Correction of a Secure Comparison Protocol for Encrypted Integers in IEEE WIFS 2012 (Short Paper)

Baptiste Vinh Mau[1] and Koji Nuida[2,3(✉)]

[1] ESI Japan Ltd., Tokyo, Japan
bvinhmau[at]gmail.com
[2] Advanced Cryptography Research Group,
Information Technology Research Institute,
National Institute of Advanced Industrial Science and Technology (AIST),
Tokyo, Japan
k.nuida[at]aist.go.jp
[3] Japan Science and Technology Agency (JST) PRESTO Researcher, Tokyo, Japan

Abstract. In secure multi-party computation, one of the most useful and basic functionalities that have been realized over additive homomorphic encryption is secure comparison of two integers, where one party has encrypted integers to be compared while only the other party has the decryption key. In IEEE WIFS 2012, Veugen proposed an efficient protocol for this problem in the semi-honest model, which provides perfect security against the latter party. In this paper, we point out that the protocol by Veugen outputs an incorrect value in some cases, and then propose a way to fix the flaws with only slight overhead in efficiency. Our proposed correction is not straightforward, in the sense that it required an "outsourced" homomorphic multiplication protocol for two encrypted values, which was not needed in the original protocol.

1 Introduction

Secure multi-party computation is a cryptographic technology that enables two or more parties to jointly compute a function value with their local inputs while keeping the local input of each party secret against any other party. A recent trend of this area is to develop efficient protocols by cleverly combining several sub-protocols based on methodologies of different types according to advantages of each methodology. One of such sub-protocols, which was indeed used by the paper [1] in NDSS 2015, is a secure comparison protocol for two encrypted integers proposed in IEEE WIFS 2012 by Veugen [6] based on additive homomorphic encryption (HE). As explained below in detail, the protocol by Veugen has two versions, and the aim of the present paper is to point out that one version of Veugen's protocol has flaws (i.e., it sometimes outputs an incorrect value) and to propose a way to correct the original protocol.

In Veugen's protocols, a party Alice has two encrypted integers $x, y \geq 0$ to be compared, encrypted by an additive HE scheme with plaintext space $\mathbb{Z}/N\mathbb{Z}$.

S. Obana and K. Chida (Eds.): IWSEC 2017, LNCS 10418, pp. 181–191, 2017.
DOI: 10.1007/978-3-319-64200-0_11

The other party Bob has (and Alice does not have) the decryption key. The output of the protocol is a ciphertext, possessed by Alice only, of the bit indicating which of x and y is larger. As mentioned above, Veugen's protocol has two versions, namely Protocols 2 and 4 in [6]. The former ("statistical security" version) provides statistical security (against Bob), while the latter ("perfect security" version) provides *perfect* security with slight increase of executing costs. Moreover, the bit lengths of input integers for the former version must be significantly shorter (depending on the required security level) than the plaintext size $\log_2 N$; while the latter version has no such restriction. The protocol used in [1] is the statistical security version; and if we were able to apply the perfect security version instead, we would achieve enhanced security and would also remove the redundancy of the plaintext space in comparison to the sizes of input integers. However, in fact the perfect security version has flaws, as shown in this paper.

Our Results. In this paper we point out that the perfect security version of Veugen's comparison protocol for encrypted integers (Protocol 4 in [6]) sometimes outputs an incorrect value, and then propose a way to fix the flaws.

In Veugen's protocol, the difference $u := x - y$ of two input (encrypted) integers has absolute value $< 2^\ell$ (with parameter ℓ). To obtain the encrypted sign of u, Alice homomorphically adds a random r to encrypted u and sends it to Bob, and Bob decrypts it and obtains $z := u + r$. By writing $a \div 2^\ell := \lfloor a/2^\ell \rfloor$ and letting a bit χ be 1 if and only if $z \bmod 2^\ell \geq r \bmod 2^\ell$, it follows (since $|u| < 2^\ell$) that $\delta := z \div 2^\ell - r \div 2^\ell + \chi$ equals 1 if $u \geq 0$, and equals 0 if $u < 0$. Then Alice can obtain an encrypted δ since a ciphertext of χ is available by applying to local unencrypted inputs $r \bmod 2^\ell$ and $z \bmod 2^\ell$ a secure comparison protocol proposed by Damgård, Geisler, and Krøigaard in [2] (*DGK comparison protocol*).

In fact, there is an issue for this idea; the underlying HE scheme has plaintext space $\mathbb{Z}/N\mathbb{Z}$, therefore if the value of z "overflows" the range between 0 and $N-1$, then the process of taking the remainder of z modulo N in the plaintext space may affect the values of $z \div 2^\ell$ and $z \bmod 2^\ell$ in the computation of δ. The aforementioned disadvantages of the statistical security version against the perfect security version is actually caused by restricting the range of r to prevent the overflow. On the other hand, in the perfect security version, Veugen has introduced to the protocol an auxiliary (encrypted) bit d indicating whether such an overflow occurred, and then modified the internal DGK comparison protocol to consider the effect of the overflow to the value of $z \bmod 2^\ell$. However, the other effect of the overflow to the value of $z \div 2^\ell$ was not considered in [6]; this is the main source of the incorrectness which we point out in the paper.

To resolve the flaw, we had to precisely analyze the difference of the (encrypted) output value caused by the overflow, which should be homomorphically added to the original output value. However, our analysis found that the difference, say η, is available for Alice in an *encrypted* form only, while the difference should be added only when $d = 1$, which is also an *encrypted* bit from Alice's viewpoint. As the underlying encryption scheme is just an additive HE, this required us to use another sub-protocol (described in e.g., [4]), not needed

in the original protocol, that enables Alice to obtain a ciphertext of the product of two encrypted values d and η with the help of Bob having the decryption key. See Sect. 4 for details. We also show that our correction of the protocol does not compromise the security in the semi-honest model, and experimentally evaluate the practical efficiency of our corrected protocol (see Sect. 5).

2 Preliminaries

Homomorphic Encryption. In this paper, any homomorphic encryption (HE) scheme is an additive homomorphic public key encryption scheme with error-free decryption algorithm and is supposed to be semantically secure. We write a ciphertext of a plaintext m as $[[m]]$. An HE scheme also admits *homomorphic operations* to generate, from given ciphertexts $[[m_1]], [[m_2]]$, new ciphertexts $[[m_1]] \boxplus [[m_2]]$ and $[[m_1]] \boxminus [[m_2]]$ of $m_1 + m_2$ and $m_1 - m_2$, respectively; and from given ciphertext $[[m]]$ and integer r, a new ciphertext $r \boxdot [[m]]$ of rm.

An example of HE schemes is the Paillier cryptosystem [5] with plaintext space $\mathbb{Z}/N\mathbb{Z}$ where N is a very large composite integer. Another example was proposed by Damgård, Geisler, and Krøigaard in [2,3] (*DGK cryptosystem*). Its plaintext space is a finite *field* $\mathbb{F}_p = \mathbb{Z}/p\mathbb{Z}$ with a fairly large prime p (see [2,3] for further details). The DGK cryptosystem provides very efficient decryption by using a lookup table; while the use of a lookup table yields a limitation that the plaintext size cannot practically be as large as the Paillier cryptosystem.

The DGK Comparison Protocol. Damgård et al. [2] also proposed a secure comparison protocol for unencrypted integers based on an HE scheme (*DGK comparison protocol*). We write $\chi_{\mathsf{P}} = 1$ if a condition P is satisfied, or else $\chi_{\mathsf{P}} = 0$. In their protocol, two parties Alice and Bob have their local integer inputs $x \geq 0$ and $y \geq 0$, respectively, and Bob also has a secret key of the underlying HE scheme. The local outputs of Alice and Bob are bits δ_A, δ_B, respectively, satisfying $\delta_A \oplus \delta_B = \chi_{x \leq y}$ where $\oplus$ denotes XOR. The security model is the semi-honest model, hence the information on Alice's (respectively, Bob's) secret input is not leaked to Bob (respectively, Alice) while they are supposed to execute the protocol correctly. See [2] or Protocol 1 in [6] for details.

The authors of [2] constructed their protocol using the DGK cryptosystem. We note that, the requirement for the underlying HE scheme is that the plaintext space can handle roughly $\log_2(3\ell)$-bit integers, where ℓ is a parameter bounding the inputs as $x, y < 2^\ell$. The reason of using the DGK cryptosystem in [2] seems to be the flexibility of the plaintext size, in contrast to the Paillier cryptosystem where the plaintext size must correlate with the security parameter.

An Outsourced Multiplication. In this paper we also need an auxiliary protocol to enable *multiplication* of encrypted integers. Here Alice has two encrypted integers $[[x]], [[y]]$, while Bob has a secret key for the underlying HE scheme. Alice wants to obtain a ciphertext $[[x]] \boxtimes [[y]] := [[xy]]$ (without knowing x, y, or xy) by outsourcing to Bob the decryption and multiplication of decrypted integers, while Alice also wants to conceal x, y even from Bob.

In this paper, we adopt the following protocol in the semi-honest model, where the HE scheme has plaintext space $\mathbb{Z}/N\mathbb{Z}$: (i) Alice randomly chooses $a, b \leftarrow \{0, 1, \ldots, N-1\}$, computes $[[x']] \leftarrow [[x]] \boxplus [[a]]$ and $[[y']] \leftarrow [[y]] \boxplus [[b]]$, and sends $[[x']]$ and $[[y']]$ to Bob; (ii) Bob decrypts $[[x']]$ and $[[y']]$ to get $x' = x + a \bmod N$ and $y' = y + b \bmod N$, computes $[[x'y']]$ and sends $[[x'y']]$ to Alice; (iii) Alice outputs $[[x'y']] \boxminus \Big(b \boxdot [[x]]\Big) \boxminus \Big(a \boxdot [[y]]\Big) \boxminus [[ab]]$. We note that this protocol is not new and had already appeared in previous work such as [4].

3 Veugen's Comparison Protocol with Statistical Security

Before revisiting the perfect security version of Veugen's comparison protocol, in this section we first explain the statistical security version (Protocol 2 in [6]). We also mention some very slight errors in [6] for the sake of completeness.

Veugen's protocol used two HE schemes, one of which (called "outer scheme") is for encrypting inputs and outputs and the other (called "inner scheme") is for performing the DGK comparison protocol as a sub-routine. In [6], the Paillier and the DGK cryptosystems were the outer and the inner HE schemes, respectively. We write sk_{O} and $[[m]]_{\mathrm{O}}$ for secret keys and ciphertexts for the outer scheme, while sk and $[[m]]$ for the inner scheme. We suppose that the public keys are distributed in advance and hence are made implicit in the descriptions. For the two schemes, we use common symbols $\boxplus$, $\boxminus$, and $\boxdot$ for homomorphic operations.

For integers $a \geq 0$ and $b > 0$, we write $a \bmod b \in \{0, 1, \ldots, b-1\}$ to denote the remainder of a modulo b, and write $a \div b = \lfloor a/b \rfloor$; hence $a = (a \div b) \cdot b + (a \bmod b)$. We write $\chi_{\mathsf{P}} = 1$ if a condition P is satisfied, or else we define $\chi_{\mathsf{P}} = 0$.

Suppose that the outer HE scheme has plaintext space $\mathbb{Z}/N\mathbb{Z}$, identified naturally with $\{0, 1, \ldots, N-1\}$. Let $\ell > 0$ be an integer. Then, in this version of Veugen's protocol, Alice is supposed to have two encrypted integers $[[x]]_{\mathrm{O}}, [[y]]_{\mathrm{O}}$ with $0 \leq x < 2^\ell$ and $0 \leq y < 2^\ell$. Bob is supposed to have the two secret keys $\mathsf{sk}_{\mathrm{O}}, \mathsf{sk}$. The parameters have a constraint $\ell + 2 + \sigma < \log_2 N$, where σ is an integer parameter (e.g., $\sigma = 80$ as indicated in [6]) relevant to the security level. (We note that the constraint was written as $\ell + \sigma < \log_2 N$ in Protocol 2 in [6], which is in fact not enough as observed below.) The goal of the protocol is to let Alice obtain a ciphertext $[[\chi_{x \geq y}]]_{\mathrm{O}}$ of the bit $\chi_{x \geq y}$ while Bob obtains no output. (We note that the encrypted output bit was specified to be $\chi_{x \leq y}$ in Protocol 2 in [6], but the inequality '$\leq$' should be reversed as '$\geq$' as observed below.)

For the sake of completeness, we describe Veugen's protocol (Protocol 2 in [6]) in Fig. 1. We note that our description of the protocol basically coincides with [6] except for the two slight corrections mentioned in the previous paragraph and some other notational changes.

For the sake of intuitive understanding of the protocol, here we divide the plaintext space $\{0, 1, \ldots, N-1\}$ of the outer HE scheme into 0-th to $(N \div 2^\ell)$-th "buckets", where the i-th bucket (with $0 \leq i < (N \div 2^\ell)$) consists of the numbers $i \cdot 2^\ell + j$ for $0 \leq j < 2^\ell$, and the last $(N \div 2^\ell)$-th bucket consists of the remaining $N \bmod 2^\ell$ numbers from $(N \div 2^\ell) \cdot 2^\ell$ to $N - 1$. Then for a plaintext $m = (m \div 2^\ell) \cdot 2^\ell + (m \bmod 2^\ell)$, we call $m \div 2^\ell$ the "bucket number" of m and call

(Alice) Input: $[[x]]_{\mathsf{O}}$ and $[[y]]_{\mathsf{O}}$	Output: $[[\chi_{x\geq y}]]_{\mathsf{O}}$
(Bob) Input: sk_{O} and sk	Output: (none)

1. Alice chooses a random $(\ell+1+\sigma)$-bit number r, computes $[[z]]_{\mathsf{O}} \leftarrow [[x]]_{\mathsf{O}} \boxminus [[y]]_{\mathsf{O}} \boxplus [[2^{\ell}+r]]_{\mathsf{O}}$, and sends $[[z]]_{\mathsf{O}}$ to Bob.
2. Bob decrypts $[[z]]_{\mathsf{O}}$ to get $z = x-y+2^{\ell}+r \bmod N$, and computes $\beta \leftarrow z \bmod 2^{\ell}$.
3. Alice computes $\alpha \leftarrow r \bmod 2^{\ell}$.
4. Alice and Bob execute the DGK comparison protocol using the inner HE scheme with private inputs α and β, which results in local output bits δ_A and δ_B, respectively, with $\delta_A \oplus \delta_B = \chi_{\alpha\leq\beta}$.
5. Bob computes $z \div 2^{\ell}$, and sends $[[z \div 2^{\ell}]]_{\mathsf{O}}$ and $[[\delta_B]]_{\mathsf{O}}$ to Alice.
6. - If $\delta_A = 1$, then Alice sets $[[\gamma]]_{\mathsf{O}} \leftarrow [[\delta_B]]_{\mathsf{O}}$.
   - Otherwise, Alice sets $[[\gamma]]_{\mathsf{O}} \leftarrow [[1]]_{\mathsf{O}} \boxminus [[\delta_B]]_{\mathsf{O}}$.
7. Alice outputs $[[z \div 2^{\ell}]]_{\mathsf{O}} \boxminus \Big([[r \div 2^{\ell}]]_{\mathsf{O}} \boxplus [[\gamma]]_{\mathsf{O}}\Big)$.

Fig. 1. Veugen's comparison protocol with statistical security (here $0 \leq x < 2^{\ell}$, $0 \leq y < 2^{\ell}$, $\ell + 2 + \sigma < \log_2 N$, and the outer HE scheme has plaintext space $\mathbb{Z}/N\mathbb{Z}$)

$m \bmod 2^{\ell}$ the "relative position" of m inside the bucket. On the other hand, we note that the encrypted bit γ computed in Step 6 is equal to $\mathrm{NOT}(\delta_A \oplus \delta_B)$, therefore $\gamma = \chi_{\alpha>\beta}$ by the functionality of the DGK comparison protocol.

For Step 2 of the protocol, when $x - y$ (with $|x - y| < 2^{\ell}$) is added to $2^{\ell} + r$, either an "overflow" of the relative position inside the bucket (in case where $x > y$ and $(r \bmod 2^{\ell}) + (x - y) \geq 2^{\ell}$) or an "underflow" of the relative position inside the bucket (in case where $x < y$ and $(r \bmod 2^{\ell}) - (y - x) < 0$) may occur. On the other hand, since $0 \leq x - y + 2^{\ell} + r \leq 2^{\ell+2+\sigma} < N$, any overflow or underflow of the bucket number does not occur in the computation of $x-y+2^{\ell}+r$. In other words, in the present case, we always have (as integers) $z = x - y + 2^{\ell} + r \bmod N = x - y + 2^{\ell} + r$.

First, we consider the simplest case where any of an overflow or an underflow of the relative position has not occurred at Step 2. In this case, we have $\alpha > \beta$ if and only if $x-y < 0$ (or $x < y$), while we have $z \div 2^{\ell} = (2^{\ell}+r) \div 2^{\ell} = (r \div 2^{\ell})+1$. This implies that, Alice's output is a ciphertext of $(z \div 2^{\ell}) - ((r \div 2^{\ell}) + \gamma) = 1 - \gamma = 1 - \chi_{\alpha>\beta} = 1 - \chi_{x<y} = \chi_{x\geq y}$.

Secondly, we suppose that an overflow has occurred at Step 2, which implies that $x > y$ and hence Alice's output should be $[[1]]_{\mathsf{O}}$. Now the effect of the overflow of the relative position ensures that $z \div 2^{\ell}$ is incremented by 1 in comparison to the previous case and we always have $\alpha > \beta$ and $\gamma = 1$. Hence the plaintext for Alice's output becomes $2 - \gamma = 1$, as desired. Similarly, when an underflow has occurred at Step 2 (hence $x < y$ and Alice's output should be $[[0]]_{\mathsf{O}}$), the effect of the underflow of the relative position ensures that $z \div 2^{\ell}$ is decremented by 1 in comparison to the previous case and we always have $\alpha < \beta$ and $\gamma = 0$; hence the plaintext for Alice's output becomes $0 - \gamma = 0$, as desired.

4 Correcting Veugen's Protocol with Perfect Security

In this section, we revisit the perfect security version of Veugen's comparison protocol (Protocol 4 in [6]), show that the original protocol is not correct in the original form, and then propose a way to correct the flaws of the original protocol. We use notations similar to those in Sect. 3, and we write $(\nu_{\ell-1}\cdots\nu_1\nu_0)_2$ to indicate the binary expression of an integer $\nu=\sum_{i=0}^{\ell-1}\nu_i 2^i$.

Figure 2 describes the corrected version of the original protocol, where the boxed parts are the main differences from the original protocol in [6]. More precisely, these two protocols have the following differences:

- A common HE scheme is used as the outer and the inner HE schemes in our corrected protocol; while those were distinct schemes in [6].
- Alice in our corrected protocol is supposed to output $[[\chi_{x\geq y}]]$; while Alice was supposed to output $[[\chi_{x\leq y}]]_{\mathrm{O}}$ in [6].
- The ciphertexts $[[c_{-1}]]$ in Steps 4h, i, and j of our corrected protocol did not appear in [6].
- The random values r'_i in Step 4i of our corrected protocol are chosen from the whole of invertible elements of the plaintext space (to mask the values of c_i completely); while those were not chosen from the whole domain in [6].
- Step 7' of our corrected protocol is newly added, which did not exist in [6].
- In Step 7 of our corrected protocol, the ciphertext $[[d]]\boxtimes[[\eta]]$ is homomorphically added to the output ciphertext, which was not done in [6].

4.1 Example and Observation for Flaws of the Original Protocol

Here we use the notations in Fig. 2. We set $N=263$, $\ell=6$ (hence $2^\ell=64$), $x=5$, and $y=2$. We choose random numbers $r=200$ and $\delta_A=0$. Then the original protocol proceeds as follows: (Step 2) $z=(5-2+200+64) \bmod 263=4$ and $\beta=4 \bmod 64=4=(000100)_2$; (Step 3) $\alpha=200 \bmod 64=8=(001000)_2$; (Steps 4a, 4c) $d=1$; (Step 4d) $\alpha_i\oplus\beta_i=0,0,1,1,0,0$ for $i=5,4,3,2,1,0$; (Step 4e) $\tilde{\alpha}=(200-263) \bmod 64=1=(000001)_2$ and $w_i=0,0,0,1,0,-1$ for $i=5,4,3,2,1,0$; (Step 4f) $w_i\leftarrow 0,0,0,4,0,-1$ for $i=5,4,3,2,1,0$; (Step 4g) $s=1-2\cdot 0=1$; (Step 4h) $c_2=1+0+(0-0)\cdot 1-1+3\cdot(0+0+0)=0$ (other c_i's are omitted here, since these are now irrelevant); (Step 4j) $\delta_B=1$; (Step 6) $\gamma=1-1=0$. Finally, Alice outputs a ciphertext of $(z\div 2^\ell)-((r\div 2^\ell)+\gamma)=(4\div 64)-(200\div 64)=-3$. But the correct output should be a ciphertext of $\chi_{x\geq y}=1$. This shows that Veugen's original protocol is not correct.

In this example, r is close to the upper bound $N-1$, and an overflow of the bucket number has occurred in the computation of $x-y+2^\ell+r$ at Step 2. This makes the value of $z\div 2^\ell$ small ($\approx$0), while the value of $r\div 2^\ell$ is large ($\approx N\div 2^\ell$). On the other hand, the value γ at Step 6 is in $\{0,1\}$ by the construction. Therefore, the plaintext of Alice's output $(z\div 2^\ell)-(r\div 2^\ell)-\gamma$ cannot be in $\{0,1\}$ (regardless of whether the computation of γ works as intended by the author of [6]) though it must be a bit due to the expected functionality of the protocol. In other words, Step 7 of the original protocol does not appropriately concern the effect of an overflow (or underflow) of the bucket number.

(Alice) Input: $[[x]]$ and $[[y]]$ — Output: $\boxed{[[\chi_{x\geq y}]]}$

(Bob) Input: sk — Output: (none)

1. Alice chooses $r \leftarrow_R \{0, 1, \ldots, p-1\}$, computes $[[z]] \leftarrow [[x]] \boxminus [[y]] \boxplus [[2^\ell + r]]$, and sends $[[z]]$ to Bob.
2. Bob decrypts $[[z]]$ to get $z = x - y + 2^\ell + r \bmod p$, and computes $\beta = (\beta_{\ell-1} \cdots \beta_1 \beta_0)_2 \leftarrow z \bmod 2^\ell$.
3. Alice computes $\alpha = (\alpha_{\ell-1} \cdots \alpha_1 \alpha_0)_2 \leftarrow r \bmod 2^\ell$.

4a. Bob computes $d \leftarrow \chi_{z<(p-1)/2}$ and sends $[[d]]$ to Alice.

4b. For each $i = 0, 1, \ldots, \ell-1$, Bob sends $[[\beta_i]]$ to Alice.

4c. If $0 \leq r < (p-1)/2$, then Alice resets $[[d]] \leftarrow [[0]]$.

4d. For each $i = 0, 1, \ldots, \ell-1$:
- If $\alpha_i = 0$, then Alice computes $[[\alpha_i \oplus \beta_i]] \leftarrow [[\beta_i]]$.
- Otherwise, Alice computes $[[\alpha_i \oplus \beta_i]] \leftarrow [[1]] \boxminus [[\beta_i]]$.

4e. Alice computes $\widetilde{\alpha} = (\widetilde{\alpha}_{\ell-1} \cdots \widetilde{\alpha}_1 \widetilde{\alpha}_0)_2 \leftarrow (r - p) \bmod 2^\ell$, and for each $i = 0, 1, \ldots, \ell-1$:
- If $\alpha_i = \widetilde{\alpha}_i$, then Alice sets $[[w_i]] \leftarrow [[\alpha_i \oplus \beta_i]]$.
- Otherwise, Alice sets $[[w_i]] \leftarrow [[\alpha_i \oplus \beta_i]] \boxminus [[d]]$.

4f. For each $i = 0, 1, \ldots, \ell-1$, Alice modifies $[[w_i]]$ by $[[w_i]] \leftarrow 2^i \boxdot [[w_i]]$.

4g. Alice chooses $\delta_A \leftarrow_R \{0, 1\}$ and computes $s \leftarrow 1 - 2\delta_A$.

4h. For each $i = 0, 1, \ldots, \ell-1$, Alice computes

$$[[c_i]] \leftarrow [[s]] \boxplus [[\alpha_i]] \boxplus \Big((\widetilde{\alpha}_i - \alpha_i) \boxdot [[d]]\Big) \boxminus [[\beta_i]] \boxplus \Big(3 \boxdot \Big([[w_{i+1}]] \boxplus [[w_{i+2}]] \boxplus \cdots \boxplus [[w_{\ell-1}]]\Big)\Big).$$

$$\boxed{[[c_{-1}]] \leftarrow [[\delta_A]] \boxplus \Big(2 \boxdot \Big([[w_0]] \boxplus [[w_1]] \boxplus \cdots \boxplus [[w_{\ell-1}]]\Big)\Big).}$$

4i. For each $i = \boxed{-1}, 0, 1, \ldots, \ell-1$, Alice chooses $1 \leq r'_i \leq p-1$ uniformly at random, and modifies $[[c_i]]$ by $[[c_i]] \leftarrow r'_i \boxdot [[c_i]]$. Then Alice sends $\boxed{[[c_{-1}]]}, [[c_0]], \ldots, [[c_{\ell-1}]]$ to Bob in uniformly random order.

4j. Bob decrypts $\boxed{[[c_{-1}]]}, [[c_0]], \ldots, [[c_{\ell-1}]]$, and:
- If at least one of $\boxed{c_{-1}}, c_0, \ldots, c_{\ell-1}$ is 0, then Bob sets $\delta_B \leftarrow 1$.
- Otherwise, Bob sets $\delta_B \leftarrow 0$.

5. Bob computes $z \div 2^\ell$, and sends $[[z \div 2^\ell]]$ and $[[\delta_B]]$ to Alice.

6. - If $\delta_A = 1$, then Alice sets $[[\gamma]] \leftarrow [[\delta_B]]$.
- Otherwise, Alice sets $[[\gamma]] \leftarrow [[1]] \boxminus [[\delta_B]]$.

7'. $\boxed{\begin{array}{l}\text{- If } r < p - 2^\ell, \text{ then Alice sets } [[\eta]] \leftarrow [[(r \div 2^\ell) + 1]] \boxplus [[\gamma]].\\ \text{- Otherwise, Alice sets } [[\eta]] \leftarrow [[(r \div 2^\ell) - \big((2^\ell + r - p) \div 2^\ell\big) + 1]].\end{array}}$

7. Alice outputs $[[z \div 2^\ell]] \boxminus \Big([[r \div 2^\ell]] \boxplus [[\gamma]]\Big) \boxed{\boxplus ([[d]] \boxtimes [[\eta]])}$.

Fig. 2. Corrected version of Veugen's comparison protocol with perfect security (here $0 \leq x < 2^\ell$, $0 \leq y < 2^\ell$, $\ell + 2 < \log_2 p$, the HE scheme has plaintext space $\mathbb{F}_p = \mathbb{Z}/p\mathbb{Z}$ with prime p, and the symbol $\boxtimes$ indicates an outsourced homomorphic multiplication operation described in Sect. 2; the boxed parts are the main differences from the original protocol)

4.2 Revisiting the Main Part of the Original Protocol

We analyze the original protocol in detail. First, the bit d after Step 4c becomes 1 if and only if $z < (N-1)/2 \leq r$. Since $0 \leq x - y + 2^\ell \leq 2^{\ell+1} - 1 < N - 1$, the situation $z < (N-1)/2 \leq r$ above occurs if and only if $x - y + 2^\ell + r \geq N$, i.e., the overflow of the bucket number occurs. Hence, we have $d = 1$ if the overflow of the bucket number occurs, or else we have $d = 0$, as intended in [6].

In the rest of this subsection, we revisit Step 4 (i.e., Steps 4a–j) of the protocol, which is the most complicated part. As mentioned in [6], the part of the protocol is intended to be a modification of the DGK comparison protocol for handling the effect of the overflow of the bucket number.

For case $d = 0$. Now we have $w_i = 2^i \cdot (\alpha_i \oplus \beta_i)$ for $0 \leq i \leq \ell - 1$ after Step 4f, and for Step 4h, we have $c_i = s + \alpha_i - \beta_i + 3\sum_{j=i+1}^{\ell-1} w_j$. Now by the definition of w_j's, the term $3\sum_{j=i+1}^{\ell-1} w_j$ becomes 0 if $\alpha_j \oplus \beta_j = 0$ (or $\alpha_j = \beta_j$) for every $j = i+1, \ldots, \ell-1$, or else the term has absolute value ≥ 3. Since $|s + \alpha_i - \beta_i| \leq 2$, it follows that we have $c_i = 0$ if and only if $\alpha_j = \beta_j$ for all $j = i+1, \ldots, \ell-1$ and $s + \alpha_i - \beta_i = 0$. Moreover, since $s = 1 - 2\delta_A = \pm 1$, the condition $s + \alpha_i - \beta_i = 0$ above implies $\alpha_i \neq \beta_i$. Therefore, the only index i that can satisfy $c_i = 0$ is the largest index (if exists) at which position the bits of α and β differ.

Now, in the case $\alpha \neq \beta$, such an index i indeed exists uniquely, and we have $c_i = 0$ (hence $\delta_B = 1$) if and only if $(\delta_A, \alpha_i, \beta_i) = (0, 0, 1)$ (now $\alpha < \beta$ and $\gamma = 0$) or $(\delta_A, \alpha_i, \beta_i) = (1, 1, 0)$ (now $\alpha > \beta$ and $\gamma = 1$); while we have $c_i \neq 0$ (hence $\delta_B = 0$) if and only if $(\delta_A, \alpha_i, \beta_i) = (0, 1, 0)$ (now $\alpha > \beta$ and $\gamma = 1$) or $(\delta_A, \alpha_i, \beta_i) = (1, 0, 1)$ (now $\alpha < \beta$ and $\gamma = 0$). Summarizing, we have $\gamma = \chi_{\alpha > \beta}$.

In the other case $\alpha = \beta$, we always have $\delta_B = 0$ since such an index i as above does not exist, therefore $\gamma = 1 - \delta_A$. This is a uniformly random bit, which looks not an intended behavior. In fact, an idea to resolve the issue was already mentioned in Sect. 2.A of [6] (though not explicitly reflected in Protocol 4 in [6]). Following this idea, in addition to $[[c_i]]$ for $i = 0, \ldots, \ell - 1$ generated at Step 4h, we also use a ciphertext $[[c_{-1}]] \leftarrow [[\delta_A]] \boxplus \Big(2 \boxdot \big([[w_0]] \boxplus \cdots \boxplus [[w_{\ell-1}]]\big)\Big)$.

Now the condition $\alpha = \beta$ implies $w_j = 0$ for every $0 \leq j \leq \ell - 1$; therefore, if $\delta_A = 0$ then $c_{-1} = 0$ and $\delta_B = 1$, while if $\delta_A = 1$ then $c_{-1} \neq 0$ and $\delta_B = 0$. Hence $\gamma = 0 = \chi_{\alpha > \beta}$ in any case, which coincides with the behavior in the case $\alpha \neq \beta$ above. We note that this modification introducing $[[c_{-1}]]$ does not affect the correctness when ($d = 0$ and) $\alpha \neq \beta$, since now at least one of w_j's is non-zero and hence we always have $c_{-1} \neq 0$ by the definition.

Summarizing, when $d = 0$, the protocol satisfies $\gamma = \chi_{\alpha > \beta}$, assuming that we introduce the additional ciphertext $[[c_{-1}]]$ mentioned above.

For case $d = 1$. Now we have $w_i \in \{0, \pm 2^i\}$ and $c_i = 1 - 2\delta_A + \widetilde{\alpha}_i - \beta_i + 3\sum_{j=i+1}^{\ell-1} w_j$ for $0 \leq i \leq \ell - 1$, and $c_{-1} = \delta_A + 2\sum_{j=0}^{\ell-1} w_j$. On the other hand, the construction of Step 4e implies that we have $w_j = 0$ if and only if $\widetilde{\alpha}_j = \beta_j$ (regardless of the value of α_j). Hence, an argument similar to the case $d = 0$ implies the following. If $\widetilde{\alpha} = \beta$, then $c_i \neq 0$ for every $i = 0, 1, \ldots, \ell - 1$ and $c_{-1} = \delta_A$; therefore $\delta_B = 1 - \delta_A$ and $\gamma = 0 = \chi_{\widetilde{\alpha} > \beta}$. On the other hand, if $\widetilde{\alpha} \neq \beta$,

then the only index i (including -1) at which c_i can be zero is the largest index i with $\widetilde{\alpha}_i \neq \beta_i$, and for the index i, we have $1 - \delta_B = \chi_{c_i \neq 0} = \delta_A \oplus \chi_{\widetilde{\alpha} > \beta}$ and $\gamma = \chi_{\widetilde{\alpha} > \beta}$. Hence, when $d = 1$, the protocol satisfies $\gamma = \chi_{\widetilde{\alpha} > \beta}$ in any case.

Revisiting the Original Protocol Further. As discussed above, the protocol (when introducing $[[c_{-1}]]$) satisfies $\gamma = \chi_{\alpha > \beta}$ if $d = 0$, and $\gamma = \chi_{\widetilde{\alpha} > \beta}$ if $d = 1$. Now if $d = 0$, then we have $z = x - y + 2^\ell + r < N$ by the first paragraph of Sect. 4.2, while we have $\gamma = \chi_{\alpha > \beta}$ as above. This implies that, in this case the behavior of the protocol coincides with the statistical security version of Veugen's protocol; hence Alice's output is $[[\chi_{x \geq y}]]_{\mathrm{O}}$ as explained in Sect. 3.

For the other case $d = 1$, put $\rho := (z \div 2^\ell) - (r \div 2^\ell) - \gamma$, which is the plaintext for the output of the original protocol. Since $d = 1$ and $\ell + 2 < \log_2 N$, we have $N \leq x - y + 2^\ell + r < 2N$ by the argument in the first paragraph of Sect. 4.2, therefore $z = (x - y + 2^\ell + r) \bmod N = x - y + 2^\ell + r - N$ (as integers). The fact $\gamma = \chi_{\widetilde{\alpha} > \beta}$ mentioned above and the definition of $\widetilde{\alpha}$ suggest that, now the number $r - N$ instead of r would play a certain role in the protocol.

Put $\widetilde{r} := 2^\ell + r - N$. First, we consider the case $\widetilde{r} < 0$. Since $z = x - y + \widetilde{r} \geq 0$, we have $x > y$, therefore $\chi_{x \geq y} = 1$. Now $z < x - y < 2^\ell$ and $z \div 2^\ell = 0$, therefore $\rho = -(r \div 2^\ell) - \gamma$. Hence, the difference $\chi_{x \geq y} - \rho = (r \div 2^\ell) + \gamma + 1$ should be (homomorphically) added to the output of the original protocol.

Secondly, we consider the other case $\widetilde{r} \geq 0$. Put $\widetilde{z} := z + 2^\ell$. Then we have $0 \leq \widetilde{z} < x - y + 2^\ell + 2^\ell < 3 \cdot 2^\ell < N$ and $0 \leq \widetilde{r} < 2^\ell < N$ since $r - N < 0$ and $\ell + 2 < \log_2 N$. Moreover, we have $\widetilde{z} = x - y + 2^\ell + \widetilde{r}$, $\widetilde{z} \bmod 2^\ell = z \bmod 2^\ell = \beta$, and $\widetilde{r} \bmod 2^\ell = (r - N) \bmod 2^\ell = \widetilde{\alpha}$. Therefore, the same argument as the case $d = 0$ implies now that $\chi_{x \geq y} = (\widetilde{z} \div 2^\ell) - (\widetilde{r} \div 2^\ell) - \gamma$, which is equal to $(z \div 2^\ell) + 1 - (\widetilde{r} \div 2^\ell) - \gamma = \rho + (r \div 2^\ell) - (\widetilde{r} \div 2^\ell) + 1$. Hence, the difference $\chi_{x \geq y} - \rho = (r \div 2^\ell) - (\widetilde{r} \div 2^\ell) + 1$ should be (homomorphically) added to the output of the original protocol.

Summarizing, to obtain the correct (encrypted) output $\chi_{x \geq y}$, the value $d \cdot \eta$ should be (homomorphically) added to the output of the original protocol, where $\eta := (r \div 2^\ell) + \gamma + 1$ if $2^\ell + r - N < 0$ and $\eta := (r \div 2^\ell) - \big((2^\ell + r - N) \div 2^\ell\big) + 1$ if $2^\ell + r - N \geq 0$. The final issue remaining is how Alice computes the product of d and η, both being available only in an encrypted form. A basic strategy is to use an outsourced multiplication operation $\boxtimes$ described in Sect. 2, which would enable Alice to obtain a ciphertext $[[d \cdot \eta]]_{\mathrm{O}}$ and then perform an additive homomorphic operation with the original output ciphertext. However, in the original protocol, d is encrypted by the inner HE scheme rather than the outer HE scheme, therefore the operation $\boxtimes$ cannot be applied immediately. A simple way to resolve this issue is to use a common HE scheme as both the outer and the inner HE schemes. The resulting corrected protocol is described in Fig. 2.

5 Security and Performance Analyses

Theorem 1. *In our protocol, any object sent from Bob to Alice is a ciphertext of the underlying HE scheme, and any information sent from Alice to Bob is*

independent and uniformly random over the domain from which the information is chosen (provided the same properties hold for the outsourced multiplication).

Proof. The former part is obvious. For the latter claim, for Step 1, the plaintext $z = x - y + 2^\ell + r \bmod p$ sent (in encrypted form) to Bob is uniformly random over $\mathbb{F}_p$ as well as r. For Steps 4i and j, the argument in Sect. 4.2 implies that there are precisely the following two cases: (I) Exactly one of the values $c_{-1}, c_0, \ldots, c_{\ell-1}$ is zero; and $\delta_B = 1$. (II) All of the values $c_{-1}, c_0, \ldots, c_{\ell-1}$ are non-zero; and $\delta_B = 0$. Now the masking procedure in Step 4i ensures that Bob at Step 4j can only know which of (I) and (II) happens. Moreover, the argument in Sect. 4.2 also implies that, after the values of r, z, and d are determined, the bit γ at Step 6 is also determined (i.e., either $\chi_{\alpha>\beta}$ or $\chi_{\widetilde{\alpha}>\beta}$ depending on d), while we have $\delta_B = \gamma \oplus \delta_A$. Hence, from Bob's viewpoint, the possibilities of (I) and (II) are uniformly random and independent of z obtained at Step 1, since the uniformly random bit δ_A is not known by Bob. This completes the proof.

We implemented the offline version of our protocol and performed computer experiments for efficiency evaluation. We used C++ and the ZZ class of the NTL library (http://shoup.net/ntl/), and the benchmark was ran on a notebook with a Core N-5Y10c@0.80 GHz CPU. The underlying HE scheme is the (factoring-based) DGK cryptosystem with 1024-bit keys. Then our protocol to compare 8-bit, 12-bit, 16-bit, and 18-bit integers took 245 ms, 330 ms, 350 ms, and 399 ms running times and 3.25 kB, 4.25 kB, 5.25 kB, and 5.75 kB (estimated) bandwidth costs, respectively (larger plaintexts could not be handled in our environment due to the lookup tables in decryption). We remind that the unitary cost of each operation in the DGK cryptosystem is almost independent of the plaintext size; while the cost of the entire protocol does depend on the plaintext size, since the numbers of iterations in some loops and the numbers of communicated ciphertexts during the protocol are increased as the input size increases.

Acknowledgments. The authors thank Goichiro Hanaoka, Kana Shimizu and more generally all the members of Advanced Cryptography Research Group, Information Technology Research Institute from AIST for their supervision, help and collaboration. They also thank Jean-Pierre Hubaux from the EPFL Laboratory for Communications and Applications for assuming the role of supervising teacher of the first author's Master Thesis during which this paper was mainly produced. They also thank the anonymous reviewers for their comments. This work was supported by JST PRESTO Grant Number JPMJPR14E8, Japan.

References

1. Bost, R., Popa, R.A., Tu, S., Goldwasser, S.: Machine learning classification over encrypted data. In: NDSS 2015, California, USA, February 2015
2. 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). doi:10.1007/978-3-540-73458-1_30

3. Damgård, I., Geisler, M., Krøigaard, M.: A correction to 'efficient and secure comparison for on-line auctions'. Int. J. Appl. Cryptography **1**(4), 323–324 (2009)
4. Hallgren, P., Ochoa, M., Sabelfeld, A.: BetterTimes. In: Au, M.-H., Miyaji, A. (eds.) ProvSec 2015. LNCS, vol. 9451, pp. 291–309. Springer, Cham (2015). doi:10.1007/978-3-319-26059-4_16
5. 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). doi:10.1007/3-540-48910-X_16
6. Veugen, T.: Improving the DGK comparison protocol. In: Proceedings of IEEE WIFS 2012, pp. 49–54 (2012)