Efficient Secure Comparison Protocols

Geoffroy Couteau

ENS, Paris, France *

Abstract. A secure comparison protocol allows players to evaluate the greater-than predicate on hidden values; it addresses a problem that belongs to the field of multiparty computation, in which players wish to jointly and privately evaluate a function on secret inputs. Introduced by Yao under the name *millionaires' problem* [68], secure comparisons have received a great deal of attention. They have proven to be one of the most fundamental building block in a large variety of multiparty computation protocols. However, due to their inherent non-arithmetic structure, they are less efficient than other fundamental primitives, making them a major bottleneck in multiparty computation protocols.
In this work, we design a new two-party protocol for the greater-than functionality, improving over the state of the art. We prove the security of our protocol in the UC model, with respect to passive corruption (that is, semi-honest adversaries), assuming only oblivious transfers. Our construction can readily be used in a large variety of protocols in which secure comparisons constitute the main efficiency bottleneck. We construct all our protocols in the preprocessing model, with an extremely efficient information-theoretically secure online phase. We use oblivious transfer extensions to get rid of all but a constant amount of expensive computations. Toward our goal of secure comparison, we also design protocols for testing equality between private inputs, which improve similarly over the state of the art. The latter contribution is of independent interest.

Keywords. Two-party computation, Secure comparison, Equality test, 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 [68] and Golwasser, Micali, and Widgerson [30,31]. 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 (historically) the very

* CNRS – UMR 8548 and INRIA – EPI Cascade

first MPC problem ever considered (introduced in [68] under the name of millionaire's problem), it is 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 (face recognition, fingerprint recognition), 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.

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 comparisons appear to be a major bottleneck in secure algorithms. 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 improving secure comparison protocols is one of the major roads toward making multiparty computation truly practical.

In this work, we consider secure comparison on inputs privately held by each player, so that the output is shared between the players. This setting captures every possible applications, as simple folklore methods allow to reduce the problem of comparing shared or encrypted inputs to the problem of comparing inputs held by the parties. We construct new two-party protocols for securely comparing private inputs which 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 (this statement will be made precise later on). The security of our protocols is proven in the universal composability framework of Canetti [12], 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, but are willing to gain any possible information from the transcript 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 Comparison

The secure comparison problem has been a very active research field in the past decade, with far too many contributions to mention them all. Therefore, we choose to regroup these protocols into three main categories, and discuss the most prominent constructions (to our knowledge) in each category. To avoid unnecessary details in the presentation, we assume some basic knowledge on classical cryptographic primitives, such as garbled circuits, oblivious transfer and cryptosystems. Preliminaries on oblivious transfers are given Section 2. In the following, we let κ denote a security parameter.

From Garbled Circuits. The first category regroups protocols following the garbled circuit approach of Yao [68]. The protocol of [45], which was later improved in [44], is, to our knowledge, the most communication-efficient secure comparison protocol. In the original protocol, the output of the protocol is revealed to the players, but this can be easily modified by letting the first player pick a random bit and garble a circuit computing the result of the comparison xored with this bit — the two players then end up with secret shares of the output. For ℓ-bit inputs, the protocol of [44] proceeds by letting the first player garble a circuit containing ℓ comparison gates, each gate consisting of three xor gates and one AND gate. Using the free-xor trick [45], the xor gates are garbled for free. The resulting garbled circuit therefore consists of ℓ garbled AND gates. Using the recent result of [71] which reduces the size of garbled AND gates while remaining compatible with the free-xor trick, each garbled AND gate is of size $2(\kappa + 1)$. The first player sends the garbled circuit together with the keys corresponding to his input, then players perform ℓ parallel oblivious transfers so that the second player obliviously recovers the keys corresponding to his own input. In addition to being very communication-efficient (it was compared favorably to several alternative candidates in a survey [63]),[1] in a setting where several instances of the comparison protocol are likely to be invoked, it can rely on oblivious transfer extensions [39] so that any number n of secure comparison protocol, on inputs of any length ℓ, can be executed using a constant number of public key operations (independent of both n and ℓ) and only cheap, symmetric operations per invocation of the secure protocol, making it computationally very efficient.

From Homomorphic Encryption. The second category contains protocols based on some special-purpose homomorphic cryptosystem. In [21], the authors construct a new factorization-based cryptosystem, the DGK cryptosystem, which is additively homomorphic modulo some small prime (they corrected a flaw in the original proposal in [22]). Their protocol involves 2ℓ DGK ciphertexts (the size of the ciphertext depends of the hardness of factorization; current recommendation indicate that a 2048-bit RSA modulus is necessary to reach 112 bits of security) and is often regarded as one of the most computationally efficient. The more recent construction of [28] relies on the flexibility of lattice based cryptosystems to design 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 realistic parameters.

From the Arithmetic Black Box Model. The third category consists of protocols built on top of an arithmetic black box [18] (ABB), which is an ideal reac-

[1] It should be mentioned that this survey does not take into account recent optimizations on garbled circuits and uses highly unoptimized oblivious transfers; we expect an optimized implementation to be considerably more efficient than the one used in [63], hence to compare even better with current alternatives.

tive 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 [26,56] or additively homomorphic encryption (most articles advocate the Paillier scheme [54]). Protocols in this category vary greatly in structure. Most protocols [13,20,27,53,58,69] 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 (for the sake of simplicity, we consider only the total communication of the protocols; in some constructions, most of the work can be performed in a preprocessing phase). The protocols of Toft [62], and Toft and Lipmaa [49], provide solutions using only a sublinear (in ℓ) number of invocations to the cryptographic primitive; however, the total bit complexity remains superlinear in ℓ as each invocation of the cryptographic primitive involves $O(\ell + \kappa)$ bits of communication. In addition, because of the constants involved, these protocols are only competitive for large values of ℓ. Eventually, the protocol of [70] relies on probabilistically correct conversion of shares modulo various values, and can be implemented (with negligible error probability) with bit complexity $O(\ell + \kappa^2)$; again, the constants involved here make the protocol competitive for very large values of ℓ only.[2]

General Overview. The most practical constructions transmit at least $O(\lambda\ell)$ bits, for some security parameter $\lambda = \mathsf{poly}(\kappa)$ which depends of the particular construction. Some constructions are asymptotically more efficient [28, 70], but this comes at the cost of very large constants or somewhat homomorphic encryption, hence these constructions do not beat the most practical constructions [21, 44] for realistic values of ℓ. Asymptotically, the most efficient construction is the $O(\ell + \kappa^2)$ protocol of [70]. Regarding computation, we expect the protocol of [44] to be the most efficient in any large-scale protocol relying on secure comparison, due to its possibility to use oblivious transfer extensions to be implemented with a small (constant) number of public-key operations and cheap symmetric operations.

1.2 Our Contribution

In this work, we construct new protocols for secure comparison, which improve over the best state-of-the-art protocols. More precisely, we construct protocols enjoying the following features:

- Our protocols are secure in the universal composability framework, assuming only the existence of oblivious transfer [26, 56]. The latter can be instantiated from a large variety of cryptographic assumptions. Apart from oblivious transfer, our protocols involve only cheap modular additions over small groups. In particular, this implies that our protocols perform very

[2] A rough estimation indicates that this approach becomes competitive with e.g. [44] only for inputs whose bit-size ℓ is of the order of several thousands.

well regarding computation in an *amortized* setting, in which many secure comparisons are involved. Indeed, oblivious transfer extensions [39] allow to confine all the computationally involved public-key operations to a single run of a constant number of base oblivious transfers, while performing polynomially many secure comparisons. In addition, our protocols are tailored to benefit from the advances in the design of efficient oblivious transfer extension. In particular, we heavily rely on the short-string oblivious transfer extension protocol of [43].

- Our protocols work in the *preprocessing model*: they are designed in two phases, the first of which depends only on the size of the inputs, and not on their actual values.
- When instantiated with specific constructions for oblivious transfer extension (in particular the protocol of [43]), our protocols are, to our knowledge, more efficient than any existing secure comparison protocol regarding communication, for inputs of any bit-size. In particular, they improve over the protocol of [44] by a factor 20 to 30 in the online phase, and by approximately 40% overall.

  Note that numerous two-party computation protocols rely essentially on two components: secure multiplication and secure comparison (additions can in general be performed without interactions). Secure multiplication protocols can be very efficiently precomputed, hence the online communication of these protocols essentially consists in performing the secure comparisons, which is a major efficiency bottleneck. Our result shows that secure comparisons can also be precomputed extremely efficiently, with an online phase which involve just exchanging a few strings; hence, in all such protocols, the online phase can be dramatically reduced.
- Our protocols are designed in a modular way, so that one can easily adapt them to the constraints of a particular setting (e.g., by choosing to optimize the online communication or the overall communication, or by reducing the round complexity at the cost of increasing the communication).
- Our protocols have a low asymptotic communication complexity. Asymptotically, the complexity of our logarithmic-round protocol is $\tilde{O}\left(\ell + \kappa^2\right)$. This approaches the complexity of the best protocols regarding asymptotic communication [28, 70], while remaining efficient for realistic parameters. To our knowledge, our protocols are the first to enjoy both a low asymptotic complexity and a low cost for practical parameters.

We view our contribution as a further indication that basing multiparty computation on oblivious transfers is one of the major directions toward making it truly practical. Additional contributions resulting from the methods we use include:

- New efficient protocols for checking whether two strings are equal,
- A simple method thar reduces by 25% the communication of the Naor-Pinkas oblivious transfer protocol [51], when the size of the transmitted strings is lower than $\kappa/2$ (see Section 3).

1.3 Universal Composability

As we prove the security of our protocols in the universal composability framework (UC), we assume that the reader has some familiarity with it. The universal composability framework has been introduced by Canetti in [12]. It defines protocols by the mean of systems of interactive Turing machines. The expected behavior of the protocol is captured by an ideal functionality F. This functionality is a very simple interactive machine, which is connected to a set of dummy parties, some of whom might be corrupted by an ideal adversary $\mathscr{Sim}$, through perfectly secure authenticated channels. In the real execution of a protocol π, probabilistic polynomial time players, some of whom might be corrupted by a real adversary Adv, interact with each other through some channels. The *environment* refers to an interactive machine Z that oversees the execution of the protocol in one of the two worlds (the ideal world with the functionality F, or the real world with the protocol π). We refer to [12], for the definitions of the real world ensembles $\mathsf{EXEC}_{\pi,\mathsf{Adv},Z}$ and the ideal world ensemble $\mathsf{EXEC}_{F,\mathscr{Sim},Z}$. A protocol UC securely implements a functionality F if for any adversary Adv, there is a simulator $\mathscr{Sim}$ so that the real world ensemble and the ideal world ensemble are indistinguishable for any environment Z.

1.4 Our Method

The approach on which we rely is close to the intuition underlying the secure comparisons in [49, 62]: to compare two strings, one can divide them in equal length blocks, and compare the first block on which they differ. We design a protocol that uses both oblivious transfers and equality tests to obliviously select such a block. A (secure) equality test protocol is a protocol which, on input two strings (x, y), each string being privately held by a player, outputs shares (modulo 2) of a bit which is 1 if and only if $x = y$ (i.e., let b be the bit which is 1 if and only if $x = y$; the two players get respective bits b_0, b_1 such that $b_0 + b_1 = b \bmod 2$).

Keeping this approach in mind, we start by designing an equality test protocol which is based solely on oblivious transfer. It relies on a classical observation: two strings are equal if and only if their Hamming distance is zero. Using this observation, we design a protocol which reduces an equality test on ℓ-bit strings to an equality test on (approximately) $\log \ell$-bit strings (a similar approach was used in [28], using the very powerful primitive of somewhat homomorphic encryption). Eventually, when the strings are small enough, we call an oblivious transfer-based equality test protocol which is tailored to small inputs.

Our equality test protocol do also improve over previous constructions. Using oblivious transfer extensions, in an amortized setting, it improves over prior works by two orders of magnitude during the online phase, and by 50% overall (regarding communication). This contribution is of independent interest: equality test protocols enjoy independent applications as building blocks in various multiparty computation protocols. Examples include, but are not limited to, protocols for switching between homomorphic encryption schemes [17], secure

linear algebra [19], secure pattern matching [36], and secure evaluation of linear programs [61]. In addition, we provide as a supplementary material 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 improve the performances (it reduces the communication of our equality test protocol by up to 50%). Due to space constraints, we postpone its description to the appendix.

1.5 Applications

Our secure comparison protocols are readily usable as building blocks in a variety of semi-honest two-player secure algorithms. In this subsection, we outline a non-exhaustive list of some interesting applications for which they suit particularly well. In general, our protocols compare well to state-of-the-art protocols in settings were a large number of comparisons are involved and a preprocessing step, independent of the inputs, can be executed. In the applications listed thereafter, we expect our secure comparison protocol to perform well compared to prior alternatives, both in terms of computation and communication, and to result in strong efficiency improvements for the application. Note, however, that our protocols are quite interactive, making them less suited in settings were latency would be a major issue.

Obliviously Sorting Data. Sorting data is probably one of the most widely used basic algorithmic operation, as well as a computationally involved one. As a consequence, sorting encrypted value has proven useful in contexts such as private auctions [7,52], oblivious RAM [29,55,64,65], or private set intersection [38], but it remains to date quite slow (implementations [34] report that sorting over a million 32-bit words takes between 5 and 20 minutes, depending on the method used), even though it received a lot of attention [32–35, 41, 72]. Various oblivious sorting algorithms have been designed, but they all rely crucially on secure comparisons; in most algorithms, sorting m integers requires $O(m \log m)$ secure comparisons (in $O(\log m)$ rounds, with $O(m)$ parallel secure comparisons at each round), which easily amounts to millions of secure comparisons on words which are generally 32-bit or 64-bit words.

Biometric Authentication. The field of biometric authentication is a very active research field. While efficiently solving the issues related to the use of passwords, which are hard to remember and easy to crack, they raise concerns regarding the privacy of the individuals, as they involve storing and manipulating private data associated to an individual, such as fingerprints, iris, or faces. As such, a large body of work has been dedicated to the problem of secure biometric matching (see [8,50] for surveys on the topic). One of the most important primitives for such protocols (used in e.g., secure face recognition [24, 59, 67]) is a secure protocol to find the minimum value in a database (e.g., a database of features) of size n. This protocol involves $O(n)$ secure comparisons; for example, the protocol of [24] was evaluated on a database with 320 features; it requires

720 secure comparison protocols on 64-bit inputs (larger databases are likely to be used in realistic applications).

Data Mining and Machine Learning. The enormous amount of data which is generated daily, the emergence of cloud computing, and the constantly growing power of our computers, have placed data mining at the heart of any company who wishes to use those data in a beneficial way. When storing data in the cloud, which is owned by some potentially ill-intentioned cloud provider, the natural solution to ensure the privacy of the data is to encrypt it; multiparty computation protocols can then allow companies or individuals to privately evaluate machine learning algorithms on the encrypted data. Secure comparisons are required for very basic machine learning operations, such as classification [11, 57] or evaluating decision trees [66]. They are necessary in a large variety of applications of secure machine learning; examples include generating private recommendations [25], spam classification [66], multimedia analysis [16], clinical decisions [57], evaluation of disease risk [5], or image feature extraction [47]. For example, the protocols of [66] privately evaluate decision trees and random forest, and require a secure comparison on 64-bit numbers per decision node. For classical applications, a random forest can contain thousands, or tens of thousands, of decision nodes.

Securely Solving Combinatorial Problems. Combinatorial problems, such as finding the flow of maximum capacity in a weighted graph, or searching for the shortest path between two nodes, are encountered in many applications. Their secure counterpart have been investigates in e.g. [2,3,9,46] and have applications in several cryptographic protocols, such as private fingerprint matching (using a secure algorithm to find the maximum matching size in a bipartite graph, see [9]), privacy-preserving GPS guidance, or privacy-preserving determination of topological features in social networks (which is a special case of the maximum flow problem, see [2]). Secure protocols for combinatorial problems typically involve a very large number of secure comparisons: according to [2], for a graph with n nodes and $m \leq n^2$ edges, secure algorithms use n^2 comparisons for Dijkstra's shortest path algorithm, nm comparisons for Bellman-Ford's shortest path algorithm, and nm^2 comparisons for Edmond-Karp's maximum flow algorithms. Hence, even for graphs of reasonable size, a very large number of secure comparisons is required; this is indeed pointed out as being (by far) the dominant cost in those secure protocols.

Computing on Non-Integer Values. While there is a vast literature on multiparty computation on (modular) integers because of their mathematically convenient structure, in realistic applications data is often stored as floating numbers or as fixed-point numbers, and some very classical applications involve non-integer operations, such as computing square roots. This motivated the development of multiparty computation protocol on rational numbers [1,13,15,48]. Secure algorithms for fixed-point arithmetic, as well as for arithmetic on floating

numbers, heavily rely on comparisons. It was pointed out in [13] that comparisons and integer truncation are core components of fixed-point arithmetic, and the most important performance bottlenecks in complex applications.

Other Applications. Other applications for secure comparisons include various types of secure auctions [21,37], range queries over encrypted databases [60] (which involve a secure comparison per element of the database), or secure algorithms for optimization problems (for example, securely evaluating a simplex algorithm [14, 61], which occurs in supply chain planning between concurrent suppliers, might involve hundreds of secure comparisons on $\approx$ 100-bits values, according to [14]).

1.6 Organization

In Section 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. Section 4 focus on the construction of secure comparison protocols. Eventually, we describes a variant of our equality test in Appendix A of the supplementary material.

1.7 Notations

Given a finite set S, the notation $x \leftarrow_R S$ means a uniformly random affectation of an element of S to the variable x. 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 $\mathbb{Z}_2^k$. 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 the value of the corresponding predicate. 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 $\boldsymbol{x}$, $(\boldsymbol{a}, \boldsymbol{b}) \leftarrow_R \langle \boldsymbol{x} \rangle$ 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 [26, 56]. An oblivious transfer is a two-party protocol between a sender and a receiver, where the sender obliviously transfer 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:

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

The primitive can be extended naturally to k-out-of-n oblivious transfers; we let $\binom{n}{k}$-OT_ℓ^t denotes 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 [40, 42] 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 [39], assuming the random oracle model,[3]. We briefly recall the intuition of the OT extension protocol of [39]. 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 receiver (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.

Optimization. An optimization to the [39] paper was proposed in [4] (and discovered independently in [43]). 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 [43], 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 [43]. In our concrete efficiency estimations, we

[3] The random oracle model can be avoided by assuming that the hash function is a correlation-robust function, see [43], AppendixA.2

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 [43] and [4] can be combined: $\log n$ correlated $\binom{2}{1}$-OT can be reduced to one correlated $\binom{n}{1}$-OT (defined 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

3.1 Equality Test Protocol

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

Ideal Functionalities. The ideal functionality for our ET protocol is represented Figure 3. 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 Figure 3.

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 of at most one of the players. 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 $\mathscr{F}_{\mathsf{ET\text{-}prep}}$.

Protocol Π_{ET}

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, ℓ) from both players, the functionality picks $(x, y) \leftarrow_R (\mathbb{Z}_2^\ell)^2$ and sets $(\boldsymbol{a}, \boldsymbol{b}) \leftarrow_R \langle x \oplus y \rangle_{\ell+1}$. $\mathscr{F}_{\mathsf{ET\text{-}prep}}$ 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

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. Let $(I_k)_{1 \le k \le 2^n - 2}$ 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 sends $(\alpha, (\alpha_k)_{k \le 2^n - 2})$ and Bob sends $(\beta, (\beta_k)_{k \le 2^n - 2})$.
3. Alice outputs
$$\bigoplus_{k \le 2^n - 2} (a[k] \oplus \beta_k X_k) \oplus \prod_{l \le n} (1 \oplus x_i[l]) \oplus \alpha \oplus \beta$$
Bob outputs
$$\bigoplus_{k \le 2^n - 2} (b[k] \oplus \alpha_k s[k]) \oplus \prod_{l \le n} y_i[l] \oplus \alpha \oplus \beta$$

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.*

3.2 Security Analysis

Let Adv be an adversary that interacts with Alice and Bob, running the protocol Π_{ET}. We will construct a simulator $\mathscr{S}im$ which interacts with $\mathscr{F}_{\mathsf{ET}}$, so that no environment Z can distinguish an interaction with Adv in Π_{ET} from an interaction with $\mathscr{S}im$ in the ideal world. $\mathscr{S}im$ starts by invoking a copy of Adv. Each time $\mathscr{S}im$ received from Z an input value, he writes it on Adv's input tape as if coming from Z. Each time Adv writes on its output tape, $\mathscr{S}im$ writes the same thing on his output tape.

One Player is Corrupted. We focus here on the case of a corrupted Bob; as the protocol is perfectly symmetrical, the simulation is similar for a corrupted Alice.

Initialize: $\mathscr{S}im$ runs a local copy of $\mathscr{F}_{\mathsf{ET\text{-}prep}}$. He honestly answers to each call to the SR command, and stores the outputs of each call (This step does not require the input of Alice).

Equality Test:

1. When $\mathscr{S}im$ receives $y'_1 = s_1 \oplus y$ from Adv, he retrieves s_1 from his memory and computes $y = y'_1 \oplus s_1$. $\mathscr{S}im_A$ sends (ET, y) to $\mathscr{F}_{\mathsf{ET}}$ on behalf of the corrupted party, and receives a bit T. During each round of the size reduction protocol, $\mathscr{S}im$ does only send uniformly random values x'_i of the appropriate size on behalf of Alice. Moreover, for $i \geq 1$ and while $j > n$, $\mathscr{S}im$ stores $y_{i+1} \leftarrow \sum_{l=1}^{j} (-1)^{z_i[l]} \boldsymbol{b}_i[l] + z_i[l] \bmod j+1$, using the vector $\boldsymbol{b_i}$ stored in $\mathscr{F}_{\mathsf{ET\text{-}prep}}$ and the string $z_i = x'_i \oplus y'_i$.
2. When $\mathscr{S}im$ receives $(\beta, (\beta_k)_{k \leq 2^n - 2})$, he retrieves Bob's output (b, s) to the PS command, and picks uniformly random bits $(\alpha_k)_{k \leq 2^n-2}$. $\mathscr{S}im$ sets
$$\alpha \leftarrow \bigoplus_{k \leq 2^n - 2} (b[k] \oplus \alpha_k s[k]) \oplus \prod_{l \leq n} y_i[l] \oplus T \oplus \beta$$
and sends $(\alpha, (\alpha_k)_{k \leq 2^n-2})$ to Bob.

Remaining Cases. When both parties are corrupted, $\mathscr{S}im$ simply runs Adv internally. When neither party is corrupted, $\mathscr{S}im$ internally runs Alice and Bob honestly, with inputs $(0, 0)$, and forwards the messages exchanged to Adv.

Indistinguishability. We focus on the case were Bob is corrupted; the argument follows symmetrically for a corrupted Alice, and is straightforward when both players are corrupted, or no player is corrupted. We show that the joint view of Z and Adv in the real world is indistinguishable from the view of Z and the simulated Adv in the ideal world. Let t be the number of repetitions of step 1 during the execution of the protocol. As the corrupted player is semi-honest, it honestly follows the specifications of the protocol. Let $(s, b, (x'_i)_{i \leq t}, \alpha, (\alpha_j)_{j \leq 2^n - 2})$ be the view of Adv in a run of Π_{ET} with inputs (x, y). Let (o_A, o_B) denote the outputs of Alice and Bob, that are sent to Z.

Claim. For every $i \leq t$, $[x = y] = [x_i = y_i]$.

We show that for every $i \leq t-1$, $[x_i = y_i] = [x_{i+1} = y_{i+1}]$. As $(x, y) = (x_1, y_1)$, the claim follows. Let $i \leq t-1$ be an integer. As the players follow the protocol, it holds that $y_{i+1} - x_{i+1} = \sum_{l=1}^{j}(-1)^{z_i[l]}(\boldsymbol{b}_i[l] + \boldsymbol{a}_i[l]) + z_i[l] \bmod j+1$, with $z_i = x_i \oplus y_i \oplus r_i \oplus s_i$. Furthermore, for any $l \leq j$, $\boldsymbol{b}_i[l] + \boldsymbol{a}_i[l] = r_i[l] \oplus s_i[l] \bmod j+1$. Observe that when $z_i[l] = 0$, it holds that $x_i[l] \oplus y_i[l] = r_i[l] \oplus s_i[l]$, whereas when $z_i[l] = 1$, it holds that $x_i[l] \oplus y_i[l] = 1 - (r_i[l] \oplus s_i[l])$. Overall, it holds that $(-1)^{z_i[l]}(r_i[l] \oplus s_i[l]) + z_i[l] = x_i[l] \oplus y_i[l]$. Therefore, $y_{i+1} - x_{i+1} = \sum_{l=1}^{j}(x_i[l] \oplus y_i[l]) \bmod j+1$. But the right hand term is exactly the Hamming distance between x_i and y_i, which is bounded by the bitsize j of the strings (hence no overflow occurs with the modulus $j+1$). Observe that the Hamming distance between two strings is 0 if and only if the two strings are equal. Therefore, $y_{i+1} - x_{i+1} = 0$ if and only if $x_i = y_i$. The claim follows.

Claim. The outputs of Alice and Bob in Π_{ET} form shares over $\mathbb{Z}_2$ of $[x = y]$.

By the previous claim, it suffices to show that the output of Alice and Bob in Π_{ET} form shares over $\mathbb{Z}_2$ of $[x_t = y_t]$. It holds that $o_A \oplus o_B = \bigoplus_{k \leq 2^n-2}(a[k] \oplus b[k] \oplus \beta_k X_k \oplus \alpha_k s[k]) \oplus \prod_{l \leq n}(1 - x_t[l]) \oplus \prod_{l \leq n} y_t[l]$. As for $k \leq 2^n - 2$, $\alpha_k = r[k] \oplus X_k$, $\beta_k = s[k] \oplus Y_k$, and $a[k] \oplus b[k] = r[k]s[k]$, this simplifies to $o_A \oplus o_B = \bigoplus_{k \leq 2^n-2} X_k Y_k \oplus \prod_{l \leq n}(1 - x_t[l]) \oplus \prod_{l \leq n} y_t[l]$.

Observe that the right hand term is exactly the product $\prod_{l=1}^{n}((x_t[l] \oplus 1) \oplus y_t[l])$, in developed form. Moreover, this product evaluates to 1 if and only if it holds that for any $l \leq n$, $(x_t[l] \oplus 1) \oplus y_t[l] = 1$, which happens exactly when $x_t[l] = y_t[l]$, and to 0 otherwise. Therefore, $\prod_{l=1}^{n}((x_t[l] \oplus 1) \oplus y_t[l]) = [x_t = y_t]$.

Claim. When Bob is corrupted, the joint distribution $(s, b, (x'_i)_{i \leq t}, \alpha, (\alpha_j)_{j \leq 2^n-2}, o_A, o_B)$ is perfectly indistinguishable from the transcript of an interaction with $\mathscr{S}im$ together with the output of $\mathscr{F}_{\mathsf{ET}}$.

Recall that $\mathscr{S}im$ honestly picks (s, b). Moreover, for each $i \leq t$, $x'_i = x_i \oplus r_i$ is perfectly masked by the random value r_i, and for each $j \leq 2^n - 2$, $\alpha_j = r[j] \oplus X_k$ is perfectly masked by the random bit $r[k]$. The value α is simulated so that the output computed by Bob is exactly the output T of $\mathscr{F}_{\mathsf{ET}}$ for the ideal version of Bob. As α is masked by T, which is a uniformly random bit from the viewpoint of Bob (by definition of $\mathscr{F}_{\mathsf{ET}}$), α is also uniform. The outputs of $\mathscr{F}_{\mathsf{ET}}$ form shares of $[x = y]$, as do (o_A, o_B) (from our above analysis). The claim follows.

3.3 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.

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 $(\boldsymbol{a}[i]+x[i] \bmod \ell+1, \boldsymbol{a}[i]+1-x[i] \bmod \ell+1)_{i\leq\ell}$ 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 $(a[i], a[i] \oplus x[i])_{i\leq n}$ for Alice and y for Bob. Let b denote Bob's output. Alice outputs (x, a) and Bob outputs (y, b).

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.*

Note that we will not prove that $\Pi_{\mathsf{ET\text{-}prep}}$ UC implements $\mathscr{F}_{\mathsf{ET\text{-}prep}}$, as it is not the case (enhancing it to full UC security would degrade its efficiency). Instead, we show that $\Pi_{\mathsf{ET\text{-}prep}}$ satisfies a weaker notion, called input-privacy. This notion was formally defined in [10], whose authors extend the UC framework to show that it is sufficient for a protocol to be input-private, when composed with a UC secure protocol, to assert the security of the composed protocol (under some natural conditions). We postpone the security argument to Appendix B, noting that it is rather straightforward as the framework of of [10] was specifically designed to assert the security of this kind of protocols.

3.4 Communication Complexity

We first analyse the complexity of our protocol using any standard oblivious transfer protocol, which transmits $O(\kappa)$ bits. The size reduction procedure transmits $O(\ell\kappa)$ bits on input ℓ, and the product sharing procedure transmits $O(2^n\kappa)$ bits on input n. Setting n to a small constant value (e.g. $n = 3$), the bit communication of Π_{ET} is dominated by its first size reduction procedure, which transmits $O(\kappa\ell)$ bits in the initialization phase (recall that this is a preprocessing phase, performed before the parties get to know their inputs). In the online phase, where the equality test is performed using the preprocessed material, only $O(\ell)$ bits are transmitted.

We now analyse the complexity of our protocol in an amortized setting, where many equality test protocols are likely to be invoked (not necessarily in parrallel). Observe that 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 start by hashing his input, using a hash function $H : \{0,1\}^* \mapsto \{0,1\}^\kappa$. When H is modeled as a random oracle[4], this preserves 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

[4] The random oracle model is already assumed in the OT extension protocol we rely on, hence this does not add any assumption to the protocol.

can benefit from the short-string oblivious transfer extension protocol of [43]. 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 than 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.

3.5 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 both the efficiency in a single run setting, and in an amortized setting, where we can use oblivious transfer extension.

Comparison with Equality Test from Garbled Circuit. We compare our protocol to the garbled-circuit-based protocol of [44], which is to our knowledge the most efficient state-of-the-art protocol for equality test (in the description of the protocol, the result of the test is revealed to both players, but the protocol can be trivially modified so that the output is shared between the players).

Let us provide an intuition of this protocol; details on garbled circuits and the free-xor trick can be found in [45]. First, the comparison function is represented as a circuit with ℓ comparison gates, each gate being implemented with three xor gates (which are for free in the construction, in the sense that they do not require communicating anything) and a and gate. During the preprocessing phase, Alice starts by assigning two keys to each wire of the circuit (corresponding to the two possible values 0 and 1), and for each gate g, she computes a *garbled gate*, which encrypts the output-wire keys of the gate so that given two input-wire keys corresponding to inputs (b, b'), only the output-wire key corresponding to $g(b, b')$ can be recovered. Using the recent result of [71], each AND gate can be garbled quite efficiently, using only two $(\kappa+1)$-bit strings. Then, once the inputs are revealed to the players, Alice sends to Bob the ℓ keys corresponding to the bits of her entry, and acts as sender in ℓ parallel oblivious transfers, using as input each pair of keys corresponding to the possible values for an input of Bob. Bob's selection bits are his input bits. Once he has recovered the necessary keys, Bob can evaluate the circuit securely and get the output.

Note that the pairs of keys in this scheme satisfy some correlation, hence the optimization of [4] for correlated oblivious transfer extensions can be applied. However, this prevents Alice from constructing and sending the garbled circuit during the preprocessing phase, as the values of the keys will be determined by the correlated oblivious transfers (in which one of the outputs is a random value). In our estimations, we choose not to use this optimization, which results in a

slight loss in overall communication, but almost cuts in half the communication during the online phase.

Non-Amortized Setting. We now evaluate the concrete efficiency of our protocol. We first focus on the simpler setting, the non-amortized setting, in which a single ET will be performed. We stop the size reduction protocol as soon as $n \leq 4$ (stopping at $n \leq 3$ or $n \leq 2$ saves a few hundreds of bits for some sizes of ℓ, at the cost of additional rounds). For the oblivious transfer, we use the scheme of Naor and Pinkas [51], whose security in the random oracle model relies on the decisional Diffie-Hellman (DDH) assumption (we use an elliptic curve of prime order p, which can be taken of bit-size $\log p = 2\kappa$ according to recommended parameters). The protocols of [51] have the following communication:

- t executions of a $\binom{2}{1}$-OT_ℓ on strings of size $\ell \leq \kappa$ transmit $\kappa 4t$ bits. The initialization phase consists of two group elements sent by the sender which amounts to 4κ bits.
- t executions of a $\binom{N}{1}$-OT_ℓ on strings of size $\ell \leq \kappa$ transmit $\kappa(N+2)t$ bits. The initialization phase consists of $N+1$ group elements sent by the sender which amounts to $2(N+1)\kappa$ bits.

We will use this $\binom{N}{1}$-OT_ℓ in a crucial way. One of the constructions described in [51] gives a trade-off between communication (which is increased) and computation (which is decreased). The idea is that to perform $\log N$ oblivious transfers on ℓ-bit strings, it suffices to perform a single $\binom{N}{1}$-$\mathsf{OT}_{\ell \log N}$, in which the N inputs are all the 2^{logN} possible concatenations of one input from each of the $\log n$ input pairs. But recall that the communication of the protocol of [51] is always the same for any $\ell \leq \kappa$ (for larger values, the oblivious transfer are performed on keys, which are used to encrypt the values, adding 2ℓ bits of overhead to the protocol). Our equality test protocol involves only oblivious transfers on very small strings, of size smaller than $|\kappa + 1|$. Hence, by picking a sufficiently small N so that $\ell \log N \leq \kappa$, the trade-off protocol of Naor and Pinkas does in fact *reduce* the communication. Indeed, performing $\log N$ oblivious transfers on short strings transmits $4\kappa \log N$ bits, while using instead a single $\binom{N}{1}$-$\mathsf{OT}_{\ell \log N}$ transmits $(N+2)\kappa$ bits if $\ell \log N \leq \kappa$. This amounts to $(N+2)\kappa / \log N$ bits per $\binom{2}{1}$-OT, which is minimized for $N = 4$ and transmits 3κ bits. Hence, performing the $\binom{2}{1}$-OT by pairs, as a single $\binom{4}{1}$-OT, reduces the communication by 25% when the transmitted strings are of size $\ell \leq \kappa/2$.

Table 1 sums up the costs of our equality test protocol for various values of ℓ, and compares it to the garbled-circuit-based protocol of [44]. Note that we use the oblivious transfer of [51] in both our ET and the protocol of [44], but while we can use the optimization described above to reduce the communication in our protocol (as it transmits short strings), this does not hold for garbled circuits, in which the transmitted values are κ-bit keys (and our optimization does not result in any improvement in this case). Hence, t $\binom{2}{1}$-OT transmit $3\kappa t$ bits in our ET, but $4\kappa t$ bits in [44]. As one can see from Table 1, our protocol transmits more bits than [44] in the preprocessing phase, but has a communication two orders of

magnitudes smaller in the online phase. Overall, our protocol is approximately 50% more efficient than [44].

Table 1: Communication of ℓ-bit ETs, single run setting

	Our ET		[44]	
ℓ	length	rounds	length	rounds
Preprocessing Phase				
4	5376 bits	2 rounds	1032 bits	1 round
8	8448 bits	3 rounds	2064 bits	1 round
16	10365 bits	4 rounds	4128 bits	1 round
32	16896 bits	4 rounds	8256 bits	1 round
64	29568 bits	4 rounds	16512 bits	1 round
128	57600 bits	4 rounds	33024 bits	1 round
Online Phase				
4	28 bits	1 rounds	2560 bits	2 rounds
8	44 bits	2 rounds	5120 bits	2 rounds
16	54 bits	3 rounds	10240 bits	2 rounds
32	88 bits	3 rounds	20480 bits	2 rounds
64	154 bits	3 rounds	40960 bits	2 rounds
128	300 bits	3 rounds	81920 bits	2 rounds

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 [44]. Adapting the construction of [43] 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 [43, 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, and compares it again with [44]; oblivious transfers for the garbled circuit approach of [44] 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 [44] by up to 80% overall. During the

online phase, our protocol is extremely efficient, two orders of magnitude faster than [44].

Table 2: Amortized communication of ℓ-bit ETs using OT extension

	Our ET		[44]	
ℓ	length	rounds	length	rounds
Preprocessing Phase				
4	1106 bits	2 rounds	1032 bits	1 round
8	2018 bits	3 rounds	2064 bits	1 round
16	2945 bits	4 rounds	4128 bits	1 round
32	5212 bits	4 rounds	8256 bits	1 round
64	9863 bits	4 rounds	16512 bits	1 round
128	20194 bits	4 rounds	33024 bits	1 round
Online Phase				
4	28 bits	1 rounds	2048 bits	2 rounds
8	44 bits	2 rounds	4096 bits	2 rounds
16	54 bits	3 rounds	8192 bits	2 rounds
32	88 bits	3 rounds	16384 bits	2 rounds
64	154 bits	3 rounds	32768 bits	2 rounds
128	300 bits	3 rounds	65536 bits	2 rounds

4 Secure Comparison from Equality Test

4.1 Comparison Protocol

In this section, we design a secure comparison (SC) protocol to securely evaluate shares over $\mathbb{Z}_2$ of the greater-than predicate. Note that when we consider the comparison of elements of $\mathbb{Z}_n$ for some n, we refer to the comparison of these elements seen as integers smaller than n.

Ideal Functionalities. The ideal functionality $\mathscr{F}_{\mathsf{SC}}$ for our SC protocol is represented Figure 4. Our implementation will be close in spirit to the ET protocol of the previous section: it relies on a preprocessing functionality $\mathscr{F}_{\mathsf{SC\text{-}prep}}$, which contains a size reduction command, used to perform several steps of reduction of the input size (while preserving the comparison predicate), and a product sharing protocol, used to perform a secure comparison on small inputs. However, the size reduction procedure is way more involved than in our ET protocol. To

simplify the exposition, we initially assume that the players have access to an intermediate functionality $\mathscr{F}_{\mathsf{compress}}$, which performs the reduction procedure. We will implement this functionality in the $(\mathscr{F}_{\mathsf{ET}}, \mathscr{F}_{\mathsf{SC\text{-}prep}})$-hybrid model afterward. Both $\mathscr{F}_{\mathsf{compress}}$ and $\mathscr{F}_{\mathsf{SC\text{-}prep}}$ are represented Figure 4.

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

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

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

The functionality runs with two parties, Alice and Bob. Upon receiving $(\mathsf{compress}, \ell, \lambda, x)$ from Alice and $(\mathsf{compress}, \ell, \lambda, y)$ from Bob, where $\lambda < \ell$ and (x, y) are ℓ-bit long, it picks two uniformly random values $(\hat{x}, \hat{y}) \leftarrow_R \mathbb{Z}_{2^{\lambda+1}}^2$ such that $\hat{x} \leq \hat{y}$ if and only if $x \leq y$. It returns $\hat{x}$ to Alice and $\hat{y}$ to Bob.

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

The functionality runs with two parties, Alice and Bob.

Size Reduction: Upon receiving $(\mathsf{SR}, \lambda, \mu)$ from both players, the functionality picks $(\boldsymbol{s_0}, \boldsymbol{s_1}, \boldsymbol{t_0}, \boldsymbol{t_1}, \boldsymbol{u_0}, \boldsymbol{u_1}) \leftarrow_R (\mathbb{Z}_{\mu+1}^{\mu})^2 \times (\mathbb{Z}_{2^{\lambda+1}}^{\mu})^4$ and $(c, d, e) \leftarrow_R (\mathbb{Z}_2^{\mu})^4$. $\mathscr{F}_{\mathsf{SC\text{-}prep}}$ sets

$$(\boldsymbol{s}, \boldsymbol{t}, \boldsymbol{u}) \leftarrow \left(\left(\boldsymbol{s_{c[i]}}[i]\right)_{i \leq \mu}, \left(\boldsymbol{t_{d[i]}}[i]\right)_{i \leq \mu}, \left(\boldsymbol{u_{e[i]}}[i]\right)_{i \leq \mu} \right)$$

It outputs $(e, \boldsymbol{s_0}, \boldsymbol{s_1}, \boldsymbol{t_0}, \boldsymbol{t_1}, \boldsymbol{u})$ to Alice and $(c, d, \boldsymbol{s}, \boldsymbol{t}, \boldsymbol{u_0}, \boldsymbol{u_1})$ to Bob.

Product Sharing: Upon receiving (PS, n) from both players, the functionality picks $(\rho, \sigma) \leftarrow_R (\mathbb{Z}_2^{2^n-1})^2$ and sets $(a, b) \leftarrow_R \langle \rho * \sigma \rangle_2$. $\mathscr{F}_{\mathsf{ET\text{-}prep}}$ outputs (ρ, a) to Alice and (σ, b) to Bob.

Fig. 2: Ideal Functionalities for Secure Comparison and Preprocessing

Protocol. We now describe our implementation of $\mathscr{F}_{\mathsf{SC}}$, in the $(\mathscr{F}_{\mathsf{ET}}, \mathscr{F}_{\mathsf{compress}})$-hybrid model, with respect to passive corruption of at most one of the players. The protocol runs with two players, Alice and Bob. It is parametrized by two integers (ℓ, n), where n is the threshold of the protocol. The players recursively perform size reduction steps using the material produced by the size reduction procedure of $\mathscr{F}_{\mathsf{SC\text{-}prep}}$. Each step reduces inputs of size ℓ to inputs of size roughly $\sqrt{\ell}$ while preserving the comparison 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 comparison predicate is computed on the small inputs with the material produced by the product sharing procedure of $\mathscr{F}_{\mathsf{SC\text{-}prep}}$.

Protocol Π_{SC}

Initialize: Both players call $\mathscr{F}_{\mathsf{SC\text{-}prep}}$ on input (PS, n) to get (ρ, a) and (σ, b).

Secure Comparison: On input two ℓ-bit integers, x from Alice and y from Bob, let $i \leftarrow 1$, $x_i \leftarrow x$, and $y_i \leftarrow y$.

1. The players agree on two integers λ_i, μ_i such that $\lambda_i \leq \ell$ and μ_i is the smallest integer satisfying $\lambda_i \mu_i \geq \ell$. Alice calls $\mathscr{F}_{\mathsf{compress}}$ on input $(\mathsf{compress}, \ell, \lambda_i, x_i)$, and Bob on input $(\mathsf{compress}, \ell, \lambda_i, y_i)$. Let (x_{i+1}, y_{i+1}) denote their respective outputs. The players set $\ell \leftarrow \lambda_i$ and $i \leftarrow i+1$. If $|\ell| > n-1$, the players iterate the step 1.
2. Alice sets $x'_i \leftarrow x_i - 1$. Let $f : (j,l) \mapsto l - 1 + 2^{j-1}$. For $j = 1$ to $n-1$, let $(I_l^j)_{1 \leq l \leq 2^j}$ denote the list of subsets of $\{1, \cdots, j\}$ (in any arbitrary fixed order). For $j = 1$ to n, for $l = 1$ to 2^{j-1}, Alice picks $\alpha \leftarrow_R \{0,1\}$ and sets $\alpha_{jl} \leftarrow \rho[f(j,l)] \oplus x'_i[j] \cdot \prod_{k \in I_l^{j-1}} (1 \oplus x'_i[k])$, and Bob picks $\beta \leftarrow_R \{0,1\}$ and sets $\beta_{jl} \leftarrow \sigma[f(j,l)] \oplus (1 \oplus y_i[j]) \cdot \prod_{k \notin I_l^{j-1}} y_i[k]$.[5] Alice sends $(\alpha, (\alpha_{jl})_{jl})$ and Bob sends $(\beta, (\beta_{jl})_{jl})$ (this amounts to 2^{n+1} bits exchanged).
3. Alice outputs

$$\bigoplus_{\substack{j \leq n \\ l \leq 2^j - 1}} \rho[f(j,l)]\beta_{jl} \oplus a[f(j,l)] \oplus \alpha \oplus \beta$$

Bob outputs

$$\bigoplus_{\substack{j \leq n \\ l \leq 2^j - 1}} (\sigma[f(j,l)] \oplus \beta_{jl})\alpha_{jl} \oplus b[f(j,l)] \oplus \alpha \oplus \beta$$

Theorem 3. *The protocol Π_{SC} securely implements $\mathscr{F}_{\mathsf{SC}}$ in the $(\mathscr{F}_{\mathsf{SC\text{-}prep}}, \mathscr{F}_{\mathsf{compress}})$-hybrid model, with respect to passive corruption.*

We postpone the proof of this theorem to Appendix B, noting that it closely resemble the proof of Theorem 1.

4.2 Compression Functionality

We now implement the functionality $\mathscr{F}_{\mathsf{compress}}$, in the $(\mathscr{F}_{\mathsf{SC\text{-}prep}}, \mathscr{F}_{\mathsf{ET}})$-hybrid model.

Protocol Π_{compress}

Let (ℓ, λ) be two integers such that $\lambda \leq \ell$. Let μ be the smallest integer such that $\lambda\mu \geq \ell$. On input x from Alice and y from Bob, both of size ℓ-bit,

[5] By convention, an empty product is equal to 1.

Initialize: The players call $\mathscr{F}_{\mathsf{SC\text{-}prep}}$ on input $(\mathsf{SR}, \lambda, \mu)$ to get outputs $(e, \boldsymbol{s_0}, \boldsymbol{s_1}, \boldsymbol{t_0}, \boldsymbol{t_1}, \boldsymbol{u})$ for Alice and $(c, d, \boldsymbol{s}, \boldsymbol{t}, \boldsymbol{u_0}, \boldsymbol{u_1})$ for Bob.

Compression: Let $(x_j)_{j\leq\mu} \in \mathbb{Z}_{2^\lambda}$ (resp. $(y_j)_{j\leq\mu} \in \mathbb{Z}_{2^\lambda}$) be the decomposition of x (resp. y) into μ blocs of size λ (i.e., $x = \sum_{j=1}^{\mu} x_j 2^{\lambda(j-1)}$ and $y = \sum_{j=1}^{\mu} y_j 2^{\lambda(j-1)}$). The players perform the following operations:

1. For $j = 1$ to μ, the players call $\mathscr{F}_{\mathsf{ET}}$ on inputs (ET, x_j) and (ET, y_j). Let $(\alpha_j, \beta_j)_{j\leq\mu} \in \mathbb{Z}_2^{2\mu}$ denote their respective outputs.
2. Alice picks $\boldsymbol{r} \leftarrow_R \mathbb{Z}_{\mu+1}^{\mu}$. For $j = 1$ to μ, Bob sends $\beta_j \oplus c[j]$ to Alice. If this is 0, Alice sends $(\boldsymbol{s_0}[j] + \alpha_j + \boldsymbol{r}[j] \bmod \mu+1, \boldsymbol{s_1}[j] + 1 - \alpha_j + \boldsymbol{r}[j] \bmod \mu+1)$; else, she sends this pair in permuted order. This allows Bob to recover $\boldsymbol{r}[j] + (\alpha_j \oplus \beta_j) \bmod \mu+1$. For $j = 1$ to μ, Alice sets $x'_j \leftarrow \sum_{k=1}^{j} \boldsymbol{r}[k] \bmod \mu+1$ and Bob sets $y'_j \leftarrow \sum_{k=1}^{j} \boldsymbol{r}[k] + (\alpha_k \oplus \beta_k) \bmod \mu+1$. Observe that $(x'_j, y'_j) \in \mathbb{Z}_{\mu+1}^2$.
3. For $j = 1$ to μ, the players call $\mathscr{F}_{\mathsf{ET}}$ on inputs (ET, x'_j) and (ET, y'_j). Let $(\alpha'_j, \beta'_j)_{j\leq\mu} \in \mathbb{Z}_2^{2\mu}$ denote their respective outputs and $(\alpha'_0, \beta'_0) \leftarrow (0, 0)$. For $j = 1$ to μ_i, Alice sets $\gamma_j \leftarrow \alpha_{j-1} \oplus \alpha_j$ and bob sets $\delta_j \leftarrow \beta_{j-1} \oplus \beta_j$. The following steps 4 and 5 are executed in parrallel:
4. Alice picks $\boldsymbol{r_A} \leftarrow_R \mathbb{Z}_{2^{\lambda+1}}^{\mu}$. For $j = 1$ to μ, Bob sends $\delta_j \oplus d[j]$ to Alice. If this is 0, Alice sends $(\boldsymbol{t_0}[j] + \gamma_j x_j + \boldsymbol{r_A}[j] \bmod 2^{\lambda+1}, \boldsymbol{t_1}[j] + (1-\gamma_j)x_j + \boldsymbol{r_A}[j] \bmod 2^{\lambda+1})$; else, she sends this pair in permuted order. This allows Bob to recover $\boldsymbol{r_A}[j] + (\gamma_j \oplus \delta_j)x_j \bmod 2^{\lambda+1}$.
5. Bob picks $\boldsymbol{r_B} \leftarrow_R \mathbb{Z}_{2^{\lambda+1}}^{\mu}$. For $j = 1$ to μ, Alice sends $\gamma_j \oplus e[j]$ to Bob. If this is 0, Bob sends $(\boldsymbol{u_0}[j] + \delta_j y_j + \boldsymbol{r_B}[j] \bmod 2^{\lambda+1}, \boldsymbol{u_1}[j] + (1-\delta_j)y_j + \boldsymbol{r_B}[j] \bmod 2^{\lambda+1})$; else, he sends this pair in permuted order. This allows Alice to recover $\boldsymbol{r_B}[j] + (\gamma_j \oplus \delta_j)y_j \bmod 2^{\lambda+1}$.

Output: Alice outputs $\hat{x} \leftarrow 2^\lambda + \sum_{j=1}^{\mu} \boldsymbol{r_A}[j] + \boldsymbol{r_B}[j] + (\gamma_j \oplus \delta_j)y_j \bmod 2^{\lambda+1}$ and Bob outputs $\hat{y} \leftarrow \sum_{j=1}^{\mu} \boldsymbol{r_B}[j] + \boldsymbol{r_A}[j] + (\gamma_j \oplus \delta_j)x_j \bmod 2^{\lambda+1}$.

Theorem 4. *The protocol Π_{compress} securely implements $\mathscr{F}_{\mathsf{compress}}$ in the ($\mathscr{F}_{\mathsf{ET}}$, $\mathscr{F}_{\mathsf{SC\text{-}prep}}$)-hybrid model, with respect to passive corruption.*

We postpone the proof of this theorem to Appendix B.

4.3 Implementing the Preprocessing Functionality

We now describe the implementation of the functionality $\mathscr{F}_{\mathsf{SC\text{-}prep}}$, in the $(\mathscr{F}_{\mathsf{OT}}, \mathscr{F}_{\mathsf{ROT}})$-hybrid model.

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

Size-Reduction(ℓ)**:** The players perform the following operations:

1. The players call $\mathscr{F}_{\mathsf{ROT}}^{\mu,\mu+1}$, with Alice acting as sender and Bob as receiver. Let $(\boldsymbol{s_0}, \boldsymbol{s_1})$ denote Alice's output, and let $e \in \mathbb{Z}_2^{\mu+1}$ and $\boldsymbol{s} \leftarrow \left(\boldsymbol{s_{e[i]}}[i]\right)_{i\leq\mu}$ denote Bob's output.
2. The players call $\mathscr{F}_{\mathsf{ROT}}^{\mu,2^{\lambda+1}}$, with Alice acting as sender and Bob as receiver. Let $(\boldsymbol{t_0}, \boldsymbol{t_1})$ denote Alice's output, and let $d \in \mathbb{Z}_2^{2^{\lambda+1}}$ and $\boldsymbol{t} \leftarrow \left(\boldsymbol{t_{d[i]}}[i]\right)_{i\leq\mu}$ denote Bob's output.
3. The players call $\mathscr{F}_{\mathsf{ROT}}^{\mu,2^{\lambda+1}}$, with Alice acting as receiver and Bob as sender. Let $(\boldsymbol{u_0}, \boldsymbol{u_1})$ denote Bob's output, and let $e \in \mathbb{Z}_2^{2^{\lambda+1}}$ and $\boldsymbol{u} \leftarrow \left(\boldsymbol{u_{e[i]}}[i]\right)_{i\leq\mu}$ denote Alice's output.

Product-Sharing(n)**:** Alice picks $(x, a) \leftarrow_R (\mathbb{Z}_2^{2^n-2})^2$, and Bob picks $y \leftarrow_R \mathbb{Z}_2^{2^n-1}$. The players call $\mathscr{F}_{\mathsf{OT}}^{2^n-2,1}$ on input $(a[i], a[i] \oplus x[i])_{i\leq n}$ for Alice and y for Bob. Let b denote Bob's output. Alice outputs (x, a) and Bob outputs (y, b).

Theorem 5. *The protocol Π_{SC} securely implements $\mathscr{F}_{\mathsf{SC}}$ when calls to $\mathscr{F}_{\mathsf{SC\text{-}prep}}$ in Π_{SC} are replaced by executions of $\Pi_{\mathsf{SC\text{-}prep}}$ in the $(\mathscr{F}_{\mathsf{ROT}}, \mathscr{F}_{\mathsf{OT}})$-hybrid model, with respect to passive corruption.*

The proof of this theorem is essentially identical to the proof of Theorem 2.

4.4 Efficiency Analysis

As for the equality test protocol, we estimate both the asymptotic complexity and the concrete efficiency of our protocols; however, we focus only on the amortized setting here, which is more meaningful in most applications. In all our numerical applications, we set the security parameter κ to 128. We consider two protocols in our estimations:

- The protocol Π_{SC} described above, which performs a logarithmic (in the bitsize ℓ of the inputs) number of size reduction steps (hence has logarithmic round complexity)
- A constant-round variant of Π_{SC} where only a constant number c of size reduction steps are performed, and the final comparison on smaller entries is executed using the protocol of [44] (which transmits $O(\kappa t)$ bits on t-bit inputs).

Communication Complexity. For any integer t, we denote by Π_{ET}^t the protocol Π_{ET} for t-bit inputs. The full protocol involves μ parallel executions of $\Pi_{\mathsf{ET}}^{\lambda}$, $\Pi_{\mathsf{ET}}^{|\mu+1|}$, $\mathsf{OT}_{|\mu+1|}$, 2μ executions of $\mathsf{OT}_{\lambda+1}$, and performing a secure comparison on $(\lambda+1)$-bit inputs. Asymptotically, an equality test transmits $O(\kappa^2/\log\kappa)$ bits

independently of the size of ℓ, as the size of the strings to be compared can be reduced while statistically preserving their equality. In the constant-round setting, this gives us a $O(c \cdot \log^* \kappa)$-round protocol with asymptotic communication

$$O\left(c\left(\frac{\ell \log \kappa}{\kappa}\right)^{\frac{1}{c+1}} \frac{\kappa^2}{\log \kappa} + \ell\right)$$

In the logarithmic-round setting, we set $c = O(\log \ell / \log\log \kappa)$ (hence the round complexity becomes $O(\log \ell \cdot \log^* \kappa / \log\log \kappa)$); the asymptotic communication becomes

$$O\left(\frac{\kappa^2 \log \ell}{\log \kappa \log\log \kappa} + \ell\right)$$

Concrete Efficiency. We now estimate the efficiency of our secure comparison protocol, in an amortized setting (using oblivious transfer extension). We use the equality test of the previous section, with short-string correlated oblivious transfer extension [4,43]. The results are given in Table 3; they correspond to the results obtained using the optimal block-decomposition of the inputs. The notes in Table 3 indicate the optimal values of λ_i, μ_i for each value of ℓ. SC 1 denotes the protocol obtained by recursively applying the reduction protocol, until the inputs are small enough so that the small-string secure comparison protocol becomes efficient. We set the thresholds of both the secure comparison protocol and the equality-tests subprotocol to 4. If one is willing to reduce the round complexity of the protocol at the cost of transmitting more bits, the threshold can be increased. SC 2 denotes the protocol obtained by performing a single reduction step, then using the garbled circuit approach of [44] to complete the protocol. This approach is interesting only for $\ell > 16$, as for $\ell \leq 32$, the optimal values for λ are equal to 4 or less, hence applying the small-string equality test directly is more efficient than using garbled circuits (and has the same round complexity). As one can see from Table 3, the communication in SC_1 is reduced by 30 to 45% compared to the garbled circuit approach overall, and by $97-99\%$ during the online phase. SC_2 has comparable overall efficiency, but offers slightly less communication improvements during the online phase, in exchange for a better round complexity.

5 Conclusion and Open Questions

In this work, we proposed new two-player protocols for equality test and comparison secure against honest-but-curious adversaries, which improve over prior state-of-the-art. This leaves room for improvements in several directions. First, our result cannot be immediatly generalized to n players as such; extending the protocols to handle an arbitrary number of players holding shares of the two inputs would be an interesting improvement. Second, due to the highly non-algebraic structure of our protocols, standard method for enhancing their security into security against malicious adversaries would be rather inefficient

Table 3: Amortized communication of ℓ-bit SC

	SC 1		SC 2		[44]	
ℓ	length	rounds	length	round	length	round
			Preprocessing Phase			
4	1185 bits	2 rounds	-		1032 bits	1 round
8[1]	3572 bits	2 rounds	-		2064 bits	1 round
16[2]	8396 bits	2 rounds	-		4128 bits	1 round
32[3]	15120 bits	3 rounds	13450 bits	3 rounds	8256 bits	1 round
64[4]	31388 bits	3 rounds	29880 bits	3 rounds	16512 bits	1 round
128[5]	52121 bits	3 rounds	49291 bits	3 rounds	33024 bits	1 round
			Online Phase			
4	30 bits	2 rounds	-		2048 bits	2 rounds
8	162 bits	6 rounds	-		4096 bits	2 rounds
16	308 bits	6 rounds	-		8192 bits	2 rounds
32	530 bits	12 rounds	4014 bits	7 rounds	16384 bits	2 rounds
64	1120 bits	12 rounds	5154 bits	7 rounds	32768 bits	2 rounds
128	2101 bits	12 rounds	7071 bits	7 rounds	65536 bits	2 rounds

[1] $\mu_1 = 4, \lambda_1 = 2$
[2] $\mu_1 = 6, \lambda_1 = 3$ reduces the input size to $\ell = 5$, then $\mu_2 = 3, \lambda_2 = 2$
[3] $\mu_1 = 6, \lambda_1 = 6$ reduces the input size to $\ell = 7$, then $\mu_2 = 4, \lambda_2 = 2$
[4] $\mu_1 = 10, \lambda_1 = 7$ reduces the input size to $\ell = 8$, then $\mu_2 = 4, \lambda_2 = 2$
[5] $\mu_1 = 15, \lambda_1 = 9$ reduces the input size to $\ell = 10$, then $\mu_2 = 4, \lambda_2 = 2$

here. Hence, enhancing our protocols to malicious security in an efficient way might be a challenging problem. Eventually, one might consider trying to optimize the round efficiency of our protocols, which are way more interactive than the garbled circuit approach.

Acknowledgments. We are grateful to David Pointcheval for his fruitful observations and his countless advices.

References

1. Aliasgari, M., Blanton, M., Zhang, Y., Steele, A.: Secure computation on floating point numbers. In: NDSS 2013 (Feb 2013)
2. Aly, A., Cuvelier, E., Mawet, S., Pereira, O., Vyve, M.V.: Securely solving simple combinatorial graph problems. In: Sadeghi, A.R. (ed.) FC 2013. LNCS, vol. 7859, pp. 239–257. Springer (Apr 2013)
3. Aly, A., Vyve, M.V.: Securely solving classical network flow problems. In: ICISC 14. pp. 205–221. LNCS, Springer (2015)

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 13. pp. 535–548. ACM Press (Nov 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" 13). 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. Bektaş, A., Kiraz, M.S., Uzunkol, O.: A secure and efficient protocol for electronic treasury auctions. In: Cryptography and Information Security in the Balkans, pp. 123–140. Springer (2014)
8. Belguechi, R., Alimi, V., Cherrier, E., Lacharme, P., Rosenberger, C.: An overview on privacy preserving biometrics. Recent Application in Biometrics pp. 65–84 (2011)
9. Blanton, M., Saraph, S.: Oblivious maximum bipartite matching size algorithm with applications to secure fingerprint identification. pp. 384–406. LNCS, Springer (2015)
10. 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`
11. 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`
12. Canetti, R.: Universally composable security: A new paradigm for cryptographic protocols. In: 42nd FOCS. pp. 136–145. IEEE Computer Society Press (Oct 2001)
13. Catrina, O., de Hoogh, S.: Improved primitives for secure multiparty integer computation. In: Garay, J.A., Prisco, R.D. (eds.) SCN 10. LNCS, vol. 6280, pp. 182–199. Springer (Sep 2010)
14. 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 (Sep 2010)
15. Catrina, O., Saxena, A.: Secure computation with fixed-point numbers. In: Sion, R. (ed.) FC 2010. LNCS, vol. 6052, pp. 35–50. Springer (Jan 2010)
16. 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)
17. Couteau, G., Peters, T., Pointcheval, D.: Encryption switching protocols. to appear in the proceedings of CRYPTO (2016), `http://eprint.iacr.org/2015/990`
18. Cramer, R., Damgård, I., Nielsen, J.B.: Multiparty computation from threshold homomorphic encryption. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 280–299. Springer (May 2001)
19. 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 (Aug 2007)
20. 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 (Mar 2006)

21. Damgård, I., Geisler, M., Krøigaard, M.: Efficient and secure comparison for on-line auctions. In: Pieprzyk, J., Ghodosi, H., Dawson, E. (eds.) ACISP 07. LNCS, vol. 4586, pp. 416–430. Springer (Jul 2007)
22. Damgård, I., Geisler, M., Krøigaard, M.: A correction to "efficient and secure comparison for on-line auctions". Cryptology ePrint Archive, Report 2008/321 (2008), http://eprint.iacr.org/2008/321
23. Damgård, I., Jurik, M.: A generalisation, a simplification and some applications of Paillier's probabilistic public-key system. In: Kim, K. (ed.) PKC 2001. LNCS, vol. 1992, pp. 119–136. Springer (Feb 2001)
24. Erkin, Z., Franz, M., Guajardo, J., Katzenbeisser, S., Lagendijk, I., Toft, T.: Privacy-preserving face recognition. In: Privacy Enhancing Technologies. pp. 235–253. Springer (2009)
25. Erkin, Z., Veugen, T., Toft, T., Lagendijk, R.L.: Generating private recommendations efficiently using homomorphic encryption and data packing. Information Forensics and Security, IEEE Transactions on 7(3), 1053–1066 (2012)
26. Even, S., Goldreich, O., Lempel, A.: A randomized protocol for signing contracts. In: Chaum, D., Rivest, R.L., Sherman, A.T. (eds.) CRYPTO'82. pp. 205–210. Plenum Press, New York, USA (1982)
27. Garay, J.A., 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 (Apr 2007)
28. Gentry, C., Halevi, S., Jutla, C.S., Raykova, M.: Private database access with HE-over-ORAM architecture. In: ACNS 15. pp. 172–191. LNCS, Springer (2015)
29. 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)
30. Goldreich, O., Micali, S., Wigderson, A.: How to play any mental game or A completeness theorem for protocols with honest majority. In: Aho, A. (ed.) 19th ACM STOC. pp. 218–229. ACM Press (May 1987)
31. Goldreich, O., Micali, S., Wigderson, A.: How to prove all NP-statements in zero-knowledge, and a methodology of cryptographic protocol design. In: Odlyzko, A.M. (ed.) CRYPTO'86. LNCS, vol. 263, pp. 171–185. Springer (Aug 1987)
32. Goodrich, M.T.: Randomized shellsort: A simple oblivious sorting algorithm. In: Charika, M. (ed.) 21st SODA. pp. 1262–1277. ACM-SIAM (Jan 2010)
33. 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)
34. 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
35. Hamada, K., Kikuchi, R., Ikarashi, D., Chida, K., Takahashi, K.: Practically efficient multi-party sorting protocols from comparison sort algorithms. In: Information Security and Cryptology–ICISC 2012, pp. 202–216. Springer (2012)
36. Hazay, C., Toft, T.: Computationally secure pattern matching in the presence of malicious adversaries. Journal of Cryptology 27(2), 358–395 (Apr 2014)
37. Huang, Q., Gui, Y., Wu, F., Chen, G., Zhang, Q.: A general privacy-preserving auction mechanism for secondary spectrum markets (2015)
38. Huang, Y., Evans, D., Katz, J.: Private set intersection: Are garbled circuits better than custom protocols? In: NDSS 2012 (Feb 2012)
39. 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 (Aug 2003)

40. Ishai, Y., Prabhakaran, M., Sahai, A.: Founding cryptography on oblivious transfer - efficiently. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 572–591. Springer (Aug 2008)
41. Jónsson, K.V., Kreitz, G., Uddin, M.: Secure multi-party sorting and applications. Cryptology ePrint Archive, Report 2011/122 (2011), http://eprint.iacr.org/2011/122
42. Kilian, J.: Founding cryptography on oblivious transfer. In: 20th ACM STOC. pp. 20–31. ACM Press (May 1988)
43. Kolesnikov, V., Kumaresan, R.: Improved OT extension for transferring short secrets. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013, Part II. LNCS, vol. 8043, pp. 54–70. Springer (Aug 2013)
44. 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 09. LNCS, vol. 5888, pp. 1–20. Springer (Dec 2009)
45. 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, Part II. LNCS, vol. 5126, pp. 486–498. Springer (Jul 2008)
46. Laud, P.: A private lookup protocol with low online complexity for secure multiparty computation. In: ICICS 14. pp. 143–157. LNCS, Springer (2015)
47. Li, P., Li, T., Yao, Z.A., Tang, C.M., Li, J.: Privacy-preserving outsourcing of image feature extraction in cloud computing. Soft Computing pp. 1–11 (2016)
48. Liedel, M.: Secure distributed computation of the square root and applications. In: Information Security Practice and Experience, pp. 277–288. Springer (2012)
49. Lipmaa, H., Toft, T.: Secure equality and greater-than tests with sublinear online complexity. In: Fomin, F.V., Freivalds, R., Kwiatkowska, M.Z., Peleg, D. (eds.) ICALP 2013, Part II. LNCS, vol. 7966, pp. 645–656. Springer (Jul 2013)
50. Mu, B.: A survey on secure processing of similarity queries (2014)
51. Naor, M., Pinkas, B.: Efficient oblivious transfer protocols. In: Kosaraju, S.R. (ed.) 12th SODA. pp. 448–457. ACM-SIAM (Jan 2001)
52. Nishide, T., Iwamoto, M., Iwasaki, A., Ohta, K.: Secure (m+ 1) st-price auction with automatic tie-break. In: Trusted Systems, pp. 422–437. Springer (2014)
53. 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 (Apr 2007)
54. Paillier, P.: Public-key cryptosystems based on composite degree residuosity classes. In: Stern, J. (ed.) EUROCRYPT'99. LNCS, vol. 1592, pp. 223–238. Springer (May 1999)
55. Pinkas, B., Reinman, T.: Oblivious RAM revisited. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 502–519. Springer (Aug 2010)
56. Rabin, M.: How to exchange secrets by oblivious transfer. Technical Report TR-81, Harvard University, (1981)
57. 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. Dependable and Secure Computing, IEEE Transactions on 11(5), 467–479 (2014)
58. Reistad, T.I., Toft, T.: Secret sharing comparison by transformation and rotation. In: Information Theoretic Security, pp. 169–180. Springer (2007)
59. Sadeghi, A.R., Schneider, T., Wehrenberg, I.: Efficient privacy-preserving face recognition. In: Lee, D., Hong, S. (eds.) ICISC 09. LNCS, vol. 5984, pp. 229–244. Springer (Dec 2010)

60. Samanthula, B.K., Jiang, W., Bertino, E.: Lightweight and secure two-party range queries over outsourced encrypted databases. arXiv preprint arXiv:1401.3768 (2014)
61. Toft, T.: Solving linear programs using multiparty computation. In: Dingledine, R., Golle, P. (eds.) FC 2009. LNCS, vol. 5628, pp. 90–107. Springer (Feb 2009)
62. 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 (Mar 2011)
63. Veugen, T., Blom, F., de Hoogh, S.J., Erkin, Z.: Secure comparison protocols in the semi-honest model. Selected Topics in Signal Processing, IEEE Journal of 9(7), 1217–1228 (2015)
64. Williams, P., Sion, R.: Usable PIR. In: NDSS 2008 (Feb 2008)
65. Williams, P., Sion, R., Carbunar, B.: Building castles out of mud: practical access pattern privacy and correctness on untrusted storage. In: Ning, P., Syverson, P.F., Jha, S. (eds.) ACM CCS 08. pp. 139–148. ACM Press (Oct 2008)
66. 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`
67. Xiang, C., Tang, C.: Privacy-preserving face recognition with outsourced computation. Cryptology ePrint Archive, Report 2014/969 (2014), `http://eprint.iacr.org/2014/969`
68. Yao, A.C.C.: How to generate and exchange secrets (extended abstract). In: 27th FOCS. pp. 162–167. IEEE Computer Society Press (Oct 1986)
69. Yu, C.H.: Sign modules in secure arithmetic circuits. Cryptology ePrint Archive, Report 2011/539 (2011), `http://eprint.iacr.org/2011/539`
70. Yu, C.H., Yang, B.Y.: Probabilistically correct secure arithmetic computation for modular conversion, zero test, comparison, MOD and exponentiation. In: Visconti, I., Prisco, R.D. (eds.) SCN 12. LNCS, vol. 7485, pp. 426–444. Springer (Sep 2012)
71. Zahur, S., Rosulek, M., Evans, D.: Two halves make a whole - reducing data transfer in garbled circuits using half gates. pp. 220–250. LNCS, Springer (2015)
72. Zhang, B.: Generic constant-round oblivious sorting algorithm for MPC. In: Boyen, X., Chen, X. (eds.) ProvSec 2011. LNCS, vol. 6980, pp. 240–256. Springer (Oct 2011)

A Batch ET from Additively Homomorphic Encryption

In this section, we present a batch protocol to efficiently perform simultaneous equality tests. Unlike the other protocols of this article, this construction assumes an additively homomorphic encryption scheme, with a few additional properties. Our protocol share some similarities with the equality test protocol of [28] (which relies on ciphertext packing to amortize the communication of equality tests), and in fact matches the communication complexity of [28], which has to our knowledge the best communication complexity among existing works. However, contrary to [28], we do not need somewhat homomorphic encryption; our protocol can be instantiated with e.g. factorization-based additively homomorphic cryptosystems such as the Paillier scheme [54] or the Damgard-Jurik scheme [23]. For concrete parameters, the amortized communication strongly improves upon every prior ET protocol we know of, including the protocol described Section 3.

A.1 Encryption Scheme

Definition 6. *(Encryption Scheme) An IND-CPA encryption scheme is a tuple of algorithms* $(\mathsf{Setup}, \mathsf{Enc}, \mathsf{Dec})$ *such that:*

- $\mathsf{Setup}(1^\kappa)$ *outputs a key-pair* $(\mathsf{pk}, \mathsf{sk})$*;* pk *implicitly defines a plaintext space* $\mathscr{M}$ *and a ciphertext space* $\mathscr{C}$*.*
- $\mathsf{Enc}(\mathsf{pk}, m)$*, on input* pk *and a plaintext* $m \in \mathscr{M}$*, outputs a ciphertext* $c \in \mathscr{C}$*.*
- $\mathsf{Dec}(\mathsf{sk}, c)$*, on input* sk *and a ciphertext* $c \in \mathscr{C}$*, deterministically outputs a plaintext* $m' \in \mathscr{M}$*.*

In addition, an IND-CPA encryption scheme satisfies the properties of correctness *and* IND-CPA security*, defined below.*

The *correctness* states that decryption is the reverse operation of encryption: for any $(\mathsf{pk}, \mathsf{sk}) \leftarrow_R \mathsf{Setup}(1^\kappa)$, for any $m \in \mathscr{M}$ and any $c \leftarrow_R \mathsf{Enc}(\mathsf{pk}, m)$, $\mathsf{Dec}(\mathsf{sk}, c) = m$. The *IND-CPA security* is defined by considering the following game between an adversary and a challenger:

- The challenger picks $(\mathsf{pk}, \mathsf{sk}) \leftarrow_R \mathsf{Setup}(1^\kappa)$ and sends pk to the adversary.
- The adversary sends $(m_0, m_1) \leftarrow_R \mathscr{M}^2$ to the challenger.
- The challenger picks $b \leftarrow_R \{0, 1\}$ and sends $c \leftarrow_R \mathsf{Enc}(\mathsf{pk}, m_b)$ to the challenger.
- The challenger outputs a guess b' and wins the game if $b' = b$.

An encryption scheme is IND-CPA secure if no polynomial-time adversary can win the game with non-negligible advantage over the random guess.

Additively Homomorphic Encryption Scheme. An encryption scheme is additively homomorpic if there is a law $\boxplus : \mathscr{C}^2 \mapsto \mathscr{C}$ such that for any $(m_0, m_1) \in \mathscr{M}^2$, for any $(c_0, c_1) \leftarrow_R (\mathsf{Enc}(\mathsf{pk}, m_0), \mathsf{Enc}(\mathsf{pk}, m_1))$, $\mathsf{Dec}(\mathsf{sk}, c_0 \boxplus c_1) = m_0 + m_1$. Note that this trivially implies than one can add a constant value to a ciphertext (by first encrypting it and then using $\boxplus$); one can also see that via a square-and-multiply algorithm, given an encryption of some m and an integer λ, one can compute an encryption of λm. We will denote $\bullet$ this external multiplication.

Randomizable Encryption Scheme. A randomizable encryption scheme is an encryption scheme with an additional algorithm Rand which, on input pk and an encryption of some plaintext m, outputs a ciphertext taken uniformly at random in the distribution $\{\mathsf{Enc}(\mathsf{pk}, m)\}$ of encryptions of m.

Expendable Plaintext Space. In our protocol, we require the message space to be of the form $\mathbb{Z}_P$, for some integer $P = 2^{\mathrm{poly}(\kappa)}$. In addition the plaintext space must be *expendable*, in the sense that one can specify a threshold T when calling $\mathsf{Setup}(1^\kappa, T)$, so that the message space $\mathscr{M} = \mathbb{Z}_P$ it specifies is of size $P \geq T$. For example, for the Paillier encryption scheme and its variants, this would simply correspond to taking the modulus bigger than T.

A.2 Batch Equality Test

We let $(\mathsf{Setup}, \mathsf{Enc}, \mathsf{Dec})$ denote a randomizable additively homomorphic encryption scheme with expendable plaintext space. Let n be the number of equality tests to be performed. As there is no possible confusion, we write $\mathsf{Enc}(m)$ for $\mathsf{Enc}(\mathsf{pk}, m)$.

Inputs: n pairs of ℓ-bit strings $(x^{(i)}, y^{(i)})_{i \leq n}$.

Outputs: n bits $(b_i^A)_{i \leq n}$ for Alice, and n bits $(b_i^B)_{i \leq n}$ for Bob, such that for all $i \leq n$, $b_i^A \oplus b_i^B = [x^{(i)} \leq y^{(i)}]$.

Batch reduction: In this step, Alice and Bob rely on the additively homomorphic encryption scheme to compute shares of the Hamming distances between each $x^{(i)}, y^{(i)}$, modulo coprime integers p_i.

- Let $(p_0, \cdots p_{n-1})$ be the n smallest pairwise coprime numbers such that $p_0 > \ell$; let $M \leftarrow \prod_i p_i$. Alice calls $\mathsf{Setup}(1^\kappa, 2^{\kappa+2|M|+2})$ and gets $(\mathsf{pk}, \mathsf{sk})$; pk implicitely defines a plaintext space $\mathbb{Z}_P$ of size $P \geq 2^{\kappa+2|M|+2}$. For $j = 0$ to $\ell - 1$, let $x_j \in \mathbb{Z}_M$ (resp. y_j) be the smallest integer satisfying $x_j = x^{(i)}[j] \bmod p_i$ (resp. $y_j = y^{(i)}[j] \bmod p_i$) for every $i \leq n - 1$. Alice sends $c_j \leftarrow_R \mathsf{Enc}(x_j)$ for $j = 0$ to $\ell - 1$ to Bob.
- For $j = 0$ to $\ell - 1$, Bob picks $r_j \leftarrow_R \mathbb{Z}_{2^{\kappa+2}M^2}$, computes and sends $c'_j \leftarrow_R \mathsf{Rand}(\mathsf{pk}, y_j \bullet c_j \boxplus r_j)$ to Alice, who decrypts all the ciphertexts to get some values s_j.
- For $j = 0$ to $\ell - 1$, Alice sets $\sigma_j \leftarrow s_j \bmod M$ and Bob sets $\rho_j \leftarrow r_j \bmod M$. Note that it holds that for all $(i, j) \in [n-1] \times [\ell - 1]$,

$$2(\rho_j - \sigma_j) + x^{(i)}[j] + y^{(i)}[j] = x^{(i)}[j] \oplus y^{(i)}[j] \bmod p_i$$

  Hence, $(-2\sigma_j + x^{(i)}[j] \bmod p_i)$ and $(2\rho_i + y^{(i)}[j] \bmod p_i)$ form shares of the bits of $x^{(i)} \oplus y^{(i)}$ modulo p_i.
- Alice computes $\alpha_i \leftarrow \sum_{j=0}^{\ell-1} -2\sigma_j + x^{(i)}[j] \bmod p_i$ and Bob computes $\beta_i \leftarrow \sum_{j=0}^{\ell-1} -2\rho_j + y^{(i)}[j] \bmod p_i$. Note that as $p_i > \ell$ is greater than the Hamming distance H_d between $x^{(i)}$ and $y^{(i)}$, it holds that $\alpha_i + \beta_i = H_d(x^{(i)}, y^{(i)})$, which is 0 if and only if $x^{(i)} = y^{(i)}$. Hence, seeing from now on α_i and β_i as integers, the problem was reduced to finding whether $\alpha_i = p_i - \beta_i$, which are strings of size $O(\log \ell \log\log \ell)$.

Reduced Equality Test: Alice and Bob perform n ET with respective input size $|p_i|$, on respective inputs $(\alpha_i, p_i - \beta_i)$, to get the n outputs of the protocol.

Note that as for our protocol Section 3, this protocol can be executed on random inputs in a preprocessing phase; the online phase is then essentially the same than our previous ET protocol.

Intuition of the Protocol. The protocol exploits the following observation: given an index $j < \ell$, computing shares of $(x^{(i)}[j] \oplus y^{(i)}[j])_{i \leq n}$ (modulo various coprime numbers) can be reduced to performing a single multiplication protocol modulo $M = \prod_i p_i$. This protocol is performed over the integers by using an additively

homomorphic scheme of sufficiently large plaintext space, the resulting shares masking statistically the result over the integer. The players then get all the shares of the $(x^{(i)}[j] \oplus y^{(i)}[j])_{i \leq n}$ by reducing there shares modulo M and using the chinese remainder theorem on there shares. This reduces n equality tests on ℓ-bit strings to n equality tests on strings of sizes ranging from $|p_1 + 1|$ to $|p_n + 1|$. As this method does not allow to reduce further the size of the inputs, n OT-based equality tests are then called in parallel on the reduced inputs.

Communication. The batch reduction involves 2ℓ ciphertexts, hence a total of $2\ell|\mathscr{C}|$ bits. Under the extended Riemann hypothesis, the nth prime number larger than ℓ is of size $O(\log(\ell + n \log n))$, hence $M = O(n \log(\ell + n \log n))$. Under this assumption, the n reduced equality tests transmit $O(n\kappa \log(\ell + n \log n)/\log \kappa)$ bits.

Most additively homomorphic that satisfy our requirements have ciphertexts of size $O(k + \kappa)$ for k-bit inputs with large enough k; taking this condition in account, the amortized communication becomes $O(\ell \log \kappa + \kappa)$ bits.

Table 4: Amortized communication of ℓ-bit ET over n executions

	Damgard-Jurik based ET				ET of Section 3	
	$n = 1000$		$n = 100$			
ℓ	length	rounds	length	rounds	length	rounds
			Preprocessing Phase			
16	3339 bits	4 rounds	3183 bits	4 rounds	2945 bits	4 rounds
32	4199 bits	4 rounds	4568 bits	4 rounds	5212 bits	4 rounds
64	5913 bits	4 rounds	7275 bits	4 rounds	9863 bits	4 rounds
128	9342 bits	4 rounds	12670 bits	4 rounds	20194 bits	4 rounds
			Online Phase			
16	96 bits	3 rounds	81 bits	3 rounds	54 bits	3 rounds
32	129 bits	3 rounds	115 bits	3 rounds	88 bits	3 rounds
64	193 bits	3 rounds	181 bits	3 rounds	154 bits	3 rounds
128	321 bits	3 rounds	313 bits	3 rounds	300 bits	3 rounds

Concrete Efficiency. We now estimate the concrete efficiency of our protocol, and compare it to our previous solution. We use the Damgard-Jurik generalization [23] of the Paillier encryption scheme, which enjoys better ciphertext over plaintext size ratio as the size of the plaintext space increases. More precisely, the Damgard-Jurik cryptosystem for an RSA modulus N is parametrized with an integer s, so that its plaintext space is $\mathbb{Z}_{N^s}$, and its ciphertext space is $\mathbb{Z}_{N^{s+1}}$. We consider a 2048-bit RSA modulus, as recommended by the NIST standard,

and set arbitrarily the number n of parallel ETs to 100 and 1000 respectively. For the oblivious transfers, we use $\kappa = 128$.

For each value of ℓ in the table, s is taken to be the smallest integer such that $s \cdot 2048 \geq 2 \log \mathsf{P_n}(\ell) + 129$, where $\mathsf{P_n}(\ell)$ is the product of the smallest n pairwise coprime numbers, starting with $\ell + 1$. Each ciphertext is of size $(s + 1) \cdot 2048$. Table 4 indicates the average number of bits transmitted per ET. The actual value of ℓ has very little influence on s; in fact, $s = 12$ is the optimal parameters for all the values of ℓ that we consider (hence the ciphertexts are of size 26624 bits). With those parameters, n ET on ℓ-bit strings are reduced to n ET on strings of bit-size 6 to 13 (as all the p_i are different, the reduction gives different bit-sizes); experimentally, it turns out that this improves over our OT-base ET for $\ell > 16$.

B Security Proofs

B.1 Ideal Functionalities for Equality Test

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, ℓ) from both players, the functionality picks $(x, y) \leftarrow_R (\mathbb{Z}_2^\ell)^2$ and sets $(\boldsymbol{a}, \boldsymbol{b}) \leftarrow_R \langle x \oplus y \rangle_{\ell+1}$. $\mathscr{F}_{\mathsf{ET\text{-}prep}}$ 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. 3: Ideal Functionalities for Equality Test and Preprocessing

The ideal functionalities $\mathscr{F}_{\mathsf{ET}}, \mathscr{F}_{\mathsf{ET\text{-}prep}}$ are recalled Figure 3. We recall the protocol $\Pi_{\mathsf{ET\text{-}prep}}$:

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 $(\boldsymbol{a}[i] + x[i] \bmod \ell + 1, \boldsymbol{a}[i] + 1 - x[i] \bmod \ell +$

$1)_{i\leq \ell}$ 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 $(a[i], a[i] \oplus x[i])_{i\leq n}$ for Alice and y for Bob. Let b denote Bob's output. Alice outputs (x, a) and Bob outputs (y, b).

B.2 Security Analysis of $\Pi_{\mathsf{ET\text{-}prep}}$

We now prove the following theorem:

Theorem 2 (Repeated from Section 3)**.** *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.*

Let $\diamond$ denote the composition operator; let Π be the protocol such that $\Pi \diamond \mathscr{F}_{\mathsf{ET\text{-}prep}} = \Pi_{\mathsf{ET}}$. The natural way of proving Theorem 2 would be to prove that $\Pi_{\mathsf{ET\text{-}prep}}$ UC implements $\mathscr{F}_{\mathsf{ET\text{-}prep}}$ in the $\mathscr{F}_{\mathsf{OT}}$-hybrid model; as $\Pi \diamond \mathscr{F}_{\mathsf{ET\text{-}prep}}$ UC implements $\mathscr{F}_{\mathsf{ET}}$, it would follow from the composition theorem of the UC framework that $\Pi \diamond \Pi_{\mathsf{ET\text{-}prep}}$ UC implements $\mathscr{F}_{\mathsf{ET}}$ in the $\mathscr{F}_{\mathsf{OT}}$-hybrid model.

However, this approach fails here. The reason is that $\Pi_{\mathsf{ET\text{-}prep}}$ does in fact *not* UC implement $\mathscr{F}_{\mathsf{ET\text{-}prep}}$. To understand the issue, recall that a proof in the UC framework is done by exhibiting a simulator with access to the functionality (here $\mathscr{F}_{\mathsf{ET\text{-}prep}}$), so that the view produced by the simulator *together with the outputs of the corrupted parties* are indistinguishable from the view and the outputs in a real execution of the protocol. The simulator extracts the inputs of the corrupted parties, queries the ideal functionality on those inputs, and somehow forces the corrupted players to obtain the same outputs. Observe now that in the protocol $\Pi_{\mathsf{ET\text{-}prep}}$, parts of the outputs of both Alice and Bob are picked by the player themselves, independently of the behaviour of their opponent. Therefore, no simulator can possibly ensure that the corrupted player will compute the same output than what was returned by $\mathscr{F}_{\mathsf{ET\text{-}prep}}$. This is a bit counterintuitive, as the protocol $\Pi_{\mathsf{ET\text{-}prep}}$ clearly does "exactly what $\mathscr{F}_{\mathsf{ET\text{-}prep}}$ does" (recall that the players are semi-honest, hence they honestly follow the specifications of the protocol). We could solve this by adding some resharing step in $\Pi_{\mathsf{ET\text{-}prep}}$, but this would noticeably increase the communication of our protocol. Fortunately, this exact issue was handled in great details in [10]. Therefore, we start by recalling the results of [10], and use them to complete the proof of Theorem 2.

Input Private Protocols. The authors of [10] extend the universal composability framework, by defining the notion of *input-privacy*. Informally, a protocol is input-private if there is a simulator (with access to the corresponding ideal functionality) which produces a view indistinguishable from an execution of the protocol for any environment *that completely ignores the output of the protocol.*

We refer to [10, Section 4] for a formal definition. The core result is a composition theorem, which states that an *ordered* composition $\Pi_\diamond$ of an input-private protocol Π_{ip} and a UC secure protocol Π_{UC} is a UC secure protocol (the model considered is that of static, passive corruption, as in the present paper). The term "ordered" refers to a restricted class of composition, in which there is a one-way communication from Π_{ip} to Π_{UC}. An additional requirement is that the outputs of $\Pi_\diamond$ must all come from Π_{UC} – in other words, Π_{ip} does not output any value in the composed protocol, but only intermediate random values, perfectly independent of the outputs. Intuitively, this result is tailored to protocols Π_{ip} that return random shares of some output, as the view of a corrupted player can be simulated without the output.

Security. The protocol $\Pi \diamond \Pi_{\mathsf{ET\text{-}prep}}$ immediatly satisfies the constraint of ordered composition with predicable output specified in [10] (outputs from $\Pi_{\mathsf{ET\text{-}prep}}$ are used in Π_{ET}, but there is no data dependency in the other direction). The functionality $\mathscr{F}_{\mathsf{ET\text{-}prep}}$ can be virtually seen as a functionality that does not produce outputs, but only stores the result of some computation (this storage being represented by secret shares in the real protocol). The input-privacy of $\Pi_{\mathsf{ET\text{-}prep}}$ with respect to $\mathbb{F}_{\mathsf{ET\text{-}prep}}$ in the $\mathscr{F}_{\mathsf{OT}}$-hybrid model is straightforward: it is trivial to see that the protocol is correct, and no messages are exchanged between the players during the protocol, which consists entirely in calls made to $\mathscr{F}_{\mathsf{OT}}$. Therefore, the simulator simply runs a local copy of $\mathscr{F}_{\mathsf{OT}}$ and answers honestly to the queries of the players. The view of any environment that ignores the output of the protocol contains only the responses of $\mathscr{F}_{\mathsf{OT}}$ to the corrupted player, hence the simulation is trivially perfect.

By the input-privacy of $\Pi_{\mathsf{ET\text{-}prep}}$, and the ordered composition with predictable output of $\Pi \diamond \Pi_{\mathsf{ET\text{-}prep}}$, as $\Pi \diamond \mathscr{F}_{\mathsf{ET\text{-}prep}}$ UC implements $\mathscr{F}_{\mathsf{ET}}$, the theorem 2 from [10, Section 7.2] (which is the composition theorem mentionned above) shows that $\Pi \diamond \Pi_{\mathsf{ET\text{-}prep}}$ UC implements $\mathscr{F}_{\mathsf{ET}}$ in the $\mathscr{F}_{\mathsf{OT}}$-hybrid model.

B.3 Ideal Functionalities for Secure Comparison

We restate Figure 4 the ideal functionalities $\mathscr{F}_{\mathsf{SC}}, \mathscr{F}_{\mathsf{SC\text{-}prep}}, \mathscr{F}_{\mathsf{compress}}$ given Section 4.

B.4 Security Analysis of Π_{SC}

We recall the protocol Π_{SC}, parametrized by (ℓ, n):

Protocol Π_{SC}

Initialize: Both players call $\mathscr{F}_{\mathsf{SC\text{-}prep}}$ on input (PS, n) to get (ρ, a) and (σ, b).
Secure Comparison: On input two ℓ-bit integers, x from Alice and y from Bob, let $i \leftarrow 1$, $x_i \leftarrow x$, and $y_i \leftarrow y$.

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

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

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

The functionality runs with two parties, Alice and Bob. Upon receiving $(\mathsf{compress}, \ell, \lambda, x)$ from Alice and $(\mathsf{compress}, \ell, \lambda, y)$ from Bob, where $\lambda < \ell$ and (x, y) are ℓ-bit long, it picks two uniformly random values $(\hat{x}, \hat{y}) \leftarrow_R \mathbb{Z}_{2^{\lambda+1}}^2$ such that $\hat{x} \leq \hat{y}$ if and only if $x \leq y$. It returns $\hat{x}$ to Alice and $\hat{y}$ to Bob.

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

The functionality runs with two parties, Alice and Bob.

Size Reduction: Upon receiving $(\mathsf{SR}, \lambda, \mu)$ from both players, the functionality picks $(\boldsymbol{s_0}, \boldsymbol{s_1}, \boldsymbol{t_0}, \boldsymbol{t_1}, \boldsymbol{u_0}, \boldsymbol{u_1}) \leftarrow_R (\mathbb{Z}_{\mu+1}^{\mu})^2 \times (\mathbb{Z}_{2^{\lambda+1}}^{\mu})^4$ and $(c, d, e) \leftarrow_R (\mathbb{Z}_2^{\mu})^4$. $\mathscr{F}_{\mathsf{SC\text{-}prep}}$ sets

$$(\boldsymbol{s}, \boldsymbol{t}, \boldsymbol{u}) \leftarrow \left(\left(\boldsymbol{s_{c[i]}}[i]\right)_{i \leq \mu}, \left(\boldsymbol{t_{d[i]}}[i]\right)_{i \leq \mu}, \left(\boldsymbol{u_{e[i]}}[i]\right)_{i \leq \mu} \right)$$

It outputs $(e, \boldsymbol{s_0}, \boldsymbol{s_1}, \boldsymbol{t_0}, \boldsymbol{t_1}, \boldsymbol{u})$ to Alice and $(c, d, \boldsymbol{s}, \boldsymbol{t}, \boldsymbol{u_0}, \boldsymbol{u_1})$ to Bob.

Product Sharing: Upon receiving (PS, n) from both players, the functionality picks $(\rho, \sigma) \leftarrow_R (\mathbb{Z}_2^{2^n-1})^2$ and sets $(a, b) \leftarrow_R \langle \rho * \sigma \rangle_2$. $\mathscr{F}_{\mathsf{ET\text{-}prep}}$ outputs (ρ, a) to Alice and (σ, b) to Bob.

Fig. 4: Ideal Functionalities for Secure Comparison and Preprocessing

1. The players agree on two integers λ_i, μ_i such that $\lambda_i \leq \ell$ and μ_i is the smallest integer satisfying $\lambda_i \mu_i \geq \ell$. Alice calls $\mathscr{F}_{\mathsf{compress}}$ on input $(\mathsf{compress}, \ell, \lambda_i, x_i)$, and Bob on input $(\mathsf{compress}, \ell, \lambda_i, y_i)$. Let (x_{i+1}, y_{i+1}) denote their respective outputs. The players set $\ell \leftarrow \lambda_i$ and $i \leftarrow i + 1$. If $|\ell| > n - 1$, the players iterate the step 1.
2. Alice sets $x_i' \leftarrow x_i - 1$. Let $f : (j, l) \mapsto l - 1 + 2^{j-1}$. For $j = 1$ to $n-1$, let $(I_l^j)_{1 \leq l \leq 2^j}$ denote the list of subsets of $\{1, \cdots, j\}$ (in any arbitrary fixed order). For $j = 1$ to n, for $l = 1$ to 2^{j-1}, Alice picks $\alpha \leftarrow_R \{0, 1\}$ and sets $\alpha_{jl} \leftarrow \rho[f(j,l)] \oplus x_i'[j] \cdot \prod_{k \in I_l^{j-1}} (1 \oplus x_i'[k])$, and Bob picks $\beta \leftarrow_R \{0, 1\}$ and sets $\beta_{jl} \leftarrow \sigma[f(j,l)] \oplus (1 \oplus y_i[j]) \cdot \prod_{k \notin I_l^{j-1}} y_i[k]$.[6] Alice sends $(\alpha, (\alpha_{jl})_{jl})$ and Bob sends $(\beta, (\beta_{jl})_{jl})$ (this amounts to 2^{n+1} bits exchanged).
3. Alice outputs

$$\bigoplus_{\substack{j \leq n \\ l \leq 2^j - 1}} \rho[f(j,l)]\beta_{jl} \oplus a[f(j,l)] \oplus \alpha \oplus \beta$$

[6] By convention, an empty product is equal to 1.

Bob outputs

$$\bigoplus_{\substack{j\leq n\\ l\leq 2^j-1}} (\sigma[f(j,l)] \oplus \beta_{jl})\alpha_{jl} \oplus b[f(j,l)] \oplus \alpha \oplus \beta$$

We now prove the following theorem:

Theorem 3 (Repeated from Section 4)**.** *The protocol Π_{SC} securely implements $\mathscr{F}_{\mathsf{SC}}$ in the $(\mathscr{F}_{\mathsf{SC\text{-}prep}}, \mathscr{F}_{\mathsf{compress}})$-hybrid model, with respect to passive corruption.*

Let Adv be an adversary that interacts with Alice and Bob, running the protocol Π_{SC}. We will construct a simulator $\mathscr{S}im$ which interacts with $\mathscr{F}_{\mathsf{SC}}$, so that no environment Z can distinguish an interaction with Adv in Π_{SC} from an interaction with $\mathscr{S}im$ in the ideal world. $\mathscr{S}im$ starts by invoking a copy of Adv. Each time $\mathscr{S}im$ received from Z an input value, he writes it on Adv's input tape as if coming from Z. Each time Adv writes on its output tape, $\mathscr{S}im$ writes the same thing on his output tape.

One Player is Corrupted. We focus here on the case of a corrupted Bob; as the protocol is perfectly symmetrical, the simulation is similar for a corrupted Alice.

Initialize: $\mathscr{S}im$ runs local copies of $(\mathscr{F}_{\mathsf{compress}}, \mathscr{F}_{\mathsf{SC\text{-}prep}})$. He honestly answers to the call to the PS command, and stores the output. (This step does not require the input of Alice)

Secure Comparison:

1. When $\mathscr{S}im$ receives $(\mathsf{compress}, \ell, \lambda_1, y_1)$ from Bob, he stores $y = y_1$ and sends (SC, y) to $\mathscr{F}_{\mathsf{SC}}$ on behalf of Bob in the ideal world. $\mathscr{S}im$ receives an output bit T. For each compression round, $\mathscr{S}im$ queries $\mathscr{F}_{\mathsf{compress}}$ with random inputs of the appropriate size on behalf of Alice.
2. When $\mathscr{S}im$ receives $(\beta, (\beta_{jl})_{jl})$, he retrieves Bob's output (σ, b) to the PS command, and picks uniformly random bits $(\alpha_{jl})_{j\leq n-1, l\leq 2^{j-1}}$. $\mathscr{S}im$ sets

$$\alpha \leftarrow \bigoplus_{\substack{j\leq n\\ l\leq 2^j-1}} \rho[f(j,l)]\beta_{jl} \oplus a[f(j,l)] \oplus T \oplus \beta$$

and sends $(\alpha, (\alpha_{jl})_{jl})$ to Bob.

Remaining Cases. When both parties are corrupted, $\mathscr{S}im$ simply runs Adv internally. When neither party is corrupted, $\mathscr{S}im$ internally runs Alice and Bob honestly, with inputs $(0, 0)$, and forwards the messages exchanged to Adv.

Indistinguishability. We focus on the case were Bob is corrupted; the argument follows symmetrically for a corrupted Alice, and is straightforward when both players are corrupted, or no player is corrupted. We show that the joint view of Z and Adv in the real world is indistinguishable from the view of Z and the simulated Adv in the ideal world. As no messages are exchanged in step 1, we must only deal with the view produced in step 2. During an execution of the real protocol, the environment will see $(\alpha, (\alpha_{jl})_{jl})$, as well as the outputs (o_A, o_B) of the players. In the ideal world, on input (x, y), the outputs of $\mathscr{F}_{\mathsf{SC}}$ are random shares of $[x \leq y]$. We must therefore first show that this holds in the real protocol too. Recall that the players are semi-honest: they follow the specifications of the protocol.

Claim. $o_A \oplus o_B = [x \leq y]$.

Let t be the number of repetitions of step 1. Let $x'_t \leftarrow x_t - 1$. By definition of $\mathscr{F}_{\mathsf{compress}}$, $[x'_t < y_t] = [x_t \leq y_t] = [x \leq y]$. We now show that $o_A \oplus o_B = [x'_t < y_t]$. Replacing $(\alpha_{jl}, \beta_{jl})_{jl}$ by the corresponding expression, and using the fact that $\rho[l]\sigma[l] = a[l] \oplus b[l]$, we get

$$o_A \oplus o_B = \bigoplus_{j \leq n} x'_t[j](1 \oplus y_t[j]) \left(\bigoplus_{l \leq 2^j - 1} \prod_{k \in I_l^{j-1}} (1 \oplus x'_t[k]) \cdot \prod_{k \notin I_l^{j-1}} y_t[k] \right)$$

The term between the parenthesis is simply the product $\prod_{k \leq j-1}(1 \oplus x_t[k] \oplus y_t[t])$ developed. This product evaluates to 1 if and only if it hold for each $k \leq j-1$ that $x_t[k] = y_t[k]$ (which is equivalent to $1 \oplus x_t[k] \oplus y_t[k] = 1$). Observe now that $[x'_t < y_t]$ can be computed recursively using the following formula:

$$[x'_t < y_t] = [x'_t[1] < y_t[1]] \oplus [x'_t[1] = y_t[1]] \cdot [x'_t[2] \cdots x'_t[n] < y_t[2] \cdots y_t[n]]$$

As for any $j \leq n$, $[x'_t[j] < y_t[j]] = (1 \oplus x'_t[j])y_t[j]$ and $[x'_t[j] = y_t[j]] = 1 \oplus x'_t[j] \oplus y_t[j]$, recursively applying the above formula gives

$$[x'_t < y_t] = \bigoplus_{j \leq n} x'_t[j](1 \oplus y_t[j]) \left(\prod_{k=1}^{j-1} (1 \oplus x'_t[k] \oplus y_t[k]) \right)$$

Which concludes the proof of the claim. Moreover, each value α_{jl} sent during the protocol is perfectly masked by $\rho[f(j,l)]$, hence all the α_{jl} are perfectly indistinguishable from uniformly random values, and all the simulated α_{jl} are uniformly random bits. α is random in the real protocol, and is masked by the output T of $\mathscr{F}_{\mathsf{SC}}$ in the simulated protocol, which is a uniformly random bit by definition of $\mathscr{F}_{\mathsf{SC}}$. It is straightforward to see that the semi-honest Bob will indeed obtain the bit T as output in the simulated protocol. Therefore, the joint view of Z and Adv in the real protocol is perfectly indistinguishable from their joint view in the simulated protocol.

B.5 Security Analysis of Π_{compress}

We recall the protocol Π_{compress}:

Protocol Π_{compress}

Let (ℓ, λ) be two integers such that $\lambda \leq \ell$. Let μ be the smallest integer such that $\lambda\mu \geq \ell$. On input x from Alice and y from Bob, both of size ℓ-bit,

Initialize: The players call $\mathscr{F}_{\mathsf{SC\text{-}prep}}$ on input $(\mathsf{SR}, \lambda, \mu)$ to get outputs $(e, \boldsymbol{s_0}, \boldsymbol{s_1}, \boldsymbol{t_0}, \boldsymbol{t_1}, \boldsymbol{u})$ for Alice and $(c, d, \boldsymbol{s}, \boldsymbol{t}, \boldsymbol{u_0}, \boldsymbol{u_1})$ for Bob.

Compression: Let $(x_j)_{j\leq\mu} \in \mathbb{Z}_{2^\lambda}$ (resp. $(y_j)_{j\leq\mu} \in \mathbb{Z}_{2^\lambda}$) be the decomposition of x (resp. y) into μ blocs of size λ (i.e., $x = \sum_{j=1}^{\mu} x_j 2^{\lambda(j-1)}$ and $y = \sum_{j=1}^{\mu} y_j 2^{\lambda(j-1)}$). The players perform the following operations:

1. For $j = 1$ to μ, the players call $\mathscr{F}_{\mathsf{ET}}$ on inputs (ET, x_j) and (ET, y_j). Let $(\alpha_j, \beta_j)_{j\leq\mu} \in \mathbb{Z}_2^{2\mu}$ denote their respective outputs.
2. Alice picks $\boldsymbol{r} \leftarrow_R \mathbb{Z}_{\mu+1}^{\mu}$. For $j = 1$ to μ, Bob sends $\beta_j \oplus c[j]$ to Alice. If this is 0, Alice sends $(\boldsymbol{s_0}[j] + \alpha_j + \boldsymbol{r}[j] \bmod \mu + 1, \boldsymbol{s_1}[j] + 1 - \alpha_j + \boldsymbol{r}[j] \bmod \mu + 1)$; else, she sends this pair in permuted order. This allows Bob to recover $\boldsymbol{r}[j] + (\alpha_j \oplus \beta_j) \bmod \mu + 1$. For $j = 1$ to μ, Alice sets $x'_j \leftarrow \sum_{k=1}^{j} \boldsymbol{r}[k] \bmod \mu + 1$ and Bob sets $y'_j \leftarrow \sum_{k=1}^{j} \boldsymbol{r}[k] + (\alpha_k \oplus \beta_k) \bmod \mu + 1$. Observe that $(x'_j, y'_j) \in \mathbb{Z}_{\mu+1}^2$.
3. For $j = 1$ to μ, the players call $\mathscr{F}_{\mathsf{ET}}$ on inputs (ET, x'_j) and (ET, y'_j). Let $(\alpha'_j, \beta'_j)_{j\leq\mu} \in \mathbb{Z}_2^{2\mu}$ denote their respective outputs and $(\alpha'_0, \beta'_0) \leftarrow (0, 0)$. For $j = 1$ to μ_i, Alice sets $\gamma_j \leftarrow \alpha_{j-1} \oplus \alpha_j$ and bob sets $\delta_j \leftarrow \beta_{j-1} \oplus \beta_j$. The following steps 4 and 5 are executed in parrallel:
4. Alice picks $\boldsymbol{r_A} \leftarrow_R \mathbb{Z}_{2^{\lambda+1}}^{\mu}$. For $j = 1$ to μ, Bob sends $\delta_j \oplus d[j]$ to Alice. If this is 0, Alice sends $(\boldsymbol{t_0}[j] + \gamma_j x_j + \boldsymbol{r_A}[j] \bmod 2^{\lambda+1}, \boldsymbol{t_1}[j] + (1 - \gamma_j)x_j + \boldsymbol{r_A}[j] \bmod 2^{\lambda+1})$; else, she sends this pair in permuted order. This allows Bob to recover $\boldsymbol{r_A}[j] + (\gamma_j \oplus \delta_j)x_j \bmod 2^{\lambda+1}$.
5. Bob picks $\boldsymbol{r_B} \leftarrow_R \mathbb{Z}_{2^{\lambda+1}}^{\mu}$. For $j = 1$ to μ, Alice sends $\gamma_j \oplus e[j]$ to Bob. If this is 0, Bob sends $(\boldsymbol{u_0}[j] + \delta_j y_j + \boldsymbol{r_B}[j] \bmod 2^{\lambda+1}, \boldsymbol{u_1}[j] + (1 - \delta_j)y_j + \boldsymbol{r_B}[j] \bmod 2^{\lambda+1})$; else, he sends this pair in permuted order. This allows Alice to recover $\boldsymbol{r_B}[j] + (\gamma_j \oplus \delta_j)y_j \bmod 2^{\lambda+1}$.

Output: Alice outputs $\hat{x} \leftarrow 2^\lambda + \sum_{j=1}^{\mu} \boldsymbol{r_A}[j] + \boldsymbol{r_B}[j] + (\gamma_j \oplus \delta_j)y_j \bmod 2^{\lambda+1}$ and Bob outputs $\hat{y} \leftarrow \sum_{j=1}^{\mu} \boldsymbol{r_B}[j] + \boldsymbol{r_A}[j] + (\gamma_j \oplus \delta_j)x_j \bmod 2^{\lambda+1}$.

We now prove the following theorem:

Theorem 4 (Repeated from Section 4)**.** *The protocol Π_{compress} securely implements $\mathscr{F}_{\mathsf{compress}}$ in the $(\mathscr{F}_{\mathsf{ET}}, \mathscr{F}_{\mathsf{SC\text{-}prep}})$-hybrid model, with respect to passive corruption of at most one of the players.*

We first look at the correctness; the simulation will then be rather straightforward. The general idea of the protocol is that to compare two strings, it suffices

to divide these strings in blocs, and to compare the first block on which they differ. The purpose of the compression step is for the players to obliviously select this block. The inputs (x, y) are first divided into μ blocks of size λ. At the end of step 1, the players obtain shares (α_j, β_j) (over $\mathbb{Z}_2$) of all the bits $[x_j = y_j]$.

During step 2, the players compute values (x'_j, y'_j) whose difference modulo $\mu + 1$ is $\sum_{k=1}^{j}[x_k = y_k]$. This requires to use some preprocessed material. Let j^* be the first block on which x differs from y. Observe that $\sum_{k=1}^{j}[x_k = y_k] = 0$ for $j < j^*$, and $\sum_{k=1}^{j}[x_k = y_k] > 0$ afterward.

The players perform in step 3 equality tests on the values (x'_j, y'_j). Therefore, they obtain shares of the bits $[x'_i - y'_j = 0]$. Observe that these bits are 1 for $j \leq j^*$, and 0 afterward. From these shares, the players can locally compute shares (γ_j, δ_j) of bits which are 0 for every $j \neq j^*$, and 1 for $j = j^*$ (If $x = y$, the shares (γ_j, δ_j) will be shares of 0 for every j).

In step 4 and 5, using preprocessed material again, the players can compute shares of $\sum_{j=1}^{\mu}(\gamma_j \oplus \delta_j)x_j = x_{j^*}$ (denoted $(x^A_{j^*}, x^B_{j^*})$) and $\sum_{j=1}^{\mu}(\gamma_j \oplus \delta_j)y_j = y_{j^*}$ (denoted $(y^A_{j^*}, y^B_{j^*})$). We know that $[x \leq y] = [x_{j^*} \leq y_{j^*}]$, but (x_{j^*}, y_{j^*}) are not privately known to each player: they are *secretly shared* between the players. However, note that the shares of the values are computed modulo $2^{\lambda+1}$, and it necessarily holds that $x_{j^*} \leq 2^\lambda$ (resp. $y_{j^*} \leq 2^\lambda$). Therefore, we can easily show that

$$[x_{j^*} \leq y_{j^*}] = [(2^\lambda + x^A_{j^*} - y^A_{j^*} \bmod 2^{\lambda+1}) \leq (x^B_{j^*} - y^B_{j^*} \bmod 2^{\lambda+1})]$$

The terms of the right-hand term can be locally computed by Alice and Bob from their shares, and correspond to the values $\hat{x}, \hat{y}$ in the description of Π_{compress}. If $x = y$, the "shares of (x_{j^*}, y_{j^*})" computed by the players are instead shares of 0; as $0 \leq 2^\lambda$ and $[0 \leq 0] = [x \leq y] = 1$, the correctness is maintained. We therefore have $[x \leq y] = [\hat{x} \leq \hat{y}]$, where $(\hat{x}, \hat{y})$ are $\lambda + 1$ bit long. Note that as $(\hat{x}, \hat{y})$ have been locally computed as modular sums of uniformly random shares of (x_{j^*}, y_{j^*}), they are uniformly random from the viewpoint of each player.

With the correctness argument in mind, the simulation is straightforward (as before, we focus on the case of a corrupted Bob): the simulator $\mathscr{S}im$ runs local copies of $\mathscr{F}_{\mathsf{ET}}$ and $\mathscr{F}_{\mathsf{SC\text{-}prep}}$ and perform honestly the initialization phase, storing the outputs. In step 1, $\mathscr{S}im$ extracts Bob's input y from his calls to ET, and sends $(\mathsf{compress}, \ell, \lambda, y)$ to $\mathscr{F}_{\mathsf{compress}}$ on behalf of Bob in the ideal world. $\mathscr{S}im$ gets an output $\hat{y}'$. In steps 1 - 3, $\mathscr{S}im$ makes random calls to the ET command and sends random values of the appropriate size on behalf of Alice.

$\mathscr{S}im$ broadcasts μ random bits γ'_j in step 5 and waits to receive the tuple sent by Bob in step 5 together with the values $d'_j = \delta_j \oplus d[j]$ sent in step 4 (steps 4 and 5 are executed simultaneously). Using the value stored in the initialization phase, $\mathscr{S}im$ extracts $(\delta_1, \cdots, \delta_\mu, \boldsymbol{r_B})$. $\mathscr{S}im$ picks $(R_j)_{j \leq \mu} \leftarrow_R \mathbb{Z}_{2^{\lambda+1}}$ subject to $\sum_j R_j = \hat{y}' \bmod 2^{\lambda+1}$. For $j = 1$ to μ, if $\delta_j = 0$, $\mathscr{S}im$ sends $(\boldsymbol{t_{d[j]}}[j] - \boldsymbol{r_B}[j] + R_j \bmod 2^{\lambda+1}, \boldsymbol{t_{1 \oplus d[j]}})$ (note that $\boldsymbol{t_{d[j]}}$ is known to Bob, while $\boldsymbol{t_{1 \oplus d[j]}}$ is a perfectly random element of $\mathbb{Z}_{2^{\lambda+1}}$ from his point a view). If $\delta_j = 1$, $\mathscr{S}im$ sends this pair in permuted order.

It is easy to see that the semi-honest Bob will then obtain $\hat{y}'$ as output. In the simulated protocol as well as in the real protocol, all the values exchanged are perfectly indistinguishable from uniformly random values. Moreover, in both the ideal world and the real protocol, the outputs obtained are random values (o_A, o_B) satisfying $[o_A \leq o_B] = [x \leq y]$. Therefore, the real execution is perfectly indistinguishable from a simulated execution in the ideal world.

B.6 Security Analysis of $\Pi_{\mathsf{SC\text{-}prep}}$ (Sketch)

We will not detail a security argument for $\Pi_{\mathsf{ET\text{-}prep}}$; it suffices to observe that the security argument is essentially identical to the security argument for $\Pi_{\mathsf{ET\text{-}prep}}$, as $\Pi_{\mathsf{SC\text{-}prep}}$ and $\Pi_{\mathsf{ET\text{-}prep}}$ are very similar, and Π_{SC} has essentially the same structure than Π_{ET}. For completeness we recall the protocol $\Pi_{\mathsf{SC\text{-}prep}}$:

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

Size-Reduction(ℓ)**:** The players perform the following operations:

1. The players call $\mathscr{F}_{\mathsf{ROT}}^{\mu,\mu+1}$, with Alice acting as sender and Bob as receiver. Let $(\boldsymbol{s_0}, \boldsymbol{s_1})$ denote Alice's output, and let $e \in \mathbb{Z}_2^{\mu+1}$ and $\boldsymbol{s} \leftarrow \left(\boldsymbol{s_{e[i]}}[i]\right)_{i \leq \mu}$ denote Bob's output.
2. The players call $\mathscr{F}_{\mathsf{ROT}}^{\mu,2^{\lambda+1}}$, with Alice acting as sender and Bob as receiver. Let $(\boldsymbol{t_0}, \boldsymbol{t_1})$ denote Alice's output, and let $d \in \mathbb{Z}_2^{2^{\lambda+1}}$ and $\boldsymbol{t} \leftarrow \left(\boldsymbol{t_{d[i]}}[i]\right)_{i \leq \mu}$ denote Bob's output.
3. The players call $\mathscr{F}_{\mathsf{ROT}}^{\mu,2^{\lambda+1}}$, with Alice acting as receiver and Bob as sender. Let $(\boldsymbol{u_0}, \boldsymbol{u_1})$ denote Bob's output, and let $e \in \mathbb{Z}_2^{2^{\lambda+1}}$ and $\boldsymbol{u} \leftarrow \left(\boldsymbol{u_{e[i]}}[i]\right)_{i \leq \mu}$ denote Alice's output.

Product-Sharing(n)**:** Alice picks $(x, a) \leftarrow_R (\mathbb{Z}_2^{2^n-2})^2$, and Bob picks $y \leftarrow_R \mathbb{Z}_2^{2^n-1}$. The players call $\mathscr{F}_{\mathsf{OT}}^{2^n-2,1}$ on input $(a[i], a[i] \oplus x[i])_{i \leq n}$ for Alice and y for Bob. Let b denote Bob's output. Alice outputs (x, a) and Bob outputs (y, b).

The following theorem can be proven as Theorem 2:

Theorem 5 (Repeated from Section 4)**.** *The protocol Π_{SC} securely implements $\mathscr{F}_{\mathsf{SC}}$ when calls to $\mathscr{F}_{\mathsf{SC\text{-}prep}}$ in Π_{SC} are replaced by executions of $\Pi_{\mathsf{SC\text{-}prep}}$ in the $(\mathscr{F}_{\mathsf{ROT}}, \mathscr{F}_{\mathsf{OT}})$-hybrid model, with respect to passive corruption.*