A Sender Verifiable Mix-Net and a New Proof of a Shuffle*

Douglas Wikström

Royal Institute of Technology (KTH)
KTH, Nada, SE-100 44 Stockholm, Sweden
dog@nada.kth.se

March 19, 2009

Abstract. We introduce the first El Gamal based mix-net in which each mix-server partially decrypts and permutes its input, i.e., no re-encryption is necessary. An interesting property of the construction is that a sender can verify non-interactively that its message is processed correctly. We call this *sender verifiability.*
We prove the security of the mix-net in the UC-framework against static adversaries corrupting any minority of the mix-servers. The result holds under the decision Diffie-Hellman assumption, and assuming an ideal bulletin board and an ideal zero-knowledge proof of knowledge of a correct shuffle.
Then we construct the first proof of a decryption-permutation shuffle, and show how this can be transformed into a zero-knowledge proof of knowledge in the UC-framework. The protocol is sound under the strong RSA-assumption and the discrete logarithm assumption.
Our proof of a shuffle is not a variation of existing methods. It is based on a novel idea of independent interest, and we argue that it is at least as efficient as previous constructions.

Keywords: mix-net, proof of a shuffle, electronic election, anonymous channel.

A conference version of this work is accepted to Asiacrypt 2005.

1 Introduction

The notion of a mix-net was invented by Chaum [10]. Properly constructed a mix-net takes a list of cryptotexts and outputs the cleartexts permuted using a secret random permutation. Usually a mix-net is realized by a set of mix-servers organized in a chain that collectively execute a protocol. Each mix-server receives a list of encrypted messages from the previous mix-server, transforms them, using partial decryption or random re-encryption, reorders them, and outputs the result. The secret permutation is shared by the mix-servers.

* The proof of one of the claims in this paper has been corrected. Appendix K details the error and sketches the solution. The modifications in the paper are marked by footnotes, starting with Footnote 1 on Page 12.

1.1 Previous Work

Chaum's original "anonymous channel" [10,43] enables a sender to send mail anonymously. When constructing election schemes [10,16,45,50,42] a mix-net can be used to ensure that the vote of a given voter cannot be revealed. Abe gives an efficient construction of a general mix-net [1], and argues about its properties. Jakobsson has written (partly with Juels) more general papers on the topic of mixing [31,30,32] focusing on efficiency. There are two known approaches to proving a correct shuffle efficiently. These are introduced by Furukawa et al. [20,19,18], and Neff [39,40] respectively. Groth [27] generalizes Neff's protocol to form an abstract protocol for any homomorphic cryptosystem.

Desmedt and Kurosawa [12] describe an attack on a protocol by Jakobsson [31]. Similarly Mitomo and Kurosawa [38] exhibit a weakness in another protocol by Jakobsson [30]. Pfitzmann has given some general attacks on mix-nets [47,46], and Michels and Horster give additional attacks in [37]. Wikström [52] gives several attacks for a protocol by Golle et al. [25]. He also gives attacks for the protocols by Jakobsson [30] and Jakobsson and Juels [33]. Abe [3] has independently found related attacks.

Canetti [8], and independently Pfitzmann and Waidner [48] proposed security frameworks for reactive processes. We use the former universal composability (UC) framework. Both frameworks have composition theorems, and are based on older definitional work. The initial ideal-model based definitional approach for secure function evaluation is informally proposed by Goldreich, Micali, and Wigderson in [22]. The first formalizations appear in Goldwasser and Levin [23], Micali and Rogaway [36], and Beaver [5]. Canetti [7] presents the first definition of security that is preserved under composition. See [7,8] for an excellent background on these definitions.

Wikström [53] defines the notion of a mix-net in the UC-framework, and provides a construction that is provably secure against static adversaries under the decisional Diffie-Hellman assumption. The scheme is practical when the number of mix-servers is small, but for more than a handful of mix-servers it is not.

1.2 Contributions

We introduce a new type of El Gamal based mix-net in which each mix-server only decrypts and permutes its input. No re-encryption is necessary. This allows an individual sender to verify non-interactively that its message was processed correctly, i.e., the scheme is *sender verifiable*. Although some older constructions have this property, our is the first provably secure scheme.

Then we give the first proof of a decrypt-permutation shuffle of El Gamal cryptotexts. There are two known approaches, [39,27] and [20], to construct such a protocol, but our solution is based on a novel idea of independent interest, and we argue that it is at least as efficient as previous schemes.

We also give the first transformation of a proof of a shuffle into an efficient zero-knowledge proof of knowledge in the UC-framework. An important technical advantage of the new decrypt and permute construction is that witnesses are much smaller than for previous shuffle relations.

Combined, our results give a mix-net that is universally composable under the DDH-assumption and the strong RSA-assumption against static adversaries corrupting any minority of the mix-servers. The mix-net is efficient for any number of mix-servers, improving the result in Wikström [53].

1.3 Outline of the Paper

The paper is organized as follows. We introduce notation in Section 2. In Section 3 we define the ideal mix-net functionality we wish to realize. The first partial result in this direction is given in Section 4, where we describe a sender verifiable mix-net and discuss what sender verifiability means. In Section 5 we describe a zero-knowledge proof of knowledge that a mix-server processes its input correctly. Then in Section 6 we show how this can be transformed into a realization of an ideal zero-knowledge functionality in the UC-framework.

Proofs of all claims and formal definitions of our assumptions are given in the appendix.

2 Notation

Throughout, $S_1, \ldots, S_N$ denote senders and $M_1, \ldots, M_k$ mix-servers. All participants are modeled as interactive Turing machines. We abuse notation and use S_i and M_j to denote both the machines themselves and their identity. We denote the set of permutations of N elements by Σ_N. We use the term "randomly" instead of "uniformly and independently at random". A function $f : \mathbb{N} \rightarrow [0, 1]$ is said to be negligible if for each $c > 0$ there exists a $K_0 \in \mathbb{N}$ such that $f(K) < K^{-c}$ for $K > K_0 \in \mathbb{N}$. A probability $p(K)$ is overwhelming if $1 - p(K)$ is negligible.

We assume that G_q is a group of prime order q with generator g for which the Decision Diffie-Hellman (DDH) Assumption holds. Definition 11 in Section H.4 formalizes this assumption. Informally, it means that it is hard to distinguish the distributions $(g^\alpha, g^\beta, g^{\alpha\beta})$ and $(g^\alpha, g^\beta, g^\gamma)$ when $\alpha, \beta, \gamma \in \mathbb{Z}_q$ are randomly chosen. This implies that also the Discrete Logarithm (DL) assumption holds, namely that it is hard to compute the logarithm in base g of a random element in G_q. For concreteness we let G_q be a subgroup of prime order q of the multiplicative group $\mathbb{Z}_p^*$ for some prime p. When we say that an element in $\mathbb{Z}_q$ is prime, we mean that its representative in $\{0, \ldots, q-1\}$ is a prime when considered as an integer.

We review the El Gamal [13] cryptosystem employed in G_q. The private key x is generated by choosing $x \in \mathbb{Z}_q$ randomly. The corresponding public key is (g, y), where $y = g^x$. Encryption of a message $m \in G_q$ using the public key (g, y) is given by $E_{(g,y)}(m, r) = (g^r, y^r m)$, where r is chosen randomly from $\mathbb{Z}_q$, and decryption of a cryptotext on the form $(u, v) = (g^r, y^r m)$ using the private key x is given by $D_x(u, v) = u^{-x} v = m$. Tsionis and Yung [51] show that the El Gamal cryptosystem is semantically secure [24,35] under the DDH-assumption. We also use an RSA modulus $\mathbf{N} = \mathbf{pq}$, where $\mathbf{p}$ and $\mathbf{q}$ are strong primes. We denote by $\mathrm{QR}_{\mathbf{N}}$ the group of squares in $\mathbb{Z}_{\mathbf{N}}$ and adopt the convention that any element $\mathbf{b}$ in $\mathrm{QR}_{\mathbf{N}}$ is written in boldface. We assume that the strong RSA-assumption holds for such rings. Definition 12 in Section H.5 formalizes this assumption. Informally, it means that given random $(\mathbf{N}, \mathbf{h})$, where $\mathbf{h} \in \mathrm{QR}_{\mathbf{N}}$, it is hard to find a non-trivial eth root $\mathbf{b}$ of $\mathbf{h}$, i.e., an $e \neq \pm 1$ such that $\mathbf{b}^e = \mathbf{h}$. This differs from the RSA-assumption in that e is not fixed.

The primary security parameter K_1 is the number of bits in q. Several other security parameters are introduced later in the paper.

We denote by PRG a pseudo-random generator as defined in Section H.2. We denote by Sort the algorithm that given a list of strings as input outputs the same set of strings in lexicographical order.

2.1 The Universally Composable Security Framework

We give a concise and formal review of the framework in Appendix I, and present an informal review here. The informal review should suffice to make the paper intelligible, but to understand the proofs the reader may need to consult Appendix I (or [8]).

The core of the framework consists of the real model, the ideal model, and many different hybrid models. In all models the corresponding adversary may corrupt a certain fraction of the parties.

The real model is a model of real world computing, i.e. a list of interactive Turing machines execute a protocol over an asynchronous authenticated open network. The real adversary can see all communication and decide when messages are delivered. The ideal model contains an ideal functionality, i.e. a trusted party, that defines a service we wish to implement. Thus, a protocol in the ideal model is trivial and consists of machines that forwards any input to the ideal functionality, and gives any output from the ideal functionality as output. The ideal adversary decides when messages are delivered from the ideal functionality but it cannot see any contents. To be able to seamlessly move from a real model to an ideal model there are many hybrid models. A protocol running in a hybrid model is a list of interactive Turing machines that has access to some set of ideal functionalities.

The definition of security is based on the simulation paradigm. A protocol is said to securely realize an ideal functionality if for any real adversary in the real model, there is an ideal adversary in the ideal model that has the same advantage. In contrast to classical definitions the distinguisher is present during the execution and may influence the adversary based on part of the transcript.

The definition of security allows secure composition of protocols, i.e. given a protocol secure in a hybrid model, and protocols that securely realize all ideal functionalities in use, we have a natural protocol that turns out to be secure.

Universally composable security may be viewed both as a tool for modular analysis of protocols and as an argument of the security of a protocol run in a turbulent environment. Appendix I contains details for how the communication model is defined.

The notion of a communication model, $\mathcal{C}_\mathcal{I}$, used below is not explicit in Canetti [8]. It works as a router between participants and between participants and ideal functionalities. Given the input $((A_1, B_1, C_1, \ldots), \ldots, (A_s, B_s, C_s, \ldots)$ it interprets A_j as the receiver of $(B_j, C_j, \ldots)$. The adversary cannot read the correspondence with ideal functionalities, but it has full control over when a message is delivered.

Our results hold for both blocking and non-blocking adversaries, where a blocking adversary is allowed to block the delivery of a message indefinitely.

Definition 1. *We define $\mathcal{M}_l$ to be the set of* static adversaries *that corrupt less than l out of k participants of the mix-server type, and arbitrarily many participants of the sender type.*

Throughout we implicitly assume that a message handed to an ideal functionality that is not on the form prescribed in its definition is returned to the sender immediately. In particular this includes verifying membership in G_q when appropriate. We use the same convention for definitions of protocols.

3 The Ideal Mix-Net

Although other definitions of security of mix-nets have been proposed, the most natural definition is given by Wikström [53] in the UC-framework. He formalizes a trusted party that waits for messages from senders, and then when a majority of the mix-servers request it, outputs these messages but in lexicographical order. For simplicity it accepts only one input from each sender. We prove security relative this functionality.

Functionality 1 (Mix-Net (cf. [53])). The ideal functionality for a *mix-net*, $\mathcal{F}_{\mathrm{MN}}$, running with mix-servers $M_1, \ldots, M_k$, senders $P_1, \ldots, P_N$, and ideal adversary $\mathcal{S}$ proceeds as follows

1. Initialize a list $L = \emptyset$, and set $J_P = \emptyset$ and $J_M = \emptyset$.
2. Suppose $(P_i, \texttt{Send}, m_i)$, $m_i \in G_q$, is received from $\mathcal{C}_\mathcal{I}$. If $i \notin J_P$, set $J_P \leftarrow J_P \cup \{i\}$, and append m_i to the list L. Then hand $(\mathcal{S}, P_i, \texttt{Send})$ to $\mathcal{C}_\mathcal{I}$.
3. Suppose $(M_j, \texttt{Run})$ is received from $\mathcal{C}_\mathcal{I}$. Set $J_M \leftarrow J_M \cup \{j\}$. If $|J_M| > k/2$, then sort the list L lexicographically to form a list L', and hand $((\mathcal{S}, M_j, \texttt{Output}, L'), \{(M_l, \texttt{Output}, L')\}_{l=1}^k)$ to $\mathcal{C}_\mathcal{I}$. Otherwise, hand $\mathcal{C}_\mathcal{I}$ the list $(\mathcal{S}, M_j, \texttt{Run})$.

4 A Sender Verifiable El Gamal Based Mix-Net

In recent El Gamal based mix-nets the mix-servers form a chain, and each mix-server randomly permutes, partially decrypts, and *re-encrypts* the output of the previous mix-server. In older constructions decryption is instead carried out jointly at the end of the chain. Our construction is different in that each mix-server *partially decrypts* and *sorts* the output of the previous mix-server. Thus, no cryptotext is re-encrypted and the permutation is not random, but determined by the lexicographical order of the cryptotexts.

Let us consider why re-encryption is often considered necessary. In several previous mix-nets each mix-server M_j holds a secret key $x_j \in \mathbb{Z}_q$ corresponding to a public key $y_j = g^{x_j}$. A joint public key $y = \prod_{j=1}^{k} y_j$ is used by a sender S_i to compute a cryptotext $(u_{0,i}, v_{0,i}) = (g^{r_i}, y^{r_i} m_i)$ of a message m_i for a random $r_i \in \mathbb{Z}_q$. The mix-servers take turns and compute

$$(u_{j,i}, v_{j,i})_{i=1}^N = \left(g^{s_{j,i}} u_{j-1,\pi_j(i)}, \left(\prod_{l=j+1}^{k} y_l \right)^{s_{j,i}} v_{j-1,\pi_j(i)} u_{j-1,\pi_j(i)}^{-x_j} \right)_{i=1}^N ,$$

for random $s_{j,i} \in \mathbb{Z}_q$ and $\pi_j \in \Sigma_N$, i.e., each mix-server permutes, partially decrypts and re-encrypts its input. In the end $(v_{k,i})_{i=1}^N = (m_{\pi(i)})_{i=1}^N$ for some random joint permutation π. The reason that re-encryption is necessary with this type of scheme is that otherwise the first component $u_{0,i}$ of each cryptotext remains unchanged during the transformation, which allows anybody to break the anonymity of all senders. For the older type of construction it is obvious why re-encryption is necessary.

4.1 Our Modification

We modify the El Gamal cryptosystem to ensure that also the first component $u_{j-1,i}$ is changed during partial decryption. Each mix-server is given a secret key $(w_j, x_j) \in \mathbb{Z}_q^2$ and a corresponding public key $(z_j, y_j) = (g^{w_j}, g^{x_j})$. To partially decrypt and permute its input it computes

$$(u_{j-1,i}^{1/w_j}, v_{j-1,i}/u_{j-1,i}^{x_j/w_j})_{i=1}^N \quad , \tag{1}$$

from L_{j-1}, and sorts the result lexicographically. The result is denoted by $L_j = (u_{j,i}, v_{j,i})_{i=1}^N$. Note that both components of each cryptotext are transformed using the secret key of the mix-server. For this transformation to make any sense we must also modify the way the joint key is formed. We define

$$(Z_{k+1}, Y_{k+1}) = (g, 1) \quad \text{and} \quad (Z_j, Y_j) = (Z_{j+1}^{w_j}, Y_{j+1} Z_{j+1}^{x_j}) \ . \tag{2}$$

A sender encrypts its message using the public key (Z_1, Y_1), i.e., $(u_{0,i}, v_{0,i}) = (Z_1^{r_i}, Y_1^{r_i} m_i)$ for some random r_i. The structure of the keys are chosen such that a cryptotext on the form $(u_{j-1,i}, v_{j-1,i}) = (Z_j^{r_i}, Y_j^{r_i} m_i)$ given as input to mix-server M_j satisfies

$$\begin{aligned}(u_{j-1,i}^{1/w_j}, v_{j-1,i}/u_{j-1,i}^{x_j/w_j}) &= (Z_j^{r_i/w_j}, Y_j^{r_i}/Z_j^{r_i x_j/w_j} m_i) \\ &= ((Z_j^{1/w_j})^{r_i}, (Y_j/Z_j^{x_j/w_j})^{r_i} m_i) = (Z_{j+1}^{r_i}, Y_{j+1}^{r_i} m_i) \ .\end{aligned}$$

Thus, each mix-server M_j transforms a cryptotext $(u_{j-1,i}, v_{j-1,i})$ encrypted with the public key (Z_j, Y_j) into a cryptotext $(u_{j,i}, v_{j,i})$ encrypted with the public key (Z_{j+1}, Y_{j+1}). Note that $\mathrm{Sort}(\{v_{k,i}\}_{i=1}^N) = \mathrm{Sort}(\{m_i\}_{i=1}^N)$, since $Y_{k+1} = 1$.

There are several seemingly equivalent ways to set up the scheme, but some of these do not allow a reduction of the security of the mix-net to the DDH-assumption. In fact the relation in Equation (1) is carefully chosen to allow such a reduction.

4.2 Sender Verifiability

An important consequence of our modification is that a sender can compute $(Z_{j+1}^{r_i}, Y_{j+1}^{r_i} m_i)$ and verify that this pair is contained in L_j for $j = 1, \ldots, k$. Furthermore, if this is not the case the sender can easily prove to any outsider that its message was tampered with. We call this *sender verifiability*, since it allows a sender to verify that its cryptotext is processed correctly by the mix-servers. This is not a new property. In fact Chaum's original construction [10] has this property, but our construction is the first provably secure scheme with this property.

We think that sender verifiability is an important property that deserves more attention. The verification process is unconditional and easily explained to anybody with only a modest background in mathematics, and a verification program can be implemented with little skills in programming. This means that in the main application of mix-nets, electronic elections, a sender can convince herself that her vote was processed correctly without a deep understanding of cryptography.

The reader may worry that this allows a voter to point out its vote to a coercer. This is the case, but the sender can do this in previous mix-nets as well by pointing at its message in the original list L_0 of cryptotexts and revealing the randomness used during encryption, so this problem is not specific to our scheme. Furthermore, our scheme becomes coercion-free whenever the sender does not know the randomness of its cryptotext, as other El Gamal based mix-nets, but sender verifiability is then lost.

4.3 A Technical Advantage

There is also an important technical consequence of the lack of re-encryption in the mixing process. The witness of our shuffle relation consists of a pair (w_j, x_j), which makes it easy to turn our proof of knowledge into a secure realization of the ideal functionality $\mathcal{F}_{\mathrm{ZK}}^{R_{\mathrm{DP}}}$. This should be contrasted with all previous shuffle relations, where the witness contains a long list of random exponents used to re-encrypt the input that must somehow be extracted by the ideal adversary in the UC-setting.

A potential alternative to our approach is to formalize the proof of a shuffle as a proof of membership [41] in the UC-framework. However, a proof of membership is not sufficient for the older constructions where decryption is carried out jointly at the end of the mixing chain. The problem is that the adversary could corrupt the last mix-server M_k and instruct it to output L_0 instead of a re-encryption and permutation of L_{k-1}. This would obviously break the anonymity of all senders. The malicious behavior is not detected, since the ideal proof of membership only expects an element in the language and no witness from corrupted parties, and L_0 is a re-encryption and permutation of L_{k-1}. Interestingly, it seems that the adversary cannot attack the real protocol if the proof of membership of a correct shuffle is implemented using a proof of knowledge in the classical sense.

It is an open question if a proof of membership suffices for mix-nets where each mix-server *partially decrypts* and then re-encrypts and permutes its input. Another more general open question is if it is possible to formalize some sort of zero-knowledge proof in the UC-framework that avoids the need for straight-line extraction of the witness, but enforces that a witness can be extracted using rewinding.

4.4 Preliminaries

We describe the mix-net in a hybrid model as defined in the UC-framework. This means that the mix-servers and senders have access to a set of ideal functionalities introduced in this section.

We assume the existence of an authenticated bulletin board. All parties can write to it, but no party can erase any message from it. In Section H.1 we reproduce from [53] the definition of the ideal functionality $\mathcal{F}_{\mathrm{BB}}$. We also assume an ideal functionality corresponding to the key set-up sketched in Section 4.1. This is given below.

Functionality 2 (Special El Gamal Secret Key Sharing). The ideal *Special El Gamal Secret Key Sharing over G_q*, $\mathcal{F}_{\mathrm{SKS}}$, with mix-servers $M_1, \ldots, M_k$, senders $P_1, \ldots, P_N$, and ideal adversary $\mathcal{S}$.

1. Initialize sets $J_j = \emptyset$ for $j = 0, \ldots, k$.
2. Until $|J_0| = k$ wait for $(M_j, \mathtt{MyKey}, w_j, z_j, x_j, y_j)$ from $\mathcal{C}_{\mathcal{I}}$ such that $w_j, x_j \in \mathbb{Z}_q$, $z_j = g^{w_j}$, $y_j = g^{x_j}$, and $j \notin J_0$. Set $J_0 \leftarrow J_0 \cup \{j\}$ and hand $(\mathcal{S}, \mathtt{PublicKey}, M_j, w_j, z_j)$ to $\mathcal{C}_{\mathcal{I}}$.
3. Set $(Z_{k+1}, Y_{k+1}) = (g, 1)$ and $(Z_j, Y_j) = (Z_{j+1}^{w_j}, Y_{j+1} Z_{j+1}^{x_j})$. Then hand $((\mathcal{S}, \mathtt{PublicKeys}, (Z_j, Y_j, z_j, y_j)_{j=1}^{k}), \{(P_i, \mathtt{PublicKeys}, (Z_j, Y_j, z_j, y_j)_{j=1}^{k})\}_{i=1}^{N}, \{(M_l, \mathtt{Keys}, w_l, x_l, (Z_j, Y_j, z_j, y_j)_{j=1}^{k})\}_{l=1}^{k})$ to $\mathcal{C}_{\mathcal{I}}$.
4. If $(M_j, \mathtt{Recover}, M_l)$ is received from $\mathcal{C}_{\mathcal{I}}$, set $J_l \leftarrow J_l \cup \{j\}$. If $|J_l| > k/2$, then hand $((\mathcal{S}, \mathtt{Recovered}, M_l, w_l, x_l), \{(M_j, \mathtt{Recovered}, M_l, w_l, x_l)\}_{j=1}^{k})$ to $\mathcal{C}_{\mathcal{I}}$, and otherwise hand $(\mathcal{S}, M_j, \mathtt{Recover}, M_l)$ to $\mathcal{C}_{\mathcal{I}}$.

The above functionality can be securely realized by letting each mix-server secret share its secret key using Feldman's [14] verifiable secret sharing scheme. Note that the functionality explicitly allows corrupted mix-servers to choose their keys in a way that depends on the public keys of uncorrupted mix-servers. The special joint keys would then be computed iteratively using Equation (2), and during this process each mix-server would prove that it does this correctly using standard methods.

Each mix-server partially decrypts each cryptotext and sorts the resulting cryptotexts. Thus, proving correct behavior corresponds to proving knowledge of a secret key (w, x) such that the cryptotexts (u_i, v_i) input to a mix-server are related to the cryptotexts (u_i', v_i') it outputs by the following relation.

Definition 2 (Knowledge of Correct Decryption-Permutation). *Define for each N a relation $R_{\mathrm{DP}} \subset (G_q^3 \times G_q^{2N} \times G_q^{2N}) \times (\mathbb{Z}_q \times \mathbb{Z}_q)$, by*

$$((g, z, y, \{(u_i, v_i)\}_{i=1}^N, \{(u_i', v_i')\}_{i=1}^N), (w, x)) \in R_{\mathrm{DP}}$$

precisely when $z = g^w$, $y = g^x$ and $(u_i', v_i') = (u_{\pi(i)}^{1/w}, v_{\pi(i)} u_{\pi(i)}^{-x/w})$ for $i = 1, \ldots, N$ and some permutation $\pi \in \Sigma_N$ such that the list $\{(u_i', v_i')\}_{i=1}^N$ is sorted lexicographically.

To avoid a large class of "relation attacks" [47,46,52] no sender can be allowed to construct a cryptotext of a message related to the message encrypted by some other sender. To ensure this each sender is required to prove knowledge of the randomness it uses to form its cryptotexts. This corresponds to the following relation.

Definition 3 (Knowledge of Cleartext). *Define a relation $R_{\mathrm{C}} \subset G_q^4 \times \mathbb{Z}_q$ by $((Z, Y, u, v), r) \in R_{\mathrm{C}}$ precisely when $\log_Z u = r$.*

Formally, we need a secure realization of the following functionality with the above relations.

Functionality 3 (Zero-Knowledge Proof of Knowledge). Let $\mathcal{L}$ be a language given by a binary relation R. The ideal *zero-knowledge proof of knowledge* functionality $\mathcal{F}_{\mathrm{ZK}}^R$ of a witness w to an element $x \in \mathcal{L}$, running with provers $P_1, \ldots, P_N$, and verifiers $M_1, \ldots, M_k$, proceeds as follows.

1. Upon receipt of $(P_i, \texttt{Prover}, x, w)$ from $\mathcal{C}_{\mathcal{I}}$, store w under the tag (P_i, x), and hand $(\mathcal{S}, P_i, \texttt{Prover}, x, R(x, w))$ to $\mathcal{C}_{\mathcal{I}}$. Ignore further messages from P_i.
2. Upon receipt of $(M_j, \texttt{Question}, P_i, x)$ from $\mathcal{C}_{\mathcal{I}}$, let w be the string stored under the tag (P_i, x) (the empty string if nothing is stored), and hand $((\mathcal{S}, M_j, \texttt{Verifier}, P_i, x, R(x, w)), (M_j, \texttt{Verifier}, P_i, R(x, w))$ to $\mathcal{C}_{\mathcal{I}}$.

In [53] a secure realization π_{C} of $\mathcal{F}_{\mathrm{ZK}}^{R_{\mathrm{C}}}$ is given, under the DDH-assumption, which is secure against $\mathcal{M}_{k/2}$-adversaries. The functionality $\mathcal{F}_{\mathrm{ZK}}^{R_{\mathrm{DP}}}$ is securely realized in Section 6.

4.5 The Mix-Net

We now give the details of our mix-net. It executes in a hybrid model with access to the ideal functionalities described above.

Protocol 1 (Mix-Net). The mix-net protocol $\pi_{\mathrm{MN}} = (P_1, \ldots, P_N, M_1, \ldots, M_k)$ consists of senders P_i, and mix-servers M_j.

SENDER P_i. Each sender P_i proceeds as follows.

1. Wait for $(\texttt{PublicKeys}, (Z_j, Y_j, z_j, y_j)_{j=1}^k)$ from $\mathcal{F}_{\mathrm{SKS}}$.
2. Wait for an input $(\texttt{Send}, m_i)$, $m_i \in G_q$. Then choose $r_i \in \mathbb{Z}_q$ randomly and compute $(u_i, v_i) = E_{(Z_1,Y_1)}(m_i, r_i) = (Z_1^{r_i}, Y_1^{r_i} m_i)$.
3. Hand $(\texttt{Prover}, (Z_1, Y_1, u_i, v_i), r_i)$ to $\mathcal{F}_{\mathrm{ZK}}^{R_{\mathrm{C}}}$.
4. Hand $(\texttt{Write}, (u_i, v_i))$ to $\mathcal{F}_{\mathrm{BB}}$.

Mix-Server M_j. Each mix-server M_j proceeds as follows.

1. Choose $w_j, x_j \in \mathbb{Z}_q$ randomly and hand $(\texttt{MyKey}, w_j, z_j, x_j, y_j)$ to $\mathcal{F}_{\mathrm{SKS}}$.
2. Wait for $(\texttt{Keys}, (w_j, x_j), (Z_j, Y_j, z_j, y_j)_{j=1}^k)$ from $\mathcal{F}_{\mathrm{SKS}}$, where $w_j, x_j \in \mathbb{Z}_q$ and $Z_j, Y_j, z_j, y_j \in G_q$.
3. Wait for an input $(\texttt{Run})$, and then send $(\texttt{Write}, \texttt{Run})$ to $\mathcal{F}_{\mathrm{BB}}$.
4. Wait until more than $k/2$ different mix-servers have written $\texttt{Run}$ on $\mathcal{F}_{\mathrm{BB}}$, and let the last entry of this type be $(c_{\texttt{Run}}, M_i, \texttt{Run})$.
5. Form the list $L_* = \{(u_\gamma, v_\gamma)\}_{\gamma \in I_*}$, for some index set I_*, by choosing for $\gamma = 1, \ldots, N$ the entry $(c, P_\gamma, (u_\gamma, v_\gamma))$ with the smallest $c < c_{\mathrm{run}}$ such that $u_\gamma, v_\gamma \in G_q$, if present.
6. For each $\gamma \in I_*$ do the following,
   (a) Hand $(\texttt{Question}, P_\gamma, (Z_1, Y_1, u_\gamma, v_\gamma))$ to $\mathcal{F}_{\mathrm{ZK}}^{R_{\mathrm{C}}}$.
   (b) Wait for $(\texttt{Verifier}, P_\gamma, b_\gamma)$ from $\mathcal{F}_{\mathrm{ZK}}^{R_{\mathrm{C}}}$.
   Then form $L_0 = \{(u_{0,i}, v_{0,i})\}_{i=1}^{N'}$ consisting of pairs (u_γ, v_γ) such that $b_\gamma = 1$.
7. For $l = 1, \ldots, k$ do
   (a) If $l \neq j$, then do
      i. Wait until an entry $(c, M_l, (\texttt{List}, L_l))$ appears on $\mathcal{F}_{\mathrm{BB}}$, where L_l is on the form $\{(u_{l,i}, v_{l,i})\}_{i=1}^{N'}$ for $u_{l,i}, v_{l,i} \in G_q$.
      ii. Hand $(\texttt{Question}, M_l, (g, z_l, y_l, L_{l-1}, L_l))$ to $\mathcal{F}_{\mathrm{ZK}}^{R_{\mathrm{DP}}}$, and wait for $(\texttt{Verifier}, M_l, b_l)$ from $\mathcal{F}_{\mathrm{ZK}}^{R_{\mathrm{DP}}}$.
      iii. If $b_l = 0$, then hand $(\texttt{Recover}, M_l)$ to $\mathcal{F}_{\mathrm{SKS}}$, and wait for $(\texttt{Recovered}, M_l, (w_l, x_l))$ from $\mathcal{F}_{\mathrm{SKS}}$. Then compute

$$L_l = \{(u_{l,i}, v_{l,i})\}_{i=1}^{N'} = \mathrm{Sort}(\{(u_{l-1,i}^{1/w_l}, v_{l-1,i} u_{l-1,i}^{-x_l/w_l})\}_{i=1}^{N'}) \enspace .$$

   (b) If $l = j$, then compute

$$L_j = \{(u_{j,i}, v_{j,i})\}_{i=1}^{N'} = \mathrm{Sort}(\{(u_{j-1,i}^{1/w_j}, v_{j-1,i} u_{j-1,i}^{-x_j/w_j})\}_{i=1}^{N'}) \enspace ,$$

   Finally hand $(\texttt{Prover}, (g, z_j, y_j, L_{j-1}, L_j), (w_j, x_j))$ to $\mathcal{F}_{\mathrm{ZK}}^{R_{\mathrm{DP}}}$, and hand $(\texttt{Write}, (\texttt{List}, L_j))$ to $\mathcal{F}_{\mathrm{BB}}$.
8. Output $(\texttt{Output}, \mathrm{Sort}(\{v_{k,i}\}_{i=1}^{N'}))$.

Theorem 1. *The ideal functionality $\mathcal{F}_{\mathrm{MN}}$ is securely realized by π_{MN} in the $(\mathcal{F}_{\mathrm{BB}}, \mathcal{F}_{\mathrm{SKS}}, \mathcal{F}_{\mathrm{ZK}}^{R_{\mathrm{C}}}, \mathcal{F}_{\mathrm{ZK}}^{R_{\mathrm{DP}}})$-hybrid model with respect to $\mathcal{M}_{k/2}$-adversaries under the DDH-assumption in G_q.*

The number of mix-servers actually doing any shuffling of cryptotexts can be reduced to $\lceil k/2 \rceil$ without any loss in security, thus reducing the overall complexity. We state the protocol in a symmetrical way for sake of simplicity.

5 A New Efficient Proof of a Shuffle

We want to securely realize the ideal functionality $\mathcal{F}_{\mathrm{ZK}}^{R_{\mathrm{DP}}}$. It turns out that a useful step in this direction is to construct a statistical zero-knowledge proof for the relation R_{DP}, i.e., a proof of the decryption-permutation shuffle. First we explain the key ideas in our approach. Then we give a detailed description of our protocol. Finally, we explain how it can be turned into a public coin protocol.

5.1 Our Approach

The protocol for proving the relation R_{DP} is complex, but the underlying ideas are simple. To simplify the exposition we follow Neff [39,40] and consider the problem of proving that a list of elements in G_q has been exponentiated and permuted. More precisely, let $y, u_1, \ldots, u_N, u'_1, \ldots, u'_N \in G_q$ be defined by $y = g^x$ and $u'_i = u^x_{\pi(i)}$ for a permutation π. Only the prover knows x and π and he must show that the elements satisfy such a relation. We also omit numerous technical details. In particular we remove several blinding factors, hence the protocols are not zero-knowledge as sketched here.

Extraction Using Linear Independence. The verifier chooses a list $P = (p_i)_{i=1}^N \in \mathbb{Z}_q^N$ of random primes and computes $U = \prod_{i=1}^N u_i^{p_i}$. Then it requests that the prover computes $U' = \prod_{i=1}^N (u'_i)^{p_{\pi(i)}}$, proves that $U' = U^x$ and that it knows a permutation π such that $U' = \prod_{i=1}^N (u'_i)^{p_{\pi(i)}}$.

The idea is then that if a prover succeeds in doing this it can be rewound and run several times with different random vectors P_j, giving different U_j and U'_j, until a set $P_1, \ldots, P_N$ of linearly independent vectors in $\mathbb{Z}_q^N$ are found. Linear independence implies that there are coefficients $a_{l,j}$ such that $\sum_{j=1}^N a_{l,j} P_j$ equals the lth unity vector e_l, i.e., the vector with a one in the lth position and all other elements zero. We would then like to conclude that

$$u_l^x = \left(\prod_{j=1}^N U_j^{a_{l,j}} \right)^x = \prod_{j=1}^N (U'_j)^{a_{l,j}} = \prod_{j=1}^N \left(\prod_{i=1}^N (u'_i)^{p_{j,\pi^{-1}(i)}} \right)^{a_{l,j}} = u'_{\pi(l)} \ , \quad (3)$$

since that would imply that the elements satisfy the shuffle-relation.

Proving a Permutation of Prime Exponents. The prover can use standard techniques to prove knowledge of integers $\rho_1, \ldots, \rho_N$ such that $U' = \prod_{i=1}^N (u'_i)^{\rho_i}$, but it must also prove that $\rho_i = p_{\pi(i)}$ for some permutation π.

Suppose that $\prod_{i=1}^N p_i = \prod_{i=1}^N \rho_i$ over $\mathbb{Z}$. Then unique factorization in $\mathbb{Z}$ implies that each ρ_i equals some product of the p_i and -1. If in addition we demand that no $\rho_i = \pm 1$, then every product must contain exactly one prime factor.[1] Thus, $\rho_i = \pm p_{\pi(i)}$ for some permutation π. If we also have $\sum_{i=1}^N p_i = \sum_{i=1}^N \rho_i$, then we must clearly have $\rho_i = p_{\pi(i)}$.

We observe that proving the above is relatively simple over a group of unknown order such as the group $\mathrm{QR}_{\mathbf{N}}$ of squares modulo an RSA modulus $\mathbf{N}$. The prover forms commitments

$$\mathbf{b}_0 = \mathbf{g} \ , \quad (\mathbf{b}_i, \mathbf{b}_i')_{i=1}^N = (\mathbf{h}^{t_i} \mathbf{b}_{i-1}^{p_{\pi(i)}}, \mathbf{h}^{t_i'} \mathbf{g}^{p_{\pi(i)}})_{i=1}^N \ ,$$

with random t_i and t_i' and proves, using standard methods, knowledge of ρ_i, τ_i, τ_i' such that

$$U' = \prod_{i=1}^N (u_i')^{\rho_i} \ , \quad \mathbf{b}_i = \mathbf{h}^{\tau_i} \mathbf{b}_{i-1}^{\rho_i} \ , \quad \text{and} \quad \mathbf{b}_i' = \mathbf{h}^{\tau_i'} \mathbf{g}^{\rho_i} \ . \tag{4}$$

Note that $\mathbf{b}_N = \mathbf{h}^\tau \mathbf{g}^{\prod_{i=1}^N \rho_i}$ for some τ, so the verifier can check that $\prod_{i=1}^N \rho_i = \prod_{i=1}^N p_i$ by asking the prover to show that it knows τ such that $\mathbf{b}_N / \mathbf{g}^{\prod_{i=1}^N p_i} = \mathbf{h}^\tau$. Since $\prod_{i=1}^N \mathbf{b}_i' = \mathbf{h}^{\tau'} \mathbf{g}^{\sum_{i=1}^N \rho_i}$ for a $\tau' = \sum_{i=1}^N \tau_i'$, the verifier can check that $\sum_{i=1}^N \rho_i = \sum_{i=1}^N p_i$ by asking the prover to show that it knows τ' such that $\prod_{i=1}^N \mathbf{b}_i' / \mathbf{g}^{\sum_{i=1}^N p_i} = \mathbf{h}^{\tau'}$.

Fixing a Permutation and Avoiding $\boldsymbol{\rho_i = \pm 1}$. In Equation (3) above it is assumed that a fixed permutation π is used for all prime vectors $P_1, \ldots, P_N$. Unfortunately, this is not necessarily the case unless we add a component to the protocol, e.g., the permutation used in the jth proof could depend on j and we should really write π_j.

To solve this technical problem we force the prover to commit to a fixed permutation π before it receives the prime vector P. The commitment is on the form $(w_i)_{i=1}^N = (g^{r_i'} g_{\pi^{-1}(i)})_{i=1}^N$. The verifier then computes $W = \prod_{i=1}^N w_i^{p_i}$ and the prover proves that $W = g^{r'} \prod_{i=1}^N g_i^{\rho_i}$ in addition to Equations (4). From the linearly independent prime vectors $P_1, \ldots, P_N$ we extract ρ_i' and $\rho_{i,l}'$ such that $w_i = g^{\rho_i'} \prod_{l=1}^N g_l^{\rho_{i,l}'}$, regardless of if all π_j are equal or not (and even regardless of if the extracted ρ_i are even primes).

Consider what happens if we use another prime vector $P_{N+1} = (p_1, \ldots, p_N)$ (obviously then contained in the space spanned by the previous vectors) from an additional interaction using the same commitments $(w_1, \ldots, w_N)$ as those used in the previous N extractions. Then we have

$$\left(g^{\rho_i'} \prod_{l=1}^N g_l^{\rho_{i,l}'} \right)^{p_i} = \prod_{i=1}^N w_i^{p_i} = g^{r'} \prod_{i=1}^N g_i^{\rho_i} \ .$$

[1] In the original version of this paper it was claimed that the protocol directly guaranteed that $\rho_i \in [-2^K + 1, 2^K - 1]$, which would also give that each ρ_i contains exactly one prime factor, but this argument is flawed. The flaw is discussed in Appendix K and the next modification of the original paper is marked by Footnote 2 on Page 33.

This gives a non-trivial representation of 1 in G_q unless

$$(p_1, \ldots, p_N) \times (\rho'_{i,l}) = (\rho_1, \ldots, \rho_N) \ ,$$

where $\times$ denotes matrix multiplication over $\mathbb{Z}_q$. It is obvious that for any fixed matrix $(\rho'_{i,l})$ and random $p_1, \ldots, p_N$ the probability that there exists $(\rho_1, \ldots, \rho_N)$ with some $\rho_i = \pm 1$ satisfying the matrix equation is negligible. Thus, we may safely assume that no $\rho_i = \pm$, which by the previous section means that the ρ_i equal the primes p_i in some permuted order. It is not hard to see that for a fixed non-permutation matrix $(\rho'_{i,l})$ the probability over p_i that there exists some permutation π such that $\rho_i = p_{\pi(i)}$ satisfies the matrix equation is negligible.

5.2 An Honest Verifier Statistical Zero-Knowledge Computationally Convincing Proof of Knowledge of a Decryption-Permutation

In this section we describe our proof of a shuffle in detail. Although we consider a decrypt-permutation relation, our approach can be generalized to a proof of a shuffle for the other shuffle relations considered in the literature. For completeness we detail such protocols in Section F.

We introduce several security parameters. We use K_1 to denote the number of bits in q, the order of the group G_q, and similarly K_2 to denote the number of bits in the RSA-modulus $\mathbf{N}$. We use K_3 to denote the number bits used in the random primes mentioned above. At some point in the protocol the verifier hands a challenge to the prover. We use K_4 to denote the number of bits in this challenge. At several points exponents must be padded with random bits to achieve statistical zero-knowledge. We use K_5 to denote the number additional random bits used to do this. We assume that the security parameters are chosen such that $K_3 + K_4 + K_5 < K_1, K_2$, and $K_5 < K_3 - 2$. Below the protocol we explain how the informal description above relates to the different components of the protocol.

Protocol 2 (Proof of Decryption-Permutation). The common input consists of an RSA modulus $\mathbf{N}$ and $\mathbf{g}, \mathbf{h} \in \mathrm{QR}_{\mathbf{N}}$, generators $g, g_1, \ldots, g_N \in G_q$, a public key $(z, y) \in G_q^2$, and two lists $L = (u_i, v_i)_{i=1}^N$ and $L' = (u'_i, v'_i)_{i=1}^N$ in G_q^{2N}. The private input to the prover consists of $(w, x) \in \mathbb{Z}_q^2$ such that $(z, y) = (g^w, g^x)$ and $(u'_i, v'_i) = (u_{\pi(i)}^{1/w}, v_{\pi(i)}/u_{\pi(i)}^{x/w})$ for a permutation $\pi \in \Sigma_N$ such that L' is lexicographically sorted.

1. The prover chooses $r'_i \in \mathbb{Z}_q$ randomly, computes $(w_i)_{i=1}^N = (g^{r'_i} g_{\pi^{-1}(i)})_{i=1}^N$, and hands $(w_i)_{i=1}^N$ to the verifier.
2. The verifier chooses random primes $p_1, \ldots, p_N \in [2^{K_3-1}, 2^{K_3} - 1]$, and hands $(p_i)_{i=1}^N$ to the prover.
3. Both parties compute $(U, V, W) = (\prod_{i=1}^N u_i^{p_i}, \prod_{i=1}^N v_i^{p_i}, \prod_{i=1}^N w_i^{p_i})$.
4. The prover chooses the following elements randomly $k_1, k_2, k_3, k_4, k_5 \in \mathbb{Z}_q$, $l_1, \ldots, l_7, l_{r'}, l_{1/w}, l_{x/w}, l_w, l_x \in \mathbb{Z}_q$, $t_i, t'_i \in [0, 2^{K_2+K_5} - 1]$, $s_i, s'_i \in [0, 2^{K_2+K_4+2K_5} - 1]$, $r_i \in [0, 2^{K_3+K_4+K_5} - 1]$ for $i = 1, \ldots, N$,

$s \in [0, 2^{K_2+NK_3+K_4+K_5+\log_2 N}-1]$, and $s' \in [0, 2^{K_2+K_5+\log_2 N}-1]$. Then the prover computes

$$
\begin{aligned}
(b_1, b_2) &= (g^{k_1} U^{1/w}, g^{k_2} U^{x/w}) && (5)\\
(b_3, b_4, b_5) &= (g_1^{k_3} g^{1/w}, g_1^{k_4} b_3^x, g_1^{k_5} b_3^w) && (6)\\
(\beta_1, \beta_2) &= \left(g^{l_1} U^{l_{1/w}}, g^{l_2} U^{l_{x/w}}\right) && (7)\\
(\beta_3, \beta_4) &= (g_1^{l_3} g^{l_{1/w}}, g_1^{l_6} g^{l_{x/w}}) && (8)\\
(\beta_5, \beta_6, \beta_7, \beta_8, \beta_9) &= (g_1^{l_4} b_3^{l_x}, g^{l_x}, g_1^{l_5} b_3^{l_w}, g^{l_w}, g_1^{l_7}) && (9)\\
(\alpha_1, \alpha_2, \alpha_3) &= \left(g^{l_1} \prod_{i=1}^{N} (u_i')^{r_i}, g^{-l_2} \prod_{i=1}^{N} (v_i')^{r_i}, g^{l_{r'}} \prod_{i=1}^{N} g_i^{r_i}\right) && (10)\\
\mathbf{b}_0 &= \mathbf{g} && (11)\\
(\mathbf{b}_i, \mathbf{b}_i')_{i=1}^N &= (\mathbf{h}^{t_i} \mathbf{b}_{i-1}^{p_{\pi(i)}}, \mathbf{h}^{t_i'} \mathbf{g}^{p_{\pi(i)}})_{i=1}^N && (12)\\
(\boldsymbol{\gamma}_i, \boldsymbol{\gamma}_i')_{i=1}^N &= (\mathbf{h}^{s_i} \mathbf{b}_{i-1}^{r_i}, \mathbf{h}^{s_i'} \mathbf{g}^{r_i})_{i=1}^N && (13)\\
(\boldsymbol{\gamma}, \boldsymbol{\gamma}') &= (\mathbf{h}^s, \mathbf{h}^{s'}) \ , && (14)
\end{aligned}
$$

and $((b_i)_{i=1}^5, (\beta_i)_{i=1}^9, (\alpha_1, \alpha_2, \alpha_3), (\mathbf{b}_i, \mathbf{b}_i')_{i=1}^N, (\boldsymbol{\gamma}_i, \boldsymbol{\gamma}_i')_{i=1}^N, (\boldsymbol{\gamma}, \boldsymbol{\gamma}'))$ is handed to the verifier.

5. The verifier chooses $c \in [2^{K_4-1}, 2^{K_4}-1]$ randomly and hands c to the prover.
6. Define $t = t_N + p_{\pi(N)}(t_{N-1} + p_{\pi(N-1)}(t_{N-2} + p_{\pi(N-2)}(t_{N-3} + p_{\pi(N-3)}(\ldots))))$, $t' = \sum_{i=1}^N t_i'$, $r' = \sum_{i=1}^N r_i' p_i$, $k_6 = k_4 + k_3 x$, and $k_7 = k_5 + k_3 w$. The prover computes

$$
\begin{aligned}
(f_i)_{i=1}^7 &= (ck_i + l_i)_{i=1}^7 && \bmod q\\
f_{1/w} &= c/w + l_{1/w} && \bmod q\\
f_{x/w} &= cx/w + l_{x/w} && \bmod q\\
f_w &= cw + l_w && \bmod q\\
f_x &= cx + l_x && \bmod q\\
f_{r'} &= cr' + l_{r'} && \bmod q\\
(e_i, e_i')_{i=1}^N &= (ct_i + s_i, ct_i' + s_i')_{i=1}^N && \bmod 2^{K_2+K_4+2K_5}\\
(d_i)_{i=1}^N &= (cp_{\pi(i)} + r_i)_{i=1}^N && \bmod 2^{K_3+K_4+K_5}\\
e &= ct + s && \bmod 2^{K_2+NK_3+K_4+K_5+\log_2 N}\\
e' &= ct' + s' && \bmod 2^{K_2+K_5+\log_2 N}
\end{aligned}
$$

Then it hands $(((f_i)_{i=1}^7, f_{1/w}, f_{x/w}, f_w, f_x, f_{r'}), (e_i, e_i')_{i=1}^N, (d_i)_{i=1}^N, (e, e'))$ to the verifier.

7. The verifier checks that $b_i, \beta_i, \alpha_i \in G_q$, and that L' is lexicographically sorted and that

$$(b_1^c\beta_1, b_2^c\beta_2) = (g^{f_1}U^{f_{1/w}}, g^{f_2}U^{f_{x/w}}) \tag{15}$$

$$(b_3^c\beta_3, b_4^c\beta_4) = (g_1^{f_3}g^{f_{1/w}}, g_1^{f_6}g^{f_{x/w}}) \tag{16}$$

$$(b_4^c\beta_5, y^c\beta_6) = (g_1^{f_4}b_3^{f_x}, g^{f_x}) \tag{17}$$

$$(b_5^c\beta_7, z^c\beta_8, (b_5/g)^c\beta_9) = (g_1^{f_5}b_3^{f_w}, g^{f_w}, g_1^{f_7}) \tag{18}$$

$$(b_1^c\alpha_1, (V/b_2)^c\alpha_2, W^c\alpha_3) = \left(g^{f_1}\prod_{i=1}^{N}(u_i')^{d_i}, g^{-f_2}\prod_{i=1}^{N}(v_i')^{d_i}, g^{f_{r'}}\prod_{i=1}^{N}g_i^{d_i}\right) \tag{19}$$

$$(\mathbf{b}_i^c\boldsymbol{\gamma}_i, (\mathbf{b}_i')^c\boldsymbol{\gamma}_i')_{i=1}^N = (\mathbf{h}^{e_i}\mathbf{b}_{i-1}^{d_i}, \mathbf{h}^{e_i'}\mathbf{g}^{d_i})_{i=1}^N \tag{20}$$

$$(\mathbf{g}^{-\prod_{i=1}^N p_i}\mathbf{b}_N)^c\boldsymbol{\gamma} = \mathbf{h}^e \tag{21}$$

$$\left(\mathbf{g}^{-\sum_{i=1}^N p_i}\prod_{i=1}^{N}\mathbf{b}_i'\right)^c\boldsymbol{\gamma}' = \mathbf{h}^{e'} \ . \tag{22}$$

Equations (5)-(9) are used to prove that $(b_1, V/b_2) = (g^{\kappa_1}U^{1/w}, g^{-\kappa_2}V/U^{x/w})$ using standard Schnorr-like proofs of knowledge of logarithms. Equations (12) contain commitments corresponding to those in the outline of our approach. Equations (13) are used to prove knowledge of exponents τ_i, τ_i', ρ_i such that $(\mathbf{b}_i, \mathbf{b}_i') = (\mathbf{h}^{\tau_i}\mathbf{b}_{i-1}^{\rho_i}, \mathbf{h}^{\tau_i'}\mathbf{g}^{\rho_i})$. We remark that the verifier need not check that $\mathbf{b}_i, \mathbf{b}_i', \boldsymbol{\gamma}_i, \boldsymbol{\gamma}_i', \boldsymbol{\gamma}, \boldsymbol{\gamma}' \in \mathrm{QR}_{\mathbf{N}}$ for our analysis to go through. Equations (14) are used to prove that $\prod_{i=1}^N \rho_i = \prod_{i=1}^N p_i$ and $\sum_{i=1}^N \rho_i = \sum_{i=1}^N p_i$, i.e., that ρ_i in fact equals $p_{\pi(i)}$ for some permutation π. Equation (10) is used to prove that $(b_1, V/b_2)$ also equals
$(g^{k_1}\prod_{i=1}^N(u_i^{1/w_j})^{p_i}, g^{-k_2}\prod_{i=1}^N(v_i/u_i^{x_j/w_j})^{p_i})$. If the two ways of writing b_1 and b_2 are combined we have

$$(U^{1/w}, V/U^{x/w}) = \left(\prod_{i=1}^{N}(u_i^{1/w_j})^{p_i}, \prod_{i=1}^{N}(v_i/u_i^{x_j/w_j})^{p_i}\right) \ ,$$

which by the argument in Section 5.1 implies that $((g, z, y, L, L'), (w, x)) \in R_{\mathrm{DP}}$.

5.3 Security Properties

Formally, the security properties of our protocol are captured by the following propositions.

Proposition 1 (Zero-Knowledge). *Protocol 2 is honest verifier statistical zero-knowledge.*

The protocol could be modified by adding a first step, where the verifier chooses $(\mathbf{N}, \mathbf{g}, \mathbf{h})$ and $(g_1, \ldots, g_N)$. This would give a computationally sound proof of knowledge. However, in our application we wish to choose these parameters jointly and only once, and then let the mix-servers execute the proof with these parameters as common inputs. Thus, there may be a negligible portion of the parameters on which the prover can convince the verifier of false statements. Because of this we cannot hope to prove that the protocol is a proof of knowledge in the formal sense. Damgård and Fujisaki [11] introduce the notion of a computationally convincing proof of knowledge to deal with situations like these. We do not use the notion of "computationally convincing proofs" explicitly in our security analysis, but the proposition below implies that our protocol satisfies their definition.

We consider a malicious prover A which is given $\boldsymbol{\Gamma} = (\mathbf{N}, \mathbf{g}, \mathbf{h})$ and $\overline{g} = (g_1, \ldots, g_N)$ as input and run with internal randomness r_{p}. The prover outputs an instance $I_A(\boldsymbol{\Gamma}, \overline{g}, r_{\mathrm{p}})$, i.e., public keys $z, y \in G_q$ and two lists $L, L' \in G_q^{2N}$ and then interacts with the honest verifier on the common input consisting of $(\boldsymbol{\Gamma}, \overline{g}, z, y, L, L')$. Denote by $T_A(\boldsymbol{\Gamma}, \overline{g}, r_{\mathrm{p}}, r_{\mathrm{v}})$ the transcript of such an interaction when the verifier runs with internal randomness r_{v}. Let Acc be the predicate taking a transcript T as input that outputs 1 if the transcript is accepting and 0 otherwise. Let $L_{R_{\mathrm{DP}}}$ be the language corresponding to the decryption-permutation relation R_{DP}. We prove the following proposition.

Proposition 2 (Soundness). *Suppose the strong RSA-assumption and the DL-assumption are true. Then for all polynomial-size circuit families $A = \{A_K\}$ it holds that $\forall c > 0$, $\exists K_0$, such that for $K_1 \geq K_0$*

$$\Pr_{\boldsymbol{\Gamma}, \overline{g}, r_{\mathrm{p}}, r_{\mathrm{v}}}[\mathrm{Acc}(T_A(\boldsymbol{\Gamma}, \overline{g}, r_{\mathrm{p}}, r_{\mathrm{v}})) = 1 \wedge I_A(\boldsymbol{\Gamma}, \overline{g}, r_{\mathrm{p}}) \notin L_{R_{\mathrm{DP}}}] < \frac{1}{{K_1}^c} \ .$$

5.4 Generation of Primes From a Small Number of Public Coins

In our protocol the verifier must generate vectors in $\mathbb{Z}_q^N$ such that each component is a "randomly" chosen prime in $[2^{K_3-1}, 2^{K_3}-1]$. We define a generator PGen that generates prime vectors from public coins. Let $p(n)$ be the smallest prime at least as large as n. Our generator PGen takes as input N random integers $n_1, \ldots, n_N \in [2^{K_3-1}, 2^{K_3}-1]$ and defines $p_i = p(n_i)$. To find p_i it first redefines n_i such that it is odd by incrementing by one if necessary. Then it executes the Miller-Rabin primality test for $n_i, n_i+2, n_i+4, \ldots$ until it finds a prime. We put an explicit bound on the running time of the generator by bounding the number of integers it considers and the number of iterations of the Miller-Rabin test it performs in total. If the generator stops due to one of these bounds it outputs $\perp$. If $N \geq K_3$, the bound corresponds to $\frac{6{K_3}^4}{{K_1}^3}N$ exponentiations modulo a K_1-bit integer.

The generator can be used in the obvious way to turn the protocol above into a public-coin protocol. The verifier sends $n_1, \ldots, n_N$ to the prover instead of $p_1, \ldots, p_N$ and the prover and verifier generates the primes by computing $(p_1, \ldots, p_N) = \mathsf{PGen}(n_1, \ldots, n_N)$.

A result by Baker and Harman [4] implies that the resulting distribution is close to uniform.

Theorem 2 (cf. [4]). *For large integers n there exists a prime in $[n-n^{0.535}, n]$.*

Corollary 1. *For all primes $p \in [2^{K_3-1}, 2^{K_3}-1]$, $\Pr[p(n) = p] \leq 2^{-0.465(K_3-1)}$, where the probability is taken over a random choice of $n \in [2^{K_3-1}, 2^{K_3}-1]$*

The corollary gives a very pessimistic bound. It is commonly believed that the theorem is true with 0.465 replaced by any constant less than one. Furthermore, Cramér argues probabilistically that there is a prime in every interval $[n - \log^2 n, n]$. See Ribenboim [49] for a discussion on this.

We must argue that the generator fails with negligible probability. There are two ways the generator can fail. Either it outputs $p_1, \ldots, p_N$, where $p_i \neq p(n_i)$ for some i, or it outputs $\perp$.

Lemma 1. *The probability that $\mathsf{PGen}(n_1, \ldots, n_N) \neq (p(n_1), \ldots, p(n_N))$ conditioned on $\mathsf{PGen}(n_1, \ldots, n_N) \neq \perp$ is negligible.*

Unfortunately, the current understanding of the distribution of the primes does not allow a strict analysis of the probability that $\mathsf{PGen}(n_1, \ldots, n_N) = \perp$. Instead we give a heuristic analysis in Cramér's probabilistic model of the primes defined below.

Definition 4 (Cramér's Model). *For each integer n, let X_n be an independent binary random variable such that $\Pr[X_n = 1] = 1/\ln n$. An integer n is said to be prime* if $X_n = 1$.*

The idea is to consider the primality of the integers as a typical outcome of the sequence $(X_n)_{n \in \mathbb{Z}}$. Thus, when we analyze the generator we assume that the primality of an integer n is given by X_n, and our analysis is both over the internal randomness of PGen and the randomness of X_n. We prove the following lemma.

Lemma 2. *In Cramér's model the probability that $\mathsf{PGen}(n_1, \ldots, n_N) = \perp$ is negligible.*

We stress that zero-knowledge and soundness of the modified protocol are not heuristic. The zero-knowledge property holds for arbitrarily distributed *integers* p_i. Soundness follows from Lemma 1. It is only completeness that is argued heuristically. Although this is not always clear, similar heuristic arguments are common in the literature, e.g. to generate strong primes and to encode arbitrary messages in G_q.

Although we now have a public-coin protocol it requires many random bits. This can be avoided by use of a pseudo-random generator PRG as suggested by Groth [28]. Instead of choosing $n_1, \ldots, n_N$ randomly and sending these integers to the prover, the verifier chooses a random seed $s \in [0, 2^{K_1}-1]$ and hands this to the prover. The prover and verifier then computes $(n_1, \ldots, n_N) = \mathsf{PRG}(s)$ and computes the primes from the integers as described above. The output $(n_1, \ldots, n_N)$ may not appear to the prover as random, since he holds the seed s. However, we prove that if we define $P_j = \mathsf{PGen}(\mathsf{PRG}(s))$ and let $P_1, \ldots, P_{j-1} \in \mathbb{Z}_q^N$ be any linearly independent vectors, the probability that $P_j \in \mathrm{Span}(P_1, \ldots, P_{j-1})$ or $p_{j,i} = p_{j,l}$ for some $i \neq l$ is negligible for all $1 \leq j \leq N$. This is all we need in our application.

Universal Verifiability and Random Oracles. Several authors propose turning their proofs of a shuffle into a non-interactive zero-knowledge proofs in the random oracle model using the Fiat-Shamir heuristic. This allows any outsider to check, non-interactively, that a mix-server behaves correctly. If the verification involves no trusted parameters the resulting mix-net is called "universally verifiable".

The Fiat-Shamir heuristic can be applied to our protocol as well. However, we do not see how the prover can generate the RSA parameters $(\mathbf{N}, \mathbf{g}, \mathbf{h})$ by itself and not know the factorization of $\mathbf{N}$ or a non-trivial root. Thus, a verifier must trust that the RSA parameters are generated in a secure way, and the resulting mix-net is not really universally verifiable although very little interaction is necessary.

However, we can achieve universal verifiability under the root assumption in class groups with prime discriminant. A class group is defined by its discriminant Δ. It is conjectured that finding non-trivial roots in a class group with discriminant $\Delta = -p$ for a prime p is hard (cf. [29]). The idea would be to generate a prime p of suitable size from random coins handed to the prover by the verifier in the first round. Then the integer part of the protocol would be executed in the class group defined by $\Delta = -p$. With this modification the protocol gives a universally verifiable mix-net.

It is important to understand that universal verifiability only gives heuristic security because of the dependence of the random oracle model. Furthermore, in practice we expect only a handful of outsiders to implement software to verify the actions of the mix-servers, and given the efficiency of the proof of a shuffle the mix-servers can readily answer such requests interactively. In an interactive proof the outsider can choose the RSA parameters, since the protocol is statistically zero-knowledge. For this to hold the outsider must prove that $\mathbf{g}$ is contained in the group generated by $\mathbf{h}$.

5.5 Complexity

In Section C we describe in detail the assumptions we make in our estimates and how we have summed the cost of each step in the protocol. Here we discuss the results on how they compare with previous protocols.

Comparing the complexity of protocols is tricky, since any comparison must take place for equal security rather than for equal security parameters. The only rigorous method to do this is to perform an exact security analysis of each protocol and choose the security parameters accordingly. Various optimization and pre-computing techniques are also applicable to different degrees in different protocols and in different applications. Despite this we argue informally about the complexity of our protocol.

Furukawa [18] estimates the complexity of previous shuffle-proofs, and claims that his protocol is the most efficient and that it requires the least amount of rounds for a shuffle that involves decryption, namely 5 rounds. It requires $8N$ and $6N$ exponentiations for the prover and verifier respectively. Using standard optimizations this corresponds to less than $2N$ general exponentiations in G_q for each party.

Our protocol requires 5 rounds as well. We estimate the complexity of our proof of a shuffle without any optimizations and then with the same optimizations as Furukawa [18] uses. The asymptotic complexity of our protocol is roughly $5N$ and $2N$ exponentiations in G_q for the prover and verifier respectively. With optimizations this corresponds to about $N/2$ and $N/5$ general exponentiations in G_q.

To give the reader an idea of the practical complexity of our protocol we estimate the complexity for a common set of parameters. We set K_1 relatively large to ensure long term security, e.g. $K_1 = 2048$. A much smaller value of K_2 suffices, since it need only guarantee security during the execution of the protocol, e.g. $K_2 = 1024$. A small challenge suffices since the protocol is run interactively, e.g. $K_4 = 160$. The size of the primes need only guarantee that prime vectors are linearly independent with high probability, e.g. $K_3 = 100$. Finally, a small value of K_5 suffices to ensure the statistical zero-knowledge of the protocol, e.g. $K_5 = 50$. With these parameters the complexity is less than $2.5N$ and $1.6N$ exponentiations in G_q for the prover and verifier. With optimizations as in [18] this corresponds to $0.5N$ and $0.8N$ general exponentiations in G_q. This indicates that our scheme is at least as efficient as previous protocols.

Consider now the overall complexity of the mix-net for these parameters. We ignore the cost for realizing $\mathcal{F}_{\mathrm{CF}}$ and $\mathcal{F}_{\mathrm{RSA}}$, since a joint RSA-modulus can be generated in advance and the only costly invocation of $\mathcal{F}_{\mathrm{CF}}$ can be done in advance as well. The cost for realizing $\mathcal{F}_{\mathrm{ZK}}^{R_{\mathrm{C}}}$ for each sender is less than $1.5N$ (cf. [53]). From the above discussion and using the observation at the end of Section 4.5 we see that the complexity of one step in the mixing chain is bounded by $2N + 0.5N + 0.8N \leq 3.5N$ and the complexity of the mix-net as a whole is bounded by $(2k + 4)N$.

6 Secure Realization of $\mathcal{F}_{\mathbf{ZK}}^{R_{\mathrm{DP}}}$

In this section we transform the proof of a shuffle into a secure realization of $\mathcal{F}_{\mathrm{ZK}}^{R_{\mathrm{DP}}}$ in a $(\mathcal{F}_{\mathrm{RSA}}, \mathcal{F}_{\mathrm{CF}}, \mathcal{F}_{\mathrm{BB}})$-hybrid model, where $\mathcal{F}_{\mathrm{RSA}}$ is an RSA common reference string functionality, and $\mathcal{F}_{\mathrm{CF}}$ is a coin flipping functionality.

Functionality 4 (RSA Common Reference String). The ideal *RSA Common Reference String*, $\mathcal{F}_{\mathrm{RSA}}$, with mix-servers $M_1, \ldots, M_k$, senders $P_1, \ldots, P_N$, and ideal adversary $\mathcal{S}$ proceeds as follows.

1. Generate two random $K_2/2$-bit primes $\mathbf{p}$ and $\mathbf{q}$ such that $(\mathbf{p} - 1)/2$ and $(\mathbf{q} - 1)/2$ are prime and compute $\mathbf{N} = \mathbf{pq}$. Then choose $\mathbf{g}$ and $\mathbf{h}$ randomly in $\mathrm{QR}_{\mathbf{N}}$. Finally, hand $((\mathcal{S}, \mathtt{RSA}, \mathbf{N}, \mathbf{g}, \mathbf{h}), \{(P_j, \mathtt{RSA}, \mathbf{N}, \mathbf{g}, \mathbf{h})\}_{j=1}^{N}, \{(M_j, \mathtt{RSA}, \mathbf{N}, \mathbf{g}, \mathbf{h})\}_{j=1}^{k})$ to $\mathcal{C}_{\mathcal{I}}$.

There are special purpose protocols [6,15] for generating a joint RSA modulus, but these are not analyzed in the UC-framework, so for technical reasons we cannot apply these directly. If these protocols cannot be modified to give a UC-secure protocol, general methods [9] can be used since this need only be done once.

Functionality 5 (Coin-Flipping). The ideal *Coin-Flipping functionality*, $\mathcal{F}_{\mathrm{CF}}$, with mix-servers $M_1, \ldots, M_k$, and adversary $\mathcal{S}$ proceeds as follows.

1. Set $J_{K,} = \emptyset$ for all K.
2. On reception of $(M_j, \texttt{GenerateCoins}, K)$ from $\mathcal{C_I}$, set $J_K \leftarrow J_K \cup \{j\}$. If $|J_K| > k/2$, then set $J_K \leftarrow \emptyset$ choose $c \in \{0,1\}^K$ and hand $((\mathcal{S}, \texttt{Coins}, c), \{(P_j, \texttt{Coins}, c)\}_{j=1}^N, \{(M_j, \texttt{Coins}, c)\}_{j=1}^k)$ to $\mathcal{C_I}$.

It is not hard to securely realize the coin-flipping functionality using a UC-secure verifiable secret sharing scheme (cf. [2]). Each mix-server M_j chooses a random string c_j of K bits and secretly shares it. Then all secrets are reconstructed and c is defined as $\oplus_{j=1}^k c_j$.

Finally, we give the protocol in a hybrid model which securely realizes $\mathcal{F}_{\mathrm{ZK}}^{R_{\mathrm{DP}}}$. This is essentially a translation of Protocol 2 into a multiparty protocol in the UC-setting.

Protocol 3 (Zero-Knowledge Proof of Decryption-Permutation). The protocol $\pi_{\mathrm{DP}} = (M_1, \ldots, M_k)$ consists of mix-servers M_j and proceeds as follows.

Mix-Server M_j. Each mix-server M_j proceeds as follows.

1. Wait for $(\texttt{RSA}, \mathbf{N}, \mathbf{g}, \mathbf{h})$ from $\mathcal{F}_{\mathrm{RSA}}$.
2. Hand $(\texttt{GenerateCoins}, NK_1)$ to $\mathcal{F}_{\mathrm{CF}}$ and wait until it returns $(\texttt{Coins}, (g_1', \ldots, g_N'))$. Then map these strings to elements in G_q by $g_i = (g_i')^{(p-1)/q} \mod p$ (recall that $G_q \subset \mathbb{Z}_p^*$).
3. On input $(\texttt{Prover}, (g, z, y, L, L'), (w, x))$, where $((g, z, y, L, L'), (w, x)) \in L_{R_{\mathrm{DP}}}$ do
   (a) Hand $(\texttt{Prover}, (g, z, 1, 1), w)$ and $(\texttt{Prover}, (g, y, 1, 1), x)$ to $\mathcal{F}_{\mathrm{ZK}}^{R_{\mathrm{C}}}$.
   (b) Denote by W the first message of the prover in Protocol 2. Then hand $(\texttt{Write}, \texttt{W}, W)$ to $\mathcal{F}_{\mathrm{BB}}$.
   (c) Then hand $(\texttt{GenerateCoins}, K_1)$ to $\mathcal{F}_{\mathrm{CF}}$ and wait until it returns $(\texttt{Coins}, s)$. Then set $P = \mathsf{PGen}(\mathsf{PRG}(s))$.
   (d) Denote by C the second message of the prover in Protocol 2. Hand $(\texttt{Write}, \texttt{C}, C)$ to $\mathcal{F}_{\mathrm{BB}}$. Then hand $(\texttt{GenerateCoins}, K_4 - 1)$ to $\mathcal{F}_{\mathrm{CF}}$ and wait until it returns $(\texttt{Coins}, c')$. Define $c = c' + 2^{K_4 - 1}$.
   (e) Denote by R the third message of the prover in Protocol 2. Hand $(\texttt{Write}, \texttt{R}, R)$ to $\mathcal{F}_{\mathrm{BB}}$.
4. On input $(\texttt{Question}, M_l, (g, z, y, L, L'))$, where $L, L' \in G_q^{2N}$ and $(z, y) \in G_q$ do
   (a) Hand $(\texttt{Question}, M_l, (g, z, 1, 1))$ to $\mathcal{F}_{\mathrm{ZK}}^{R_{\mathrm{C}}}$ and wait until it returns $(\texttt{Verifier}, M_l, b_{z,l})$. Then hand $(\texttt{Question}, M_l, (g, y, 1, 1))$ and wait until it returns $(\texttt{Verifier}, M_l, b_{y,l})$. If $b_{z,l} = 0$ or $b_{y,l} = 0$ output $(\texttt{Verifier}, M_l, 0)$.

(b) Then wait until $(M_l, \mathtt{W}, W)$ appears on $\mathcal{F}_{\mathrm{BB}}$. Hand $(\mathtt{GenerateCoins}, K_1)$ to $\mathcal{F}_{\mathrm{CF}}$ and wait until it returns $(\mathtt{Coins}, s)$. Then set $P = \mathsf{PGen}(\mathsf{PRG}(s))$.
(c) Wait until $(M_l, \mathtt{C}, C)$ appears on $\mathcal{F}_{\mathrm{BB}}$. Then hand $(\mathtt{GenerateCoins}, K_4 - 1)$ to $\mathcal{F}_{\mathrm{CF}}$ and wait until it returns $(\mathtt{Coins}, c')$, and until $(M_l, \mathtt{R}, R)$ appears on $\mathcal{F}_{\mathrm{BB}}$. Define $c = c' + 2^{K_4 - 1}$. Then verify (W, P, C, c, R) as in Protocol 2 and set $b_j = 1$ or $b_j = 0$ depending on the result.
(d) Hand $(\mathtt{Write}, \mathtt{Judgement}, M_l, b_j)$ to $\mathcal{F}_{\mathrm{BB}}$ and wait until $(\cdot, \mathtt{Judgement}, M_l, b_{l'})$ appears on $\mathcal{F}_{\mathrm{BB}}$ for $l' \neq j$. Then set $b = 1$ if there are more than $k/2$ distinct $b_{l'} = 1$ and otherwise $b = 0$. Finally, output $(\mathtt{Verifier}, M_l, L, L', b)$.

Theorem 3. *The ideal functionality $\mathcal{F}_{\mathrm{ZK}}^{R_{\mathrm{DP}}}$ is securely realized by π_{DP} in the $(\mathcal{F}_{\mathrm{ZK}}^{R_{\mathrm{C}}}, \mathcal{F}_{\mathrm{CF}}, \mathcal{F}_{\mathrm{RSA}}, \mathcal{F}_{\mathrm{BB}})$-hybrid model with respect to $\mathcal{M}_{k/2}$-adversaries under the DL-assumption and the strong RSA-assumption.*

Corollary 2. *The composition of π_{MN}, π_{C}, π_{DP}, securely realizes $\mathcal{F}_{\mathrm{MN}}$ in the $(\mathcal{F}_{\mathrm{SKS}}, \mathcal{F}_{\mathrm{CF}}, \mathcal{F}_{\mathrm{RSA}}, \mathcal{F}_{\mathrm{BB}})$-hybrid model with respect to $\mathcal{M}_{k/2}$-adversaries under the DDH-assumption and the strong RSA-assumption.*

As indicated in the body of the paper all assumptions except the assumption of a bulletin board can be eliminated. The assumption of a bulletin board can only be eliminated for blocking adversaries (cf. [53]).

7 Conclusion

We have introduced a novel way to construct a mix-net, and given the first provably secure sender verifiable mix-net. We have also introduced a novel approach to construct a proof of a shuffle, and showed how this can be used to securely realize the ideal zero-knowledge proof of knowledge functionality for a decrypt-permutation relation. Combined, this gives the first universally composable mix-net that is efficient for any number of mix-servers.

8 Acknowledgments

I thank Johan Håstad for excellent advise. In particular for discussing efficient generation of the primes. I also thank Mårten Trolin for discussions.

References

1. M. Abe. Universally verifiable mix-net with verification work independent of the number of mix-centers. In *Advances in Cryptology – Eurocrypt '98*, volume 1403 of *Lecture Notes in Computer Science*, pages 437–447. Springer Verlag, 1998.
2. M. Abe and S. Fehr. Adaptively secure feldman vss and applications to universally-composable threshold cryptography. In *Advances in Cryptology – Crypto 2004*, volume 3152 of *Lecture Notes in Computer Science*, pages 317–334. Springer Verlag, 2004. (Full version at Cryptology ePrint Archive, Report 2004/118, http://eprint.iacr.org, May, 2004.).

3. M. Abe and H. Imai. Flaws in some robust optimistic mix-nets. In *Australasian Conference on Information Security and Privacy – ACISP2003*, volume 2727 of *Lecture Notes in Computer Science*, pages 39–50. Springer Verlag, 2003.
4. R. C. Baker and G. Harman. The difference between consecutive primes. *Proceedings of the London Mathematical Society*, 72(3):261–280, 1996.
5. D. Beaver. Foundations of secure interactive computation. In *Advances in Cryptology – Crypto '91*, volume 576 of *Lecture Notes in Computer Science*, pages 377–391. Springer Verlag, 1991.
6. D. Boneh and M. Franklin. Efficient generation of shared RSA keys. In *Advances in Cryptology – Crypto '97*, volume 1233 of *Lecture Notes in Computer Science*, pages 425–439. Springer Verlag, 1997.
7. R. Canetti. Security and composition of multi-party cryptographic protocols. *Journal of Cryptology*, 13(1):143–202, 2000.
8. R. Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *42nd IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 136–145. IEEE Computer Society Press, 2001. (Full version at Cryptology ePrint Archive, Report 2000/067, http://eprint.iacr.org, October, 2001.).
9. R. Canetti, Y. Lindell, R. Ostrovsky, and A. Sahai. Universally composable two-party and multi-party secure computation. In *34th ACM Symposium on the Theory of Computing (STOC)*, pages 449–503. ACM Press, 2002.
10. D. Chaum. Untraceable electronic mail, return addresses and digital pseudo-nyms. *Communications of the ACM*, 24(2):84–88, 1981.
11. I. Damgård and E. Fujisaki. A statistically-hiding integer commitment scheme based on groups with hidden order. In *Advances in Cryptology – Asiacrypt 2002*, volume 2501 of *Lecture Notes in Computer Science*, pages 125–142. Springer Verlag, 2002.
12. Y. Desmedt and K. Kurosawa. How to break a practical mix and design a new one. In *Advances in Cryptology – Eurocrypt 2000*, volume 1807 of *Lecture Notes in Computer Science*, pages 557–572. Springer Verlag, 2000.
13. T. ElGamal. A public key cryptosystem and a signiture scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, 31(4):469–472, 1985.
14. P. Feldman. A practical scheme for non-interactive verifiable secret sharing. In *28th IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 427–438. IEEE Computer Society Press, 1987.
15. P. Fouque and J. Stern. Fully distributed threshold rsa under standard assumptions. Cryptology ePrint Archive, Report 2001/008, 2001. http://eprint.iacr.org/.
16. A. Fujioka, T. Okamoto, and K. Ohta. A practical secret voting scheme for large scale elections. In *Advances in Cryptology – Auscrypt '92*, volume 718 of *Lecture Notes in Computer Science*, pages 244–251. Springer Verlag, 1992.
17. E. Fujisaki and T. Okamoto. Statistical zero knowledge protocols to prove modular polynomial relations. In *Advances in Cryptology – Crypto '97*, volume 1294 of *Lecture Notes in Computer Science*, pages 16–30. Springer Verlag, 1997.
18. J. Furukawa. Efficient, verifiable shuffle decryption and its requirements of unlinkability. In *Public Key Cryptography – PKC 2004*, volume 2947 of *Lecture Notes in Computer Science*, pages 319–332. Springer Verlag, 2004.
19. J. Furukawa, H. Miyauchi, K. Mori, S. Obana, and K. Sako. An implementation of a universally verifiable electronic voting scheme based on shuffling. In *Financial Cryptography 2002*, volume 2357 of *Lecture Notes in Computer Science*, pages 16–30. Springer Verlag, 2002.

20. J. Furukawa and K. Sako. An efficient scheme for proving a shuffle. In *Advances in Cryptology – Crypto 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 368–387. Springer Verlag, 2001.
21. O. Goldreich. *Foundations of Cryptography – Basic Tools*. Cambridge University Press, 2001.
22. O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game. In *19th ACM Symposium on the Theory of Computing (STOC)*, pages 218–229. ACM Press, 1987.
23. S. Goldwasser and L. Levin. Fair computation of general functions in presence of immoral majority. In *Advances in Cryptology – Crypto '90*, volume 537 of *Lecture Notes in Computer Science*, pages 77–93. Springer Verlag, 1990.
24. S. Goldwasser and S. Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28(2):270–299, 1984.
25. P. Golle, S. Zhong, D. Boneh, M. Jakobsson, and A. Juels. Optimistic mixing for exit-polls. In *Advances in Cryptology – Asiacrypt 2002*, volume 2501 of *Lecture Notes in Computer Science*, pages 451–465. Springer Verlag, 2002.
26. T. Granlund. Gnu multiple precision arithmetic library (GMP). Software available at http://swox.com/gmp, Mars 2005.
27. 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.
28. J. Groth. Personal communication, 2004.
29. S. Hamdy J. Buchmann. A survey on iq cryptography. In *Public-Key Cryptography and Computational Number Theory*, pages 1–15. Walter de Gruyter, 2001.
30. M. Jakobsson. Flash mixing. In *19th ACM Symposium on Principles of Distributed Computing – PODC*, pages 83–89. ACM Press, 1998.
31. M. Jakobsson. A practical mix. In *Advances in Cryptology – Eurocrypt '98*, volume 1403 of *Lecture Notes in Computer Science*, pages 448–461. Springer Verlag, 1998.
32. M. Jakobsson and A. Juels. Millimix: Mixing in small batches. Technical Report 1999-33, Center for Discrete Mathematics and Theoretical Computer Science (DIMACS), June 1999.
33. M. Jakobsson and A. Juels. An optimally robust hybrid mix network. In *20th ACM Symposium on Principles of Distributed Computing – PODC*, pages 284–292. ACM Press, 2001.
34. A. Menezes, P. Oorschot, and S. Vanstone. *Handbook of Applied Cryptography.* CRC Press, 1997.
35. S. Micali, C. Rackoff, and B. Sloan. The notion of security for probabilistic cryptosystems. *SIAM Journal on Computing*, 17(2):412–426, 1988.
36. S. Micali and P. Rogaway. Secure computation. In *Advances in Cryptology – Crypto '91*, volume 576 of *Lecture Notes in Computer Science*, pages 392–404. Springer Verlag, 1991.
37. M. Michels and P. Horster. Some remarks on a reciept-free and universally verifiable mix-type voting scheme. In *Advances in Cryptology – Asiacrypt '96*, volume 1163 of *Lecture Notes in Computer Science*, pages 125–132. Springer Verlag, 1996.
38. M. Mitomo and K. Kurosawa. Attack for flash mix. In *Advances in Cryptology – Asiacrypt 2000*, volume 1976 of *Lecture Notes in Computer Science*, pages 192–204. Springer Verlag, 2000.
39. A. Neff. A verifiable secret shuffle and its application to e-voting. In *8th ACM Conference on Computer and Communications Security (CCS)*, pages 116–125. ACM Press, 2001.

40. A. Neff. Verifiable mixing (shuffling) of elgamal pairs. manuscript, 2005. Available at http://www.votehere.com/documents.html.
41. J. Buus Nielsen. Universally composable zero-knowledge proof of membership. http://www.brics.dk/ buus/, April 2005.
42. V. Niemi and A. Renvall. How to prevent buying of votes in computer elections. In *Advances in Cryptology – Asiacrypt '94*, volume 917 of *Lecture Notes in Computer Science*, pages 164–170. Springer Verlag, 1994.
43. W. Ogata, K. Kurosawa, K. Sako, and K. Takatani. Fault tolerant anonymous channel. In *1st International Conference on Information and Communication Security (ICICS)*, volume 1334 of *Lecture Notes in Computer Science*, pages 440–444. Springer Verlag, 1997.
44. P. Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *Advances in Cryptology – Eurocrypt '99*, volume 1592 of *Lecture Notes in Computer Science*, pages 223–238. Springer Verlag, 1999.
45. C. Park, K. Itoh, and K. Kurosawa. Efficient anonymous channel and all/nothing election scheme. In *Advances in Cryptology – Eurocrypt '93*, volume 765 of *Lecture Notes in Computer Science*, pages 248–259. Springer Verlag, 1994.
46. B. Pfitzmann. Breaking an efficient anonymous channel. In *Advances in Cryptology – Eurocrypt '94*, volume 950 of *Lecture Notes in Computer Science*, pages 332–340. Springer Verlag, 1995.
47. B. Pfitzmann and A. Pfitzmann. How to break the direct rsa-implementation of mixes. In *Advances in Cryptology – Eurocrypt '89*, volume 434 of *Lecture Notes in Computer Science*, pages 373–381. Springer Verlag, 1990.
48. B. Pfitzmann and M. Waidner. Composition and integrity preservation of secure reactive systems. In *7th ACM Conference on Computer and Communications Security (CCS)*, pages 245–254. ACM Press, 2000.
49. P. Ribenboim. *The new book of prime number records*. Springer Verlag, 3 edition, 1996.
50. K. Sako and J. Killian. Reciept-free mix-type voting scheme. In *Advances in Cryptology – Eurocrypt '95*, volume 921 of *Lecture Notes in Computer Science*, pages 393–403. Springer Verlag, 1995.
51. Y. Tsiounis and M. Yung. On the security of elgamal based encryption. In *Public Key Cryptography – PKC '98*, volume 1431 of *Lecture Notes in Computer Science*, pages 117–134. Springer Verlag, 1998.
52. D. Wikström. Five practical attacks for "optimistic mixing for exit-polls". In *Selected Areas in Cryptography – SAC 2003*, volume 3006 of *Lecture Notes in Computer Science*, pages 160–174. Springer Verlag, 2003.
53. D. Wikström. A universally composable mix-net. In *1st Theory of Cryptography Conference (TCC)*, volume 2951 of *Lecture Notes in Computer Science*, pages 315–335. Springer Verlag, 2004.

A Security Analysis of Protocol 1

Proof (Theorem 1 on Page 10). We describe an ideal adversary $\mathcal{S}(\cdot)$ that runs any hybrid adversary $\mathcal{A}$ as a black-box. Then we show that if $\mathcal{S}(\mathcal{A})$ does not simulate $\mathcal{A}$ sufficiently well, we can break the DDH-assumption.

THE IDEAL ADVERSARY $\mathcal{S}$. Let I_P and I_M be the set of indices of participants corrupted by $\mathcal{A}$ of the sender type and the mix-server type respectively. The ideal adversary $\mathcal{S}$ corrupts the dummy participants $\tilde{P}_i$ for which $i \in I_P$, and the dummy participants $\tilde{M}_i$ for which $i \in I_M$. The ideal adversary is best described by starting with a copy of the original hybrid ITM-graph

$$(V, E) = \mathcal{Z}'(\mathcal{H}(\mathcal{A}, \pi^{(\tilde{\pi}_1^{\mathcal{F}_{\mathrm{BB}}}, \tilde{\pi}_2^{\mathcal{F}_{\mathrm{SKS}}}, \tilde{\pi}_3^{\mathcal{F}_{\mathrm{ZK}}^{R_{\mathrm{C}}}}, \tilde{\pi}_4^{\mathcal{F}_{\mathrm{ZK}}^{R_{\mathrm{DP}}}})})) \enspace ,$$

where $\mathcal{Z}$ is replaced by a machine $\mathcal{Z}'$.

The adversary $\mathcal{S}$ simulates all machines in V except those in $\mathcal{A}$, and the corrupted machines P_i for $i \in I_P$ and M_i for $i \in I_M$ under $\mathcal{A}$'s control. We now describe how each machine is simulated.

$\mathcal{S}$ simulates the machines P_i, $i \notin I_P$ and the ideal functionalities $\mathcal{F}_{\mathrm{BB}}$, $\mathcal{F}_{\mathrm{ZK}}^{R_{\mathrm{C}}}$ and $\mathcal{F}_{\mathrm{SKS}}$ honestly. All M_j for $j \notin I_M$ are also simulated honestly, except for M_l, where l is chosen as the maximal index not in I_M, i.e. the last honest mix-server. The machine M_l plays a special role.

Simulation of Links $(\mathcal{Z}, \mathcal{A})$, $(\mathcal{Z}, P_i)$ for $i \in I_P$, and $(\mathcal{Z}, M_j)$ for $j \in I_M$. $\mathcal{S}$ simulates $\mathcal{Z}'$, $\tilde{P}_i$, for $i \in I_P$, and $\tilde{M}_j$ for $j \in I_M$, such that it appears as if $\mathcal{Z}$ and $\mathcal{A}$, $\mathcal{Z}$ and P_i for $i \in I_P$, and $\mathcal{Z}$ and M_j for $j \in I_M$ are linked directly.

1. When $\mathcal{Z}'$ receives m from $\mathcal{A}$, m is written to $\mathcal{Z}$, by $\mathcal{S}$. When $\mathcal{S}$ receives m from $\mathcal{Z}$, m is written to $\mathcal{A}$ by $\mathcal{Z}'$. This is equivalent to that $\mathcal{Z}$ and $\mathcal{A}$ are linked directly.
2. When $\mathcal{Z}'$ receives m from P_i for $i \in I_P$, m is written to $\mathcal{Z}$ by $\tilde{P}_i$. When $\tilde{P}_i$, $i \in I_P$, receives m from $\mathcal{Z}$, m is written to P_i by $\mathcal{Z}'$. This is equivalent to that $\mathcal{Z}$ and P_i are linked directly for $i \in I_P$.
3. When $\mathcal{Z}'$ receives m from M_j for $j \in I_M$, m is written to $\mathcal{Z}$ by $\tilde{M}_j$. When $\tilde{M}_j$, $j \in I_M$, receives m from $\mathcal{Z}$, m is written to M_j by $\mathcal{Z}'$. This is equivalent to that $\mathcal{Z}$ and M_j are linked directly for $j \in I_M$.

Extraction from Corrupt Mix-Servers and Simulation of Honest Mix-Servers. When a corrupt mix-server M_j, for $j \in I_M$, writes `Run` on $\mathcal{F}_{\mathrm{BB}}$, $\mathcal{S}$ must make sure that $\tilde{M}_j$ sends $(\texttt{Run})$ to $\mathcal{F}_{\mathrm{MN}}$. Otherwise it may not be possible to deliver an output to honest mix-servers at a later stage. If an honest dummy mix-server $\tilde{M}_j$, for $j \notin I_M$, receives $(\texttt{Run})$ from $\mathcal{Z}$, $\mathcal{S}$ must make sure that M_j receives $(\texttt{Run})$ from $\mathcal{Z}'$. If an honest mix-server M_j, for $j \notin I_M$, outputs $(\texttt{Output}, L')$, $\mathcal{S}$ must make sure that $\tilde{M}_j$ does the same. This is done as follows.

1. Let $j \in I_M$. If $(\cdot, M_j, \texttt{Run})$ appears on $\mathcal{F}_{\mathrm{BB}}$ $\tilde{M}_j$ hands $(\texttt{Run})$ to $\mathcal{F}_{\mathrm{MN}}$. When $\mathcal{S}$ receives $(\mathcal{S}, \tilde{M}_j, \texttt{Run})$ or $((\mathcal{S}, \tilde{M}_j, \texttt{Output}, L'), \{(\tilde{M}_l, \tau_l)\}_{l=1}^k)$ from $\mathcal{C}_{\mathcal{I}}$ the simulation of $\mathcal{F}_{\mathrm{BB}}$ is continued.
2. Let $j \notin I_M$. If $\mathcal{S}$ receives $(\mathcal{S}, \tilde{M}_j, \texttt{Run})$ or $((\mathcal{S}, M_j, \texttt{Output}, L'), \{(M_l, \tau_l)\}_{l=1}^k)$ from $\mathcal{F}_{\mathrm{MN}}$, $\mathcal{Z}'$ hands $(\texttt{Run})$ to M_j.
3. Let $j \notin I_M$. If $\mathcal{Z}'$ receives $(\texttt{Output}, L')$ from M_j, $\mathcal{S}$ sends $(1, \tau_j)$ to $\mathcal{C}_{\mathcal{I}}$, i.e. $\mathcal{S}$ instructs $\mathcal{C}_{\mathcal{I}}$ to deliver $(\texttt{Output}, L')$ to $\tilde{M}_j$.

Extraction from Corrupt Senders and Simulation of Honest Senders. If a corrupt sender P_i, for $i \in I_P$, in the hybrid protocol produces a cryptotext and informs

$\mathcal{F}_{\text{ZK}}^{R_{\text{C}}}$ such that its input is deemed valid, then $\mathcal{S}$ must instruct $\tilde{P}_i$ to hand this as input to $\mathcal{F}_{\text{MN}}$.

When an honest dummy sender $\tilde{P}_i$, for $i \notin I_P$, receives a message m_i from $\mathcal{Z}$, $\mathcal{S}$ must ensure that P_i receives some message m_i' from $\mathcal{Z}'$. But $\mathcal{S}$ cannot see m_i, and must therefore hand some other message $m_i' \neq m_i$ to P_i, and then later fix this flaw in the simulation before $\mathcal{A}$ or $\mathcal{Z}$ notice it. This is done as follows.

1. Let $i \in I_P$. Until $\mathcal{S}$ receives $((\mathcal{S}, M_j, \texttt{Output}, L'), \{(M_l, \tau_l)\}_{l=1}^{k})$ from $\mathcal{C}_{\mathcal{I}}$.
   (a) If $\mathcal{F}_{\text{ZK}}^{R_{\text{C}}}$ receives a message $(P_i, \texttt{Prover}, (Z_1, Y_1, u_i, v_i), r_i)$ such that $((Z_1, Y_1, u_i, v_i), r_i) \in R_{\text{DP}}$, then consult the storage of $\mathcal{F}_{\text{BB}}$ and look for a pair $(c, P_i, (u_i, v_i))$.
   (b) If $\mathcal{F}_{\text{BB}}$ receives $(P_i, \texttt{Write}, (u_i, v_i))$ then look if $\mathcal{F}_{\text{ZK}}^{R_{\text{C}}}$ stored r_i under $(P_i, (Z_1, Y_1, u_i, v_i))$ such that $((Z_1, Y_1, u_i, v_i), r_i) \in R_{\text{DP}}$.
   If such a pair $[(c, P_i, (u_i, v_i)), (P_i, (Z_1, Y_1, u_i, v_i), r_i)]$ is found then $\tilde{P}_i$ sends $m_i = v_i Y_1^{-r_i}$ to $\mathcal{F}_{\text{MN}}$ and ignores further such pairs. When $\mathcal{F}_{\text{MN}}$ writes $(\tilde{P}_i, \texttt{Send})$ to $\mathcal{S}$, the simulation, of $\mathcal{F}_{\text{ZK}}^{R_{\text{C}}}$ or $\mathcal{F}_{\text{BB}}$ respectively, is continued.
2. Let $i \notin I_P$. When $\mathcal{S}$ receives $(\tilde{P}_i, \texttt{Send})$ from $\mathcal{F}_{\text{MN}}$, then $\mathcal{Z}'$ sends a randomly chosen message $m_i' \in G_q$ to P_i. By definition P_i chooses a random $r_i' \in \mathbb{Z}_q$ and forms its cryptotext as $(u_i, v_i) = (Z_1^{r_i'}, Y_1^{r_i'} m_i')$. Note that this corresponds to a pair of random elements in G_q.

How M_l and $\mathcal{F}_{\text{ZK}}^{R_{\text{DP}}}$ fix the flaw in the simulation. $\mathcal{S}$ must make sure that the faulty messages $m_i' \neq m_i$ introduced during simulation of honest senders, because it does not know the real messages m_i of the honest dummy participants $\tilde{P}_i$ for $i \in i_P$, are not noticed. This is done by modifying M_l and $\mathcal{F}_{\text{ZK}}^{R_{\text{DP}}}$ as follows.

1. If $\mathcal{F}_{\text{ZK}}^{R_{\text{DP}}}$ receives a tuple $(M_j, \texttt{Question}, M_l, (g, z_l, y_l, L_{l-1}, L_l))$ it verifies that a tuple on the form $(M_l, \texttt{Prover}, (g, z_l, y_l, L_{l-1}, L_l), \cdot)$ has been received. If so it sets $b = 1$ and otherwise $b = 0$. Finally it hands $((\mathcal{S}, M_j, \texttt{Verifier}, M_l, (z_l, y_l, L_{l-1}, L_l), b), (M_j, \texttt{Verifier}, M_l, b))$ to $\mathcal{C}_{\mathcal{I}}$.
2. Note that by construction $\mathcal{S}$ has received $((\mathcal{S}, M_j, \texttt{Output}, L'), \ldots)$, i.e. it knows the output L'. Let $\{m_i\}_{i=1}^{N'}$ be the messages in L', except that they are ordered such that m_i is the message sent by P_i for all $i \in I_P$. The other messages are ordered arbitrarily. Note that $\mathcal{S}$ knows r_i and m_i for all $i \in I_P$, since it simulated the handing of these to $\mathcal{F}_{\text{MN}}$ itself.
   M_l does the following instead of Step 7b in the protocol. It chooses $r_i \in \mathbb{Z}_q$, for $i \notin I_P$, and $\pi_l \in \Sigma_N$ randomly, and computes the list
   $$L_l = \{(u_{l,i}, v_{l,i})\}_{i=1}^{N'} = \mathrm{Sort}\left(\left\{\left(Z_{l+1}^{r_i}, Y_{l+1}^{r_i} m_i\right)\right\}_{i=1}^{N'}\right) .$$
   Finally it hands $(\texttt{Prover}, (g, z_l, y_l, L_{l-1}, L_l), \cdot)$ to $\mathcal{F}_{\text{ZK}}^{R_{\text{DP}}}$, and it hands $(\texttt{Write}, (\texttt{List}, L_l))$ to $\mathcal{F}_{\text{BB}}$.

The first step ensures that $\mathcal{F}_{\text{ZK}}^{R_{\text{DP}}}$ plays along with M_l and pretends to other M_j that M_l did prove his knowledge properly. The second step ensures that M_l fixes the flaw in the simulation introduced by $\mathcal{S}$ at the point when it did not know the messages sent by honest dummy participants $\tilde{P}_i$, for $i \notin I_P$.

Note that the cryptotexts of all corrupted parties are identically distributed as in the real protocol. The same randomness r_i is used to encrypt the message m_i sent by P_i for $i \in I_P$.

This concludes the definition of the ideal adversary $\mathcal{S}$.

Reaching a Contradiction. Next we show, using a hybrid argument, that if the ideal adversary $\mathcal{S}$ defined above does not imply the security of Protocol 1, then we can break the DDH-assumption.

Suppose that $\mathcal{S}$ does not imply the security of the protocol. Then there exists a hybrid adversary $\mathcal{A}$, an environment $\mathcal{Z}$ with auxiliary input $z = \{z_n\}$, a constant $c > 0$ and an infinite index set $\mathcal{N} \subset \mathbb{N}$ such that for $n \in \mathcal{N}$

$$|\Pr[\mathcal{Z}_z(\mathcal{I}(\mathcal{S}, \tilde{\pi}^{\mathcal{F}_{\mathrm{MN}}})) = 1] - \Pr[\mathcal{Z}_z(\mathcal{H}(\mathcal{A}', \pi^{(\tilde{\pi}_1^{\mathcal{F}_{\mathrm{BB}}}, \tilde{\pi}_2^{\mathcal{F}_{\mathrm{SKS}}}, \tilde{\pi}_3^{\mathcal{F}_{\mathrm{ZK}}^{R_{\mathrm{C}}}}, \tilde{\pi}_4^{\mathcal{F}_{\mathrm{ZK}}^{R_{\mathrm{DP}}}})})) = 1]| \geq \frac{1}{n^c}$$

where $\mathcal{S}$ runs $\mathcal{A}$ as a black-box as described above, i.e. $\mathcal{S} = \mathcal{S}(\mathcal{A})$.

Defining the Hybrids. Without loss we assume that $\{1, \ldots, N\} \backslash I_P = \{1, \ldots, \eta\}$, and define an array of hybrid machines $T_0, \ldots, T_\eta$. Set $T_0 = \mathcal{Z}_z(\mathcal{I}(\mathcal{S}(\mathcal{A}), \tilde{\pi}^{\mathcal{F}_{\mathrm{MN}}}))$, and then define T_s by the following modification to T_0.

1. When $\mathcal{S}$ receives $(\tilde{P}_i, \texttt{Send})$ from $\mathcal{F}_{\mathrm{MN}}$, for $i \notin I_P$, it checks if $i \in \{1, \ldots, s\}$.
   (a) If so it consults the storage of $\mathcal{F}_{\mathrm{MN}}$ to find the message m_i sent by $\tilde{P}_i$. Then $\mathcal{Z}'$ sends m_i to P_i and treats i as if $i \in I_P$ in the simulation of M_l. This means that $r_i = r'_i$, and $m'_i = m_i$.
   (b) Otherwise $\mathcal{Z}'$ sends a random message $m'_i \in \mathbb{Z}_q$ to P_i, as in the original simulation.

By inspection of the constructions we see that the output of T_η is identically distributed to the output of $\mathcal{Z}_z(\mathcal{H}(\mathcal{A}', \pi^{(\tilde{\pi}_1^{\mathcal{F}_{\mathrm{BB}}}, \tilde{\pi}_2^{\mathcal{F}_{\mathrm{SKS}}}, \tilde{\pi}_3^{\mathcal{F}_{\mathrm{ZK}}^{R_{\mathrm{C}}}}, \tilde{\pi}_4^{\mathcal{F}_{\mathrm{ZK}}^{R_{\mathrm{DP}}}})}))$ since the only essential difference is that M_l does not hand knowledge of his transformation to $\mathcal{F}_{\mathrm{ZK}}^{R_{\mathrm{DP}}}$, but $\mathcal{F}_{\mathrm{ZK}}^{R_{\mathrm{DP}}}$ ignores M_l's inability so this is not discovered by $\mathcal{A}$ or $\mathcal{Z}$.

If we set $p_s = \Pr[T_s = 1]$, we have $\frac{1}{n^c} \leq |p_0 - p_\eta| \leq \sum_{i=1}^{\eta} |p_{s-1} - p_s|$, which implies that there exists some fixed $0 < s \leq \eta$ such that $|p_{s-1} - p_s| \geq \frac{1}{\eta n^c} \geq \frac{1}{N n^c}$.

Defining a Distinguisher. We are now finally ready to define a distinguisher D for the experiment considered in Lemma 11, i.e., a variation of the DDH-experiment.

D is confronted with the following test. An oracle first chooses $r, w, r', x, r'' \in \mathbb{Z}_q$ and a bit $b \in \{0, 1\}$ randomly

1. If $b = 0$, then it defines $(a, y, v, z, u) = (g^r, g^w, g^{r'}, g^x, g^{r''})$.
2. If $b = 1$, then it defines $(a, y, v, z, u) = (g^r, g^w, g^{rw}, g^x, g^{rx})$.

The distinguisher D must guess the value of b, i.e., which type of input it gets.

D replaces (z_l, y_l) by (z, y) in M_l's key generation. This does not change the distribution of this key and thus does not change any of the hybrids. D computes Z_j and Y_j as follows. It computes $Z_l = z_l^{\prod_{j=l+1}^{k} w_j}$, and for $j \neq l$ it computes $Z_j = Z_{j+1}^{w_j}$ as usual. It computes $Y_l = Y_{l+1} y_l^{\prod_{j=l+1}^{k} w_j}$, and for $j \neq l$ it computes

$Y_j = Y_{j+1} Z_{j+1}^{x_j}$ as usual. Note that the scheme is carefully chosen to allow the simulator to generate Z_j and Y_j without knowledge of w or x.

Since M_l appears to behave honestly (with the help of $\mathcal{F}_{\mathrm{ZK}}^{R_{\mathrm{DP}}}$), the fact that M_l does not know $w = \log_g z_l$ or $x = \log_g y_l$ is never revealed, and since less than $k/2$ mix-servers are corrupted w or x need never be recovered. The reader should think of r as the randomness of a sender, w as w_l and x as x_l. D simulates T_s until P_s receives the message $(\texttt{Send}, m_s)$, at which point it forms a cryptotext

$$(u_s, v_s) = \left(u^{\prod_{j \neq l} w_j}, a^{\sum_{j \neq l} x_j \prod_{j'=j+1}^{k} w_{j'}} v^{\prod_{j=l+1}^{k} w_j} m_s \right) \quad .$$

Note that if $b = 1$, we have $(u_s, v_s) = (Z_1^r, Y_1^r m_i)$. The sender P_s is modified to hand $(\texttt{Write}, (u_s, v_s))$ to $\mathcal{F}_{\mathrm{BB}}$, and the tuple $(\texttt{Prover}, (Z_1, Y_1, u_s, v_s), 1)$ to $\mathcal{F}_{\mathrm{ZK}}^{R_{\mathrm{C}}}$. Furthermore, $\mathcal{F}_{\mathrm{ZK}}^{R_{\mathrm{C}}}$ is modified to a handle this message as if we had $((Z_1, Y_1, u_s, v_s), 1) \in R_{\mathrm{C}}$, i.e. it lies on P_i's behalf. Finally, we must change the way M_l forms its output. Suppose that $(u_{j,l-1}, v_{j,l-1})$ corresponds to the input (u_s, v_s). Instead of decrypting this pair M_l replaces it by

$$(u_s', v_s') = \left(a^{\prod_{j=l+1}^{k} w_j}, a^{\sum_{j=l+1}^{k} x_j \prod_{j'=j+1}^{k} w_{j'}} m_i \right) = (Z_{l+1}^r, Y_{l+1}^r m_i) \quad .$$

D then continues the simulation of T_s until it outputs a bit b', which is then output by D.

If $b = 0$, then u and v are random elements in G_q. This implies that (u_s, v_s) is identically distributed to the corresponding cryptotext in the simulation and the output of D is identically distributed to the output of T_{s-1}.

If $b = 1$, then (u_s, v_s) is a valid encryption of m_i with randomness r, and (u_s', v_s') is a partial decryption of (u_s, v_s) using the secret keys $w_l = w$ and $x_l = x$. This implies that the output of D is identically distributed to the output of T_s.

We conclude that

$$\begin{aligned} &|\Pr[D(g^r, g^w, g^{rw}, g^x, g^{rx}) = 1] - \Pr[D(g^r, g^w, g^{r'}, g^x, g^{r''}) = 1]| \\ &= |p_{s-1} - p_s| \geq \frac{1}{Nn^c} \quad . \end{aligned}$$

Lemma 11 in Section H shows that this contradicts the DDH-assumption.

B Analysis of the Prime Vector Generator

The generator PGen is not allowed to test more than $N' = \ln n(N + \sqrt{NK_3})$ integers, and it is not allowed to do more than $N'' = NK_3 + 2(N' + \sqrt{N'K_3})$ iterations of the Miller-Rabin test in total.

Since each iteration of the Miller-Rabin test corresponds to computing a single exponentiation modulo an K_3-bit integer, the running time of the generator may be expressed as $(NK_3 + 2(N' + \sqrt{N'K_3})) \frac{{K_3}^3}{{K_1}^3}$. If we assume that $N \geq K_3$ this can be bounded by $\frac{6{K_3}^4}{{K_1}^3} N$ exponentiations modulo a K_1-bit integer.

Remark 1. Using GMP [26] we have compared for $N = 1000$ the complexity of running the generator with $K_3 = 100$ and the complexity of exponentiation modulo a K_1-bit integer for $K_1 = 1000$. Our estimate suggests a ratio 6/10, whereas the experiment gives 0.46, so the estimate seems reasonable.

Proof (Lemma 1 on Page 17). Suppose that $\mathsf{PGen}(n_1, \ldots, n_N) \neq \perp$. Then PGen outputs a list of integers $(p_1, \ldots, p_N)$ and for each integer n between n_i and p_i an iteration of the Miller-Rabin primality test has found a witness that n is composite. Thus, there exists no primes between n_i and p_i. The probability that p_i is considered to prime despite that it is not, is bounded by 2^{-K_3} since K_3 iterations of the Miller-Rabin test are executed. The union bound implies that the probability that $p_i \neq p(n_i)$ for any i is bounded by $N2^{-K_3}$, which is negligible.

Proof (Lemma 2 on Page 17). We must bound the probability that the generator need too many invocations of Miller-Rabin or too many iterations in total. For simplicity we assume that no integer is tested twice for primality.

Denote by Y_i the event that the integer tested in the ith invocation of Miller-Rabin is a prime. Since the generator never checks the same integer twice, we assume that the Y_i are independent. Set $Y = \sum_{i=1}^{N'} Y_i$. Then $\mathrm{Exp}[Y] = \frac{N'}{\ln n} = N + \sqrt{NK_3}$. We bound the probability that the generator fails by having to check to many integers by

$$\Pr\left[Y \leq N\right] = \Pr\left[Y \leq \mathrm{Exp}[Y] - \sqrt{NK_3}\right] \leq e^{-2K_3} \ .$$

Suppose now that at least N of the Y_i are ones. We must bound the probability that the generator need more than N'' iterations in the Miller-Rabin test. At most N primes are tested and for each such prime K_3 iterations are needed giving NK_3 iterations for the primes. Along the way a number of composite numbers are tested. Our bound stipulates that at most $2(N' + \sqrt{N'K_3})$ iterations can be spent on composites, and at most N' integers can be tested at all. Denote by Z_i the indicator variable for the event that the ith iteration of the Miller-Rabin test, when run on a composite outputs "composite". Define $Z = \sum_{i=1}^{2(N'+\sqrt{N'K_3})} Z_i$. In each iteration of the Miller-Rabin test on a composite the test outputs "composite" with probability at least 1/2 so $\Pr[Z_i] \geq \frac{1}{2}$ and $\mathrm{Exp}[Z] \geq N' + \sqrt{N'K_3}$.

The probability of failure is bounded by the probability that the generator is not allowed to perform more Miller-Rabin tests despite that there are more integers to be tested. This probability is captured below.

$$\Pr\left[Z \leq N'\right] = \Pr\left[Z \leq \mathrm{Exp}[Z] - \sqrt{N'K_3}\right] \leq e^{-2K_3} \ .$$

Thus, it follows that the probability that the generator outputs $\perp$ negligible.

C Complexity Analysis of Protocol 2

In this section we substantiate our complexity claims. Denote by t_{EXP} the time it takes to compute an exponentiation in G_q, i.e. with an K_1-bit exponent modulo a K_1-bit integer. If the exponent instead has K bits, we assume that computing the exponentiation takes time $\frac{K}{K_1} t_{\mathrm{EXP}}$. This is reasonable since exponentiation is normally implemented using the square-and-multiply algorithm. We must also relate the time it takes to compute an exponentiation with a K-bit exponent and modulus to t_{EXP}. We assume that this takes time $(\frac{K}{K_1})^3 t_{\mathrm{EXP}}$. This is reasonable since exponentiation normally takes cubic time in the bit-size of the inputs. We use the same conventions for exponentiation in $\mathrm{QR}_{\mathbf{N}}$ with K_1 replaced by K_2.

We use the same convention as Furukawa [18] when we estimate the effect of standard optimization techniques [34], i.e., simultaneous exponentiation reduces the cost by a factor of $1/3$, and fixed base exponentiation reduces the cost by a factor of $1/12$.

First we consider the theoretical requirements. For simplicity we set $K_1 = K_2$. Recall that, using the number field sieve, discrete logarithms modulo an K_1-bit prime can be solved in time $\exp(O({K_1}^{1/3} \log K_1)) \leq \exp(O({K_1}^{2/5}))$. Thus, from a theoretical point of view we may assume that $K_3 = {K_1}^{2/5}$. We count all terms that do not go to zero with increasing K_1. It can be seen that the cost for the prover and verifier is roughly $5N$ and $2N$ exponentiations respectively. With optimizations the corresponding estimates would be $\frac{5}{12}N$ and $\frac{1}{6}N$.

Our estimates for the practical complexity of the protocol follows from the Scheme program below. We have simply counted the number of exponentiations using the assumptions above. Each function `veriStep`i computes the complexity of verifier in the ith step of the protocol and correspondingly for `proStep`i. In Table 1 we give the complexity for some common parameters.

Parameters					Non-Optimized		Optimized	
K_1	K_2	K_4	K_3	K_5	Prover	Verifier	Prover	Verifier
1024	1024	160	100	50	$7.8N$	$8.2N$	$1.4N$	$4.8N$
2048	1024	160	100	50	$2.4N$	$1.5N$	$0.4N$	$0.8N$
2048	2048	160	100	50	$6.4N$	$3.7N$	$0.9N$	$1.1N$
3072	2048	160	100	50	$2.9N$	$1.4N$	$0.4N$	$0.4N$
3072	3072	160	100	50	$5.9N$	$3.0N$	$0.7N$	$0.6N$

Table 1. The table gives estimates of the complexity, without and with optimization, of the prover and verifier in terms of general exponentiations in G_q for some common security parameters.

D Security Analysis of Protocol 2

In this section we analyze the security of Protocol 2. The zero-knowledge property is relatively straightforward, so most of our effort is spent on analyzing the soundness of the protocol.

```
; Load file with (load ``complexity.scm'')
; Compute complexity with (comp 2048 1024 160 100 50 (/ 1 12) (/ 1 3))

(define (veriStep2 k1 k2 k3 k4 k5 fixbasefak simulfak)
  (/ (* 6 k3 k3 k3 k3) (* k1 k1 k1)))

(define (veriStep3 k1 k2 k3 k4 k5 fixbasefak simulfak)
  (* simulfak 3 (/ k3 k1)))

(define (veriStep7 k1 k2 k3 k4 k5 fixbasefak simulfak)
  (+ (* simulfak 3 (/ (+ k3 k4 k5) k1))
     (* (/ (* k2 k2 k2) (* k1 k1 k1))
        (+ (+ (* fixbasefak (/ (+ k2 k4 (* 2 k5)) k2))
              (/ (+ k3 k4 k5) k2))
           (* fixbasefak
              (+ (/ (+ k2 k4 (* 2 k5)) k2) (/ (+ k3 k4 k5) k2)))
           (/ k3 k2)))))

(define (veri k1 k2 k3 k4 k5 fixbasefak simulfak)
  (float (+ (veriStep2 k1 k2 k3 k4 k5 fixbasefak simulfak)
            (veriStep3 k1 k2 k3 k4 k5 fixbasefak simulfak)
            (veriStep7 k1 k2 k3 k4 k5 fixbasefak simulfak))))


(define (proStep1 k1 k2 k3 k4 k5 fixbasefak simulfak)
  fixbasefak)

(define (proStep3 k1 k2 k3 k4 k5 fixbasefak simulfak)
  (veriStep3 k1 k2 k3 k4 k5 fixbasefak simulfak))

(define (proStep4 k1 k2 k3 k4 k5 fixbasefak simulfak)
  (+ (* simulfak 3 (/ (+ k3 k4 k5) k1))
     (* (/ (* k2 k2 k2) (* k1 k1 k1))
        (+ (* fixbasefak (/ (+ k2 k5) k2))
           (/ k3 k2)
           (* fixbasefak (+ (/ (+ k2 k5) k2) (/ k3 k2)))
           (* fixbasefak (/ (+ k2 k4 (* 2 k5)) k2))
           (/ (+ k3 k4 k5) k2)
           (* fixbasefak
              (+ (/ (+ k2 k4 (* 2 k5)) k2)
                 (/ (+ k3 k4 k5) k2)))))))

(define (pro k1 k2 k3 k4 k5 fixbasefak simulfak)
  (float (+ (proStep1 k1 k2 k3 k4 k5 fixbasefak simulfak)
            (proStep3 k1 k2 k3 k4 k5 fixbasefak simulfak)
            (proStep4 k1 k2 k3 k4 k5 fixbasefak simulfak))))
```

Table 2. Scheme program for estimation of the complexity of the protocol.

D.1 The Protocol is Honest Verifier Statistical Zero-Knowledge

Proof (Proposition 1 on Page 15). The simulator chooses $p_1, \ldots, p_N$ and c honestly. Then it chooses $f_1, \ldots, f_7, f_{1/w}, f_{x/w}, f_w, f_x, f_{r'} \in \mathbb{Z}_q$, $e \in [0, 2^{K_2+NK_3+K_4+K_5+\log_2 N} - 1]$, $e' \in [0, 2^{K_2+K_5+\log_2 N} - 1]$, $(e_i, e'_i)_{i=1}^N \in [0, 2^{K_2+K_4+2K_5} - 1]^N$, and $(d_i)_{i=1}^N \in [0, 2^{K_3+K_4+K_5} - 1]^N$ randomly. It also chooses $b_1, \ldots, b_5 \in G_q$ and $\mathbf{b}_i, \mathbf{b}'_i \in \mathrm{QR}_{\mathbf{N}}$ randomly. Finally, the simulator defines, $(\beta_1, \ldots, \beta_9)$ by Equations (15)- (18), $(\alpha_1, \alpha_2, \alpha_3)$ by Equations (19), $\boldsymbol{\gamma}_i$ and $\boldsymbol{\gamma}'_i$ by Equation (20), $\boldsymbol{\gamma}$ by Equation (21), and $\boldsymbol{\gamma}'$ by Equation (22) respectively. The distribution of the resulting elements is statistically close to the distribution of the corresponding elements in the protocol.

D.2 The Protocol is Sound

As explained at the end of Section 5.2, we cannot hope to prove formally that the protocol is an interactive proof, or a proof of knowledge. Instead we prove that for every adversary A, the probability, over the random choice of the RSA-parameters $(\mathbf{N}, \mathbf{g}, \mathbf{h})$ and the independent generators $g_1, \ldots, g_N \in G_q$, that the adversary first outputs a common input $(z, y, L, L') \notin R_{\mathrm{DP}}$ and then convinces the verifier that it knows a witness (w, x) such that $((g, z, y, L, L'), (w, x)) \in R_{\mathrm{DP}}$ is negligible.

Although our proof of a shuffle is complex the analysis is similar in structure to most analyses of proofs of knowledge. First we identify conditions on a set of related transcripts that allow us to extract the knowledge held by the prover. This corresponds to finding a "fork" in standard Schnorr-like proofs of knowledge of a logarithm. Then we describe how such transcripts can be generated by interacting with a prover.

Notation. We need to consider many transcripts and be able to distinguish between these. To simplify we denote the jth transcript in a list of transcripts by $T_j = (I_j, W_j, P_j, C_j, c_j, R_j)$ where I_j denotes the common input, W_j denotes the list of commitments w_i, and

$$
\begin{aligned}
P_j &= (p_{j,1}, \ldots, p_{j,N}) \\
C_j &= ((b_{j,i})_{i=1}^5, (\beta_{j,i})_{i=1}^9, (\alpha_{j,1}, \alpha_{j,2}, \alpha_{j,3}), (\mathbf{b}_{j,i}, \mathbf{b}'_{j,i})_{i=1}^N, (\boldsymbol{\gamma}_{j,i}, \boldsymbol{\gamma}'_{j,i})_{i=1}^N, (\boldsymbol{\gamma}_j, \boldsymbol{\gamma}'_j)) \\
R_j &= ((f_{j,i})_{i=1}^7, f_{j,1/w}, f_{j,x/w}, f_{j,w}, f_{j,x}, f_{j,r'}), (e_{j,i}, e'_{j,i})_{i=1}^N, (d_{j,i})_{i=1}^N, (e_j, e'_j)) \ .
\end{aligned}
$$

Since I_j and W_j are fixed for all j in most of our analysis we do not introduce any notation indexed on j for their parts. We think of (P_j, C_j, c_j, R_j) as "the primes", "the commitments", "the challenge" and "the reply" in the jth transcript.

Extraction From Suitable Transcripts. Recall that for standard Schnorr-like proofs of knowledge it suffices to find a "fork", i.e., two accepting transcripts with identical commitments from the prover, but distinct challenges from the verifier, to extract the knowledge held by the prover.

Our protocol has a similar, but more complicated property. The lemma below shows that given $2N$ transcripts of a special form, we can either extract a witness of the decryption-permutation relation, or one of a small number of special cases occur.

In a later section we show that if one of the special cases occur with non-negligible probability we can break a standard complexity assumption, so the reader should think of the lemma as saying that we can extract a witness for the decryption-permutation relation.

Lemma 3. *Let $T_1, \ldots, T_{2N}$ be accepting transcripts such that*

1. $(I_1, W_1) = (I_2, W_2) = \ldots = (I_N, W_N)$,
2. $(P_j, C_j) = (P_{j+N}, C_{j+N})$ *and* $c_j \neq c_{j+N}$, *and*
3. $\mathrm{Span}(P_1, \ldots, P_N) = \mathbb{Z}_q^N$ *and* $p_{j,i} \neq p_{j,l}$ *for* $i \neq l$.

Then we can find one of the following things in polynomial time

1. MAIN CONCLUSION. *Elements $w, x \in \mathbb{Z}_q$ and a permutation $\pi \in \Sigma_N$ s.t.*
$$((g, z, y, L, L'), (w, x)) \in R_{\mathrm{DP}} \enspace .$$
2. *An element $\mathbf{b} \in \mathrm{QR}_{\mathbf{N}}$, and integers $\eta_0 \neq 0$, and η_1, η_2, not both zero such that one of the following holds.*
   (a) The integer η_0 does not divide both η_1 and η_2, and $\mathbf{b}^{\eta_0} = \mathbf{h}^{\eta_1}\mathbf{g}^{\eta_2}$.
   (b) The integer η_0 does not divide η_1, and $\mathbf{b}^{\eta_0} = \mathbf{h}^{\eta_1}$.
3. *Integers η_1, η_2, not both zero, such that $\mathbf{h}^{\eta_1} = \mathbf{g}^{\eta_2}$.*
4. *Elements ρ'_i and $\rho'_{i,j}$ in $\mathbb{Z}_q$ such that $w_i = g^{\rho'_i} \prod_{j=1}^N g_j^{\rho'_{i,j}}$, and such that $(\rho'_{ij})_{i,j=1}^N$ is not a permutation matrix.*
5. *Elements $\eta_0, \ldots, \eta_N \in \mathbb{Z}_q$, not all zero, such that $g^{\eta_0} \prod_{i=1}^N g_i^{\eta_i} = 1$.*
6. *Integers $\rho_{j1}, \ldots, \rho_{jN}$, with $\rho_{ji} = \pm 1$ for at least one $1 \leq i \leq N$, such that $\prod_{i=1}^N p_{ji} = \prod_{i=1}^N \rho_{ji}$ and $\prod_{i=1}^N w_i^{p_{j,i}} = g^{\gamma'_j} \prod_{i=1}^N g_i^{\rho_{j,i}}$, in which case these integers can be extracted using only the two transcripts T_j and T_{j+N}.*[2]

Proof. The proof is quite complex and is divided into a number of cases, some of which are subdivided into sub-cases. The main track of the proof leads to the main conclusion. To aid the reader we adopt the convention that whenever the proof divides into two cases it is always the first case that leads to the main conclusion.

We first analyze the integer commitments of the protocol, and only then proceed with the analysis of the components in G_q. Our integer commitments are not standard, since the bases $\mathbf{b}_{j,i}$ are chosen by the adversary during the protocol, and we compose commitments in various ways. This said, we do use ideas from Fujisaki and Okamoto [17] and Damgård and Fujisaki [11].

[2] This case was not present in the original flawed proof. It is used later (see Footnote 3) to find a contradiction to the discrete logarithm assumption. The flaw in the original proof is discussed in Appendix K and the next modification of the proof is marked by Footnote 3 on Page 35.

The Prover Can Open $\mathbf{b}_{j,i}$ and $\mathbf{b}'_{j,i}$ To Some ρ_i. We consider Equations (20) iteratively for $i = 1, \ldots, N$. For a given i, the equations imply that

$$\begin{aligned} \mathbf{b}_{j,i}^{c_{j+N}-c_j} &= \mathbf{h}^{e_{j+N,i}-e_{j,i}} \mathbf{b}_{j,i-1}^{d_{j+N,i}-d_{j,i}} \quad , \text{ and} \\ (\mathbf{b}'_{j,i})^{c_{j+N}-c_j} &= \mathbf{h}^{e'_{j+N,i}-e'_{j,i}} \mathbf{g}^{d_{j+N,i}-d_{j,i}} \quad . \end{aligned}$$

There are several cases to consider.

1. If $c_{j+N} - c_j$ divides $e_{j+N,i} - e_{j,i}$, $d_{j+N,i} - d_{j,i}$ and $e'_{j+N,i} - e'_{j,i}$, we set $\tau_i = (e_{j+N,i} - e_{j,i})/(c_{j+N} - c_j)$, $\tau'_i = (e'_{j+N,i} - e'_{j,i})/(c_{j+N} - c_j)$ and

$$\rho_i = (d_{j+N,i} - d_{j,i})/(c_{j+N} - c_j) \ , \tag{23}$$

and conclude that

$$\mathbf{b}_{j,i} = \mathbf{h}^{\tau_i} \mathbf{b}_{j,i-1}^{\rho_i} \ , \ \text{ and } \ \mathbf{b}'_{j,i} = \mathbf{h}^{\tau'_i} \mathbf{g}^{\rho_i} \ . \tag{24}$$

We apply the equalities iteratively and get

$$\mathbf{b}_{j,N} = \mathbf{h}^{\tau} \mathbf{g}^{\prod_{i=1}^{N} \rho_i}$$

where $\tau = \tau_N + \rho_N(\tau_{N-1} + \rho_{N-1}(\tau_{N-2} + \rho_{N-2}(\tau_{N-3} + \rho_{N-3}(\ldots))))$.
2. Suppose now that $c_{j+N} - c_j$ divides $e_{j+N,i} - e_{j,i}$, $e'_{j+N,i} - e'_{j,i}$ and $d_{j+N,i} - d_{j,i}$ for all $i < l$, but not for $i = l$. Then we can only conclude that

$$\mathbf{b}_{j,l-1} = \mathbf{h}^{\tau_*} \mathbf{g}^{\prod_{i=1}^{l-1} \rho_i}$$

where $\tau_* = \tau_{l-1} + \rho_{l-1}(\tau_{l-2} + \rho_{l-2}(\tau_{l-3} + \rho_{l-3}(\tau_{l-4} + \rho_{l-4}(\ldots))))$. This implies that

$$\mathbf{b}_{j,l}^{c_{j+N}-c_j} = \mathbf{h}^{\tau_*(d_{j+N,l}-d_{j,l})+(e_{j+N,l}-e_{j,l})} \mathbf{g}^{(\prod_{i=1}^{l-1} \rho_i)(d_{j+N,i}-d_{j,i})} \ .$$

Remark 2. For readers familiar with the proof in Damgård and Fujisaki [11], we point out that here it seems impossible to conclude that one of the exponents on the left is not divisible by the exponent on the right, which is necessary to reach a contradiction to the strong RSA-assumption. This is different from [11], where no additional factors corresponding to τ_* and $\prod_{i=1}^{l-1} \rho_i$ are present. The sole purpose of the $\mathbf{b}'_{j,i}$ commitments is to handle this problem. Thus, if the problem could be solved otherwise this would simplify the protocol and improve its efficiency somewhat.

There are two cases to consider.

(a) If $c_{j+N} - c_j$ does not divide $e'_{j+N,l} - e'_{j,l}$ and $d_{j+N,l} - d_{j,l}$, then set $\mathbf{b} = \mathbf{b}'_{j,l}$, $\eta_0 = c_{j+N} - c_j$, $\eta_1 = e'_{j+N,l} - e'_{j,l}$, and $\eta_2 = d_{j+N,l} - d_{j,l}$. This implies that $\mathbf{b}^{\eta_0} = \mathbf{h}^{\eta_1} \mathbf{g}^{\eta_2}$ and η_0 does not divide both η_1 and η_2, i.e. Conclusion 2a of the lemma is satisfied.

(b) If $c_{j+N} - c_j$ divides $e'_{j+N,i} - e'_{j,i}$ and $d_{j+N,l} - d_{j,l}$, then set

$$\mathbf{b} = \mathbf{b}_{j,l} \left(\mathbf{h}^{\tau_*} \mathbf{g}^{\prod_{i=1}^{l-1} \rho_i} \right)^{-\frac{d_{j+N,l} - d_{j,l}}{c_{j+N} - c_j}} \quad ,$$

$\eta_0 = c_{j+N} - c_j$, and $\eta_1 = e_{j+N,l} - e_{j,l}$. By assumption η_0 does not divide η_1. This implies that $\mathbf{b}^{\eta_0} = \mathbf{h}^{\eta_1}$ and η_0 does not divide η_1, i.e. Conclusion 2b of the lemma is satisfied.

There is no need to consider Case 2 further, but we must show that Case 1 leads to one of the conclusions in the lemma.

Form of Exponentiated Product of Commitments. From Equation (19) we have

$$W_j^{c_{j+N} - c_j} = g^{f_{j+N,r'} - f_{j,r'}} \prod_{i=1}^{N} g_i^{d_{j+N,i} - d_{j,i}} \quad .$$

We define $\gamma'_j = (f_{j+N,r'} - f_{j,r'})/(c_{j+N} - c_j)$ and conclude that

$$\prod_{i=1}^{N} w_i^{p_{j,i}} = g^{\gamma'_j} \prod_{i=1}^{N} g_i^{\rho_i} \quad . \tag{25}$$

Each ρ_i Equals a Prime $p_{j,\pi_j(i)}$ Up to Sign For Some $\pi_j \in \Sigma_N$. Equation (21) implies that

$$(\mathbf{g}^{-\prod_{i=1}^{N} p_{ji}} \mathbf{b}_{j,N})^{c_{j+N} - c_j} = \mathbf{h}^{e_{j+N} - e_j} \quad .$$

There are two cases to consider.

1. If $c_{j+N} - c_j$ divides $e_{j+N} - e_j$ we set $\tau_* = (e_{j+N} - e_j)/(c_{j+N} - c_j)$ and conclude that

$$\mathbf{b}_{j,N} = \mathbf{h}^{\tau_*} \mathbf{g}^{\prod_{i=1}^{N} p_{ji}} \quad .$$

There are two sub-cases to consider.

(a) If $(\tau_*, \prod_{i=1}^{N} p_{ji}) = (\tau, \prod_{i=1}^{N} \rho_i)$, then it follows from unique factorization in $\mathbb{Z}$ that $\rho_i = \pm \prod_{l \in \Gamma_i} p_{j,l}$ for some disjunct sets $\Gamma_i \subset \{1, \ldots, N\}$ such that $\bigcup_i \Gamma_i = \{1, \ldots, N\}$. Thus, either $|\Gamma_i| = 1$ for all i, or there exists an i such that $\rho_i = \pm 1$. The latter combined with Equation (25) implies that Conclusion 6 is satisfied. The former implies that $\rho_i \neq \pm 1$ for all i. From the pigeon-hole principle it follows[3] that $\rho_i = \pm p_{j,\pi_j(i)}$ for some permutation $\pi_j \in \Sigma_N$.

[3] In the original proof it was claimed that $\rho_i \in [-2^{K_3+K_5}+1, 2^{K_3+K_5}-1]$, which would imply that ρ_i contained at most one prime factor. By the pigeon-hole principle it would then also contain at least one prime factor, which is what we need. Here we extract the case where some ρ_i contains more than one prime factor into Conclusion 6 and deal with this later (see Footnote 4). The flaw is discussed in Appendix K and the next modification of the original proof is marked by Footnote 4 on Page 49.

(b) If $(\tau_*, \prod_{i=1}^N p_{ji}) \neq (\tau, \prod_{i=1}^N \rho_i)$, we have

$$\mathbf{h}^{\tau-\tau_*}\mathbf{g}^{\prod_{i=1}^N \rho_i - \prod_{i=1}^N p_{ji}} = 1 \ .$$

If we set $\eta_1 = \tau - \tau_*$, and $\eta_2 = -(\prod_{i=1}^N \rho_i - \prod_{i=1}^N p_{ji})$, these integers satisfy Conclusion 3 of the lemma.

2. If $c_{j+N} - c_j$ does not divide $e_{j+N} - e_j$, then set $\mathbf{b} = \mathbf{g}^{-\prod_{i=1}^N p_{ji}} \mathbf{b}_{j,N}$, $\eta_0 = c_{j+N} - c_j$, and $\eta_1 = e_{j+N} - e_j$. Then $\mathbf{b}^{\eta_0} = \mathbf{h}^{\eta_1}$ and η_0 does not divide η_1, i.e., Conclusion 2a of the lemma is satisfied.

The only case we must consider further is Case 1a.

All ρ_i Are Positive. Equation (24) implies that

$$\prod_{i=1}^N \mathbf{b}'_{j,i} = \mathbf{h}^{\sum_{i=1}^N \tau'_i}\mathbf{g}^{\sum_{i=1}^N \rho_i} \ . \tag{26}$$

From Equation (22) we conclude that

$$\left(\mathbf{g}^{-\sum_{i=1}^N p_{j,i}} \prod_{i=1}^N \mathbf{b}'_{j,i}\right)^{c_{j+N}-c_j} = \mathbf{h}^{e'_{j+N}-e'_j} \ .$$

There are two cases to consider.

1. If $c_{j+N} - c_j$ divides $e'_{j+N} - e'_j$, we define $\tau' = (e'_{j+N} - e'_j)/(c_{j+N} - c_j)$ and conclude that

$$\prod_{i=1}^N \mathbf{b}'_{j,i} = \mathbf{h}^{\tau'}\mathbf{g}^{\sum_{i=1}^N p_{j,i}} \ .$$

There are two sub-cases to consider.

(a) If $(\tau', \sum_{i=1}^N p_{j,i}) = (\sum_{i=1}^N \tau_i, \sum_{i=1}^N \rho_i)$, we conclude that ρ_i is positive. To see this, note that $p_{j,i} > 0$ and $\rho_i = \pm p_{j,\pi_j(i)} \neq 0$. Thus, if any ρ_i is negative we have $\sum_{i=1}^N p_{j,i} > \sum_{i=1}^N \rho_i$.

(b) If $(\tau', \sum_{i=1}^N p_{j,i}) \neq (\sum_{i=1}^N \tau_i, \sum_{i=1}^N \rho_i)$, we have

$$\mathbf{h}^{\tau' - \sum_{i=1}^N \tau_i}\mathbf{g}^{\sum_{i=1}^N p_{j,i} - \sum_{i=1}^N \rho_i} = 1 \ .$$

If we set $\eta_1 = \tau' - \sum_{i=1}^N \tau_i$ and $\eta_2 = -(\sum_{i=1}^N p_{j,i} - \sum_{i=1}^N \rho_i)$, these integers satisfy Conclusion 3 of the lemma.

2. If $c_{j+N} - c_j$ does not divide $e'_{j+N} - e'_j$, then set $\mathbf{b} = \mathbf{g}^{-\sum_{i=1}^N p_{j,i}} \prod_{i=1}^N \mathbf{b}'_{j,i}$, $\eta_0 = c_{j+N} - c_j$, and $\eta_1 = e'_{j+N} - e'_j$. This implies that $\mathbf{b}^{\eta_0} = \mathbf{h}^{\eta_1}$ and η_0 does not divide η_1, i.e., Conclusion 2b of the lemma is satisfied.

We have now established that either we have $\rho_i = p_{j,\pi_j(i)}$ for some permutation π_j, or one of the Conclusions 2a, 2b, or 3 of the lemma holds. We stress that we do not necessarily have $\pi_j = \pi_{j'}$ for $j \neq j'$.

From this point on we compute in G_q, and q is prime so the exponents live in the finite field $\mathbb{Z}_q$, i.e., every non-zero element can be inverted.

THE COMMITMENTS w_i ARE ON THE EXPECTED FORM. We now argue that the commitments w_i are a commitment of a permutation π. We may rewrite Equation (25) as

$$\prod_{i=1}^{N} w_i^{p_{j,i}} = g^{\gamma'_j} \prod_{i=1}^{N} g_i^{p_{j,\pi_j(i)}} \quad , \tag{27}$$

since for each P_j the set of extracted exponents equals $p_{j,\pi_j(i)}$ for some permutation π_j.

Since the vectors $P_1, \ldots, P_N$ are linearly independent over $\mathbb{Z}_q^N$ by assumption, this means that for each i there exists coefficients $a_{i,1}, \ldots, a_{i,N} \in \mathbb{Z}_q$ such that $\sum_{j=1}^{N} a_{i,j} P_j = (\delta_{i,1}, \ldots, \delta_{i,N})$ with $\delta_{i,j} = 0$ for $j \neq i$ and $\delta_{i,i} = 1$. This implies that

$$w_i = \prod_{j=1}^{N} \left(\prod_{l=1}^{N} w_l^{p_{j,l}} \right)^{a_{i,j}} = \prod_{j=1}^{N} \left(g^{\gamma'_j} \prod_{l=1}^{N} g_l^{p_{j,\pi_j(l)}} \right)^{a_{i,j}} = g^{\rho'_i} \prod_{l=1}^{N} g_l^{\rho'_{i,l}} \quad ,$$

where $\rho'_i = \sum_{j=1}^{N} \gamma'_j a_{i,j}$ and $\rho'_{i,l} = \sum_{j=1}^{N} p_{j,\pi_j(l)} a_{i,j}$.

We expect that

$$\begin{pmatrix} \rho'_{11} & \rho'_{12} & \cdots & \rho'_{1N} \\ \rho'_{21} & \rho'_{22} & \cdots & \rho'_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ \rho'_{N1} & \rho'_{N2} & \cdots & \rho'_{NN} \end{pmatrix}$$

is a permutation matrix. If it is not, then Conclusion 4 is satisfied, so we assume it is a permutation matrix such that

$$w_i = g^{\rho'_i} g_{\pi^{-1}(i)} \quad . \tag{28}$$

ALL PERMUTATIONS USED ARE EQUAL. We must show that all permutations π_j are equal. If $\pi_j \neq \pi$, we have from Equation (27) and (28) that

$$g^{\gamma'_j} \prod_{i=1}^{N} g_i^{p_{j,\pi_j(i)}} = \prod_{i=1}^{N} w_i^{p_{j,i}} = g^{\sum_{i=1}^{N} \rho'_i p_{j,i}} \prod_{i=1}^{N} g_i^{p_{j,\pi(i)}}$$

with some $p_{j,\pi_j(i)} - p_{j,\pi(i)} \neq 0$, since $p_{j,i} \neq p_{j,l}$ if $i \neq l$ by assumption. Then setting $\eta_0 = \gamma'_j - \sum_{i=1}^{N} \rho'_i p_{j,i}$ and $\eta_i = p_{j,\pi_j(i)} - p_{j,\pi(i)}$ implies $g^{\eta_0} \prod_{i=1}^{N} g_i^{\eta_i} = 1$ and Conclusion 5 of the lemma is satisfied.

THE COMMON INPUT (g, z, y, L, L') SATISFIES $((g, z, y, L, L'), (w, x)) \in R_{\mathrm{DP}}$. From the above we may assume that $\pi_j = \pi$ for all $j = 1, \ldots, N$, so we drop the subscript and simply write π from now on. Equations (15) and (16) imply that

$$
\begin{aligned}
b_1^{c_{j+N}-c_j} &= g^{f_{j+N,1}-f_{j,1}} U^{f_{j+N,1/w}-f_{j,1/w}} \ , \\
b_2^{c_{j+N}-c_j} &= g^{f_{j+N,2}-f_{j,2}} U^{f_{j+N,x/w}-f_{j,x/w}} \ , \\
b_3^{c_{j+N}-c_j} &= g^{f_{j+N,3}-f_{j,3}} g^{f_{j+N,1/w}-f_{j,1/w}} \ , \text{ and} \\
b_4^{c_{j+N}-c_j} &= g^{f_{j+N,6}-f_{j,6}} U^{f_{j+N,x/w}-f_{j,x/w}} \ .
\end{aligned}
$$

Define

$$
\begin{aligned}
\omega' &= (f_{j+N,1/w} - f_{j,1/w})/(c_{j+N} - c_j) \ , \\
\xi' &= (f_{j+N,x/w} - f_{j,x/w})/(c_{j+N} - c_j) \ , \\
\kappa_1 &= (f_{j+N,1} - f_{j,1})/(c_{j+N} - c_j) \ , && (29) \\
\kappa_2 &= (f_{j+N,2} - f_{j,2})/(c_{j+N} - c_j) \ , && (30) \\
\kappa_3 &= (f_{j+N,3} - f_{j,3})/(c_{j+N} - c_j) \ , \text{ and} && (31) \\
\kappa_6 &= (f_{j+N,6} - f_{j,6})/(c_{j+N} - c_j) \ . && (32)
\end{aligned}
$$

Then we have

$$
\begin{aligned}
b_1 &= g^{\kappa_1} U^{\omega'} \ , && (33) \\
b_2 &= g^{\kappa_2} U^{\xi'} \ , && (34) \\
b_3 &= g_1^{\kappa_3} g^{\omega'} \ , \text{ and} && (35) \\
b_4 &= g_1^{\kappa_6} g^{\xi'} \ . && (36)
\end{aligned}
$$

Equations (17) and (18) imply that

$$
\begin{aligned}
b_4^{c_{j+N}-c_j} &= g_1^{f_{j+N,4}-f_{j,4}} b_3^{f_{j+N,x}-f_{j,x}} \ , && (37) \\
y^{c_{j+N}-c_j} &= g^{f_{j+N,x}-f_{j,x}} \ , && (38) \\
b_5^{c_{j+N}-c_j} &= g_1^{f_{j+N,5}-f_{j,5}} b_3^{f_{j+N,w}-f_{j,w}} \ , && (39) \\
z^{c_{j+N}-c_j} &= g^{f_{j+N,w}-f_{j,w}} \ , && (40) \\
(b_5/g)^{c_{j+N}-c_j} &= g^{f_{j+N,7}-f_{j,7}} \ . && (41)
\end{aligned}
$$

Define $\omega = (f_{j+N,w} - f_{j,w})/(c_{j+N} - c_j)$ $\xi = (f_{j+N,x} - f_{j,x})/(c_{j+N} - c_j)$, and

$$
\begin{aligned}
\omega &= (f_{j+N,w} - f_{j,w})/(c_{j+N} - c_j) \ , \\
\xi &= (f_{j+N,x} - f_{j,x})/(c_{j+N} - c_j) \ , \\
\kappa_4 &= (f_{j+N,4} - f_{j,4})/(c_{j+N} - c_j) \ , && (42) \\
\kappa_5 &= (f_{j+N,5} - f_{j,5})/(c_{j+N} - c_j) \ , \text{ and} && (43) \\
\kappa_7 &= (f_{j+N,7} - f_{j,7})/(c_{j+N} - c_j) \ . && (44)
\end{aligned}
$$

Then we have $g^x = y = g^\xi$ and $g^w = z = g^\omega$, so $\xi = x$ and $\omega = w$. This means that we have

$$
\begin{aligned}
b_4 &= g_1^{\kappa_4} b_3^x && (45) \\
b_5 &= g_1^{\kappa_5} b_3^w && (46) \\
b_5 &= g_1^{\kappa_7} g && (47)
\end{aligned}
$$

If we combine Equation (35) and Equation (46) we get

$$b_5 = g_1^{\kappa_5} g_1^{w\kappa_3} g^{w\omega'} \quad . \tag{48}$$

There are two cases

1. If $\omega' = 1/w$ we conclude from Equations (35) and (36) that

$$b_3 = g_1^{\kappa_3} g^{1/w} \quad , \text{ and} \tag{49}$$
$$b_4 = g_1^{\kappa_4 + x/\kappa_3} g^{x/w} \quad . \tag{50}$$

   If $\xi' \neq x/w$, we set $\eta_0 = \kappa_4 + x\kappa_3 - \kappa_6$ and $\eta_1 = x/w - \xi'$. This gives $g^{\eta_0} g_1^{\eta_1} = 1$, i.e., Conclusion 5 is satisfied. Thus, we may assume that $\xi' = x/w$. Combined with Equations (33) and (34) this gives

$$b_1 = g^{\kappa_1} U^{1/w} \quad , \text{ and} \tag{51}$$
$$b_2 = g^{\kappa_2} U^{x/w} \quad . \tag{52}$$

2. If $\omega' \neq 1/w$ we define $\eta_0 = w\omega' - 1$ and $\eta_1 = \kappa_5 + w\kappa_3 - \kappa_7$. From Equations (48) and (47) we conclude that $g^{\eta_0} g_1^{\eta_1} = 1$, i.e., Conclusion 5 is satisfied.

   From Equation (19) we have

$$\begin{aligned} b_1^{c_{j+N} - c_j} &= g^{f_{j+N,1} - f_{j,1}} \prod_{i=1}^{N} (u_i')^{d_{j+N,i} - d_{j,i}} \quad , \text{ and} \\ (V/b_2)^{c_{j+N} - c_j} &= g^{-(f_{j+N,2} - f_{j,2})} \prod_{i=1}^{N} (v_i')^{d_{j+N,i} - d_{j,i}} \quad . \end{aligned}$$

Our definitions of κ_1 and κ_2 in Equations (29) and (30), and the definition of ρ_i, which equals $p_{j,\pi(i)}$ in Equation (23) imply that

$$\begin{aligned} b_1 &= g^{\kappa_1} \prod_{i=1}^{N} (u_i')^{p_{j,\pi(i)}} \quad , \text{ and} \\ V/b_2 &= g^{-\kappa_2} \prod_{i=1}^{N} (v_i')^{p_{j,\pi(i)}} \quad . \end{aligned}$$

If we combine Equations (51) and (52) with the two equations above we have

$$\prod_{i=1}^{N} (u_i^{1/w})^{p_{ji}} = \quad U^{1/w} \quad = \prod_{i=1}^{N} (u_i')^{p_{j,\pi(i)}} \quad , \text{ and} \tag{53}$$
$$\prod_{i=1}^{N} (v_i/u_i^{x/w})^{p_{ji}} = V/U^{x/w} = \prod_{i=1}^{N} (v_i')^{p_{j,\pi(i)}} \quad , \tag{54}$$

for $j = 1, \ldots, N$. We apply the coefficients $a_{l,1}, \ldots, a_{l,N} \in \mathbb{Z}_q$ introduced above to the Equations (53) and (54) and conclude that

$$u_l^{1/w} = \prod_{j=1}^{N} \left(\prod_{i=1}^{N} (u_i^{1/w})^{p_{ji}} \right)^{a_{l,j}} = \prod_{j=1}^{N} \left(\prod_{i=1}^{N} (u_i')^{p_{j,\pi(i)}} \right)^{a_{l,j}} = u'_{\pi^{-1}(l)} \ , \ \text{and}$$

$$v_l/u_l^{x/w} = \prod_{j=1}^{N} \left(\prod_{i=1}^{N} (v_i/u_i^{x/w})^{p_{ji}} \right)^{a_{l,j}} = \prod_{j=1}^{N} \left(\prod_{i=1}^{N} (v_i')^{p_{j,\pi(i)}} \right)^{a_{l,j}} = v'_{\pi^{-1}(l)} \ .$$

This concludes the proof.

Random Prime Vectors are Linearly Independent. We show that given linearly independent vectors $P_1, \ldots, P_{j-1} \in \mathbb{Z}_q^N$, a random prime vector P_j is contained in the subspace spanned by $P_1, \ldots, P_{j-1}$ with negligible probability as long as $j \leq N$.

Lemma 4. *Let $P_1, \ldots, P_{j-1} \in \mathbb{Z}_q^N$ be linearly independent vectors. If $P_j = (p_{j,1}, \ldots, p_{j,N})$ is a list of independently and identically distributed primes such that $\Pr[p_{j,i} = p] \leq \epsilon$ for every prime p, we have*

$$\Pr[P_j \in \mathrm{Span}(P_1, \ldots, P_{j-1})] \leq \epsilon^{N-j+1} \ .$$

Proof. The vectors $P_1, \ldots, P_{j-1}$ form a matrix

$$P = \left(\begin{array}{cccc|ccc} p_{1,1} & p_{1,2} & \cdots & p_{1,j-1} & p_{1,j} & \cdots & p_{1,N} \\ p_{2,1} & p_{2,2} & \cdots & p_{2,j-1} & p_{2,j} & \cdots & p_{2,N} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ p_{j-1,1} & p_{j-1,2} & \cdots & p_{j-1,j-1} & p_{j-1,j} & \cdots & p_{j-1,N} \end{array} \right) \ .$$

Suppose that we replace $P_1, \ldots, P_{j-1}$ with a set of rows $P_1', \ldots, P_{j-1}'$ formed by elementary row operations. Then clearly $\Pr[P_j \in \mathrm{Span}(P_1, \ldots, P_{j-1})] = \Pr[P_j \in \mathrm{Span}(P_1', \ldots, P_{j-1}')]$, since $\mathrm{Span}(P_1', \ldots, P_{j-1}') = \mathrm{Span}(P_1, \ldots, P_{j-1})$. Given a permutation π of N elements, we denote by P_i^π the vector with permuted components defined as $(p_{i,\pi(1)}, p_{i,\pi(2)}, \ldots, p_{i,\pi(N)})$. Since the primes $p_{j,1}, \ldots, p_{j,N}$ are identically and independently distributed we conclude that

$$\Pr[P_j \in \mathrm{Span}(P_1, \ldots, P_{j-1})] = \Pr[P_j \in \mathrm{Span}(P_1^\pi, \ldots, P_{j-1}^\pi)]$$

for every permutation π of N elements.

Thus, we can by elementary row operations and by permuting the columns in the matrix form a new matrix P' from P on the form

$$P' = \begin{pmatrix} P_1' \\ P_2' \\ \vdots \\ P_{j-1}' \end{pmatrix} = \begin{pmatrix} 1 & 0 & \cdots & 0 & p'_{1,j} & \cdots & p'_{1,N} \\ 0 & 1 & \cdots & 0 & p'_{2,j} & \cdots & p'_{2,N} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 & p'_{j-1,j} & \cdots & p'_{j-1,N} \end{pmatrix} \ ,$$

with $\Pr[P_j \in \mathrm{Span}(P_1, \ldots, P_{j-1})] = \Pr[P_j \in \mathrm{Span}(P'_1, \ldots, P'_{j-1})]$. We have $P_j \in \mathrm{Span}(P'_1, \ldots, P'_{j-1})$ if and only if

$$\sum_{i=1}^{j-1} p_{j,i} p'_{i,l} = p_{j,l}$$

for $l = j, \ldots, N$. From independence we conclude that the probability of this event is at most ϵ^{N-j+1}, which concludes the proof.

Corollary 3. *Let $P_1, \ldots, P_{j-1} \in \mathbb{Z}_q^N$ be linearly independent vectors and define $P_j = \mathsf{PGen}(\mathsf{PRG}(s))$ for a randomly chosen seed $s \in [0, 2^{K_1} - 1]$.*

Then $\Pr[P_j \in \mathrm{Span}(P_1, \ldots, P_{j-1})]$ is negligible.

Proof. Suppose that $\Pr[P_j \in \mathrm{Span}(P_1, \ldots, P_{j-1})] \geq 1/{K_1}^c$ for some linearly independent vectors $P_1, \ldots, P_{j-1} \in \mathbb{Z}_q^N$ and K_1 in some infinite set $\mathcal{N}$ of security parameters. Consider the distinguisher A that is given $(n_1, \ldots, n_N)$ generated either by choosing $n_i \in [0, 2^{K_3} - 1]$ randomly, or by choosing $s \in [0, 2^{K_1} - 1]$ randomly and computing $(n_1, \ldots, n_N) = \mathsf{PRG}(s)$. The distinguisher simply checks if $P_j \in \mathrm{Span}(P_1, \ldots, P_{j-1})$ and outputs 1 or 0 depending on the result. It follows from Lemma 4 that A can distinguish uniformly and independently generated integers from pseudo-randomly generated integers. This contradicts the fact that PRG is a pseudo-random generator as in Definition 9.

A Non-Permutation Matrix Does Not Behave Like One.

Lemma 5. *Let $B = (b_{ij})$ be an $N \times N$-matrix over $\mathbb{Z}_q$ and let $X = (X_1, \ldots, X_N)$ consist of independently and identically distributed random variables such that $\Pr[X_i = x_i] \leq \epsilon$ for all $x_i \in \mathbb{Z}_q$. Then if B is not a permutation matrix*

$$\Pr[\exists \pi \in \Sigma_N \;\; s.t. \;\; BX = X^\pi] \leq N\epsilon \enspace .$$

Proof. The set of permutation matrices are characterized as the matrices such that in each row and and in column exactly one element equal one and the rest are zero. We say that a column or row is bad if it does not have the above property. Thus, if B is not a permutation matrix, it must have a bad row or a bad column.

Suppose first that there is a bad row. Without loss we assume that the first row is bad. We have

$$\Pr[\exists \pi \in \Sigma_N \;\text{ s.t. }\; BX = X^\pi] \leq \Pr\left[\exists 1 \leq j \leq N \;\text{ s.t. }\; \sum_{l=1}^N b_{1l} X_l - X_j = 0\right]$$
$$\leq \sum_{j=1}^N \Pr\left[\sum_{l=1}^N b_{1l} X_l - X_j = 0\right] \leq N\epsilon \enspace .$$

where we use the union bound, and the fact that $\Pr[\sum_{l=1}^N b_{il} X_l - X_j = 0] \leq \epsilon$, since the expression $\sum_{l=1}^N b_{il} X_l - X_j$ is not identically zero for any j.

Suppose now that no row is bad. Then there are exactly N ones in total in the matrix B, so if some column is bad there must be an all zero column as well. Suppose that the jth column is an all zero column, i.e., $b_{i,j} = 0$ for $i = 1, \ldots, N$. This implies that $\Pr[\sum_{l=1}^{N} b_{il} X_l = X_j] \leq \epsilon$ for all i, since X_j is independent of all expressions $\sum_{l=1}^{N} b_{il} X_l$ for $i = 1, \ldots, N$. Thus, it follows that $\Pr[\exists \pi \in \Sigma_N \text{ s.t. } BX = X^{\pi}] \leq N\epsilon$ also in this case.

Corollary 4. *Let $B = (b_{ij})$ be an $N \times N$-matrix over $\mathbb{Z}_q$ and define $P = \mathsf{PGen}(\mathsf{PRG}(s))$ for a randomly chosen seed $s \in [0, 2^{K_1} - 1]$. Then if B is not a permutation matrix*

$$\Pr[\exists \pi \in \Sigma_N \ \ s.t. \ \ BP = P^{\pi}] \leq N\epsilon \ .$$

Proof. The proof is almost identical to the proof of Corollary 3 and omitted.

Generation of Suitable Transcripts. We describe how the adversary can extract transcripts that satisfy Lemma 3. Let A be the machine that given an RSA modulus $\mathbf{N}$ and $\mathbf{g}, \mathbf{h} \in \mathrm{QR}_{\mathbf{N}}$, and $g, g_1, \ldots, g_N \in G_q$ as input generates the remainder of the common input consisting of a public key $z, y \in G_q$, and two lists $L = (u_i, v_i)_{i=1}^{N} \in G_q^{2N}$ and $L' = (u_i', v_i')_{i=1}^{N} \in G_q^{2N}$, and then plays the role of the prover in Protocol 2 on common input $((\mathbf{N}, \mathbf{g}, \mathbf{h}), (g, g_1, \ldots, g_N), (z, y), (L, L'))$. To simplify the exposition we sometimes write $\boldsymbol{\Gamma}$ and $\overline{g}$ instead of $(\mathbf{N}, \mathbf{g}, \mathbf{h})$ and $(g_1, \ldots, g_N)$ respectively.

We consider the transcript from the interaction of A with an honest verifier as a list of functions $T_A = (I_A, W_A, P, C_A, c, R_A)$, where I_A is the part of the common input constructed by A, W_A is the first commitments computed by A, P is the list of N primes chosen by the verifier, C_A is the second set of commitments of the prover, c is the challenge chosen by the verifier, and R_A is the reply of the prover. Denote by r_{p} and r_{v} the random input of the prover and verifier respectively. Then T_A is clearly a function of $\boldsymbol{\Gamma} = (\mathbf{N}, \mathbf{g}, \mathbf{h})$, $\overline{g} = (g_1, \ldots, g_N)$, r_{p}, and r_{v}, but not all parts depend on all variables. If we divide r_{v} in two parts ${r_{\mathrm{v}}}'$ and ${r_{\mathrm{v}}}''$, where the former is used to construct the list of primes P, and the latter is used to construct the challenge c the dependencies are given by

$$\begin{aligned} T_A(\boldsymbol{\Gamma}, \overline{g}, r_{\mathrm{p}}, r_{\mathrm{v}}) = \big(& I_A(\boldsymbol{\Gamma}, \overline{g}, r_{\mathrm{p}}), W_A(\boldsymbol{\Gamma}, \overline{g}, r_{\mathrm{p}}), P({r_{\mathrm{v}}}'), C_A(\boldsymbol{\Gamma}, \overline{g}, r_{\mathrm{p}}, {r_{\mathrm{v}}}'), \\ & c({r_{\mathrm{v}}}''), R_A(\boldsymbol{\Gamma}, \overline{g}, r_{\mathrm{p}}, {r_{\mathrm{v}}}', {r_{\mathrm{v}}}'') \big) \ . \end{aligned}$$

Denote by Acc a predicate that given a transcript T outputs 1 or 0, depending on if the verifier accepts the proof or not, and define

$$\delta_A(\boldsymbol{\Gamma}, \overline{g}, r_{\mathrm{p}}) = \Pr_{r_{\mathrm{v}}}[\mathrm{Acc}(T_A(\boldsymbol{\Gamma}, \overline{g}, r_{\mathrm{p}}, r_{\mathrm{v}})) = 1] \ .$$

This is the probability that the prover A outputs an accepting transcript on input $(\boldsymbol{\Gamma}, \overline{g})$, when run on the fixed random input r_{p}. The probability is taken over the random choices of the honest verifier.

Denote by B denote the distribution on a bit defined by $\Pr_{b \leftarrow B}[b = 1] = \delta_A(\boldsymbol{\Gamma}, \overline{g}, r_{\mathrm{p}})/32$. This distribution can be sampled efficiently without explicit knowledge of $\delta_A(\boldsymbol{\Gamma}, \overline{g}, r_\mathrm{p})$ by choosing $r_\mathrm{v} \in \{0,1\}^*$ and $n \in \{1, \ldots, 32\}$ randomly and outputting 1 if $\mathrm{Acc}(T_A(\boldsymbol{\Gamma}, \overline{g}, r_\mathrm{p}, r_\mathrm{v})) = 1$ and $n = 1$, and 0 otherwise.

We define an algorithm $\mathcal{FF}$ that given $(\boldsymbol{\Gamma}, \overline{g})$ and a list $(P_1, \ldots, P_{j-1})$ of vectors in $\mathbb{Z}_q^N$ outputs a pair of transcripts (T_j, T_{j+N}) such that $\mathrm{Acc}(T_j) = 1$, $\mathrm{Acc}(T_{j+N}) = 1$, $(I_j, W_j, P_j) = (I_{j+N}, W_{j+N}, P_{j+N})$, and $c_j \neq c_{j+N}$. Furthermore, $P_j \notin \mathrm{Span}(P_1, \ldots, P_{j-1})$. Note that this is a fork in the sense of the forking lemma.

Algorithm 1 (Fork Finder).

$\mathcal{FF}(\boldsymbol{\Gamma}, \overline{g}, r_\mathrm{p}, (P_1, \ldots, P_{j-1}))$
`Loop`
    `Do {`
        ${r_\mathrm{v}}', {r_\mathrm{v}}'' \leftarrow \{0,1\}^*$
        $T_j \leftarrow T(\boldsymbol{\Gamma}, \overline{g}, r_\mathrm{p}, {r_\mathrm{v}}', {r_\mathrm{v}}'')$
    `} While` $(\mathrm{Acc}(T_j) = 0$ or $P_j \in \mathrm{Span}(P_1, \ldots, P_{j-1})$
                or $\exists i, l : i \neq l$ and $p_{j,i} = p_{j,l})$
    `Do {`
        ${r_\mathrm{v}}''' \leftarrow \{0,1\}^*$, $b \leftarrow B$
        $T_{j+N} \leftarrow T(\boldsymbol{\Gamma}, \overline{g}, {r_\mathrm{v}}', {r_\mathrm{v}}''')$
    `} While` $(\mathrm{Acc}(T_{j+N}) = 0$ or $c_{j+N} = c_j)$ and $b \neq 1)$
    `If` $((\mathrm{Acc}(T_{j+N}) = 1$ and $c_{j+N} = c_j)$ `Then`
        `Return` (T_j, T_{j+N})
    `EndIf`
`EndLoop`

Lemma 6. *Consider an execution of $\mathcal{FF}$ on input $(\boldsymbol{\Gamma}, \overline{g}, r_\mathrm{p}, (P_1, \ldots, P_{j-1}))$. If $\delta_A(\boldsymbol{\Gamma}, \overline{g}, r_\mathrm{p})$ is non-negligible the expected number of times it invokes A is $O(1/\delta_A)$.*

Proof. Consider a fixed iteration of the outer loop. By definition, the first condition in the first while-loop is satisfied with probability $\delta_A(\boldsymbol{\Gamma}, \overline{g}, r_\mathrm{p})$. Corollary 3 and Corollary 1 imply that the second and third conditions are satisfied with overwhelming probability. The union bound then implies that both conditions are satisfied with probability at least $\delta_A(\boldsymbol{\Gamma}, \overline{g}, r_\mathrm{p})/2$. Thus, the expected number of iterations of the loop and the number of invocations of A is bounded by $2/\delta_A(\boldsymbol{\Gamma}, \overline{g}, r_\mathrm{p})$.

Consider now the second while-loop. Denote by I_heavy the set of ${r_\mathrm{v}}'$ such that

$$\Pr_{r_\mathrm{v}}[\mathrm{Acc}(T_A(\boldsymbol{\Gamma}, \overline{g}, r_\mathrm{p}, r_\mathrm{v})) = 1 \wedge P_j \notin \mathrm{Span}(P_1, \ldots, P_{j-1}) \mid {r_\mathrm{v}}' \in I_\mathrm{heavy}] \\ \geq \delta_A(\boldsymbol{\Gamma}, \overline{g}, r_\mathrm{p})/4 \ .$$

An averaging argument implies that

$$\Pr_{r_\mathrm{v}}[{r_\mathrm{v}}' \in I_\mathrm{heavy} \mid \mathrm{Acc}(T_A(\boldsymbol{\Gamma}, \overline{g}, r_\mathrm{p}, r_\mathrm{v})) = 1 \wedge P_j \notin \mathrm{Span}(P_1, \ldots, P_{j-1})] \geq 1/2 \ .$$

Thus, with probability 1/2 the value of ${r_\mathrm{v}}'$ fixed in the first loop belongs to I_heavy.

By definition of the distribution B we have $\Pr[b=1] = \delta_A(\boldsymbol{\Gamma}, \overline{g}, r_{\mathrm{p}})/32$. Thus, if the first condition of the while-loop is removed the expected number of iterations is $32/\delta_A(\boldsymbol{\Gamma}, \overline{g}, r_{\mathrm{p}})$, and the probability that more than $16/\delta_A(\boldsymbol{\Gamma}, \overline{g}, r_{\mathrm{p}})$ iterations are needed is at least $(1-\delta_A(\boldsymbol{\Gamma}, \overline{g}, r_{\mathrm{p}})/32)^{16/\delta_A(\boldsymbol{\Gamma}, \overline{g}, r_{\mathrm{p}})}$. Set $\delta = \delta_A(\boldsymbol{\Gamma}, \overline{g}, r_{\mathrm{p}})/32$. Then this can be bounded by $(1-\delta)^{1/2\delta} \geq (e^{-\delta-\delta^2/2})^{1/2\delta} \geq e^{-3/4} \geq 2/5$

The number of iterations in the unmodified while-loop is obviously also bounded by $32/\delta_A(\boldsymbol{\Gamma}, \overline{g}, r_{\mathrm{p}})$. Thus, the expected number of invocations of A in the second while-loop is bounded by $64/\delta_A(\boldsymbol{\Gamma}, \overline{g}, r_{\mathrm{p}})$, since one invocation is needed in the construction of the transcript and one invocation in the sampling of B.

We must estimate the probability that last if-statement is satisfied, conditioned on $r_{\mathrm{v}}{}' \in I_{\mathrm{heavy}}$. We estimate the probability that the second while-loop is terminated by the first condition. The probability that $c_{j+N} = c_j$ is 2^{K_4-1}. Thus, the union bound implies that the first condition is satisfied, conditioned on $r_{\mathrm{v}}{}' \in I_{\mathrm{heavy}}$, with probability at least $\delta_A(\boldsymbol{\Gamma}, \overline{g}, r_{\mathrm{p}})/8$. If the second condition of the while-loop is removed the expected number of iterations would then be bounded by $8/\delta_A(\boldsymbol{\Gamma}, \overline{g}, r_{\mathrm{p}})$ and the probability that more than $16/\delta_A(\boldsymbol{\Gamma}, \overline{g}, r_{\mathrm{p}})$ iterations are necessary is at most $1/2$ by Markov's inequality. This implies that the probability that the second loop is terminated by the first condition is at least $\frac{1}{2}\frac{2}{5} = \frac{1}{5}$, conditioned on $r_{\mathrm{v}}{}' \in I_{\mathrm{heavy}}$.

We have argued that the expected number of invocations in a single iteration of the outer loop is $O(1/\delta_A(\boldsymbol{\Gamma}, \overline{g}, r_{\mathrm{p}}))$ and the probability that the return statement is reached in such an iteration is at least $\frac{1}{2}\frac{1}{5} = \frac{1}{10}$, and the claim follows.

Algorithm 2 (Transcript Finder).

```
𝒯ℱ(Γ, ḡ, r_p)
For (j = 1, ..., N) Do
      (T_j, T_{j+N}) ← ℱℱ(Γ, ḡ, r_p, (P_1, ..., P_{j-1}))
EndLoop
Return (T_1, ..., T_{2N})
```

Corollary 5. *Consider an execution of the algorithm $\mathcal{TF}$ on input $(\boldsymbol{\Gamma}, \overline{g}, r_{\mathrm{p}})$. If $\delta_A(\boldsymbol{\Gamma}, \overline{g}, r_{\mathrm{p}})$ is non-negligible, the expected number of times it invokes A is at most $O(N/\delta_A(\boldsymbol{\Gamma}, \overline{g}, r_{\mathrm{p}}))$. The output satisfy the hypothesis of Lemma 3.*

Proof. This follows immediately from Lemma 6.

Concluding that the Protocol is Sound. We now show that if there exists an adversary that can generate a common input $(g, z, y, (L, L')) \notin L_{R_{\mathrm{DP}}}$, and still convince the verifier of the contrary, we can construct algorithms that break either the strong RSA-assumption or the DDH-assumption. More precisely we prove Proposition 2.

Denote by $\mathcal{C}_{\mathrm{RSA}}, \mathcal{C}_{\mathrm{RSA}'}, \mathcal{C}_{\mathrm{REP}}$, the algorithms that given transcripts $T_1, \ldots, T_{2N}$ that satisfy the hypothesis of Lemma 3 try to extract values corresponding to Conclusion 2, Conclusion 3, or one of Conclusion 4, 5, or 6 respectively. If this is not possible the algorithms output $\perp$.

Proof (Proposition 2 on Page 16). Assume that the proposition is false. Then there exists an adversary A, a constant c, and an infinite index set $\mathcal{N}$ that shows that the inequality is false. In the rest of the proof we consider only security parameters in $\mathcal{N}$. Denote by I_{good} the set of good tuples, i.e. tuples such that

$$\Pr_{\boldsymbol{\Gamma},\overline{g},r_{\mathrm{p}},r_{\mathrm{v}}}[\mathrm{Acc}(T_A(\boldsymbol{\Gamma},\overline{g},r_{\mathrm{p}},r_{\mathrm{v}}))=1 \wedge I_A(\boldsymbol{\Gamma},\overline{g},r_{\mathrm{p}}) \notin L_{R_{\mathrm{DP}}} \mid (\boldsymbol{\Gamma},\overline{g},r_{\mathrm{p}}) \in I_{\mathrm{good}}]$$
$$\geq \frac{1}{2{K_1}^c} \ .$$

It follows straightforwardly that the probability that a random tuple is good is relatively high as well.

Claim 1. $\Pr_{\boldsymbol{\Gamma},\overline{g},r_{\mathrm{p}}}[(\boldsymbol{\Gamma},\overline{g},r_{\mathrm{p}}) \in I_{\mathrm{good}}] \geq \frac{1}{2{K_1}^c}$.

Claim 2. $(\boldsymbol{\Gamma},\overline{g},r_{\mathrm{p}}) \in I_{\mathrm{good}}$ implies that $I_A(\boldsymbol{\Gamma},\overline{g},r_{\mathrm{p}}) \notin L_{R_{\mathrm{DP}}}$.

Proof. Consider any fixed tuple $(\boldsymbol{\Gamma},\overline{g},r_{\mathrm{p}})$ such that $I_A(\boldsymbol{\Gamma},\overline{g},r_{\mathrm{p}}) \in L_{R_{\mathrm{DP}}}$. Then we must have $\Pr_{r_{\mathrm{v}}}[\mathrm{Acc}(T_A(\boldsymbol{\Gamma},\overline{g},r_{\mathrm{p}},r_{\mathrm{v}}))=1 \wedge I_A(\boldsymbol{\Gamma},\overline{g},r_{\mathrm{p}}) \notin L_{R_{\mathrm{DP}}}]=0$.

By definition of I_{good} and Corollary 5, $\mathcal{TF}$ invokes A expected $O(N{K_1}^c)$ times on inputs from I_{good}. Similarly, $\mathcal{FF}$ invokes A expected $O({K_1}^c)$ times on such inputs. In the following we assume that these algorithms are turned into strict polynomial machines by bounding the number of invocations of A to twice the number of expected invocations. By Markov's inequality the resulting algorithms delivers an output with probability at least $1/2$.

Suppose now we execute $\mathcal{TF}$ on a fixed input $(\boldsymbol{\Gamma},\overline{g},r_{\mathrm{p}}) \in I_{\mathrm{good}}$. Then $\mathcal{TF}$ delivers a set of transcripts $T_1,\ldots,T_{2N}$ that satisfy the hypothesis of Lemma 3 with probability $1/2$, and as a consequence we have one of the conclusions of the lemma. Claim 2 implies that the main conclusion of the lemma cannot be true. Thus, one of the other conclusions must be true if $\mathcal{TF}$ gives an output. A simple averaging argument implies that one of the following equations must hold

$$\Pr_{r_{\mathcal{TF}}}[\mathcal{C}_{\mathrm{RSA}}(\mathcal{TF}(\boldsymbol{\Gamma},\overline{g},r_{\mathrm{p}})) \neq \perp] \geq 1/6 \ ,$$
$$\Pr_{r_{\mathcal{TF}}}[\mathcal{C}_{\mathrm{RSA}'}(\mathcal{TF}(\boldsymbol{\Gamma},\overline{g},r_{\mathrm{p}})) \neq \perp] \geq 1/6 \ , \quad \text{or}$$
$$\Pr_{r_{\mathcal{TF}}}[\mathcal{C}_{\mathrm{REP}}(\mathcal{TF}(\boldsymbol{\Gamma},\overline{g},r_{\mathrm{p}})) \neq \perp] \geq 1/6 \ ,$$

where the probability is taken over the internal randomness $r_{\mathcal{TF}}$ of $\mathcal{TF}$. As a consequence, we also have

$$\Pr_{\boldsymbol{\Gamma},\overline{g},r_{\mathrm{p}},r_{\mathcal{TF}}}[\mathcal{C}_{\mathrm{RSA}}(\mathcal{TF}(\boldsymbol{\Gamma},\overline{g},r_{\mathrm{p}})) \neq \perp \mid (\boldsymbol{\Gamma},\overline{g},r_{\mathrm{p}}) \in I_{\mathrm{good}}] \geq 1/6 \ , \tag{55}$$
$$\Pr_{\boldsymbol{\Gamma},\overline{g},r_{\mathrm{p}},r_{\mathcal{TF}}}[\mathcal{C}_{\mathrm{RSA}'}(\mathcal{TF}(\boldsymbol{\Gamma},\overline{g},r_{\mathrm{p}})) \neq \perp \mid (\boldsymbol{\Gamma},\overline{g},r_{\mathrm{p}}) \in I_{\mathrm{good}}] \geq 1/6 \ , \quad \text{or} \tag{56}$$
$$\Pr_{\boldsymbol{\Gamma},\overline{g},r_{\mathrm{p}},r_{\mathcal{TF}}}[\mathcal{C}_{\mathrm{REP}}(\mathcal{TF}(\boldsymbol{\Gamma},\overline{g},r_{\mathrm{p}})) \neq \perp \mid (\boldsymbol{\Gamma},\overline{g},r_{\mathrm{p}}) \in I_{\mathrm{good}}] \geq 1/6 \ . \tag{57}$$

To conclude the proof of the lemma we show that if the first or second equations hold, then the strong RSA-assumption is false, and if the third equation holds the DL-assumption is false.

Claim 3. If Equation (55) holds, then the strong RSA-assumption is false.

Proof. We describe an adversary that contradicts the strong RSA-assumption. Our proof follows the proof given in [11] closely. Denote by extgcd the extended Euclidean algorithm, i.e., given input (η_0, η_1) it outputs (f, a, b), where $f = \gcd(\eta_0, \eta_1)$ and $f = a\eta_0 + b\eta_1$. The adversary tries to extract a non-trivial root of $\mathbf{h}$.

Algorithm 3 (Root Finder).

$\mathcal{RSAR}(\mathbf{N}, \mathbf{h})$
$r_\mathrm{p} \leftarrow \{0,1\}^*$
$e \leftarrow [0, 2^{K_2+K_5} - 1]$
$\mathbf{g} \leftarrow \mathbf{h}^e$
$g_1, \ldots, g_N \leftarrow G_q$
$(T_1, \ldots, T_{2N}) \leftarrow \mathcal{TF}((\mathbf{N}, \mathbf{g}, \mathbf{h}), (g, g_1, \ldots, g_N), r_\mathrm{p})$
$(\mathbf{b}, \eta_0, \eta_1, \eta_2) \leftarrow \mathcal{C}_{\mathrm{RSA}}(T_1, \ldots, T_{2N})$
$(f, a, b) \leftarrow \mathrm{extgcd}(\eta_0, \eta_1 + e\eta_2)$
$(\mathbf{b}, \eta) \leftarrow (\mathbf{g}^a \mathbf{b}^b, \eta_0 / f)$
`Return` $(\mathbf{b}, \eta)$

With the modification of $\mathcal{TF}$ above the algorithm runs in polynomial time, since the adversary A is assumed to be polynomial time. We must argue that its success probability is notable.

The probability that the input $(\boldsymbol{\Gamma}, \overline{g}, r_\mathrm{p}) = ((\mathbf{N}, \mathbf{g}, \mathbf{h}), (g, g_1, \ldots, g_N), r_\mathrm{p})$ belongs to I_{good} is at least $\frac{1}{2{K_1}^c}$ by Claim 1. By assumption the probability that $\mathcal{C}_{\mathrm{RSA}}(\mathcal{TF}(\boldsymbol{\Gamma}, \overline{g}, r_\mathrm{p})) \neq \perp$, conditioned on $(\boldsymbol{\Gamma}, \overline{g}, r_\mathrm{p}) \in I_{\mathrm{good}}$, is at least $1/6$. Thus, with probability $\frac{1}{12{K_1}^c}$ the tuple $(\mathbf{b}, \eta_0, \eta_1, \eta_2)$ satisfies Conclusion 2a, i.e. it satisfies

$$\mathbf{b}^{\eta_0} = \mathbf{h}^{\eta_1} \mathbf{g}^{\eta_2} = \mathbf{h}^{\eta_1 + e\eta_2} \ , \tag{58}$$

and η_0 does not divide both of η_1 and η_2.

We argue that for any fixed $(\mathbf{g}, \eta_0, \eta_1, \eta_2)$ the probability that $\eta_0 \nmid (\eta_1 + e\eta_2)$ is at least $1/4$ over the random choice of e, conditioned on $\mathbf{g} = \mathbf{h}^e$.

To start with we note that if $\eta_0 \mid (\eta_1 + e\eta_2)$ and $\eta_0 \mid \eta_2$ then clearly $\eta_0 \mid \eta_1$ as well, which is a contradiction. Thus, if $\eta_0 \mid \eta_2$, the probability that $\eta_0 \nmid (\eta_1 + e\eta_2)$ is one.

Consider now the case where $\eta_0 \nmid \eta_2$. This implies that there exists a prime power p^i such that $p^i \mid \eta_0$ but $p^i \nmid \eta_2$. It follows that

$$\Pr_e[\eta_0 \mid (\eta_1 + e\eta_2) \mid \mathbf{g} = \mathbf{h}^e] \leq \Pr[\eta_1 + e\eta_2 = 0 \mod p^i \mid \mathbf{g} = \mathbf{h}^e] \ .$$

We define $t = (\mathbf{p} - 1)(\mathbf{q} - 1)/4$ and write $e = e't + (e \bmod t)$. By definition $\gcd(t, p^i) = 1$, so t is a generator in $\mathbb{Z}_{p^i}$. This implies that

$$\begin{aligned} &\Pr_e[\eta_1 + e\eta_2 = 0 \mod p^i \mid \mathbf{g} = \mathbf{h}^e] \\ &= \Pr_e[\eta_1 + (e \bmod t)\eta_2 + e't = 0 \mod p^i \mid \mathbf{g} = \mathbf{h}^e] \ . \end{aligned}$$

Note that $\eta_1 + (e \bmod t)\eta_2$ is constant for any fixed $(\mathbf{g}, \eta_0, \eta_1, \eta_2)$, but since t is a generator in $\mathbb{Z}_{p^i}$ the probability that $e't$ takes on any given value is at most $\lfloor p^i/K_5 \rfloor + 1/K_5 \leq 3/4$.

We have established that $(\mathbf{b}, \eta_0, \eta_1, \eta_2)$ satisfies Equation (58) and $\eta_0 \nmid (\eta_1 + e\eta_2)$ with probability at least $\frac{1}{52{K_1}^c}$. We now argue that if this event occurs, the output of $\mathcal{RSAR}$ is a non-trivial root of $\mathbf{h}$. Suppose that $\eta_0 \nmid (\eta_1 + e\eta_2)$. Then

$$\mathbf{h}^f = \mathbf{h}^{a\eta_0 + b(\eta_1 + e\eta_2)} = (\mathbf{h}^a \mathbf{b}^b)^{\eta_0} \enspace .$$

The value of f satisfies $f \neq 0$, $f \mid \eta_0$ and $f \neq \pm\eta_0$, so $\eta = \eta_0/f \neq \pm 1$, and the output requirement on the adversary in the strong RSA-assumption is satisfied with probability $\frac{1}{52{K_1}^c}$, which is a contradiction.

Claim 4. If Equation (56) holds, then the strong RSA-assumption is false.

Proof. We describe an adversary that contradicts the strong RSA-assumption, but this adversary is quite different from the adversary above.

Algorithm 4 (Second Root Finder).

$\mathcal{RSAR}'(\mathbf{N}, \mathbf{h})$
$r_{\mathrm{p}} \leftarrow \{0,1\}^*$
$e \leftarrow [0, 2^{K_2 + K_5} - 1]$
$\mathbf{g} \leftarrow \mathbf{h}^e$
$g_1, \ldots, g_N \leftarrow G_q$
$d \leftarrow \{0,1\}$, $\quad \mathbf{g}' \leftarrow \mathbf{g}^d \mathbf{h}^{1-d}$, , $\quad \mathbf{h}' \leftarrow \mathbf{g}^{1-d} \mathbf{h}^d$
$(T_1, \ldots, T_{2N}) \leftarrow \mathcal{TF}((\mathbf{N}, \mathbf{g}', \mathbf{h}'), (g, g_1, \ldots, g_N), r_{\mathrm{p}})$
$(\eta_1, \eta_2) \leftarrow \mathcal{C}_{\mathrm{RSA}'}(T_1, \ldots, T_{2N})$
`If` $(\eta_1 = \pm\eta_2)$ `Then`
  $(\mathbf{b}, \eta) \leftarrow (\mathbf{h}, \eta_1 + 1)$
`Else`
  $(f, a, b) \leftarrow \mathrm{extgcd}(\eta_1, \eta_2)$
  $(\mathbf{b}, \eta) \leftarrow ((\mathbf{g}')^{-a} (\mathbf{h}')^b, (d\eta_2 + (1-d)\eta_1)/f)$
`EndIf`
`Return` $(\mathbf{b}, \eta)$

That the algorithm is polynomial time can be seen as in the proof of the previous claim. We must argue that its success probability is notable.

Assume that $d = 1$, i.e. all probabilities are conditioned on this event. The probability that the input $(\boldsymbol{\Gamma}, \overline{g}, r_{\mathrm{p}}) = ((\mathbf{N}, \mathbf{g}, \mathbf{h}), (g, g_1, \ldots, g_N), r_{\mathrm{p}})$ belongs to I_{good} is at least $\frac{1}{2{K_1}^c}$ by Claim 1.

By assumption the probability that $\mathcal{C}_{\mathrm{RSA}'}(\mathcal{TF}(\boldsymbol{\Gamma}, \overline{g}, r_{\mathrm{p}})) \neq \perp$, conditioned on $(\boldsymbol{\Gamma}, \overline{g}, r_{\mathrm{p}}) \in I_{\mathrm{good}}$, is at least $1/6$. Thus, with probability $\frac{1}{12{K_1}^c}$ the pair (η_1, η_2) satisfies Conclusion 3, i.e. it satisfies

$$\mathbf{h}^{\eta_1} = \mathbf{g}^{\eta_2} \enspace , \tag{59}$$

and $(\eta_1, \eta_2) \neq (0, 0)$.

If $\eta_1 = \pm\eta_2$ then $(\mathbf{h}'/\mathbf{g}')^{\eta_1} = 1$ or $(\mathbf{h}'\mathbf{g}')^{\eta_1} = 1$. There are two possible explanations for this. Either $\mathbf{g} = \mathbf{h}$ or $\mathbf{h}^{-1}$, which happens with negligible probability. The other explanation is that $t \mid \eta_1$, with $t = (\mathbf{p}-1)(\mathbf{q}-1)/4$, which implies that $\mathbf{h}^{\eta_1+1} = \mathbf{h}$ and the output is a non-trivial root.

If $\eta_1 \neq \pm\eta_2$ there are two cases. If $\eta_2 \nmid \eta_1$ we have

$$\mathbf{h}^f = \mathbf{h}^{a\eta_1+b\eta_2} = (\mathbf{g}^a\mathbf{h}^b)^{\eta_2} \enspace .$$

The value of f satisfies $f \neq 0$, and $\eta = \eta_2/f \neq \pm 1$ so the output is a non-trivial root.

If $\eta_2 \mid \eta_1$ we have a problem, since then $f = \eta_2$ and the extracted root is trivial. Note that if the roles of $\mathbf{g}$ and $\mathbf{h}$ are reversed in Equation (59) the problem disappears.

The idea of the algorithm is that the adversary cannot tell if $(\mathbf{g}', \mathbf{h}')$ was defined as $(\mathbf{g}, \mathbf{h})$ or as $(\mathbf{h}, \mathbf{g})$, i.e. if it outputs (η_1, η_2) such that $\mathbf{h}^{\eta_1} = \mathbf{g}^{\eta_2}$ or such that $\mathbf{g}^{\eta_1} = \mathbf{h}^{\eta_2}$. In fact the distribution of (η_1, η_2) conditioned on $d = 0$ is statistically close to its distribution conditioned on $d = 1$, since the distribution of $\mathbf{g}$ is statistically close to the distribution of $\mathbf{h}$. Thus, the probability that $(d\eta_2 + (1-d)\eta_1)/f = \pm 1$, conditioned on $\eta_1 \neq \pm\eta_2$ is less than $3/4$.

We conclude that the algorithm outputs a non-trivial root with probability at least $\frac{1}{52{K_1}^c}$, which again contradicts the strong RSA-assumption.

Claim 5. If Equation (57) holds, then the DL-assumption is false.

Proof. We construct an adversary that outputs a non-trivial representation of $1 \in G_q$ with notable probability.

Algorithm 5 (Representation Finder).

```
REP(ḡ)
r_p ← {0,1}*
Γ ← F_RSA
(T_1,...,T_2N) ← TF(Γ, ḡ, r_p)
s ← C_REP(T_1,...,T_2N)
If (s is on the form (η_j)_{j=0}^N) Then
    Return s
Else
    (ρ', (ρ'_ij)) ← s
EndIf
(T_{2N+1}, T_{2N+2}) ← FF(Γ, ḡ, r_p)
η_0 ← Σ_{i=1}^N ρ'_i p_{2N+1,i} − γ'_{2N+1}
(η_j)_{j=1}^N ← (Σ_{i=1}^N ρ'_{i,j} p_{2N+1,i} − ρ_{2N+1,j})_{j=1}^N
Return (η_j)_{j=0}^N
```

Here $\rho_{2N+1,j}$ denotes the integer extracted from T_{2N+1} and T_{2N+2} corresponding to ρ_j in Equation (23), that may or may not be equal to a prime $p_{2N+1,\pi_{2N+1}(j)}$ appearing in the prime vector used in both the transcripts T_{2N+1} and T_{2N+2}.

The probability that the input $(\boldsymbol{\Gamma}, \overline{g}, r_\mathrm{p}) = ((\mathbf{N}, \mathbf{g}, \mathbf{h}), (g, g_1, \ldots, g_N), r_\mathrm{p})$ belongs to I_good is at least $\frac{1}{2{K_1}^c}$ by Claim 1. By assumption the probability that $\mathcal{C}_\mathrm{REP}(\boldsymbol{\Gamma}, \overline{g}, r_\mathrm{p}) \neq \bot$, conditioned on $(\boldsymbol{\Gamma}, \overline{g}, r_\mathrm{p}) \in I_\mathrm{good}$ is at least $\frac{1}{6}$. Thus, with probability $\frac{1}{18{K_1}^c}$ the output s satisfies Conclusion 4, 5, or 6 of Lemma 3, i.e. either (1) the output of $\mathcal{C}_\mathrm{REP}$ is a non-trivial representation $(\eta_j)_{j=0}^N$, (2) it consists of integers $(\rho', (\rho'_{ij}))$ such that $w_i = g^{\rho'_i} \prod_{j=1}^N g_j^{\rho'_{i,j}}$ and (ρ'_{ij}) is not a permutation matrix, or (3) (ρ'_{ij}) is any matrix, but one of the extracted ρ_{ji} exponents equals 1 or -1.

The first type of output of $\mathcal{C}_\mathrm{REP}$ obviously implies that that $\mathcal{REP}$ outputs a non-trivial representation of $1 \in G_q$ with non-negligible probability.

Consider the third type of output[4] from $\mathcal{C}_\mathrm{REP}$. This type of output is special in that it is extracted from a pair of transcripts T_j and T_{j+N} (a fork) and all other transcripts are ignored. This implies that such a pair of transcripts are encountered by the single call to the fork-finding routine $\mathcal{FF}$ with probability $\frac{1}{18{K_1}^c N} - \epsilon$ for some negligible ϵ. The error term comes from the fact that the transcript finder requires each pair of transcript it gets from $\mathcal{FF}$ to satisfy some additional assumptions, e.g., the prime vectors should be linearly independent, but this happens with overwhelming probability.

Equation (25) in the proof of Lemma 3 implies that

$$\begin{aligned} g^{\gamma'_{2N+1}} \prod_{i=1}^N g_i^{\rho_{2N+1,i}} &= \prod_{i=1}^N w_i^{p_{2N+1,i}} = \prod_{i=1}^N \left(g^{\rho'_i} \prod_{j=1}^N g_j^{\rho'_{i,j}} \right)^{p_{2N+1,i}} \\ &= g^{\sum_{i=1}^N \rho'_i p_{2N+1,i}} \prod_{j=1}^N g_j^{\sum_{i=1}^N \rho'_{i,j} p_{2N+1,i}} \quad . \end{aligned}$$

It is obvious that for fixed $\rho'_{i,j}$ the probability that $\sum_{i=1}^N \rho'_{i,j} p_{2N+1,i} = \pm 1$ is negligible for each $j = 1, \ldots, N$. From the proof of Lemma 6 and Markov's inequality we know that $\mathcal{FF}$ executes its outer loop more than 16 times with probability at most $1/2$. Thus, $\mathcal{REP}$ outputs a non-trivial representation of 1 in G_q with probability at least $\frac{1}{37 K_1 N}$.

Consider now the second type of output. From the previous case we may now assume that for every pair (T_j, T_{j+N}) there exists a permutation π_j such that the extracted integer $\rho_{j,i} = p_{j,\pi_j(i)}$ except with negligible probability. Equation

[4] This type of output was not considered in the original version, but it is necessary due to an error in the original proof. More precisely, it corresponds to the case described in Point 1a on Page 35, where some ρ_i contains more than one prime factor. The flaw is discussed in Appendix K.

(27) in the proof of Lemma 3 implies that

$$g^{\gamma'_{2N+1}} \prod_{i=1}^{N} g_i^{p_{2N+1,\pi_{2N+1}(i)}} = \prod_{i=1}^{N} w_i^{p_{2N+1,i}} = \prod_{i=1}^{N} \left(g^{\rho'_i} \prod_{j=1}^{N} g_j^{\rho'_{i,j}} \right)^{p_{2N+1,i}}$$
$$= g^{\sum_{i=1}^{N} \rho'_i p_{2N+1,i}} \prod_{j=1}^{N} g_j^{\sum_{i=1}^{N} \rho'_{i,j} p_{2N+1,i}} \quad .$$

Again, we use that $\mathcal{FF}$ executes its outer loop more than 16 times with probability at most 1/2. Thus, Corollary 4 and Corollary 1 imply that the probability that $\sum_{i=1}^{N} \rho'_{i,j} p_{2N+1,i} = p_{2N+1,\pi_{2N+1}(j)}$ for $j = 1, \ldots, N$ is negligible.

We conclude that the probability that $\mathcal{REP}$ outputs a non-trivial representation of $1 \in G_q$ is at least $\frac{1}{37K_1N}$. Lemma 10 then implies that the DL-assumption is false.

E Security Analysis of Protocol 3

Proof (Theorem 3 on Page 21). We describe an ideal adversary $\mathcal{S}(\cdot)$ that runs any hybrid adversary $\mathcal{A}$ as a black-box. Then we show that if $\mathcal{S}$ does not imply that the protocol is secure, then we can break one of the assumptions.

THE IDEAL ADVERSARY $\mathcal{S}$. Let I_M be the set of mix-servers corrupted by $\mathcal{A}$. The ideal adversary $\mathcal{S}$ corrupts the dummy participants $\tilde{M}_i$ for which $i \in I_M$. The ideal adversary is best described by starting with a copy of the original hybrid ITM-graph

$$(V, E) = \mathcal{Z}'(\mathcal{H}(\mathcal{A}, \pi_{\mathrm{DP}}^{(\tilde{\pi}_1^{\mathcal{F}_{\mathrm{BB}}}, \tilde{\pi}_2^{\mathcal{F}_{\mathrm{ZK}}^{R_{\mathrm{C}}(z)}}, \tilde{\pi}_3^{\mathcal{F}_{\mathrm{ZK}}^{R_{\mathrm{C}}(y)}}, \tilde{\pi}_4^{\mathcal{F}_{\mathrm{CF}}}, \tilde{\pi}_5^{\mathcal{F}_{\mathrm{GG}}})}) \quad ,$$

where $\mathcal{Z}$ is replaced by a machine $\mathcal{Z}'$. The adversary $\mathcal{S}$ simulates all ideal functionalities honestly except $\mathcal{F}_{\mathrm{CF}}$, $\mathcal{F}_{\mathrm{ZK}}^{R_{\mathrm{C}}(z)}$, and $\mathcal{F}_{\mathrm{ZK}}^{R_{\mathrm{C}}(y)}$. The simulation of these functionalities and M_j for $j \notin I_M$ is described below.

Simulation of Links $(\mathcal{Z}, \mathcal{A})$, and $(\mathcal{Z}, M_j)$ for $j \in I_M$. $\mathcal{S}$ simulates $\mathcal{Z}'$, and $\tilde{M}_j$ for $j \in I_M$, such that it appears as if $\mathcal{Z}$ and $\mathcal{A}$, and $\mathcal{Z}$ and M_j for $j \in I_M$ are linked directly.

1. When $\mathcal{Z}'$ receives m from $\mathcal{A}$, m is written to $\mathcal{Z}$, by $\mathcal{S}$. When $\mathcal{S}$ receives m from $\mathcal{Z}$, m is written to $\mathcal{A}$ by $\mathcal{Z}'$. This is equivalent to that $\mathcal{Z}$ and $\mathcal{A}$ are linked directly.
2. When $\mathcal{Z}'$ receives m from M_j for $j \in I_M$, m is written to $\mathcal{Z}$ by $\tilde{M}_j$. When $\tilde{M}_j$, $j \in I_M$, receives m from $\mathcal{Z}$, m is written to M_j by $\mathcal{Z}'$. This is equivalent to that $\mathcal{Z}$ and M_j are linked directly for $j \in I_M$.

Extraction from Corrupt Provers. When a corrupt prover M_j, for $j \in I_M$, manages to convince the honest verifiers that it knows a witness, that witness must be forwarded to $\mathcal{F}_{\text{ZK}}^{R_{\text{DP}}}$. To do that the witness must be extracted, but this is easy.

The ideal adversary waits until $(M_j, \mathtt{R}, R)$ appears on $\mathcal{F}_{\text{BB}}$. Then it interrupts the simulation of $\mathcal{F}_{\text{BB}}$ and checks if the honest verifiers would accept R as the final message of the prover in Protocol 2. If so, it looks at the internal tapes of $\mathcal{F}_{\text{ZK}}^{R_{\text{C}}(z)}$ and $\mathcal{F}_{\text{ZK}}^{R_{\text{C}}(y)}$ to see if w and x have been stored under the tags (M_j, z) and (M_j, y) respectively. If this is the case it instructs $\tilde{M}_j$ to hand $(\mathtt{Prover}, (g, z, y, (u_i, v_i)_{i=1}^N, (u_i', v_i')_{i=1}^N), (w, x))$ to $\mathcal{F}_{\text{ZK}}^{R_{\text{DP}}}$. Then it waits until it receives $(M_j, \mathtt{Prover}, (g, z, y, (u_i, v_i)_{i=1}^N, (u_i', v_i')_{i=1}^N), b)$ from $\mathcal{F}_{\text{ZK}}^{R_{\text{DP}}}$, at which point it resumes the simulation of $\mathcal{F}_{\text{BB}}$. Note that there is no automatic guarantee that $b = 1$. In fact we essentially prove below that $b = 1$ with overwhelming probability.

Simulation of Honest Provers. When an honest dummy prover $\tilde{M}_j$ hands a witness to $\mathcal{F}_{\text{ZK}}^{R_{\text{DP}}}$, the ideal adversary must simulate a proof that convinces the corrupted verifiers. This must be done without knowledge of the witness.

When $\mathcal{S}$ receives $(M_j, \mathtt{Prover}, (g, z, y, (u_i, v_i)_{i=1}^N, (u_i', v_i')_{i=1}^N), 1)$ from $\mathcal{F}_{\text{ZK}}^{R_{\text{DP}}}$, it inputs $(\mathtt{Prover}, (g, z, y, (u_i, v_i)_{i=1}^N, (u_i', v_i')_{i=1}^N), (1, 1))$ to M_j, and instructs it to ignore the fact that the witnesses are invalid. It also instructs $\mathcal{F}_{\text{ZK}}^{R_{\text{C}}(z)}$ and $\mathcal{F}_{\text{ZK}}^{R_{\text{C}}(y)}$ respectively to behave as if the submitted witnesses from M_j are correct, despite that it is handed $(z, 1)$ and $(y, 1)$ respectively.

Then $c \in [2^{K_4 - 1}, 2^{K_4} - 1]$ is chosen randomly and the simulator described in the proof of Proposition 1, i.e. the simulator of the honest verifier zero-knowledge public coin proof is invoked, to generate C. Then M_j is instructed to use this C, instead of computing it, and $\mathcal{F}_{\text{CF}}$ is instructed to output c. This implies that all verifiers accept the proof.

Reaching a Contradiction. Next we show that if the ideal adversary $\mathcal{S}$ defined above does not imply the security of Protocol 3, then we can break the DL-assumption in G_q or the strong RSA-assumption.

Suppose that $\mathcal{S}$ does not imply the security of the protocol. Then there exists a hybrid adversary $\mathcal{A}$, an environment $\mathcal{Z}$ with auxiliary input $z = \{z_n\}$, a constant $c > 0$ and an infinite index set $\mathcal{N} \subset \mathbb{N}$ such that for $K_1 \in \mathcal{N}$

$$
\begin{aligned}
&| \Pr[\mathcal{Z}_z(\mathcal{I}(\mathcal{S}, \tilde{\pi}^{\mathcal{F}_{\text{ZK}}^{R_{\text{DP}}}})) = 1] \\
&\quad - \Pr[\mathcal{Z}_z(\mathcal{H}(\mathcal{A}, \pi_{\text{DP}}^{(\tilde{\pi}_1^{\mathcal{F}_{\text{BB}}}, \tilde{\pi}_2^{\mathcal{F}_{\text{ZK}}^{R_{\text{C}}(z)}}, \tilde{\pi}_3^{\mathcal{F}_{\text{ZK}}^{R_{\text{C}}(y)}}, \tilde{\pi}_4^{\mathcal{F}_{\text{CF}}}, \tilde{\pi}_5^{\mathcal{F}_{\text{GG}}})})) = 1]| \geq \frac{1}{{K_1}^c} \enspace ,
\end{aligned}
$$

where $\mathcal{S}$ runs $\mathcal{A}$ as a black-box as described above, i.e. $\mathcal{S} = \mathcal{S}(\mathcal{A})$.

Denote by T the machine that simulates $\mathcal{Z}_z(\mathcal{I}(\mathcal{S}(\mathcal{A}'), \tilde{\pi}^{\mathcal{F}_{\text{MN}}}))$, except that instead of simulating honest provers M_j for $j \notin I_M$ as described above, it simply looks at the internal tapes of $\mathcal{F}_{\text{ZK}}^{R_{\text{DP}}}$ to extract the message on the form $(M_j, \mathtt{Prover}, (g, z, y, (u_i, v_i)_{i=1}^N, (u_i', v_i')_{i=1}^N), (w, x))$ handed to $\mathcal{F}_{\text{ZK}}^{R_{\text{DP}}}$ by M_j, and then inputs $(\mathtt{Prover}, (g, z, y, (u_i, v_i)_{i=1}^N, (u_i', v_i')_{i=1}^N), (w, x))$ to M_j, which then follows the protocol honestly.

From Proposition 1 we know that Protocol 2 is statistical zero-knowledge. In fact the proof of this proposition implies that

$$|\Pr[T = 1] - \Pr[\mathcal{Z}_z(\mathcal{I}(\mathcal{S}(\mathcal{A}'), \tilde{\pi}^{\mathcal{F}_{\mathrm{MN}}})) = 1]| < O(2^{-K_5}) \enspace .$$

For simplicity we ignore this negligible difference in the remainder of the proof.

We know from Lemma 2 that the prime vectors of the honest parties are identical with overwhelming probability. Thus, we ignore also this difference in the remainder of the proof.

The only remaining difference between the simulation carried out by T and an execution in the real model, is that it could happen that $\mathcal{S}$ hands $(\texttt{Prover}, (g, z, y, (u_i, v_i)_{i=1}^N, (u_i', v_i')_{i=1}^N), (w, x))$ to $\mathcal{F}_{\mathrm{ZK}}^{R_{\mathrm{DP}}}$, despite that $((g, z, y, (u_i, v_i)_{i=1}^N, (u_i', v_i')_{i=1}^N), (w, x)) \notin R_{\mathrm{DP}}$.

Denote by B_l the event that this happens the lth time a proof is carried out. Since the adversary $\mathcal{A}$, the environment $\mathcal{Z}$, and all honest parties and functionalities run in polynomial time for some polynomial $p(K_1)$, there are at most $p(K_1)$ potential such l.

Since the simulation is statistically close to the real model in all respects except that some event B_l may occur, and we know that the environment can distinguish the hybrid model from the ideal model with notable probability, we conclude that

$$\Pr\left[B_1 \vee B_2 \vee \cdots \vee B_{p(K_1)}\right] \geq \frac{1}{{K_1}^c} \enspace .$$

An averaging argument implies that $\Pr[B_l] \geq \frac{1}{p(K_1){K_1}^c}$ for some fixed l.

Next we argue that this contradicts Proposition 2, i.e., the soundness of the proof of a shuffle given in Protocol 2. Denote by A' the adversary that accepts $(\boldsymbol{\Gamma}, \overline{g})$ as input and simulates simulates T, except for the following changes. It uses its input, instead of generating these parameters during the simulation of $\mathcal{F}_{\mathrm{RSA}}$ and $\mathcal{F}_{\mathrm{CF}}$. The simulation is continued until the lth proof is about to be executed. Then it waits for an input $s \in [0, 2^{K_1} - 1]$ and instructs $\mathcal{F}_{\mathrm{CF}}$ to output s in the generation of the primes, i.e., $P_j = \mathsf{PGen}(\mathsf{PRG}(s))$. Then it waits for another input $c \in [2^{K_4 - 1}, 2^{K_4} - 1]$ and instructs $\mathcal{F}_{\mathrm{CF}}$ to output $c' = c - 2^{K_4 - 1}$ in the generation of the challenge. It follows that

$$\Pr_{\boldsymbol{\Gamma}, \overline{g}, r_{\mathrm{p}}, r_{\mathrm{v}}}[\mathrm{Acc}(T_{A'}(\boldsymbol{\Gamma}, \overline{g}, r_{\mathrm{p}}, r_{\mathrm{v}})) = 1 \wedge I_{A'}(\boldsymbol{\Gamma}, \overline{g}, r_{\mathrm{p}}) \notin L_{R_{\mathrm{DP}}}] \geq \frac{1}{p(K_1){K_1}^c} \enspace ,$$

where r_{p} denotes the randomness of all machines in the simulation of T, and r_{v} denotes the randomness of the honest verifier. This contradicts Proposition 2.

F Two Other Shuffles

As explained in Section 4 we consider a new shuffle-relation in the main part of the paper. However, without too much work our approach gives proofs of shuffles for the two previously considered relations, i.e., the re-encryption shuffle and decryption-re-encryption shuffle. For each type of proof of a shuffle we describe briefly the structure of the mix-net in which it can be employed. Then we give a detailed description of the protocol, and sketch the minor changes needed in the security analysis of the proof of a shuffle. We stress that we do not prove the overall security of these mix-nets.

F.1 The Re-encryption Shuffle

The first El Gamal based mix-nets exploits that a cryptotext can be randomly re-encrypted. A secret key $x \in \mathbb{Z}_q$ is shared somehow between the mix-servers. The corresponding public key $y = g^x$ is known by all parties. To encrypt its message m_i a sender forms $(u_i, v_i) = E_{(g,y)}(m_i, r_i) = (g^{r_i}, y^{r_i} m_i)$ and also proves knowledge of r_i to avoid the relation attacks. The mix-servers form a list $(u_{0,i}, v_{0,i})_{i=1}^N$ from the cryptotexts with valid proofs of knowledge. Then for $j = 1, \ldots, k$ mix-server M_j computes $L_j = (u_{j,i}, v_{j,i})_{i=1}^N = (g^{x_{j,\pi_j(i)}} u_{j-1,\pi_j(i)}, y^{x_{j,\pi_j(i)}} v_{j-1,\pi_j(i)})$, for a randomly chosen permutation $\pi_j \in \Sigma_N$ and random exponents $x_{j,i} \in \mathbb{Z}_q$, and then proves knowledge of $(x_{j,\pi_j(i)})_{i=1}^N$ that satisfy this relation. Finally, the mix-servers jointly and verifiably decrypt the output $(u_{k,i}, v_{k,i})_{i=1}^N$ of the last mix-server. The last step corresponds to a joint decryption in a threshold version of El Gamal.

The relation that the input L_{j-1} and output L_j of a mix-server M_j must satisfy is formalized as follows.

Definition 5 (Knowledge of Correct Re-encryption-Permutation). *Define for each N a relation $R_{\mathrm{RP}} \subset (G_q^2 \times G_q^{2N} \times G_q^{2N}) \times \mathbb{Z}_q^N$, by*

$$((g, y, \{(u_i, v_i)\}_{i=1}^N, \{(u_i', v_i')\}_{i=1}^N), (x_i)_{i=1}^N) \in R_{\mathrm{RP}}$$

precisely when $(u_i', v_i') = (g^{x_{\pi(i)}} u_{\pi(i)}, y^{x_{\pi(i)}} v_{\pi(i)})$ for $i = 1, \ldots, N$ and some permutation $\pi \in \Sigma_N$.

We are now ready to transform Protocol 2 into a protocol for proving the above relation instead.

Protocol 4 (Proof of Re-encryption-Permutation). The common input consists of an RSA modulus $\mathbf{N}$ and $\mathbf{g}, \mathbf{h} \in \mathrm{QR}_{\mathbf{N}}$, generators $g, g_1, \ldots, g_N \in G_q$, a public key y, and two lists $L = (u_i, v_i)_{i=1}^N$ and $L' = (u_i', v_i')_{i=1}^N$ such that $(u_i', v_i') = (g^{x_{\pi(i)}} u_{\pi(i)}, y^{x_{\pi(i)}} v_{\pi(i)})$ for some permutation π of N elements. The private input to the prover consists of $(x_i)_{i=1}^N$.

1. The prover chooses $r_i' \in \mathbb{Z}_q$ randomly, computes $(w_i)_{i=1}^N = (g^{r_i'} g_{\pi^{-1}(i)})_{i=1}^N$, and hands $(w_i)_{i=1}^N$ to the verifier.
2. The verifier chooses random primes $p_1, \ldots, p_N \in [2^{K_3-1}, 2^{K_3} - 1]$, and hands $(p_i)_{i=1}^N$ to the prover. Then the prover defines $x_s = \sum_{i=1}^N x_i p_i$.
3. Both parties compute $(U, V, W) = (\prod_{i=1}^N u_i^{p_i}, \prod_{i=1}^N v_i^{p_i}, \prod_{i=1}^N w_i^{p_i})$.

4. The prover chooses $k_1, k_2 \in \mathbb{Z}_q$, $l_1, l_2, l_s, l_{r'} \in \mathbb{Z}_q$ randomly. Then it chooses $t_i, t'_i \in [0, 2^{K_2+K_5}-1]$, $s_i, s'_i \in [0, 2^{K_2+K_4+2K_5}-1]$, and $r_i \in [0, 2^{K_3+K_4+K_5}-1]$ for $i = 1, \ldots, N$ randomly. Then it chooses an element $s \in [0, 2^{K_2+NK_3+K_4+K_5+\log_2 N}-1]$, and $s' \in [0, 2^{K_2+K_5+\log_2 N}-1]$ randomly and computes

$$(b_1, b_2) = (g_1^{k_1} g^{x_s}, g_1^{k_2} y^{x_s}) \tag{60}$$

$$(\beta_1, \beta_2) = \left(g_1^{l_1} g^{l_s}, g_1^{l_2} y^{l_s}\right) \tag{61}$$

$$(\alpha_1, \alpha_2, \alpha_3) = \left(g_1^{l_1} \prod_{i=1}^{N} (u'_i)^{r_i}, g_1^{l_2} \prod_{i=1}^{N} (v'_i)^{r_i}, g^{l_{r'}} \prod_{i=1}^{N} g_i^{r_i}\right) \tag{62}$$

$$\mathbf{b}_0 = \mathbf{g} \tag{63}$$

$$(\mathbf{b}_i, \mathbf{b}'_i)_{i=1}^N = (\mathbf{h}^{t_i} \mathbf{b}_{i-1}^{p_{\pi(i)}}, \mathbf{h}^{t'_i} \mathbf{g}^{p_{\pi(i)}})_{i=1}^N \tag{64}$$

$$(\boldsymbol{\gamma}_i, \boldsymbol{\gamma}'_i)_{i=1}^N = (\mathbf{h}^{s_i} \mathbf{b}_{i-1}^{r_i}, \mathbf{h}^{s'_i} \mathbf{g}^{r_i})_{i=1}^N \tag{65}$$

$$(\boldsymbol{\gamma}, \boldsymbol{\gamma}') = (\mathbf{h}^{s}, \mathbf{h}^{s'}) \ , \tag{66}$$

and $((b_1, b_2), (\beta_1, \beta_2), (\alpha_1, \alpha_2, \alpha_3), (\mathbf{b}_i, \mathbf{b}'_i)_{i=1}^N, (\boldsymbol{\gamma}_i, \boldsymbol{\gamma}'_i)_{i=1}^N, (\boldsymbol{\gamma}, \boldsymbol{\gamma}'))$ is handed to the verifier.
5. The verifier chooses $c \in [2^{K_4-1}, 2^{K_4}-1]$ randomly and hands c to the prover.
6. The prover computes

$$\begin{aligned}
f_1 &= ck_1 + l_1 && \bmod q \\
f_2 &= ck_2 + l_2 && \bmod q \\
f_s &= cx_s + l_s && \bmod q \\
f_{r'} &= cr' + l_{r'} && \bmod q \\
(e_i, e'_i)_{i=1}^N &= (ct_i + s_i, ct'_i + s'_i)_{i=1}^N && \bmod 2^{K_2+K_4+2K_5} \\
(d_i)_{i=1}^N &= (cp_{\pi(i)} + r_i)_{i=1}^N && \bmod 2^{K_3+K_4+K_5} \\
e &= ct + s && \bmod 2^{K_2+NK_3+K_4+K_5+\log_2 N} \\
e' &= ct' + s' && \bmod 2^{K_2+K_5+\log_2 N}
\end{aligned}$$

Then it hands $((f_1, f_2, f_s, f_{r'}), (e_i, e'_i)_{i=1}^N, (d_i)_{i=1}^N, (e, e'))$ to the verifier.
7. The verifier checks that L' is lexicographically sorted and that

$$(b_1^c \beta_1, b_2^c \beta_2) = (g_1^{f_1} g^{f_s}, g_1^{f_2} y^{f_s}) \tag{67}$$

$$((b_1 U)^c \alpha_1, (b_2 V)^c \alpha_2, W^c \alpha_3) = \left(g_1^{f_1} \prod_{i=1}^{N} (u'_i)^{d_i}, g_1^{f_2} \prod_{i=1}^{N} (v'_i)^{d_i}, g^{f_{r'}} \prod_{i=1}^{N} g_i^{d_i}\right) \tag{68}$$

$$(\mathbf{b}_i^c \boldsymbol{\gamma}_i, (\mathbf{b}'_i)^c \boldsymbol{\gamma}'_i)_{i=1}^N = (\mathbf{h}^{e_i} \mathbf{b}_{i-1}^{d_i}, \mathbf{h}^{e'_i} \mathbf{g}^{d_i})_{i=1}^N \tag{69}$$

$$(\mathbf{g}^{-\prod_{i=1}^N p_i} \mathbf{b}_N)^c \boldsymbol{\gamma} = \mathbf{h}^{e} \tag{70}$$

$$\left(\mathbf{g}^{-\sum_{i=1}^N p_i} \prod_{i=1}^{N} \mathbf{b}'_i\right)^c \boldsymbol{\gamma}' = \mathbf{h}^{e'} \ . \tag{71}$$

Security Analysis. The security analysis of the protocol is almost identical to the analysis of Protocol 2. For completeness we give a careful description of all modifications that are necessary.

Proposition 3. *Protocol 4 is honest verifier statistical zero-knowledge.*

Proof (Proposition 3). The simulator chooses $p_1, \ldots, p_N$ and c honestly. Then it chooses $f_1, f_1, f_x, f_{r'} \in \mathbb{Z}_q$, $e \in [0, 2^{K_2+NK_3+K_4+K_5+\log_2 N} - 1]$, $e' \in [0, 2^{K_2+K_5+\log_2 N} - 1]$, $(e_i, e'_i)_{i=1}^N \in [0, 2^{K_2+K_4+2K_5} - 1]^N$, and $(d_i)_{i=1}^N \in [0, 2^{K_3+K_4+K_5} - 1]^N$ randomly. It also chooses $b_1, b_2 \in G_q$ and $\mathbf{b}_i, \mathbf{b}'_i \in \mathrm{QR}_{\mathbf{N}}$ randomly. Finally, the simulator defines, (β_1, β_2) by Equations (67), $(\alpha_1, \alpha_2, \alpha_3)$ by Equations (68), $\boldsymbol{\gamma}_i$ and $\boldsymbol{\gamma}'_i$ by Equation (69), $\boldsymbol{\gamma}$ by Equation (70), and $\boldsymbol{\gamma}'$ by Equation (71) respectively. The distribution of the resulting elements is statistically close to the distribution of the corresponding elements in the protocol.

We now discuss the soundness of the protocol. Compared with the analysis of Protocol 2 we must change the notation slightly. We denote the jth transcript in a list of transcripts by $T_j = (I_j, W_j, P_j, C_j, c_j, R_j)$ where I_j denotes the common input, W_j denotes the list of commitments w_i, and

$$
\begin{aligned}
P_j &= (p_{j,1}, \ldots, p_{j,N}) \\
C_j &= ((b_{j,1}, b_{j,2}), (\beta_{j,1}, \beta_{j,2}), (\alpha_{j,1}, \alpha_{j,2}, \alpha_{j,3}), (\mathbf{b}_{j,i}, \mathbf{b}'_{j,i})_{i=1}^N, (\boldsymbol{\gamma}_{j,i}, \boldsymbol{\gamma}'_{j,i})_{i=1}^N, (\boldsymbol{\gamma}_j, \boldsymbol{\gamma}'_j)) \\
R_j &= ((f_{j,1}, f_{j,2}, f_{j,s}, f_{j,r'}), (e_{j,i}, e'_{j,i})_{i=1}^N, (d_{j,i})_{i=1}^N, (e_j, e'_j)) \enspace .
\end{aligned}
$$

Lemma 7. *Lemma 3 holds also for Protocol 4, but with the notation above and with the main conclusion replaced by*

1. Main Conclusion*. *Elements $x_1, \ldots, x_N \in \mathbb{Z}_q$ and a permutation $\pi \in \Sigma_N$ s.t.*

$$((g, y, L, L'), (x_i)_{i=1}^N) \in R_{\mathrm{RP}} \enspace .$$

Proof. The proof is identical except for the last section of the proof. This must be replaced by the following.

The Common Input (g, y, L, L') Satisfies $((g, y, L, L'), (x_i)_{i=1}^N) \in R_{\mathrm{RP}}$. From the above we may assume that $\pi_j = \pi$ for all $j = 1, \ldots, N$, so we drop the subscript and simply write π from now on. Equations (67) imply that

$$
\begin{aligned}
b_1^{c_{j+N} - c_j} &= g_1^{f_{j+N,1} - f_{j,1}} g^{f_{j+N,s} - f_{j,s}} \enspace , \text{ and} \\
b_2^{c_{j+N} - c_j} &= g_1^{f_{j+N,2} - f_{j,2}} y^{f_{j+N,s} - f_{j,s}} \enspace .
\end{aligned}
$$

Define

$$
\begin{aligned}
\kappa_1 &= (f_{j+N,1} - f_{j,1})/(c_{j+N} - c_j) \enspace , && (72) \\
\kappa_2 &= (f_{j+N,2} - f_{j,2})/(c_{j+N} - c_j) \enspace , \text{ and} && (73) \\
\xi_{j,s} &= (f_{j+N,s} - f_{j,s})/(c_{j+N} - c_j) \enspace . && (74)
\end{aligned}
$$

Then we have

$$b_1 = g_1^{\kappa_1} g^{\xi_{j,s}} \quad , \ \text{and} \tag{75}$$
$$b_2 = g_1^{\kappa_2} y^{\xi_{j,s}} \quad . \tag{76}$$

From Equation (68) we have

$$(b_1 U)^{c_{j+N}-c_j} = g_1^{f_{j+N,1}-f_{j,1}} \prod_{i=1}^{N} (u_i')^{d_{j+N,i}-d_{j,i}} \quad , \ \text{and}$$
$$(b_2 V)^{c_{j+N}-c_j} = g_1^{f_{j+N,2}-f_{j,2}} \prod_{i=1}^{N} (v_i')^{d_{j+N,i}-d_{j,i}} \quad .$$

Our definitions of κ_1 and κ_2 in Equations (72) and (73), and the definition of ρ_i, which equals $p_{j,\pi(i)}$ in Equation (23) imply that

$$b_1 U = g_1^{\kappa_1} \prod_{i=1}^{N} (u_i')^{p_{j,\pi(i)}} \quad , \ \text{and}$$
$$b_2 V = g_1^{\kappa_2} \prod_{i=1}^{N} (v_i')^{p_{j,\pi(i)}} \quad .$$

If we combine Equations (75) and (76) with the two equations above we have

$$g^{\xi_{j,s}} \prod_{i=1}^{N} u_i^{p_{ji}} = g^{\xi_{j,s}} U = \prod_{i=1}^{N} (u_i')^{p_{j,\pi(i)}} \quad , \ \text{and} \tag{77}$$
$$y^{\xi_{j,s}} \prod_{i=1}^{N} v_i^{p_{ji}} = y^{\xi_{j,s}} V = \prod_{i=1}^{N} (v_i')^{p_{j,\pi(i)}} \quad , \tag{78}$$

for $j = 1, \ldots, N$. We apply the coefficients $a_{l,1}, \ldots, a_{l,N} \in \mathbb{Z}_q$ introduced above to the Equations (77) and (78) and conclude that

$$g^{\sum_{j=1}^{N} \xi_{j,s} a_{l,j}} u_l = \prod_{j=1}^{N} \left(g^{\xi_{j,s}} \prod_{i=1}^{N} u_i^{p_{ji}} \right)^{a_{l,j}} = \prod_{j=1}^{N} \left(\prod_{i=1}^{N} (u_i')^{p_{j,\pi(i)}} \right)^{a_{l,j}} = u'_{\pi^{-1}(l)} \quad , \ \text{and}$$
$$y^{\sum_{j=1}^{N} \xi_{j,s} a_{l,j}} v_l = \prod_{j=1}^{N} \left(y^{\xi_{j,s}} \prod_{i=1}^{N} v_i^{p_{ji}} \right)^{a_{l,j}} = \prod_{j=1}^{N} \left(\prod_{i=1}^{N} (v_i')^{p_{j,\pi(i)}} \right)^{a_{l,j}} = v'_{\pi^{-1}(l)} \quad .$$

This concludes the proof, since we can set $x_l = \sum_{j=1}^{N} \xi_{j,s} a_{l,j}$.

Consider the following definitional changes compared with the analysis of Protocol 2. Let A be the machine that given an RSA modulus $\mathbf{N}$ and $\mathbf{g}, \mathbf{h} \in \mathrm{QR}_{\mathbf{N}}$, and $g, g_1, \ldots, g_N \in G_q$ as input generates the remainder of the common input consisting of a public key $y \in G_q$, and two lists $L = (u_i, v_i)_{i=1}^{N} \in G_q^{2N}$ and

$L' = (u'_i, v'_i)_{i=1}^N \in G_q^{2N}$, and then plays the role of the prover in Protocol 4 on common input $((\mathbf{N}, \mathbf{g}, \mathbf{h}), (g, g_1, \ldots, g_N), y, (L, L'))$. We let Acc be the predicate that given a transcript T outputs 1 or 0 depending on if the verifier in the protocol above would accept the transcript or not.

Given these changes is straightforward to see, from the analysis in Sections D.2 and D.2, that the following proposition holds.

Proposition 4. *Assume the strong RSA-assumption and the DL-assumption. Then for all polynomial-size circuit families $A = \{A_K\}$ it holds that $\forall c > 0$, $\exists K_0$, such that for $K_1 \geq K_0$*

$$\Pr_{\boldsymbol{\Gamma}, \overline{g}, r_{\mathrm{p}}, r_{\mathrm{v}}}[\mathrm{Acc}(T_A(\boldsymbol{\Gamma}, \overline{g}, r_{\mathrm{p}}, r_{\mathrm{v}})) = 1 \wedge I_A(\boldsymbol{\Gamma}, \overline{g}, r_{\mathrm{p}}) \notin L_{R_{\mathrm{RP}}}] < \frac{1}{{K_1}^c} \enspace .$$

F.2 Decryption-Re-encryption Shuffle

More recent El Gamal based mix-nets, considered in [19,40,53], exploit re-encryption combined with partial decryption. In other words each mix-server both partially decrypts and re-encrypts the input cryptotexts before permuting them. Each mix-server holds a secret key x_j and a public key $y_j = g^{x_j}$ and a sequence of joint keys is defined by $Y_j = \prod_{l=j+1}^k y_j$. The secret key x_j of every mix-server is secretely shared among the other mix-servers. To encrypt its message m_i a sender forms $(u_i, v_i) = E_{(g, Y_1)}(m_i, r_i) = (g^{r_i}, Y_1^{r_i} m_i)$ and also proves knowledge of r_i to avoid the relation attacks. The mix-servers form a list $(u_{0,i}, v_{0,i})_{i=1}^N$ from the cryptotexts with valid proofs of knowledge. Then for $j = 1, \ldots, k$ mix-server M_j computes

$$L_j = (u_{j,i}, v_{j,i})_{i=1}^N = (g^{x_{j,\pi_j(i)}} u_{j-1,\pi_j(i)}, Y_j^{x_{j,\pi_j(i)}} v_{j-1,\pi_j(i)} / u_{j-1,\pi_j(i)}^{x_j})$$

for a randomly chosen permutation $\pi_j \in \Sigma_N$ and random exponents $x_{j,i} \in \mathbb{Z}_q$, and then proves knowledge of $((x_{j,\pi_j(i)})_{i=1}^N, x)$ that satisfy this relation. The list of cleartexts is given by $(v_{k,i})_{i=1}^N$. The relation that the input L_{j-1} and output L_j of a mix-server M_j must satisfy is formalized as follows.

Definition 6 (Knowledge of Correct Decryption-Re-encryption-Permutation). *Define for each N a relation $R_{\mathrm{DRP}} \subset (G_q^3 \times G_q^{2N} \times G_q^{2N}) \times (\mathbb{Z}_q \times \mathbb{Z}_q)$, by*

$$((g, y, Y, \{(u_i, v_i)\}_{i=1}^N, \{(u'_i, v'_i)\}_{i=1}^N), ((x_i)_{i=1}^N, x) \in R_{\mathrm{DRP}}$$

precisely when $y = g^x$ and $(u'_i, v'_i) = (g^{x_{\pi(i)}} u_{\pi(i)}, Y^{x_{\pi(i)}} v_{\pi(i)} / u_{\pi(i)}^x)$ for $i = 1, \ldots, N$ and some permutation $\pi \in \Sigma_N$.

We are now ready to transform Protocol 2 into a protocol for proving the above relation instead.

Protocol 5 (Proof of Decryption-Re-encryption-Permutation). The common input consists of an RSA modulus $\mathbf{N}$ and $\mathbf{g}, \mathbf{h} \in \mathrm{QR}_{\mathbf{N}}$, generators $g, g_1, \ldots, g_N \in G_q$, public keys $y = g^x$, Y, and two lists $L = (u_i, v_i)_{i=1}^N$ and $L' = (u'_i, v'_i)_{i=1}^N$ such that $(u'_i, v'_i) = (g^{x_{\pi(i)}} u_{\pi(i)}, Y^{x_{\pi(i)}} v_{\pi(i)}/u^x_{\pi(i)})$ for some permutation π of N elements. The private input to the prover consists of $((x_i)_{i=1}^N, x)$.

1. The prover chooses $r'_i \in \mathbb{Z}_q$ randomly, computes $(w_i)_{i=1}^N = (g^{r'_i} g_{\pi^{-1}(i)})_{i=1}^N$, and hands $(w_i)_{i=1}^N$ to the verifier.
2. The verifier chooses random primes $p_1, \ldots, p_N \in [2^{K_3-1}, 2^{K_3}-1]$, and hands $(p_i)_{i=1}^N$ to the prover.
3. Both parties compute $(U, V, W) = (\prod_{i=1}^N u_i^{p_i}, \prod_{i=1}^N v_i^{p_i}, \prod_{i=1}^N w_i^{p_i})$.
4. The prover chooses $k_1, k_2 \in \mathbb{Z}_q$, $l_1, l_2, l_x, l_s, l_{r'} \in \mathbb{Z}_q$ randomly. Then it chooses $t_i, t'_i \in [0, 2^{K_2+K_5}-1]$, $s_i, s'_i \in [0, 2^{K_2+K_4+2K_5}-1]$, and $r_i \in [0, 2^{K_3+K_4+K_5}-1]$ for $i = 1, \ldots, N$ randomly. Then it chooses an element $s \in [0, 2^{K_2+NK_3+K_4+K_5+\log_2 N}-1]$, and $s' \in [0, 2^{K_2+K_5+\log_2 N}-1]$ randomly and computes
$$(b_1, b_2) = (g_1^{k_1} g^{x_s}, g_1^{k_2} Y^{x_s} U^{-x}) \tag{79}$$
$$(\beta_1, \beta_2, \beta_3) = \left(g_1^{l_1} g^{l_s}, g_1^{l_2} Y^{l_s} U^{-l_x}, g^{l_x}\right) \tag{80}$$
$$(\alpha_1, \alpha_2, \alpha_3) = \left(g_1^{l_1} \prod_{i=1}^N (u'_i)^{r_i}, g_1^{l_2} \prod_{i=1}^N (v'_i)^{r_i}, g^{l_{r'}} \prod_{i=1}^N g_i^{r_i}\right) \tag{81}$$
$$\mathbf{b}_0 = \mathbf{g} \tag{82}$$
$$(\mathbf{b}_i, \mathbf{b}'_i)_{i=1}^N = (\mathbf{h}^{t_i} \mathbf{b}_{i-1}^{p_{\pi(i)}}, \mathbf{h}^{t'_i} \mathbf{g}^{p_{\pi(i)}})_{i=1}^N \tag{83}$$
$$(\boldsymbol{\gamma}_i, \boldsymbol{\gamma}'_i)_{i=1}^N = (\mathbf{h}^{s_i} \mathbf{b}_{i-1}^{r_i}, \mathbf{h}^{s'_i} \mathbf{g}^{r_i})_{i=1}^N \tag{84}$$
$$(\boldsymbol{\gamma}, \boldsymbol{\gamma}') = (\mathbf{h}^s, \mathbf{h}^{s'}) \ , \tag{85}$$
and $((b_1, b_2), (\beta_1, \beta_2, \beta_3), (\alpha_1, \alpha_2, \alpha_3), (\mathbf{b}_i, \mathbf{b}'_i)_{i=1}^N, (\boldsymbol{\gamma}_i, \boldsymbol{\gamma}'_i)_{i=1}^N, (\boldsymbol{\gamma}, \boldsymbol{\gamma}'))$ is handed to the verifier.
5. The verifier chooses $c \in [2^{K_4-1}, 2^{K_4}-1]$ randomly and hands c to the prover.
6. Define $x_s = \sum_{i=1}^N x_i p_i$. The prover computes
$$\begin{aligned}
f_1 &= ck_1 + l_1 && \bmod q \\
f_2 &= ck_2 + l_2 && \bmod q \\
f_x &= cx + l_x && \bmod q \\
f_s &= cx_s + l_s && \bmod q \\
f_{r'} &= cr' + l_{r'} && \bmod q \\
(e_i, e'_i)_{i=1}^N &= (ct_i + s_i, ct'_i + s'_i)_{i=1}^N && \bmod 2^{K_2+K_4+2K_5} \\
(d_i)_{i=1}^N &= (cp_{\pi(i)} + r_i)_{i=1}^N && \bmod 2^{K_3+K_4+K_5} \\
e &= ct + s && \bmod 2^{K_2+NK_3+K_4+K_5+\log_2 N} \\
e' &= ct' + s' && \bmod 2^{K_2+K_5+\log_2 N}
\end{aligned}$$
Then it hands $((f_1, f_2, f_s, f_{r'}), (e_i, e'_i)_{i=1}^N, (d_i)_{i=1}^N, (e, e'))$ to the verifier.

7. The verifier checks that L' is lexicographically sorted and that

$$(b_1^c \beta_1, b_2^c \beta_2, y^c \beta_3) = (g_1^{f_1} g^{f_s}, g_1^{f_2} Y^{f_s} U^{-f_x}, g^{f_x}) \tag{86}$$

$$((b_1 U)^c \alpha_1, (b_2 V)^c \alpha_2, W^c \alpha_3) = \left(g_1^{f_1} \prod_{i=1}^{N} (u_i')^{d_i}, g_1^{f_2} \prod_{i=1}^{N} (v_i')^{d_i}, g^{f_{r'}} \prod_{i=1}^{N} g_i^{d_i} \right) \tag{87}$$

$$(\mathbf{b}_i^c \boldsymbol{\gamma}_i, (\mathbf{b}_i')^c \boldsymbol{\gamma}_i')_{i=1}^N = (\mathbf{h}^{e_i} \mathbf{b}_{i-1}^{d_i}, \mathbf{h}^{e_i'} \mathbf{g}^{d_i})_{i=1}^N \tag{88}$$

$$(\mathbf{g}^{-\prod_{i=1}^N p_i} \mathbf{b}_N)^c \boldsymbol{\gamma} = \mathbf{h}^e \tag{89}$$

$$\left(\mathbf{g}^{-\sum_{i=1}^N p_i} \prod_{i=1}^{N} \mathbf{b}_i' \right)^c \boldsymbol{\gamma}' = \mathbf{h}^{e'} \quad . \tag{90}$$

Security Analysis. Again, the security analysis of the protocol is almost identical to the analysis of Protocol 2.

Proposition 5. *Protocol 4 is honest verifier statistical zero-knowledge.*

Proof (Proposition 5). The simulator chooses $p_1, \ldots, p_N$ and c honestly. Then it chooses $f_1, f_1, f_x, f_{r'} \in \mathbb{Z}_q$, $e \in [0, 2^{K_2+NK_3+K_4+K_5+\log_2 N}-1]$, $e' \in [0, 2^{K_2+K_5+\log_2 N} - 1]$, $(e_i, e_i')_{i=1}^N \in [0, 2^{K_2+K_4+2K_5} - 1]^N$, and $(d_i)_{i=1}^N \in [0, 2^{K_3+K_4+K_5} - 1]^N$ randomly. It also chooses $b_1, b_2 \in G_q$ and $\mathbf{b}_i, \mathbf{b}_i' \in \mathrm{QR}_{\mathbf{N}}$ randomly. Finally, the simulator defines, $(\beta_1, \beta_2, \beta_3)$ by Equations (86), $(\alpha_1, \alpha_2, \alpha_3)$ by Equations (87), $\boldsymbol{\gamma}_i$ and $\boldsymbol{\gamma}_i'$ by Equation (88), $\boldsymbol{\gamma}$ by Equation (89), and $\boldsymbol{\gamma}'$ by Equation (90) respectively. The distribution of the resulting elements is statistically close to the distribution of the corresponding elements in the protocol.

We denote the jth transcript in a list of transcripts by $T_j = (I_j, W_j, P_j, C_j, c_j, R_j)$ where I_j denotes the common input, W_j denotes the list of commitments w_i, and

$$P_j = (p_{j,1}, \ldots, p_{j,N})$$
$$C_j = ((b_{j,1}, b_{j,2}), (\beta_{j,1}, \beta_{j,2}, \beta_{j,3}), (\alpha_{j,1}, \alpha_{j,2}, \alpha_{j,3}), (\mathbf{b}_{j,i}, \mathbf{b}_{j,i}')_{i=1}^N, (\boldsymbol{\gamma}_{j,i}, \boldsymbol{\gamma}_{j,i}')_{i=1}^N, (\boldsymbol{\gamma}_j, \boldsymbol{\gamma}_j'))$$
$$R_j = ((f_{j,1}, f_{j,2}, f_{j,x}, f_{j,s}, f_{j,r'}), (e_{j,i}, e_{j,i}')_{i=1}^N, (d_{j,i})_{i=1}^N, (e_j, e_j')) \quad .$$

Lemma 8. *Lemma 3 holds also for Protocol 5, but with the notation above and with the main conclusion replaced by*

1. Main Conclusion**. *Elements $x, x_1, \ldots, x_N \in \mathbb{Z}_q$ and a permutation $\pi \in \Sigma_N$ s.t.*

$$((g, y, L, L'), (x, (x_i)_{i=1}^N)) \in R_{\mathrm{DRP}} \quad .$$

Proof. The proof is identical except for the last section of the proof. This must be replaced by the following.

The Common Input (g, y, Y, L, L') Satisfies $((g, y, Y, L, L'), ((x_i)_{i=1}^N, x)) \in R_{\mathrm{RP}}$. From the above we may assume that $\pi_j = \pi$ for all $j = 1, \ldots, N$, so we drop the subscript and simply write π from now on. Equations (86) imply that

$$\begin{aligned} b_1^{c_{j+N}-c_j} &= g_1^{f_{j+N,1}-f_{j,1}} g^{f_{j+N,s}-f_{j,s}} \quad , \\ b_2^{c_{j+N}-c_j} &= g_1^{f_{j+N,2}-f_{j,2}} Y^{f_{j+N,s}-f_{j,s}} U^{-(f_{j+N,x}-f_{j,x})} \quad , \text{ and} \\ y^{c_{j+N}-c_j} &= g^{f_{j+N,x}-f_{j,x}} \quad . \end{aligned}$$

Define

$$\kappa_1 = (f_{j+N,1} - f_{j,1})/(c_{j+N} - c_j) \quad , \tag{91}$$

$$\kappa_2 = (f_{j+N,2} - f_{j,2})/(c_{j+N} - c_j) \quad , \tag{92}$$

$$\xi = (f_{j+N,x} - f_{j,x})/(c_{j+N} - c_j) \quad , \text{ and} \tag{93}$$

$$\xi_{j,s} = (f_{j+N,s} - f_{j,s})/(c_{j+N} - c_j) \quad . \tag{94}$$

Then we have $g^x = y = g^\xi$, so $\xi = x$. We also have

$$b_1 = g_1^{\kappa_1} g^{\xi_{j,s}} \quad , \text{ and} \tag{95}$$

$$b_2 = g_1^{\kappa_2} Y^{\xi_{j,s}} U^{-x} \quad . \tag{96}$$

From Equation (87) we have

$$\begin{aligned} (b_1 U)^{c_{j+N}-c_j} &= g_1^{f_{j+N,1}-f_{j,1}} \prod_{i=1}^N (u_i')^{d_{j+N,i}-d_{j,i}} \quad , \text{ and} \\ (b_2 V)^{c_{j+N}-c_j} &= g_1^{f_{j+N,2}-f_{j,2}} \prod_{i=1}^N (v_i')^{d_{j+N,i}-d_{j,i}} \quad . \end{aligned}$$

Our definitions of κ_1 and κ_2 in Equations (91) and (92), and the definition of ρ_i, which equals $p_{j,\pi(i)}$ in Equation (23) imply that

$$\begin{aligned} b_1 U &= g_1^{\kappa_1} \prod_{i=1}^N (u_i')^{p_{j,\pi(i)}} \quad , \text{ and} \\ b_2 V &= g_1^{\kappa_2} \prod_{i=1}^N (v_i')^{p_{j,\pi(i)}} \quad . \end{aligned}$$

If we combine Equations (95) and (96) with the two equations above we have

$$g^{\xi_{j,s}} \prod_{i=1}^N u_i^{p_{ji}} = \quad g^{\xi_{j,s}} U \quad = \prod_{i=1}^N (u_i')^{p_{j,\pi(i)}} \quad , \text{ and} \tag{97}$$

$$Y^{\xi_{j,s}} U^{-x} \prod_{i=1}^N v_i^{p_{ji}} = Y^{\xi_{j,s}} U^{-x} V = \prod_{i=1}^N (v_i')^{p_{j,\pi(i)}} \quad , \tag{98}$$

for $j = 1, \ldots, N$. We apply the coefficients $a_{l,1}, \ldots, a_{l,N} \in \mathbb{Z}_q$ introduced above to the Equations (97) and (98) and conclude that

$$g^{\sum_{j=1}^{N} \xi_{j,s} a_{l,j}} u_l = \prod_{j=1}^{N} \left(g^{\xi_{j,s}} \prod_{i=1}^{N} u_i^{p_{ji}} \right)^{a_{l,j}} = \prod_{j=1}^{N} \left(\prod_{i=1}^{N} (u_i')^{p_{j,\pi(i)}} \right)^{a_{l,j}} = u'_{\pi^{-1}(l)} \ , \text{ and}$$

$$Y^{\sum_{j=1}^{N} \xi_{j,s} a_{l,j}} v_l u_l^{-x} = \prod_{j=1}^{N} \left(Y^{\xi_{j,s}} U^{-x} \prod_{i=1}^{N} v_i^{p_{ji}} \right)^{a_{l,j}} = \prod_{j=1}^{N} \left(\prod_{i=1}^{N} (v_i')^{p_{j,\pi(i)}} \right)^{a_{l,j}} = v'_{\pi^{-1}(l)} \ .$$

This concludes the proof, since we can set $x_l = \sum_{j=1}^{N} \xi_{j,s} a_{l,j}$.

Consider the following definitional changes compared with the analysis of Protocol 2. Let A be the machine that given an RSA modulus $\mathbf{N}$ and $\mathbf{g}, \mathbf{h} \in \mathrm{QR}_{\mathbf{N}}$, and $g, g_1, \ldots, g_N \in G_q$ as input generates the remainder of the common input consisting of public keys $y, Y \in G_q$, and two lists $L = (u_i, v_i)_{i=1}^{N} \in G_q^{2N}$ and $L' = (u_i', v_i')_{i=1}^{N} \in G_q^{2N}$, and then plays the role of the prover in Protocol 2 on common input $((\mathbf{N}, \mathbf{g}, \mathbf{h}), (g, g_1, \ldots, g_N), y, Y, (L, L'))$. We let Acc be the predicate that given a transcript T outputs 1 or 0 depending on if the verifier in the protocol above would accept the transcript or not.

Given these changes is straightforward to see, from the analysis in Sections D.2 and D.2, that the following propositions hold.

Proposition 6. *Assume the strong RSA-assumption and the DL-assumption. Then for all polynomial-size circuit families $A = \{A_K\}$ it holds that $\forall c > 0$, $\exists K_0$, such that for $K_1 \geq K_0$*

$$\Pr_{\boldsymbol{\Gamma}, \overline{g}, r_{\mathrm{p}}, r_{\mathrm{v}}} [\mathrm{Acc}(T_A(\boldsymbol{\Gamma}, \overline{g}, r_{\mathrm{p}}, r_{\mathrm{v}})) = 1 \wedge I_A(\boldsymbol{\Gamma}, \overline{g}, r_{\mathrm{p}}) \notin L_{R_{\mathrm{RP}}}] < \frac{1}{{K_1}^c} \ .$$

F.3 Further Shuffle Relations

We have given a detailed description of our approach for the relations R_{DP}, R_{RP}, and R_{DRP}, but the approach is easily adapted to a proof of a shuffle for many other relations that involve permutation of the cryptotexts.

G Re-encryption Shuffle of Paillier Cryptotexts

In this section we show how a re-encryption shuffle for the Paillier cryptosystem can be constructed by a small modification to the protocol in Section F.1.

G.1 The Paillier Cryptosystem

Paillier [44] introduces a cryptosystem based on the decisional composite residuosity (DCR) assumption, namely that it is hard to distinguish nth residues from nth non-residues modulo n, when n is a product of two large primes. This assumption is formalized in Section H.

The Basic Scheme. The scheme is described as follows. On input 1^K the key generator chooses two safe $K/2$-bit primes p and q randomly and defines the public key $\mathsf{n} = \mathsf{pq}$. For notational convenience we define $\mathsf{g} = \mathsf{n} + 1$ and $\mathsf{z} = (\mathsf{p}-1)(\mathsf{q}-1)/4$. Then it chooses a random secret key d under the restriction $d = 0 \bmod \mathsf{z}$ and $d = 1 \bmod \mathsf{n}$ and outputs (n, d).

The cryptosystem is based on the algebraic properties of n and g. Recall that $\mathbb{Z}^*_{\mathsf{n}^2} = G_{\mathsf{n}} \times G_{\mathsf{z}} \times V$, where G_{n} and G_{z} are groups of order n and z, and V is the Klein 4-group. Furthermore, g is a generator of G_{n} and each element in $G_{\mathsf{z}} \times V$ can be written $r^{\mathsf{n}} \bmod \mathsf{n}^2$ with $r \in \mathbb{Z}^*_{\mathsf{n}}$. Note that $\mathsf{g}^m = 1 + m\mathsf{n} \bmod \mathsf{n}^2$. We define $L(u) = (u-1)/\mathsf{n}$ and have $L(\mathsf{g}^m) = m$.

To encrypt a message $m \in \mathbb{Z}_{\mathsf{n}}$ a random $r \in \mathbb{Z}^*_{\mathsf{n}}$ is chosen and the cryptotext is defined by $u = E_{\mathsf{n}}(m, r) = \mathsf{g}^m r^{2\mathsf{n}} \bmod \mathsf{n}^2$. To decrypt a cryptotext u the decryptor computes $D_d(u) = L(u^d \bmod \mathsf{n}^2)$. To see why this works it suffices to note that every cryptotext $u \in G_{\mathsf{n}} \times G_{\mathsf{z}}$ and when $d = 0 \bmod \mathsf{z}$ and $d = 1 \bmod \mathsf{n}$ we have $u^d \bmod \mathsf{n}^2 = \mathsf{g}^m$. In other words exponentiation by d collapses the group G_{z}, but keeps G_{n} fixed.

We remark that in the original version of the cryptosystem, encryption is defined as $\mathsf{g}^m r^{\mathsf{n}} \bmod \mathsf{n}^2$, i.e., we exponentiate with $2\mathsf{n}$ instead of n. It is easy to see that this does not decrease the security of the scheme. Thus, the following proposition follows from [44].

Proposition 7. *The Paillier cryptosystem is semantically secure under the DCR-assumption.*

An interesting property of the scheme is that given a cryptotext $K_1 = E_{\mathsf{n}}(1, R_1) = \mathsf{g}R_1^{2\mathsf{n}} \bmod \mathsf{n}^2$ of 1 an alternative way to encrypt a message m is to compute $E_{K_1,\mathsf{n}}(m, r) = {K_1}^m r^{2\mathsf{n}} \bmod \mathsf{n}^2$, i.e., (K_1, n) may be viewed as a variant public key.

Let $\mathsf{g_z}$ be a generator of G_{z}. Then there exists a $2\mathsf{n}$th root of $\mathsf{g_z}$ and an alternative way to encrypt a message is as $E_{\mathsf{g_z},\mathsf{n}}(m, s) = \mathsf{g}^m \mathsf{g_z}^s \bmod \mathsf{n}^2$, where s is chosen randomly in $[0, 2^{K+K_5} - 1]$. The distribution of the new type of cryptotext is statistically close to the distribution of the type we define above and K_5 decides the statistical distance.

The cryptosystem is homomorphic, i.e., $E_{\mathsf{n}}(m_1)E_{\mathsf{n}}(m_2) = E_{\mathsf{n}}(m_1 + m_2)$. As a consequence it is possible to re-encrypt a cryptotext u using randomness $s \in \mathbb{Z}^*_{\mathsf{n}}$ by computing $uE_{\mathsf{n}}(0, s) = E_{\mathsf{n}}(m, rs)$, or alternatively as $uE_{\mathsf{g_z},\mathsf{n}}(0, s) = E_{\mathsf{n}}(m, r\mathsf{g_z}^{s/2\mathsf{n}})$ using randomness $s \in [0, 2^{K+K_5} - 1]$. The distributions of the resulting cryptotexts are again statistically close. We use the latter type of re-encryption.

Using the re-encryption property it is straightforward to define a Paillier based mix-net with the same overall structure as that described in Section F.1. The re-encryption-permutation relation corresponding to the transformation computed by each mix-server is then defined as follows.

Definition 7 (Knowledge of Correct Re-encryption-Permutation). *Define for each N and n a relation $R_{\mathrm{PRP}} \subset (\mathrm{QR}^N_{\mathsf{n}^2} \times \mathrm{QR}^N_{\mathsf{n}^2}) \times (\mathbb{Z} \times \mathbb{Z}^N)$, by*

$$((\{u_i\}_{i=1}^N, \{u'_i\}_{i=1}^N), (a, (x_i)_{i=1}^N)) \in R_{\mathrm{PRP}}$$

precisely when $a < \sqrt{\mathsf{n}}/4$ *equals one or is prime and* $(u_i')^a = \mathsf{g_z}^{x_{\pi(i)}} u_{\pi(i)}^a \bmod \mathsf{n}^2$ *for* $i = 1, \ldots, N$ *and some permutation* $\pi \in \Sigma_N$ *such that the list* $\{u_i'\}_{i=1}^N$ *is sorted lexicographically.*

Note that when $a < \sqrt{\mathsf{n}}/4$, a is relatively prime to n. Thus, a is invertible in $\mathbb{Z}_\mathsf{n}$ and u_i and u_i encrypts identical messages. The reason for introducing a is to allow extraction of other witnesses than the witness used by the prover. This simplifies the analysis. In the protocol below the private input to the prover always satisfy $a = 1$ and $x_i \in [0, 2^{K+K_5} - 1]$.

We are now ready to transform Protocol F.1 into a protocol for proving the Paillier based re-encryption-permutation relation.

Protocol 6 (Proof of Re-encryption-Permutation). The common input consists of an RSA modulus $\mathbf{N}$ and $\mathbf{g}, \mathbf{h} \in \mathrm{QR}_\mathbf{N}$, generators $g, g_1, \ldots, g_N \in G_q$, a public key n, and two lists $L = (u_i)_{i=1}^N$ and $L' = (u_i')_{i=1}^N$ such that $u_i' = \mathsf{g_z}^{x_{\pi(i)}} u_{\pi(i)}$ mod n^2 for some permutation π of N elements. The private input to the prover consists of $(x_i)_{i=1}^N$.

1. The prover chooses $r_i' \in \mathbb{Z}_q$ randomly, computes $(w_i)_{i=1}^N = (g^{r_i'} g_{\pi^{-1}(i)})_{i=1}^N$, and hands $(w_i)_{i=1}^N$ to the verifier.
2. The verifier chooses random primes $p_1, \ldots, p_N \in [2^{K_3-1}, 2^{K_3} - 1]$, and hands $(p_i)_{i=1}^N$ to the prover. Then the prover defines $x_s = \sum_{i=1}^N x_i p_i$.
3. Both parties compute $(U, W) = (\prod_{i=1}^N u_i^{p_i}, \prod_{i=1}^N w_i^{p_i})$.
4. The prover chooses $k_1 \in [0, 2^{K_7+K_5} - 1]$, $k_2 \in [0, 2^{K_2+K_5} - 1]$, $l_1 \in [0, 2^{K_7+2K_5} - 1]$, $l_2 \in [0, 2^{K_2+2K_5} - 1]$, $l_s \in [0, 2^{K_7+K_3+2K_5+\log_2 N} - 1]$, and $l_{r'} \in \mathbb{Z}_q$ randomly. Then it chooses $t_i, t_i' \in [0, 2^{K_2+K_5} - 1]$, $s_i, s_i' \in [0, 2^{K_2+K_4+2K_5} - 1]$, and $r_i \in [0, 2^{K_3+K_4+K_5} - 1]$ for $i = 1, \ldots, N$ randomly. Then it chooses an element $s \in [0, 2^{K_2+NK_3+K_4+K_5+\log_2 N} - 1]$, and $s' \in [0, 2^{K_2+K_5+\log_2 N} - 1]$ randomly and computes

$$(b_1, b_2, b_3) = (\mathsf{g_z}^{k_1} \mathsf{g_z}^{x_s}, \mathbf{h}^{k_2} \mathbf{g}^{k_1}, \mathbf{h}^{k_3} \mathbf{g}^{x_s}) \tag{99}$$

$$(\beta_1, \beta_2, \beta_3) = \left(\mathsf{g_z}^{l_1} \mathsf{g_z}^{l_s}, \mathbf{h}^{l_2} \mathbf{g}^{l_1}, \mathbf{h}^{l_3} \mathbf{g}^{l_s}\right) \tag{100}$$

$$(\alpha_1, \alpha_3) = \left(\mathsf{g_z}^{l_1} \prod_{i=1}^N (u_i')^{r_i}, g^{l_{r'}} \prod_{i=1}^N g_i^{r_i}\right) \tag{101}$$

$$\mathbf{b}_0 = \mathbf{g} \tag{102}$$

$$(\mathbf{b}_i, \mathbf{b}_i')_{i=1}^N = (\mathbf{h}^{t_i} \mathbf{b}_{i-1}^{p_{\pi(i)}}, \mathbf{h}^{t_i'} \mathbf{g}^{p_{\pi(i)}})_{i=1}^N \tag{103}$$

$$(\boldsymbol{\gamma}_i, \boldsymbol{\gamma}_i')_{i=1}^N = (\mathbf{h}^{s_i} \mathbf{b}_{i-1}^{r_i}, \mathbf{h}^{s_i'} \mathbf{g}^{r_i})_{i=1}^N \tag{104}$$

$$(\boldsymbol{\gamma}, \boldsymbol{\gamma}') = (\mathbf{h}^s, \mathbf{h}^{s'}) \ , \tag{105}$$

   and $((b_1, b_2), (\beta_1, \beta_2), (\alpha_1, \alpha_2, \alpha_3), (\mathbf{b}_i, \mathbf{b}_i')_{i=1}^N, (\boldsymbol{\gamma}_i, \boldsymbol{\gamma}_i')_{i=1}^N, (\boldsymbol{\gamma}, \boldsymbol{\gamma}'))$ is handed to the verifier.
5. The verifier chooses $c \in [2^{K_4-1}, 2^{K_4} - 1]$ randomly and hands c to the prover.

6. The prover computes

$$\begin{aligned}
f_1 &= ck_1 + l_1 && \bmod 2^{K_7+2K_5} \\
f_2 &= ck_2 + l_2 && \bmod 2^{K_2+2K_5} \\
f_3 &= ck_3 + l_3 && \bmod 2^{K_2+2K_5} \\
f_s &= cx_s + l_s && \bmod 2^{K_7+K_3+2K_5+\log_2 N} \\
f_{r'} &= cr' + l_{r'} && \bmod q \\
(e_i, e'_i)_{i=1}^N &= (ct_i + s_i, ct'_i + s'_i)_{i=1}^N && \bmod 2^{K_2+K_4+2K_5} \\
(d_i)_{i=1}^N &= (cp_{\pi(i)} + r_i)_{i=1}^N && \bmod 2^{K_3+K_4+K_5} \\
e &= ct + s && \bmod 2^{K_2+NK_3+K_4+K_5+\log_2 N} \\
e' &= ct' + s' && \bmod 2^{K_2+K_5+\log_2 N}
\end{aligned}$$

Then it hands $((f_1, f_2, f_s, f_{r'}), (e_i, e'_i)_{i=1}^N, (d_i)_{i=1}^N, (e, e'))$ to the verifier.
7. The verifier checks that L' is lexicographically sorted and that

$$(b_1^c\beta_1, b_2^c\beta_2, b_3^c\beta_3) = (\mathsf{g}_{\mathsf{z}}^{f_1}\mathsf{g}_{\mathsf{z}}^{f_s}, \mathbf{h}^{f_2}\mathbf{g}^{f_1}, \mathbf{h}^{f_3}\mathbf{g}^{f_s}) \tag{106}$$

$$((b_1U)^c\alpha_1, W^c\alpha_3) = \left(\mathsf{g}_{\mathsf{z}}^{f_1}\prod_{i=1}^N (u'_i)^{d_i}, g^{f_{r'}}\prod_{i=1}^N g_i^{d_i}\right) \tag{107}$$

$$(\mathbf{b}_i^c\boldsymbol{\gamma}_i, (\mathbf{b}'_i)^c\boldsymbol{\gamma}'_i)_{i=1}^N = (\mathbf{h}^{e_i}\mathbf{b}_{i-1}^{d_i}, \mathbf{h}^{e'_i}\mathbf{g}^{d_i})_{i=1}^N \tag{108}$$

$$(\mathbf{g}^{-\prod_{i=1}^N p_i}\mathbf{b}_N)^c\boldsymbol{\gamma} = \mathbf{h}^e \tag{109}$$

$$\left(\mathbf{g}^{-\sum_{i=1}^N p_i}\prod_{i=1}^N \mathbf{b}'_i\right)^c \boldsymbol{\gamma}' = \mathbf{h}^{e'} \quad . \tag{110}$$

Security Analysis. The security analysis of the protocol is similar to the analysis of the previous protocols. Thus, we only indicate the changes needed for the proof to go through.

Proposition 8. *Protocol 6 is honest verifier statistical zero-knowledge.*

Proof (Proposition 3). The simulator chooses $p_1, \ldots, p_N$ and c honestly. Then it chooses $f_1 \in [0, 2^{K_7+2K_5}-1]$, $f_2 \in [0, 2^{K_2+2K_5}-1]$, $f_s \in [0, 2^{K_7+K_3+2K_5+\log_2 N} - 1]$, $f_{r'} \in \mathbb{Z}_q$, $e \in [0, 2^{K_2+NK_3+K_4+K_5+\log_2 N} - 1]$, $e' \in [0, 2^{K_2+K_5+\log_2 N} - 1]$, $(e_i, e'_i)_{i=1}^N \in [0, 2^{K_2+K_4+2K_5} - 1]^N$, and $(d_i)_{i=1}^N \in [0, 2^{K_3+K_4+K_5} - 1]^N$ randomly. It also chooses $b_1, b_2 \in G_q$ and $\mathbf{b}_i, \mathbf{b}'_i \in \mathrm{QR}_{\mathbf{N}}$ randomly. Finally, the simulator defines, (β_1, β_2) by Equations (67), (α_1, α_3) by Equations (68), $\boldsymbol{\gamma}_i$ and $\boldsymbol{\gamma}'_i$ by Equation (69), $\boldsymbol{\gamma}$ by Equation (70), and $\boldsymbol{\gamma}'$ by Equation (71) respectively. The distribution of the resulting elements is statistically close to the distribution of the corresponding elements in the protocol.

We denote the jth transcript in a list of transcripts by $T_j = (I_j, W_j, P_j, C_j, c_j, R_j)$ where I_j denotes the common input, W_j denotes the list of commitments w_i, and

$$
\begin{aligned}
P_j &= (p_{j,1}, \ldots, p_{j,N}) \\
C_j &= ((b_{j,1}, b_{j,2}, b_{j,3}), (\beta_{j,1}, \beta_{j,2}, \beta_{j,3}), (\alpha_{j,1}, \alpha_{j,3}), (\mathbf{b}_{j,i}, \mathbf{b}'_{j,i})_{i=1}^N, (\boldsymbol{\gamma}_{j,i}, \boldsymbol{\gamma}'_{j,i})_{i=1}^N, (\boldsymbol{\gamma}_j, \boldsymbol{\gamma}'_j)) \\
R_j &= ((f_{j,1}, f_{j,2}, f_{j,x}, f_{j,s}, f_{j,r'}), (e_{j,i}, e'_{j,i})_{i=1}^N, (d_{j,i})_{i=1}^N, (e_j, e'_j)) \enspace .
\end{aligned}
$$

Lemma 9. *Lemma 3 holds also for Protocol 6, but with the modified notation and with the main conclusion replaced by*

1. Main Conclusion***. *An integer $a < \sqrt{\mathsf{n}}/2$ equal to one or prime, elements $x_1, \ldots, x_N \in \mathbb{Z}$ and a permutation $\pi \in \Sigma_N$ s.t.*

$$((L^a, (L')^a), (x_i)_{i=1}^N) \in R_{\mathrm{RP}_\mathsf{n}} \enspace ,$$

where we by L^a and $(L')^a$ mean termwise exponentiation.

Proof. The proof is identical except for the last section of the proof. This must be replaced by the following.

The Common Input (L, L') Satisfies $((L, L'), (x_i)_{i=1}^N) \in R_{\mathrm{RP}_\mathsf{n}}$. From the above we may assume that $\pi_j = \pi$ for all $j = 1, \ldots, N$, so we drop the subscript and simply write π from now on. Equations (106) imply that

$$
\begin{aligned}
b_1^{c_{j+N} - c_j} &= \mathsf{g}_\mathsf{z}^{f_{j+N,1} - f_{j,1}} \mathsf{g}_\mathsf{z}^{f_{j+N,s} - f_{j,s}} \enspace , \\
b_2^{c_{j+N} - c_j} &= \mathbf{h}^{f_{j+N,2} - f_{j,2}} \mathbf{g}^{f_{j+N,1} - f_{j,1}} \enspace , \text{ and} \\
b_3^{c_{j+N} - c_j} &= \mathbf{h}^{f_{j+N,3} - f_{j,3}} \mathbf{g}^{f_{j+N,s} - f_{j,s}} \enspace .
\end{aligned}
$$

If $c_{j+N} - c_j$ does not divide $f_{j+N,1} - f_{j,1}$, $f_{j+N,2} - f_{j,2}$, $f_{j+N,3} - f_{j,3}$, and $f_{j+N,s} - f_{j,s}$ Conclusion 2a of the lemma is satisfied.

Thus, we may assume that it does and define

$$\kappa_1 = (f_{j+N,1} - f_{j,1})/(c_{j+N} - c_j) \enspace , \tag{111}$$
$$\kappa_2 = (f_{j+N,2} - f_{j,2})/(c_{j+N} - c_j) \enspace , \tag{112}$$
$$\kappa_2 = (f_{j+N,3} - f_{j,3})/(c_{j+N} - c_j) \enspace , \text{ and} \tag{113}$$
$$\xi_{j,s} = (f_{j+N,s} - f_{j,s})/(c_{j+N} - c_j) \enspace . \tag{114}$$

Then we have

$$b_1 = \mathsf{g}_\mathsf{z}^{\kappa_1} \mathsf{g}_\mathsf{z}^{\xi_{j,s}} \enspace . \tag{115}$$

From Equation (107) we have

$$(b_1 U)^{c_{j+N} - c_j} = \mathsf{g}_\mathsf{z}^{f_{j+N,1} - f_{j,1}} \prod_{i=1}^N (u'_i)^{d_{j+N,i} - d_{j,i}} \enspace .$$

Our definitions of κ_1 and κ_2 in Equations (111) and (112), and the definition of ρ_i, which equals $p_{j,\pi(i)}$ in Equation (23) imply that

$$b_1 U = \mathsf{g}_\mathsf{z}^{\kappa_1} \prod_{i=1}^{N} (u'_i)^{p_{j,\pi(i)}} \quad .$$

If we combine Equation (115) and the equation above we have

$$\mathsf{g}_\mathsf{z}^{\xi_{j,s}} \prod_{i=1}^{N} u_i^{p_{ji}} = \mathsf{g}_\mathsf{z}^{\xi_{j,s}} U = \prod_{i=1}^{N} (u'_i)^{p_{j,\pi(i)}} \quad . \tag{116}$$

for $j = 1, \ldots, N$. Here we can not apply the coefficients $a_{l,1}, \ldots, a_{l,N} \in \mathbb{Z}_q$ introduced in the proofs of the protocols above. The problem is that we need that the lists $P_1, \ldots, P_N$ are linearly independent as vectors over $\mathbb{Z}_{\mathsf{nz}}$, since nz is the order of the group $\mathbb{Z}_{\mathsf{n}^2}^*$. The problem with this is that in most natural applications of the proof of a shuffle we must be able to extract a witness without knowledge of z. In other words we do not know how to invert elements in $\mathbb{Z}_{\mathsf{nz}}$.

However, we can perform Gauss elimination over the integers and start with column l. This gives a list of integers $a_{l,1}, \ldots, a_{l,N}$ such that $\sum_{j=1}^{N} a_{l,j} P_j = (\delta_{l,1}, \ldots, \delta_{l,N})$ with $\delta_{l,j} = 0$ for $j \neq l$ and $\delta_{l,l} = p_{l,l}$. We apply these coefficients to Equation (116) and conclude that

$$\mathsf{g}_\mathsf{z}^{\sum_{j=1}^{N} \xi_{j,s} a_{l,j}} u_l^{p_{l,l}} = \prod_{j=1}^{N} \left(\mathsf{g}_\mathsf{z}^{\xi_{j,s}} \prod_{i=1}^{N} u_i^{p_{ji}} \right)^{a_{l,j}} = \prod_{j=1}^{N} \left(\prod_{i=1}^{N} (u'_i)^{p_{j,\pi(i)}} \right)^{a_{l,j}} = (u'_{\pi^{-1}(l)})^{p_{l,l}} \quad .$$

This concludes the proof, since we can set $x_l = \sum_{j=1}^{N} \xi_{j,s} a_{l,j}$.

Consider the following definitional changes compared with the analysis of Protocol 2. Let A be the machine that given an RSA modulus $\mathbf{N}$ and $\mathbf{g}, \mathbf{h} \in \mathrm{QR}_\mathbf{N}$, and $g, g_1, \ldots, g_N \in G_q$ as input generates the remainder of the common input consisting of a public key $y \in G_q$, and two lists $L = (u_i, v_i)_{i=1}^{N} \in G_q^{2N}$ and $L' = (u'_i, v'_i)_{i=1}^{N} \in G_q^{2N}$, and then plays the role of the prover in Protocol 4 on common input $((\mathbf{N}, \mathbf{g}, \mathbf{h}), (g, g_1, \ldots, g_N), y, (L, L'))$. We let Acc be the predicate that given a transcript T outputs 1 or 0 depending on if the verifier in the protocol above would accept the transcript or not.

Given these changes it is straightforward to see, from the analysis in Sections D.2 and D.2, that the following proposition holds.

Proposition 9 (Knowledge Extraction). *There exists an algorithm $\mathcal{X}_A$ running an adversary A as a blackbox with the following properties.*

1. *If $\delta_A(\boldsymbol{\Gamma}, \overline{g}, r_\mathrm{p})$ is non-negligible, the expected number of calls made by $\mathcal{X}$ to A on input $(\boldsymbol{\Gamma}, \overline{g}, r_\mathrm{p})$ is $O(N/\delta_A(\boldsymbol{\Gamma}, \overline{g}, r_\mathrm{p}))$.*
2. *For every $c > 0$ the probability $\Pr[(I_A(\boldsymbol{\Gamma}, \overline{g}, r_\mathrm{p}), \mathcal{X}_A(\boldsymbol{\Gamma}, \overline{g}, r_\mathrm{p})) \in R_\mathrm{PRP} \mid \delta_A(\boldsymbol{\Gamma}, \overline{g}, r_\mathrm{p}) \geq K^{-c}]$ is overwhelming, where the probability is taken over $\boldsymbol{\Gamma}$, $\overline{g}$, r_p and the internal randomness of $\mathcal{X}$.*

H Cryptographic Assumptions

In this section we define the cryptographic assumptions we need.

H.1 The Ideal Bulletin Board

Functionality 6 (Bulletin Board (cf. [53])). The ideal *bulletin board* functionality, $\mathcal{F}_{\mathrm{BB}}$, running with participants $P_1, \ldots, P_k$ and ideal adversary $\mathcal{S}$.

1. $\mathcal{F}_{\mathrm{BB}}$ holds a database indexed on integers. Initialize a counter $c = 0$.
2. Upon receiving $(P_i, \texttt{Write}, m_i)$, $m_i \in \{0,1\}^*$, from $\mathcal{C_I}$, store (P_i, m_i) under the index c in the database, hand $(\mathcal{S}, \texttt{Write}, c, P_i, m_i)$ to $\mathcal{C_I}$, and set $c \leftarrow c+1$.
3. Upon receiving $(P_j, \texttt{Read}, c)$ from $\mathcal{C_I}$ check if a tuple (P_i, m_i) is stored in the database under c. If so hand $((\mathcal{S}, P_j, \texttt{Read}, c, P_i, m), (P_j, \texttt{Read}, c, P_i, m_i))$ to $\mathcal{C_I}$. If not, hand $((\mathcal{S}, P_j, \texttt{NoRead}, c), (P_j, \texttt{NoRead}, c))$ to $\mathcal{C_I}$.

H.2 Pseudo-Random Generator

Definition 8 (Pseudo-Random Ensemble). *The ensemble $X = \{X_n\}_{n\in\mathbb{N}}$ of random variables is pseudo-random if there exists an ensemble $U = \{U_{l(n)}\}_{n\mathbb{N}}$ such that X and U are indistinguishable in polynomial time.*

Definition 9 (Pseudo-Random Generator). *A pseudo-random generator is a deterministic polynomial-time algorithm* PRG *satisfying the following two conditions:*

1. *There exists a function $l : \mathbb{N} \to \mathbb{N}$ such that $l(n) > n$ for all $n \in \mathbb{N}$ and $|\mathsf{PRG}(s)| = l(|s|)$ for all $s \in \{0,1\}^*$.*
2. *The ensemble $\{\mathsf{PRG}(U_n)\}_{n\in\mathbb{N}}$ is pseudo-random.*

H.3 The Discrete Logarithm Assumption

Definition 10 (Discrete Logarithm Assumption). *Let G_q be a group of prime order q with generator g, and let h be randomly chosen in G_q. The Discrete Logarithm (DL) assumption states that for all polynomial-size circuit families $A = \{A_K\}$, $\forall c > 0$, $\exists K_0$, such that for $K > K_0$ we have*

$$\Pr[A(h) = \log_g h] < \frac{1}{K^c} \ .$$

Lemma 10 (Representation Problem). *Let G_q be a group of prime order q with generator g, and let $g_1, \ldots, g_N$ be randomly chosen in G_q (with N polynomial in K). Then under the DL-assumption it holds that for all polynomial-size circuit families $A = \{A_K\}$, $\forall c > 0$, $\exists K_0$, such that for $K > K_0$ we have*

$$\Pr[A(g_1, \ldots, g_N) = (\eta_1, \ldots, \eta_N) \neq 0 \wedge \prod_{i=1}^{N} g_i^{\eta_i} = 1] < \frac{1}{K^c} \ .$$

Proof. If the lemma is false there exists an adversary A, a constant c, and an infinite index set $\mathcal{N}$ such that the left side is larger than $1/K^c$ for $K \in \mathcal{N}$. Consider the adversary A' defined as follows. Given (g, h) as input it sets $e_1 = 1$, chooses $e_2, \ldots, e_N \in \mathbb{Z}_q$ and $2 \leq i \leq N$ randomly and defines $g_1 = g$, $g_l = g^{e_l}$ for $l \neq i$ and $g_i = h$. Then it computes $(\eta_l)_{l=0}^N \leftarrow A((g_l)_{l=1}^N)$ and outputs $\frac{1}{e_i} \sum_{l \neq i} e_l \eta_l$ if $e_i \neq 0$ and $\perp$ otherwise. If $e_i \neq 0$ we have

$$\prod_{l \neq i} g_l^{\eta_l} = h^{-e_i}$$

and it follows that the output is the logarithm of h.

The event $e_i \neq 0$ is independent of the event that A outputs a non-trivial representation, since the generated $(g_1, \ldots, g_N)$ is identically distributed to those in the lemma. Thus, from independence follows that A' outputs $\log_g h$ with probability at least $\frac{1}{NK^c}$, which contradicts the DL-assumption.

H.4 The Decision Diffie-Hellman Assumption

Definition 11 (Decision Diffie-Hellman Assumption). *Let $\alpha, \beta, \gamma \in \mathbb{Z}_q$ be randomly chosen. The (non-uniform) Decision Diffie-Hellman Assumption (DDH-assumption) for G_q states that for all polynomial-size circuit families $A = \{A_K\}$, $\forall c > 0$, $\exists K_0$, such that for $K > K_0$ we have*

$$|\Pr[A(g^\alpha, g^\beta, g^\gamma) = 1] - \Pr[A(g^\alpha, g^\beta, g^{\alpha\beta}) = 1]| < \frac{1}{K^c} \enspace .$$

We use a variant of the DDH-problem captured by the lemma below.

Lemma 11 (Variant DDH-Assumption). *Let $\alpha, \beta, \beta', \gamma, \gamma' \in \mathbb{Z}_q$ be randomly chosen. Then under the DDH-assumption it holds that for all polynomial-size circuit families $A = \{A_K\}$, $\forall c > 0$, $\exists K_0$, such that for $K > K_0$*

$$|\Pr[A(g^\alpha, g^\beta, g^\gamma, g^{\beta'}, g^{\gamma'}) = 1] - \Pr[A(g^\alpha, g^\beta, g^{\alpha\beta}, g^{\beta'}, g^{\alpha\beta'}) = 1]| < \frac{1}{K^c} \enspace .$$

Proof. Suppose that the lemma is false. Then there exists a circuit family A, a constant $c > 0$, and an infinite index set $\mathcal{N}$ such that

$$|\Pr[A(g^\alpha, g^\beta, g^\gamma, g^{\beta'}, g^{\gamma'}) = 1] - \Pr[A(g^\alpha, g^\beta, g^{\alpha\beta}, g^{\beta'}, g^{\alpha\beta'}) = 1]| \geq \frac{1}{K^c} \enspace .$$

This implies that one of the following inequalities hold

$$|\Pr[A(g^\alpha, g^\beta, g^\gamma, g^{\beta'}, g^{\gamma'}) = 1] - \Pr[A(g^\alpha, g^\beta, g^\gamma, g^{\beta'}, g^{\alpha\beta'}) = 1]| \geq \frac{1}{2K^c} \enspace , \text{ and}$$

$$|\Pr[A(g^\alpha, g^\beta, g^\gamma, g^{\beta'}, g^{\alpha\beta'}) = 1] - \Pr[A(g^\alpha, g^\beta, g^{\alpha\beta}, g^{\beta'}, g^{\alpha\beta'}) = 1]| \geq \frac{1}{2K^c} \enspace .$$

The former is impossible, since given a triple (u, v, w), the tuple $(u, g^\beta, g^\gamma, v, w)$ for random $\beta, \gamma \in \mathbb{Z}_q$ is identically distributed to the input to A in the right or left probability in the first equation depending on if (u, v, w) is a DDH-triple or not. The latter is impossible, since given a triple (u, v, w), the tuple $(u, v, w, g^{\beta'}, u^{\beta'})$, for a random $\beta' \in \mathbb{Z}_q$, is identically distributed to the input to A to the left or right probability in the second equation depending on if (u, v, w) is a random triple or if it is a DDH-triple.

H.5 The Strong RSA-Assumption

The strong RSA-assumption says that it is hard to compute any non-trivial root in $\mathbb{Z}_{\mathbf{N}}$ where $\mathbf{N}$ is an RSA modulus, even if allowed to select which root to compute. This differs from the standard RSA-assumption, where the root to compute is predetermined.

Definition 12 (Strong RSA-Assumption). *The strong RSA-assumption states that for all polynomial-size circuit families $A = \{A_K\}$, $\forall c > 0$, $\exists K_0$, such that for $K > K_0$*

$$\Pr[(\mathbf{p}, \mathbf{q}) \leftarrow \mathsf{RSA}(1^K), \mathbf{h} \leftarrow \mathbb{Z}^*_{\mathbf{pq}}, (\mathbf{b}, e) \leftarrow A(\mathbf{pq}, \mathbf{h}), \mathbf{b}^e = \mathbf{h}, e \neq \pm 1] < \frac{1}{K^c} \ .$$

I Review of the UC-Security Framework

In this section we give a short review of the universally composable security framework of Canetti [8]. This framework is very general, quite complex, and hard to describe both accurately and concisely. We have chosen to use a slightly simplified approach. For a general in depth discussion, intuition, and more details we refer the reader to Canetti [8]. Note that we consider only static adversaries.

Following Goldreich [21] and Canetti [8] we define the participants to be interactive Turing machines, and denote the set of interactive Turing machines by ITM.

Canetti assumes the existence of an "operating system" that takes care of the creation of subprotocols when needed. This is necessary to handle protocols with a large number of possible trees of calls to subprotocols, but for our purposes we may assume that all subprotocols are instantiated already at the start of the protocol.

Canetti models an asynchronous communication network, where the adversary has the power to delete, modify, and insert any messages of his choice. To do this he is forced to give details for exactly what the adversary is allowed to do. This becomes quite complex in the hybrid model. We instead factor out all aspects of the communication network into a separate concrete "communication model"-machine. The real, ideal, and hybrid models are then defined solely on how certain machines are linked. The adversary is defined as any ITM, and how the adversary can interact with other machines follows implicitly from the definitions of the real and ideal communication models.

Since each protocol or subprotocol communicate through its own copy of the "communication model", and all protocols are instantiated at the start of the protocol we need not bother with session ID:s. Such ID:s would clearly be needed if our protocols would be rewritten in the more general original security framework, but it is notationally convenient to avoid them.

We also assume that we may connect any pair of machines by a "link". Such a link is more or less equivalent to the notion of a link introduced by Goldreich [21]. But the original notion of a link has the problem that it requires a machine to have a pair of communication tapes for each link, which is problematic when the number of potential links is unbounded. This is a purely definitional problem of no importance and we trust the reader to fill in the details of this in any way he or she chooses. Thus the following is meaningful.

Definition 13. *An* ITM-*graph is a set* $V = \{P_1, \dots, P_t\} \subset \mathrm{ITM}$ *with a set of links* E *such that* (V, E) *is a connected graph, and no* P_i *is linked to any machine outside* V. *Let* ITMG *be the set of* ITM-*graphs.*

During the execution of an ITM-*graph, at most one participant is* active. *An active participant may deactivate itself and activate any of its neighbors, or it may halt, in which case the execution of the* ITM-*graph halts.*

The real communication model models an asynchronous communication network, in which the adversary can read, delete, modify, and insert any message of its choice. We will not make use of this communication model, since all communication in our protocols take place over an ideal bulletin board, but for completeness we give a definition.

Definition 14. *A* real communication model $\mathcal{C}$ *is a machine with a link* l_{P_i}, *to* P_i *for* $i = 1, \dots, k$, *and a link* $l_{\mathcal{A}}$ *to a real adversary* $\mathcal{A}$. *Its program is defined as follows.*

1. *If* m *is read on* l_s, *where* $s \in \{P_1, \dots, P_k\}$, *then* (s, m) *is written on* l_A *and* $\mathcal{A}$ *is activated.*
2. *If* (r, m) *is read on* $l_{\mathcal{A}}$, *where* $r \in \{P_1, \dots, P_k\}$, *then* m *is written on* l_r, *and* r *is activated.*

The ideal communication model below captures the fact that the adversary may decide if and when it would like to deliver a message from the ideal functionality to a participant, but it cannot read the contents of the communication between participants and the ideal functionality.

Definition 15. *An* ideal communication model $\mathcal{C}_{\mathcal{I}}$ *is a machine with a link* l_{P_i}, *to* P_i *for* $i = 1, \dots, k$, *and links* $l_{\mathcal{F}}$, *and* $l_{\mathcal{S}}$ *to an ideal functionality* $\mathcal{F}$ *and an ideal adversary* $\mathcal{S}$ *respectively. Its program is defined as follows.*

1. *If a message* m *is read on* l_s, *where* $s \in \{P_1, \dots, P_k\}$, *then* (s, m) *is written on* $l_{\mathcal{F}}$ *and* $\mathcal{F}$ *is activated.*
2. *If a message* (s, m) *written on* $l_{\mathcal{F}}$ *is returned unaltered*[5], m *is written on* l_s. *If not, any string read from* $l_{\mathcal{F}}$ *is interpreted as a list* $((r_1, m_1), \dots, (r_t, m_t))$, *where* $r_i \in \{\mathcal{S}, P_1, \dots, P_k\}$. *For each* m_i *a random string* $\tau_i \in \{0, 1\}^n$ *is*

[5] This special rule simplifies security proofs.

chosen, and (r_i, m_i) is stored under τ_i. Then $((r_1, |m_1|, \tau_1), \ldots, (r_t, |m_t|, \tau_t))$, where $|m_i|$ is the bit-length of m_i, is written to $l_{\mathcal{S}}$ and $\mathcal{S}$ is activated.

3. *Any string read from $l_{\mathcal{S}}$ is interpreted as a pair (b, τ), where $b \in \{0, 1\}$ and τ is an arbitrary string. If $b = 1$ and (r_i, m_i) is stored in the database under the index τ, m_i is written on l_{r_i} and r_i is activated. Otherwise $(\mathcal{S}, \tau)$ is written to $l_{\mathcal{F}}$ and $\mathcal{F}$ is activated.*

An *adversary* can normally corrupt some subset of the participants in a protocol. A *dummy participant* is a machine that given two links writes any message from one of the links on the other. There may be many copies of the dummy participant. Following Canetti we use the $\tilde{}$ -notation, e.g. $\tilde{P}$, for dummy participants.

The ideal model below captures the setup one wishes to realize, i.e. the environment may interact with the ideal functionality $\mathcal{F}$, except that the adversary $\mathcal{S}$ has some control over how the communication model behaves.

Definition 16. *The* ideal model *is defined to be a map* $\mathcal{I} : \mathrm{ITM}^2 \times \tilde{\mathrm{ITM}}^* \rightarrow \mathrm{ITMG}$*, where* $\mathcal{I} : (\mathcal{F}, \mathcal{S}, \tilde{P}_1, \ldots, \tilde{P}_k) \mapsto (V, E)$ *is given by:*

$$V = \{\mathcal{C}_{\mathcal{I}}, \mathcal{F}, \mathcal{S}, \tilde{P}_1, \ldots, \tilde{P}_k\}, \quad E = \{(\mathcal{S}, \mathcal{C}_{\mathcal{I}}), (\mathcal{C}_{\mathcal{I}}, \mathcal{F})\} \cup \bigcup_{i=1}^{k} \{(\tilde{P}_i, \mathcal{C}_{\mathcal{I}})\} \ .$$

If $\tilde{\pi} = (\tilde{P}_1, \ldots, \tilde{P}_k)$, we write $\mathcal{I}(\mathcal{S}, \tilde{\pi}^{\mathcal{F}})$ instead of $\mathcal{I}(\mathcal{F}, \mathcal{S}, \tilde{P}_1, \ldots, \tilde{P}_k)$ to ease notation.

The real model is supposed to capture the properties of the real world. The participants may interact over the real communication model.

Definition 17. *The* real model *is defined to be a map* $\mathcal{R} : \mathrm{ITM}^* \rightarrow \mathrm{ITMG}$*, where* $\mathcal{R} : (\mathcal{A}, P_1, \ldots, P_k) \mapsto (V, E)$ *is given by:*

$$V = \{\mathcal{C}, \mathcal{A}, P_1, \ldots, P_k\}, \quad E = \{(\mathcal{A}, \mathcal{C})\} \cup \bigcup_{i=1}^{k} \{(P_i, \mathcal{C})\} \ .$$

Let $(V, E) = \mathcal{I}(\mathcal{F}, \mathcal{S}, \tilde{P}_1, \ldots, \tilde{P}_k)$. Then we write $\mathcal{Z}(\mathcal{I}(\mathcal{F}, \mathcal{S}, \tilde{P}_1, \ldots, \tilde{P}_k))$ for the ITM-graph (V', E') defined by $V' = V \cup \{\mathcal{Z}\}$, and $E' = E \cup \{(\mathcal{Z}, \mathcal{S})\} \cup \bigcup_{i=1}^{k} \{(\mathcal{Z}, \tilde{P}_i)\}$. We use the corresponding notation in the real model case.

A hybrid model is a mix between a number of ideal and real models, and captures the execution of a real world protocol with access to some ideal functionalities. It is also a tool to modularize security proofs. It may be viewed as if we "glue" a number of ideal and real models onto an original real model.

Definition 18. *Suppose that we are given $(V, E) = \mathcal{R}(\mathcal{A}, \pi)$, $\pi = (P_1, \ldots, P_k)$. Let $(V_j, E_j) = \mathcal{I}(\mathcal{S}_j, \tilde{\pi}_j^{\mathcal{F}_j})$, $\tilde{\pi}_j = (\tilde{P}_{j,1}, \ldots, \tilde{P}_{j,k})$ for $j = 1, \ldots, t$, and $(V_j, E_j) = \mathcal{R}(\mathcal{S}_j, \pi_j)$, $\pi_j = (P_{j,1}, \ldots, P_{j,k})$ for $j = t + 1, \ldots, s$.*

We denote by $\mathcal{H}(\mathcal{A}^{(\mathcal{S}_1,\ldots,\mathcal{S}_t)}, \pi^{(\tilde{\pi}_1^{\mathcal{F}_1},\ldots,\tilde{\pi}_t^{\mathcal{F}_t},\pi_{t+1},\ldots,\pi_s)})$ *the* hybrid model *defined as the* ITM-*graph* (V', E')*, where*

$$V' = V \cup \bigcup_{j=1}^{t} V_j, \;\; and \;\; E' = E \cup \bigcup_{j=1}^{t} E_j \cup \bigcup_{i=1}^{k} \left(\{(\mathcal{S}_i, \mathcal{A})\} \cup \bigcup_{j=1}^{t} \{(P_i, \tilde{P}_{j,i})\} \right) \quad .$$

Similarly as above we write $\mathcal{Z}(\mathcal{H}(\mathcal{A}^{(\mathcal{S}_1,\ldots,\mathcal{S}_t)}, \pi^{(\tilde{\pi}_1^{\mathcal{F}_1},\ldots,\tilde{\pi}_t^{\mathcal{F}_t},\pi_{t+1},\ldots,\pi_s)}))$ to denote the ITM-graph (V'', E'') defined by $V'' = V' \cup \{\mathcal{Z}\}$, and $E'' = E' \cup \{(\mathcal{Z}, \mathcal{A})\} \cup \bigcup_{i=1}^{k} \{(\mathcal{Z}, P_i)\}$.

Note that all real subprotocols π_j, for $j = t+1, \ldots, s$, above may be integrated into the original real protocol π. Thus a hybrid model with no ideal functionalities involved is equivalent to a real model, except that it may use several communication models. One may either augment the definition of a real model to allow this, or only use communication models with the property that two communication models can be simulated using a single communication model. The real communication model above, Definition 14, has this property.

The concept of hybrid models is generalized in the natural way, e.g. we write $\mathcal{H}(\mathcal{A}^{(A_1^{\mathcal{S}_{11}}, A_2^{\mathcal{S}_{21}})}, \pi^{(\pi_1^{\tilde{\pi}_{11}^{\mathcal{F}}}, \pi_2^{\tilde{\pi}_{21}^{\mathcal{F}}})})$ for a hybrid model for a real protocol that executes two subprotocols, where each subprotocol has access to a separate copy of the ideal functionality $\mathcal{F}$. Some care needs to be taken when defining the adversary for such models. If an adversary corrupts a participant, it automatically corrupts all its sub-participants that are involved in subprotocols[6].

We also write $\mathcal{Z}_z$ to denote that $\mathcal{Z}$ takes auxiliary input z, and always assume that in any execution of such an ITM-graph, $\mathcal{Z}$ is activated first.

The following definition is somewhat sloppy in that we have not defined the notion of $\mathcal{M}$-adversaries rigorously. We trust the reader to resolve this, and assume that $\mathcal{M}$ is some class of adversaries.

Definition 19 (Secure Realization). *Let* $\mathcal{F}$ *be an ideal functionality. Let* $\pi = (P_1, \ldots, P_k)$*, and let* $\tilde{\pi}_j = (\tilde{P}_{j,i}, \ldots, \tilde{P}_{j,i})$ *be the corresponding dummy participants for* $\mathcal{F}_j$*, for* $j = 1, \ldots, t$*.*

Then $\pi^{(\tilde{\pi}_1^{\mathcal{F}_1},\ldots,\tilde{\pi}_t^{\mathcal{F}_t})}$ *realizes* $\tilde{\pi}^{\mathcal{F}}$ *securely with regards to* $\mathcal{M}$*-adversaries if for all* $\mathcal{M}$*-adversaries* $\mathcal{A}^{(S_1,\ldots,S_t)}$ *with auxiliary input* $z = \{z_n\}$*,* $\exists \mathcal{S} \in \mathrm{ITM}$ *such that* $\forall c > 0$*,* $\exists n_0$*, such that* $\forall n > n_0$

$$|\Pr[\mathcal{Z}_z(\mathcal{I}(\mathcal{S}, \tilde{\pi}^{\mathcal{F}})) = 1] - \Pr[\mathcal{Z}_z(\mathcal{H}(\mathcal{A}^{(S_1,\ldots,S_t)}, \pi^{(\tilde{\pi}_1^{\mathcal{F}_1},\ldots,\tilde{\pi}_t^{\mathcal{F}_t})})) = 1]| < \frac{1}{n^c} \quad .$$

Since the dummy participants are of no real importance we also say that π *realizes* $\mathcal{F}$ *in the* $(\mathcal{F}_1, \ldots, \mathcal{F}_t)$*-hybrid model.*

Canetti [8] proves a powerful composition theorem that can handle polynomially many instances of a constant number of ideal functionalities, but we only need the following weaker special case.

[6] The most general definition allows some violations of this rule.

Theorem 4 (Composition Theorem). *Suppose that* $\pi^{(\tilde{\pi}_1^{\mathcal{F}_1},\ldots,\tilde{\pi}_t^{\mathcal{F}_t})}$ *securely realizes* $\tilde{\pi}^{\mathcal{F}}$*, and that* $\pi_i^{(\tilde{\pi}_{i1}^{\mathcal{F}_{i1}},\ldots,\tilde{\pi}_{it_i}^{\mathcal{F}_{it_i}})}$ *securely realizes* $\tilde{\pi}_i^{\mathcal{F}_i}$*, for* $i = 1,\ldots,l$*, with regards to* $\mathcal{M}$*-adversaries.*

Then $\pi^{(\pi_1^{(\tilde{\pi}_{11}^{\mathcal{F}_{11}},\ldots,\tilde{\pi}_{1t_1}^{\mathcal{F}_{1t_1}})},\ldots,\pi_l^{(\tilde{\pi}_{l1}^{\mathcal{F}_{l1}},\ldots,\tilde{\pi}_{lt_l}^{\mathcal{F}_{lt_l}})},\tilde{\pi}_{l+1}^{\mathcal{F}_{l+1}},\ldots,\tilde{\pi}_t^{\mathcal{F}_t})}$ *securely realizes* $\tilde{\pi}^{\mathcal{F}}$ *with regards to* $\mathcal{M}$*-adversaries.*

Proof. A full proof of the more general statement can be found in Canetti [8].

Note that the hybrid protocol can be transformed into a protocol in the $(\mathcal{F}_{11},\ldots,\mathcal{F}_{1t_1},\ldots,\mathcal{F}_{l1},\ldots,\mathcal{F}_{lt_l},\mathcal{F}_{l+1},\ldots,\mathcal{F}_t)$-hybrid model. However, in this hybrid model the underlying real model may use several different communication models.

J Chernoff Bounds

Theorem 5. *Let* $X_1,\ldots,X_N$ *be mutually independent indicator variables and define* $X = \sum_{i=1}^N X_i$*. Then for arbitrary* $\gamma > 0$ *we have:*

$$\Pr[X < \mathrm{Exp}[X] - \gamma N] < e^{-2\gamma^2 N} \quad .$$

Theorem 6. *Let* $X_1,\ldots,X_N$ *be mutually independent indicator variables and define* $X = \sum_{i=1}^N X_i$*. Then for arbitrary* $\gamma > 0$ *we have:*

$$\Pr[X < (1-\gamma)\mathrm{Exp}[X]] < e^{-\frac{\gamma^2 \mathrm{Exp}[X]}{2}} \quad .$$

K The Flaw In the Original Proof of Soundness

We remark that the error was discovered by Bao and Peng independently of the author and reported in an unpublished manuscript. The error is on page 34, point 1(a) in the paragraph: "Each ρ_i Equals a Prime $p_{j,\pi_j(i)}$ Up to Sign for Some π_j" in ePrint Report 2005/137, September 7, 2005.

It is claimed that $c_j\rho_j \in [-2^{K_3+K_4+K_5}+1, 2^{K_3+K_4+K_5}-1]$. This would lead to $\rho_i \in [-2^{K_3+K_5}+1, 2^{K_3+K_5}-1]$, showing that ρ_i has less than $2K_3$ bits, since c_j is larger than 2^{K_4-1} and K_5 is chosen strictly smaller than K_3.

Unfortunately this argument is incorrect. The only guarantee we have on the bit size of ρ_i is what follows from Equation (23) on page 33, i.e., it belongs to $[-2^{K_3+K_4+K_5+1}+1, 2^{K_3+K_4+K_5+1}-1]$. The latter follows, since $d_{j+N,i}$ and $d_{j,i}$ are both positive and belongs to $[-2^{K_3+K_4+K_5}+1, 2^{K_3+K_4+K_5}-1]$ (as in most protocols in the literature, it is assumed that the verifier checks its input).

Thus, on page 34, point 1(a) we only know that each ρ_i is a product of at most two primes and possibly -1, from which we can not exclude that $\rho_i = \pm 1$ for some i.

K.1 Correction By Changing the Size of Parameters

The obvious way to correct the problem is to choose the security parameters such that $3K_3 > K_4 + K_5$. Then the original proof would go through after noting that this implies that no ρ_i can have more than one prime factor. However, this would make the bit size of the primes in the prime vectors necessarily large.

K.2 Correction Used In a Nutshell

Recall that the prover shows that $\prod_{i=1} \rho_i = \prod_{i=1}^{N} p_i$ over the integers and that this implies that either $\rho_i = p_{\pi(i)}$ for some permutation, or that $\rho_i = \pm 1$ for some i. We need only argue that the latter event happens with negligible probability.

Suppose to the contrary that the event $\rho_i = \pm 1$ happens with non-negligible probability. The prover has committed itself to a (possibly non-permutation) matrix (ρ'_{ij}), which can be extracted using batch-proof techniques and rewinding, see page 37. By assumption we may rewind the prover and extract with non-negligible probability another accepting execution for an additional random vector $(p_1, \ldots, p_N)$ such that the corresponding $\rho_1, \ldots, \rho_N$ satisfies $\rho_i = \pm 1$ for some i. On the other hand, unless the discrete logarithm assumption is false, it must hold that

$$(p_1, \ldots, p_N) \times (\rho'_{ij}) = (\rho_1, \ldots, \rho_N)$$

where $\times$ denotes matrix multiplication in $\mathbb{Z}_q^N$. More precisely, if the above equality does not hold, a non-trivial representation of $1 \in G_q$ can be extracted which is infeasible under the discrete logarithm assumption. Finally, for an *arbitrary* fixed matrix (ρ'_{ij}) and a random vector $(p_1, \ldots, p_N)$ the probability that a vector $(\rho_1, \ldots, \rho_N)$ satisfying the above equation also satisfies $\rho_i = \pm 1$ for any i is negligible. Thus, if the probability that $\rho_i = \pm 1$ is non-negligible then the discrete logarithm assumption is false, i.e., we may conclude that $\rho_i = p_{\pi(i)}$ for some permutation π.

The remainder of the proof proceeds as before to show both that (ρ'_{ij}) is a permutation matrix and that the same permutation π_j is used in all the extracted accepting interactions.

Details are found in the current paper. The changes introduced due to the flaw in the original version of this work are commented in footnotes 1 (page 12), 3 (page 33), 2 (page 35), and 4 (page 49).

Interestingly, the corrected analysis *avoids the need for any implicit range proofs*, i.e., it is not even necessary that each ρ_i is a product of at most two primes to start with which gives more freedom in choosing the security parameters.