Secure Equality and Greater-Than Tests with Sublinear Online Complexity

Helger Lipmaa[1] and Tomas Toft[2]

[1] Institute of CS, University of Tartu, Estonia
[2] Dept. of CS, Aarhus University, Denmark

Abstract. Secure multiparty computation (MPC) allows multiple parties to evaluate functions without disclosing the private inputs. Secure comparisons (testing equality and greater-than) are important primitives required by many MPC applications. We propose two equality tests for ℓ-bit values with $O(1)$ online communication that require $O(\ell)$ respectively $O(\kappa)$ total work, where κ is a correctness parameter.

Combining these with ideas of Toft [16], we obtain (i) a greater-than protocol with sublinear online complexity in the arithmetic black-box model ($O(c)$ rounds and $O(c \cdot \ell^{1/c})$ work online, with $c = \log \ell$ resulting in logarithmic online work). In difference to Toft, we do not assume two mutually incorruptible parties, but $O(\ell)$ offline work is required, and (ii) two greater-than protocols with the same online complexity as the above, but with overall complexity reduced to $O(\log \ell(\kappa + \text{loglog}\, \ell))$ and $O(c \cdot \ell^{1/c}(\kappa + \log \ell))$; these require two mutually incorruptible parties, but are highly competitive with respect to online complexity when compared to existing protocols.

Keywords: Additively homomorphic encryption, arithmetic black box, secure comparison, secure equality test.

1 Introduction

Secure multiparty computation (MPC) considers the following problem: n parties hold inputs, $x_1, \ldots, x_n$, for a function, f; they wish to evaluate f without disclosing their inputs to each other or third-parties. Numerous solutions to this problem exist; many provide secure arithmetic over a field or ring, e.g., $\mathbb{Z}_M$ for an appropriate M, by relying either on secret sharing or additively homomorphic encryption. The overall structure of those solutions is similar, thus the details of the constructions may be abstracted away and MPC-protocols can be constructed based on secure arithmetic. This idea was formalized as the arithmetic black-box (ABB) by Damgård and Nielsen [7]. For a longer discussion of MPC and the ABB, see Sect. 2.

Secure $\mathbb{Z}_M$-arithmetic may be used to emulate integer computation when inputs/outputs are less than M (which typically can be chosen quite freely). However, other operations may be needed. Secure comparison – equality testing (Eq) and greater-than testing (GT) – are two important problems in the (MPC)

F.V. Fomin et al. (Eds.): ICALP 2013, Part II, LNCS 7966, pp. 645–656, 2013.

Table 1. A comparison of sublinear GT protocols for bitlength ℓ

Result	Online rounds	Online work	Overall work	Correctness
Adversary structure with two mutually incorruptible parties				
[16]	$O(c)$	$O(c \cdot \ell^{1/c}(\kappa + \log \ell))$	$O(c \cdot \ell^{1/c}(\kappa + \log \ell))$	Statistical
[16]	$O(\log \ell)$	$O(\log \ell(\kappa + \text{loglog } \ell))$	$O(\log \ell(\kappa + \text{loglog } \ell))$	Statistical
This paper	$O(c)$	$O(c \cdot \ell^{1/c})$	$O(c \cdot \ell^{1/c}(\kappa + \log \ell))$	Statistical
This paper	$O(\log \ell)$	$O(\log \ell)$	$O(\log \ell(\kappa + \text{loglog } \ell)))$	Statistical
Arbitrary adversary structure				
[18]	$O(1)$	$O(\sqrt{\ell/\log \ell})$	$O(\ell)$	Perfect
This paper	$O(c)$	$O(c \cdot \ell^{1/c})$	$O(\ell)$	Perfect
This paper	$O(\log \ell)$	$O(\log \ell)$	$O(\ell)$	Perfect

literature. They are required for tasks as diverse as auctions, data-mining, and benchmarking. A prime example is the first real-world MPC execution [4], which required both integer additions and GT tests.

In this paper, we introduce two new Eq tests and improve over state of the art GT testing in the ABB model. The main focus is online efficiency, i.e., parties may generate joint randomness in advance (e.g, while setting up an auction) to increase efficiency once the inputs have been supplied (bids have been given).

Related Work. Secure comparison and its applications is a very active topic with too many papers to mention all. Damgård et al. [6] proposed the first constant-rounds protocols which required $O(\ell \log \ell)$ secure multiplications. Nishide and Ohta [13] improved this to $O(\ell)$ work for GT and $O(\kappa)$ work for equality where κ is a correctness parameter.

Until recently, all GT tests had a complexity (at least) linear in the bitlength, ℓ, of the inputs, but in [16], Toft proposed the first sublinear constructions. These utilized proofs of boundedness and required the presence of 2 mutually incorruptible parties, i.e., one of two named parties was required to be honest. This is naturally satisfied in the two-party case ($n = 2$), but the multiparty case is left with either a corruption threshold of 1 or a non-standard adversary structure. In [18], Yu proposed a sublinear, constant-rounds protocol in the ABB model based on *sign modules*. His protocol requires $O(\sqrt{\ell/\log \ell})$ operations online and works for an ABB over a finite field, i.e., prime M. It does not appear that the ideas work with composite M such as is needed by Paillier encryption. See Table 1 for an overview of existing sublinear GT tests.

Contribution. We propose a collection of actively secure protocols. We first introduce two new protocols for equality testing of ℓ-bit values. The first is perfectly correct with $O(1)$ ABB-operations online and $O(\ell)$ ABB-operations overall. The second reduces overall communication to $O(\kappa)$ at the cost of imperfect correctness, i.e., κ is a correctness parameter; it also requires two mutually

incorruptible parties. Both improve online complexity dramatically over previous work. Additionally, we use these in combination with ideas of [16] to obtain new GT tests for ℓ-bit values in the ABB model. We end up with multiple variations.

First, ABB-protocols with $O(\log \ell)$ work and rounds (respectively $O(c \cdot \ell^{1/c})$ work in $O(c)$ rounds for constant c) online; $O(\ell)$ work overall. Second, we reduce overall work to $O(\log \ell \cdot (\kappa + \log\log \ell))$ ($O(c \cdot \ell^{1/c}(\kappa + \log \ell))$ respectively) at the cost of requiring two mutually incorruptible parties. All constructions require proofs of boundedness to prevent active attacks. In contrast to [18], we do not utilize sign modules, hence our protocols work for Paillier encryption-based MPC as well. In that setting our GT tests are the first with sublinear online complexity and arbitrary adversary structure.

2 Preliminaries

The Arithmetic Black-Box. Many MPC protocols work by having parties supply their inputs "secretly," e.g., using secret sharing, which allows a value to be split between parties such that it remains unknown unless sufficiently many agree. A homomorphic scheme allows parties to compute sums, while secure multiplication requires interaction. Once the desired result has been computed, it is straightforward to output it by reconstructing. The arithmetic black-box of [7] captures this type of behaviour, making it a convenient model for presenting MPC protocols. This allows protocol construction with focus on the task at hand rather than "irrelevant details" such as the specifics and security guarantees of the underlying cryptographic primitives.

Formally, the arithmetic black-box is an ideal functionality, $\mathcal{F}_{\text{ABB}}$, and protocols are constructed in a hybrid model where access to this functionality is given. $\mathcal{F}_{\text{ABB}}$ can be thought of as a (virtual) trusted third party, who provides storage of elements of a ring, $\mathbb{Z}_M$, as well as arithmetic computation on stored values. Here, M will be either a prime or an RSA-modulus, i.e., the product of two large, odd primes. We provide an intuitive presentation of the ABB here; see [7] or [17] for a formal definition. Full simulation-based proofs are possible; due to space constraints we merely sketch privacy proofs.

Secure storage (input/output) can be thought of as secret sharing and we use the notation of Damgård et al. [6], writing ABB-values in square brackets, $[\![x]\!]$. ABB-arithmetic is written using *"plaintext space,"* infix operators, e.g., $[\![x \cdot y + z]\!] \leftarrow [\![x]\!] \cdot [\![y]\!] + [\![z]\!]$. Naturally, such operations eventually refer to protocols between $P_1, \ldots, P_n$, e.g., the protocols of Ben-Or et al. [3].

The complexity of a protocol in the $\mathcal{F}_{\text{ABB}}$ hybrid model is the number of basic operations performed, input/output and arithmetic. We assume that these operations may be executed concurrently, i.e., that executing the underlying cryptographic protocols concurrently does not violate security. Round complexity is defined as the number of sequential sets of concurrent operations performed (basic operations typically require a constant number of rounds; in this case constant-rounds in the ABB model implies constant-rounds in the actual protocol). Finally, we focus on communication complexity and therefore consider

addition costless; typical ABB realizations are based on additively homomorphic primitives, and this is a standard choice. Additionally, ABB-computatation occurs in two phases: (i) random values are generated within $\mathcal{F}_{\text{ABB}}$ before the inputs are known (preprocessing or offline phase), and (ii) when the inputs are available within $\mathcal{F}_{\text{ABB}}$, the result is computed (online phase). Focus is predominantly on the efficiency of the online phase.

Known ABB Constructions and Additional Primitives. The following known primitives are needed in the proposed constructions. These are exclusively needed as part of the preprocessing phase; in practice it may be preferable to utilize simpler (non-constant-rounds) solutions.

- `RandElem`: Generates a uniformly random, secret element of $\mathbb{Z}_M$ stored within the ABB. Considered as 1 multiplication and 1 rounds, [6].
- `RandBit`: Generates a uniformly random, secret bit stored within the ABB. $O(1)$ multiplications in $O(1)$ rounds, [6].[1]
- `RandBits`: Generates a uniformly random $\mathbb{Z}_M$-value r and its binary representation $r = \sum 2^i r_i$, $r_i \in \{0,1\}$ stored as elements of $\mathbb{Z}_M$, $O(\log M)$ multiplications in $O(1)$ rounds, [6].
- `RandInv`: Generates a uniformly random element in $\mathbb{Z}_M^*$ along with its inverse; $O(1)$ multiplications in $O(1)$ rounds, [6].
- $\texttt{prefix}_{\times}$: Prefix product takes a vector of invertible, secret values, $[\![r_1]\!]$, $[\![r_2]\!]$, ..., $[\![r_m]\!]$, and computes the prefix-product, i.e., $[\![\prod_{i=1}^{j} r_i]\!]$ for $1 \leq j \leq m$, using $O(m)$ ABB operations in $O(1)$ rounds [2,6].

We also require that the ABB can verify that an input is of bounded size, e.g., $[\![x]\!] < 2^\ell$. x is known by the inputter, P_i, so this corresponds to executing a *proof of boundedness*. A communication-efficiently solution ($\Theta(1)$ group elements) can be obtained using the sum-of-four-squares technique, [10]: P_i supplies an integer input (decomposed into squares) which is converted to a $\mathbb{Z}_M$ element; this can be done using *integer commitments* (for encryption) or *linear integer secret sharing scheme* [15] (for Shamir sharing). An alternative is to use the constant-communication non-interactive zero-knowledge argument of [5]; there, P_i commits to a vector of digits of x and uses the techniques of [8,11] to prove that the *encrypted* x belongs to the given range.

Disclose-If-Equal. In a disclose-if-equal (DIE) protocol between Alice and Bob, Alice gets to know Bob's secret β exactly if she encrypted x (where x is a value known to Bob only, or possibly to both). Otherwise, Alice should obtain a (pseudo)random plaintext. See [1,9] for original definitions.

If the plaintext space is $\mathbb{Z}_M$ for a prime M (as it is in the case of the secret sharing setting), one can use the following simple protocol inspired by [1] (here, Enc_{pk} means encryption by public key pk and Dec_{sk} means decryption by the corresponding secret key sk): (1) Alice sends $q \leftarrow Enc_{pk}(\alpha)$ to Bob. (2) If the ciphertext is invalid, Bob returns $\perp$. Otherwise, he returns $a \leftarrow (q \cdot Enc_{pk}(-x))^r \cdot$

[1] When M is an RSA-modulus, complexity is linear in the number of parties, for simplicity we assume that this is constant. (This is only used in preprocessing.)

$Enc_{pk}(\beta)$, where $r \leftarrow \mathbb{Z}_M$. (3) Alice computes $Dec_{sk}(a) = r(\alpha - x) + \beta$, which is equal to β when $\alpha = x$. Clearly, this protocol is perfectly complete, and encryption, decryption, and exponent-arithmetic can be replaced by ABB operations.

If M is not a prime but has sufficiently large prime factors (like in the case of existing additively homomorphic public-key cryptosystems), then the resulting DIE protocol, proposed by Laur and Lipmaa in [9], is somewhat more complicated. Let ℓ be the bitlength of β. Let $T \leftarrow \lfloor 2^{-\ell} \cdot M \rfloor$. Let $spf(M)$ be the smallest prime factor of the plaintext group order M. We assume $\ell \leq \frac{1}{2} \log_2 M + \log_2 \varepsilon$, where $\varepsilon \leq 2^{-80}$ is the hiding parameter. Here we assume that Bob knows the public key and Alice knows the secret key and the parties use an additively homomorphic public-key cryptosystem like the one by Paillier [14]. (1) Alice sends $q \leftarrow Enc_{pk}(\alpha)$ to Bob. (2) If the ciphertext is invalid, Bob returns $\perp$. Otherwise, he returns $a \leftarrow (q \cdot Enc_{pk}(-x))^r \cdot Enc_{pk}(\beta + 2^\ell \cdot t)$, where $r \leftarrow \mathbb{Z}_M$ and $t \leftarrow \mathbb{Z}_T$. (3) Alice computes $Dec_{sk}(a) \mod 2^\ell$.

As shown in [9], this protocol is $(1 - \varepsilon)$-semisimulatable [12] (that is, game-based computationally private against a malicious server, and simulation-based statistically private against a malicious client) as long as $2^{\ell-1}/spf(M)$ is bounded by ε. That is, if $x \neq \alpha$ then the distribution of $U(\mathbb{Z}_M) \cdot (\alpha - x) + 2^\ell \cdot U(\mathbb{Z}_T)$ is ε-far from the uniform distribution $U(\mathbb{Z}_M)$ on $\mathbb{Z}_M$. Since in the case of Paillier, $spf(M) \approx \sqrt{M}$, we need that $\ell - 1 - \frac{1}{2} \cdot \log_2 M \leq \log_2 \varepsilon$ or $\ell < \frac{1}{2} \cdot \log_2 M + \log_2 \varepsilon$, as mentioned. The idea behind including the additional term $2^\ell \cdot t$ in the Laur-Lipmaa protocol is that if M is composite, then $Dec_{pk}((q \cdot Enc_{pk}(-x))^{U(\mathbb{Z}_M)}) = U(\mathbb{Z}_M) \cdot (\alpha - x)$ can be a random element of a nontrivial subgroup of $\mathbb{Z}_M$ and thus far from random in $\mathbb{Z}_M$; adding $2^\ell \cdot U(\mathbb{Z}_T)$ guarantees that the result is almost uniform in $\mathbb{Z}_M$.

3 Secure Equality Tests

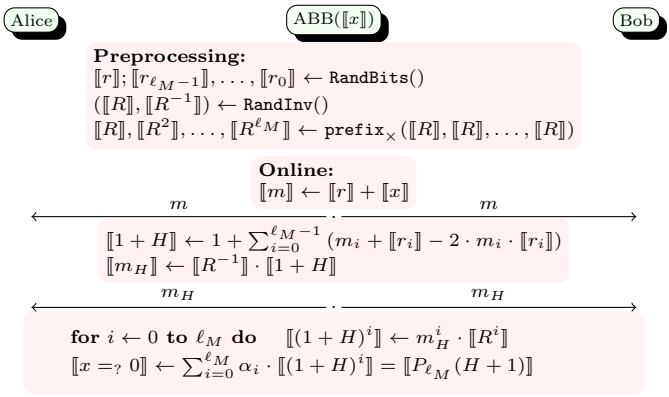
It is well-known that equality testing can be implemented using a zero-test (given additively homomorphic primitives) as $x = y \Leftrightarrow x - y = 0$; w.l.o.g., we focus on testing whether x equals 0 and present two new, secure protocols.

The first zero-test is based on the Hamming distance between a mask and the masked value. Complexity is linear in the bit-length, but only $O(1)$ ABB multiplications and outputs are needed online. Hence, when a preprocessing phase is present, this is highly efficient. Additionally, we present a variation allowing comparison of ℓ-bit numbers with $O(\ell)$ preprocessing and $O(1)$ work online, when $2^{\ell+k+\log n} \ll M$ for statistical security parameter k, and n parties.

The second approach is based on DIE and reduces the problem from arbitrary size inputs to κ-bit inputs, where κ is a correctness parameter, e.g., 80. This simpler problem may then be solved, e.g., using the Hamming-based approach.

3.1 Equality from Hamming Distance

Let $\ell_M = \lceil \log_2 M \rceil$ be the bitlength of M. The protocol, denoted $\mathtt{eq}_H$, is seen as Protocol 1. It is a variation of [13] with a highly optimized online phase. (Though phrased differently, Nishide and Ohta [13] did essentially the same thing).



Protocol 1. $\mathtt{eq}_H$, secure zero-testing based on Hamming distance

Correctness. Picking a uniformly random, unknown $[\![r]\!]$ and revealing $m = [\![x]\!] + [\![r]\!]$ allows testing $x = 0$ by testing whether $m = r$. If $[\![r]\!]$ in generated in binary, we can compute the Hamming distance $[\![H]\!] = \sum_{i=0}^{\ell_M-1} [\![r_i]\!] \oplus m_i = \sum_{i=0}^{\ell_M-1} (m_i + [\![r_i]\!] - 2 \cdot m_i \cdot [\![r_i]\!])$, and test if $H = 0$. Since $H \leq \ell_M$, the latter is simpler than the general zero-test. Let $P_{\ell_M}(x) = \sum_{i=0}^{\ell_M} \alpha_i \cdot x^i$ denote the (at most) ℓ_M-degree polynomial that maps 1 to 1 and $x \in \{2, 3, \ldots, \ell_M + 1\}$ to 0.[2] Evaluating P_{ℓ_M} at $1 + H$ determines $H + 1 = 1 \Leftrightarrow m = r \Leftrightarrow x = 0$.

To avoid $\Omega(\ell_M)$ online multiplications when computing the $\ell_M + 1$ powers of $[\![1 + H]\!]$, the following trick is used: A uniformly random value, $[\![R]\!] \in \mathbb{Z}_M^*$ is chosen in advance, and its exponents $[\![R^0]\!], [\![R^1]\!], \ldots, [\![R^{\ell_M}]\!]$ and inverse, $[\![R^{-1}]\!]$, are computed in the offline phase. In the online phase, $m_H = [\![R^{-1}]\!] \cdot [\![1 + H]\!]$ is computed and revealed, and the powers of $[\![1 + H]\!]$ are computed from the powers of $[\![R]\!]$ and the powers of m_H, which can be done locally by all parties: $m_H^i \cdot [\![R^i]\!] = (R^{-1}(1 + H))^i \cdot [\![R^i]\!] = [\![(1 + H)^i]\!]$.

Privacy. Two values are revealed in $\mathtt{eq}_H$, m and m_H. Since r is uniformly random in $\mathbb{Z}_M$, then so is $m = x + r$. Similarly, since $1 + \ell_M$ is smaller[3] than the smallest prime factor of M we have $1 + H \in \mathbb{Z}_M^*$. Thus $(1 + H) \cdot R^{-1}$ is uniformly random in $\mathbb{Z}_M^*$ as R is uniformly random. Simulation in the $\mathcal{F}_{\mathrm{ABB}}$-hybrid model consists of providing "fake" m and m_H distributed as the real ones.

Complexity. The preprocessing phase consists of generating $[\![r]\!]$ along with its bits, $[\![r_{i-1}]\!]$ as well as $[\![R]\!]$, $[\![R^{-1}]\!]$, and $[\![R^i]\!]$ for $i \in \{1, \ldots, \ell_M\}$. Overall this amounts to $O(\ell_M)$ work. Online, only 1 ABB-multiplication (to compute m_H) and 2 outputs are needed. Computing the Hamming distance and evaluating P_{ℓ_M} are costless.

[2] P_{ℓ_M} exists both when M is a prime or an RSA-modulus and the coefficients, α_i, can be computed using Lagrange interpolation. For technical reasons, the input to P_{ℓ_M} must belong to $\mathbb{Z}_M^*$, this is ensured by adding 1.

[3] Always the case since M is either a prime or the product of two large primes.

Bounded Inputs. If the input is of bounded size, $[\![x]\!] < 2^\ell$, and $2^{\ell+k+\log n} \ll M$ where k is a statistical security parameter, the following variation is possible: Each party P_j inputs a uniformly random k-bit value, $r^{(j)}$, and the n parties jointly generate ℓ random bits, $[\![r_i]\!]$, using `RandBit`. The ABB then computes $[\![r]\!] \leftarrow \sum_{j=1}^{n} [\![r^{(j)}]\!] \cdot 2^\ell + (\sum_{i=0}^{\ell-1} 2^i [\![r_i]\!])$. Here, r statistically masks x: $m \bmod 2^\ell$ is uniformly random, while a single $r^{(j)}$ masks the ℓ'th carrybit of the addition, $x + r$, i.e., $\lfloor m/2^\ell \rfloor$ is statistically indistinguishable from a sum of uniformly random k-bit values plus the r_i of malicious parties. Testing equality between $r \bmod 2^\ell$ and $m \bmod 2^\ell$ is sufficient; note that this zero-test allows equality testing even when the difference between the inputs is negative.

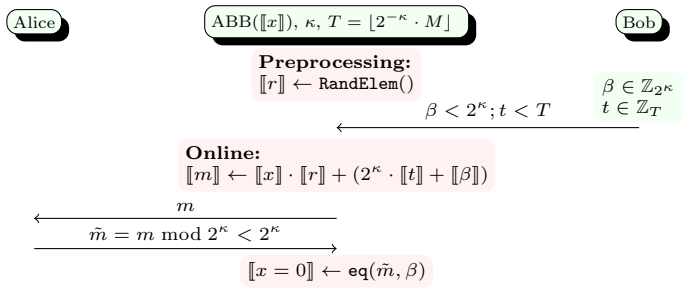
Theorem 1. *Given two ℓ-bit values $[\![x]\!]$ and $[\![y]\!]$ stored in an n-party arithmetic black-box for $\mathbb{Z}_M$ augmented with a proof of boundedness, equality may be computed with 2 outputs and 1 ABB-multiplication in the online phase and $O(\ell)$ operations overall. This is the case both when $\ell = \ell_M$ as well as when $2^{\ell+k+\log n} \ll M$, where k is a statistical security parameter.*

3.2 Equality from DIE

We utilize the DIE protocol in the ABB model to construct a statistically correct zero test (and hence an equality test) in the presence of mutually incorruptible parties, denoted Alice and Bob. Complexity linear in the correctness parameter, κ, i.e., it is only useful when the input is of greater bitlength, say $\ell = 1000$ and $\kappa = 80$. For the sake of concreteness, we describe the case where M is composite.

The idea is to transform $[\![x]\!]$, $x \in \{0,1\}^\ell$, to $[\![y]\!]$, where $y = 0$ when $x = 0$, and y is $(1-\varepsilon)$-close to uniformly random, for an exponentially small ε, when $x \neq 0$. Note that here we use the security parameter κ as the bitlength in the DIE protocol. (See Sect. 2 for the explanation of $\varepsilon = 2^{\kappa-1}/spf(M)$.) The value y is then used to "mask" $t \cdot 2^\kappa + \beta$, i.e., disclose it when $x = 0$ and hide it otherwise. The value revealed to Alice is always statistically close to uniformly random, hence reducing it modulo 2^κ and testing equality with β provides a zero test with a probability of failure of $2^{-\kappa}$. Details are seen as Protocol 2, where `eq` denotes the equality test from Sect. 3.1 but for κ-bit inputs. We focus on the case when M is an RSA-modulus and limit the description to the two-party case. The main benefit of this combined protocol is that by combining it with the equality test above replaces the $O(\ell)$ offline computation/communication with $O(\kappa)$ offline computation/communication. As a drawback, it requires two mutually incorruptible parties and has only has statistical (not perfect) correctness.

Correctness. When $x = 0$, we have $m = t \cdot 2^\kappa + \beta$ and therefore $\tilde{m} = \beta$. Thus, the final equality test correctly determines equality with 0. When $x \neq 0$, $[\![x]\!] \cdot [\![r]\!]$ is $(1-\varepsilon)$-close to uniformly random since $[\![r]\!]$ is generated using `RandElem` and therefore guaranteed to be uniformly random. This implies that m is statistically close to uniformly random, independently of $t \cdot 2^\kappa + \beta$. Thus, m reveals statistically almost no information about β. We remark that the ABB must verify not only



Protocol 2. $\mathtt{eq}_{\mathrm{DIE}}$, secure zero-testing based on disclose-if-equal

$\tilde{m} < 2^\kappa$, but also that $\tilde{m} = m \bmod 2^\kappa$. This can be done by providing not only $\tilde{m} = m \bmod 2^\kappa < 2^\kappa$, but also $\lfloor m/2^\kappa \rfloor < \lfloor M/2^\kappa \rfloor$, and verifying that $m = \lfloor m/2^\kappa \rfloor \cdot 2^\kappa + \tilde{m}$ (e.g., by outputting the difference).

Privacy. A corrupt Bob receives no outputs from the ABB, hence simulation is trivial: do nothing. For a corrupt Alice, note that the only value leaving the ABB is m, hence this is the only possible information leak. Since Bob is honest, $t \cdot 2^\kappa + \beta$ is chosen correctly, thus, no matter the value of x, m will be statistically close to uniformly random – either due to Bob's random choice or the addition of $x \cdot r$. Hence, simulation will consist of a uniformly random element.

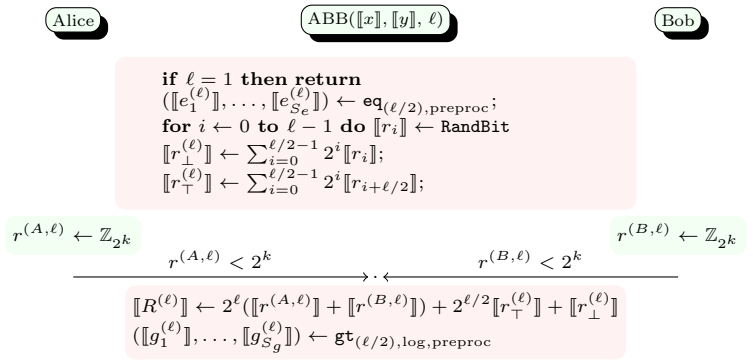
Complexity. The protocol consists of one random element generation, three inputs, and one output plus the invocation of $\mathtt{eq}$. Using $\mathtt{eq}_H$ of the previous subsection, implies $O(\kappa)$ work overall and $O(1)$ (but a slightly worse constant) work online. We state the following theorem:

Theorem 2. *Given two ℓ-bit values $[\![x]\!]$ and $[\![y]\!]$ stored in an n-party arithmetic black-box for $\mathbb{Z}_M$ augmented with a proof of boundedness, equality may be computed with* 3 *outputs and* 2 *ABB-multiplication in the online phase and $O(\kappa)$ operations overall when two mutually incorruptible parties are present.*

4 Greater-Than with Sublinear online Complexity

Toft [16] recently introduced the first sublinear GT protocols, i.e., protocols computing $[\![x \geq y]\!]$ from $[\![x]\!]$ and $[\![\mathrm{y}]\!]$. Utilizing $\mathtt{eq}_H$ and $\mathtt{eq}_{\mathrm{DIE}}$ from Sect. 3, we propose two different (and orthogonal) improvements: (i) We can eliminate the need for two mutually incorruptible parties; this comes at the cost of linear preprocessing, or (ii) We improve efficiency when two mutually incorruptible parties exist by an order of magnitude. Similarly to [16] we assume $2^{\ell+k+\log n} \ll M$, where k is a statistical security parameter and n the number of parties.

The overall idea behind Toft's construction is to perform a GT-test through $\log \ell$ equality tests: If the $\ell/2$ most significant bits of $[\![\mathrm{x}]\!]$ and $[\![\mathrm{y}]\!]$ differ then



Protocol 3. $\mathtt{gt}_{(\ell),\log,\mathrm{preproc}}$: Preprocessing for the secure, ℓ-bit GT test, $\mathtt{gt}_{(\ell),\log}$

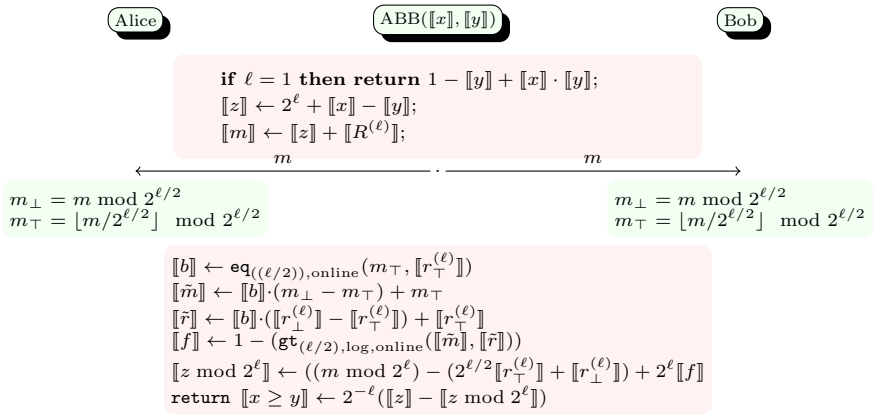
ignore the $\ell/2$ least significant ones; if they are equal then continue with the $\ell/2$ least significant ones. (This description is not correct, but provides sufficient intuition at this point.)

4.1 Sublinear Online Communication in the ABB Model

The main idea of the construction, $\mathtt{gt}_{(\ell),\log} = (\mathtt{gt}_{(\ell),\log,\mathrm{preproc}}, \mathtt{gt}_{(\ell),\log,\mathrm{online}})$, for comparing ℓ-bit numbers is to take the two mutually incorruptible parties of [16] and implement one using the ABB and executing the other "publicly." The core task of the ABB-party is then to generate appropriately distributed random values. Letting $\mathtt{eq}_{(\ell)} = (\mathtt{eq}_{(\ell),\mathrm{preproc}}, \mathtt{eq}_{(\ell),\mathrm{online}})$ denote an equality test for ℓ-bit numbers (and its offline and online phases), preprocessing consists of invoking $\mathtt{eq}_{(2^j),\mathrm{preproc}}$ as well as generating $\log \ell$ random values $[\![R^{(2^j)}]\!] \leftarrow 2^{(2^j)}(\sum_{i=1}^n [\![r^{(i,2^j)}]\!]) + 2^{(2^{j-1})}[\![r_\top^{(2^j)}]\!] + [\![r_\perp^{(2^j)}]\!]$ for $j \in \{1, \ldots, \log \ell\}$, where $[\![r^{(i,2^j)}]\!]$ is a uniformly random k-bit number supplied by P_i and $[\![r_\top^{(2^j)}]\!], [\![r_\perp^{(2^j)}]\!]$ are uniformly random 2^{j-1}-bit values unknown to all. Details are seen as Protocol 3.

The online phase of the construction is seen as Protocol 4 and explained in the correctness argument below. For clarity, we include preprocessed values implicitly in invocations of subprotocols.

Correctness. Correctness is immediate for single bit inputs: $1 - y + x \cdot y$ is 1 exactly when $x \geq y$. For $\ell > 1$, the goal is to transform the comparison of ℓ-bit integers to a comparison of $\ell/2$-bit integers. Observe that $\lfloor z/2^\ell \rfloor$ equals the desired result and that this can be computed as $2^{-\ell}([\![z]\!] - [\![z \bmod 2^\ell]\!])$. Further, since $2^{\ell+k+\log n} \ll M$, we have $z \bmod 2^\ell \equiv m - r \bmod 2^\ell$. We reduce m and $[\![r]\!]$ before the subtraction, which ensures that the result lies between -2^ℓ and 2^ℓ. The correct result is obtained by adding 2^ℓ when this is negative, i.e., when $[\![r \bmod 2^\ell]\!] > m \bmod 2^\ell$. The latter implies $[\![f]\!] = 1$ since we recursively compare the $\ell/2$ most- or least-significant bits of $[\![r \bmod 2^\ell]\!]$ and $m \bmod 2^\ell$ depending on whether the $\ell/2$ most significant bits differed.



Protocol 4. $\mathtt{gt}_{(\ell),\log,\mathrm{online}}$: Online phase of the secure, ℓ-bit GT test, $\mathtt{gt}_{(\ell),\log}$

Privacy. In each recursive call, $m = z + r$ is revealed, but this is statistically indistinguishable from a random value distributed as r – as above r statistically masks z as the bit-length is (at least) k bits longer; for honest ith party P_i, $2^\ell \cdot r^{(i)} + 2^{\ell/2} \cdot r_\top + r_\perp$ is uniformly random.

Complexity. Preprocessing requires $O(\ell)$ work: Though there is a logarithmic number of rounds, each one deals with a problem of half size. Hence, the combined $r_\top$, $r_\perp$ and random masks for $\mathtt{eq}_{H,(\cdot)}$ are only $O(\ell)$ bits overall. Round complexity is $O(1)$, as the iterations can be preprocessed in parallel.

Each online iteration (for $j \in \{1, \ldots, 2^\ell\}$) requires an output ($m$) and an execution of $\mathtt{eq}_{(2^j),\mathrm{online}}$. Additionally, an ABB-multiplication is used to copy the most significant differing halves (if these exist). The remaining computation is purely local or in the form of ABB-additions. Thus, the overall complexity is $O(\log \ell)$ given that $\mathtt{eq}_{(\cdot),\mathrm{online}}$ requires a constant number of ABB operations.

Implementing $\mathtt{eq}_{\cdot}$ as $\mathtt{eq}_{H,(\cdot)}$, the above results in a protocol with $3 \log \ell$ outputs and $2 \log \ell + 1$ ABB-multiplications online; three outputs and two ABB-multiplications per iteration and a single secure ABB-multiplication in the final, single-bit comparison. We state the following theorem:

Theorem 3. *Given two ℓ-bit values $[\![x]\!]$ and $[\![y]\!]$ stored in an n-party arithmetic black-box for $\mathbb{Z}_M$ augmented with a proof of boundedness, greater-than may be computed with $3 \log \ell$ outputs and $2 \log \ell + 1$ ABB-multiplications in the online phase when $2^{\ell+k+\log n} \ll M$, where k is a statistical security parameter.*

We may adapt the constant-rounds protocol of [16] to the present setting. Sketching the solution, let c be a (constant) integer and split $m \bmod 2^\ell$ into $\ell^{1/c}$ strings of $\ell^{1-1/c}$ length. The most significant differing strings may be determined using $O(\ell^{1/c})$ equality tests and arithmetic; these are then compared recursively. Overall this requires c iterations and $O(c \cdot \ell^{1/c})$ equality tests and ABB-multiplications/outputs.

Theorem 4. *Given two ℓ-bit values $[\![x]\!]$ and $[\![y]\!]$ stored in an n-party arithmetic black-box for $\mathbb{Z}_M$ augmented with a proof of boundedness, greater-than may be computed with $O(c \cdot \ell^{1/c})$ ABB operations in $O(c)$ rounds in the online phase when $2^{\ell+k+\log n} \ll M$, where k is a security parameter.*

4.2 Sublinear, DIE-Based Greater-Than

$\mathsf{eq}_{\mathrm{DIE},(\cdot)}$ is much more efficient than the equality test used in [16]. Thus, combining this with Toft's original protocol[4] improves practical efficiency and reduces the theoretical online complexity – $O(\log \ell)$ rounds and work online and[5] $O(\log \ell(\kappa + \mathrm{loglog}\, \ell))$ ABB-operations overall. The constant-rounds protocol may also be combined with $\mathsf{eq}_{\mathrm{DIE}}$ resulting in an $O(c)$ rounds protocol with $O(c \cdot \ell^{1/c})$ work online and $O(\ell^{1/c}(\kappa + \log \ell))$ work overall. We state the following theorems:

Theorem 5. *Given two ℓ-bit values $[\![x]\!]$ and $[\![y]\!]$ stored in an n-party arithmetic black-box for $\mathbb{Z}_M$ augmented with a proof of boundedness, GT may be computed in the presence of two mutually incorruptible parties with $4 \log \ell$ outputs and $3 \log \ell + 1$ ABB-multiplications in the online phase and $O(\log \ell(\kappa + \mathrm{loglog}\, \ell))$ operations overall when $2^{\ell+k+\log n} \ll M$, where k is a statistical security parameter.*

Theorem 6. *Given two ℓ-bit values $[\![x]\!]$ and $[\![y]\!]$ stored in an n-party arithmetic black-box for $\mathbb{Z}_M$ augmented with a proof of boundedness, greater-than may be computed in the presence of two mutually incorruptible parties with $O(c \cdot \ell^{1/c})$ ABB-operations in $O(c)$ rounds in the online phase and $O(\ell^{1/c}(\kappa + \log \ell))$ operations overall when $2^{\ell+k+\log n} \ll M$, where k is a statistical security parameter.*

Acknowledgements. The first author was supported by the Estonian Research Council, and European Union through the European Regional Development Fund. The second author was supported by COBE financed by "The Danish Agency for Science, technology and Innovation." Additional support from the Danish National Research Foundation and The National Science Foundation of China (under the grant 61061130540) for the Sino-Danish Center for the Theory of Interactive Computation.

References

1. Aiello, W., Ishai, Y., Reingold, O.: Priced Oblivious Transfer: How to Sell Digital Goods. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 119–135. Springer, Heidelberg (2001)
2. Bar-Ilan, J., Beaver, D.: Non-Cryptographic Fault-Tolerant Computing in a Constant Number of Rounds of Interaction. In: Rudnicki, P. (ed.) PODC 1989, pp. 201–209. ACM Press (1989)

[4] The key difference from Protocol 4 is that Bob selects r, while only Alice learns m.
[5] We add $\mathrm{loglog}\, \ell$ to κ to compensate for a non-constant number of equality tests.

3. Ben-Or, M., Goldwasser, S., Wigderson, A.: Completeness Theorems for Non-Cryptographic Fault-Tolerant Distributed Computation. In: STOC 1988, pp. 1–10. ACM Press (1988)
4. Bogetoft, P., Christensen, D.L., Damgård, I., Geisler, M., Jakobsen, T., Krøigaard, M., Nielsen, J.D., Nielsen, J.B., Nielsen, K., Pagter, J., Schwartzbach, M., Toft, T.: Secure Multiparty Computation Goes Live. In: Dingledine, R., Golle, P. (eds.) FC 2009. LNCS, vol. 5628, pp. 325–343. Springer, Heidelberg (2009)
5. Chaabouni, R., Lipmaa, H., Zhang, B.: A Non-interactive Range Proof with Constant Communication. In: Keromytis, A.D. (ed.) FC 2012. LNCS, vol. 7397, pp. 179–199. Springer, Heidelberg (2012)
6. Damgård, I.B., Fitzi, M., Kiltz, E., Nielsen, J.B., Toft, T.: Unconditionally Secure Constant-Rounds Multi-party Computation for Equality, Comparison, Bits and Exponentiation. In: Halevi, S., Rabin, T. (eds.) TCC 2006. LNCS, vol. 3876, pp. 285–304. Springer, Heidelberg (2006)
7. Damgård, I.B., Nielsen, J.B.: Universally Composable Efficient Multiparty Computation from Threshold Homomorphic Encryption. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 247–264. Springer, Heidelberg (2003)
8. Groth, J.: Short Pairing-Based Non-interactive Zero-Knowledge Arguments. In: Abe, M. (ed.) ASIACRYPT 2010. LNCS, vol. 6477, pp. 321–340. Springer, Heidelberg (2010)
9. Laur, S., Lipmaa, H.: A New Protocol for Conditional Disclosure of Secrets and Its Applications. In: Katz, J., Yung, M. (eds.) ACNS 2007. LNCS, vol. 4521, pp. 207–225. Springer, Heidelberg (2007)
10. Lipmaa, H.: On Diophantine Complexity and Statistical Zero-Knowledge Arguments. In: Laih, C.-S. (ed.) ASIACRYPT 2003. LNCS, vol. 2894, pp. 398–415. Springer, Heidelberg (2003)
11. Lipmaa, H.: Progression-Free Sets and Sublinear Pairing-Based Non-Interactive Zero-Knowledge Arguments. In: Cramer, R. (ed.) TCC 2012. LNCS, vol. 7194, pp. 169–189. Springer, Heidelberg (2012)
12. Naor, M., Pinkas, B.: Oblivious Transfer and Polynomial Evaluation. In: STOC 1999, pp. 245–254. ACM Press (1999)
13. Nishide, T., Ohta, K.: Multiparty Computation for Interval, Equality, and Comparison Without Bit-Decomposition Protocol. In: Okamoto, T., Wang, X. (eds.) PKC 2007. LNCS, vol. 4450, pp. 343–360. Springer, Heidelberg (2007)
14. Paillier, P.: Public-Key Cryptosystems Based on Composite Degree Residuosity Classes. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 223–238. Springer, Heidelberg (1999)
15. Thorbek, R.: Linear Integer Secret Sharing. Ph.D. thesis, Aarhus University (2009)
16. Toft, T.: Sub-linear, Secure Comparison with Two Non-colluding Parties. In: Catalano, D., Fazio, N., Gennaro, R., Nicolosi, A. (eds.) PKC 2011. LNCS, vol. 6571, pp. 174–191. Springer, Heidelberg (2011)
17. Toft, T.: Primitives and Applications for Multiparty Computation. Ph.D. thesis, Aarhus University (2007)
18. Yu, C.H.: Sign Modules in Secure Arithmetic Circuits. Tech. Rep. 2011/539, IACR (October 1, 2011), `http://eprint.iacr.org/2011/539` (checked in February 2013)