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* [Yao86], 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 in general less efficient than other fundamental primitives, and as such, are often a major bottleneck in multiparty computation protocols.

In this work, we design new two-party protocols for the greater-than functionality, secure against honest-but-curious adversaries (who follow the specifications of the protocol), improving over the state of the art. They can be readily used in a large variety of applications in which secure comparisons constitute the main efficiency bottleneck. Our protocols are defined in the preprocessing model, and are extremely efficient during the online phase. They are based solely on oblivious transfers, and can therefore 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, 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 [Yao86] and Golwasser, Micali, and Widgerson [GMW87b, GMW87a]. 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

[⋆] CNRS – UMR 8548 and INRIA – EPI Cascade (historically) the very first MPC problem ever considered (introduced in [Yao86] 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 focus on the most conservative version of the two-party secure comparison problem, in which neither the inputs nor the output are revealed to the players (the inputs and the output are either encrypted or secretly shared between the players).[1] This is the setting of most applications, in which secure comparison will be used as an internal building block of a larger protocol, hence disclosing the result of the comparison to the players would reveal values at intermediate steps of the larger protocol. 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 extremely suited for large scale secure computation protocols using secure comparison as a basic routine (this statement will be made precise later on). As this is the model used in most applications, we focus on the honest-but-curious setting, 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 (who might deviate arbitrarily from the specifications of the protocol), 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. Hence, 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

[1] Note, however, that our protocols can trivially be adapted to inputs and/or outputs in clear and do still improve over the state-of-the-art in these settings.

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 in the two-party setting following the garbled circuit approach of Yao [Yao86]. The protocol of [KS08], which was later improved in [KSS09], is, to our knowledge, the most communication-efficient secure comparison protocol. It requires each party to know one of the two private inputs, but there are very simple folklore methods to reduce the problem of comparing shared or encrypted inputs to the problem of comparing inputs held by the parties (we will recall such methods in our construction). Similarly, 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 [KSS09] 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 [KS08], the xor gates are garbled for free. The resulting garbled circuit therefore consists of ℓ garbled and gates. Using the recent result of [ZRE15] 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 [VBdHE15]),[2] in a setting where several instances of the comparison protocol are likely to be invoked, it can rely on oblivious transfer extensions [IKNP03] 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 [DGK07], 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 [DGK08]). 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

[2] 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 [VBdHE15], hence to compare even better with current alternatives.

efficient. The more recent construction of [GHJR15] 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 [CDN01] (ABB), which is an ideal reactive functionality for performing securely basic operations (such as additions and multiplications) over secret values loaded in the ABB. The ABB itself can be implemented from various primitives, such as oblivious transfer [Rab81, EGL82] or additively homomorphic encryption (most articles advocate the Paillier scheme [Pai99]). Protocols in this category vary greatly in structure. Most protocols [DFK+06, GSV07, RT07, NO07, Cd10a, Yu11] 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 [Tof11], and Toft and Lipmaa [LT13], 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 [YY12] 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.[3]

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 [YY12, GHJR15], but this comes at the cost of very large constants or somewhat homomorphic encryption, hence these constructions do not beat the most practical constructions [KSS09, DGK07] for realistic values of ℓ. Asymptotically, the most efficient construction is the $O(\ell+\kappa^2)$ protocol of [YY12]. Regarding computation, we expect the protocol of [KSS09] 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.

[3] A rough estimation indicates that this approach becomes competitive with e.g. [KSS09] only for inputs whose bit-size ℓ is of the order of several thousands.

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 based solely on the existence of oblivious transfer [Rab81, EGL82]. 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 also implies that our protocols perform very well regarding computation in an *amortized* setting, in which many secure comparisons are involved. Indeed, using oblivious transfer extension [IKNP03] allows 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 [KK13].
– Our protocols are 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. The online phases of our protocols are information-theoretically secure. Although our protocols compare already very favorably to the state-of-the-art during the preprocessing phase, they perform particularly well in the online phase.
– When instantiated with specific constructions for oblivious transfer extension (in particular the protocol of [KK13]), 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 [KSS09] 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 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, 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 $O\left(\frac{\ell \log \ell}{\log \log \kappa} + \frac{\kappa^2}{\log \kappa}\right)$. This approaches the complexity of the best protocols regarding asymptotic

communication [YY12, GHJR15], 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.

1.3 Our Method

The approach on which we rely is close to the intuition underlying the secure comparisons in [Tof11, LT13]: to compare two strings, one can divide them in equal length blocks, and compare the first block on which they differ. It remains to obliviously select this block; this is done using both oblivious transfers and equality tests. 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 $b_0 + b_1 = b \bmod 2$). The players must not learn anything from the transcript of the protocol.

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 [GHJR15]). 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). 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 [CPP16], protocols for secure linear algebra [CKP07], secure pattern matching [HT14], and secure evaluation of linear programs [Tof09]. Therefore, our equality test protocol is a contribution of independent interest. 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%). However, the latter construction would be usable in our secure comparison protocol only for very large strings and so is essentially a contribution of independent interest. Therefore, we postpone its description to the appendix.

1.4 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 very 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.

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 [NIIO14, BKU14], oblivious RAM [Gol87, WSC08, WS08, PR10], or private set intersection [HEK12], but it remains to date quite slow (implementations [HICT14] report that sorting over a million 32-bit words takes between 5 and 20 minutes, depending on the method used), in spite of receiving a lot of attention [Goo10, JKU11, Zha11, HKI+12, HICT14, Goo14]. 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 [BAC+11, Mu14] for surveys on the topic). One of the most important primitives for such protocols (used in e.g., secure face recognition [SSW10, EFG+09, XT14]) 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 [EFG+09] 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 hoping 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 [BPTG14,RPV$^+$14] or evaluating decision trees [WFNL15]. They are necessary in a large variety of applications of secure machine learning; examples include generating private recommendations [EVTL12], spam classification [WFNL15], multimedia analysis [CC15], clinical decisions [RPV$^+$14], evaluation of disease risk [ARL$^+$13], or image feature extraction [LLY$^+$16]. For example, the protocols of [WFNL15] 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. [ACM$^+$13, AV15, Lau15, BS15] and have applications in several cryptographic protocols, such as private fingerprint matching (using a secure algorithm to find the maximum matching size in a bipartite graph, see [BS15]), privacy-preserving GPS guidance, or privacy-preserving determination of topological features in social networks (which is a special case of the maximum flow problem, see [ACM$^+$13]). Secure protocols for combinatorial problems typically involve a very large number of secure comparisons: according to [ACM$^+$13], 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 [CS10, Cd10a, Lie12, ABZS13]. Secure algorithms for fixed-point arithmetic, as well as for arithmetic on floating numbers, heavily rely on comparisons. It was pointed out in [Cd10a] that comparisons and integer truncation are core components of fixed-point arithmetic, and the most important performance bottlenecks in complex applications.

Other Applications. The above list is far from exhaustive; other applications for secure comparisons include various types of secure auctions [DGK07, HGW$^+$15], range queries over encrypted databases (which involve a secure comparison per element of the database) [SJB14], or secure algorithms for optimization problems (for example, securely evaluating a simplex algorithm [Tof09, Cd10b], which occurs in supply chain planning between concurrent suppliers, might involve hundreds of secure comparisons on $\approx$ 100-bits values, according to [Cd10b]).

1.5 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.

2 Preliminaries

In this section, we recall the necessary background on oblivious transfer, the cryptographic primitive on which our protocols are based. We present some existing results, on which we will rely when estimating the efficiency of our protocols. These constructions allow to execute oblivious transfers very efficiently (in an amortized setting), and some of them are adapted to inputs satisfying particular conditions (namely, being *short* or *correlated*) that will be satisfied in our protocols.

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. We let κ denote a security parameter (we will use $\kappa = 128$ in our estimation). The symbol $\oplus$ denotes the xor operation (when applied on bit-strings, it denotes the bitwise xor). Given integers (x, y), we write $[x = y]$ (resp. $[x \leq y]$) to denote the bit value which is 1 if this predicate is true, and 0 else. Given a rational number f, $\lceil f \rceil$ and $\lfloor f \rfloor$ denote respectively the smallest integer above f and the largest integer below f. For a group $\mathbb{G}$, we denote by $|\mathbb{G}|$ the length of the group, which is the bit-size of (a representation of) the elements of $\mathbb{G}$. Eventually, we denote by $H_d(x, y)$ the Hamming distance between bit-strings x and y, which is the number of bits on which they differ.

2.1 Oblivious Transfers

Oblivious transfers (OT) were introduced in [Rab81, EGL82]. A $\binom{2}{1}$-OT is a two-party functionality which, on input two messages (m_0, m_1) from a *sender* and a bit b from a *receiver*, outputs m_b to the receiver. *Sender privacy* states that the entire view of the receiver can be simulated from m_b only. *Receiver privacy* states the views of the sender when interacting with a receiver with input 0 or with input 1 are indistinguishable (see [NP01] for more details on these security notions). The primitive can be extended naturally to k-out-of-n oblivious transfers; $\binom{n}{k}$-OT^t_ℓ 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 [Kil88, IPS08] and can be made very efficient.

Naor-Pinkas Oblivious Transfer Protocol For the sake of completeness, we recall the $\binom{2}{1}$-OT_ℓ^t protocol proposed in [NP01]. It is secure against malicious adversaries in the random oracle model.[4]

Input: The sender holds t pairs of ℓ-bit strings $(x_i^0, x_i^1)_{i \leq t}$, and the receiver holds t selection bits $(b_i)_{i \leq t}$.

Output: The receiver gets $(x_i^{b_i})_{i \leq t}$; the sender gets nothing.

Setting: The parties agree on a group $\mathbb{G}$ of order p with generator g, such that DDH is conjectured to hold over $\mathbb{G}$. Let C be a random element of $\mathbb{G}$ chosen by the sender. Let $H : \{0,1\}^* \mapsto \{0,1\}^\kappa$ be a hash function.

1. For $i = 1$ to t, the receiver picks $r_i \leftarrow_R \mathbb{Z}_p$ and sets $\mathsf{pk}_{b_i}^i \leftarrow g^{r_i}$ and $\mathsf{pk}_{1-b_i}^i \leftarrow C/\mathsf{pk}_{b_i}^i$. The receiver sends $(\mathsf{pk}_0^i)_{i \leq t}$.
2. The sender computes $\mathsf{pk}_1^i \leftarrow C/\mathsf{pk}_0^i$ for $i \leq t$ and picks $r \leftarrow_R \mathbb{Z}_p$. He sends $u \leftarrow g^r$ together with $(E_0^i, E_1^i) \leftarrow (H((\mathsf{pk}_0^i)^r, 0) \oplus x_0^i, H((\mathsf{pk}_1^i)^r, 1) \oplus x_1^i)$.
3. The sender computes his output as $(E_{b_i}^i \oplus H(u^{r_i}, b_i))_{i \leq t}$.

The security of the protocol in the random oracle model relies on the decisional Diffie-Hellman (DDH) assumption, which states that the distribution of tuples of the form (g^x, g^y, g^{xy}) for random exponents x, y is computationally indistinguishable from the distribution of random triples of group elements. The protocol transmits $(t+1)|\mathbb{G}| + 2t\ell$ bits. Using recommended parameters for the length of $\mathbb{G}$ gives $|\mathbb{G}| \approx 2\kappa$ when $\mathbb{G}$ is instantiated as a well-chosen elliptic curve.

Preprocessing Oblivious Transfers. There is a very simple and black-box way to preprocess oblivious transfers before knowing the inputs, which was introduced in [Bea95]. Let us consider two players, a sender Alice and a receiver Bob. In the preprocessing stage, the players do not know their actual inputs to the protocol, but they know the bit-size of these inputs. Alice picks two uniformly random inputs (x_0, x_1) of the correct bit-size, and acts as sender in an oblivious transfer on those random inputs. Bob picks a uniformly random selection bit b and gets x_b from the protocol. Later, in the online phase, Alice gets an input (m_0, m_1), and Bob gets a selection bit β. Bob simply tells Alice whether $b = \beta$, by sending a single bit $b \oplus \beta$. If indeed $b = \beta$, then Alice sends $(y_0, y_1) \leftarrow (x_0 \oplus m_0, x_1 \oplus m_1)$ to Bob, who computes $x_b \oplus y_b = m_b = m_\beta$. Else, $\beta = 1 - b$ and Alice sends $(y_0, y_1) \leftarrow (x_0 \oplus m_1, x_1 \oplus m_0)$ to Bob, who computes $x_b \oplus y_b = m_{1-b} = m_\beta$. When the bit-size ℓ of the input is large (say, of bit-size $\ell = \kappa$), these $2\ell + 1$ bits of communication add a notable overhead to the oblivious transfer; however, for very small inputs, this preprocessing phase allows to precompute very efficiently the oblivious transfers. We will rely on this observation when designing our secure comparison protocols.

[4] The random oracle model [BR93] assumes that the hash function H acts as a truly random function to which the players are only given black-box access.

2.2 Oblivious Transfer Extension

An oblivious transfer extension protocol is a protocol which allows two players to reduce polynomially many OTs to a constant number of base OTs. They play a crucial role in secure multiparty computation; indeed, while it is provably impossible to construct MPC protocols without public key operations (without assuming an honest majority, which is in particular the case for two-party computation), the existence of oblivious transfer extension shows that all the public key operations of an MPC protocol can be confined to a single, constant-size preprocessing phase. OT extensions were introduced in [Bea96], but the construction was essentially of theoretical interest. [IKNP03] proposed the first truly practical OT extension protocol. Assuming the random oracle model,[5] polynomially many OTs can be reduced to κ base OTs of κ-bit strings, using only cheap operations (such as evaluating hash functions). The idea of the protocol is the following: a $\binom{2}{1}\text{-}\mathsf{OT}_t^\kappa$ can be directly obtained from a $\binom{2}{1}\text{-}\mathsf{OT}_\kappa^\kappa$; indeed, the sender associates two κ-bit keys to each pair of messages and the receiver gets to know one of the two keys, corresponding to its selection bit, of each pair. 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 $\binom{2}{1}\text{-}\mathsf{OT}_\ell^t$ itself can be implemented with a *single call* to a $\binom{2}{1}\text{-}\mathsf{OT}_t^\kappa$ functionality, in which the receiver plays the role of the receiver (and reciprocally). The total communication of the reduction from $\binom{2}{1}\text{-}\mathsf{OT}_\ell^t$ to $\binom{2}{1}\text{-}\mathsf{OT}_\kappa^\kappa$ is $2t\ell+2t\kappa$ bits. Regarding the computational complexity, once the base OTs have been performed, each OT essentially consists in three evaluations of a hash function.

An optimization to the [IKNP03] paper was proposed in [ALSZ13] (and discovered independently in [KK13]). This optimization reduces the communication of the OT extension protocol from $2t\ell + 2t\kappa$ bits to $2t\ell + t\kappa$ bits. This optimization does also allow 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. Note also that oblivious transfer extensions can be preprocessed too, as for classical oblivious transfers; this leads again to a particularly efficient online phase when ℓ is small, which transmits only $2\ell + 1$ bits.

Oblivious Transfer of Short Strings. [KK13] constructed OT extension protocols tailored to the case of short inputs. More specifically, the authors describe a reduction of $\binom{2}{1}\text{-}\mathsf{OT}_\ell^t$ to $\binom{2}{1}\text{-}\mathsf{OT}_\kappa^\kappa$ with $t(\kappa'/\log n + n \cdot \ell)$ bits of communication (for some security parameter κ' which should be taken equal to 2κ), 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 $\binom{2}{1}\text{-}\mathsf{OT}$ to one invocation of $\binom{n}{1}\text{-}\mathsf{OT}$; the result is then obtained by combining this reduction with a new $\binom{n}{1}\text{-}\mathsf{OT}$ extension protocol introduced in [KK13]. In our concrete efficiency estimations, we

[5] The random oracle model can be avoided by assuming that the hash function is a correlation-robust function, see [KK13], AppendixA.2

will heavily rely on this result as our equality test protocol involves OTs on very short strings.

Correlated Oblivious Transfers. The authors of [ALSZ13] 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). We note that the optimizations of [KK13] and [ALSZ13] 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 $t(\kappa'/\log n + (n-1)\cdot\ell)$ bits.

3 Fast Oblivious Transfer-Based Equality Test

In this section, we design a protocol allowing two players, holding respective ℓ-bit inputs x and y, to securely compute shares (modulo 2) of the bit $b = [x = y]$. Our equality test (ET) protocol enjoys the following features:

- It relies solely on oblivious transfer; in particular, this means that it can be based on a large variety of cryptographic assumptions, and in an amortized setting, we can use oblivious transfer extension and perform polynomially many ETs using only cheap, symmetric operations (such as evaluating hash functions).
- It can be very efficiently preprocessed. During the online phase, only slightly more than 2ℓ bits are exchanged, and only modular additions are computed.
- Compared to the state-of-the-art, it strongly improves the communication (by up to 80%), and has very low amortized computation.

This ET protocol is the core building block of our secure comparison protocols. Our protocol is in the spirit of [GHJR15]: using the fact that two strings of length ℓ are equal if and only if their Hamming distance (of length $\log\ell$) is zero, we perform several reduction steps, each step reducing an equality test on k-bit inputs to an equality test on $\log k$-bit inputs. After $\log^*\ell$ such steps, we invoke a protocol dedicated to equality tests on very small strings.

Equality Tests on Large Strings. When the inputs to the equality test are of size $\ell > \kappa$, the players can use the following folklore method to reduce it to an equality test on κ-bit strings: the players agree on the description of a hash function $H : \{0,1\}^\ell \mapsto \{0,1\}^\kappa$ and perform the protocol on the hash of their inputs. This reduces the size of the inputs to κ-bits while preserving equality. Note, however, that this adds a computational assumption to the protocol, namely, that the hash function is collision-resistant. Under this (mild) requirement, for large strings, the equality test protocol can be made independent of the size of the inputs.

3.1 Sub-Protocols

We start by describing the two components of our protocol, the small-string equality test protocol and the equality-preserving size-reduction protocol.

Small-Strings Equality Test. We design a protocol for testing equality between two k-bit strings, for some given k. The protocol involves $2^k - 2$ oblivious transfers on bits, hence is suitable only for very small values of k. In the following, we let $(I_j)_{0 \leq j \leq 2^k-3}$ denote the list of non-empty strict subsets of $\{0, \cdots, k-1\}$ (in any arbitrary fixed order).

Input: $a = (a_i)_{i \leq k-1} \in \{0,1\}^k$ for Alice and $b = (b_i)_{i \leq k-1} \in \{0,1\}^k$ for Bob.
Output: A random bit x for Alice and $\prod_{i=0}^{k-1}(a_i \oplus b_i) \oplus x$ for Bob.
Preprocessing phase: (the inputs are not required in this phase; only their length must be known)

1. Alice picks random bits $(\alpha_j, c_j)_{0 \leq j \leq 2^k-3} \leftarrow_R (\{0,1\}^2)^{2^k-2}$. Bob picks random bits $(\beta)_{0 \leq j \leq 2^k-3} \leftarrow_R \{0,1\}^{2^k-2}$.
2. Alice acts as sender in a $\binom{2}{1}$-$\mathsf{OT}_1^{2^k-2}$ on inputs $(c_j, \alpha_j \oplus c_j)$ for $j = 0$ to $2^k - 3$. Bob's selection bits are the β_j, for $j = 0$ to $2^k - 3$; let $(c'_j)_{0 \leq j \leq 2^k-3}$ be his outputs.

Online phase: Once the inputs are revealed to the players,

1. For $j = 0$ to $2^k - 3$, Alice sends $\gamma_j \leftarrow \alpha_j \oplus \prod_{i \in I_j} \bar{a}_i$; simultaneously, Bob sends $\delta_j \leftarrow \beta_j \oplus \prod_{i \in \{0,k-1\} \setminus I_j} b_i$.
2. Alice sets her output to $\bigoplus_{0 \leq j \leq 2^k-3}(c_j \oplus \alpha_j\delta_j \oplus \gamma_j\delta_j) \oplus \bigoplus_{0 \leq i \leq k-1} \bar{a}_i$. Bob sets his output to $\bigoplus_{0 \leq j \leq 2^k-3}(c'_j \oplus \beta_j\gamma_j) \oplus \bigoplus_{0 \leq i \leq k-1} b_i$.

The idea of the protocol is fairly simple: observe that $[a = b] = \prod_{i=0}^{k-1}(\bar{a}_i \oplus b_i)$. This product can be developed and rewritten

$$[a = b] = \bigoplus_{0 \leq i \leq k-1} (\bar{a}_i \oplus b_i) \oplus \bigoplus_{j=0}^{2^k-3} \prod_{i \in I_j} \bar{a}_i \cdot \prod_{i \in \{0,k-1\} \setminus I_j} b_i$$

Hence, the players simply compute shares of all those products, using a single oblivious transfer per value of j, and xor all their shares together. Each oblivious transfer is performed on random inputs during the precomputation phase; the players exchange two correction bits per bit-OT to get the correct result during the online phase, using a standard precomputation technique.

***k*-Bit Equality-Preserving Size-Reduction (k−EPSR).** This protocol allows two players holding inputs of length k to *reduce* the length of their input while preserving the equality between the inputs: the protocol outputs strings of size $\lceil \log(k+1) \rceil$ which are equal if and only if the input strings are equal. The communication consists of k parallel invocations of a one-out-of-two oblivious transfer on strings of size $\lceil \log(k+1) \rceil$.

Input: $r = (r_i)_{i \leq k} \in \{0,1\}^k$ for Alice and $s = (s_i)_{i \leq k} \in \{0,1\}^k$ for Bob.
Output: Two $\lceil \log(k+1) \rceil$-bit strings, $r' = (r'_i)_{i \leq \lceil \log(k+1) \rceil} \in \{0,1\}^{\lceil \log(k+1) \rceil}$ for Alice and $s' = (s'_i)_{i \leq \lceil \log(k+1) \rceil} \in \{0,1\}^{\lceil \log(k+1) \rceil}$ for Bob so that $r' = s'$ if and only if $r = s$.
Preprocessing phase: (the inputs are not required in this phase; only their length must be known)

1. Alice picks a random string $x = x_1 x_2 \cdots x_k$ and $(\rho_i)_{i \leq k} \leftarrow_R \mathbb{Z}_{k+1}^k$. Bob picks a random string $y = y_1 y_2 \cdots y_k$. Alice acts as sender in a $\binom{2}{1}$-$\mathsf{OT}^k_{\log(k+1)}$ on k inputs $(\rho_i, 1 - 2x_i + \rho_i \bmod k+1)_{i \leq k}$. Bob gets $(\sigma_i)_i$ on selection bits y_i.
2. Alice sets $r' \leftarrow \sum_i \rho_i \bmod k+1$. Bob sets $s' \leftarrow \sum_i \sigma_i \bmod k+1$.

Online phase: Once the inputs are revealed to the players,

1. Alice sends $r \oplus x$ to Bob. Simultaneously, Bob sends $s \oplus y$ to Alice. Both players compute $z \leftarrow r \oplus x \oplus s \oplus y$ (the strings are xored bitwise).
2. Alice sets $r' \leftarrow \sum_i (-1)^{z_i}(\rho_i - x_i) \bmod k+1$; Bob sets $s' \leftarrow \sum_i w_i + (-1)^{z_i}\sigma_i \bmod k+1$.

It follows easily from a careful inspection of the protocol that $r' - s' = H_d(r,s) \bmod k+1$. As r and s are both k-bit long, it holds that $H_d(r,s) \leq k$, hence $r' = s'$ holds over the integers if and only if $H_d(r,s) = 0$ holds over the integers, which happens only when $r = s$.

3.2 Equality Test Protocol

Using the sub-protocols described above, we are now ready to describe an equality test protocol on ℓ-bit strings. The protocol performs $\log^* \ell$ rounds of $k-\mathsf{EPSR}$, up to a point where the players hold strings of length upper bounded by n, for some threshold $n \geq 2$. Then, the players perform a small string equality test protocol, which involves $2^n - 2$ oblivious transfers on bits. One should note that setting $n = 2$ always minimizes the communication; however, it might be desirable in some applications to stop at $n = 3$ or $n = 4$, which saves one round at the cost of a slightly increased communication (by a few hundreds of bits).

Input: $r = (r_i)_{i \leq \ell} \in \{0,1\}^\ell$ for Alice and $s = (s_i)_{i \leq \ell} \in \{0,1\}^\ell$ for Bob. The players agree on some threshold n.
Output: Two bits, b_A for Alice and b_B for Bob, with $b_A \oplus b_B = 1$ iff $r = s$.

1. $(r_0, s_0) \leftarrow (r, s)$, $k \leftarrow \ell$, $i \leftarrow 0$
2. **While** $k > n$,
   – Alice and Bob perform an equality-preserving k-bit size-reduction protocol on (r_i, s_i); let (r_{i+1}, s_{i+1}) be the output of the protocol. Note that to optimize the round complexity, Alice and Bob should exchange their roles as sender and receiver in two consecutive $k-\mathsf{EPSR}$.
   – $i \leftarrow i+1$, $k \leftarrow \lceil \log(k+1) \rceil$
3. Alice and Bob perform a small-string equality test protocol on the strings (r_i, s_i) (of length smaller than n); the output of this protocol is their final output.

Note that the preprocessing phase of this protocol (which is the preprocessing phase of the sub-protocols it invokes) corresponds exactly to executing equality tests on random strings and relies solely on oblivious transfer, while the online phase is information-theoretically secure; it involves only sending a few strings and computing some modular additions in small-order groups.

3.3 Security Analysis

Let us show that our equality test protocol is secure against honest-but-curious adversaries, for both Alice and Bob, if the oblivious transfer protocol is secure against honest-but-curious adversaries. Note that if the inputs are of size $\ell > \kappa$ and the players wish to reducing the communication by hashing their inputs, one must also assume a collision-resistant hash function.

First, note that in all the oblivious transfers involved in both the small-string equality test and in the $k-\mathsf{EPSR}$, the sender uses two independently random inputs, i.e., two inputs (m_0, m_1) such that for $b \in \{0,1\}$, m_b is perfectly indistinguishable from a random input (of the appropriate length) from the viewpoint of an adversary, given that this adversary does not know m_{1-b}. Therefore, the *sender privacy* of the OTs, which guarantees that at most one inputs out of two can be recovered by a (semi-honest) receiver, ensures that the view of the receiver in each OT consists solely of uniformly random bit-strings.

In the small-string equality test, Alice acts as sender in $2^k - 2$ OTs on independently random inputs $(c_j, \alpha_j \oplus c_j)$. As c_j is random, by the sender privacy of the OTs, the value α_j remains perfectly hidden from Bob's viewpoint. Later, in the online phase, Alice reveals $\gamma_j = \alpha_j \oplus \prod_{i \in I_j} \bar{a}_i$. As α_j is uniformly random and remained perfectly hidden during the preprocessing phase, γ_j perfectly masks the bit $\prod_{i \in I_j} \bar{a}_i$, which ensures Alice's privacy during the small-string equality test. Bob's privacy is immediately implied by the receiver privacy of the oblivious transfers: by the receiver privacy, Alice learns nothing at all during the preprocessing phase, and Bob reveals $\delta_j \leftarrow \beta_j \oplus \prod_{i \in \{0,k-1\} \setminus I_j} b_i$ in the online phase, which perfectly masks $\prod_{i \in \{0,k-1\} \setminus I_j} b_i$ as β_j is uniformly random from the viewpoint of Alice.

We now turn our attention to the equality-preserving size-reduction protocols. The argument is similar: during the preprocessing phase, the inputs to the OTs are independently random and therefore, by the sender privacy of the OTs, they perfectly mask the bits of x. Hence, $r \oplus x$ perfectly masks r, as x is uniformly random and remained hidden from the viewpoint of Bob during the preprocessing phase. The receiver privacy implies that Alice learns nothing during the preprocessing phase; during the online phase, $s \oplus y$ perfectly masks s as y is uniformly random. Note that the security is trivially maintained when performing several $k-\mathsf{EPSR}$ sequentially.

Overall, the view of each player in the full equality test is indistinguishable from uniformly random, if the oblivious transfer protocol ensures both sender privacy and receiver privacy. $\square$

3.4 Communication Complexity

Asymptotically, the cost of the first $\ell-\mathsf{EPSR}$ protocol clearly dominates the communication, hence we focus on this sub-protocol to estimate the communication complexity of the equality test protocol. Using any standard constant-round oblivious transfer protocol, which communicates $O(\kappa)$ bits, we get a protocol with bit complexity $O(\kappa\ell)$ in the preprocessing phase, and $O(\ell)$ in the online phase. However, in an amortized setting where many equality tests are likely to be invoked, we can do better. Using the short-string oblivious transfer extension protocol of [KK13], we can reduce m $\ell-\mathsf{EPSR}$ to κ base OTs on κ-bit strings, using $O(m\ell(\kappa/\log x + x \cdot \lceil \log \ell \rceil))$ bits of communication, where x is a parameter than can be arbitrarily set so as to minimize this cost. Taking $x = O(\mathfrak{W}(e^{\kappa/\lceil \log \ell \rceil}))$ (where $\mathfrak{W}$ denotes the Lambert function, which is the reciprocal of the function $u \mapsto ue^u$), the communication becomes $O(m\ell\kappa/\log \mathfrak{W}(e^{\kappa/\lceil \log \ell \rceil}))$, which is approximately $O(m\ell\kappa/\log \kappa)$ up to some $\log \log$ term (recall that we can always assume $\ell \leq \kappa$, by letting the players hash their input first if its size exceeds κ, hence $\log \ell \leq \log \kappa$). 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 $\ell > \kappa$, the players can hash their inputs so that 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 [KSS09], 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 [KS08]. 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 [ZRE15], 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 [ALSZ13] 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 [NP01] recalled in the previous section, setting the group as an elliptic curve of prime order p (which can be taken of bit-size $\log p = 2\kappa$ according to recommended parameters). The security of the protocol in the random oracle model can be reduced to the DDH assumption. We recall the complexity of the OT of [NP01], assuming the bit-size of group elements is 2κ and the random oracle outputs κ-bit strings:

- 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 [NP01] 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 [NP01] 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 k-bit equality test protocol involves only oblivious transfers on very small strings, of size $\ell = \log k \leq \log \kappa$. 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 at $N = 4$ 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 [KSS09]. Note that we use the oblivious transfer of [NP01] in both our ET and the protocol of [KSS09], 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 [KSS09]. Note also that we do not preprocess oblivious transfers, because this would involve an overhead of $2\kappa\ell$ bits in the overall communication of the protocol of [KSS09] (and is not required at all in our ET). As one can see from Table 1, our protocol transmits more bits than [KSS09] 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 [KSS09].

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

	Our ET		[KSS09]	
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 [KSS09]. Adapting the construction of [KK13] 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 $m(\kappa'/\log x + (x-1)t)$ for some security parameter (this takes into account an optimization described in the appendix A of [KK13] and the optimization for correlated inputs of [ALSZ13]). However, the authors of [KK13] point out that to reach κ bits of security, κ' must be set to 2κ; hence the actual cost is $m(2\kappa/\log x + (x-1)t)$ bits. This implies that the amortized cost of a $k-\mathsf{EPSR}$ 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 [KSS09]; oblivious transfers for the garbled circuit approach of [KSS09] are performed using the OT extension protocol of [ALSZ13] on κ-bit inputs, which transmits 3κ bits per OT. As shown in Table 2, our protocol improves over the construction of [KSS09] by up to 80% overall. During the online phase, our protocol is extremely efficient, two orders of magnitude faster than [KSS09]. Note that there are equality test protocols in the literature whose online phase transmits a small constant number of ciphertexts, independent of ℓ, the most prominent example being the equality test of [LT13]. It transmits 2ℓ ciphertexts in the preprocessing phase, and only 3 ciphertexts in the online phase. However, using the Paillier encryption scheme advocated by the authors, this amounts to $3 \cdot 4096 = 12288$ bits in the online phase (using the recommended parameters for 112 bits of security), which is still far less efficient than our protocol.

4 Secure Comparison from Equality Test

We now come back to the main goal of this paper and build a secure comparison protocol. Our protocol can be declined in two variants, depending of the preferred setting. It builds upon our equality test protocol described in the previous section. Note that relying on equality test protocols to perform secure comparisons is a standard method, used e.g. in [Tof11, LT13, GHJR15].

4.1 Sub-Protocols

Given inputs of size $\ell = \lambda\mu$, λ and μ being two parameters which can be set arbitrarily so as to optimize the protocol, the core protocol is a reduction from a secure comparison of ℓ-bit inputs to a secure comparison of λ-bit inputs. Depending of the latency of the network, the players can either stop there and compute the λ-bit secure comparison using a dedicated constant round protocol (e.g. the garbled circuit approach of [KS08, KSS09]); or, in the same spirit than the equality test of the previous section, they can apply the reduction again,

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

	Our ET		[KSS09]	
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

decomposing λ as $\lambda'\mu'$ and so on, until they end up with two short strings that can be securely compared using an oblivious transfer-based small-string secure comparison protocol. Note that this second alternative is not constant round (although the number of rounds remains reasonable for realistic inputs) and relies solely on the existence of oblivious transfer, while the first alternative is constant round and relies also on the assumption underlying the secure comparison protocol invoked after the reduction step.[6]

Small-String Secure Comparison. We start by presenting, in the spirit of the previous section, a secure comparison protocol for k-bit inputs which relies on $2^k - 1$ parallel calls to a bit oblivious transfer primitive, which makes it suitable for very small values of k only. The protocol follows easily from the following observation: let $a = a_0 a_1 \cdots a_{k-1}$ and $b = b_0 b_1 \cdots b_{k-1}$ be two k-bit inputs. It holds that $[a > b] = [a_0 > b_0] \vee [a_0 = b_0] \wedge [a_1 \cdots a_{k-1} > b_1 \cdots b_{k-1}]$. Using the fact that $x \vee y = x \oplus y \oplus xy$, we can rewrite this equality as

$$\begin{aligned}[a > b] = a_0 \cdot \bar{b}_0 \oplus (\bar{a}_0 \oplus b_0) \cdot [a_1 \cdots a_{k-1} > b_1 \cdots b_{k-1}] \\ \oplus (a_0 \cdot \bar{b}_0) \cdot ((\bar{a}_0 \oplus b_0) \cdot [a_1 \cdots a_{k-1} > b_1 \cdots b_{k-1}])\end{aligned}$$

[6] For the garbled circuit protocol of [KS08, KSS09], this does not add any assumption, but forces us to work in the random oracle model, which we do anyway in our concrete instantiations with OT extensions.

Observe now that $a_0 \cdot \bar{b}_0 \cdot (\bar{a}_0 \oplus b_0) \cdot [a_1 \cdots a_{k-1} > b_1 \cdots b_{k-1}] = 0$, hence the equation simplifies to $[a > b] = a_0 \cdot \bar{b}_0 \oplus (\bar{a}_0 \oplus b_0) \cdot [a_1 \cdots a_{k-1} > b_1 \cdots b_{k-1}]$. We can therefore apply this equation recursively, using the fact that $[a_{k-1} > b_{k-1}] = a_{k-1} \cdot \bar{b}_{k-1}$ and $[a_{k-1} = b_{k-1}] = a_{k-1} \oplus b_{k-1}$. Written *in extenso*, we get an equation of the form $[a > b] = \bigoplus_{i=1}^{2^k-1} A_i \cdot B_i$, where each A_i (resp. B_i) can be locally computed from a only (resp. b only). Shares of $[a > b]$ can therefore be computed by the two players using $2^k - 1$ oblivious transfer: Alice picks random bits $(c_i)_{i \leq 2^k-1}$ and acts as sender in $2^k - 1$ parallel OTs with Bob, with inputs $(c_i, A_i \oplus c_i)$ for $i = 1$ to $2^k - 1$; Bob's selection bits are the B_i. Let $(c'_i)_i$ denote Bob's outputs; observe that $c'_i = c_i \oplus A_i \cdot B_i$. Therefore, Alice sets her output to $\bigoplus_i c_i$ and Bob sets his output to $\bigoplus_i c'_i$.

Efficiency. The protocol involve oblivious transfers on bits, which can be efficiently preprocessed (see subsection 2.1). $2^k - 1$ bit-OT are performed in the preprocessing phase, and $2(2^k - 1) + 1$ bits are exchanged in the online phase. Although the small-string secure comparison protocol is obviously not suited for large strings, it is useful for concrete efficiency estimations to know from which bit-length it becomes less efficient than the garbled circuit approach. Using $\kappa = 128$, this protocol is more efficient that the garbled circuit approach for $k \leq 5$; it becomes (very quickly) less efficient for bigger values of k.

Conversion Method. We describe a non-interactive method to convert a secure comparison protocol in which both players hold shares (over the integers) of the two k-bit inputs to a secure comparison protocol in which each player holds one of the two k-bit inputs. This conversion protocol is a folklore method.

Input: Two integers (x_A, y_A) for Alice and (x_B, y_B) for Bob; let us denote $(x, y) \leftarrow (x_A + x_B, y_A + y_B)$.
output: $(z_A, b_A) \in \{0,1\}^k \times \{0,1\}$ for Alice and $(z_B, b_B) \in \{0,1\}^k \times \{0,1\}$ for Bob, such that $[x \leq y] = b_A \oplus b_B \oplus [z_A \leq z_B]$.
Method: Alice sets $z_A \leftarrow x_A - y_A \bmod 2^k$ and $b_A \leftarrow \lfloor (2^k + x_A - y_A)/2^k \rfloor \bmod 2$. Bob sets $z_B \leftarrow y_B - x_B \bmod 2^k$ and $b_B \leftarrow \lfloor (y_B - x_B)/2^k \rfloor \bmod 2$.

Note that in most applications, the inputs will be given to one player encrypted using an additively homomorphic encryption scheme, while the other player holds the secret key of the scheme. Assuming that the plaintext space of the scheme is larger than $2^{\kappa+k}$, the first player can pick z_A at random over $2^{\kappa+k}$ (κ bits must be added to the bit-length of the inputs to mask them statistically), and compute an encryption of z_B from the ciphertexts, that he sends to the second player. Hence this involves sending a single ciphertext. The requirement of the plaintext space being way larger than the size of the encrypted values is easily met in practice[7], as additively homomorphic encryption schemes have in

[7] An alternative solution is for the plaintext space $\mathbb{Z}_p$ to be twice larger than the inputs, and for the inputs to be shared modulo p between the player; we use this in our actual protocol and describe this method in the *intuition of the protocol* paragraph.

general large plaintext spaces (ranging from 2^{256} to 2^{2048} for standard security parameters), while encrypted values are often 32-bit to 64-bit words.

Let us now explain why this method is correct. It holds that $(2^k + x_A - y_A) - (y_B - x_B) = 2^k + x - y$. Observe that this is necessarily positive, as $x - y \leq 2^k$. Observe also that the target output, $[x \leq y]$, can be computed as $[x \leq y] = \lfloor (2^k + x - y)/2^k \rfloor = \lfloor ((2^k + x_A - y_A) - (y_B - x_B))/2^k \rfloor$. As $2^k + x_A - y_A \geq y_B - x_B$, decomposing $2^k + x_A - y_A = 2^k u_A + z_A$, and $y_B - x_B = 2^k u_B + z_B$, it necessarily holds that $u_A \geq u_B$, hence the integer division of $(2^k + x_A - y_A) - (y_B - x_B) = 2^k(u_A - u_B) + (z_A - z_B)$ by 2^k is equal to $u_A - u_B$ if $z_A \geq z_B$, and $u_A - u_B - 1$ else, which gives us the equation $[x \leq y] = u_A - u_B - [z_A \leq z_B]$. Reducing this equation modulo 2, using the fact that $(u_A \bmod 2) = b_A$ and $(u_B \bmod 2) = b_B$, gives the claimed result.

Reduction Protocol. We now design a protocol to reduce a secure comparison on ℓ-bit values to a secure comparison protocol on smaller values. We assume that each player holds a private input; when entries are given as shares, the conversion method can be applied to get private inputs. Let $(a_i)_{i \leq \mu}$ (resp. $(b_i)_{i \leq \mu}$) be the input of Alice (resp. Bob) written *by blocks*: for all $i \leq \mu$, $|a_i| = |b_i| = \lambda$. For a parameter l, ET_l denotes the execution of the equality test protocol from Section 3 on l-bit inputs.

Preprocessing phase: The inputs are not required in this phase; only their length must be known. The following steps are all performed in parallel.

1. The players perform μ parallel preprocessing steps for ET_λ and $\mathsf{ET}_{\log(\mu+1)}$.
2. The players perform μ parallel oblivious transfers, where Alice acts as sender with random inputs (ρ_i, ρ'_i) from $\mathbb{Z}_{\mu+1}$ for $i = 1$ to μ, and Bob uses random selection bits $(\theta_i)_{i \leq \mu}$. Let $(\sigma_i)_{i \leq \mu}$ denote Bob's outputs.
3. The players perform μ parallel oblivious transfers, where Alice acts as sender with random inputs (χ_i, χ'_i) from $\mathbb{Z}_{2^{\lambda+1}}$ for $i = 1$ to μ, and Bob uses random selection bits $(\tau_i)_{i \leq \mu}$. Let $(\eta_i)_{i \leq \mu}$ denote Bob's outputs.
4. The players perform μ parallel oblivious transfers, where Bob acts as sender with random inputs (ξ_i, ξ'_i) from $\mathbb{Z}_{2^{\lambda+1}}$ for $i = 1$ to μ, and Bob uses random selection bits $(\zeta_i)_{i \leq \mu}$. Let $(\nu_i)_{i \leq \mu}$ denote Alice's outputs.
5. The players perform the preprocessing step of a secure comparison on $(\lambda + 1)$-bit private inputs, using the method of their choice (for example, the garbled circuit of [KSS09], or recursively applying the reduction protocol).

Online phase: Once the inputs are revealed to the players,

1. The players perform μ online parallel ET_λ on respective inputs (a_i, b_i) for all $i \leq \mu$. Let $(x_i^A, x_i^B)_{i \leq \mu}$ be their respective outputs.
2. Bob sends $(\iota_i)_{i \leq \mu} \leftarrow (x_i^B \oplus \theta_i)_{i \leq \mu}$. Alice picks $(\alpha_i)_{i \leq \mu} \leftarrow_R \mathbb{Z}_{\mu+1}^{\mu}$ and sends $(u_i^0, u_i^1)_{i \leq \mu} \leftarrow (\alpha_i + x_i^A - ((1 - \iota_i)\rho_i + \iota_i \rho'_i) \bmod \mu + 1, \alpha_i + 1 - x_i^A - ((1 - \iota_i)\rho'_i + \iota_i \rho_i) \bmod \mu + 1)_{i \leq \mu}$. Let $(\beta_i)_{i \leq \mu} \leftarrow (u^{x_i^B} + \sigma_i \bmod \mu + 1)_{i \leq \mu}$ denote Bob's output.
3. Alice sets $(\alpha'_i)_{i \leq \mu} \leftarrow \sum_{j=1}^{i} (1 - \alpha_i) \bmod \mu + 1$; Bob sets $(\beta'_i)_{i \leq \mu} \leftarrow \sum_{j=1}^{i} (1 - \beta_i) \bmod \mu + 1$. The players perform μ parallel online $\mathsf{ET}_{\log(\mu+1)}$

on respective inputs $(\alpha'_i, \beta'_i)_{i\leq\mu}$. Let $(y_i^A, y_i^B)_{i\leq\mu}$ be their respective outputs. Each player P sets $z_1^P \leftarrow y_1^P$ and $(z_i^P)_{2\leq i\leq\mu} \leftarrow (y_i^P \oplus y_{i-1}^P)_{2\leq i\leq\mu}$.

4. Bob sends $(\iota'_i)_{i\leq\mu} \leftarrow (z_i^B \oplus \tau_i)_{i\leq\mu}$. Alice picks $r_i^A \leftarrow_R \mathbb{Z}_{2^{\lambda+1}}$ for $i = 1$ to μ, and sends $(v_i^0, v_i^1)_{i\leq\mu} \leftarrow (z_i^A a_i + r_i^A - ((1-\iota'_i)\chi_i + \iota'_i\chi'_i) \bmod 2^{\lambda+1}, (1-z_i^A)a_i + r_i^A - ((1-\iota'_i)\chi'_i + \iota'_i\chi_i) \bmod 2^{\lambda+1})_{i\leq\mu}$. Let $(R_i^B)_{i\leq\mu} \leftarrow (v^{z_i^B} + \eta_i 2^{\lambda+1})_{i\leq\mu}$ denote Bob's output.
5. Alice sends $(\iota''_i)_{i\leq\mu} \leftarrow (z_i^A \oplus \zeta_i)_{i\leq\mu}$. Bob picks $r_i^B \leftarrow_R \mathbb{Z}_{2^{\lambda+1}}$ for $i = 1$ to μ, and sends $(w_i^0, w_i^1)_{i\leq\mu} \leftarrow ((-z_i^B b_i + r_i^B - ((1-\iota''_i)\xi_i + \iota''_i\xi'_i) \bmod 2^{\lambda+1}, (z_i^B - 1)b_i + r_i^B - ((1-\iota''_i)\xi'_i + \iota''_i\xi_i) \bmod 2^{\lambda+1})_{i\leq\mu}$. Let $(R_i^A)_{i\leq\mu} \leftarrow (w^{z_i^A} + \nu_i \bmod 2^{\lambda+1})_{i\leq\mu}$ denote Alice's output.
6. Alice and Bob perform the online step of a $(\lambda+1)$-bit secure comparison protocol, on respective inputs $s_A \leftarrow \sum_i (r_i^A - R_i^A) \bmod 2^{\lambda+1}$ and $s_B \leftarrow 2^\lambda - \sum_i (r_i^B - R_i^B) \bmod 2^{\lambda+1}$.

On the Preprocessing of OTs. One can see from the description of the protocol that we preprocess the oblivious transfers, as outlined in subsection 2.1. This is efficient as long as it transmits short strings only. For the $\mathsf{OT}_{\log(\mu+1)}$, this will be verified even for huge values of ℓ (say, millions of bits). However, for the $\mathsf{OT}_{\lambda+1}$, the optimal value of λ will become quite large for inputs with a few thousands of bits. If this is the case, the preprocessing steps 3 and 4 should be removed, and the oblivious transfers on $(\lambda+1)$-bit ,àk;inputs should be performed directly during the online phase. We include these steps in our description of the protocol as for most realistic values of ℓ (say, up to a few hundreds of bits, which captures nearly any plausible application), λ will remain small.

Intuition of the Protocol. The inputs a and b of the players are divided into μ blocks of size λ. The protocol builds upon the observation that to compare the two inputs, it is sufficient to compare the *first block on which they differ*[8]. The purpose of steps $1-3$ is therefore to let Alice and Bob compute shares $(z_i^A, z_i^B)_i$ of bits $(z_i)_i = (z_i^A \oplus z_i^B)_i$ such that $z_i = 1$ if and only if i is the highest index satisfying $a_i \neq b_i$ (let us denote i^* this index). This is done by first computing equality tests for each of their blocks; let $t_1 \cdots t_\mu$ denote the μ bits such that $t_i = 1$ if and only if $a_i = b_i$. Using oblivious transfers, the players get shares of $(\sum_{j\leq i} t_i \bmod \mu+1)_{i\leq\mu}$; observe that those values are zero for all $i < i^*$ and non-zero for all $i \geq i^*$. Using equality tests again, the players get shares of values which are therefore 0 for all $i < i^*$ and 1 for all $i \geq i^*$. Computing the differences locally gives shares of bits z_i which are all zero, except for the z_{i^*}, which is 1. Those selection bits allow the players to compute shares of $\sum_i z_i a_i = a_{i^*}$ and $\sum_i z_i b_i = b_{i^*}$, using oblivious transfers again. The reason why the OTs are performed modulo $2^{\lambda+1}$ will become clear afterward.

It remains for the players to execute a secure comparison protocol on a_{i^*} and b_{i^*}. However, note that those values are *shared* between the players (modulo $2^{\lambda+1}$), and not privately known to each party. Hence, the players must first reduce the problem of securely comparing shared value to securely comparing

[8] A similar intuition is used in [Tof11, LT13]

values held by each player. To do so, we use the following observation, initially made in [NO07]: for any modulus p, the value $[a_{i^*} \leq b_{i^*}]$ can be determined from $[a_{i^*} \leq p/2]$, $[b_{i^*} \leq p/2]$, and $[a_{i^*} - b_{i^*} \bmod p \leq p/2]$. More precisely, one can observe that we can compute $[a_{i^*} \leq b_{i^*}]$ as $[a_{i^*} \leq b_{i^*}] = (1 \oplus [a_{i^*} \leq p/2] \oplus [b_{i^*} \leq p/2]) \cdot (1 \oplus [a_{i^*} - b_{i^*} \bmod p \leq p/2]) \oplus [a_{i^*} \leq p/2]$.

In our protocol, we have set $p = 2^{\lambda+1}$, so that $p/2 = 2^{\lambda}$. As all the blocks a_i, b_i are λ-bit long, the conditions $[a_{i^*} \leq p/2 = 2^{\lambda}]$ and $[b_{i^*} \leq p/2 = 2^{\lambda}]$ are necessarily verified. Therefore, the equation becomes $[a_{i^*} \leq b_{i^*}] = [a_{i^*} - b_{i^*} \bmod 2^{\lambda+1} \leq 2^{\lambda}]$. The latter comparison involves a shared value, and a public value. Let us denote $(a^A_{i^*}, b^A_{i^*})$ and $(a^B_{i^*}, b^B_{i^*})$ the shares of (a_{i^*}, b_{i^*}) modulo $2^{\lambda+1}$ held by Alice and Bob respectively. Observe that

$$[a_{i^*} - b_{i^*} \bmod 2^{\lambda+1} \leq 2^{\lambda}] = [a^A_{i^*} - b^A_{i^*} \bmod 2^{\lambda+1} \leq 2^{\lambda} - a^B_{i^*} + b^B_{i^*} \bmod 2^{\lambda+1}].$$

Hence, Alice computes $s_A \leftarrow a^A_{i^*} - b^A_{i^*} \bmod 2^{\lambda+1}$, Bob computes $s_B \leftarrow 2^{\lambda} - a^B_{i^*} + b^B_{i^*} \bmod 2^{\lambda+1}$, and both players can conclude the protocol by performing a secure comparison on the $(\lambda + 1)$-bit values s_A and s_B.

4.2 Security Analysis (sketch)

Let us sketch an argument showing that our secure comparison protocol is secure against honest-but-curious adversaries, for both Alice and Bob, if the oblivious transfer protocol is secure against honest-but-curious adversaries. The argument for the small-string secure comparison protocol is identical than for the small-string equality test protocol (both protocols having exactly the same structure). In the reduction protocol, the sender privacy ensures that the view of the receiver is always perfectly indistinguishable from random, as the inputs to each (pre-processed) OT are independently random; the receiver privacy of the OTs states that the sender learns nothing. During each equality test and under the security of the underlying OT scheme, the view of both players (including their output in the ET) is indistinguishable from random. Therefore, the view of both players during the reduction protocol is perfectly indistinguishable from random. As the output of each reduction protocol is a uniformly random $(\lambda + 1)$-bit string for each player, sequentially performing multiple reduction protocols will not alter the security.

4.3 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 settings, an (almost) constant-round and a logarithmic-round setting. In the constant-round setting, c reduction steps are performed (for some constant c); the final secure comparison is performed using the protocol of [KSS09] (which transmits $O(\kappa\ell)$ bits on ℓ-bit inputs). In the logarithmic-round setting, our protocol is recursively applied until the strings are small enough, then the small-string secure comparison protocol is called.

Table 3: Amortized communication of ℓ-bit SC

	SC 1		SC 2		[KSS09]	
ℓ	length	rounds	length	round	length	round
			Preprocessing Phase			
4	1185 bits	2 rounds	-		1032 bits	2 rounds
8[1]	3572 bits	2 rounds	-		2064 bits	2 rounds
16[2]	8396 bits	2 rounds	-		4128 bits	2 rounds
32[3]	15120 bits	3 rounds	13450 bits	3 rounds	8256 bits	2 rounds
64[4]	31388 bits	3 rounds	29880 bits	3 rounds	16512 bits	2 rounds
128[5]	52121 bits	3 rounds	49291 bits	3 rounds	33024 bits	2 rounds
			Online Phase			
4	30 bits	2 rounds	-		2048 bits	1 round
8	162 bits	6 rounds	-		4096 bits	1 round
16	308 bits	6 rounds	-		8192 bits	1 round
32	530 bits	12 rounds	4014 bits	7 rounds	16384 bits	1 round
64	1120 bits	12 rounds	5154 bits	7 rounds	32768 bits	1 round
128	2101 bits	12 rounds	7071 bits	7 rounds	65536 bits	1 round

[1] $\mu = 4, \lambda = 2$
[2] $\mu = 6, \lambda = 3$ reduces the SC to $\ell = 5$, then $\mu = 3, \lambda = 2$ for the next reduction
[3] $\mu = 6, \lambda = 6$ reduces the SC to $\ell = 7$, then $\mu = 4, \lambda = 2$ for the next reduction
[4] $\mu = 10, \lambda = 7$ reduces the SC to $\ell = 8$, then $\mu = 4, \lambda = 2$ for the next reduction
[5] $\mu = 15, \lambda = 9$ reduces the SC to $\ell = 10$, then $\mu = 4, \lambda = 2$ for the next reduction

Communication Complexity. The full protocol involves μ parallel executions of ET_λ, $\mathsf{ET}_{\log(\mu+1)}$, $\mathsf{OT}_{\log(\mu+1)}$, 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 strings to be compared can be reduced while statistically preserving their equality. In the (almost) 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 [KK13, ALSZ13]. 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, μ 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. In practice, we apply the small-string secure comparison protocol as soon as $\lambda \leq 4$; in the equality tests, we set the threshold of the size-reduction protocol to $n = 4$. 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 [KSS09] 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 overall communication is reduced by 30 to 45% compared to the garbled circuit approach, and the communication during the online phase is considerably lower.

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 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.

References

ABZS13. M. Aliasgari, M. Blanton, Y. Zhang, and A. Steele. Secure computation on floating point numbers. In *NDSS 2013*, February 2013. (Page 8.)

ACM$^+$13. A. Aly, E. Cuvelier, S. Mawet, O. Pereira, and M. V. Vyve. Securely solving simple combinatorial graph problems. In *FC 2013*, *LNCS* 7859, pages 239–257. Springer, April 2013. (Page 8.)

ALSZ13. G. Asharov, Y. Lindell, T. Schneider, and M. Zohner. More efficient oblivious transfer and extensions for faster secure computation. In *ACM CCS 13*, pages 535–548. ACM Press, November 2013. (Pages 11, 12, 17, 19, and 25.)

ARL$^+$13. E. Ayday, J. L. Raisaro, M. Laren, P. Jack, J. Fellay, and J.-P. Hubaux. 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)*, number EPFL-CONF-187118, 2013. (Page 8.)

AV15. A. Aly and M. V. Vyve. Securely solving classical network flow problems. In *ICISC 14*, LNCS, pages 205–221. Springer, 2015. (Page 8.)

BAC$^+$11. R. Belguechi, V. Alimi, E. Cherrier, P. Lacharme, and C. Rosenberger. An overview on privacy preserving biometrics. *Recent Application in Biometrics*, pages 65–84, 2011. (Page 7.)

Bea95. D. Beaver. Precomputing oblivious transfer. In *CRYPTO'95*, *LNCS* 963, pages 97–109. Springer, August 1995. (Page 10.)

Bea96. D. Beaver. Correlated pseudorandomness and the complexity of private computations. In *28th ACM STOC*, pages 479–488. ACM Press, May 1996. (Page 11.)

BKU14. A. Bektaş, M. S. Kiraz, and O. Uzunkol. A secure and efficient protocol for electronic treasury auctions. In *Cryptography and Information Security in the Balkans*, pages 123–140. Springer, 2014. (Page 7.)

BPTG14. R. Bost, R. A. Popa, S. Tu, and S. Goldwasser. Machine learning classification over encrypted data. Cryptology ePrint Archive, Report 2014/331, 2014. `http://eprint.iacr.org/2014/331`. (Page 8.)

BR93. M. Bellare and P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *ACM CCS 93*, pages 62–73. ACM Press, November 1993. (Page 10.)

BS15. M. Blanton and S. Saraph. Oblivious maximum bipartite matching size algorithm with applications to secure fingerprint identification. LNCS, pages 384–406. Springer, 2015. (Page 8.)

CC15. W.-T. Chu and F.-C. Chang. A privacy-preserving bipartite graph matching framework for multimedia analysis and retrieval. In *Proceedings of the 5th ACM on International Conference on Multimedia Retrieval*, pages 243–250. ACM, 2015. (Page 8.)

Cd10a. O. Catrina and S. de Hoogh. Improved primitives for secure multiparty integer computation. In *SCN 10*, *LNCS* 6280, pages 182–199. Springer, September 2010. (Pages 4 and 8.)

Cd10b. O. Catrina and S. de Hoogh. Secure multiparty linear programming using fixed-point arithmetic. In *ESORICS 2010*, *LNCS* 6345, pages 134–150. Springer, September 2010. (Pages 8 and 9.)

CDN01. R. Cramer, I. Damgård, and J. B. Nielsen. Multiparty computation from threshold homomorphic encryption. In *EUROCRYPT 2001*, *LNCS* 2045, pages 280–299. Springer, May 2001. (Page 4.)

CKP07. R. Cramer, E. Kiltz, and C. Padró. A note on secure computation of the Moore-Penrose pseudoinverse and its application to secure linear algebra. In *CRYPTO 2007*, *LNCS* 4622, pages 613–630. Springer, August 2007. (Page 6.)

CPP16. G. Couteau, T. Peters, and D. Pointcheval. Encryption switching protocols. to appear in the proceedings of CRYPTO, 2016. `http://eprint.iacr.org/2015/990`. (Page 6.)

CS10. O. Catrina and A. Saxena. Secure computation with fixed-point numbers. In *FC 2010*, *LNCS* 6052, pages 35–50. Springer, January 2010. (Page 8.)

DFK$^+$06. I. Damgård, M. Fitzi, E. Kiltz, J. B. Nielsen, and T. Toft. Unconditionally secure constant-rounds multi-party computation for equality, comparison, bits and exponentiation. In *TCC 2006*, *LNCS* 3876, pages 285–304. Springer, March 2006. (Page 4.)

DGK07. I. Damgård, M. Geisler, and M. Krøigaard. Efficient and secure comparison for on-line auctions. In *ACISP 07*, *LNCS* 4586, pages 416–430. Springer, July 2007. (Pages 3, 4, and 8.)

DGK08. I. Damgård, M. Geisler, and M. Krøigaard. A correction to "efficient and secure comparison for on-line auctions". Cryptology ePrint Archive, Report 2008/321, 2008. `http://eprint.iacr.org/2008/321`. (Page 3.)

DJ01. I. Damgård and M. Jurik. A generalisation, a simplification and some applications of Paillier's probabilistic public-key system. In *PKC 2001*, *LNCS* 1992, pages 119–136. Springer, February 2001. (Pages 31 and 33.)

EFG$^+$09. Z. Erkin, M. Franz, J. Guajardo, S. Katzenbeisser, I. Lagendijk, and T. Toft. Privacy-preserving face recognition. In *Privacy Enhancing Technologies*, pages 235–253. Springer, 2009. (Page 7.)

EGL82. S. Even, O. Goldreich, and A. Lempel. A randomized protocol for signing contracts. In *CRYPTO'82*, pages 205–210. Plenum Press, New York, USA, 1982. (Pages 4, 5, and 9.)

EVTL12. Z. Erkin, T. Veugen, T. Toft, and R. L. Lagendijk. Generating private recommendations efficiently using homomorphic encryption and data packing. *Information Forensics and Security, IEEE Transactions on*, 7(3):1053–1066, 2012. (Page 8.)

GHJR15. C. Gentry, S. Halevi, C. S. Jutla, and M. Raykova. Private database access with HE-over-ORAM architecture. In *ACNS 15*, LNCS, pages 172–191. Springer, 2015. (Pages 4, 6, 12, 19, and 31.)

GMW87a. O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game or A completeness theorem for protocols with honest majority. In *19th ACM STOC*, pages 218–229. ACM Press, May 1987. (Page 1.)

GMW87b. O. Goldreich, S. Micali, and A. Wigderson. How to prove all NP-statements in zero-knowledge, and a methodology of cryptographic protocol design. In *CRYPTO'86*, *LNCS* 263, pages 171–185. Springer, August 1987. (Page 1.)

Gol87. O. Goldreich. Towards a theory of software protection and simulation by oblivious RAMs. In *19th ACM STOC*, pages 182–194. ACM Press, May 1987. (Page 7.)

Goo10. M. T. Goodrich. Randomized shellsort: A simple oblivious sorting algorithm. In *21st SODA*, pages 1262–1277. ACM-SIAM, January 2010. (Page 7.)

Goo14. M. T. Goodrich. Zig-zag sort: a simple deterministic data-oblivious sorting algorithm running in $O(n \log n)$ time. In *46th ACM STOC*, pages 684–693. ACM Press, 2014. (Page 7.)

GSV07. J. A. Garay, B. Schoenmakers, and J. Villegas. Practical and secure solutions for integer comparison. In *PKC 2007*, *LNCS* 4450, pages 330–342. Springer, April 2007. (Page 4.)

HEK12. Y. Huang, D. Evans, and J. Katz. Private set intersection: Are garbled circuits better than custom protocols? In *NDSS 2012*, February 2012. (Page 7.)

HGW$^+$15. Q. Huang, Y. Gui, F. Wu, G. Chen, and Q. Zhang. A general privacy-preserving auction mechanism for secondary spectrum markets. 2015. (Page 8.)

HICT14. K. Hamada, D. Ikarashi, K. Chida, and K. Takahashi. 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`. (Page 7.)

HKI$^+$12. K. Hamada, R. Kikuchi, D. Ikarashi, K. Chida, and K. Takahashi. Practically efficient multi-party sorting protocols from comparison sort algorithms. In *Information Security and Cryptology–ICISC 2012*, pages 202–216. Springer, 2012. (Page 7.)

HT14. C. Hazay and T. Toft. Computationally secure pattern matching in the presence of malicious adversaries. *Journal of Cryptology*, 27(2):358–395, April 2014. (Page 6.)

IKNP03. Y. Ishai, J. Kilian, K. Nissim, and E. Petrank. Extending oblivious transfers efficiently. In *CRYPTO 2003*, *LNCS* 2729, pages 145–161. Springer, August 2003. (Pages 3, 5, and 11.)

IPS08. Y. Ishai, M. Prabhakaran, and A. Sahai. Founding cryptography on oblivious transfer - efficiently. In *CRYPTO 2008*, *LNCS* 5157, pages 572–591. Springer, August 2008. (Page 9.)

JKU11. K. V. Jónsson, G. Kreitz, and M. Uddin. Secure multi-party sorting and applications. Cryptology ePrint Archive, Report 2011/122, 2011. `http://eprint.iacr.org/2011/122`. (Page 7.)

Kil88. J. Kilian. Founding cryptography on oblivious transfer. In *20th ACM STOC*, pages 20–31. ACM Press, May 1988. (Page 9.)

KK13. V. Kolesnikov and R. Kumaresan. Improved OT extension for transferring short secrets. In *CRYPTO 2013, Part II*, *LNCS* 8043, pages 54–70. Springer, August 2013. (Pages 5, 11, 12, 16, 19, and 25.)

KS08. V. Kolesnikov and T. Schneider. Improved garbled circuit: Free XOR gates and applications. In *ICALP 2008, Part II*, *LNCS* 5126, pages 486–498. Springer, July 2008. (Pages 3, 16, 19, and 20.)

KSS09. V. Kolesnikov, A.-R. Sadeghi, and T. Schneider. Improved garbled circuit building blocks and applications to auctions and computing minima. In *CANS 09*, *LNCS* 5888, pages 1–20. Springer, December 2009. (Pages 3, 4, 5, 16, 18, 19, 20, 22, 24, 25, and 26.)

Lau15. P. Laud. A private lookup protocol with low online complexity for secure multiparty computation. In *ICICS 14*, LNCS, pages 143–157. Springer, 2015. (Page 8.)

Lie12. M. Liedel. Secure distributed computation of the square root and applications. In *Information Security Practice and Experience*, pages 277–288. Springer, 2012. (Page 8.)

LLY$^+$16. P. Li, T. Li, Z.-A. Yao, C.-M. Tang, and J. Li. Privacy-preserving outsourcing of image feature extraction in cloud computing. *Soft Computing*, pages 1–11, 2016. (Page 8.)

LT13. H. Lipmaa and T. Toft. Secure equality and greater-than tests with sublinear online complexity. In *ICALP 2013, Part II*, *LNCS* 7966, pages 645–656. Springer, July 2013. (Pages 4, 6, 19, and 23.)

Mu14. B. Mu. A survey on secure processing of similarity queries. 2014. (Page 7.)

NIIO14. T. Nishide, M. Iwamoto, A. Iwasaki, and K. Ohta. Secure (m+ 1) st-price auction with automatic tie-break. In *Trusted Systems*, pages 422–437. Springer, 2014. (Page 7.)

NO07. T. Nishide and K. Ohta. Multiparty computation for interval, equality, and comparison without bit-decomposition protocol. In *PKC 2007*, *LNCS* 4450, pages 343–360. Springer, April 2007. (Pages 4 and 24.)

NP01. M. Naor and B. Pinkas. Efficient oblivious transfer protocols. In *12th SODA*, pages 448–457. ACM-SIAM, January 2001. (Pages 9, 10, 17, and 18.)

Pai99. P. Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *EUROCRYPT'99*, *LNCS* 1592, pages 223–238. Springer, May 1999. (Pages 4 and 31.)

PR10. B. Pinkas and T. Reinman. Oblivious RAM revisited. In *CRYPTO 2010*, *LNCS* 6223, pages 502–519. Springer, August 2010. (Page 7.)

Rab81. M. Rabin. How to exchange secrets by oblivious transfer. *Technical Report TR-81, Harvard University,*, 1981. (Pages 4, 5, and 9.)

RPV$^+$14. Y. Rahulamathavan, R. C.-W. Phan, S. Veluru, K. Cumanan, and M. Rajarajan. 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. (Page 8.)

RT07. T. I. Reistad and T. Toft. Secret sharing comparison by transformation and rotation. In *Information Theoretic Security*, pages 169–180. Springer, 2007. (Page 4.)

SJB14. B. K. Samanthula, W. Jiang, and E. Bertino. Lightweight and secure two-party range queries over outsourced encrypted databases. *arXiv preprint arXiv:1401.3768*, 2014. (Page 8.)

SSW10. A.-R. Sadeghi, T. Schneider, and I. Wehrenberg. Efficient privacy-preserving face recognition. In *ICISC 09*, *LNCS* 5984, pages 229–244. Springer, December 2010. (Page 7.)

Tof09. T. Toft. Solving linear programs using multiparty computation. In *FC 2009*, *LNCS* 5628, pages 90–107. Springer, February 2009. (Pages 6 and 8.)

Tof11. T. Toft. Sub-linear, secure comparison with two non-colluding parties. In *PKC 2011*, *LNCS* 6571, pages 174–191. Springer, March 2011. (Pages 4, 6, 19, and 23.)

VBdHE15. T. Veugen, F. Blom, S. J. de Hoogh, and Z. Erkin. Secure comparison protocols in the semi-honest model. *Selected Topics in Signal Processing, IEEE Journal of*, 9(7):1217–1228, 2015. (Page 3.)

WFNL15. D. J. Wu, T. Feng, M. Naehrig, and K. Lauter. Privately evaluating decision trees and random forests. Cryptology ePrint Archive, Report 2015/386, 2015. `http://eprint.iacr.org/2015/386`. (Page 8.)

WS08. P. Williams and R. Sion. Usable PIR. In *NDSS 2008*, February 2008. (Page 7.)

WSC08. P. Williams, R. Sion, and B. Carbunar. Building castles out of mud: practical access pattern privacy and correctness on untrusted storage. In *ACM CCS 08*, pages 139–148. ACM Press, October 2008. (Page 7.)

XT14. C. Xiang and C. Tang. Privacy-preserving face recognition with outsourced computation. Cryptology ePrint Archive, Report 2014/969, 2014. `http://eprint.iacr.org/2014/969`. (Page 7.)

Yao86. A. C.-C. Yao. How to generate and exchange secrets (extended abstract). In *27th FOCS*, pages 162–167. IEEE Computer Society Press, October 1986. (Pages 1, 2, and 3.)

Yu11. C.-H. Yu. Sign modules in secure arithmetic circuits. Cryptology ePrint Archive, Report 2011/539, 2011. `http://eprint.iacr.org/2011/539`. (Page 4.)

YY12. C.-H. Yu and B.-Y. Yang. Probabilistically correct secure arithmetic computation for modular conversion, zero test, comparison, MOD and exponentiation. In *SCN 12*, *LNCS* 7485, pages 426–444. Springer, September 2012. (Pages 4 and 6.)

Zha11. B. Zhang. Generic constant-round oblivious sorting algorithm for MPC. In *ProvSec 2011*, *LNCS* 6980, pages 240–256. Springer, October 2011. (Page 7.)

ZRE15. S. Zahur, M. Rosulek, and D. Evans. Two halves make a whole - reducing data transfer in garbled circuits using half gates. LNCS, pages 220–250. Springer, 2015. (Pages 3 and 16.)

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 [GHJR15] (which relies on ciphertext packing to amortize the communication of equality tests), and in fact matches the communication complexity of [GHJR15], which has to our knowledge the best communication complexity among existing works. However, contrary to [GHJR15], 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 [Pai99] or the Damgard-Jurik scheme [DJ01]. For concrete parameters, the amortized communication improves upon every prior ET protocol we know of, including the OT-based protocol described Section 3.

A.1 Encryption Scheme

Definition 1. *(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 probability at most negligibly higher than 1/2.

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 imply 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 $P = 2^{\text{poly}(\kappa)}$. 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 can simply corresponds to taking the modulus bigger than the threshold.

A.2 Batch Equality Test

We let $\Pi = (\mathsf{Setup}, \mathsf{Enc}, \mathsf{Dec})$ 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. This corresponds to a batch $\ell-\mathsf{EPSR}$.

- 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 $\mathsf{Setup}(1^\kappa, 2^{\kappa+2\log M+2})$ and gets $(\mathsf{pk}, \mathsf{sk})$; pk implicitely defines a plaintext space $\mathbb{Z}_P$ of size $P \geq 2^{\kappa+2\log M+2}$. 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 $\lceil \log p_i \rceil$, 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 integer 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 $\lceil \log(p_1 + 1) \rceil$ to $\lceil \log(p_n + 1) \rceil$. 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.

Concrete Efficiency. We now estimate the concrete efficiency of our protocol, and compare it to our previous solution. We use the Damgard-Jurik generalization [DJ01] of the Paillier encryption scheme, which have 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 $s \cdot 2048 \geq 2 \log M(\ell) + \kappa + 1$, where $M(\ell)$ is the product of the smallest 1000 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 $\ell > 16$.

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