Threshold Properties of Prime Power Subgroups with Application to Secure Integer Comparisons

Rhys Carlton, Aleksander Essex, and Krzysztof Kapulkin

Western University, London, Canada
{rcarlton, aessex, kkapulki}@uwo.ca

Abstract. We present a semantically secure somewhat homomorphic public-key cryptosystem working in sub-groups of $\mathbb{Z}_n^*$ of prime power order. Our scheme introduces a novel threshold homomorphic property, which we use to build a two-party protocol for secure integer comparison. In contrast to related work which encrypts and acts on each bit of the input separately, our protocol compares multiple input bits simultaneously within a *single* ciphertext. Compared to the related protocol of Damgård et al. [8,9] we present results showing this approach to be both several times faster in computation and lower in communication complexity.

Keywords: Public-key encryption, homomorphic encryption, homomorphic threshold, secure integer comparison.

1 Introduction

Numerous solutions to the problem of secure integer comparison have been proposed going back to Yao's original solution to the *Millionaires* problem [23]. Although previous work has employed a variety of methods including oblivious transfers, garbled circuits and homomorphic encryption, the underlying approach has largely been to realize the comparison operation using a Boolean circuit acting in a bitwise fashion on the inputs.

In this paper we propose a new approach to secure integer comparison using a novel threshold scalar homomorphic property of subgroups of $\mathbb{Z}_n^*$ of prime power order. We construct a protocol efficiently comparing two encrypted integers through the (nearly) direct application of the homomorphism on a single encrypted value.

A one-sided homomorphic threshold function. Let t be a positive integer defining a threshold. In Section 4 we present a cryptosystem that introduces novel scalar homomorphism allowing two parties P_1, P_2 each with a message $m_1, m_2 \in \mathbb{N}$ to securely evaluate the following one-sided threshold function:

$$f_t(m_1, m_2) = \begin{cases} m_1 + m_2 & m_1 + m_2 < t \\ 0 & \text{otherwise.} \end{cases}$$

Throughout this paper we use the term *threshold* in the context of a cryptosystem which homomorphically computes this threshold function, while noting that the term *threshold homomorphic cryptosystem* is widely used in the literature to refer to the (unrelated) notion of a cryptosystem requiring a threshold of participants to decrypt a ciphertext such as e.g., the threshold cryptosystem of Schoenmakers and Tuyls [19].

Paper Organization. Related work is described in Section 2. Mathematical preliminaries of the construction are given in Section 3. The encryption scheme is presented in Section 4 and the secure comparison protocol is presented in Section 5. A security analysis of the protocol is given in Section 6 and Section 7 discusses performance of the implementation.

2 Related Work

Garbled circuits are the original construction solving the secure comparison problem [23]. The approach involves decomposing inputs into their bitwise representation and securely evaluating them in a Boolean circuit. Since that time numerous protocols have focused on improving performance and reducing communication cost [15] [14] [1] [5]. Recent advances in implementations of oblivious transfers [6] have made this approach quite computationally efficient in practice.

Another category of secure computation is the *arithmetic black box model* which seeks to abstract arithmetical operations into ideal reactive functionalities [20][17][24]. It departs at some level from the garbled circuit model by making invocations of the functionalities sublinear in the bit size of the inputs, but they remain superlinear in bit complexity and can have large constants affecting performance.

The third type of approach uses *homomorphic encryption*. Fishlin [10] first introduced this approach using a boolean circuit for a secure comparison of two numbers based on the semantically secure cryptosystem due to Goldwasser and Micali [12]. Other examples of secure Boolean evaluation of bit-wise encrypted values include the schemes of Blake and Kolesnikov [3], Garay et al. [11] and Lin et al. [16]. The approach was later improved by Damgård, Geisler, and Krøigaard (DGK) [8,9]. A slight improvement to their approach was made by Veugen [21,22] utilizing additional cryptosystems such as the one due to Paillier [18].

Homomorphic encryption based solutions are typically less computationally efficient than their garbled circuit counterparts. Nevertheless, homomorphic based comparison protocols can be more straightforward to implement, and can offer a lower overall communication cost, which is why new research into faster constructions remains important.

Related Cryptosystems in $\mathbb{Z}_n^*$. The cryptosystem used in the DGK comparison protocol is closely related to that of Groth [13]. DGK initially used a subgroup of $\mathbb{Z}_n^*$ of prime order dividing both $(p-1)$ and $(q-1)$, but a correction was made [9] when it was realized this value was leaked by the public key.

Groth's scheme suggested parameterizing the respective subgroups of p, and q such that their combined order was still large relative to the discrete logarithm problem, but individually smaller for efficiency. Coron et al. [7] showed an attack breaking semantic security below the expected attack complexity. Following this, Groth's scheme and DGK parameterize the randomization space identically, and have identical encryption functions, differing only in the message space: Groth fills up the remaining space of $\mathbb{Z}_n^*$ with smooth subgroups of unknown order to accommodate a large message space. DGK uses small message spaces similar to the cryptosystem of Benaloh [2], however the latter devotes the entirety of $\mathbb{Z}_n^*$ to being in the ciphertext space, which is highly efficient for encryption, but not nearly as efficient as DGK for decryption.

Most closely related to our cryptosystem is the system of Joye et al. [**?**,**?**], a generalization of the cryptosystem due to Goldwasser and Micali [12] in which the message space has order 2^k for $k \geq 1$. Their approach exploits the efficiency of computing 2^k-th power residue symbols given knowledge of the factorization of m, allowing fast decryption and higher bandwidth, i.e., a larger message space relative to public key length.

Our Cryptosystem. The cryptosystem presented in Section 4 differs from the above cryptosystems in two main ways. First is that we work with subgroups of $\mathbb{Z}_n^*$ of *prime power*, i.e., order b^d for a prime base b and exponent $d > 1$ (though our scheme is most similar to Joye et al. when $b = 2$). Second is that unlike the schemes above which encrypt a message m as $g^m h^r$, ciphertexts in our scheme take the form $g^{b^m} h^r$, which introduces a novel threshold homomorphic property outlined in Section 4.1. Using the homomorphic properties of this scheme, our protocol for secure comparisons departs from the standard approach of bit-wise encryption of inputs, instead performing the comparison on the *entire* value inside a single ciphertext. There are, of course, qualifications. One is if the input range is sufficiently large it may become more efficient to break the input into blocks (cf. Section 5.4). The other is that the threshold homomorphism of the proposed cryptosystem is one-sided, meaning the difference between the two messages is only hidden in the case where $m_1 > m_2$, and is revealed when $m_1 \leq m_2$. This property is useful and interesting in its own right, however additional components beyond the base cryptosystem (cf. Section 5) are required for two-sided (Millionaires) comparisons.

3 Preliminaries

Throughout the paper, we will work with an RSA modulus $n = p \cdot q$, where p and q are primes chosen in such a way that:

$$p = 2b^d p_s p_t + 1 \quad \text{and} \quad q = 2b^d q_s q_t + 1.$$

Here, b is a small prime base (e.g., 2), d is a positive integer greater than 1, and p_s, p_t, q_s, q_t are pairwise distinct primes. We note that

$$\begin{aligned}\mathbb{Z}_n^* &\cong \mathbb{Z}_{2b^d p_s p_t} \times \mathbb{Z}_{2q_s q_t} \\ &\cong (\mathbb{Z}_2)^2 \times (\mathbb{Z}_{b^d})^2 \times \mathbb{Z}_{p_s q_s} \times \mathbb{Z}_{p_t q_t}\end{aligned}$$

and hence $\mathbb{Z}_n^*$ has a cyclic subgroup $\mathbb{G}$ of order b^d and a unique (necessarily cyclic) subgroup $\mathbb{H}$ of order $p_s q_s$. Primes p_t, q_t are present to increase p and q to their required lengths.

Let $g \leftarrow \mathbb{G}$ and $h \leftarrow \mathbb{H}$ be random generators of their respective subgroups. The public key is then given by $\mathcal{PK} = (n, b, d, g, h, u)$, where u is the bit-length of both p_s and q_s. Let the notation $x \leftarrow_\$ S$ denote a value x sampled uniformly at random from a set S. To encrypt a message $0 \leq m < d-1$, one chooses a random $r \leftarrow_\$ \{1, \ldots, 2^u - 1\}$ and computes $c = g^{b^d} h^r \mod n$. To decrypt c, one first computes $c^{p_s q_s} = (g^{b^d})^{p_s q_s}$ by virtue of h having order $p_s q_s$. Further, let x denote the inverse of $p_s q_s$ in $\mathbb{Z}_{b^d}$. By raising $c^{p_s q_s}$ to the power of x, it suffices to solve the discrete logarithm problem:

$$g^{b^d} = (c^{p_s q_s})^x.$$

Since g is an element of order b^d this can be done in $O(d\sqrt{b})$ operations, which is efficient when b, d are small. It is clear that one can choose the numbers b, p_s, q_s, p_t, q_t at random (testing primality) in an efficient way. To complete the mathematical description of the scheme, we need to explain how to efficiently choose the generators g and h of the respective subgroups $\mathbb{G}$ and $\mathbb{H}$. Generator h is chosen in the same manner as the generators of the respective randomizer spaces of the schemes of Groth [13] and Damgård et al. [8], namely we find generator h_{p_s} (resp. h_{q_s}) of the subgroup of $\mathbb{Z}_p^*$ (resp. $\mathbb{Z}_q^*$) of order p_s (resp. q_s). The procedure for finding h_{p_s} and h_{q_s} is straightforward, and is found in most software implementations of the discrete logarithm problem over finite fields (e.g., Diffie-Helmman, DSA, Elgamal, etc). Next use the Chinese remainder theorem to find h such that

$$\begin{aligned}h &\equiv h_{p_s} \bmod p \\ h &\equiv h_{q_s} \bmod q.\end{aligned}$$

g is chosen in the same manner, however, importantly, because the order of g is public, it is necessary for security that it have identical order in $\mathbb{Z}_p^*$ and $\mathbb{Z}_q^*$. Therefore to find a generator g_{b^d} of a subgroup of order b^d separately in $\mathbb{Z}_p^*$ and $\mathbb{Z}_q^*$ and use the Chinese remainder theorem to compute g in the manner above.

Generator of a prime power subgroup. The procedure for finding a generator of prime power order is not commonly found in the literature, so we outline it here. To find a generator g_{b^d} of a subgroup of $\mathbb{Z}_p^*$ (for a prime p) of order b^d, it is sufficient to perform the following:

while True :
  $x \leftarrow_{\$} \{2 \ldots p-2\}$
  $y \leftarrow x^{(p-1)/b} \bmod p$
  if $y \neq 1$:
    return $x^{(p-1)/b^d}$.

This procedure is repeated to find a generator g_{b^d} of a subgroup of $\mathbb{Z}_q^*$ (for prime q), and these two generators are combined using the Chinese remainder theorem to produce g, a generator of a subgroup of order g_{b^d} of $\mathbb{Z}_n^*$ (where $n = pq$).

3.1 The Small RSA Subgroup Decision Assumption

We construct our hardness assumption to make the proof of semantic security for our system (cf. Theorem 2) as straightforward as possible. In brief, given the parameters as above, it should be infeasible to distinguish between a randomly selected quadratic residue mod n and an element of order $p_s q_s$ in $\mathbb{Z}_n^*$, without factoring n.

To make this intuition precise, we begin by extracting the essential information from our public key generation algorithm.

Definition 1. *An* RSA quintuple *is a quintuple (n, b, d, g, u) where:*

1. *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;*
2. *b is a prime of bit-length less than u;*
3. *d is an integer greater than 1;*
4. *n is an integer of the form $n = pq$, whose factorization is infeasible, where:*

$$p = 2b^d p_s p_t + 1 \quad \textit{and} \quad q = 2b^d q_s q_t + 1;$$

   and where in turn p_s and q_s are primes of bit-length u, and p_t, q_t are primes whose bit-length is not u;
5. *g is an element of order b^d in $\mathbb{Z}_n^*$.*

We point out that an RSA quintuple (n, b, d, g, u) is only one number short of a public key in our encryption scheme (Section 4). This is intentional in that we will use the final parameter to define the problem and the corresponding hardness assumption.

In particular, we note that the procedure for public key generation described earlier in this section can be used to generate an RSA quintuple by simply disregarding h.

Definition 2 (Small RSA Subgroup Decision Problem). *Given an RSA quintuple (n, b, d, g, u) and $x \in QR_n$, output* ‘yes’ *if x has order $p_s q_s$ and* ‘no’ *otherwise. (Here, we write QR_n for the set of quadratic residues mod n.)*

Note that due to the requirements on the length of p_s, q_s, p_t, and q_t, this gives a well-defined decision problem.

Of course, if we could factor n, then the problem would be easy to solve. However, in the other case, it appears to be infeasible, which leads us to the following definition:

Definition 3 (Small RSA Subgroup Decision Assumption). *Given an RSA quintuple (n, b, d, g, u) and $x \in \mathrm{QR}_n$, we say that $\mathcal{G}$ satisfies the Small RSA Subgroup Decision Assumption if for any polynomial time algorithm $\mathcal{A}$, the advantage of $\mathcal{A}$ in solving the Small RSA Subgroup Decision Problem is negligible.*

Our assumption (and naming convention) closely resembles that of Groth (cf. [13, Def. 2]), although it cannot be directly reduced. Indeed, in his assumption, Groth compares the distribution of the message space to the distribution of quadratic residues in $\mathbb{Z}_n^*$. This is not possible in our case, since the order of the message space is revealed as part of the public key and so instead we compare the distribution of the randomizer space (i.e., the unique subgroup of order $p_s q_s$) and the distribution of a random quadratic residue.

4 Encryption Scheme

We now describe the algorithms making up our encryption scheme. We define an algorithm $\mathcal{G}$ that, when given a security parameter $\tau \in \mathbb{Z}^+$, outputs a pair (ℓ, u) where ℓ defines a length for which the factorization of the product of two random ℓ-bit primes is computationally infeasible, and where u defines a length for which computing the discrete logarithm in a group of prime u-bit order is computationally infeasible.

$\mathsf{KGen}(\tau)$**:** Given security parameter $\tau > 0$, run $\mathcal{G}(\tau)$ to obtain (ℓ, u). Pick a small prime base b and message space upper bound $d \in \mathbb{Z}^+$. Let $n = pq$ for ℓ-bit primes p and q be constructed in the following manner:

$$p = 2b^d p_s p_t + 1$$
$$q = 2b^d q_s q_t + 1.$$

Let p_s, q_s be independently chosen random u-bit primes, and p_t, q_t be independently chosen random v-bit primes such that $b^d < \frac{1}{4}|n| - \tau$ (see Section 4). If $\lceil \log_2(b^d) \rceil + u \leq \ell$, let $v = \ell - (\lceil \log_2(b^d) \rceil + u)$. Otherwise if $\lceil \log_2(b^d) \rceil + u > \ell$, let $v = 0$ and set $p_t = q_t = 1$. Next let $\mathbb{G}$ be a subgroup of $\mathbb{Z}_n^*$ of order b^d, and $\mathbb{H}$ be a subgroup of $\mathbb{Z}_n^*$ of order $p_s q_s$. Pick a generator g of $\mathbb{G}$ such that g has order b^d in both $\mathbb{Z}_p^*$ and $\mathbb{Z}_q^*$ and pick a generator h of $\mathbb{H}$ such that h has order p_s in $\mathbb{Z}_p^*$ and q_s in $\mathbb{Z}_q^*$ (cf. Section 3). Finally let $x = p_s q_s x'$ where $x' = (p_s q_s)^{-1} \bmod b^d$.

The public key is $\mathcal{PK} = (n, b, d, g, h, u)$. The private key is $\mathcal{SK} = (x)$.

$\mathsf{Enc}(\mathcal{PK}, m)$**:** The message space consists of integers in the range $\{0 \ldots d-1\}$. To encrypt message m using public key $\mathcal{PK}$, pick random $r \leftarrow_{\$} \{1 \ldots 2^u - 1\}$ and compute

$$C = g^{b^m} h^r \bmod n.$$

Output ciphertext C.

$\mathsf{Dec}(\mathcal{SK}, C)$**:** To decrypt a ciphertext C using private key $\mathcal{SK}$, compute

$$(C)^x \bmod n = (g^{b^m} h^r)^{p_s q_s x'} = (g^{b^m})^{p_s q_s x'} (h^r)^{p_s q_s x'} = g^{b^m p_s q_s x'} = g^{b^m}.$$

If the result is 1, output $m = 0$. Otherwise recover m by computing $b^m = \log_g(g^{b^m} \bmod n)$ then $m = \log_b(b^m)$. Since the order of g is a power d of a small prime base b, this reduces to d computations of the discrete log in a cyclic group of order b. Since b is chosen to be small, this is efficiently computable.

Remark 1. In the special case $b = 2$, we can write p and q in the following form:

$$p = 2^d p_s p_t + 1,$$
$$q = 2^d q_s q_t + 1.$$

This case yields a cryptosystem similar to the system of Joye et al. [**?**,**?**] based on 2^d-th power residue symbols. Given knowledge of the factorization of n, d in fact can be recovered directly (i.e., without exponentiating away the h term) using the algorithm by Joye et al. (cf. Algorithm 1 of [**?**]), resulting in faster decryption.

Remark 2. By placing the message in the exponent of an exponent of g (i.e., a double exponent) and restricting the set of possible messages to $0 \leq m < d$, we obtain a cryptosystem with an interesting, and to our knowledge unexplored, homomorphic property which will next discuss next.

Bounding the length of b^d. Common factors dividing $(p-1)$ and $(q-1)$ have been used previously in related cryptosystems [**?**,**?**,8]. We consider appropriate upper bound for $|b^d|$ relative to $|p|$ and $|q|$. Recall $p = b^d p_s p_t + 1$ and $q = b^d q_s q_t + 1$ and thus $n = pq = b^{2d} p_s p_t q_s q_t + b^d(p_s p_t + q_s q_t) + 1$. Let $x = (n-1)/b^d = b^d p_s p_t q_s q_t + p_s p_t + q_s q_t$. A factorization method due to McKee and Pinch [**?**] can recover the factors of n in $O\left(\frac{n^{\frac{1}{4}}}{b^d}\right)$ operations using a baby-step giant-step approach. It is therefore necessary for security that

$$b^d < \frac{1}{4}|n| - \tau.$$

4.1 Homomorphic Properties

First we observe that in contrast to related schemes in $\mathbb{Z}_n^*$, our encryption scheme is *not* additively homomorphic:

$$\mathsf{Enc}(m_1) \cdot \mathsf{Enc}(m_2) = g^{b^{m_1}} h^{r_1} \cdot g^{b^{m_2}} h^{r_2} = g^{(b^{m_1}+b^{m_2})} h^{(r_1+r_2)}.$$

That is, multiplying ciphertexts in $\mathbb{Z}_n^*$ produces an exponent of g which may not be a power of b, which would not represent the encryption of a valid plaintext. Similar however to the scalar multiplicative homomorphism of related systems is the *scalar additive* homomorphism of our system:

$$\mathsf{Enc}(m_1)^{b^{m_2}} \bmod n = (g^{b^{m_1}} h^r)^{b^{m_2}} = g^{b^{m_1} b^{m_2}} h^{r'} = g^{b^{(m_1+m_2)}} h^{r'} = \mathsf{Enc}(m_1+m_2).$$

This gives rise to an interesting threshold homomorphic property:

$$\mathsf{Enc}(m_1)^{b^{m_2}} = \begin{cases} \mathsf{Enc}(m_1+m_2) & \text{if } m_1+m_2 < d \\ \mathsf{Enc}(0) & \text{otherwise.} \end{cases}$$

Theorem 1 (Homomorphic threshold function). *Let $f_d(m_1, m_2)$ be the threshold function outputting m_1+m_2 if $m_1+m_2 < d$, and outputting 0 otherwise. For $m_1, m_2 \in \mathbb{N}$, the scalar homomorphism above computes the encryption of f_d on m_1, m_2, i.e., $\mathsf{Enc}(m_1)^{b^{m_2}} = \mathsf{Enc}\big(f_d(m_1, m_2)\big)$.*

Proof. Since the order of g in $\mathbb{Z}_n^*$ was chosen to be b^d, then an exponent $x \in \mathbb{N}$ of g becomes $g^{x \bmod b^d} \bmod n$, thus $\mathsf{Enc}(m_1)^{b^{m_2}}$ can be written as $g^{b^{(m_1+m_2)} \bmod b^d} h^r$. If $m_1 + m_2 < d$, then $b^{m_1+m_2} \bmod b^d = b^{m_1+m_2}$. However if $m_1 + m_2 \geq d$, then $b^{m_1+m_2} = b^{d+a}$ for some $a \geq 0$. Since $b^d \equiv 0 \bmod b^d$, then $b^{d+a} = 0 \cdot b^a \equiv 0 \bmod b^d$. □

4.2 Semantic Security of Encryption

In this section, we prove the semantic security of our system.

Theorem 2. *The encryption scheme presented above is semantically secure, provided that the Composite Order Subgroup Decision Assumption of Definition 3 is satisfied.*

The proof is a straightforward application of the standard techniques (cf. e.g. [4, Thm. 3.1]), although we phrase it purely in terms of algorithms. Namely, we assume having an algorithm $\mathcal{A}'$ which breaks the semantic security of our encryption scheme with advantage $\varepsilon(\tau)$, which is non-negligible. Using it we will construct a polynomial time algorithm $\mathcal{A}$, which solves the Small RSA Subgroup Decision Problem.

The key piece of intuition here is that $\mathcal{A}$ is trying to decide whether an element x fits into a valid public key for our encryption scheme.

Proof. Suppose there exists a polynomial time algorithm $\mathcal{A}'$ breaking the semantic security of the above encryption scheme. Specifically, given a possibly invalid public key, $\mathcal{A}'$ produces two messages m_0 and m_1. If the key was valid, given a ciphertext c corresponding to one of them, it guesses correctly with probability $50\% + \varepsilon$ which message c is the encryption of. For an invalid key, $\mathcal{A}'$ chooses one of the messages at random.

Using $\mathcal{A}'$, we will construct a polynomial time algorithm $\mathcal{A}$ solving the Small RSA Subgroup Decision Problem. The algorithm $\mathcal{A}$ is given as input an RSA quintuple (n, b, d, g, u), and a quadratic residue $x \in \mathbb{Z}_n^*$. From these values we construct a (possibly invalid) public key (n, b, d, g, x, u) that can be given to $\mathcal{A}'$.

The algorithm $\mathcal{A}'$ responds by producing two plaintexts m_0 and m_1. We choose $i \leftarrow_{\$} \{0, 1\}$ and $r \leftarrow_{\$} \{1 \ldots 2^u - 1\}$, and compute the quantity $c \equiv g^{b^{m_i}} x^r \mod n$. Given this value, $\mathcal{A}'$ outputs $j \in \{0, 1\}$. Based on this information, we construct the output of $\mathcal{A}$ as follows:

$$\begin{cases} \mathsf{yes} & \text{if } i = j, \\ \mathsf{no} & \text{otherwise.} \end{cases}$$

If x is selected uniformly from the quadratic residues of $\mathbb{Z}_n^*$, then c is uniform in the appropriate coset of the subgroup generated by x. Thus as x varies, c varies uniformly as well, and so it is in particular independent of the choice of i. Thus the probability of $\mathcal{A}'$ guessing correctly is equal to 50%.

On the other hand, as stated above, $r < 2^u$ and hence crucially $r < p_s, q_s$. This gives $\mathcal{A}'$ an advantage, say ε, when x is an element of order $p_s q_s$, and this advantage is clearly seen to transfer to $\mathcal{A}$. □

5 Secure Comparison Protocol

In this section we present a protocol for the secure comparison of integers utilizing the encryption scheme presented in the previous section. As we have previously shown, the threshold homomorphic property of this scheme can be used to privately compute the encryption of the one-sided threshold function $f_d(m_1, m_2)$. This may be desirable for certain applications, however for a two-sided secure comparison protocol i.e., one that outputs the *single bit* $(m_1 \geq m_2)$, additional components are required since f_d outputs the sum $(m_1 + m_2)$ in the case where $m_1 < m_2$.

5.1 High-level Strategy

Our strategy involves using an additional (but different) cryptosystem. Cryptosystem CS_{f_d} is the cryptosystem with the threshold homomorphic property presented in Section 4. The additional cryptosystem $CS_{\oplus}$ is a generic semantically secure cryptosystem with an additive homomorphism. P_1 and P_2 hold the private keys to CS_{f_d} and $CS_{\oplus}$ respectively.

The idea is to use CS_{f_d} to compute the statement $(m_1 > m_2)$ using our homomorphic threshold approach to computing f_d by using the following inputs:

$$f_d(m_1, d - m_2) = \begin{cases} d + m_1 - m_2 & d + m_1 - m_2 < d \\ 0 & \text{otherwise.} \end{cases}$$

If $m_1 \geq m_2$, P_1 will receive the encryption of 0. Conversely if $m_1 < m_2$, P_1 will receive an encryption of their difference, which reveals information about P_2's input to P_1. To overcome this, P_2 will homomorphically add a blinding factor s to CS_{f_d} prior to P_1 decrypting. If $f_d(m_1, d - m_2) = 0$, the exponent recovered by P_1 during decryption will equal the blind factor s used by P_2, otherwise it will be $d + m_1 - m_2 + s$. The parties perform a *plaintext equality test* (PET) to privately determine whether or not these values are equal, and hence whether or not $m_1 \geq m_2$

5.2 Plaintext Equality Test Sub-protocol

Let $\mathsf{PET}(a_1, a_2)$ be a secure *plaintext equality test* conducted between two parties P_1, P_2 each of whom hold a private value a_1, a_2 respectively, and where P_2 holds the private key. The protocol accepts each party's private input and outputs 0 if $a_1 = a_2$, and outputs a random value otherwise.

Let $CS_\oplus = (\mathsf{Gen}_\oplus, \mathsf{Enc}_\oplus, \mathsf{Dec}_\oplus)$ be such a semantically secure additively homomorphic cryptosystem with a message space $\mathcal{M}_\oplus$ of large prime order. Without loss of generality, an efficient option for $CS_\oplus$ is the exponential variant of Elgamal (cf. e.g., [19]) implemented on a fast elliptic curve.

Plaintext Equality Test. The plaintext equality test $\mathsf{PET}_{CS_\oplus}$ proceeds in 3 steps:

1. P_2 computes the encryption $A_2 \leftarrow \mathsf{Enc}_\oplus(a_2)$ and sends it to P_1,
2. P_1 homomorphically computes the difference between a_1 and a_2 as $A_1 \leftarrow \mathsf{Enc}_2(-a_1) \cdot A_2$ and then blinds the plaintext result by computing $A_1' \leftarrow (A_1)^r$ for some uniform $r \neq 0$ in the message space $\mathcal{M}_\oplus$, then sends the result A_1' to P_2,
3. P_2 computes $m \leftarrow \mathsf{Dec}_\oplus(A_1')$ and outputs True if $m = 0$, and False otherwise.

This approach to plaintext equality testing is widely used so we only briefly recount its correctness and privacy. Regarding correctness, observe the result at the end of step 2 is the encryption $\mathsf{Enc}_\oplus\big(r(a_2 - a_1)\big)$, which is the encryption of 0 if $a_1 = a_2$, and a non-zero value otherwise. Briefly, P_2's privacy is guaranteed by the semantic security of $\mathsf{Enc}_\oplus$. P_1's privacy is guaranteed when $a_1 \neq a_2$ if two things things hold: (a) the difference is non zero, (b) the random factors are non-zero and (c) the message space has some prime order k. The former is true within the given case, and the latter two are true by definition, thus $(a_2 - a_1), r \in \mathbb{Z}_k^*$ are both generators of a cyclic group of order k, thus $r(a_1 - a_2)$ is uniform in $\mathbb{Z}_k^*$ if r is.

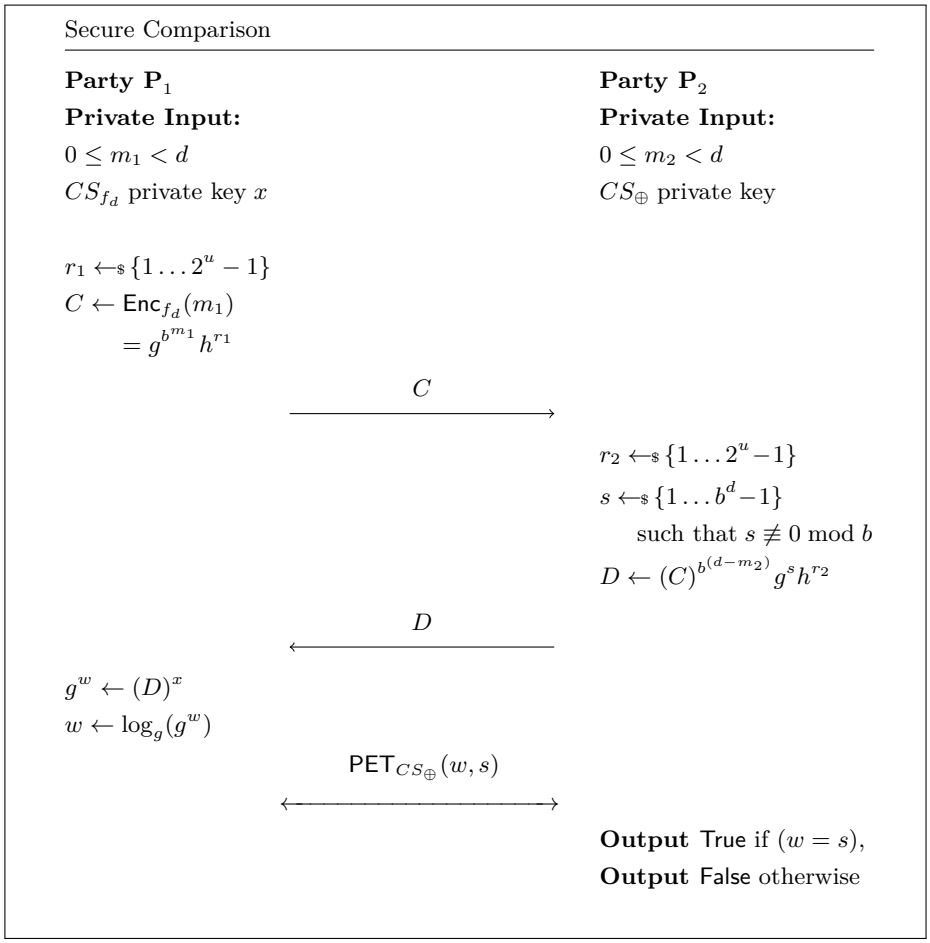

Fig. 1. Secure integer comparison protocol evaluating $(m_1 \geq m_2)$.

5.3 Secure Comparison protocol

We now present our secure integer comparison protocol in Figure 1. Correctness is shown below and security is proven in Section 6.

Theorem 3 (Correctness). *Two parties P_1, P_2 have private inputs $0 \leq m_1, m_2 < d$. The result of the protocol is Party P_2 outputs a single bit corresponding to $(m_1 \geq m_2)$, P_1 outputs nothing.*

Party P_1 begins by creating an encryption of b^{m_1} and sending to P_2 who then homomorphically computes

$$w = b^{m_1} b^{d-m_2} + s = b^{d+m_1-m_2} + s.$$

Case 1. If $m_1 \geq m_2$, then $(m_1 - m_2) \geq 0$ and thus we have b^{d+a} for some $a \geq 0$. By the homomorphic property presented in Section 4.1, $b^{d+a} = 0 \cdot b^a \equiv 0 \bmod b^d$ and thus $w = 0 + s = s$.
Case 2. If $m_1 < m_2$ then $(m_1 - m_2) < 0$ and thus we have b^{d-a} for some $0 < a < d$ and thus $b^{d+m_1-m_2} \equiv b^{m_1-m_2} \bmod b^d$. Thus $w = b^{m_1-m_2} + s \bmod b^d$.

Later P_1 decrypts and recovers w and performs a secure plaintext equality test with P_2 to privately test whether $w = s$. If the result of this test is True, then $w = s$ meaning $b^{d+m_1-m_2-1} \equiv 0 \bmod b^d$ and P_2 outputs True, i.e., $m_1 \geq m_2$. Conversely for $w \neq s$, P_2 outputs False, i.e., $m_1 < m_2$.

5.4 Extending to Arbitrary Length Comparisons with Blocking

Although it is possible to compare integers of arbitrary length using a single execution of the protocol in Figure 1, the asymptotic complexity is exponential in the bit length of the input numbers. Suppose we wish to compare two ℓ-bit numbers. Then we have $d \geq 2^\ell$. Thus we require a subgroup of $\mathbb{Z}_n^*$ of at least 2^ℓ bits (and more if $b > 2$), which implies a public key of $O(2^\ell)$ bits.

For efficiency reasons it would be helpful to consider fixed values for b and d, and extend the protocol to accommodate arbitrary input sizes by running multiple instances. The approach we take is to represent inputs in base d, and perform the comparison on each coefficient separately. This approach requires only a slight modification to the final plaintext equality testing phase.

Suppose we wish to compare two integers $0 \leq m_1, m_2 < 2^\ell$ where $2^\ell > d$. Let $k = \lceil \log_d(2^\ell) \rceil$. Rewrite integers m_1, m_2 in base d as follows:

$$m_1 = \alpha_{k-1} d^{k-1} + \alpha_{k-2} d^{k-2} + \cdots + \alpha_1 d + \alpha_0$$

and

$$m_2 = \beta_{k-1} d^{k-1} + \beta_{k-2} d^{k-2} + \cdots + \beta_1 d + \beta_0$$

for $0 \leq \alpha_i, \beta_i < d$. Next we observe that if $m_1 \geq m_2$ then exactly *one* of the following k Boolean expressions will be True:

$$(\alpha_{k-1} \geq \beta_{k-1})$$

or

$$(\alpha_{k-1} = \beta_{k-1}) \wedge (\alpha_{k-2} \geq \beta_{k-2})$$

or

$$(\alpha_{k-1} = \beta_{k-1}) \wedge (\alpha_{k-2} = \beta_{k-2}) \wedge (\alpha_{k-2} \geq \beta_{k-2})$$

or

$$\vdots$$

or

$$(\alpha_{k-1} = \beta_{k-1}) \wedge (\alpha_{k-2} = \beta_{k-2}) \wedge \cdots \wedge (\alpha_0 \geq \beta_0).$$

Conversely if $m_1 < m_2$, each of these expressions will be False. We can now apply this fact to securely evaluate $(m_1 \geq m_2)$ by running k instances of the protocol, and replacing the individual plaintext equality tests with each of the Boolean tests above. In the final pass P_1 sends the individual PET ciphertexts to P_2 in *shuffled* order. Then if one of the decryptions indicated a match, P_2 would not be able to tell which expression it was associated with—merely that a match had occurred, and thus $m_1 \geq m_2$.

6 Security

Our security model assumes a semi-honest (passive) adversary in a two-party computational setting. Parties follow the correct path through the protocol, but attempt to gain additional information about each other's inputs from messages exchanged during the protocol. We use a simulation based proof to demonstrate the protocol is secure given that the view of a participant in a real execution of the protocol is computationally indistinguishable from a simulated view given only that party's inputs and outputs. Below we define the semi-honest notion of simulation security.

Parties P_1 and P_2 interact in a protocol Π which computes the function of the protocol given the expected inputs and produces the expected outputs. Let F be a function defining the ideal functionality of the protocol Π, taking a pair of inputs $(\mathsf{in}_1, \mathsf{in}_2)$ to a pair of outputs $(\mathsf{out}_1, \mathsf{out}_2)$. The *view* of participant P_i (where $i = A, B$) will be denoted by $\mathbf{VIEW}^{\Pi}_{P_i}(\mathsf{in}_1, \mathsf{in}_2)$ and is defined as the information P_i observes and produces throughout the protocol. Let Sim_i be a simulator that takes in the inputs of party P_i and the ideal functionality of the protocol F and produces a transcript of the protocol. With this setup, we now give the definition of simulation security of a protocol.

Definition 4. *We say that a protocol Π is* secure against passive adversaries from the point of view of P_i *(for $i = A, B$) if there exists a probabilistic polynomial time simulator Sim_i for each party such that $\mathsf{Sim}_i(\mathsf{in}_i, F(\mathsf{in}_1, \mathsf{in}_2))$ is computationally indistinguishable from $(\mathbf{VIEW}^{\Pi}_{P_i}(\mathsf{in}_1, \mathsf{in}_2), \mathsf{out}_i)$.*

We say that a protocol Π is secure against passive adversaries *if it is secure from the point of view of both P_1 and P_2.*

Our goal in the remainder of this section is to prove that the comparison protocol of Figure 1, which throughout will be denoted Π, is secure against passive adversaries. We do so by proving security separately for P_1 and P_2.

In our case, the ideal functionality F is a function with the inputs (m_1, m_2) and output α (a binary indicator which results in *True* if $m_1 \geq m_2$ and *False* otherwise. It is clear that F defines the functionality of the protocol Π. When Π terminates, P_2 receives output of F. Let $\mathbf{OUTPUT}^{\Pi}(m_1, m_2)$ be the output received by P_2.

Lemma 1. *The protocol Π protects P_1's privacy.*

Proof. In order to show that P_2 does not learn anything about m_1 we will construct a valid simulator Sim_2 for P_2 with the property that

$$\mathsf{Sim}_2(m_2, CS_2 \text{ private key}, (m_1 \geq m_2)) \stackrel{c}{\equiv} \mathbf{VIEW}^{\Pi}_{P_2}(m_1, m_2).$$

Here, we write $\stackrel{c}{\equiv}$ for the relation of computational indistinguishability. The simulator Sim_2 is given m_1 and is able to simulate P_2's by first sampling a random value $C \leftarrow \mathbb{Z}_n^*$, sampling random values r, s and computing the ciphertext $D \leftarrow C^{m_2} \cdot g^s h^r$ and $\mathsf{Enc}'(s)$. To simulate the final PET ciphertext received from P_2, the simulator encrypts $\mathsf{Enc}'(0)$ if $(m_1 \geq m_2)$, otherwise samples a random non-zero value r from the plaintext space of Enc and computes $\mathsf{Enc}'(r)$ otherwise.

By the semantic security of CS_1, a polynomial-time algorithm cannot distinguish between C and a valid encryption of m_1. All other values are computable directly from C and the inputs given to Sim_2. □

Lemma 2. *The protocol Π protects P_2's privacy.*

Proof. Now we construct a simulator Sim_1 with the property that

$$\mathsf{Sim}_1(m_1, CS_1 \text{ private key},) \stackrel{c}{\equiv} (\mathbf{VIEW}^{\Pi}_{P_1}(m_1, m_2), \mathbf{OUTPUT}^{\Pi}(m_1, m_2)).$$

In the first step Sim_1 constructs $C \leftarrow \mathsf{Enc}(m_1)$ from CS_1 using its inputs. Next it constructs a CS_1 encryption $D \leftarrow \mathsf{Enc}(z)$ for $z \leftarrow_{\$} b^d$. It applies the private key of CS_1 to D to recover z. For the plaintext equality ciphertext received from P_2 it selects a random value in the ciphertext space of CS_2. For example if using Elgamal in a prime order group $\mathbb{G}$, it sends $E = \langle \alpha, \beta \rangle$ for $\alpha, \beta \leftarrow_{\$} \mathbb{G}$. Finally it computes the homomorphic difference between z and the encrypted plaintext in E, and blinds/re-randomizes using the public key of CS_2. By the semantic security of CS_2, E is a uniform value and therefore no polynomial-time algorithm has advantage distinguishing E from P_1's view of $\mathsf{Enc}'(s)$.

It only remains to show that the exponent recovered from the simulated ciphertext D, i.e., $z \leftarrow_{\$} b^d$ is computationally indistinguishable from P_1's real view of the recovered exponent w. First let us define the set $\mathcal{R} \subset \mathbb{Z}_{b^d}$ as the set of values $r \in \mathbb{Z}_{b^d}$ for which $r \not\equiv 0 \bmod b$. Let $s, z \leftarrow_{\$} \mathcal{R}$. P_1 decrypts D and recovers plaintext w, but cannot distinguish between a real-world value in which $w = b^{d+m_1-m_2-1} + s$ or a simulated value z. The latter case is a uniform value

in $\mathcal{R}$ by definition. To show the former case results in a uniform value in $\mathcal{R}$ it is sufficient to show first that $(b^{d+m_1-m_2-1}+s) \bmod b^d \in \mathcal{R}$ for all (m_1, m_2, s), and second that the result is uniform in $\mathcal{R}$.

First we note that $(b^{d+m_1-m_2}+s) \bmod b^d \in \mathcal{R}$ if $(b^{d+m_1-m_2}+s \bmod b^d) \bmod b \neq 0$. Since the inner and outer moduli share the same base we can reduce this to $(b^{d+m_1-m_2}+s) \bmod b \neq 0$. Next observe that $b^x \equiv 0 \bmod b$ for all x, and thus we are left only with the requirement that $s \bmod b \neq 0$, which is inherently satisfied from the definition of s. Therefore $(b^{d+m_1-m_2}+s) \in \mathcal{R}$. Second, since s is uniform in $\mathcal{R}$ then $b^{d+m_1-m_2}+s$ will be uniform in $\mathcal{R}$ as well. Therefore an algorithm cannot distinguish between real values of w and uniform values in $\mathcal{R}$ with advantage and thus cannot distinguish between a real ciphertext $g^{b^{d+m_1-m_2}+s}h^r$ and a simulated ciphertext $g^z h^r$. □

7 Performance Analysis

In this section we compare the performance of our protocol in Figure 1 against the 2-party secure integer comparison protocol of Damgård, Geisler, and Krøigaard (DGK) [8,9].

The primary difference between the respective approaches is that DGK performs its homomorphic operations on an element-wise encryption of the bitwise decomposition of the input integers, whereas our scheme performs the comparison inside a single encryption plus a plaintext equality test. This makes for an an interesting opportunity to compare the two approaches, since the bits of the plaintext space in our scheme grows linearly with the input size, whereas DGK uses logarithmically many ciphertexts with a logarithmic message space.

First let us consider messages in the range $0 \leq m < d$ and let $m = a_k 2^k + \cdots + a_1 k + a_0$ represent its binary decomposition. The DGK secure comparison protocol consists of $k = \lceil \log_2(d) \rceil$ ciphertexts encrypting the coefficients a_i as

$$\mathsf{Enc}_{\mathsf{DGK}}(a_i) = g^{a_i} h^r \bmod n$$

in which generator g has a small order corresponding to the next largest prime greater than $k+2$. Our scheme in its basic form consists of a single ciphertext which encrypts m directly as

$$\mathsf{Enc}_{f_d}(a_i) = g^{b^{a_i}} h^r \bmod n$$

in which generator g has a large order corresponding to b^d. As described in Section 5.4 we can extend the scheme to arbitrary bit lengths without resorting to linear growth in the modulus n by fixing b^d and performing multiple instantiations.

Eight bits for the price of one? For concreteness in this analysis we will set $b^d = 2^8 = 256$, and then compare a run of the DGK protocol involving 8 ciphertexts with an 8-bit message space against runs of our protocol involving a single ciphertext with a 256-bit message space of prime power order (plus a plaintext equality test). Messages of greater bit length, e.g., 16, 32, 64, etc, can

be achieved through 2, 4, and 8 etc. concurrent executions of our protocol with the modification to the PET outlined in Section 5.4.

Encryption and Re-randomization Cost. Notwithstanding the differences, from a performance standpoint the encryption operations are quite similar. Since the plaintext space is small, the main time consumer of time in DGK encryption comes in computing the random factor $h^r \bmod n$. We size these equivalently in both schemes (h has order $p_s q_s$ in ours, $v_p v_k$ in DGK). In both cases powers of g and h can be pre-computed, and computing h is equivalent in both schemes, however computing g is generally more costly in our scheme since we're encrypting one 256-bit value, as opposed to 8 single bit values. Since a message in our scheme consists of only 1 of 256 possible values, we can store these powers of g in a lookup table to make encryption faster.

Since re-randomization is modeled as the homomorphic addition with the encryption of 0, this operation too takes an identical amount of time in both schemes. Blinding the plaintext space, however, consists of a variable-base exponentiation which (short of what can be accomplished through addition chains) is not readily optimized, and takes longer in our scheme, give its comparably larger message space.

Decryption Cost. Damgård et al. [9] point out that decryption in their scheme can be efficiently performed in a short exponentiation modulo p (instead of $n = pq$):

$$C^{v_p} \bmod p = g^{m v_p} h^{v_p} = g^{m v_p}.$$

In the DGK protocol, decryption is only used to check if $C^{v_p} \equiv 1 \bmod p$, and thus if $m = 0$. Nominally decryption in our scheme is almost as fast, given the message space consists of only $d = 256$ possibilities for m. Our decryption scheme however must also account for the contribution of the secret exponent p_s in the plaintext space during decryption, i.e.,

$$C^{p_s} = g_p^{m p_s} h^{p_s} \equiv g^{m p_s} \bmod p.$$

In the description in Section 4 we used the factor x' to eliminate the p_s term in the exponent of g in which $x = p_s x' \equiv 1 \bmod b^d$. However this requires the receiever to perform a $\left(|p_s| + |b^d|\right)$-bit exponentiation. This can be made more efficient by insteald computing C^{p_s} and then computing the discrete logarithm to recover $m p_s$, and then computing $(m p_s)(p_s)^{-1} \bmod b^d$. Taking the discrete log is efficient for a small base such as $b = 2$. In our implementation below we use pre-computation to optimize taking the discrete log in the subgroup of order 2^{256} to approximately the cost of about one 256-bit fixed-base exponentiation.

Communication Complexity. In terms of round complexity DGK is a two-pass protocol: each party makes a single transmission. Our protocol is two-passes involving cryptosystem CS_{f_d} and two passes of $CS_{\oplus}$ in the PET sub-protocol. Both the CS_{f_d} and $CS_{\oplus}$ ciphertexts can be combined by P_2 into a single transmission, making the overall protocol 3 passes.

In terms of communication complexity our scheme operates on an 8-bit number in a single ciphertext, compared to DGK which employs 8 ciphertexs of an equivalent size. When using elliptic curves $CS_{\oplus}$ its contribution is relatively small. As an example, at the 128-bit security level each party in DGK transmits $24kb$ per comparison. In our scheme each party transmits 3.1kb—a reduction of 7.7 times, with an asymptotic trend towards 8x at higher security levels.

As a simplifying assumption we did not factor in the time cost of network transmission, though it would only impact performance in our favor given the significant difference in the total communication cost between the two protocols.

Cost of PET and $CS_{\oplus}$. Our protocol uses an additional cryptosystem $CS_{\oplus}$ to securely test for plaintext equivalence. The primary requirement of $CS_{\oplus}$ is that it be semantically secure, additively homomorphic and that the message space be of a large prime power. Many such schemes exists, providing us with a range of options.

In particular for performance we use exponential Elgamal and implement the group over a fast elliptic curve in order to minimize the cost of the $\mathsf{PET}_{CS_{\oplus}}$ sub-protocol relative to CS_{f_d} operating in $\mathbb{Z}_n^*$

Parameterizations. For cryptographic parameters we adhere to current NIST[1] minimum recommended guidelines on key lengths which prescribe bit lengths on the modulus and discrete logarithm groups. We note Groth [13] conjectured that since the order of the randomizer space of his cryptosystem is hidden, for performance reasons it may be possible to safely parameterize it to a size smaller than what would typically be required to make the discrete logarithm hard. Coron et al. [7] nonetheless found an attack on this approach essentially in $O(\sqrt{p_s})$ time and $O(\sqrt{p_s})$ space. Although the $O(\sqrt{p_s})$ space requirement makes the attack strictly worse than generic methods for solving a discrete logarithm (and in fact a significant real-world implementation challenge), we argue it would be inadvisable to go below minimum recommendations on discrete logarithm groups sizes. We parameterize the bit length u of p_s and q_s (and corresponding DGK randomizer space) accordingly. Working at the 128-bit security level requires $|n| = 3072$, $|p|, |q| = 1536$, $u = |p_s|, |q_s| = 256$, and $|p_t|, |q_t| = 1536 - 256 - \lceil \log_2(2^{256}) \rceil = 1024$. The 192-bit security level requires $|n| = 7680, |u| = 384$, and the 256-bit level requires $|n| = 15360, |u| = 512$.

For the implementation of $\mathsf{Enc}_{\oplus}$ we use Elgamal implemented over an elliptic curve. We considered the using the NIST curve `secp256r1`,[2] but chose the Edwards curve `Ed25519`[3] for performance. For the DGK implementation we use the analogous parameterizations. Using the notation of [9] we set $|n| = 3072$, $|p|, |q| = 1536$, randomizer space $|v_p|, |v_q| = 256$, and message space of order $u = 11$, which is the next prime up from $\log_2(d + 2)$.

[1] https://www.keylength.com/en/4/
[2] http://www.secg.org/SEC2-Ver-1.0.pdf
[3] https://ed25519.cr.yp.to/

Security level (bits)	Time (ms) DGK [8,9] Enc_{P_1}	Comp_{P_2}	Dec_{P_1}	Our protocol Section 4 Enc_{P_1}	Comp_{P_2}	PET_{P_1}	PET_{P_2}
128	1.04	1.19	0.46	0.13	0.37	0.26	0.01
192	6.02	6.56	3.08	0.81	1.70	0.95	0.01
256	22.6	23.3	12.8	2.84	5.22	2.88	0.01

Table 1. Amortized per bit cost of secure integer comparison protocols for respective operations.

Security level	Total Time (ms) DGK [8,9]	Our protocol	Speedup
128	2.7	0.8	3.5x
192	15.7	3.5	4.5x
256	58.7	10.9	5.4x

Table 2. Amortized total per bit cost.

Implementation. We implemented the DGK protocol [8,9] and our protocol from Figure 1 in Python using the `gmpy2` packages for optimized GMP-based integer operations. For $CS_{\oplus}$ we used `PyNaCl`, a Python binding to `libsodium`[4] which has an optimized implementation of curve Ed25519. The implementation of $CS_{\oplus}$ however was not complete since elliptic-curve based Elgamal requires point-additions, and most implementations of Ed25519 are focused on applications of ECDH and ECDSA and explicitly do not expose this low-level curve operation in their APIs. In terms of performance however, the contributions of point additions are minor relative to point multiplications, which in turn are minor relative to operations in CS_{f_d}. In each case we use optimizations such as pre-computation of fixed-base exponents and working $\bmod p$ instead of $\bmod n$.

Performance results. We benchmarked on an Intel Xeon E5-2697A @ 2.60GHz using a single-threaded instance of each protocol. We ran each protocol 1000 times using random 8-bit numbers and recorded the online computation time (i.e., excluding building lookup tables). We present our performance results in Table 1 amortized to the per-bit cost of each operation and show in Table 2 that our scheme has a per-bit comparison up to 5 times faster than DGK.

The reason our scheme becomes relatively faster at higher security levels can be attributed to two factors. One is that the arithmetic operations in the RSA setting grow faster than their elliptic curve counterparts, diminishing the relative contribution of the plaintext equality test. The other is that the order of g is

[4] https://github.com/jedisct1/libsodium

fixed at 256 bits making operations in this subgroup (comparing, blinding, etc) contribute to the total in decreasing amount relative to operations in h which grows at successive security levels.

8 Conclusion

Even after all these years, cryptosystems in $\mathbb{Z}_n^*$ continue to surprise us with new properties. In this paper we presented a new cryptosystem working in sub-groups of prime power order leading to a novel threshold homomorphic property. We exploited this property toward a public-key based secure integer comparison protocol that can perform the entire comparison in a single ciphertext faster than the conventional approach of using bitwise decompositions.

References

1. B. Applebaum, Y. Ishai, E. Kushilevitz, and B. Waters. Encoding functions with constant online rate, or how to compress garbled circuit keys. *SIAM Journal on Computing*, 44(2):433–466, 2015.
2. J. Benaloh. Dense probabilistic encryption. In *Workshop on Selected Areas of Cryptography*, 1994.
3. I. F. Blake and V. Kolesnikov. Conditional encrypted mapping and comparing encrypted numbers. In *International Conference on Financial Cryptography and Data Security*, pages 206–220. Springer, 2006.
4. D. Boneh, E.-J. Goh, and K. Nissim. Evaluating 2-DNF formulas on ciphertexts. In *Theory of cryptography*, volume 3378 of *Lecture Notes in Comput. Sci.*, pages 325–341. Springer, Berlin, 2005.
5. R. Bost, R. A. Popa, S. Tu, and S. Goldwasser. Machine learning classification over encrypted data. In *NDSS*, 2015.
6. T. Chou and C. Orlandi. *The Simplest Protocol for Oblivious Transfer*, pages 40–58. Springer International Publishing, Cham, 2015.
7. J.-S. Coron, A. Joux, A. Mandal, D. Naccache, and M. Tibouchi. Cryptanalysis of the rsa subgroup assumption from tcc 2005. In *14th International Conference on Practice and Theory in Public Key Cryptography (PKC)*, pages 147–155, 2011.
8. I. Damgård, M. Geisler, and M. Krøigaard. Efficient and secure comparison for online auctions. In *Information Security and Privacy: 12th Australasian Conference, ACISP 2007, Townsville, Australia, July 2-4, 2007. Proceedings*, pages 416–430, 2007.
9. I. Damgård, M. Geisler, and M. Krøigaard. A correction to "efficient and secure comparison for online auctions". *Int. J. Appl. Cryptol.*, 1(4):323–324, 2009.
10. M. Fischlin. A cost-effective pay-per-multiplication comparison method for millionaires. In *Cryptographers' Track at the RSA Conference*, pages 457–471. Springer, 2001.
11. J. Garay, B. Schoenmakers, and J. Villegas. Practical and secure solutions for integer comparison. In *International Workshop on Public Key Cryptography*, pages 330–342. Springer, 2007.
12. S. Goldwasser and S. Micali. Probabilistic encryption & how to play mental poker keeping secret all partial information. In *Proceedings of the Fourteenth Annual ACM Symposium on Theory of Computing*, STOC '82, pages 365–377, 1982.

13. J. Groth. Cryptography in subgroups of z_n^*. In *Theory of Cryptography: Second Theory of Cryptography Conference, TCC 2005, Cambridge, MA, USA, February 10-12, 2005. Proceedings*, pages 50–65, 2005.
14. V. Kolesnikov, A.-R. Sadeghi, and T. Schneider. How to combine homomorphic encryption and garbled circuits. *Signal Processing in the Encrypted Domain*, 100:2009, 2009.
15. V. Kolesnikov, A.-R. Sadeghi, and T. Schneider. Improved garbled circuit building blocks and applications to auctions and computing minima. *Cryptology and Network Security*, pages 1–20, 2009.
16. H.-Y. Lin, W.-G. Tzeng, et al. An efficient solution to the millionaires' problem based on homomorphic encryption. In *ACNS*, volume 5, pages 456–466. Springer, 2005.
17. H. Lipmaa and T. Toft. Secure equality and greater-than tests with sublinear online complexity. In *International Colloquium on Automata, Languages, and Programming*, pages 645–656. Springer, 2013.
18. P. Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *Advances in Cryptology - EUROCRYPT 1999*, pages 223–238. Springer-Verlag, 1999.
19. B. Schoenmakers and P. Tuyls. *Practical Two-Party Computation Based on the Conditional Gate*, pages 119–136. Springer Berlin Heidelberg, 2004.
20. T. Toft. Sub-linear, secure comparison with two non-colluding parties. In *Public Key Cryptography*, volume 6571, pages 174–191. Springer, 2011.
21. T. Veugen. Improving the dgk comparison protocol. In *Information Forensics and Security (WIFS), 2012 IEEE International Workshop on*, pages 49–54. IEEE, 2012.
22. T. Veugen. Encrypted integer division and secure comparison. *Int. J. Appl. Cryptol.*, 3(2):166–180, 2014.
23. A. C.-C. Yao. How to generate and exchange secrets. In *27th FOCS*, pages 162–167. IEEE Computer Society Press, 1986.
24. C.-H. Yu and B.-Y. Yang. *Probabilistically Correct Secure Arithmetic Computation for Modular Conversion, Zero Test, Comparison, MOD and Exponentiation*, pages 426–444. 2012.