Improved Secure Integer Comparison via Homomorphic Encryption

Florian Bourse[1], Olivier Sanders[1](✉), and Jacques Traoré[2]

[1] Orange Labs, Applied Crypto Group, Cesson-Sévigné, France
olivier.sanders@orange.com

[2] Orange Labs, Applied Crypto Group, Caen, France

Abstract. Secure integer comparison has been one of the first problems introduced in cryptography, both for its simplicity to describe and for its applications. The first formulation of the problem was to enable two parties to compare their inputs without revealing the exact value of those inputs, also called the Millionaires' problem [45]. The recent rise of fully homomorphic encryption has given a new formulation to this problem. In this new setting, one party blindly computes an encryption of the boolean $(a < b)$ given only ciphertexts encrypting a and b.

In this paper, we present new solutions for the problem of secure integer comparison in both of these settings. The underlying idea for both schemes is to avoid decomposing the integers in binary in order to improve the performances. On the one hand, our fully homomorphic based solution is inspired by [9], and makes use of the fast bootstrapping techniques developed by [12,14,23] to obtain scalability for large integers while preserving high efficiency. On the other hand, our solution to the original Millionaires' problem is inspired by the protocol of [10], based on partially homomorphic encryption. We tweak their protocol in order to minimize the number of interactions required, while preserving the advantage of comparing non-binary integers.

Both our techniques provide efficient solutions to the problem of secure integer comparison for large (even a-priori unbounded in our first scenario) integers with minimum interactions.

1 Introduction

Evaluation of algorithms over encrypted data is a major topic in cryptography which has known very important results over the past decade (*e.g.* [27]). Generic solutions supporting any operation exist but they usually require to represent the algorithm as a boolean circuit and incur very large complexity. Conversely, solutions specifically designed for a particular algorithm are more efficient, but require a large amount of work that must be started over each time the algorithm is updated.

In this context, an interesting middle-way is the one consisting in designing efficient protocols for simple tasks (but still more complex than basic operations) that are frequently used as subroutines by other algorithms. Indeed, in this case,

S. Jarecki (Ed.): CT-RSA 2020, LNCS 12006, pp. 391–416, 2020.
https://doi.org/10.1007/978-3-030-40186-3_17

the resulting protocol will be more efficient than the one generated by applying generic solutions and, at the same time, the widespread use of this subroutine will ensure that the efforts invested in the design of this protocol will benefit to a very large number of algorithms.

Perhaps the most prominent example of this approach (for both historical and practical reasons) is the one of secure integer comparison, where two parties knowing respectively secret integers m_1 and m_2 want to compare them without leaking any information beyond the result $(m_1 \leq m_2)$.

Introduced in 1982 by Yao [45] who presented it as the problem encountered by two millionaires wanting to secretly compare their respective wealth (hence its name of *Millionaires' problem*), this problem is of utter importance in many areas, especially since the rise of machine learning. Indeed, several classifiers require to sort (and therefore to compare) elements and thus need appropriate protocols when the latter are encrypted, as illustrated in [8]. More generally, the fact that most algorithms run integers comparison as subroutines emphasizes the need for counterparts handling encrypted data.

In his seminal paper [45], Yao proposed a first protocol for secure comparison based on garbled circuits, a by-now standard tool in cryptography which has become a subject on its own. However, this kind of techniques, despite several improvements (*e.g.* [5,6,16,32]), implies rather important communication complexity, which can be problematic in contexts where communications are slow.

In [2,24], the authors follow a different strategy, based on the Legendre symbol, which leads to very elegant protocols. Unfortunately, the latter can only handle integers of limited size, and it does not seem possible to extend them to support large inputs.

Another approach for secure comparison is the one based on homomorphic encryption, starting from Fischlin's work [25]. The ability to perform operations on encrypted data can remove some interactions but at the cost of greater computational complexity. Here again, several improvements followed [7,20,21,26,30,34,42] but they involve bitwise encryption of the integers, leading to a complexity of at least $\log_2(M)$ operations where M is a bound on the integers to compare.

Comparing the solutions based on garbled circuits with the ones based on homomorphic encryption is not always relevant as they are very different constructions. Garbled circuits mostly rely on symmetric primitives and thus usually offer good performance. Homomorphic encryption is a more complex tool but seems to be a promising solution to go beyond the $\log_2(M)$ barrier. Indeed, two homomorphic-based constructions [9,10] overcoming this limation have recently been introduced for different settings.

The first one (CEK), proposed by Carlton, Essex and Kapulkin [10], corresponds exactly to the Millionaires' problem scenario where two parties want to compare their respective secret values m_1 and m_2. It is based on an homomorphic threshold encryption system allowing to directly compare small integers, leading to less computations, but at the cost of more interactions compared to the DGK protocol [20]. Indeed, in their protocol, the party A knowing the decryption keys received either an encryption of 0 (if $m_1 \geq m_2$) or of some value related to m_1 and m_2 (if $m_1 < m_2$). This forces the other party B to blind the plaintext with some random value s leading to the following problem: in any case A decrypts randomness. To bypass this problem, both parties run a plaintext equality test (PET) at the end of the protocol to decide whether the randomness is s (in which case $m_1 \geq m_2$) or not. This PET implies at least one additional pass and the use of another homomorphic encryption scheme. In some way, the result of Carlton *et al.* can thus be seen as a new tradeoff between computation and communication complexity.

In the second setting, one party is given two ciphertexts for values m_1 and m_2 and has to produce a ciphertext for the boolean $(m_1 > m_2)$, whereas the other party is the only one having the secret key that allows decryption of these ciphertexts. One way to solve this problem has been to reduce it to the Millionaires' problem, as done in [41,42]. More interestingly, this problem can even be solved non-interactively, by using fully homomorphic encryption. However, the current state-of-the-art in FHE doesn't provide a fully satisfactory solution to the homomorphic evaluation of the comparison. The two main techniques are either based on somewhat homomorphic encryption, which is not suitable because the comparison cannot a priori be represented by a low degree polynomial, so the noise growth would be unmanageable, or have to deal with the bit decomposition of the messages (*e.g.* [11,19]). In [44], the authors proposed a solution based on Wilson's theorem to avoid binary decomposition. However, it requires to perform $(2M)^2$ homomorphic multiplications to compare integers smaller than M, which rapidly becomes prohibitive as M increases.

At Crypto 18, Bourse *et al.* [9] proposed a modified FHE system enabling to efficiently evaluate the sign function. This can be used to compare two encrypted values by subtracting and evaluating the sign. However, this scheme only supports a bounded message space, and the sizes of the bootstrapping key and ciphertexts grow exponentially in the size of the messages (or superlinear in the value of bound on the messages). This result is enough to work on very small sized input, or on computation that are inherently fault-tolerant, as they show with neural network evaluations, but is hardly usable in practice for less specific applications. Moreover, this requires the bound on the messages to be chosen at setup time, because the parameters of the scheme depend on it.

1.1 Our Contribution

In this work we propose two protocols that respectively improve [9] and [10] and thus the state-of-the-art of secure integer comparison.

In a first part, we describe a new FHE-based solution in the setting where B wants to blindly compare two encrypted integers. Starting from [9], we show how (1) to modify it in order to output 0 whenever the two inputs are equal, and (2) to scale the output by any chosen factor. The first part requires a careful modification of the `testVector` from [9] because the function to be computed must verify some anticyclic properties, and ternary sign doesn't satisfy those.

Hence, we had to add a slot never used into the message space. The second part might seem trivial for FHE because scaling can be performed by multiplying the output ciphertext by the chosen factor. However, this would yield too much noise. We then here again need to modify the `testVector` to take into account the scaling factor before returning the output ciphertext.

Then, relying on those two properties, we construct recursively an algorithm to compare unbounded integers, by decomposing them in some basis that can be handled by our modified scheme for bounded integers. The resulting scheme combines the generality of bitwise encryption comparisons, because we can compare unbounded integers using a fixed bootstrapping key that can be generated without knowledge of the integers to compare, together with the improved efficiency, both in computation time and in ciphertext expansion factor, of the schemes that support non-binary message spaces.

In a second part, we propose a new protocol to address the Millionaires' problem that combines (almost) all the best features of the DGK and CEK protocols. Starting from the latter, we introduce several modifications to avoid the costly PET that constitutes the last step of CEK. More specifically, we manage to replace the whole PET by a simple hash value sent by the party B in the second pass. This digest will indeed be enough for A to decide whether the decrypted plaintext is the blind factor or not. However, this idea cannot be directly applied to the CEK protocol because a simple exhaustive search on the message space enables A to recover B's value whenever $m_2 > m_1$. We therefore consider different RSA parameters to introduce new random elements in the protocol to thwart (with overwhelming probability) exhaustive searches. The point is that all these modifications do not significantly hamper the main feature of CEK, namely the ability to compare several bits at once, which means that our protocol still compares favourably to DGK (and its predecessors).

Concretely, compared to DGK, our protocol also requires two passes but divides both the computation and the communication cost by a factor up to 4. Compared to CEK, the speedup factor is harder to assess because it heavily depends on the security parameter (see Sect. 4.2) but we manage anyway to divide by up to two the number of passes. This comes at the cost of a security proof in the random oracle model (ROM) but this model is widely used in cryptography, especially to design practical constructions.

We stress that these improvements must not be measured just for one run of our protocols but must rather be put in perspective with the massive use of comparisons in algorithms. For example, the classifiers considered in [8] require to find the greater value of a list $a_1, \ldots, a_k$ of encrypted elements and so to run k secure comparison protocols. In such a case, the impact of our protocols will be multiplied by at least k.

1.2 Organization

Our paper addresses two different versions of the Millionaires' problem and is thus divided in two parts that can be read independently. In Sect. 2, we describe a solution based on fully homomorphic encryption that outputs an encrypted boolean $(m_1 > m_2)$ given two ciphertexts encrypting respectively m_1 and m_2. In Sect. 3, we consider the original scenario of the Millionaires' problem and provide a solution that enables two parties to secretly compare their respective entries.

2 Homomorphic Comparison of Integers

In this section, we build a new technique to homomorphically compare two integers. We first start by recalling all necessaries preliminaries about lattice-based cryptography and fully homomorphic encryption that we will use. Then, we start our construction by extending the work of [9] to allow ternary sign computation, and add as an input a scaling factor that will multiply the output. Finally, we show how to compare two unbounded integers by calling recursively our comparison procedure for small integers.

2.1 Preliminaries

As in [14], we present the learning with errors problem and assumptions using the torus $\mathbb{T} = \mathbb{R}/\mathbb{Z}$ (i.e., the real modulo 1), and binary vectors as the secret keys. The same results hold for the formulation over $\mathbb{Z}_q$ for any q instead of $\mathbb{T}$. However, to the best of our knowledge, binary secret keys are required for the techniques allowing a fast bootstrapping.

Learning With Errors (LWE). This problem was introduced by Regev [38] as a candidate problem that is hard to solve, even for quantum computers. Let n be a positive integer, and χ a probability distribution over $\mathbb{R}$. For any vector[1] $\mathbf{s} \in \{0,1\}^n$, we define the LWE distribution $\mathsf{LWE}_{n,\mathbf{s},\chi}$ as $(\mathbf{a}, b)$, where $\mathbf{a} \overset{\$}{\leftarrow} \mathbb{T}^n$, $e \overset{\$}{\leftarrow} \chi$, and $b = \langle \mathbf{s}, \mathbf{a} \rangle + e$.

The LWE assumption states that for $\mathbf{s} \overset{\$}{\leftarrow} \{0,1\}^n$, it is hard to distinguish between $\mathsf{LWE}_{n,\mathbf{s},\chi}$ and the uniform distribution over $\mathbb{T}^n$.

Ring Learning With Errors (RLWE). We also extend the ring variant [35] of LWE to the special case where $\mathcal{R}_N = \mathbb{R}_N[X]/\mathbb{Z}_N[X]$, with $\mathbb{R}_N[X] = \mathbb{R}[X]/(X^N+1)$ (respectively, $\mathbb{Z}_N[X] = \mathbb{Z}[X]/(X^N+1)$), i.e., the $\mathbb{Z}_N[X]$ module of polynomials of degree up to $N-1$ with coefficients in $\mathbb{T}$, with the operations done modulo X^N+1 and modulo 1. Let N be a power of two, and χ be a distribution over $\mathbb{R}_N[X]$ for the noise. For any polynomial s of degree up to $N-1$ with binary coefficients, we define the RLWE distribution $\mathsf{RLWE}_{N,s,\chi}$ as (a, b), where $a \overset{\$}{\leftarrow} \mathcal{R}_N$, $e \overset{\$}{\leftarrow} \chi$, and $b = s \cdot a + e$.

The RLWE assumption states that for a uniformly random polynomial s of degree up to $N-1$ with binary coefficients, it is hard to distinguish between $\mathsf{RLWE}_{N,s,\chi}$ and the uniform distribution over $\mathcal{R}_N^2$.

[1] This is not exactly the original LWE definition since we here consider binary coefficients for the secret key, as in [9,12,13]. Nevertheless, we will still refer to it as LWE for sake of simplicity.

LWE Encryption Scheme. As in [9] and in some previous works [1,3,31,37], we use a variant of Regev's secret key encryption scheme which supports a non-binary message space. It can easily be transformed into a public key encryption scheme using standard techniques. Let B be an integer. The message space will be $\{-B+1,\dots,B-1\}$. We define the encryption scheme as follows:

Setup(1^λ)**:** on input a security parameter λ, fix $n = n(\lambda)$, samples and returns $\mathbf{s} \overset{\$}{\leftarrow} \{0,1\}^n$;

Encrypt$(\mathbf{s}, m)$**:** on input secret key $\mathbf{s}$ and message m, samples $(\mathbf{a}, b) \overset{\$}{\leftarrow} \mathsf{LWE}_{n,\mathbf{s},\chi}$, and returns $ct = (\mathbf{ct}_0, ct_1) = (\mathbf{a}, b + \frac{m}{2B})$;

Decrypt$(\mathbf{s}, ct)$**:** computes x the representative of $ct_1 - \langle \mathbf{ct}_0, \mathbf{s}\rangle \bmod 1$ in $[-\frac{1}{2}, \frac{1}{2}[$, and returns $\lfloor 2B \cdot x \rceil$.

We note that using $2B-1$ as denominator would be enough to support the message space $\{-B+1,\dots,B-1\}$. However, we require one extra unused slot in the message space for technical reasons during the sign computation.

Some of our protocols involve LWE encryption schemes of different dimensions. In such a case, we will refer to some ciphertexts as n-LWE ciphertexts, where n is the dimension, to avoid confusion with the other ciphertexts.

This encryption scheme generalizes to RLWE in a straightforward manner.

Bootstrapping Procedure. Our construction relies on three functions **BlindRotate**, **Extract** and **KeySwitch** that are defined in [9,12]. A proper definition of these functions requires to introduce many technical details along with the ring variant of the GSW encryption scheme [28]. However, such a definition is not necessary for the understanding of our work. For sake of clarity, we then only provide an informal definition that is sufficient for our paper.

BlindRotate: on input an LWE encryption ct encrypted with key $\mathbf{s}$, and a bootstrapping key bk, returns an RLWE encryption of $X^{\bar{b} - \langle \bar{\mathbf{a}}, \mathbf{s}\rangle}$, where $\bar{b} = \lfloor 2N \cdot ct_1 \rceil$ and $\bar{\mathbf{a}} = \lfloor 2N \cdot \mathbf{ct}_0 \rceil$;

Extract: on input an RLWE encryption of a polynomial $p(X)$, returns an LWE encryption of $p(0)$;

KeySwitch: on input an LWE encryption c of m under a certain key $\mathbf{s}$ and a keyswitching key ksk (which consists of LWE encryptions of the bits s_i of $\mathbf{s}$ under secret key $\mathbf{s}'$), returns an LWE encryption of m under secret key $\mathbf{s}'$.

The key switching algorithm is not required for the construction to work, but it brings a lot of improvement in efficiency by reducing the dimension of the LWE ciphertext.

2.2 Strategy Overview

Before presenting our construction in more details, we give a high level overview of the underlying idea.

Let us assume that we are given an algorithm to compute the sign of integers in $\{-B+1,\dots,B-1\}$. It can be used to compare two numbers x and y in $[0, B-1]$ as follows:

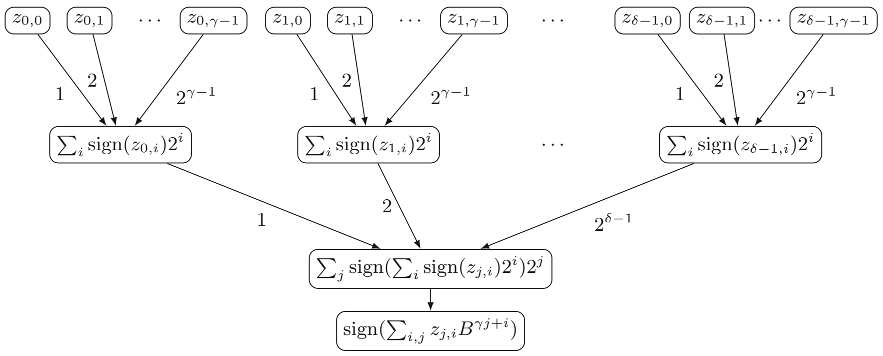
$z_{0,0}$
$z_{0,1}$
$z_{0,\gamma-1}$
$z_{1,0}$
$z_{1,1}$
$z_{1,\gamma-1}$
$z_{\delta-1,0}$
$z_{\delta-1,1}$
$z_{\delta-1,\gamma-1}$
1
2
$2^{\gamma-1}$
$\sum_i \mathrm{sign}(z_{0,i})2^i$
$\sum_i \mathrm{sign}(z_{1,i})2^i$
$\sum_i \mathrm{sign}(z_{\delta-1,i})2^i$
$2^{\delta-1}$
$\sum_j \mathrm{sign}(\sum_i \mathrm{sign}(z_{j,i})2^i)2^j$
$\mathrm{sign}(\sum_{i,j} z_{j,i}B^{\gamma j+i})$

Fig. 1. Strategy to compare unbounded integers x and y given a procedure to compute the sign of integers in $\{-B+1,\ldots,B-1\}$. Here, $z_{i,j} = x_{i,j} - y_{i,j}$ are the differences of the digits of x and y in base B and $\delta = \lceil k/\gamma \rceil$. Arrows indicate computation of the ternary sign, scaled by the factor labelling it, and nodes consist of the sums of their incoming arrows.

1. take the difference $z = x - y$;
2. compute the sign of z.

If z is positive, it means x was greater than y, and vice versa.

Now let us say we are given bigger integers x and y such that we cannot use our sign function directly. What we can do is to decompose x and y in basis B, in order to obtain numbers in $[0, B-1]$. Let $(x_i)_{i\in[0,k]}$ and $(y_i)_{i\in[0,k]}$ be the digits of x and y in base B, for some integer k. For each i in $[0,k]$, we can compute the sign of $z_i = x_i - y_i$. However, we need a trick to combine those results to obtain the comparison of x and y, which is the sign of $z = \sum_{i\in[0,k]} z_i B^i$.

In order to continue, our main observation is that the sign of z is the same as the sign of $\sum_{i\in[0,k]} \mathrm{sign}(z_i)2^i$. Thus, we can pack the values z_i by groups of $\gamma = \lfloor \log_2(B) \rfloor$ values, scale each of them by a factor 2^i, depending on their position, and carry on computing the signs in a tree-like fashion as illustrated on Fig. 1.

Intuitively, the sign of each node will be the same as the sign of the rightmost non-zero node pointing to it, assuming the digits are ordered from the least significant on the left to the most significant on the right. Hence, by induction, the final value will be the sign of the rightmost non-zero z_i, i.e., -1 if $x < y$, 0 if $x = y$, and 1 if $x > y$.[2]

This construction requires two new features that are not present in [9]:

– it requires the sign to be ternary, *i.e.* $\mathrm{sign}(0) = 0$;
– the output has to be scaled by a factor 2^i given as input.

[2] The binary sign can be obtained by applying the techniques of [9] instead of our ternary sign in the last step.

$$\begin{tikzcd}
\boxed{z_{k-1}} \arrow[dr, "B/2"] & \boxed{z_{k-2}} \arrow[d] & \boxed{z_{k-3}} \arrow[d] & \cdots & \boxed{z_0} \arrow[d] & \\
& \boxed{\frac{B}{2}\mathfrak{s}_0 + z_{k-2}} \arrow[r, "B/2"] & \boxed{\frac{B}{2}\mathfrak{s}_1 + z_{k-3}} \arrow[r, "B/2"] & \cdots \arrow[r, "B/2"] & \boxed{\frac{B}{2}\mathfrak{s}_{k-2} + z_0} \arrow[r] & \boxed{\mathfrak{s}_{k-1}}
\end{tikzcd}$$

Fig. 2. Alternative strategy to compare unbounded integers x and y given a procedure to compute the sign of integers in $\{-B+1, \ldots, B-1\}$. Here, $z_i = x_i - y_i$ are the differences of the digits of x and y in base $\frac{B}{2}$ and $k = \lceil \log_{\frac{B}{2}}(x) \rceil$. The element $\mathfrak{s}_i$ denotes the ternary sign of $\sum_{j \in \{0,\ldots,i\}} z_{k-1-j} \left(\frac{B}{2}\right)^{k-1-j}$, the $(i+1)$ most significant digits of z. Horizontal arrows indicate computation of the ternary sign, scaled by the factor labelling it, and nodes consist of the sums of their incoming arrows.

The former is required in order to propagate the comparison of least significant digits whenever the most significant digits are equal. The latter cannot be accomplished by scaling the output ciphertext, because this would yield too much noise, thus preventing correct decryption of the resulting ciphertext. We will then explain how to take this scaling factor into account before returning the ciphertext, which leads to better noise management.

We also suggest another way to bootstrap such a technique to unbounded integers that has better noise management, at the cost of slightly larger ciphertexts, and sequential computations. We give an illustration of this on Fig. 2.

The idea is now to only have one addition in order to minimize the noise growth and optimize the parameters. We thus decompose the integers in basis $\frac{B}{2}$ and start from the most significant digits. At each step, we compute the ternary sign, and scale it by $\frac{B}{2}$ before adding it to the next digits. That way, the sign of our accumulator is always the sign of the difference, up to that point.

In the following, we first build an algorithm to compute the sign with the two additional features required, and then we present a recursive algorithm for each of our strategies.

Remark 1. At first sight, a simpler solution to compare x and y could be to select a bound B greater than these two integers, to generate keys compatible with this message space and then to directly run the protocol from [9] on $z = x - y$. However, there are two problems with this solution. First, the complexity of [9] is exponential in the size of the messages so selecting large B is not a good strategy (see Sect. 2.5 for more details). Second, this solution leads to the following dilemma. Either we select a bound B large enough to handle any integers x and y or we select, for each comparison, the smallest possible value for B. The former option makes the previous complexity issue even worse. The second option makes key management quite cumbersome because it implies several keys, one for each possible range of values of x and y.

Algorithm 1. $\mathsf{HomomorphicCompare}_{B,0}$

```
    **Input:** two LWE ciphertexts $c_1, c_2$ encrypting messages $m_1, m_2 \in [0, B-1]$, a bootstrapping key $bk$ and a scaling factor $k$
    **Output:** an $N$-LWE encryption of $k \cdot \mathrm{sign}(m_1 - m_2)$
 1: $P(X) := -k \cdot \sum_{i \in \{\lceil \frac{N}{2B} \rceil, \dots, \lfloor N - \frac{N}{2B} \rfloor\}} X^i$
 2: $c := c_1 - c_2$ (the task is now to find the sign of the plaintext in $c$)
 3: $\tilde{c} := \mathrm{BlindRotate}(c, bk)$
 4: $x := \mathrm{Extract}\left(P(X) \cdot \tilde{c}\right)$
 5: **return** $x$
```

2.3 Homomorphic Comparison of Small Integers

In this subsection, we show how to compare small integers, which will be the base case for our induction. While the techniques from [9] could be used directly to compare small integers, they do not fit our strategies for larger integers. We therefore modify their scheme in order to output 0 whenever the plaintexts are equal. This will be required in order to compare unbounded integers using this simple construction as a building block.

Our homomorphic comparison for small values $\mathsf{HomomorphicCompare}_{B,0}$ is defined in Algorithm 1. For simplicity, we chose to define it without keyswitching to reduce the number of parameters, but it can easily be introduced as an optimization before returning the result. The scaling factor for the output is not important for the comparison of small integers, but will be needed to efficiently compute the comparison of larger integers in the next section. Correctness of this protocol is proved below.

Correctness. If c_1 encrypted m_1 and c_2 encrypted m_2, x encrypts k if $m_1 > m_2$, $-k$ if $m_1 < m_2$, and 0 if $m_1 = m_2$ with overwhelming probability, for well chosen parameters. Indeed, c encrypts $m_1 - m_2$, $\tilde{c}$ encrypts $X^{\bar{m}_1 - \bar{m}_2 + e}$, where $\bar{m}_i = m_i \cdot \frac{N}{B}$, and e is the error resulting from c_1, c_2, and the scalings and roundings during BlindRotate. Then, $P(X) \cdot \tilde{c}$ encrypts $-k \cdot \sum_{i \in \{\lceil \frac{N}{2B} \rceil, \dots, \lfloor N - \frac{N}{2B} \rfloor\}} X^i \cdot X^{\bar{m}_1 - \bar{m}_2 + e}$, the constant term of which is

$$\begin{aligned} &k \text{ if } \lceil \frac{N}{2B} \rceil \le \bar{m}_1 - \bar{m}_2 + e \le \lfloor N - \frac{N}{2B} \rfloor \\ -&k \text{ if } -\lfloor N - \frac{N}{2B} \rfloor \le \bar{m}_1 - \bar{m}_2 + e \le -\lceil \frac{N}{2B} \rceil \\ &0 \text{ otherwise} \end{aligned}$$

Now let us assume that $m_1 > m_2$ and that the parameters are chosen such that $|e| < \frac{N}{2B}$, we have:

$$\begin{aligned} &1 \le m_1 - m_2 \le B - 1 \\ \Leftrightarrow &\frac{N}{B} \le \frac{N(m_1 - m_2)}{B} \le \frac{N \cdot (B-1)}{B} \end{aligned}$$

$$\Leftrightarrow \frac{N}{B} + e \leq \bar{m}_1 - \bar{m}_2 + e \leq \frac{N \cdot (B-1)}{B} + e$$
$$\Rightarrow \lceil \frac{N}{2B} \rceil \leq \bar{m}_1 - \bar{m}_2 + e < \frac{N \cdot (B-1)}{B} + \frac{N}{2B}$$

where the first inequality comes from the fact that $|e| < \frac{N}{2B}$ and that $\bar{m}_1 - \bar{m}_2 + e$ is an integer. Now, if we write:

$$\frac{N \cdot (B-1)}{B} + \frac{N}{2B} = N - \frac{N}{2B}$$

we get

$$\lceil \frac{N}{2B} \rceil \leq \bar{m}_1 - \bar{m}_2 + e \leq \lfloor N - \frac{N}{2B} \rfloor$$

which ensures that x encrypts k if $m_1 - m_2 \geq 1$.
Conversely, if $m_1 < m_2$, we get

$$-\frac{N}{B} + e \geq \bar{m}_1 - \bar{m}_2 + e \geq -\frac{N \cdot (B-1)}{B} + e$$

which implies that

$$-\lceil \frac{N}{2B} \rceil \geq \bar{m}_1 - \bar{m}_2 + e \geq -\lfloor N - \frac{N}{2B} \rfloor$$

and so that x encrypts $-k$.

2.4 Homomorphic Comparison of Unbounded Integers

In Sect. 2.2, we have described two strategies for comparing unbounded integers. The first one will be referred to as *tree-based*, whereas the latter one will be referred to as *sequential*. Informally, the tree-based approach is suitable for parallel computing whereas the sequential one offers better parameters but requires sequential computations (hence its name).

Tree-Based Strategy. Let us denote $\gamma = \lfloor \log_2(B) \rfloor$. Assuming that $B \geq 4$ (i.e., $\gamma \geq 2$), we can define a family of algorithms that can homomorphically compute the comparison of unbounded integers (that is, for any size of messages, there exists an algorithm in the family that can handle it) by decomposing the number in basis B, and do the comparison recursively in a bottom-up tree fashion, where each node has up to γ children. For each of them, we use the small integer homomorphic comparison with scaling factor 2^i, with i the position of the child, starting from 0 for the least significant. By adding the resulting values and then running again the small integer comparison protocol, we get the sign of the most significant non-zero child, as illustrated in Fig. 1.

Algorithm 2. $\mathsf{HomomorphicCompare}_{B,\ell+1}$ for message space $[0, B^{\gamma^{\ell+1}} - 1]$

```
Input: two ciphertexts $c_1, c_2$ encrypting messages $m_1, m_2 \in [0, B^{\gamma^{\ell+1}} - 1]$, a bootstrapping key $bk$ and a scaling factor $k$
Output: an $N$-LWE encryption of $k \cdot \text{sign}(m_1 - m_2)$
1: for $i = 0 \dots \gamma - 1$ do
2:     $\overline{c_i} := \mathsf{HomomorphicCompare}_{B,\ell}(c_{1,i}, c_{2,i}, bk, 2^i)$
3: end for
4: $P(X) := -k \cdot \sum_{i \in \{\lceil \frac{N}{2B} \rceil, \dots, \lfloor N - \frac{N}{2B} \rfloor\}} X^i$
5: $c := \sum_{i \in [0,\gamma-1]} \overline{c_i}$ (the task is now to find the sign of the plaintext in $c$)
6: $\tilde{c} := \text{BlindRotate}(c, bk)$
7: $x := \text{Extract}\,(P(X) \cdot \tilde{c})$
8: return $x$
```

Before defining our algorithm to compare larger integers homomorphically, we need to specify how we encrypt those. Our encryption scheme is also defined by induction:

Enc$_B$**:** we use LWE.Encrypt as defined in Sect. 2.1;
Enc$_{B^{\gamma^{\ell+1}}}$**:** on input $m = \sum_{i \in [0,\gamma-1]} m_i (B^{\gamma^\ell})^i$, returns $(Enc_{B^{\gamma^\ell}}(m_i))_{i \in [0,\gamma-1]}$.

A ciphertext c encrypting a message $m \in [0, B^{\gamma^{\ell+1}}]$ thus contains γ ciphertexts c_i encrypting messages $m_i \in [0, B^{\gamma^\ell}]$. We are now ready to describe our family of algorithms to homomorphically compare large integers in Algorithm 2. By induction, this defines algorithms for homomorphic comparison with message spaces $[0, B^{\gamma^\ell} - 1]$, for any positive integer ℓ.

Correctness. By induction hypothesis, c_i encrypts $2^i \cdot \text{sign}(m_{1,i} - m_{2,i})$. Then c encrypts $\sum_{i \in [0,\gamma-1]} 2^i \cdot \text{sign}(m_{1,i} - m_{2,i})$, the sign of which is the sign of the last non-zero $m_{1,i} - m_{2,i}$, which is the sign of $m_1 - m_2$. Then, assuming the error does not grow too much, we can use the same analysis as previously to conclude that we correctly evaluate to $k \cdot \text{sign}(m_1 - m_2)$. The noise now comes from the sum of γ ciphertexts instead of 2.

Sequential Strategy. In order to minimize the noise growth during the computation, we apply the technique described in Sect. 2.2. First, we encrypt the messages by decomposing them in basis $\frac{B}{2}$ as follows:

$\overline{\textbf{Enc}_{(\frac{B}{2})^\ell}}$**:** on input $m = \sum_{i \in [0,\ell-1]} m_i (\frac{B}{2})^i$, returns $(\text{LWE.Encrypt}(m_i))_{i \in [0,\ell-1]}$.

As previously, we describe our alternative technique as a family of algorithms $\overline{\mathsf{HomomorphicCompare}_{B,\ell}}$ in Algorithm 3, for homomorphic comparison with message spaces $[0, (\frac{B}{2})^\ell - 1]$, for any positive integer ℓ.

Algorithm 3. $\overline{\mathsf{HomomorphicCompare}_{B,\ell}}$ for message space $[0, (\frac{B}{2})^\ell - 1]$

```
    Input: two ciphertexts $c_1, c_2$ encrypting messages $m_1, m_2 \in [0, (\frac{B}{2})^\ell - 1]$, a bootstrapping key $bk$ and a scaling factor $k$
    Output: an $N$-LWE encryption of $k \cdot \mathrm{sign}(m_1 - m_2)$
1:  $acc = 0$
2:  for $i = \ell - 1, \ldots, 1$ do
3:      $acc := \mathsf{HomomorphicCompare}_{B,0}(acc + c_{1,i}, c_{2,i}, bk, \frac{B}{2})$
4:  end for
5:  return $\mathsf{HomomorphicCompare}_{B,0}(acc + c_{1,0}, c_{2,0}, bk, k)$
```

Correctness. After the i-th iteration of the loop, the accumulator acc contains the sign of $m_1^{(i)} - m_2^{(i)}$ scaled by $\frac{B}{2}$, where $m_b^{(i)} = \sum_{j \in \{0,\ldots,i\}} m_{b,\ell-1-j} \left(\frac{B}{2}\right)^{\ell-1-j}$ for $b \in \{0, 1\}$. Indeed, observe that for all $i \in \{1, \ldots, \ell - 1\}$, $\left(m_1^{(i)} - m_2^{(i)}\right)$ has the same sign as

$$\frac{B}{2} \cdot \mathrm{sign}\left(m_1^{(i-1)} - m_2^{(i-1)}\right) + m_{1,\ell-1-i} - m_{2,\ell-1-i},$$

because $|m_{1,\ell-1-i} - m_{2,\ell-1-i}| < \frac{B}{2}$. The correctness then follows from the one of $\mathsf{HomomorphicCompare}_{B,0}$.

2.5 Efficiency

In order to test the efficiency of our technique, we implemented our protocol and ran it on a Core i7-3630QM laptop, on which a bootstrapping from the TFHE library takes about 33 ms. For such a processor supporting parallel computations, the tree-based approach significantly outperforms the sequential one. We will then only consider this strategy in the following. We nevertheless note that our sequential strategy offers better noise management and thus better parameters, making it more efficient if evaluated on a single core.

The fact that our protocol allows to process $\log_2(B)$ bits at once might lead to select large B. Unfortunately, the size of B impacts the parameters of our system and thus its efficiency. A careful noise analysis is therefore necessary to select optimal values for B.

Let σ_{bs}^2 be the variance of the noise at the end of the bootstrapping, as defined in [15], theorem 6.3:

$$\sigma_{bs}^2 = n(k+1)\ell N \beta^2 \sigma_{bk}^2 + n(1+kN)\varepsilon^2 + n2^{-2(t+1)} + tn\sigma_{ks}^2$$

We use the same notation as in [15]. σ_{bk}^2 (resp. σ_{ks}^2) is the variance of the error of the bootstrapping key (resp. the key-switching key). k, ℓ, N, β and ϵ are parameters of the encryption schemes involved in **BlindRotate** whereas t and n are parameters of **KeySwitch**.

We have to correctly handle a message space of $2B$ slots even after adding $\gamma = \lfloor \log_2(B) \rfloor$ ciphertexts. We also have to take into account the noise resulting from rounding to multiples of $\frac{1}{2N}$.

Table 1. Timings obtained with three different sets of parameters for comparing 32 bits integers. For all of them, we have $k = 1$.

	B	N	n	σ_{ks}	σ_{bk}	β	t	Bootstr.	32bits comp.		
									1 core	8f cores	Max parall.
Set 1	4	2048	500	2^{-20}	2^{-50}	2048	17	72 ms	2232 ms	648 ms	360 ms
Set 2	4	4096	400	2^{-14}	2^{-70}	4096	13	126 ms	3902 ms	1137 ms	620 ms
Set 3	6	4096	750	2^{-18}	2^{-70}	4096	17	240 ms	3840 ms	1200 ms	960 ms

We thus get the following probability of correctness

$$\operatorname{erf}\left(\left(\frac{1}{4B} - \frac{n+1}{4N}\right)\frac{1}{\sigma_{bs}\sqrt{2\log_2(B)}}\right),$$

where erf is the Gauss error function.

This probability shows that increasing B requires to increase N, which is not a good strategy since the complexity of the bootstrapping is superlinear in N. For a given set of parameters (selected to ensure some level of security), one then simply has to choose the largest possible value for B. Interestingly, this means that, compared to binary decomposition, the efficiency of our protocol will increase with the security level.

We note that the flexibility in the choice of B (we can choose any value $B \geq 4$) allows a better noise management than in [9]. This means that our technique can probably be adapted to improve parameters of [9] for evaluation of neural networks where the message space is large.

We have tested our implementation for different sets of parameters. The results are presented in Table 1.

These three sets of parameters respectively yield a security [4] of 90/109/211 bits for the key switching key, and 230/378/378 bits for the bootstrapping key. The probability of error for a bootstrapping is respectively less than $2^{-50}/2^{-47}/2^{-89}$. Table 1 shows that our first set of parameters, with $B = 4$ and $N = 2048$, provides the best performances.

We note that the improvements from [9,46] halve the rounding cost by slightly unfolding the loop in BlindRotate. This allows us to basically double the message space at a very small cost. With the same noise analysis technique, we suggest to modify our first set of parameters as follows:

$$(N, n, \sigma_{ks}, \sigma_{bk}, \beta, t, k, \epsilon) = (1024, 500, 2^{-20}, 2^{-38}, 2048, 17, 1, 2^{-25})$$

This set of parameters yields a security of ≈90 bits for the key switching key, and ≈107 bits for the bootstrapping key. The probability of error for a bootstrapping is less than 2^{-50}. The running time for these parameters should be roughly 33 ms, given the experiments conducted in [9,46], which yields comparison of 32 bits integers in 1023 ms on a single core, 297 ms on 8 cores, and 165 ms with maximum parallelization.

For elements of comparison, using a binary decomposition requires 128 gates [32] for greater than comparison of 32 bits integers (what we are achieving is stronger, because we test equality as well), which would yield 4224 ms on the same laptop.

3 A Protocol for the Millionaires' Problem

In this section, we improve on the CEK protocol by avoiding one round induced by the plaintext equality test. This allows us to reduce the interaction to the minimum, while preserving efficiency. We first describe the more efficient protocol for small integers, before showing how it can easily be extended to larger integers by following the techniques in [10]. Even the protocol for larger messages only deals with bounded messages, however that bound grows really fast with the size of the RSA modulus chosen.

3.1 Preliminaries

The security of our protocol will rely on the Small RSA Subgroup Decision Assumption, defined in [10], inspired by [29]. Informally, it states that it is hard to distinguish a random element in a subgroup of $\mathbb{Z}_N^*$ from a random element. Let us introduce the following notation for our RSA quintuples (u, p_0, d, N, g):

- u is an integer such that the Discrete Logarithm Problem is infeasible in a subgroup of $\mathbb{Z}_N^*$ whose order is a prime of bit-length u;
- p_0 is a prime;
- d is an integer greater than 1;
- N is an integer of the form $N = pq$, whose factorisation is infeasible, where $p = 2 \cdot p_0^d \cdot p_s \cdot p_t + 1$ and $q = 2 \cdot p_0^d \cdot q_s \cdot q_t + 1$, with p_s and q_s primes of bit-length u, and p_t and q_t primes whose bit-length is not u;
- g is an element of order p_0^d in $\mathbb{Z}_N^*$;
- $\mathbb{QR}_N$ is the set of quadratic residues mod N.

Definition 2. *We say that the small RSA subgroup decision assumption holds if given an RSA quintuple (u, p_0, d, N, g), the distributions x and $x^{p_0^d \cdot p_t \cdot q_t}$ are computationally indistinguishable, for $x \xleftarrow{\$} \mathbb{QR}_N$ a uniformly random quadratic residue mod N.*

In other words, the small RSA subgroup assumption states that it is hard to distinguish an element of order $p_s \cdot q_s$ from a random quadratic residue in $\mathbb{Z}_N^*$. Since pinpointing the optimal parameters for security and efficiency is not trivial in this setting, we discuss in more details our choices of parameters in Sect. 4.1.

3.2 Protocol for Small Integers

We describe in this section our protocol for secure integer comparison but first start by providing the intuition behind it.

Intuition. As in [10], our protocol makes use of the threshold properties of prime power subgroups of $\mathbb{Z}_N^*$. We will then assume that there exist a prime p_0 and an integer $d > 0$ such that p_0^d divides $\phi(N)$. Let g be an element of order p_0^d in $\mathbb{Z}_N^*$ and $\mathbb{G}$ be the cyclic subgroup generated by g.

In [10], the core idea is that the element $C = g^{p_0^{d+m_1-m_2}}$ can be used to compare the integers m_1 and m_2. Indeed, this element is equal to 1 if and only if $m_1 \geq m_2$. However, to prevent any leakage of information on its secret integer m_2, the second party B has to blind C using a random element $g^s \in \mathbb{G}$ leading to the following problem for the first party A : in all cases (namely $m_1 \geq m_2$ or $m_1 < m_2$) it receives a random element $g^{s'}$. To compare m_1 and m_2, Carlton *et al.* therefore propose (1) to recover s' from $g^{s'}$ (*i.e.* to compute a discrete logarithm) and (2) to run a plaintext equality test (PET) between A and B to compare s' and s. It implies at least another pass and involves additional primitives (*e.g.* homomorphic encryption in [10]).

The goal of our protocol is to remove these last steps and so to reduce the number of passes while avoiding the computational overhead of PET protocols.

Let $0 < a \leq d$ be a public integer such that $p_0^a \geq 2^\lambda$ where λ is the security parameter[3]. Of course, this requirement implies larger subgroups $\mathbb{G}$ but, as we will explain, this is not a significant problem for us since we will no longer need to compute discrete logarithms in $\mathbb{G}$. Let $\mathbb{H}$ be a subgroup of order coprime with p_0, generated by some element h.

To compare $m_1, m_2 \leq d/a$, the party A computes $C = g^{p_0^{a \cdot m_1}} \cdot h^{r_1}$, for some random scalars r_1, and sends it to B. The latter then selects three random scalars: $u \in [0, p_0^a - 1]$, $v \in [0, p_0^d - 1]$ and $r_2 \in [0, \bar{b} - 1]$ where $\bar{b}$ is some bound on the order of $\mathbb{H}$. It then computes and sends to A two elements:

$$D \leftarrow C^{u \cdot p_0^{d-a \cdot m_2}} \cdot g^v \cdot h^{r_2} \text{ and } D' \leftarrow \mathcal{H}(g^v)$$

where $\mathcal{H}$ is some hash function. One can note that $D = g^{u \cdot p_0^{d+a(m_1-m_2)}+v} \cdot h^*$ for some random element $h^* \in \mathbb{H}$. By using its knowledge of the factorization of N, A can easily remove h^* and recover $C' = g^{u \cdot p_0^{d+a(m_1-m_2)}+v}$. There are then two different cases:

1. If $m_1 \geq m_2$, then $C' = g^v$ which can easily be detected by A since $\mathcal{H}(C') = D'$ in such a case.
2. Else, $\mathcal{H}(C')$ differs from D' with overwhelming probability, leading A to conclude that $m_1 < m_2$.

From the security point of view, one can note that B always received values masked by a random element h of $\mathbb{H}$. It is thus unable to learn information on m_1 unless it can solve the small RSA subgroup problem. In the case where $m_1 \geq m_2$ the pair (D, D') received by A is independent of m_2 so this entity cannot learn any information on this value. In the case where $m_1 < m_2$, the element C' is a random element of $\mathbb{G}$ (since v is random) but A has an information on the

[3] We will provide more details on the parameters in Sect. 4.1.

$$\begin{array}{lcl}
\textbf{Party A}\ (pp, sp, m_1 \in [0, d/a]) & & \textbf{Party B}\ (pp, m_2 \in [0, d/a]) \\
\hline
r_1 \xleftarrow{\$} [1, \overline{b}-1] & & \\
C = g^{p_0^{a \cdot m_1}} h^{r_1} & \xrightarrow{\quad C \quad} & u \xleftarrow{\$} [1, p_0^a - 1],\ v \xleftarrow{\$} [1, p_0^d - 1], \\
 & & r_2 \xleftarrow{\$} [1, \overline{b}-1] \\
 & & D \leftarrow C^{u \cdot p_0^{d - a \cdot m_2}} \cdot g^v \cdot h^{r_2} \\
 & \xleftarrow{\quad (D, D') \quad} & D' \leftarrow \mathcal{H}(g^v) \\
C' \leftarrow D^c & & \\
\text{If } D' = \mathcal{H}(C'), \text{ return } (m_1 \geq m_2) & & \\
\text{Else, return } (m_1 < m_2). & &
\end{array}$$

Fig. 3. A two-pass protocol for secure comparison of small integers.

blinding factor g^v since it knows $D' = \mathcal{H}(g^v)$. Since a hash function is assumed to be one-way, A cannot recover g^v directly from D but can try to guess it either directly (with probability $1/p_0^d$) or by guessing the cofactor $g^{u \cdot p_0^{d+a(m_1-m_2)}}$. However, the latter element is of order a least $p_0^a > 2^\lambda$ which makes a correct guess very unlikely when u is random.

Our Construction. Our protocol is described in Fig. 3 and makes use of the following parameters:

- $N = p \cdot q$ is a product of two primes p and q
- p_0, p_s and q_s are prime numbers such that $p_s | p-1$, $q_s | q-1$ and p_0^d divides both $p-1$ and $q-1$ for some integer $d > 0$
- $0 < a \leq d$ is an integer smaller than d
- $g \in \mathbb{Z}_N^*$ is an element of order p_0^d in both $\mathbb{Z}_p^*$ and $\mathbb{Z}_q^*$ while $h \in \mathbb{Z}_N^*$ is an element of order $p_s \cdot q_s$.
- $\overline{b}$ is an upper bound on $p_s \cdot q_s$
- c is an integer such that $c = p_s \cdot q_s \cdot [(p_s \cdot q_s)^{-1}]_{p_0^d}$, where $[x]_{p_0^d}$ denotes x mod p_0^d.
- $\mathcal{H} : \mathbb{Z}_N^* \rightarrow \{0,1\}^*$ is a cryptographic hash function.

The public parameters pp are defined as $\{N, a, p_0, d, g, h, \overline{b}\}$ whereas the secret parameters sp, only known to A, are $\{p, q, c\}$.

Correctness. As explained above, the element C' computed by A is exactly $g^{u \cdot p_0^{d+a(m_1-m_2)}+v}$. If $m_1 \geq m_2$, then $C' = g^v$ and $D' = \mathcal{H}(C')$. Else, $m_1 - m_2 < 0$ and p_0^a divides the order of $g^{p_0^{d+a(m_1-m_2)}}$. Since $u \in [1, p_0^a - 1]$, $g^{u \cdot p_0^{d+a(m_1-m_2)}} \neq 1$ and $C' \neq g^v$. Therefore, $D = \mathcal{H}(C')$ would imply a collision of the hash function $\mathcal{H}$, which is very unlikely.

3.3 Security of the Protocol for Small Integers

We prove the security for both A and B against honest-but-curious adversaries in the random oracle model. This means that A (respectively B) will not learn any information about m_2 (resp. m_1), except whether it is bigger or smaller than m_1 (resp. m_2).

Privacy of $\boldsymbol{A}$. We first show that B learns nothing about m_1 in this protocol. More formally, we have the following security theorem.

Theorem 3. *Under the Small RSA Subgroup Decision Assumption, B's view is computationally indistinguishable from a uniformly random element in $\mathbb{QR}_N$ for any message m_1.*

Proof. We show that we can use an adversary that has probability ϵ of distinguishing B's view from a uniformly random element in $\mathbb{Z}_N^*$ to break the Small RSA Subgroup Decision Assumption with the same probability.

Let us define a first game where the reduction $\mathcal{R}$ publishes a valid set of parameters $\{N, a, p_0, d, g, h, \overline{b}\}$ (here valid means in particular that h is of order $p_s \cdot q_s$) and plays the role of A as defined in Fig. 3.

In a second game, $\mathcal{R}$ proceeds as in the previous game except that it generates a random element $z \overset{\$}{\leftarrow} \mathbb{Z}_N^*$ and sets $h = z^2$. In such a case, the element C received by B is a uniformly random element in $\mathbb{QR}_N$ for any message m_1.

Now let us assume that an adversary $\mathcal{A}$ is able to distinguish these two games with probability ϵ. On input an RSA quintuple (u, p_0, d, N, g) and an instance x to the small RSA subgroup decision problem, $\mathcal{R}$ defines the public parameters as $\{N, a, p_0, d, g, h, \overline{b}\}$, where a and $\overline{b}$ are selected as usual, but where $h = x$. If x is of order $p_s \cdot q_s$, then this is exactly our first game. Else, x is a uniformly random quadratic residue and $\mathcal{A}$ is playing our second game. Therefore, $\mathcal{A}$ will succeed in breaking the Small RSA Subgroup Decision Assumption with probability ϵ, which implies that ϵ is negligible.

Privacy of $\boldsymbol{B}$. We now show that A only learns the output of the protocol $(m_1 \geq m_2)$ and nothing else about m_2.

Theorem 4. *There exists an efficient simulator $\mathcal{S}$, such that $\mathcal{S}(1^\lambda, (m_1 \geq m_2))$ is statistically indistinguishable from A's view for any messages m_1 and m_2 in the random oracle model.*

Proof. The simulator S works as follows:

- If $m_1 < m_2$ pick random elements $v, v' \overset{\$}{\leftarrow} [1, p_0^d - 1], r \overset{\$}{\leftarrow} [1, \overline{b} - 1]$ and return $\left(g^v \cdot h^r, \mathcal{H}(g^{v'})\right)$.
- Else pick random elements $v \overset{\$}{\leftarrow} [1, p_0^d - 1]$, $r \overset{\$}{\leftarrow} [1, \overline{b} - 1]$ and return $(g^v \cdot h^r, \mathcal{H}(g^v))$.

In the first case, we show that the statistical distance between the view of A and the output of S is negligible: The two distribution only differ when the adversary queries the random oracle with input $g^{v - u \cdot p_0^{d + a(m_1 - m_2)}}$ and realizes that it's different from $\mathcal{H}(g^{v'})$. However, this can never happen with non-negligible

```
Party A $(pp, sp, m_1 = \sum_{i=0}^{\ell} m_{1,i} \cdot b^i)$        Party B $(pp, m_2 = \sum_{i=0}^{\ell} m_{2,i} \cdot b^i)$
For $i \in [0, \ell]$:   $r_{1,i} \xleftarrow{\$} [1, \bar{b} - 1]$
    $C_i = g^{-m_1^{(i)}} g^{p_0^{a(m_{1,i})}} h^{r_{1,i}}$    $\xrightarrow{\{C_i\}_{i=0}^{\ell}}$
                                        For $i \in [0, \ell]$:
                                            $u_i \xleftarrow{\$} [1, p_0^a - 1]$, $v_i \xleftarrow{\$} [1, p_0^d - 1]$,
                                            $r_{2,i} \xleftarrow{\$} [1, \bar{b} - 1]$, $D'_i \leftarrow \mathcal{H}(g^{v_i})$
                                            $D_i \leftarrow (C_i \cdot g^{m_2^{(i)}})^{u_i \cdot p_0^{d-a(m_{2,i}+1)}} \cdot g^{v_i} \cdot h^{r_{2,i}}$
                                $\xleftarrow{\pi(\{(D_i, D'_i)\}_{i=0}^{\ell})}$
For $i \in [0, \ell]$:
    $C'_i \leftarrow D_i^c$
If $\exists i$ s.t. $D'_i = \mathcal{H}(C'_i)$,
    return $(m_1 > m_2)$
Else, return $(m_1 \leq m_2)$.
```

Fig. 4. A two-pass protocol for secure integer comparison. π is a random permutation of the symmetric group $S_{\ell+1}$.

probability because $g^{u \cdot p_0^{d+a(m_1+m_2)}}$ is uniform in a subgroup of order at least p_0^a, which is exponential in the security parameter for the parameters we suggest in Sect. 4.1.

In the second case, the distribution is exactly the same as in the protocol.

3.4 A Protocol for Large Integers

As we explain in Sect. 4.1, the constraints that apply on the different parameters imply that the protocol of Fig. 3 can only be used to compare small messages. However, our protocol can be extended to compare larger integers by adapting a technique used in previous works (e.g. [10, 20]). Let $m_1 = \sum_{i=0}^{\ell} m_{1,i} \cdot b^i$ and $m_2 = \sum_{i=0}^{\ell} m_{2,i} \cdot b^i$ be the rewriting of the messages m_1 and m_2 in base $b = \lfloor d/a \rfloor$ (*i.e.* $m_{j,i} \in [0, b-1]$ for $i \in [0, \ell]$ and $\ell = \lceil \log_b(M) \rceil$, where $M < p_0^a$ is a bound on the messages m_1 and m_2). For $i \in [0, \ell]$, we define $m_j^{(i)} = \sum_{k=i+1}^{\ell} m_{j,k} b^k$. Our protocol is described in Fig. 4 and uses the same parameters as in Sect. 3.2.

Remark 5. The bound $M < p_0^a$ is not a strong constraint for most applications since $p_0^a > 2^\lambda$ (see Sect. 4.1 below). This protocol is therefore sufficient to compare integers of reasonable size but, if need be, it can easily be extended for even larger integers. Indeed, instead of including $g^{m_1^{(i)}}$ in C_i, the party A can encrypt it separately as $E_i = g^{-m_i^{(1)}} h^{r'_{1,i}}$ and sends it along with C_i. The party B will now compute D_i as $(C^{p_0^{d-a(m_2+1)}} E_i \cdot g^{m_2^{(i)}})^{u_i} \cdot g^{v_i} \cdot h^{r_{2,i}}$ leading to a much larger bound of $B < p_0^d \sim N^{1/4}$.

Correctness. We prove that $m_1 > m_2 \Leftrightarrow \exists i \in [0, \ell]$ such that $D'_i = \mathcal{H}(C'_i)$.

First note that if $m_1 > m_2$, then $\exists i \in [0,\ell]$ such that (1) $m_1^{(i)} = m_2^{(i)}$ and (2) $m_{1,i} > m_{2,i}$, or equivalently $(m_{1,i} - m_{2,i}) \geq 1$. For such an index i, we have:

$$C'_i = g^{u_i[(m_2^{(i)} - m_1^{(i)})p_0^{d-a(m_{2,i}+1)} + p_0^{d+a(m_{1,i}-m_{2,i}-1)}] + v_i} = g^{v_i}$$

which means that $D'_i = \mathcal{H}(C'_i)$. Now, let us assume that $\exists i \in [0,\ell]$ such that $D'_i = \mathcal{H}(C'_i)$. Due to the collision resistance of $\mathcal{H}$, this means (with overwhelming probability) that $g^{v_i} = C'_i$ and so that:

$$u_i[(m_2^{(i)} - m_1^{(i)})p_0^{d-a(m_{2,i}+1)} + p_0^{d+a(m_{1,i}-m_{2,i}-1)}] = 0 \bmod p_0^d$$

One can note that the powers of p_0 between the square brackets are either multiples of p_0^d or of the form p_0^t with $t \leq d - a$. Since $0 < u_i < p_0^a$, this implies that:

$$\begin{aligned}
&[(m_2^{(i)} - m_1^{(i)})p_0^{d-a(m_{2,i}+1)} + p_0^{d+a(m_{1,i}-m_{2,i}-1)}] = 0 \bmod p_0^d \\
\Leftrightarrow\ & p_0^{d-a(m_{2,i}+1)}[(m_2^{(i)} - m_1^{(i)}) + p_0^{a\cdot m_{1,i}}] = 0 \bmod p_0^d \\
\Leftrightarrow\ & (m_2^{(i)} - m_1^{(i)}) + p_0^{a\cdot m_{1,i}} = 0 \bmod p_0^{a(m_{2,i}+1)} \quad \text{(I)}
\end{aligned}$$

For all $i \in [0,\ell]$, we have $m_2^{(i)} - m_1^{(i)} \leq M - (b-1) \leq M - 1 < p_0^a - 1$. We can therefore distinguish two cases.

- Case 1: $m_{2,i} \geq m_{1,i}$. From $(m_2^{(i)} - m_1^{(i)}) + p_0^{a\cdot m_{1,i}} < p_0^a - 1 + p_0^{a\cdot m_{2,i}} \leq p_0^{a\cdot m_{2,i}+1}$ and the Equation (I), we can deduce that $(m_2^{(i)} - m_1^{(i)}) + p_0^{a\cdot m_{1,i}} \leq 0$ and in particular that $m_1^{(i)} > m_2^{(i)}$. The latter inequality means that $m_1 > m_2$, which concludes our proof.
- Case 2: $m_{2,i} < m_{1,i}$. The Equation (I) then becomes:

$$(m_2^{(i)} - m_1^{(i)}) = 0 \bmod p_0^{a(m_{2,i}+1)}.$$

  However, we know that $-p_0^a < m_2^{(i)} - m_1^{(i)} < p_0^a$, so the previous equation can only hold if $m_2^{(i)} = m_1^{(i)}$. Here again, this means that $m_1 > m_2$.

Therefore, $m_1 > m_2 \Leftrightarrow \exists i \in [0,\ell]$ such that $D'_i = \mathcal{H}(C'_i)$, which proves the correctness of our protocol.

4 Security of the Protocol for Large Integers

The proof of security for this protocol is very similar to the previous one, and the claims are similar: A's data will be computationally secure, while B's data will be statistically secure. One key observation is that each pair (D_i, D'_i) proves or disproves the statement $m_1^{(i)} = m_2^{(i)} \wedge m_{1,i} > m_{2,i}$. At most one of them can be satisfied, and one is satisfied if and only if $m_1 > m_2$.

Privacy of *A*. We first show that B learns nothing about m_1 in this protocol. More formally, we have the following security theorem.

Theorem 6. *Under the Small RSA Subgroup Decision Assumption, B's view is computationally indistinguishable from a uniformly random element in $\mathbb{QR}_N{}^{\ell+1}$ for any message m_1.*

Proof. As in the previous case, we can show this indistinguishability by replacing the element h by a small RSA subgroup decision challenge. If the element has order $p_s \cdot q_s$, then the view of B is identical to the real protocol. Otherwise, B only receives a uniformly random element in $\mathbb{QR}_N{}^{\ell+1}$. Thus, any adversary breaking the privacy of A can be used to solve the Small RSA Subgroup Decision problem.

Privacy of *B*. We now show that A only learns the output of the protocol $(m_1 > m_2)$ and nothing else about m_2.

Theorem 7. *There exists an efficient simulator $\mathcal{S}$, such that $\mathcal{S}(1^\lambda, (m_1 > m_2))$ is statistically indistinguishable from A's view for any messages m_1 and m_2 in the random oracle model.*

Proof. The simulator S works as follows:

– If $m_1 \leq m_2$, for each $i \in [0, \ell]$ pick random elements $v_i, v_i' \xleftarrow{\$} [1, p_0^d - 1], r_i \xleftarrow{\$} [1, \bar{b} - 1]$ and sets $\left(D_i = g^{v_i} \cdot h^{r_i}, D_i' = \mathcal{H}(g^{v_i'})\right)$. Then it returns $\{(D_i, D_i')\}_{i=0}^{\ell}$;
– Else pick a random index $j \in [0, \ell]$, random elements $v_j \xleftarrow{\$} [1, p_0^d - 1]$, $r_j \xleftarrow{\$} [1, \bar{b} - 1]$ and sets $\left(D_j = g^{v_j} \cdot h^{r_j}, D_j' = \mathcal{H}(g^{v_j})\right)$. Then, for each $i \in [0, \ell]$, $i \neq j$, pick random elements $v_i, v_i' \xleftarrow{\$} [1, p_0^d - 1], r_i \xleftarrow{\$} [1, \bar{b} - 1]$ and sets $\left(D_i = g^{v_i} \cdot h^{r_i}, D_i' = \mathcal{H}(g^{v_i'})\right)$. Finally, returns $\{(D_i, D_i')\}_{i=0}^{\ell}$

As previously, in the first case, we show that the statistical distance between the view of B and the output of S is negligible: The two distribution can only differ when the adversary queries the random oracle with input $g^{\tilde{v}_{k,i}}$ for some indices $i, k \in [0, \ell]$, where

$$\tilde{v}_{k,i} = v_k - u_i \cdot (m_2^{(i)} - m_1^{(i)}) \cdot p_0^{d - a(m_{2,i}+1)} + u_i \cdot p_0^{d - a(m_{1,i} - m_{2,i}+1)}.$$

However, as we have shown for correctness,

$$(m_2^{(i)} - m_1^{(i)}) \cdot p_0^{d - a(m_{2,i}+1)} + p_0^{d - a(m_{1,i} - m_{2,i}+1)} \neq 0 \bmod p_0^d,$$

unless $m_2^{(i)} = m_1^{(i)}$ and $m_{i,1} > m_{i,2}$. Thus, the $\tilde{v}_{k,i}$ are uniformly random in an exponentially big subgroup for parameters suggested in Sect. 4.1 (of order at least p_0^a). Since the adversary runs in polynomial time, the probability that he queries the random oracle on one of these input is negligible.

In the second case, the distribution is exactly the same as in the protocol for the index j that satisfies $m_2^{(j)} = m_1^{(j)}$ and $m_{j,1} > m_{j,2}$. For all the other indices, we use the same argument as in the first case: the two distributions can only differ when the adversary queries the random oracle with input $g^{\tilde{v}_{k,i}}$ for some indices $i, k \in [0, \ell]$, $k \neq j$.

4.1 Parameters

One must be careful when using RSA modulus whose prime factors have unusual decomposition, as shown in [17,18,36,39]. We discuss in this section the bounds on the different parameters to ensure the security of our protocols and their impact on efficiency.

There are several attacks that we must take into account due to the special form of our RSA modulus. One of them is the Coron et al. attack [18] that gives us a bound on the order of h: $\mathsf{log}_2(p_s) = \mathsf{log}_2(q_s) \geq 2\lambda$.

The condition $p_0^d | p-1$ and $p_0^d | q-1$ makes our protocol vulnerable to the McKee's and Pinch's attack [36] and thus imposes the upper bound $N^{1/4}/2^\lambda$ on the value of p_0^d, where λ is the security parameter. We must therefore have:

$$d \cdot \mathsf{log}(p_0) \leq \frac{1}{4}\mathsf{log}(N) - \lambda\mathsf{log}(2).$$

This gives us a bound on the messages m that can be compared in a single execution of our protocol:

$$m \leq d/a \leq \frac{\mathsf{log}(N)/4 - \lambda\mathsf{log}(2)}{\mathsf{log}(p_0) \cdot a}$$

Ideally, we would like to choose $p_0 = 2$ and $a = 1$ to get the largest bound. However, we must additionally ensure that the random scalar u cannot be guessed with non-negligible probability. This means that:

$$a \cdot \mathsf{log}(p_0) \geq \lambda\mathsf{log}(2)$$

Combining these two constraints leads to the following bound on the messages:

$$m \leq \frac{\mathsf{log}(N)/4 - \lambda\mathsf{log}(2)}{\mathsf{log}(p_0) \cdot a} \leq \frac{\mathsf{log}(N)/4 - \lambda\mathsf{log}(2)}{\lambda\mathsf{log}(2)}$$

One can note that this bound on m is independent of p_0 and a. This means that there is a great flexibility in the choice of these parameters provided that the requirement $a \cdot \mathsf{log}(p_0) \geq \lambda\mathsf{log}(2)$ is fulfilled.

Interestingly, the fact that N grows more quickly than the security parameter λ [33,40] implies that this bound depends on the security parameter. In particular, compared to previous protocols (*e.g.* [20,42,43]) that work with bit-wise encrypted values, the speedup factor will be larger for $\lambda = 256$ than for $\lambda = 128$.

4.2 Efficiency

As we mention in the introduction, there is a wide range of solutions to the Millionaire's problem from garbled circuits to homomorphic encryption. And even among solutions based on homomorphic encryption, one can find different tradeoffs such as the two-passes protocol proposed by Damgård *et al.* [20] and the protocol proposed by Carlton *et al.* [10] that allows to process several bits

Table 2. Efficiency comparison between related works and our protocol. E_t refers to the cost of an exponentiation whose exponent is smaller than t. $\mathsf{dlog}_{\mathbb{G}}$ refers to the cost of computing a discrete logarithm in the group $\mathbb{G}$. $\mathsf{Enc}_{\oplus}$, $\mathsf{Dec}_{\oplus}$ and $\mathsf{Rand}_{\oplus}$ respectively refer to the cost of encrypting, decrypting and re-randomizing with the additively homomorphic encryption scheme Π used for the PET. Finally, $C_{\oplus}$ refers to a ciphertext generated using Π and H to a digest generated by $\mathcal{H}$.

Schemes	DGK [20]	CEK [10]	Our work
Computational Cost (A)	$\log_2(M)[1\ \mathsf{E}_{\bar{b}} + 1\ \mathsf{E}_{\bar{b}p_0^d}]$	$\log_{b'}(M)[1\ \mathsf{E}_{\bar{b}} + 1\ \mathsf{E}_{\bar{b}p_0^d} + 1\ \mathsf{dlog}_{\mathbb{G}} + 1\ \mathsf{Rand}_{\oplus}]$	$\log_b(M)[1\ \mathsf{E}_{\bar{b}} + 1\ \mathsf{E}_{\bar{b}p_0^d}]$
Computational Cost (B)	$\log_2(M)[1\ \mathsf{E}_{\bar{b}}]$	$\log_{b'}(M)[1\ \mathsf{E}_{\bar{b}} + 2\ \mathsf{E}_{p_0^d} + 1\ \mathsf{Enc}_{\oplus} + 1\ \mathsf{Dec}_{\oplus}]$	$\log_b(M)[1\ \mathsf{E}_{\bar{b}} + 2\ \mathsf{E}_{p_0^d}]$
Communication Cost	$\log_2(M)[2\ \mathbb{Z}_N^*]$	$\log_{b'}(M)[2\ \mathbb{Z}_N^* + 2\ C_{\oplus}]$	$\log_b(M)[2\ \mathbb{Z}_N^* + 1\ H]$
Passes	2	3–4[a]	

[a] Carlton *et al.* explain how to combine the last pass of their protocol with the first one of the PET, leading to a protocol with 3 passes instead of 4. However, in such a case, the entity (A) that initiated the protocol does not know the result of the comparison (only B knows it), contrarily to our protocol or to the DGK one.

at once but at the cost of an extra plaintext equality check (PET) involving additional passes. We therefore choose to compare our protocol with both solutions by providing in Table 2 an assessment of the different respective costs. In particular, we stress that the cost of additional passes, and more generally the communication cost, should not be underestimated. It can indeed be very high for some devices such as smartcards, and even exceeds computational cost in some cases (see *e.g.* [22]).

For the sake of clarity, we do not consider additions, multiplications and hash evaluations whose costs are negligible compared to the other operations. We also assume, for all protocols, that the elements only depending on the messages m_1 or m_2 and on the system parameters (*e.g.* $g^{p_0^{a \cdot m_{1,i}}}$) have been pre-computed.

For proper comparison, we need to specify the values of the factors $\log_{b'}(M)$ and $\log_b(M)$ respectively used in the evaluation of the complexity of the CEK protocol and ours. This is not a trivial task as the constraints placed on our parameters prevent us from using conventional RSA moduli. This is done in Sect. 4.1 where we show that:

$$b = \lfloor \frac{\log(N)/4 - \lambda \log(2)}{\lambda \log(2)} \rfloor$$

A similar analysis for the CEK protocol shows that:

$$b' = \lfloor \frac{\log(N)/4 - \lambda \log(2)}{\log(p_0)} \rfloor$$

Our protocol can easily be compared to the DGK one since they both involve 2 passes and do not need PET. Actually, one can note that our protocol is roughly $\frac{\log(b)}{\log(2)}$ more efficient than the DGK one. For a security parameter λ of respectively 128, 192 and 256, we have $b = 5$, $b = 9$ and $b = 14$ (see [40]), which means that the speedup factor is always greater than 2 and will increase with the security level.

Conversely, comparing our solution with the CEK one is more complex, as they are very different protocols. Ours only requires 2 passes and does not require a PET, thus avoiding additional interactions and the costs associated with an homomorphic encryption system. Regarding computational costs, a single execution of our comparison protocol is more efficient than the CEK one, but this is offset by the fact that CEK requires to run less individual comparison tests because $\log_{b'}(M) < \log_b(M)$. However, we note that the ratio $\frac{\log_b(M)}{\log_{b'}(M)}$ decreases towards 1 as λ increases due to the existence of subexponential factorization algorithms (see [33,40] and references therein), meaning that the number of comparison tests for both solutions will tend to be similar in the future.

5 Conclusion

More than three decades after its introduction by Yao, the Millionaires' problem has proved very important in cryptography, and more generally in most use-cases involving secure computation (*e.g.* machine learning on private data). It has drawn attention from many researchers that have provided a wide range of solutions, based on different primitives or addressing different versions of the original problem. However, despite all this work, secure integer comparison remains a complex issue, all the existing solutions entailing either a large amount of computations or a large amount of communication.

In this work, we have introduced new solutions to the Millionaires' problem in two different settings. Our first one extends the recent FHE construction of Bourse *et al.* [9] to enable efficient computation of the encrypted boolean $(m_1 \leq m_2)$ given only the encryption of (a-priori unbounded) integers m_1 and m_2. Our second solution leverages the threshold homomorphic encryption scheme of Carlton *et al.* [10] to construct a two-passes integer comparison protocol that improves over the state-of-the art. Although these constructions are very different, they both share the same guiding principles, namely reducing as much as possible the number of interactions and avoiding bitwise decomposition of the integers. Regarding the latter point, this concretely means that our protocols achieve a $\log_2(b)$ speedup factor compared to most homomorphic-based solutions, where $b > 4$ is some integer depending on the parameters of our constructions.

Acknowledgements. This work is supported by the European Union PROMETHEUS project (Horizon 2020 Research and Innovation Program, grant 780701) and PAPAYA project (Horizon 2020 Innovation Program, grant 786767). The authors are also grateful for the support of the ANR through project ANR-16-CE39-0014 PERSOCLOUD.

References

1. Abdalla, M., Bourse, F., De Caro, A., Pointcheval, D.: Simple functional encryption schemes for inner products. In: Katz, J. (ed.) PKC 2015. LNCS, vol. 9020, pp. 733–751. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-46447-2_33
2. Abspoel, M., Bouman, N.J., Schoenmakers, B., de Vreede, N.: Fast secure comparison for medium-sized integers and its application in binarized neural networks. In: Matsui, M. (ed.) CT-RSA 2019. LNCS, vol. 11405, pp. 453–472. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-12612-4_23
3. Agrawal, S., Libert, B., Stehlé, D.: Fully secure functional encryption for inner products, from standard assumptions. In: Robshaw, M., Katz, J. (eds.) CRYPTO 2016, Part III. LNCS, vol. 9816, pp. 333–362. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-53015-3_12
4. Albrecht, M.R., Player, R., Scott, S.: On the concrete hardness of learning with errors. J. Math. Cryptol. **9**(3), 169–203 (2015)
5. Bellare, M., Hoang, V.T., Keelveedhi, S., Rogaway, P.: Efficient garbling from a fixed-key blockcipher. In: 2013 IEEE SSP, pp. 478–492 (2013)
6. Bellare, M., Hoang, V.T., Rogaway, P.: Foundations of garbled circuits. In: ACM CCS 2012, pp. 784–796 (2012)
7. Blake, I.F., Kolesnikov, V.: Conditional encrypted mapping and comparing encrypted numbers. In: Di Crescenzo, G., Rubin, A. (eds.) FC 2006. LNCS, vol. 4107, pp. 206–220. Springer, Heidelberg (2006). https://doi.org/10.1007/11889663_18
8. Bost, R., Popa, R.A., Tu, S., Goldwasser, S.: Machine learning classification over encrypted data. In: NDSS (2015)
9. Bourse, F., Minelli, M., Minihold, M., Paillier, P.: Fast homomorphic evaluation of deep discretized neural networks. In: Shacham, H., Boldyreva, A. (eds.) CRYPTO 2018, Part III. LNCS, vol. 10993, pp. 483–512. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-96878-0_17
10. Carlton, R., Essex, A., Kapulkin, K.: Threshold properties of prime power subgroups with application to secure integer comparisons. In: Smart, N.P. (ed.) CT-RSA 2018. LNCS, vol. 10808, pp. 137–156. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-76953-0_8
11. Cheon, J.H., Kim, M., Kim, M.: Search-and-compute on encrypted data. In: Brenner, M., Christin, N., Johnson, B., Rohloff, K. (eds.) FC 2015. LNCS, vol. 8976, pp. 142–159. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-48051-9_11
12. Chillotti, I., Gama, N., Georgieva, M., Izabachène, M.: Faster fully homomorphic encryption: bootstrapping in less than 0.1 s. In: Cheon, J.H., Takagi, T. (eds.) ASIACRYPT 2016, Part I. LNCS, vol. 10031, pp. 3–33. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-53887-6_1
13. Chillotti, I., Gama, N., Georgieva, M., Izabachène, M.: A homomorphic LWE based E-voting scheme. In: Takagi, T. (ed.) PQCrypto 2016. LNCS, vol. 9606, pp. 245–265. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-29360-8_16
14. Chillotti, I., Gama, N., Georgieva, M., Izabachène, M.: Faster packed homomorphic operations and efficient circuit bootstrapping for TFHE. In: Takagi, T., Peyrin, T. (eds.) ASIACRYPT 2017, Part I. LNCS, vol. 10624, pp. 377–408. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-70694-8_14
15. Chillotti, I., Gama, N., Georgieva, M., Izabachène, M.: TFHE: fast fully homomorphic encryption over the torus. Cryptology ePrint Archive, Report 2018/421 (2018). https://eprint.iacr.org/2018/421

16. Chou, T., Orlandi, C.: The simplest protocol for oblivious transfer. In: Lauter, K., Rodríguez-Henríquez, F. (eds.) LATINCRYPT 2015. LNCS, vol. 9230, pp. 40–58. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-22174-8_3
17. Coppersmith, D.: Small solutions to polynomial equations, and low exponent RSA vulnerabilities. J. Cryptol. **10**(4), 233–260 (1997)
18. Coron, J.-S., Joux, A., Mandal, A., Naccache, D., Tibouchi, M.: Cryptanalysis of the RSA subgroup assumption from TCC 2005. In: Catalano, D., Fazio, N., Gennaro, R., Nicolosi, A. (eds.) PKC 2011. LNCS, vol. 6571, pp. 147–155. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-19379-8_9
19. Crawford, J.L.H., Gentry, C., Halevi, S., Platt, D., Shoup, V.: Doing real work with FHE: the case of logistic regression. In: WAHC@CCS 2018, pp. 1–12 (2018)
20. Damgård, I., Geisler, M., Krøigaard, M.: Efficient and secure comparison for on-line auctions. In: Pieprzyk, J., Ghodosi, H., Dawson, E. (eds.) ACISP 2007. LNCS, vol. 4586, pp. 416–430. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-73458-1_30
21. Damgård, I., Geisler, M., Krøigaard, M.: A correction to 'efficient and secure comparison for on-line auctions'. IJACT **1**(4), 323–324 (2009)
22. Desmoulins, N., Lescuyer, R., Sanders, O., Traoré, J.: Direct anonymous attestations with dependent basename opening. In: Gritzalis, D., Kiayias, A., Askoxylakis, I. (eds.) CANS 2014. LNCS, vol. 8813, pp. 206–221. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-12280-9_14
23. Ducas, L., Micciancio, D.: FHEW: bootstrapping homomorphic encryption in less than a second. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015, Part I. LNCS, vol. 9056, pp. 617–640. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-46800-5_24
24. Feige, U., Kilian, J., Naor, M.: A minimal model for secure computation (extended abstract). In: 26th ACM STOC, pp. 554–563 (1994)
25. Fischlin, M.: A cost-effective pay-per-multiplication comparison method for millionaires. In: Naccache, D. (ed.) CT-RSA 2001. LNCS, vol. 2020, pp. 457–471. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-45353-9_33
26. Garay, J., Schoenmakers, B., Villegas, J.: Practical and secure solutions for integer comparison. In: Okamoto, T., Wang, X. (eds.) PKC 2007. LNCS, vol. 4450, pp. 330–342. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-71677-8_22
27. Gentry, C.: Fully homomorphic encryption using ideal lattices. In: 41st ACM STOC, pp. 169–178 (2009)
28. Gentry, C., Sahai, A., Waters, B.: Homomorphic encryption from learning with errors: conceptually-simpler, asymptotically-faster, attribute-based. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013, Part I. LNCS, vol. 8042, pp. 75–92. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-40041-4_5
29. Groth, J.: Cryptography in subgroups of $\mathbb{Z}_n^*$. In: Kilian, J. (ed.) TCC 2005. LNCS, vol. 3378, pp. 50–65. Springer, Heidelberg (2005). https://doi.org/10.1007/978-3-540-30576-7_4
30. Joye, M., Salehi, F.: Private yet efficient decision tree evaluation. In: Kerschbaum, F., Paraboschi, S. (eds.) DBSec 2018. LNCS, vol. 10980, pp. 243–259. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-95729-6_16
31. Kawachi, A., Tanaka, K., Xagawa, K.: Concurrently secure identification schemes based on the worst-case hardness of lattice problems. In: Pieprzyk, J. (ed.) ASIACRYPT 2008. LNCS, vol. 5350, pp. 372–389. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-89255-7_23

32. Kolesnikov, V., Sadeghi, A.-R., Schneider, T.: Improved garbled circuit building blocks and applications to auctions and computing minima. In: Garay, J.A., Miyaji, A., Otsuka, A. (eds.) CANS 2009. LNCS, vol. 5888, pp. 1–20. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-10433-6_1
33. Lenstra, A.K.: Key lengths. In: The Handbook of Information Security (2004)
34. Lin, H.-Y., Tzeng, W.-G.: An efficient solution to the millionaires' problem based on homomorphic encryption. In: Ioannidis, J., Keromytis, A., Yung, M. (eds.) ACNS 2005. LNCS, vol. 3531, pp. 456–466. Springer, Heidelberg (2005). https://doi.org/10.1007/11496137_31
35. Lyubashevsky, V., Peikert, C., Regev, O.: On ideal lattices and learning with errors over rings. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 1–23. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-13190-5_1
36. McKee, J., Pinch, R.: Further attacks on server-aided RSA cryptosystems (1998)
37. Peikert, C., Waters, B.: Lossy trapdoor functions and their applications. In: 40th ACM STOC, pp. 187–196 (2008)
38. Regev, O.: On lattices, learning with errors, random linear codes, and cryptography. In: 37th ACM STOC, pp. 84–93
39. Rivest, R.L., Shamir, A.: Efficient factoring based on partial information. In: Pichler, F. (ed.) EUROCRYPT 1985. LNCS, vol. 219, pp. 31–34. Springer, Heidelberg (1986). https://doi.org/10.1007/3-540-39805-8_3
40. Smart, N.P.: Algorithms, key size and protocols report, ECRYPT - CSA (2018). http://www.ecrypt.eu.org/csa/documents/D5.4-FinalAlgKeySizeProt.pdf
41. Veugen, T.: Encrypted integer division. In: 2010 IEEE International Workshop on Information Forensics and Security, pp. 1–6 (2010)
42. Veugen, T.: Improving the DGK comparison protocol. In: WIFS 2012, pp. 49–54 (2012)
43. Veugen, T.: Encrypted integer division and secure comparison. IJACT **3**(2), 166–180 (2014)
44. Wang, S., et al.: HEALER: homomorphic computation of exact logistic regression for secure rare disease variants analysis in GWAS. Bioinformatics **32**(2), 211–218 (2016)
45. Yao, A.C.-C.: Protocols for secure computations (extended abstract). In: 23rd FOCS, pp. 160–164. IEEE Computer Society Press, November 1982
46. Zhou, T., Yang, X., Liu, L., Zhang, W., Ding, Y.: Faster bootstrapping with multiple addends. Cryptology ePrint Archive, report 2017/735 (2017). http://eprint.iacr.org/2017/735