listed in the table (at the exception of [20], but including the present paper) do also construct equality tests protocols, with the same (asymptotic) complexity and from the same assumptions. The only difference in asymptotic complexity between our equality test protocol and the protocol SC_1 is with respect to the round complexity: while SC_1 has $O(\log\log\ell)$ rounds, our equality test protocol has an almost-constant number of rounds $O(\log^*\kappa)$. Note that we consider only equality tests whose output is shared between the players (as this is necessary for our secure comparison protocol); if the players get to learn the output in the clear (this is known as the socialist millionaires problem), more efficient solutions exist, but there is no simple way of designing equality tests with shared outputs from these solutions.

1.5 Applications

Equality test protocols enjoy many applications as building blocks in various multiparty computation protocols. Examples include, but are not limited to, protocols for switching between encryption schemes [16], secure linear algebra [18], secure pattern matching [31], and secure evaluation of linear programs [49]. Secure comparisons have found a tremendous number of applications in cryptography; we provide thereafter a non-exhaustive list of applications for which our protocols lead to increased efficiency. We note that in applications for which implementations have been described, the communication of secure comparisons was generally pointed out as the main efficiency bottleneck.

- *Obliviously sorting data* [28,29] has proven useful in contexts such as private auctions [42], oblivious RAM [26], or private set intersection [32], but it remains to date quite slow (in [30], sorting over a million 32-bit words takes between 5 and 20 min). All existing methods crucially rely on secure comparisons and require at least $O(m\log m)$ secure comparisons in $O(\log m)$ rounds to sort lists of size m.
- *Biometric authentication*, while solving issues related to the use of passwords, raises concerns regarding the privacy of individuals, and received a lot of attention from the cryptographic community. Protocols for tasks such as

Table 1. Amortized costs of state of the art secure comparison

Protocol	[36]	[20,21,51][a]	[40][a]	[36]+[3][a]	[54]
Preprocessing phase					
Communication	$O(\kappa\ell)$	–	$O(n\kappa \log \ell)$	$O(n\ell)$	$O(\frac{\kappa^2}{\log \kappa} + \ell)$
Computation	$O(\kappa\ell)$	$O(\ell(\kappa+\ell) \cdot C_n)$	$O(n\kappa \log \ell \cdot C_n)$	$O(n\ell \cdot C_n)$	$O(\frac{\kappa^2}{\log \kappa} + \ell)$
Rounds	1	–	$O(1)$	1	$O(1)$
Assumption	OT	–	ABB	RSA	ABB
Online phase					
Communication	$O(\kappa\ell)$	$O(n\ell)$	$O(n \log \ell)$	$O(\ell + n)$	$O(\kappa + \ell)$
Computation	$O(\kappa\ell)$	$O(\ell \log \ell \cdot C_n)$	$O(n \log \ell \cdot C_n)$	$O(\kappa\ell + n \cdot C_n)$	$O(\kappa + \ell)$
Rounds	2	2	$O(\log \ell)$	2	$O(\log \kappa)$
Assumption	OWF	DGK	ABB	RSA	None
Protocol	[24]	SC_1		$\mathsf{SC}_2, \mathsf{SC}_3$ (c is some fixed constant)	
Preprocessing phase					
Communication	$O(\frac{\kappa\ell}{\log \kappa})$	$O(\frac{\kappa\ell}{\log \kappa})$ if $\ell = o(\kappa^2)$ $O(\ell)$ else		$O(\frac{\kappa\ell}{\log \kappa})$ if $\ell^{1-1/c} = o(\kappa^2)$ $O(\ell)$ else	
Computation	$O(\frac{\kappa\ell}{\log \kappa})$	$O(\frac{\kappa\ell}{\log \kappa})$ if $\ell = o(\kappa^2)$ $O(\ell)$ else		$O(\frac{\kappa\ell}{\log \kappa})$ if $\ell^{1-1/c} = o(\kappa^2)$ $O(\ell)$ else	
Rounds	$O(\log \ell)$	$O(\log \log \ell)$		$O(c \log^* \kappa)$	
Assumption	OT	OT		OT	
Online phase					
Communication	$O(\ell)$	$O(\ell)$		$O(\ell)$	
Computation	$O(\ell)$	$O(\ell)$		$O(\ell)$	
Rounds	$O(\log \ell)$	$O(\log \log \ell)$		$O(c \log^* \kappa)$	
Assumption	None	None		OWF (SC_2) or none (SC_3)	

[a] $n > \ell + \kappa$ is the length of an RSA modulus. C_n denotes the cost of a modular multiplication modulo n. Note that [3] can also be instantiated from the DDH or the LWE assumption.

secure face recognition [47] require finding the minimum value in a database, which reduces to $O(m)$ secure comparisons in $O(\log m)$ rounds.

- Secure protocols for *machine learning* employ secure comparisons as a basic routine for tasks such as classification [10], generating private recommendations [22], spam classification [52], multimedia analysis [14], clinical decisions [46], evaluation of disease risk [5], or image feature extraction [39].
- Secure algorithms for *combinatorial problems*, such as finding the flow of maximum capacity in a weighted graph, or searching for the shortest path between two nodes, have been investigated in several works, e.g. [38], and have applications in protocols such as private fingerprint matching [8], privacy-preserving GPS guidance, or privacy-preserving determination of topological features in social networks [2]. They typically involve a very large number of secure comparisons (e.g. n^2 comparisons for Dijkstra's shortest path algorithm on an n-node graph [2]).
- Other applications that heavily rely on comparisons include computing on non integer values [1], various types of secure auctions [20], range queries over encrypted databases [48], or algorithms for optimization problems [13,49].

1.6 Organization

In Sect. 2, we recall definitions and classical results on oblivious transfers, as well as on oblivious transfer extensions. Section 3 introduces our new equality test protocol, and constitutes the main body of our work. Due to space constraints, we postpone our protocols for secure comparisons, as well as our detailed security proofs, to the full version [15]; we note that most of the security proofs are quite standard.

1.7 Notations

Given a finite set S, the notation $x \leftarrow_R S$ means that x is picked uniformly at random from S. For an integer n, $\mathbb{Z}_n$ denotes the set of integers modulo n. Throughout this paper, $+$ will always denote addition over the integers, and not modular additions. We use bold letters to denote vectors. For a vector $\boldsymbol{x}$, we denote by $\boldsymbol{x}[i]$ its i'th coordinate; we identify k-bit-strings to vectors of $\mathbb{Z}_2^k$ (but do not use bold notations for them). We denote by $\boldsymbol{x} * \boldsymbol{y}$ the Hadamard product $(\boldsymbol{x}[i] \cdot \boldsymbol{y}[i])_i$ between $\boldsymbol{x}$ and $\boldsymbol{y}$. Let $\oplus$ denote the xor operation (when applied on bit-strings, it denotes the bitwise xor). For integers (x, y), $[x = y]$, $[x < y]$, and $[x \leq y]$ denote a bit which is 1 if the equality/inequality holds, and 0 otherwise. The notation $(x \bmod k)$, between parenthesis, indicates that $x \bmod t$ is seen as an integer between 0 and $t-1$, not as an element of $\mathbb{Z}_t$. For an integer k, let $\langle\cdot\rangle_k$ denote the randomized function that, on input x, returns two uniformly random shares of x over $\mathbb{Z}_k$ (i.e., a random pair $(a, b) \in \mathbb{Z}_k$ such that $a + b = x \bmod k$). We extend this notation to vectors in a natural way: for an integer vector $\boldsymbol{x}$, $(\boldsymbol{a}, \boldsymbol{b}) \leftarrow_R \langle \boldsymbol{x} \rangle_k$ denote the two vectors obtained by applying $\langle\cdot\rangle_k$ to the coordinates of $\boldsymbol{x}$. Finally, for an integer x, we denote by $|x|$ the bit-size of x.

2 Oblivious Transfer

Oblivious transfers (OT) were introduced in [45]. An oblivious transfer is a two-party protocol between a sender and a receiver, where the sender obliviously transfers one of two string to the receiver, according to the selection bit of the latter. The ideal functionality for k oblivious transfers on l-bit strings is specified as follows:

$$\mathscr{F}_{\mathsf{OT}}^{k,l} : ((\boldsymbol{s}_0, \boldsymbol{s}_1), x) \mapsto \left(\perp, \left(\boldsymbol{s}_{x[i]}[i]\right)_{i \leq k}\right)$$

where $(\boldsymbol{s}_0, \boldsymbol{s}_1) \in (\mathbb{F}_2^l)^k \times (\mathbb{F}_2^l)^k$ is the input of the sender, and $x \in \mathbb{F}_2^k$ is the input of the receiver. In a *random oblivious transfer* (ROT), the input of the sender is picked at random:

$$\mathscr{F}_{\mathsf{ROT}}^{k,l} : (\perp, x) \mapsto \left((\boldsymbol{s}_0, \boldsymbol{s}_1), \left(\boldsymbol{s}_{x[i]}[i]\right)_{i \leq k}\right)$$

The primitive can be extended naturally to k-out-of-n oblivious transfers; we let $\binom{n}{k}$-OT_ℓ^t denote t invocations of a k-out-of-n OT on strings of length ℓ. Oblivious transfer is a fundamental primitive in MPC as it implies general multiparty computation [34] and can be made very efficient.

2.1 Oblivious Transfer Extension

Although oblivious transfer requires public-key cryptographic primitives, which can be expensive, *oblivious transfer extension* allows to execute an arbitrary number of oblivious transfers, using only cheap, symmetric operations, and a small number of base OTs. OT extensions were introduced in [6]. The first truly practical OT extension protocol was introduced in [33], assuming the random oracle model.[1] We briefly recall the intuition of the OT extension protocol of [33]. A $\binom{2}{1}$-OT_t^κ can be directly obtained from a $\binom{2}{1}$-$\mathsf{OT}_\kappa^\kappa$: the sender associates two κ-bit keys to each pair of messages and obliviously transfer one key of each pair to the receiver. Then, the receiver stretches two t-bit strings from the two keys of each pair, using a pseudo-random generator, and sends the xor of each of these strings and the corresponding message to the receiver. The $\binom{2}{1}$-OT_ℓ^t itself can be implemented with a single call to a $\binom{2}{1}$-OT_t^κ functionality, in which the receiver plays the role of the sender (and reciprocally). The total communication of the reduction from $\binom{2}{1}$-OT_ℓ^t to $\binom{2}{1}$-$\mathsf{OT}_\kappa^\kappa$ is $2t\ell + 2t\kappa$ bits. Regarding the computational complexity, once the base OTs have been performed, each OT essentially consists in three evaluations of a hash function. An optimization to the protocol of [33] was proposed in [4] (and discovered independently in [35]). It reduces the communication of the OT extension protocol from $2t\ell + 2t\kappa$ bits to $2t\ell + t\kappa$ bits, and allows to perform the base OTs without an a-priori bound on the number of OTs to be performed later (the OTs can be continuously extended).

Oblivious Transfer of Short Strings. An optimized OT extension protocol for short strings was introduced in [35], where the authors describe a reduction of $\binom{2}{1}$-OT_ℓ^t to $\binom{2}{1}$-$\mathsf{OT}_\kappa^\kappa$ with $t(2\kappa/\log n + n \cdot \ell)$ bits of communication, n being a parameter that can be chosen arbitrarily so as to minimize this cost. Intuitively, this is done by reducing $\log n$ invocations of $\binom{2}{1}$-OT to one invocation of $\binom{n}{1}$-OT; the result is then obtained by combining this reduction with a new $\binom{n}{1}$-OT extension protocol introduced in [35]. In our concrete efficiency estimations, we will heavily rely on this result as our equality test protocol involves only OTs on very short strings.

Correlated and Random Oblivious Transfers. The authors of [4] described several OT extension protocols, tailored to OTs on inputs satisfying some particular conditions. In particular, the communication of the OT extension protocol can be reduced from $2t\ell + t\kappa$ bits to $t\ell + t\kappa$ bits when the inputs to each OT are *correlated*, i.e. when each input pair is of the form $(r, f(r))$ for a uniformly random r and a function f known by the sender (which can be different for each OT). For random oblivious transfer extension, the bit-communication can be further reduced to $t\kappa$. We note that the optimizations of [4,35] can be combined: $\log n$ correlated $\binom{2}{1}$-OT can be reduced to one correlated $\binom{n}{1}$-OT (defined

[1] The random oracle model can be avoided by assuming that the hash function is a correlation-robust function, see [35], Appendix A.2.

by input pairs of the form $(r, f_1(r), \cdots f_{n-1}(r))$ for a random r and functions $f_1 \cdots f_{n-1}$ known by the sender). This gives a correlated short-string oblivious transfer extension protocol which transmits $t(2\kappa/\log n + (n-1)\cdot\ell)$ bits.

3 Equality Test

In this section, we design an equality-test (ET) protocol to securely compute shares over $\mathbb{Z}_2$ of the equality predicate.

Ideal Functionalities. The ideal functionality for our ET protocol is represented on Fig. 1. Following the common standard for multiparty computation, we design our protocol in the preprocessing model, where the players have access to a preprocessing functionality $\mathscr{F}_{\mathsf{ET\text{-}prep}}$. The preprocessing functionality is used in an initialization phase to generate material for the protocol; it does not require the inputs of the players. Our ideal preprocessing functionality is also represented on Fig. 1.

Functionality $\mathscr{F}_{\mathsf{ET}}$

The functionality runs with two parties, Alice and Bob. Upon receiving (ET, x) from Alice and (ET, y) from Bob, set $\beta \leftarrow 1$ if $x = y$, and $\beta \leftarrow 0$ else. Set $(a, b) \leftarrow_R \langle\beta\rangle_2$. Return a to Alice and b to Bob.

Functionality $\mathscr{F}_{\mathsf{ET\text{-}prep}}$

The functionality runs with two parties, Alice and Bob.

Size Reduction: Upon receiving (SR, j) from both players, the functionality picks $(x, y) \leftarrow_R (\mathbb{Z}_2^j)^2$ and sets $(\boldsymbol{a}, \boldsymbol{b}) \leftarrow_R \langle x \oplus y\rangle_{j+1}$. $\mathscr{F}_{\mathsf{ET\text{-}prep}}$ outputs $(x, \boldsymbol{a})$ to Alice and $(y, \boldsymbol{b})$ to Bob.

Product Sharing: Upon receiving (PS, n) from both players, the functionality picks $(x, y) \leftarrow_R (\mathbb{Z}_2^{2^n-2})^2$ and sets $(a, b) \leftarrow_R \langle x * y\rangle_2$. $\mathscr{F}_{\mathsf{ET\text{-}prep}}$ outputs (x, a) to Alice and (y, b) to Bob.

Fig. 1. Ideal functionalities for equality test and preprocessing

Protocol. We now describe our implementation of $\mathscr{F}_{\mathsf{ET}}$ in the $\mathscr{F}_{\mathsf{ET\text{-}prep}}$-hybrid model, with respect to passive corruption. The protocol runs with two players, Alice and Bob. It is parametrized by two integers (ℓ, n), where n is called the *threshold* of the protocol. The players recursively perform size reduction steps using the material produced by the size reduction procedure of $\mathscr{F}_{\mathsf{ET\text{-}prep}}$. Each step reduces inputs of size ℓ to inputs of size $|\ell + 1|$ while preserving the equality predicate. The players stop the reduction when the bitsize of their inputs becomes smaller than the threshold n (taken equal to 3 or 4 in our concrete estimations). The equality predicate is computed on the small inputs with the material produced by the product sharing procedure of $\mathscr{F}_{\mathsf{ET\text{-}prep}}$. The protocol is represented on Fig. 2.

Protocol Π_{ET}

Initialize: Let $i \leftarrow 1$ and $j \leftarrow \ell$. The players perform the following operations:
- (size reduction) While $j > n$, both players call $\mathscr{F}_{\mathsf{ET\text{-}prep}}$ on input (SR, j) to get outputs $(r_i, \boldsymbol{a}_i)$ and $(s_i, \boldsymbol{b}_i)$. The players set $i \leftarrow i+1$ and $j \leftarrow |j+1|$.
- (product sharing) Both players call $\mathscr{F}_{\mathsf{ET\text{-}prep}}$ on input (PS, n) to get outputs (r, a) and (s, b).

Equality Test: On input two ℓ-bit integers, x from Alice and y from Bob, let $x_1 \leftarrow x$ and $y_1 \leftarrow y$. Let $i \leftarrow 1$ and $j \leftarrow \ell$. The players perform the following operations:

1. While $j > n$, Alice sends $x'_i \leftarrow r_i \oplus x_i$ to Bob, and Bob sends $y'_i \leftarrow s_i \oplus y_i$ to Alice. Let $z_i \leftarrow x'_i \oplus y'_i$. Alice sets $x_{i+1} \leftarrow -\sum_{l=1}^{j}(-1)^{z_i[l]}\boldsymbol{a}_i[l] \bmod j+1$, and Bob sets $y_{i+1} \leftarrow \sum_{l=1}^{j}(-1)^{z_i[l]}\boldsymbol{b}_i[l] + z_i[l] \bmod j+1$. The players set $i \leftarrow i+1$ and $j \leftarrow |j+1|$. Note that $(x_i, y_i) \in \mathbb{Z}_j^2$.
2. Once $j \leq n$, let $(I_k)_{1 \leq k \leq 2^n-2}$ denote the list of non-empty strict subsets of $\{1, \cdots, n\}$ (in any arbitrary fixed order). For $k = 1$ to $2^n - 2$, Alice, sets $X_k \leftarrow \prod_{l \in I_k}(1 \oplus x_i[l])$ and $\alpha_k \leftarrow r[k] \oplus X_k$. Then, Bob sets $Y_k \leftarrow \prod_{l \notin I_k} y_i[l]$, and $\beta_k \leftarrow s[k] \oplus Y_k$. Alice picks $\alpha \leftarrow_R \{0,1\}$ and sends $(\alpha, (\alpha_k)_{k \leq 2^n-2})$, and Bob picks $\beta \leftarrow_R \{0,1\}$ and sends $(\beta, (\beta_k)_{k \leq 2^n-2})$.
3. Alice outputs
$$\bigoplus_{k \leq 2^n-2}(a[k] \oplus \beta_k X_k) \oplus \prod_{l \leq n}(1 \oplus x_i[l]) \oplus \alpha \oplus \beta$$
Bob outputs
$$\bigoplus_{k \leq 2^n-2}(b[k] \oplus \alpha_k s[k]) \oplus \prod_{l \leq n} y_i[l] \oplus \alpha \oplus \beta$$

Fig. 2. Protocol for equality test

Theorem 1. *The protocol Π_{ET} securely implements $\mathscr{F}_{\mathsf{ET}}$ in the $\mathscr{F}_{\mathsf{ET\text{-}prep}}$-hybrid model, with respect to passive corruption.*

Due to space constraints, the proof of Theorem 1 is postponed to the full version.

3.1 Implementing the Preprocessing Functionality

We now describe the implementation of the functionality $\mathscr{F}_{\mathsf{ET\text{-}prep}}$, in the $\mathscr{F}_{\mathsf{OT}}$-hybrid model. The protocol is represented on Fig. 3.

Theorem 2. *The protocol Π_{ET} securely implements $\mathscr{F}_{\mathsf{ET}}$ when calls to $\mathscr{F}_{\mathsf{ET\text{-}prep}}$ in Π_{ET} are replaced by executions of $\Pi_{\mathsf{ET\text{-}prep}}$ in the $\mathscr{F}_{\mathsf{OT}}$-hybrid model, with respect to passive corruption.*

Due to lack of space, we postpone the proof to the full version. While the proof is rather straightforward, observe that we do not claim that $\Pi_{\mathsf{ET\text{-}prep}}$ UC-securely implements $\mathscr{F}_{\mathsf{ET\text{-}prep}}$ with respect to passive corruption, but rather that the entire protocol remain secure when calls to $\mathscr{F}_{\mathsf{ET\text{-}prep}}$ are replaced by executions of $\Pi_{\mathsf{ET\text{-}prep}}$. The reason for this distinction is that $\Pi_{\mathsf{ET\text{-}prep}}$ does in fact

Protocol $\Pi_{\mathsf{ET\text{-}prep}}$

Size-Reduction(ℓ)**:** Alice picks $(x, \boldsymbol{a}) \leftarrow_R \mathbb{Z}_2^\ell \times \mathbb{Z}_{\ell+1}^\ell$, and Bob picks $y \leftarrow_R \mathbb{Z}_2^\ell$. The players call $\mathscr{F}_{\mathsf{OT}}^{\ell,|\ell+1|}$, on input $(x[i] - \boldsymbol{a}[i] \bmod \ell + 1, 1 - \boldsymbol{a}[i] - x[i] \bmod \ell + 1)_{i \leq \ell}$ for Alice and y for Bob. Let $\boldsymbol{b}$ denote Bob's output. Alice outputs $(x, \boldsymbol{a})$ and Bob outputs $(y, \boldsymbol{b})$.

Product-Sharing(n)**:** Alice picks $(x, a) \leftarrow_R (\mathbb{Z}_2^{2^n-2})^2$, and Bob picks $y \leftarrow_R \mathbb{Z}_2^{2^n-2}$. The players call $\mathscr{F}_{\mathsf{OT}}^{2^n-2,2}$ on input $(a[i], a[i] \oplus x[i])_{i \leq 2^n-2}$ for Alice and y for Bob. Let b denote Bob's output. Alice outputs (x, a) and Bob outputs (y, b).

Fig. 3. Preprocessing protocol for equality test

not UC-securely implement $\mathscr{F}_{\mathsf{ET\text{-}prep}}$. Intuitively, this comes from the fact that in $\Pi_{\mathsf{ET\text{-}prep}}$, the parties choose (part of) their outputs themselves; hence, no simulator can possibly force the parties to set their outputs to being equal to the outputs of $\mathscr{F}_{\mathsf{ET\text{-}prep}}$. While this can be solved by adding a resharing step at the end of the protocol, this would add some unnecessary interaction and communication to the protocol. Instead, we rely on an approach of [9], which was developed exactly for this purpose: we prove that the protocol is *input-private* (meaning that there is a simulator producing a view indistinguishable from an execution of the protocol for any environment that ignores the output of the protocol), which, as shown in [9], suffices to argue the security of the composed protocol as soon as some rules on ordered composition are respected.

3.2 Communication Complexity

By a classical observation (see e.g. [40]), we can always assume that the inputs of the players are less than κ-bit long: if this is not the case, each party can hash its input first, preserving the correctness of the protocol with overwhelming probability. Therefore, as the largest strings obliviously transferred during the protocol Π_{ET} are $|\ell + 1| \leq |\kappa + 1|$ bit long (for $\kappa = 128$, this corresponds to 8-bit strings), we can benefit from the short-string oblivious transfer extension protocol of [35]. Ignoring the computation of the base OTs, which is performed a single time for an arbitrary number of equality tests, k size reduction procedures on ℓ-bit inputs transmit $O(k\ell(\kappa/\log x + x \cdot |\ell|))$ bits, where x is a parameter that can be arbitrarily set so as to minimize this cost. This minimizes to $O(k\ell\kappa/\log\kappa)$, up to some $\log\log$ term. As a consequence, when performing many equality tests, the (amortized) cost of a single equality test is $O(\kappa\ell/\log\kappa)$ bits in the preprocessing phase (and still $O(\ell)$ bits in the online phase). For inputs of size $\ell > \kappa$, where the players can hash their input first, the complexity becomes $O(\kappa^2/\log\kappa)$ in the preprocessing phase, and $O(\kappa)$ in the online phase.

3.3 Concrete Efficiency

We now analyze the efficiency of our protocol for various input-lengths. In all our numerical applications, we set the security parameter κ to 128. We estimate the efficiency in an amortized setting, where we can use oblivious transfer extension.

Comparison with Equality Test from Garbled Circuit and from 2PC. We compare our protocol to the garbled-circuit-based protocol of [36], and to the solution based on generic 2PC, using the optimized circuit of [24]. We apply all possible optimizations to these two alternative approaches, using random OTs in the offline phase to precompute the online OTs, as well as oblivious transfer extensions. We use optimized OT extensions of short strings for [24], but not for [36], as it involves OT on large keys.

Amortized Setting. We now provide a concrete efficiency analysis of the protocol in an amortized setting, using oblivious transfer extensions. We do not take into account the cost of the base oblivious transfers for the OT extension scheme, as this is a constant independent of the number of equality tests performed, which is the same for both our protocol and the protocol of [36]. Adapting the construction of [35] to the case of correlated short inputs, the exact cost of reducing m oblivious transfers of t-bit strings to κ oblivious transfers of κ-bit strings is $m(2\kappa/\log x + (x-1)t)$ (this takes into account an optimization described in

Table 2. Communication of ℓ-bit ETs

ℓ	Our ET		[36]		[24]	
	Comm.[b]	Rounds	Comm.	Rounds	Comm.	Rounds
Preprocessing phase						
4	1106	2	1288	1	1264	3
8	2018	3	2832	1	3002	4
16	2945	4	5920	1	6636	5
32	5212	4	12096	1	14062	6
64	9863	4	24448	1	29072	7
128	20194	4	49152	1	59250	8
Online phase						
4	28	1	1540	2	96	3
8	44	2	3080	2	228	4
16	54	3	6160	2	504	5
32	88	3	12320	2	1068	6
64	154	3	24640	2	2208	7
128	300	3	49280	2	4500	8

[a]The one-time cost of the base OTs is ignored in the amortized setting.
[b]Comm. denotes the number of bits exchanged during a protocol run.

[35, Appendix A] and the optimization for correlated inputs of [4]).Therefore, the amortized cost of a size reduction protocol on input k is $k(2\kappa/\log x + (x-1)k)$, where x can be chosen so as to minimize this cost. Table 2 sums up the amortized costs of our equality test protocol for various values of ℓ; oblivious transfers for the garbled circuit approach of [36] are performed using the OT extension protocol of [4] on κ-bit inputs, which transmits 3κ bits per OT. As shown in Table 2, our protocol improves over the communication of [36] by up to 80% overall. During the online phase, our protocol is extremely efficient, two orders of magnitude faster than [36]. Our protocol also improves over [24] by about 70% overall, and by 95% in the online phase. Furthermore, it is considerably less interactive, although it remains more interactive than the garbled-circuit-based approach.

Amortized Computational Complexity. The computational complexity of [24,36] and our protocol are directly proportional to their communication in the amortized setting (and it is dominated by the evaluation of hash functions in both, which are required for (extended) OTs and garbled gates), hence our constructions improve upo these protocols regarding computation by factors similar to those listed in Table 2.

Acknowledgements. We thank David Pointcheval for insightful discussions and comments, and Thomas Schneider for pointing out inaccuracies in our cost estimations for the garbled circuit-based constructions of equality tests and secure comparison. The author was supported by ERC grant 339563 (project CryptoCloud) and ERC grant 724307 (project PREP-CRYPTO).

References

1. Aliasgari, M., Blanton, M., Zhang, Y., Steele, A.: Secure computation on floating point numbers. In: NDSS 2013, February 2013
2. Aly, A., Cuvelier, E., Mawet, S., Pereira, O., Van Vyve, M.: Securely solving simple combinatorial graph problems. In: Sadeghi, A.-R. (ed.) FC 2013. LNCS, vol. 7859, pp. 239–257. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-39884-1_21
3. Applebaum, B., Ishai, Y., Kushilevitz, E., Waters, B.: Encoding functions with constant online rate or how to compress garbled circuits keys. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013. LNCS, vol. 8043, pp. 166–184. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-40084-1_10
4. Asharov, G., Lindell, Y., Schneider, T., Zohner, M.: More efficient oblivious transfer and extensions for faster secure computation. In: Sadeghi, A.R., Gligor, V.D., Yung, M. (eds.) ACM CCS 2013, pp. 535–548. ACM Press, November 2013
5. Ayday, E., Raisaro, J.L., Laren, M., Jack, P., Fellay, J., Hubaux, J.P.: Privacy-preserving computation of disease risk by using genomic, clinical, and environmental data. In: Proceedings of USENIX Security Workshop on Health Information Technologies (HealthTech 2013), No. EPFL-CONF-187118 (2013)
6. Beaver, D.: Correlated pseudorandomness and the complexity of private computations. In: 28th ACM STOC, pp. 479–488. ACM Press, May 1996

7. Blake, I.F., Kolesnikov, V.: Strong conditional oblivious transfer and computing on intervals. In: Lee, P.J. (ed.) ASIACRYPT 2004. LNCS, vol. 3329, pp. 515–529. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-30539-2_36
8. Blanton, M., Saraph, S.: Oblivious maximum bipartite matching size algorithm with applications to secure fingerprint identification. In: Pernul, G., Ryan, P.Y.A., Weippl, E. (eds.) ESORICS 2015. LNCS, vol. 9326, pp. 384–406. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-24174-6_20
9. Bogdanov, D., Laud, P., Laur, S., Pullonen, P.: From input private to universally composable secure multiparty computation primitives. Cryptology ePrint Archive, Report 2014/201 (2014). http://eprint.iacr.org/2014/201
10. Bost, R., Popa, R.A., Tu, S., Goldwasser, S.: Machine learning classification over encrypted data. Cryptology ePrint Archive, Report 2014/331 (2014). http://eprint.iacr.org/2014/331
11. Canetti, R.: Universally composable security: a new paradigm for cryptographic protocols. In: 42nd FOCS, pp. 136–145. IEEE Computer Society Press, October 2001
12. Catrina, O., de Hoogh, S.: Improved primitives for secure multiparty integer computation. In: Garay, J.A., De Prisco, R. (eds.) SCN 2010. LNCS, vol. 6280, pp. 182–199. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-15317-4_13
13. Catrina, O., de Hoogh, S.: Secure multiparty linear programming using fixed-point arithmetic. In: Gritzalis, D., Preneel, B., Theoharidou, M. (eds.) ESORICS 2010. LNCS, vol. 6345, pp. 134–150. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-15497-3_9
14. Chu, W.T., Chang, F.C.: A privacy-preserving bipartite graph matching framework for multimedia analysis and retrieval. In: Proceedings of the 5th ACM on International Conference on Multimedia Retrieval, pp. 243–250. ACM (2015)
15. Couteau, G.: New protocols for secure equality test and comparison. Cryptology ePrint Archive, Report 2016/544 (2016). http://eprint.iacr.org/2016/544
16. Couteau, G., Peters, T., Pointcheval, D.: Encryption switching protocols. In: Robshaw, M., Katz, J. (eds.) CRYPTO 2016. LNCS, vol. 9814, pp. 308–338. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-53018-4_12. http://eprint.iacr.org/2015/990
17. Cramer, R., Damgård, I., Nielsen, J.B.: Multiparty computation from threshold homomorphic encryption. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 280–300. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-44987-6_18
18. Cramer, R., Kiltz, E., Padró, C.: A note on secure computation of the moore-penrose pseudoinverse and its application to secure linear algebra. In: Menezes, A. (ed.) CRYPTO 2007. LNCS, vol. 4622, pp. 613–630. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-74143-5_34
19. Damgård, I., Fitzi, M., Kiltz, E., Nielsen, J.B., Toft, T.: Unconditionally secure constant-rounds multi-party computation for equality, comparison, bits and exponentiation. In: Halevi, S., Rabin, T. (eds.) TCC 2006. LNCS, vol. 3876, pp. 285–304. Springer, Heidelberg (2006). https://doi.org/10.1007/11681878_15
20. Damgård, I., Geisler, M., Krøigaard, M.: Efficient and secure comparison for on-line auctions. In: Pieprzyk, J., Ghodosi, H., Dawson, E. (eds.) ACISP 2007. LNCS, vol. 4586, pp. 416–430. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-73458-1_30
21. Damgard, I., Geisler, M., Kroigard, M.: A correction to 'efficient and secure comparison for on-line auctions'. Int. J. Appl. Crypt. **1**(4), 323–324 (2009)

22. Erkin, Z., Veugen, T., Toft, T., Lagendijk, R.L.: Generating private recommendations efficiently using homomorphic encryption and data packing. IEEE Trans. Inf. Forensics Secur. **7**(3), 1053–1066 (2012)
23. Even, S., Goldreich, O., Lempel, A.: A randomized protocol for signing contracts. In: Chaum, D., Rivest, R.L., Sherman, A.T. (eds.) CRYPTO 1982, pp. 205–210. Plenum Press, New York (1982)
24. Garay, J., Schoenmakers, B., Villegas, J.: Practical and secure solutions for integer comparison. In: Okamoto, T., Wang, X. (eds.) PKC 2007. LNCS, vol. 4450, pp. 330–342. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-71677-8_22
25. Gentry, C., Halevi, S., Jutla, C., Raykova, M.: Private database access with HE-over-ORAM architecture. In: Malkin, T., Kolesnikov, V., Lewko, A.B., Polychronakis, M. (eds.) ACNS 2015. LNCS, vol. 9092, pp. 172–191. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-28166-7_9
26. Goldreich, O.: Towards a theory of software protection and simulation by oblivious RAMs. In: Aho, A. (ed.) 19th ACM STOC, pp. 182–194. ACM Press, May 1987
27. Goldreich, O., Micali, S., Wigderson, A.: How to prove all NP statements in zero-knowledge and a methodology of cryptographic protocol design (extended abstract). In: Odlyzko, A.M. (ed.) CRYPTO 1986. LNCS, vol. 263, pp. 171–185. Springer, Heidelberg (1987). https://doi.org/10.1007/3-540-47721-7_11
28. Goodrich, M.T.: Randomized shellsort: a simple oblivious sorting algorithm. In: Charika, M. (ed.) 21st SODA, pp. 1262–1277. ACM-SIAM, January 2010
29. Goodrich, M.T.: Zig-zag sort: a simple deterministic data-oblivious sorting algorithm running in $O(n \log n)$ time. In: 46th ACM STOC, pp. 684–693. ACM Press (2014)
30. Hamada, K., Ikarashi, D., Chida, K., Takahashi, K.: Oblivious radix sort: an efficient sorting algorithm for practical secure multi-party computation. Cryptology ePrint Archive, Report 2014/121 (2014). http://eprint.iacr.org/2014/121
31. Hazay, C., Toft, T.: Computationally secure pattern matching in the presence of malicious adversaries. J. Cryptol. **27**(2), 358–395 (2014)
32. Huang, Y., Evans, D., Katz, J.: Private set intersection: are garbled circuits better than custom protocols? In: NDSS 2012, February 2012
33. 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, Heidelberg (2003). https://doi.org/10.1007/978-3-540-45146-4_9
34. Kilian, J.: Founding cryptography on oblivious transfer. In: 20th ACM STOC, pp. 20–31. ACM Press, May 1988
35. Kolesnikov, V., Kumaresan, R.: Improved OT extension for transferring short secrets. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013. LNCS, vol. 8043, pp. 54–70. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-40084-1_4
36. Kolesnikov, V., Sadeghi, A.-R., Schneider, T.: Improved garbled circuit building blocks and applications to auctions and computing minima. In: Garay, J.A., Miyaji, A., Otsuka, A. (eds.) CANS 2009. LNCS, vol. 5888, pp. 1–20. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-10433-6_1
37. Kolesnikov, V., Schneider, T.: Improved garbled circuit: free XOR gates and applications. In: Aceto, L., Damgård, I., Goldberg, L.A., Halldórsson, M.M., Ingólfsdóttir, A., Walukiewicz, I. (eds.) ICALP 2008. LNCS, vol. 5126, pp. 486–498. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-70583-3_40
38. Laud, P.: A private lookup protocol with low online complexity for secure multiparty computation. In: Hui, L.C.K., Qing, S.H., Shi, E., Yiu, S.M. (eds.) ICICS 2014. LNCS, vol. 8958, pp. 143–157. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-21966-0_11

39. Li, P., Li, T., Yao, Z.A., Tang, C.M., Li, J.: Privacy-preserving outsourcing of image feature extraction in cloud computing. Soft Comput. **21**, 1–11 (2016)
40. Lipmaa, H., Toft, T.: Secure equality and greater-than tests with sublinear online complexity. In: Fomin, F.V., Freivalds, R., Kwiatkowska, M., Peleg, D. (eds.) ICALP 2013. LNCS, vol. 7966, pp. 645–656. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-39212-2_56
41. Naor, M., Pinkas, B.: Efficient oblivious transfer protocols. In: Kosaraju, S.R. (ed.) 12th SODA, pp. 448–457. ACM-SIAM, January 2001
42. Nishide, T., Iwamoto, M., Iwasaki, A., Ohta, K.: Secure $(M+1)$ st-price auction with automatic tie-break. In: Yung, M., Zhu, L., Yang, Y. (eds.) INTRUST 2014. LNCS, vol. 9473, pp. 422–437. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-27998-5_27
43. Nishide, T., Ohta, K.: Multiparty computation for interval, equality, and comparison without bit-decomposition protocol. In: Okamoto, T., Wang, X. (eds.) PKC 2007. LNCS, vol. 4450, pp. 343–360. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-71677-8_23
44. Paillier, P.: Public-key cryptosystems based on composite degree residuosity classes. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 223–238. Springer, Heidelberg (1999). https://doi.org/10.1007/3-540-48910-X_16
45. Rabin, M.: How to exchange secrets by oblivious transfer. Technical report TR-81, Harvard University (1981)
46. Rahulamathavan, Y., Phan, R.C.W., Veluru, S., Cumanan, K., Rajarajan, M.: Privacy-preserving multi-class support vector machine for outsourcing the data classification in cloud. IEEE Trans. Dependable Secure Comput. **11**(5), 467–479 (2014)
47. Sadeghi, A.-R., Schneider, T., Wehrenberg, I.: Efficient privacy-preserving face recognition. In: Lee, D., Hong, S. (eds.) ICISC 2009. LNCS, vol. 5984, pp. 229–244. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-14423-3_16
48. Samanthula, B.K., Jiang, W., Bertino, E.: Lightweight and secure two-party range queries over outsourced encrypted databases. arXiv:1401.3768 (2014)
49. Toft, T.: Solving linear programs using multiparty computation. In: Dingledine, R., Golle, P. (eds.) FC 2009. LNCS, vol. 5628, pp. 90–107. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-03549-4_6
50. Toft, T.: Sub-linear, secure comparison with two non-colluding parties. In: Catalano, D., Fazio, N., Gennaro, R., Nicolosi, A. (eds.) PKC 2011. LNCS, vol. 6571, pp. 174–191. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-19379-8_11
51. Veugen, T.: Improving the DGK comparison protocol. In: 2012 IEEE International Workshop on Information Forensics and Security (WIFS), pp. 49–54. IEEE (2012)
52. Wu, D.J., Feng, T., Naehrig, M., Lauter, K.: Privately evaluating decision trees and random forests. Cryptology ePrint Archive, Report 2015/386 (2015). http://eprint.iacr.org/2015/386
53. Yao, A.C.C.: How to generate and exchange secrets (extended abstract). In: 27th FOCS, pp. 162–167. IEEE Computer Society Press, October 1986
54. Yu, C.-H., Yang, B.-Y.: Probabilistically correct secure arithmetic computation for modular conversion, zero test, comparison, MOD and exponentiation. In: Visconti, I., De Prisco, R. (eds.) SCN 2012. LNCS, vol. 7485, pp. 426–444. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-32928-9_24
55. Zahur, S., Rosulek, M., Evans, D.: Two halves make a whole - reducing data transfer in garbled circuits using half gates. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015. LNCS, vol. 9057, pp. 220–250. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-46803-6_8