Verifiable Rotation of Homomorphic Encryptions

Sebastiaan de Hoogh, Berry Schoenmakers, Boris Škorić, and José Villegas

Dept. of Mathematics and Computer Science, TU Eindhoven
P.O. Box 513, 5600 MB Eindhoven, The Netherlands
s.j.a.d.hoogh@tue.nl,berry@win.tue.nl,b.skoric@tue.nl,j.a.villegas@tue.nl

Abstract. Similar to verifiable shuffling (mixing), we consider the problem of verifiable rotating a given list of homomorphic encryptions. The offset by which the list is rotated (cyclic shift) should remain hidden. Basically, we will present zero-knowledge proofs of knowledge of a rotation offset and re-encryption exponents, which define how the input list is transformed into the output list. We also briefly address various applications of verifiable rotation, ranging from 'fragile mixing' as introduced by Reiter and Wang at CCS'04 to applications in protocols for secure multiparty computation and voting.

We present two new, efficient protocols. Our first protocol is quite elegant and involves the use of the Discrete Fourier Transform (as well as the Fast Fourier Transform algorithm), and works under some reasonable conditions. We believe that this is the first time that Fourier Transforms are used to construct an efficient zero-knowledge proof of knowledge.

Our second protocol is more general (requiring no further conditions) and only slightly less efficient than the DFT-based protocol. Unlike the previously best protocol by Reiter and Wang, however, which relies on extensive use of verifiable shuffling as a building block (invoking it *four* times as a sub-protocol), our construction is direct and its performance is comparable to the performance of a *single* run of the best protocol for verifiable shuffling.

1 Introduction

The well-known problem of verifiable shuffling (or, mixing) is to transform a given list of homomorphic encryptions into a list of randomly permuted, random re-encryptions of these encryptions, such that (i) it can be verified that the multiset of plaintexts for the input list and output list are identical, and (ii) the permutation used remains hidden. The original idea of mixing was introduced by Chaum, along with applications in anonymous email and voting [3], and the explicit goal of verifiability was introduced by Sako and Kilian, as part of a paper on voting [24]. Many improved protocols for verifiable shufflers/mixers have been published since, as well as many applications. A basic property is that a cascade of verifiable shufflers is again a verifiable shuffler, which enables mutually distrusting parties to take turns in permuting a given list such that no party knows the permutation for the final output list (unless all parties collude).

In this paper we consider the related problem of verifiable rotating a given list of homomorphic encryptions. Rather than requiring the use of a random permutation, as in shuffling, we require that a random rotation $\pi_r(k) = k - r \pmod{n}$, where the offset r, $0 \leq r < n$, is chosen uniformly at random. Clearly, a cascade of verifiable rotators is also a verifiable rotator, for which no party knows by which offset the final output list has been rotated (unless all parties collude).

Verifiable rotation has actually been introduced by Reiter and Wang in the context of mixing [21]. They define 'fragile mixing' as a form of mixing that deters leaking of information: namely, when a single correspondence between an element on the input list and an element of the output list is revealed, then the correspondence between all elements of the input list and output lists is revealed. A fragile mix is therefore restricted to the use of rotations (called "loop permutations" in [21]). The protocol by Reiter and Wang, however, uses *four* invocations of a verifiable shuffle protocol (and some further work) to perform a verifiable rotation. We will reduce this to the work of about one verifiable shuffle, by following a completely different, more direct approach to the problem.

Apart from fragile mixing, however, there are many more applications of verifiable rotations. An important application arises in the context of secure integer comparisons, as noted first in [20]. A common step in many integer comparison protocols [2, 1, 10, 8, 20], requires parties to find out whether a special value occurs in a given list of encryptions. For example, whether there is a 0 among otherwise random values. The position of the special value should remain hidden. To this end, the list will be randomly permuted before decrypting the encryptions. However, rather than using a fully random permutation, as commonly proposed, a random rotation suffices to hide the position of the special value.

Similarly, it is easily seen that for protocols such as Mix & Match [16], which involve mixing of truth tables of Boolean gates, it suffices to apply a random rotation to the rows of a truth table, rather than a fully random permutation. The reason is that in the matching stage exactly one row will match, and a random rotation fully hides the corresponding row of the original truth table. The same observation applies to other forms of 'garbled circuit evaluation', which can be seen as variations on Yao's original method [27]. Likewise, in protocols for secure linear programming [17] the position of the pivot in each iteration of the simplex algorithm must be hidden to avoid leakage of information. Again, we note that the use of a rotation instead of a general permutation suffices.

Finally, we note that further applications exist in the context of electronic voting, where randomly rotated lists of encryptions are used in the construction of encrypted ballot forms (see, e.g., Prêt-à-Voter voting systems by Ryan et al. [23, 22] and references therein): voters get a receipt in which one out of n positions is marked; due to a random rotation, the marked position does not reveal the identity of the candidate chosen. Recently, [26] presented a secure protocol for determining the winner of an election in a preferential electoral system. The goal at some intermediate point is to hide the position of a distinguished value in every row of a matrix of encrypted values. They achieve that by first rotating the row vectors in the clear, and later the rotation offsets are concealed by using verifiable mixes of each of the column vectors. The use of verifiable rotation would provide an alternative to the use of shuffling steps.

The goal of this paper is thus to design efficient protocols for verifiable rotations. Given a list $X_0, X_1, \ldots, X_{n-1}$ of (homomorphic) encryptions, a rotation is performed by picking an offset r, $0 \leq r < n$, at random, and creating a list of encryptions $Y_0, Y_1, \ldots, Y_{n-1}$, where each Y_k is a random re-encryption of X_{k-r}.[1] As the rotation offset r remains hidden if the underlying cryptosystem is semantically secure, a zero-knowledge proof is used to prove the correctness of a rotation. By using a non-interactive zero-knowledge proof, one obtains a verifiable rotation.

1.1 Our Contributions

We develop two new, efficient protocols for proving the correctness of a rotated list of homomorphic encryptions. These protocols allow for a direct construction of a cascade of verifiable rotators, which can be used to efficiently implement fragile mixing, among other applications, as highlighted above. Whereas the verifiable rotation protocol of [21] required the work of at least four verifiable shuffles, the work of our protocols is approximately equal to that of a single shuffle only—in some cases even slightly better than the currently best protocol for verifiable shuffling, due to Groth [12]. Note that *a priori* it is not clear whether verifiable shuffling is harder than verifiable rotation, or the other way around.

Our first protocol is a Σ-protocol and makes essential use of the Discrete Fourier Transform (DFT). To the best of our knowledge this is the first time that the DFT is actually used in the construction of an efficient zero-knowledge proof. The Σ-protocol turns out to be competitive with the best protocols for shuffling known to date. However, our DFT-based protocol relies on some constraints on the parameters, which should be met in order for the DFT to be applicable. For instance, in case we use ElGamal encryptions for a group of prime order q, we assume that $n \mid q-1$, where n denotes the length of the list of rotated encryptions. This ensures the existence of an n-th root of unity modulo q. Furthermore, to take full advantage of the Fast Fourier Transform (FFT) algorithm, we assume that n is an integral power of two.[2]

Our second protocol is more general and does not put any constraints on the parameters. Although it is not a (3-move) Σ-protocol, it is a 6-move honest-verifier zero-knowledge protocol for which we are able to prove knowledge soundness as well (more precisely, witness extended emulation). This general protocol is computationally only slightly more expensive than the DFT-based Σ-protocol.

[1] Throughout the paper, indices are reduced modulo n. E.g., X_{k-r} is short for $X_{k-r \pmod{n}}$

[2] These constraints can be relaxed, for instance, by using the DFT for a quadratic extension of $\mathbb{F}_q$, which exists provided $n \mid q^2 - 1$. Similarly, for the FFT it suffices if all prime factors of n are small, rather than requiring that n is a power of two. We do not consider these relaxations in this paper.

Both protocols can be made non-interactive using the Fiat-Shamir heuristic, yielding efficient (publicly) verifiable non-interactive zero-knowledge proofs, secure in the random oracle model.

2 Preliminaries

We present our protocols assuming homomorphic ElGamal as the underlying cryptosystem. However, our results can be readily extended to other homomorphic cryptosystems, such as the Paillier cryptosystem.

2.1 Discrete Log Setting

Let $G = \langle g \rangle$ denote a finite cyclic (multiplicative) group of prime order q for which the Decision Diffie-Hellman (DDH) problem is assumed to be infeasible.

Homomorphic ElGamal Cryptosystem. For public key $h \in G$, a message $m \in \mathbb{Z}_q$ is encrypted as a pair $(a, b) = (g^r, g^m h^r)$ for $r \in_R \mathbb{Z}_q$. Given the private key $\alpha = \log_g h$, decryption of (a, b) is performed by calculating $b/a^\alpha = g^m$ and then solving for $m \in \mathbb{Z}_q$. As usual, it is assumed that m belongs to a sufficiently small subset of $\mathbb{Z}_q$ to make recovery of m feasible.

This encryption scheme is additively homomorphic: given $(a_1, b_1) = (g^{r_1}, g^{m_1} h^{r_1})$ and $(a_2, b_2) = (g^{r_2}, g^{m_2} h^{r_2})$, an encryption of $m_1 + m_2$ is obtained by pairwise multiplication $(a_1, b_1)(a_2, b_2) = (a_1 a_2, b_1 b_2) = (g^{r_1+r_2}, g^{m_1+m_2} h^{r_1+r_2})$. Homomorphic ElGamal is semantically secure under the DDH assumption.

As a shorthand notation for an encryption $(g^r, g^m h^r)$, we write $\mathsf{E}(m, r)$. Moreover, $\mathsf{E}(m)$ will denote an ElGamal encryption of $m \in \mathbb{Z}_q$, where the randomization is suppressed from the notation.

Pedersen Commitment. Given $\widetilde{h} \in G$, a Pedersen commitment to $m \in \mathbb{Z}_q$ is the value $b = g^m \widetilde{h}^r$ where $r \in \mathbb{Z}_q$. This commitment is opened by revealing m and r. The scheme is unconditionally hiding and computationally binding assuming that $\log_g \widetilde{h}$ cannot be computed. We use $\mathsf{C}(m, r)$ to denote a Pedersen commitment to m using randomness r, and abbreviate this to $\mathsf{C}(m)$ when suppressing the randomization from the notation.

Efficiency. As performance measure for our protocols we will count the number of exponentiations. Note that because of "Shamir's trick" [9], which is a special case of Straus' algorithm [25], the complexity of single, double and even triple exponentiations ($g_1^{r_1}$, $g_1^{r_1} g_2^{r_2}$ and $g_1^{r_1} g_2^{r_2} g_3^{r_3}$) are comparable.

2.2 Zero-Knowledge Proofs of Knowledge

A rotator must convince a verifier that the output list is indeed a rotation of the input list, without giving away any information on the offset used between these lists. For this purpose we use standard notions for zero-knowledge and knowledge soundness

A proof, or argument, of knowledge for a relation $R = \{(x, w)\}$ is a protocol between a prover and a verifier. Both parties get a value x as common input while the prover gets a witness w as private input such that $(x, w) \in R$. At the end of the protocol, the verifier decides whether it accepts or rejects the proof.

A proof must be complete and sound. Completeness means that given a pair $(x, w) \in R$, the proof is accepting if both prover and verifier follow the protocol. Soundness captures the fact that a cheating prover cannot succeed convincing a verifier if the prover does not know a witness w for x. This is shown by the definition of a knowledge extractor which uses the prover to compute a witness, see, e.g., [11]. A proof is zero-knowledge if there exists a simulator that given x and access to a malicious verifier, produces a view of the protocol that is indistinguishable from the view when the verifier interacts with a real prover. In honest-verifier zero-knowledge (HVZK) proofs the verifier is assumed to be honest but curious. Additionally, an HVZK proof is called special HVZK (SHVZK) when the simulator can produce views for a given challenge of the verifier.

Examples of special honest-verifier zero-knowledge proofs of knowledge are the well-known Σ-protocols [6, 4]. These are 3-move protocols where the prover acts first. They satisfy the so-called special-soundness property.

Our first protocol is a Σ-protocol. For our second protocol, we will actually show the existence of a knowledge extractor along the lines of Groth [12, 15]. Concretely, we show that our protocols have a witness-extended emulator. This notion, introduced by Lindell [18] implies knowledge soundness as defined by Damgård and Fujisaki [7], as shown in [14].

Informally, the witness-extended emulation property says that given an adversarial prover that produces an acceptable proof with some probability ε, there exists an expected polynomial time algorithm E, called witness-extended emulator, that produces indistinguishable transcripts which are accepting with (essentially) the same probability ε. If the transcript is accepting then a witness is provided as well. The emulator has rewindable black-box access to the prover.

It can easily be shown that a Σ-protocol has the witness-extended emulation property [13]. This fact will be used in our security proofs.

3 DFT-based Solution

Let $X = (X_0, X_1, \ldots, X_{n-1})$ be a list of homomorphic ElGamal encryptions. A random rotation is performed by picking a random offset r, $0 \leq r < n$, and computing a list of encryptions $Y = (Y_0, Y_1, \ldots, Y_{n-1})$, where $Y_k = X_{k-r}(g^{s_k}, h^{s_k})$, $s_k \in_R \mathbb{Z}_q$, for $k = 0, \ldots, n-1$. The challenge is to provide an efficient proof that the output list Y is correctly formed, for a given input list X.

The key mathematical tool for our first protocol is the Discrete Fourier Transform (DFT). Using the DFT one can express conveniently that two lists of encryptions are rotated version of each other, which allows for an efficient Σ-protocol to make a rotation verifiable.

3.1 Discrete Fourier Transform

Discrete Fourier Transform over Finite Fields. For simplicity, we present the DFT over the field $\mathbb{Z}_q$ for prime q. Suppose $n \mid q-1$ and let $\alpha \in \mathbb{Z}_q$ denote an n-th root of unity modulo q, that is, α is an element of order n in $\mathbb{Z}_q$. So, $\alpha^n = 1 \bmod q$.

The DFT for a sequence $x_k \in \mathbb{Z}_q$ w.r.t. α is a sequence $x'_k \in \mathbb{Z}_q$ defined as

$$x'_k = \sum_{j=0}^{n-1} x_j \alpha^{kj}.$$

Borrowing terminology from Fourier analysis, a Fourier Transform converts a sequence in the time domain into a sequence in the frequency (transformed) domain. The DFT can be seen as a linear transformation given by a Vandermonde matrix $A_{kj} = (\alpha^{kj})$. The inverse DFT (IDFT), which takes a sequence in the frequency domain and converts it back to the time domain, is given by

$$x_k = n^{-1} \sum_{i=0}^{n-1} x'_i \alpha^{-ik}.$$

Rotating Lists of Encryptions. Consider two sequences $x_0, x_1, \ldots, x_{n-1}$ and $y_0, y_1, \ldots, y_{n-1}$ such that $y_k = x_{k-r}$. The key property is now, with $0 \le k < n$:

$$y'_k = \sum_{j=0}^{n-1} y_j \alpha^{kj} = \sum_{j=0}^{n-1} x_{j-r} \alpha^{kj} = \sum_{j=0}^{n-1} x_j \alpha^{k(j+r)} = \alpha^{rk} x'_k = \beta^k x'_k,$$

where $\beta = \alpha^r$. Hence, if we first apply DFT to a sequence $x_0, x_1, \ldots, x_{n-1}$ yielding $x'_0, x'_1, \ldots, x'_{n-1}$, then compute $y'_0, y'_1, \ldots, y'_{n-1}$ by setting

$$y'_k = \beta^k x'_k = \alpha^{rk} x'_k,$$

for $0 \le k < n$, and finally apply IDFT to obtain sequence $y_0, y_1, \ldots, y_{n-1}$, it follows by construction that $y_k = x_{k-r}$ and thus, the two sequences are a rotation of each other.

We use this approach to perform efficient rotations of list of encryptions by means of a cascade of verifiable rotators. Since the coefficients α^{kj} can be computed publicly, one can apply DFT and IDFT to an encrypted sequence using just the homomorphic properties. These transformations are performed once, at the beginning and at the end of the cascade, respectively. The rotators will pass on transformed sequences between each other.

Concretely, each verifiable rotator will perform the following transformation to a given list of encryptions $\mathsf{E}(x'_0), \ldots, \mathsf{E}(x'_{n-1})$:

$$\mathsf{E}(y'_k) = \mathsf{E}(x'_k)^{\beta^k} (g^{s_k}, h^{s_k}), \text{for } k = 0, 1, \ldots, n-1, \tag{1}$$

with $s_k \in_R \mathbb{Z}_q$ and $\beta = \alpha^r$, $r \in_R \{0, 1, \ldots, n-1\}$. The purpose of the random re-encryptions (g^{s_k}, h^{s_k}) is to hide the rotation offset r being used.

The transformation at the beginning of the cascade of rotators can be done publicly, using the homomorphic property:

$$\mathsf{E}(x'_k) = \prod_{j=0}^{n-1} \mathsf{E}(x_j)^{\alpha^{kj}}.$$

Similarly, the transformation at the end of the cascade, if desired, can be done publicly. Below, we will introduce the use of the Fast Fourier Transform (FFT) algorithm to perform these transformation using $n \log n$ exponentiations only. This way the cost of the transformation at the beginning (and possibly at the end) of the cascade is amortized over the length of the cascade. When the length of the cascade is $\Omega(\log n)$, the work will be $O(n)$ per rotator.

3.2 DFT-based Protocol

Σ-protocol. To make a rotation verifiable, we provide a proof that the list of encryptions are transformed according to Eq. (1). To this end, a rotator needs to prove that it knows a value $\beta \in \mathbb{Z}_q$ with $\beta^n = 1 \pmod q$ and values $s_0, s_1, \ldots, s_{n-1} \in \mathbb{Z}_q$ such that Eq. (1) holds. We show how this can be done very efficiently using standard techniques.

Let $(a_k, b_k) = X'_k$ and $(d_k, e_k) = Y'_k$ be ElGamal encryptions. The rotator has to prove that it knows $\beta \in \mathbb{Z}_q$ such that $\beta^n = 1$, and values $s_0, \ldots, s_{n-1} \in \mathbb{Z}_q$ such that

$$d_k = a_k^{\beta^k} g^{s_k}, e_k = b_k^{\beta^k} h^{s_k}.$$

To prove that the exponents are of the form β^k, the prover produces auxiliary homomorphic Pedersen commitments $\mathsf{C}(\beta), \mathsf{C}(\beta^2), \ldots, \mathsf{C}(\beta^{n-1})$, and proves that this sequence is indeed a sequence of powers of some β. As a stepping-stone, we use the efficient Σ-protocol for showing that $z = xy$ for commitments $\mathsf{C}(x), \mathsf{C}(y), \mathsf{C}(z)$ (see [5]). Starting with a commitment to β, we prove iteratively the knowledge of its powers as follows. Let $c_0 = \mathsf{C}(1)$, then successively construct the commitments $c_1 = \mathsf{C}(\beta) = c_0^\beta \widetilde{h}^{t_0}$, $c_2 = c_1^\beta \widetilde{h}^{t_1} = \mathsf{C}(\beta^2), c_3 = c_2^\beta \widetilde{h}^{t_2} = \mathsf{C}(\beta^3), \ldots, c_{n-1} = c_{n-2}^\beta \widetilde{h}^{t_{n-2}} = \mathsf{C}(\beta^{n-1})$ and apply a Σ protocol to show that one and the same exponent β is used for all these commitments when expressed this way.

To prove that $\beta^n = 1$ holds, we let the rotator compute $c_n = c_{n-1}^\beta \widetilde{h}^{t_{n-1}} = \mathsf{C}(\beta^n)$ as well and prove that this is a commitment to 1. This can be done by applying a proof of knowledge of the discrete log of c_n/g with respect to the base $\widetilde{h}$, as $c_n/g = \widetilde{h}^{t^*_{n-1}}$, for some value t^*_{n-1} (see below).

Finally, now given $(a_k, b_k) = X'_k$, $c_k = \mathsf{C}(\beta^k)$, and $(d_k, e_k) = Y'_k$, the rotator has to prove that it knows values s_k, t^*_k, β^k such that

$$c_k = g^{\beta^k} \widetilde{h}^{t^*_k}, d_k = a_k^{\beta^k} g^{s_k}, e_k = b_k^{\beta^k} h^{s_k}.$$

where $t^*_k = \sum_{j=0}^k \beta^{k-j} t_j$.

The Σ-protocol which combines all these steps is shown in Fig. 1.

$$
\begin{array}{ccc}
\textbf{Prover} & \textbf{Common input:} & \textbf{Verifier} \\
(\text{knows } \beta, s_k \text{ s.t. } \beta \in \langle \alpha \rangle, & (a_k, b_k) = X'_k & \\
d_k = a_k^{\beta^k} g^{s_k}, e_k = b_k^{\beta^k} h^{s_k}) & (d_k, e_k) = Y'_k & \\
 & & \\
\widetilde{m}, b \in_R \mathbb{Z}_q & & \\
\widetilde{C} = \widetilde{h}^{\widetilde{m}} & & \\
m_k, t_k, u_k, v_k, \widetilde{m}_k \in_R \mathbb{Z}_q & & \\
c_{k+1} = c_k^{\beta} \widetilde{h}^{t_k} & & \\
C_{k+1} = c_k^{b} \widetilde{h}^{m_k} & & \\
\widetilde{C}_k = g^{u_k} \widetilde{h}^{\widetilde{m}_k} & & \\
D_k = a_k^{u_k} g^{v_k} & & \\
E_k = b_k^{u_k} h^{v_k} & & \\
 & \xrightarrow{\widetilde{C}, \{c_{k+1}, C_{k+1}, D_k, E_k, \widetilde{C}_k\}_{k=0}^{n-1}} & \\
t^*_{-1} = 0 & \xleftarrow{\lambda} & \lambda \in_R \mathbb{Z}_q \\
t^*_k = \sum_{j=0}^{k} \beta^{k-j} t_j & & \\
\sigma = b + \lambda \beta, & & \\
\eta = \widetilde{m} + \lambda t^*_{n-1} & & \\
\psi_k = m_k + \lambda t_k & & \\
\mu_k = u_k + \lambda \beta^k & & \\
\nu_k = v_k + \lambda s_k & & \\
\rho_k = \widetilde{m}_k + \lambda t^*_{k-1} & & \\
 & \xrightarrow{\sigma, \eta, \{\psi_k, \mu_k, \nu_k, \rho_k\}_{k=0}^{n-1}} & \widetilde{h}^{\eta} \overset{?}{=} \widetilde{C}(c_n/g)^{\lambda} \\
 & & c_k^{\sigma} \widetilde{h}^{\psi_k} \overset{?}{=} C_{k+1} c_{k+1}^{\lambda} \\
 & & g^{\mu_k} \widetilde{h}^{\rho_k} \overset{?}{=} \widetilde{C}_k c_k^{\lambda} \\
 & & a_k^{\mu_k} g^{\nu_k} \overset{?}{=} D_k d_k^{\lambda} \\
 & & b_k^{\mu_k} h^{\nu_k} \overset{?}{=} E_k e_k^{\lambda}
\end{array}
$$

Fig. 1. Proof of a rotation in the transformed domain of ElGamal encryptions, where $c_0 = g$ and k runs from 0 to $n-1$.

Efficiency. From Fig. 1 we can see that the generation of the proof requires $5n$ double exponentiations, while the verification requires $4n$ triple exponentiations.

An issue in the efficiency of a cascade of rotators is the computation of the DFT and possibly the IDFT under the encryptions. Although the DFT and the IDFT can be applied just using homomorphic properties, these steps may be a computational bottleneck as they involve the computation of n n-way exponentiations (which naively costs n^2 single exponentiations). However, e.g., assuming that n is an integral power of 2, one can apply the Fast Fourier Transform algorithm to reduce the computational complexity to $O(n \log n)$ exponentiations.

To illustrate the application of the FFT algorithm, assuming that $n = 2^\ell$, then the DFT computation under encryptions can be split in the following way:

$$\begin{aligned}
\mathsf{E}(x'_k) &= \prod_{j=0}^{n-1} \mathsf{E}(x_j)^{\alpha^{kj}} = \prod_{j=0}^{n/2-1} \mathsf{E}(x_{2j})^{\alpha^{k2j}} \mathsf{E}(x_{2j+1})^{\alpha^{k(2j+1)}} \\
&= \prod_{j=0}^{n/2-1} \mathsf{E}(x_{2j})^{\alpha^{2kj}} \left(\prod_{j=0}^{n/2-1} \mathsf{E}(x_{2j+1})^{\alpha^{2kj}} \right)^{\alpha^k} .
\end{aligned}$$

Noting that α^2 is a $n/2$-th root of unity modulo q, we have reduced the problem of computing the DFT of a sequence of length n to solving two DFTs of sequences of half length. We note that these two DFTs can be computed in parallel because they are independent of each other. If t_n is the number of exponentiations required to compute one element of the DFT of length n, then we get that $t_n = t_{n/2} + 1$. Finally, the total number of exponentiations is nt_n. In particular when n is a power of two, $n \log n$ exponentiations are required in total.

When using a cascade of rotators, the rotators will keep the sequence in the frequency domain. If so desired, the DFT and its inverse need to be applied only before and after the entire cascade. With this observation, we only need to transform at the first and final rotator in a cascade of rotators, and it helps as one can average the total work for transforms over the length of the cascade. In the special case of a cascade of n rotators, the work per rotator will be linear.

In some applications it may be reasonable to assume that the input and output lists are in transformed form. Another observation is that if the final output list of a cascade is going to be decrypted anyway, one may leave out the inverse DFT, and decrypt the transformed sequence. Then one can perform the inverse DFT on the plaintexts (depending on the homomorphic cryptosystem this may give an advantage, for ElGamal one would still need exponentiations).

Extensions. Using n-th roots of unity in extension fields of $\mathbb{Z}_q$ with the condition that $n \mid q-1$ can be weakened. Given a fixed n, we can adjust the extension field that we work with. This will come at the expense of increased communication and computation.

4 General Solution

The DFT-based protocol presented above puts some constraints on the parameters involved. In this section we present a different approach that does not such constraints.

We use a two-stage approach, similar to the approach for verifiable shuffles in [19, 12, 15]. We first present an auxiliary protocol to prove that a list of known committed values has been rotated. In the proofs of security we also use the Schwartz-Zippel lemma.

Lemma 1 (Schwartz-Zippel). *Let p be a multivariate polynomial of degree d over $\mathbb{Z}_q$. Then the probability that $p(x_1, x_2, \ldots, x_m) = 0$ for randomly chosen $x_1, x_2, \ldots, x_m$ over $\mathbb{Z}_q$ is at most d/q.*

We will use it for the case that $d = 1$ to test that two sequences have the same values. That is, given $(x_0, x_1, \ldots, x_{n-1})$ and $(y_0, y_1, \ldots, y_{n-1})$, then if

$$\sum_{j=0}^{n-1} \beta_j x_j = \sum_{j=0}^{n-1} \beta_j y_j,$$

for $\beta_0, \beta_1, \ldots, \beta_{n-1} \in_R \mathbb{Z}_q$, it follows that $(x_0, x_1, \ldots, x_{n-1}) \neq (y_0, y_1, \ldots, y_{n-1})$ with probability at most $1/q$.

4.1 Rotation of Known Committed Values

Let $\alpha_0, \alpha_1, \ldots, \alpha_{n-1}$ be some publicly known values. The prover produces some commitments $c_0, c_1, \ldots, c_{n-1}$, for which it will prove knowledge of an offset r and randomizers $s_0, s_1, \ldots, s_{n-1}$ satisfying:

$$c_k = g^{\alpha_{k-r}} \widetilde{h}^{s_k}, \text{ for } k = 0, 1, \ldots, n-1. \tag{2}$$

Building Blocks. We will use a Σ-protocol DL-OR$(G, \gamma_0, \gamma_1, \ldots, \gamma_{n-1})$ to prove that, given randomly selected challenges $\beta_0, \beta_1, \ldots, \beta_{n-1}$, $G = \prod_{j=0}^{n-1} c_j^{\beta_j}$ is a commitment to one of γ_0, γ_1, $\ldots, \gamma_{n-1}$ as defined in the protocol of Fig. 2.

Intuitively, the initial commitments $c_0, c_1, \ldots, c_{n-1}$ commit to a rotation of $\alpha_0, \alpha_1, \ldots, \alpha_{n-1}$ if the prover knows the inner product of a random vector with one of the possible rotated versions of the vector with the α-values.

Theorem 1. *The protocol* PUB-ROT*of Fig. 2 is a special honest-verifier zero-knowledge proof of knowledge with witness-extended emulation that a prover knows an integer r, $0 \leq r < n$ and randomizers $s_0, s_1, \ldots, s_{n-1}$ such that Eq. (2) holds.*

Proof. To show completeness, let commitments $c_0, c_1, \ldots, c_{n-1}$ be defined as in Eq. (2). For challenge $\beta_0, \beta_1, \ldots, \beta_{n-1}$ the values $\gamma_0, \gamma_1, \ldots, \gamma_{n-1}$ and the commitment G are then computed. Observe that G is a commitment to the value $\sum_{j=0}^{n-1} \alpha_{j-r}\beta_j$ due to the homomorphic properties, which is equal to γ_r. As G is a commitment to γ_r, the proof DL-OR will be accepted by the verifier.

To show SHVZK, we construct a simulator as follows. Given challenges $\beta_0, \beta_1, \ldots, \beta_{n-1}$ and λ, it follows that the values $\gamma_0, \gamma_1, \ldots, \gamma_{n-1}$ and G can be computed as specified by the protocol. From the SHVZK property of the Σ-protocol DL-OR$(G, \gamma_0, \gamma_1, \ldots, \gamma_{n-1})$, there exists a simulator which produces an indistinguishable view when it is given challenge λ and all public information, namely $G, \gamma_0, \gamma_1, \ldots, \gamma_{n-1}$.

We show knowledge soundness through witness-extended emulation. Witness extended emulation can be shown using standard techniques as done in [12, 14,

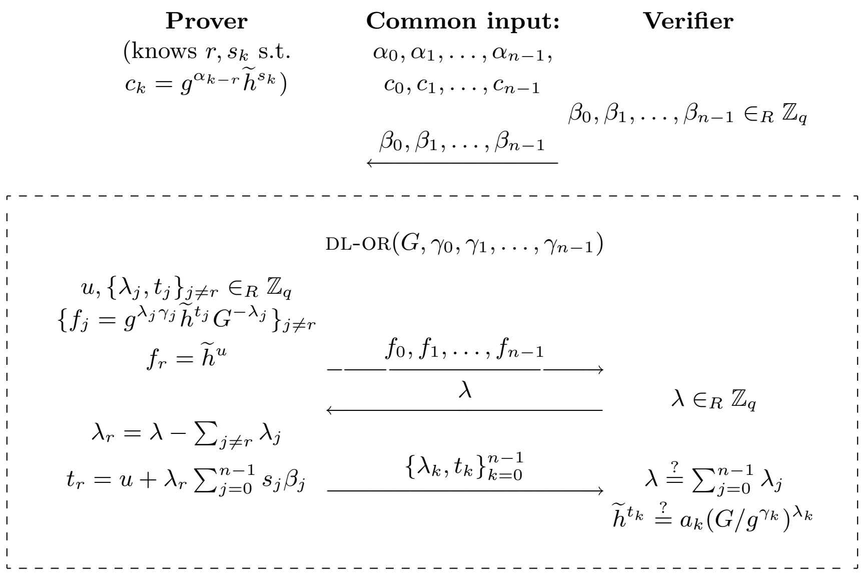
Prover
Common input:
Verifier
(knows r, s_k s.t.
$c_k = g^{\alpha_{k-r}}\widetilde{h}^{s_k}$)
$\alpha_0, \alpha_1, \ldots, \alpha_{n-1},$
$c_0, c_1, \ldots, c_{n-1}$
$\beta_0, \beta_1, \ldots, \beta_{n-1} \in_R \mathbb{Z}_q$
$\xleftarrow{\beta_0, \beta_1, \ldots, \beta_{n-1}}$
DL-OR$(G, \gamma_0, \gamma_1, \ldots, \gamma_{n-1})$
$u, \{\lambda_j, t_j\}_{j \neq r} \in_R \mathbb{Z}_q$
$\{f_j = g^{\lambda_j \gamma_j}\widetilde{h}^{t_j}G^{-\lambda_j}\}_{j \neq r}$
$f_r = \widetilde{h}^u$
$\xrightarrow{f_0, f_1, \ldots, f_{n-1}}$
$\xleftarrow{\lambda}$
$\lambda \in_R \mathbb{Z}_q$
$\lambda_r = \lambda - \sum_{j \neq r} \lambda_j$
$t_r = u + \lambda_r \sum_{j=0}^{n-1} s_j \beta_j$
$\xrightarrow{\{\lambda_k, t_k\}_{k=0}^{n-1}}$
$\lambda \overset{?}{=} \sum_{j=0}^{n-1} \lambda_j$
$\widetilde{h}^{t_k} \overset{?}{=} a_k (G/g^{\gamma_k})^{\lambda_k}$

Fig. 2. Protocol PUB-ROT for proving a rotation of known committed values, where $\gamma_k = \sum_{j=0}^{n-1} \alpha_{j-k}\beta_j$, for $k = 0, 1, \ldots, n-1$, and $G = \prod_{j=0}^{n-1} c_j^{\beta_j}$.

15]. The idea is to describe an emulator that runs in expected polynomial time producing a view indistinguishable from that of the protocol and at the same time gives a witness with the same probability as an adversarial prover produces an accepting conversation. This is achieved by first letting the prover run on random selected challenges, and if it is accepting, the witness is extracted using rewinding techniques until a sufficient number of transcripts have been obtained. As we use the Σ-protocol DL-OR as building block, we will make use of its witness-extended emulator, denoted as $E_{\text{DL-OR}}$.

The description of the emulator E is as follows. E picks random challenges $\overline{\beta}^{(1)} = \beta_0^{(1)}, \beta_1^{(1)}, \ldots, \beta_{n-1}^{(1)}$ and plays the witness-extended emulator for DL-OR, $E_{\text{DL-OR}}$, on $G^{(1)}$ and $\overline{\gamma}^{(1)}$ as defined in the protocol. This invocation will give a transcript of DL-OR along with a witness if the transcript is accepting. If the transcript is not accepting, E outputs no witness along with $(\overline{\beta}^{(1)}, \mathit{view}^{(1)})$ as the view of the protocol.

Otherwise, we have a valid transcript and a witness of DL-OR. Namely, an integer $r^{(1)}$ and a randomizer $t^{(1)}$ such that $G^{(1)} = \mathsf{C}(\gamma_{r^{(1)}}^{(1)}, t^{(1)})$.

The prover is rewound and fresh challenges $\beta_0^{(i)}, \beta_1^{(i)}, \ldots, \beta_{n-1}^{(i)}$ are chosen, for $i = 2, 3, \ldots, n$ until n valid transcripts and n witnesses for DL-OR are obtained, by subsequently invoking $E_{\text{DL-OR}}$. From all these n witnesses, E is able to compute the witness. After this is done, the output of E is the witness plus $\overline{\beta}^{(1)}$ attached to the first view obtained from $E_{\text{DL-OR}}$.

We first show how E manages to get a witness. Then, we show that this extractor runs in expected polynomial time and argue that E gives an accepting view plus a witness with essentially the same probability that the prover is able to produce accepting conversations.

From all of witnesses obtained, we get the following equalities, with $1 \leq i \leq n$.

$$G^{(i)} = \mathsf{C}(\gamma^{(i)}_{r^{(i)}}, t^{(i)}). \tag{3}$$

Also, as specified by the protocol, we have

$$G^{(i)} = \prod_{j=0}^{n-1} c_j^{\beta_j^{(i)}}. \tag{4}$$

By construction, the vectors $\overline{\beta}^{(i)}$ with $i = 1, 2 \ldots, n$ are linearly independent with overwhelming probability (for n polynomial in the size of q). The linear independence of the vectors $\overline{\beta}^{(i)}$ implies the existence of elements $d_{k,i}$ such that $\sum_{k=1}^{n} \overline{\beta}^{(i)} d_{k,i}$ is the $(k+1)$-st standard unit vector of $\mathbb{Z}_q^n$, for $k = 0, 1, \ldots, n-1$.

This implies that

$$c_k = \prod_{k=1}^{n} \big(\prod_{j=0}^{n-1} c_j^{\beta_j^{(i)}} \big)^{d_{k,i}}.$$

By Eq. (4), it in turn implies that

$$\begin{aligned} c_k &= \prod_{k=1}^{n} (G^{(i)})^{d_{k,i}} = \prod_{k=1}^{n} \mathsf{C}(\gamma^{(i)}_{r^{(i)}}, t^{(i)})^{d_{k,i}} = \prod_{k=1}^{n} \mathsf{C}(d_{k,i}\gamma^{(i)}_{r^{(i)}}, d_{k,i}t^{(i)}) = \\ &= \mathsf{C}(\sum_{k=1}^{n} d_{k,i}\gamma^{(i)}_{r^{(i)}}, \sum_{k=0}^{n-1} d_{k,i}t^{(i)}) \end{aligned}$$

Therefore, we find an opening of the commitment c_k. Let $\widetilde{\alpha}_k = \sum_{k=1}^{n} d_{k,i}\gamma^{(i)}_{r^{(i)}}$ and $s_k = \sum_{k=1}^{n} d_{k,i}t^{(i)}$.

We now prove that $\widetilde{\alpha}_k$ are a rotated version of the elements α_k. From Eqs. (3) and (4), and using the binding property of the commitment scheme, it follows that

$$\sum_{j=0}^{n-1} \beta_j^{(i)} \widetilde{\alpha}_j = \gamma^{(i)}_{r^{(i)}} = \sum_{j=0}^{n-1} \beta_j^{(i)} \alpha_{j-r^{(i)}},$$

for all $i = 1, 2, \ldots, n$.

As the $\beta_j^{(i)}$ are randomly chosen, we conclude, using the Schwartz-Zippel lemma, that with overwhelming probability $\widetilde{\alpha}_j = \alpha_{j-r^{(i)}}$ holds. This shows that indeed the committed values in c_k are a rotated list of the α_k. Note that this allows us to conclude that $r^{(i)} = r^{(j)}$ for all $i \neq j$.

In summary, we have found an integer r and randomizers $s_0, s_1, \ldots, s_{n-1}$ such that $c_k = \mathsf{C}(\alpha_{k-r}, s_k)$ which is actually the witness for PUB-ROT.

We now argue that E runs in expected polynomial time. Let $\widetilde{\varepsilon}$ denote the probability that querying $E_{\text{DL-OR}}$ on independently random selected $\overline{\beta}$'s results in an accepting transcript. Then getting an accepting transcript and therefore a witness will take expected time $(1/\widetilde{\varepsilon})T$, where T is the expected running time of $E_{\text{DL-OR}}$. Therefore, the total expected running time of the repeated calls to $E_{\text{DL-OR}}$ made by E is $T + \widetilde{\varepsilon}(n-1)T/\widetilde{\varepsilon} = nT$.

Let ε denote the probability that a real prover gives an accepting proof of PUB-ROT. To check that the difference between ε and $\widetilde{\varepsilon}$ is negligible, we observe that ε can be seen as a weighted sum of probabilities $\sum_{\overline{\beta}}(1/q^n)\varepsilon_{\text{DL-OR}(\overline{\beta})}$, where $\varepsilon_{\text{DL-OR}(\overline{\beta})}$ is the success probability of the prover in DL-OR when it got challenges $\overline{\beta}$. On the other hand, the probability that we get an accepting answer in the first query to $E_{\text{DL-OR}}$ in E happens with probability $\sum_{\overline{\beta}}(1/q^n)\widetilde{\varepsilon}_{\text{DL-OR}(\overline{\beta})}$ where $\widetilde{\varepsilon}_{\text{DL-OR}(\overline{\beta})}$ denotes the probability that $E_{\text{DL-OR}}$ gives an accepting conversation on relation induced by challenges $\overline{\beta}$. We can conclude that $|\varepsilon - \widetilde{\varepsilon}|$ is negligible, using the fact that $|\varepsilon_{\text{DL-OR}(\overline{\beta})} - \widetilde{\varepsilon}_{\text{DL-OR}(\overline{\beta})}|$ is negligible which is true by definition of witness extended emulator for DL-OR. □

Efficiency. After the verifier submitted the challenges $\overline{\beta}$, the commitment G is computed using an n-way exponentiation, which roughly costs $n/2$ double exponentiations. Then the prover has to give the DL-OR proof based on G and the γ's. The latter requires n double exponentiations for the prover and the verifier separately. Therefore, we have $1.5n$ double exponentiations to produce the proof, and $1.5n$ to verify it.

4.2 General Rotation

Given two lists of encryptions, $X_0, X_1, \ldots, X_{n-1}$ and $Y_0, Y_1, \ldots, Y_{n-1}$, the prover shows that it knows an integer r, $0 \leq r \leq n-1$, and randomizers $s_0, s_1, \ldots, s_{n-1}$ satisfying:

$$Y_k = X_{k-r}(g^{s_k}, h^{s_k}) \text{ for } k = 0, 1, \ldots, n-1. \tag{5}$$

For this task, we propose the protocol presented in Fig. 3.

Building Blocks. In this protocol, besides of the proof for rotation of known contents, we use an additional proof of knowledge. The proof EQ-EXP(h, Y, A) allows the prover to show the knowledge of α, r, t such that $h = \mathsf{C}(\alpha, r)$ and $A = Y^{\alpha}\mathsf{E}(0, t)$, where Y is an encryption. This proof is actually a basic Σ-protocol.

Before going into the security analysis, we note that the protocol requires 6 rounds of interaction. This is because some rounds in the involved building blocks can be run in parallel. Namely, we can combine the first set of messages from EQ-EXP with the first answer from the prover in the protocol description of Fig. 3. Also the first round of PUB-ROT can be directly connected with the challenge coming from EQ-EXP. In the security analysis, however, we will use the modular description of the protocol.

$$\begin{array}{ccc}
\textbf{Prover} & \textbf{Verifier} & \\
 & \textbf{Common input:} & \\
\text{(knows } r, s_k \text{ s.t.} & X_k = (a_k, b_k), & \\
Y_k = (a_{k-r}, b_{k-r})(g^{s_k}, h^{s_k})) & Y_k = (d_k, e_k) & \\
 & & \alpha_0, \alpha_1, \ldots, \alpha_{n-1} \in_R \mathbb{Z}_q \\
 & \xleftarrow{\alpha_0, \alpha_1, \ldots, \alpha_{n-1}} & \\
u_k, t_k \in_R \mathbb{Z}_q & & \\
h_k = g^{\alpha_{k-r}} \widetilde{h}^{u_k} & & \\
A_k = (d_k^{\alpha_{k-r}}, e_k^{\alpha_{k-r}})(g^{t_k}, h^{t_k}) & & \\
v = \sum_{k=0}^{n-1} (\alpha_{k-r} s_k + t_k) & & \\
 & \xrightarrow{\{h_k, A_k\}_{k=0}^{n-1}, v} &
\end{array}$$

$$\{\text{EQ-EXP}(h_k, Y_k, A_k)\}_{k=0}^{n-1}$$

$$\begin{array}{ccc}
o_k, p_k, m_k \in_R \mathbb{Z}_q & & \\
f_k = g^{o_k} \widetilde{h}^{p_k} & & \\
F_k = (d_k^{o_k}, e_k^{o_k})(g^{m_k}, h^{m_k}) & \xrightarrow{\{f_k, F_k\}_{k=0}^{n-1}} & \\
 & \xleftarrow{\lambda} & \lambda \in_R \mathbb{Z}_q \\
\tau_k = o_k + \lambda \alpha_{k-r} & & \\
\rho_k = p_k + \lambda u_k & & \\
\mu_k = m_k + \lambda t_k & & \\
 & \xrightarrow{\{\tau_k, \rho_k, \mu_k\}_{k=0}^{n-1}} & g^{\tau_k} \widetilde{h}^{\rho_k} \overset{?}{=} f_k h_k^{\lambda} \\
 & & (d_k^{\tau_k}, e_k^{\tau_k})(g^{\mu_k}, h^{\mu_k}) \\
 & & \overset{?}{=} F_k A_k^{\lambda}
\end{array}$$

$$\boxed{\text{PUB-ROT}(\alpha_0, \ldots, \alpha_{n-1}, h_0, \ldots, h_{n-1})}$$

$$\prod_{j=0}^{n-1} A_j X_j^{-\alpha_j} \overset{?}{=} (g^v, h^v).$$

Fig. 3. Protocol GEN-ROT for proving a rotation of ElGamal encryptions, where k runs from 0 to $n-1$.

Theorem 2. *The protocol* GEN-ROT *is a special honest-verifier zero-knowledge proof of knowledge with witness extended emulation that a prover knows an integer r, $0 \le r < n$ and randomizers $s_0, s_1, \ldots, s_{n-1}$ such that Eq. (5) holds.*

Proof. To prove that the protocol for general rotation is SHVZK, we will construct a simulator, given $X_0, X_1, \ldots, X_{n-1}, Y_0, Y_1, \ldots, Y_{n-1}$, and the random challenges $\alpha_0, \alpha_1, \ldots, \alpha_{n-1}$ (along with all the challenges for PUB-ROT and the one for EQ-EXP).

The simulator runs as follows. First, it computes $A_k = X_k^{\alpha_k} \mathsf{E}(0, v_k)$ for random v_k. The assignment for A_k is somewhat arbitrary and there are various ways to do it, as long as it guarantees that the proof is accepting.

Now, the simulator creates commitments $h_0, h_1, \ldots, h_{n-1}$ all of them to 0 (or any value). Despite this choice, the simulators for both PUB-ROT and EQ-EXP will still produce transcripts that are indistinguishable from real ones. Combining the obtained transcripts with the computed array of commitments h_k, encryptions A_k, and the value v, the resulting view is indistinguishable to a real transcript of GEN-ROT.

For witness-extended emulation we construct the emulator E as follows. Given challenges $\overline{\alpha}^{(1)} = (\alpha_0^{(1)}, \alpha_1^{(1)}, \ldots, \alpha_{n-1}^{(1)})$, E runs the prover until it gets responses $\overline{h}^{(1)}, \overline{k}^{(1)}, v^{(1)}$, after this, the witness-extended emulators $E_{\text{EQ-EXP}}$ and $E_{\text{PUB-ROT}}$ are run. With some probability $\widetilde{\epsilon}$, they will produce an accepting transcript. If this is not the case, E outputs no witness along with the transcripts obtained from the witness-emulators and the prover. Else, E will rewind the prover and choose fresh random challenges $\overline{\alpha}^{(i)}$, until n accepting conversations are obtained in total. For each of them, let $\alpha_0^{(i)}, \alpha_1^{(i)}, \ldots, \alpha_{n-1}^{(i)}$ denote the challenges selected by E. For each of these cases, respective witness-extended emulators for PUB-ROT and EQ-EXP's are run in order to extract their witnesses.

This is the information necessary to compute the witness for GEN-ROT. We show now, how E is able to compute an integer r and randomizers $s_0, s_1, \ldots, s_{n-1}$ such that Eq. (5) holds.

We have for a particular iteration of $i = 1, 2, \ldots, n$, and for every j with $0 \leq j \leq n-1$ that

$$\begin{aligned} A_j^{(i)} &= Y_j^{b_j^{(i)}} \mathsf{E}(0, t_j^{(i)}), \text{ and} \\ h_j^{(i)} &= \mathsf{C}(b_j^{(i)}, u_j^{(i)}). \end{aligned} \tag{6}$$

for some values $\overline{b}, \overline{t}$ and $\overline{u}$.

We have also witnesses for PUB-ROT. That is, there exist integers $r^{(i)}$ and randomizers $w_j^{(i)}$ such that

$$h_j^{(i)} = \mathsf{C}(\alpha_{j-r^{(i)}}^{(i)}, w_j^{(i)}). \tag{7}$$

From (6) and (7), due to the binding property of the commitment scheme, it follows that $b_j^{(i)} = \alpha_{j-r^{(i)}}^{(i)}$ and $u_j^{(i)} = w_j^{(i)}$. One of the consequences is that

$$A_j^{(i)} = Y_j^{\alpha_{j-r^{(i)}}^{(i)}} \mathsf{E}(0, t_j^{(i)}). \tag{8}$$

As all these equations are based on accepting conversations then the verifications of the proof passes, and therefore,

$$\frac{\prod_{j=0}^{n-1} A_k^{(i)}}{\prod_{j=0}^{n-1} X_j^{\alpha_j^{(i)}}} = \frac{\prod_{j=0}^{n-1} Y_j^{\alpha_{j-r^{(i)}}^{(i)}} \mathsf{E}(0, t_j^{(i)})}{\prod_{j=0}^{n-1} X_j^{\alpha_j^{(i)}}} = \mathsf{E}(0, v^{(i)}). \tag{9}$$

This implies that,

$$\prod_{j=0}^{n-1} \left(\frac{Y_{j-r^{(i)}}}{X_j} \right)^{\alpha_j^{(i)}} = \mathsf{E}(0, v^{(i)} - \sum_{j=0}^{n-1} t_j^{(i)})$$

From this last equality, we can conclude using the Schwartz-Zippel lemma that the X_k and Y_k encrypt the same elements up to a rotation of these elements, implying also that $r^{(i)} = r^{(j)}$ for all $i \neq j$. We denote that rotation offset as r.[3]

Let $Z_j = \frac{Y_{j-r}}{X_j}$ and $\widetilde{n}_k = v^{(i)} - \sum_{j=0}^{n-1} t_j^{(i)}$. The aim now is to extract the randomness used in Z_j which is by homomorphic properties the randomness used to re-blind the rotated list Y_j.

As the challenges throughout the extraction process are selected independently at random, it implies that the vectors $\overline{\alpha}^{(i)}$ are linearly independent. This enables the existence of elements $d_{k,i}$ such that $\sum_{j=0}^{n-1} d_{k,i}\overline{\alpha}^{(i)}$ is the $(k+1)$-st standard unit vector in $\mathbb{Z}_q^n$, for $k = 0, 1, \ldots, n-1$. Therefore,

$$Z_k = \prod_{k=1}^{n} \left(\prod_{j=0}^{n-1} Z_j^{\alpha_j^{(i)}} \right)^{d_{k,i}} = \mathsf{E}(0, \sum_{k=1}^{n} d_{k,i}\widetilde{n}_k),$$

that enables us to extract $s_k = \sum_{k=1}^{n} d_{k,i}\widetilde{n}_k$.

The way of arguing that E runs in expected polynomial time is similar to the protocol for known contents. □

Efficiency. First, we calculate the computational cost to prove one invocation of EQ-EXP. This requires 3 double exponentiations and 3 triple exponentiations to produce, and 3 triple exponentiations to verify the proof.

Now, the number of exponentiations required to compute GEN-ROT. The prover must compute the commitments h_k, costing n double exponentiations. The encryptions A_k need $2n$ more double exponentiations. Then, $3n$ for EQ-EXP and $1.5n$ for PUB-ROT, totalling $7.5n$ double exponentiations for the prover to give a proof.

The verifier needs to check the EQ-EXP and PUB-ROT which requires $3n$ triple exponentiations and $1.5n$ double exponentiations, respectively. The verification at the end of the proof requires 2 more n-way exponentiation. Then, the total cost for the verifier is $5.5n$ double exponentiations.

5 Comparison and Concluding Remarks

To the best of our knowledge, the protocol by Reiter and Wang [21] is the only one that presents protocols for proving a rotation. Their construction works in general given that appropriate homomorphic cryptosystem has been set up. In fact, they show a protocol to prove that a permutation has been applied making use of 4 invocations to the proof of shuffle.

Table 1 presents a comparison of the computational complexities for various protocols for rotation. In all cases, we assume that the protocols are based on homomorphic ElGamal and Pedersen commitments. For simplicity, we count

[3] If this is not the case, this must be because the plaintexts in X_k are all the same for all k. This is not a problem though, we just take $r = 0$ and the rest of the procedure will still be applicable.

Table 1. Performance figures for proving a rotation.

Protocol	Prove	Verify	Rounds
Loop permutations [21]	$16n$	$10n$	30
Proposed: DFT-based	$5n$	$4n$	3
Proposed: General	$7.5n$	$5.5n$	6

the number of double exponentiations. Additionally, for the protocol in [21] we assume that they use the proof of shuffle by Groth [12] which is one of the most efficient.

Acknowledgements. We thank anonymous reviewers for their valuable comments. The fourth author is supported by the research program Sentinels (`http://www.sentinels.nl`). Sentinels is being financed by Technology Foundation STW, the Dutch Organization for Scientific Research (NWO), and the Dutch Ministry of Economic Affairs.

References

1. M. J. Atallah, M. Blanton, K. B. Frikken, and J. Li. Efficient correlated action selection. In *Financial Cryptography – FC 2006*, volume 4107 of *Lecture Notes in Computer Science*, pages 296–310. Springer-Verlag, 2006.
2. I. Blake and V. Kolesnikov. Strong conditional oblivious transfer and computing on intervals. In *ASIACRYPT 2004*, volume 3329 of *Lecture Notes in Computer Science*, pages 515–529. Springer-Verlag, 2004.
3. D. Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM*, 24(2):84–88, 1981.
4. R. Cramer. *Modular Design of Secure yet Practical Cryptographic Protocols*. PhD thesis, Universiteit van Amsterdam, Netherlands, 1997.
5. R. Cramer and I. Damgård. Zero-knowledge for finite field arithmetic. Or: Can zero-knowledge be for free? In *CRYPTO 1998*, volume 1462 of *Lecture Notes in Computer Science*, pages 424–441. Springer-Verlag, 1998.
6. R. Cramer, I. Damgård, and B. Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. In *CRYPTO 1994*, volume 839 of *Lecture Notes in Computer Science*, pages 174–187. Springer-Verlag, 1994.
7. I. Damgård and E. Fujisaki. Efficient concurrent zero-knowledge in the auxiliary string model. In *ASIACRYPT 2002*, volume 2501 of *Lecture Notes in Computer Science*, pages 125–142. Springer-Verlag, 2002.
8. I. Damgård, M. Geisler, and M. Krøigaard. Efficient and secure comparison for on-line auctions. In *Australasian Conference Information Security and Privacy – ACISP 2007*, volume 4586 of *Lecture Notes in Computer Science*, pages 416–430. Springer-Verlag, 2007.
9. T. ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In *IEEE Transactions on Information Theory IT-31*, pages 469–472, 1985.

10. J. Garay, B. Schoenmakers, and J. Villegas. Practical and secure solutions for integer comparison. In *Public Key Cryptography–PKC 2007*, volume 4450 of *Lecture Notes in Computer Science*, pages 330–342. Springer-Verlag, 2007.
11. O. Goldreich. *Foundations of Cryptography.* Cambridge University Press, Cambridge, UK, 2001.
12. J. Groth. A verifiable secret shuffle of homomorphic encryptions. In *Public Key Cryptography–PKC 2003*, volume 2567 of *Lecture Notes in Computer Science*, pages 145–160. Springer-Verlag, 2003. Full version available at http://eprint.iacr.org/2005/246.
13. J. Groth. Evaluating security of voting schemes in the universal composability framework. In *Applied Cryptography and Network Security – ACNS 2004*, volume 3089 of *Lecture Notes in Computer Science*, pages 46–60. Springer-Verlag, 2004.
14. J. Groth. *Honest Verifier Zero-Knowledge Arguments Applied.* PhD thesis, University of Aarhus, 2004.
15. J. Groth and Y. Ishai. Sub-linear zero-knowledge argument for correctness of a shuffle. In *EUROCRYPT 2008*, volume 4965 of *Lecture Notes in Computer Science*, pages 379–396. Springer-Verlag, 2008.
16. M. Jakobsson and A. Juels. Mix and match: Secure function evaluation via ciphertexts. In *ASIACRYPT 2000*, volume 1976 of *Lecture Notes in Computer Science*, pages 162–177. Springer-Verlag, 2000.
17. J. Li and M. Atallah. Secure and private collaborative linear programming. *Collaborative Computing: Networking, Applications and Worksharing, 2006. CollaborateCom 2006*, pages 1–8, 2006.
18. Y. Lindell. Parallel coin-tossing and constant-round secure two-party computation. *Journal of Cryptology*, 16(3):143–184, 2001.
19. C. Neff. A verifiable secret shuffle and its application to e-voting. In *8th ACM conference on Computer and Communications Security – CCS 2001*, pages 116–125. ACM, 2001.
20. T. Reistad and T. Toft. Secret sharing comparison by transformation and rotation. In *Pre-Proceedings of the International Conference on Information Theoretic Security–ICITS 2007.* Springer-Verlag, 2007. To appear at LNCS.
21. M. K. Reiter and X. Wang. Fragile mixing. In *CCS '04: Proceedings of the 11th ACM conference on Computer and communications security*, pages 227–235, New York, NY, USA, 2004. ACM.
22. P. Ryan. Prêt-à-Voter with Paillier encryption, 2006. Technical Report CS-TR No 965, School of Computing Science, Newcastle University. `http://www.cs.ncl.ac.uk/publications/trs/papers/965.pdf`.
23. P. Ryan and F. Schneider. Prêt-à-Voter with re-encryption mixes. In *Computer Security – ESORICS 2006*, volume 4189 of *Lecture Notes in Computer Science*, pages 313–326, Berlin, 2006. Springer-Verlag.
24. K. Sako and J. Killian. Receipt-free mix-type voting scheme. In *EUROCRYPT 1995*, volume 921 of *Lecture Notes in Computer Science*, pages 393–403. Springer-Verlag, 1995.
25. E. Straus. Addition chains of vectors (problem 5125). *American Mathematical Monthly*, 71:806–808, 1964.
26. R. Wen and R. Buckland. Mix and test counting for the alternative vote electoral system, 2008. Presented at WISSec 2008.
27. A. Yao. How to generate and exchange secrets. In *27th IEEE Symposium on Foundations of Computer Science*, pages 162–168, 1986.