Securing Abe's Mix-net Against Malicious Verifiers via Witness Indistinguishability

Elette Boyle* IDC Herzliya
Saleet Klein† MIT
Alon Rosen‡ IDC Herzliya
Gil Segev§ Hebrew University

Abstract

We show that the simple and appealing unconditionally sound mix-net due to Abe (Asiacrypt'99) can be augmented to further guarantee anonymity against malicious verifiers.

As our main contribution, we demonstrate how anonymity can be attained, even if most sub-protocols of a mix-net are merely witness indistinguishable (WI). We instantiate our framework with two variants of Abe's mix-net. In the first variant, ElGamal ciphertexts are replaced by an alternative, yet comparably efficient, "lossy" encryption scheme. In the second variant, new "dummy" vote ciphertexts are injected prior to the mixing process, and then removed.

Our techniques center on new methods to introduce additional witnesses to the sub-protocols within the proof of security. This, in turn, enables us to leverage the WI guarantees against malicious verifiers. In our first instantiation, these witnesses follow somewhat naturally from the lossiness of the encryption scheme, whereas in our second instantiation they follow from leveraging combinatorial properties of the Beneš-network. These approaches may be of independent interest.

Finally, we demonstrate cases in Abe's original mix-net (without modification) where only one witness exists, such that if the WI proof leaks information on the (single) witness in these cases, then the system will not be anonymous against malicious verifiers.

1 Introduction

A mix-net, introduced by Chaum [Cha81], is a means to provide anonymity for a set of users. It has become a central tool for electronic voting, in which each voter submits an encrypted vote and the mix-net outputs the same set of votes in randomized order. Mix-nets have also found applications

*eboyle@alum.mit.edu. Supported by ISF grant 1861/16, AFOSR Award FA9550-17-1-0069, and ERC starting grant 307952.

†saleet@csail.mit.edu. This work was done in part while visiting at the IDC Herzliya FACT Center, supported by ERC starting grant 307952. Additionally supported by an Akamai Presidential Fellowship, by the NSF MACS - CNS-1413920, by the DARPA IBM - W911NF-15-C-0236, by the SIMONS Investigator award Agreement Dated 6-5-12, and by the MISTI MIT-Israel Seed Fund.

‡alon.rosen@idc.ac.il. Supported by ISF grant no. 1255/12, NSF-BSF Cyber Security and Privacy grant no. 2014/632, by the ERC under the EU's Seventh Framework Programme (FP/2007-2013) ERC Grant Agreement n. 307952, and by the MISTI MIT-Israel Seed Fund.

§segev@cs.huji.ac.il. Supported by the European Union's 7th Framework Program (FP7) via a Marie Curie Career Integration Grant (Grant No. 618094), by the European Union's Horizon 2020 Framework Program (H2020) via an ERC Grant (Grant No. 714253), by the Israel Science Foundation (Grant No. 483/13), by the Israeli Centers of Research Excellence (I-CORE) Program (Center No. 4/11), by the US-Israel Binational Science Foundation (Grant No. 2014/632), and by a Google Faculty Research Award.

in other areas, including anonymous web browsing [GGMM97], payment systems [JM98], and as a building block for secure multi-party computation [JJ00].

In some cases, for instance for electronic voting, the mix-net is required to be *verifiable*. That is, the mixing process should be accompanied by a proof that does not violate anonymity (traditionally, zero-knowledge), and at the same time convinces that the set of votes (alternatively, the vote tally) was preserved following the mixing process (soundness). Much work has been devoted to optimizing the running times of protocols, resulting in highly efficient solutions (e.g., [Nef01, GI08, Wik09, TW10, BG12]). At the same time, the strive for efficiency has almost always required assuming that verifying parties act honestly.

While there exist relatively simple methods for enforcing honest verifier behavior, very often the verifier ends up being replaced with some concrete "challenge-generating" hash function that is modeled as a random oracle. This transformation (known as the *Fiat-Shamir transform* [FS86]) only provides heuristic guarantees for anonymity, as any concrete instantiation of a hash function is far from behaving randomly (and consequently is far from emulating the behavior of an honest verifier). Moreover, there is indication that when applied to *computationally sound* protocols (which include all known mix-nets with sublinear verification) it may result in loss of soundness [GK03].

The primary reason known efficient solutions require assuming honest verifiers is that they achieve anonymity by requiring underlying protocols to be *zero-knowledge* (ZK). In some sense this is an overkill, since it may be possible to guarantee anonymity of the overall system even if some of its building blocks do not satisfy such a strong security notion. One prime example is given by Feige and Shamir, who demonstrated how to construct 4-round ZK arguments for NP by invoking sub-protocols that satisfy the notion of *witness indistinguishability* (WI) [FS90]. In contrast to ZK, WI protocols are only required to hide which of the (possibly many) NP-witnesses is used in the protocol execution. This weaker notion gives rise to very simple and consequently efficient constructions, secure even against *malicious* verifiers and sound even against *computationally unbounded* provers.

1.1 This Work

The goal of this work is to explore the possibility of constructing a simple mix-net that is secure against malicious verifiers and in addition is unconditionally sound. This would in particular mean that when applying the Fiat-Shamir transform to the proofs in the mix-net, anonymity would provably be guaranteed for *any choice* of a hash function. While soundness would still be heuristic, unconditional soundness of the protocols makes them less susceptible to theoretical doubts cast on the Fiat-Shamir transform in the case of certain computationally sound protocols [GK03].

Towards this end, we aim for a relaxed indistinguishability-based notion of anonymity, which is weaker than zero-knowledge and yet guarantees the privacy of voters in the system. We demonstrate how indistinguishability-based anonymity of an entire mix-net system can be attained, even if most of the underlying sub-protocols are merely WI. At the core of our analysis are new techniques for guaranteeing the existence of multiple witnesses in NP-verification relations upon which the soundness of mix-nets is based.

We instantiate our ideas with a very simple and appealing Beneš-network based construction due to Abe [Abe99, AH01]. While this construction does not match the sublinear verification efficiency of later mix-nets in the literature (verification time is quasi-linear in the number of voters), it does enjoy a number of desirable features, most notably high parallelizability. In addition, proving and verifying consists of invoking standard and widely used proofs of knowledge, making the mix-net easy to understand and implement.

Abe's mix-net was originally shown to be anonymous assuming honest verifiers, and specifically based on the honest-verifier ZK property of the underlying proofs of knowledge. In the case of a *malicious* verifier, these sub-protocols are known only to be witness indistinguishable; alas, this guarantees nothing in cases where there is a single witness. Moreover, (as we show) in Abe's mix-net, cases in which only one witness exists *cannot be ruled out*, and if indeed leakage on the single witness occurs in these situations we demonstrate that the system is *not anonymous*.

1.2 Our Results

We propose two different methods for modifying Abe's original proposal that result in a verifiable mix-net anonymous against malicious verifiers and sound against computationally unbounded provers. Both methods require only minor changes to Abe's original protocol:

Lossy Abe mix-net: This encryption is identical to Abe's original proposal, with the only difference being that plain ElGamal encryption is replaced with an alternative, yet comparably efficient, encryption scheme with the property that public-keys can be sampled using a "lossy" mode (this mode is only invoked in the analysis). When sampled with lossy public-keys, encrypted ciphertexts do not carry any information about the plaintext. (The same property can also be satisfied by the Goldwasser-Micali QR-based, Paillier's DCR-based and Regev's LWE-based, encryption schemes.)

Injected Abe mix-net: This method consists of running the original Abe mix-net with additional dummy ciphertexts that are injected to the system for the purpose of proving D-WI without having to modify and/or assume anything about the encryption scheme in use (beyond it being re-randomizable). The analysis of this construction relies on combinatorial properties of the Beneš-network, and may turn out to be relevant elsewhere.

These modifications correspond to two approaches for introducing additional witnesses to the sub-protocols of the mix-net verification: In the first, the extra witnesses follow from the lossiness of the encryption scheme, and in the second they follow by leveraging combinatorial properties of the Beneš-network.

In both cases, we show that the entire transcript of the mix-net system satisfies the following natural anonymity property (in the style of [NSK04]): *for any choice of votes and any two permutations on the votes, the corresponding views of an adversary are computationally indistinguishable.*

We allow the adversary to control all but one of the mix-servers, an arbitrary subset of the voters, subset of the decryption servers, and the verifier. If the adversary controls a subset of the voters, then our definition quantifies over any two permutations that are consistent on the votes that it controls. Note that this anonymity notion completely hides information about which honest voter placed which vote from the collective set of honest votes (which is necessarily revealed by the shuffled output).

Theorem. *The Lossy and Injected Abe mix-nets are anonymous against malicious verifiers.*

Our result assumes the availability of a non-malleable (more precisely, plaintext aware) encryption scheme, under which the ciphertexts are encrypted, and an efficient secure (simulatable) multi-party protocol for threshold decryption of the ciphertexts. The latter building blocks can be

constructed in an efficient manner, even if participating parties are malicious, and moreover are routinely assumed available in the cryptographic voting literature.

In a precise strong sense, the modifications introduced in the lossy and injected versions of Abe's mix-net are necessary for achieving anonymity in the case of a malicious verifier. As discussed in Appendix B, whenever only a single witness exist, WI guarantees nothing, and in fact such cases do occur in the setting of Abe's mix-net. As an additional contribution we show an attack on the anonymity of the mix-net, if leakage on the (single) witness indeed occurs.

1.3 Technical Overview

In what follows, we provide further background on verifiable mix-nets and Abe's mix-net construction, and then describe the main technical ideas behind our results.

1.3.1 Verifiable mix-nets

Ideally, a mix-net is a protocol that completely breaks the link between a user and the vote she has submitted. This remains true even if a subset of users share their votes with each other with the purpose of "de-anonymizing" votes of users outside their coalition.

In principle, if one is only interested in the tally, then a simple way to protect anonymity of individual voters would be to output the tally $\sum_i v_i$. However, specifically designing a protocol to meet this functionality limits its applicability in case one is interested in alternative tallying mechanisms. Further, such tallying solutions relying on homomorphic encryption either limit the message size or require the use of relatively complicated zero-knowledge procedures for proving the submitted vote encryptions are well formatted.

Mix-net phases. The operating assumption underlying most known mix-net constructions is that some vote-encryption mechanism is in place, resulting in a list $c_1, \ldots, c_n$ of ciphertexts where $c_i = \mathsf{Enc}_{\mathsf{pk}}(v_i; r_i)$ is an encryption of v_i with randomness r_i, under a public key pk that corresponds to a certain polling station. The output of the mix-net is a shuffled list of plaintexts $v_{\sigma(1)}, \ldots, v_{\sigma(n)}$, and we want a mix-net that hides σ even if malicious entities were involved in the mixing phase, the input phase, and the verification phase.

The public key pk is jointly generated and certified in a distributed manner by a set of trustees, so that no individual entity (or even any sufficiently small coalition of entities) is able to decrypt its corresponding ciphertexts. The assumption is that a large subset of the trustees acts as prescribed by the set-up protocol. We note that such an assumption is standard in the literature, and it does not necessitate the generation of a common reference string, at least not in its most general form.

Given such a setup, most known verifiable mix-net constructions can be conceptually decomposed to the following three stages:

Submit Ciphertexts: Each of the n users publishes their own ciphertext c_i on an authenticated bulletin board. For simplicity it is convenient to assume that the encryption is "non-malleable" (in fact, plaintext aware), which guarantees that voters cannot make their own vote depend on others'.

Verifiably Mix: Ciphertexts $c_1, \ldots, c_n$ are:

- re-randomized (i.e. $\mathsf{Enc}_{\mathsf{pk}}(v_i; r_i)$ is mapped to $\mathsf{Enc}_{\mathsf{pk}}(v_i; s_i)$ for random and independent s_i) and then
- randomly shuffled to obtain ciphertexts $c'_{\pi(i)} = \mathsf{Enc}_{\mathsf{pk}}(v_i; s_i)$, where π is randomly sampled from S_n.

In addition, the mixing party provides a proof that the set of plaintexts underlying the output ciphertexts $c'_1, \ldots, c'_n$ equals the original set of submitted votes underlying $c_1, \ldots, c_n$.

Decrypt: The ciphertexts $c'_1, \ldots, c'_n$ are collectively decrypted by means of a secure distributed protocol.

In terms of complexity, the Submit Ciphertexts and Decrypt stages can be implemented in time $O(n)$. Moreover, by using lightweight protocols for threshold decryption, the Decrypt phase can be implemented in a zero-knowledge fashion without paying much penalty in terms of efficiency. In light of this, much of the literature (including the present work) focuses on optimizing the efficiency of the Verifiably Mix stage.

A naïve implementation would require work proportional to $O(n^2)$ (by proving consistency of individual input-output ciphertext pairs). Remarkably, it has been shown how to achieve perfect ZK with verification time as little as $o(n)$ (see [Nef01, GI08, Wik09, Gro10, TW10, BG12] to name a few). As we mentioned above, in many cases this comes at the price of the assumption that the prover is computationally bounded and that verification is performed as prescribed.

1.3.2 Abe's mix-net

Abe presented [Abe99, AH01] a simple mix-net construction which performs the Verifiably Mix stage on user ciphertexts via a sequence of pairwise ciphertext rerandomize-and-swap operations, as dictated by a *Beneš permutation network*. A d-dimensional Beneš network is a "butterfly" switching network on $n = 2^d$ inputs, consisting of $(2d - 1)$ levels of $\frac{n}{2}$ switch gates (see Section 2.3). Given any permutation $\pi \in S_n$, this permutation can be implemented via some (efficiently determined) choice of the control bits for each of the switch gates, where 0 at a gate indicates its input are output in order and 1 indicates its inputs are swapped.

In Abe's mix-net construction, the mixing entity samples a random permutation $\pi \leftarrow S_n$, and identifies a corresponding choice of Beneš control bits. Then, implementing and proving the validity of the overall n-input mix reduces to the same task on each of the $O(n \log n)$ individual switch gates in its Beneš representation. Namely, the overall proof is simply a collection of independent proofs that an individual rerandomize-and-switch gate operation preserved the plaintext values underlying its input ciphertexts. (We refer the reader to Section 2.4 for a complete description of Abe's mix-net construction.)

For many common encryption schemes, this simple statement structure yields lightweight proofs of knowledge. For example, for ElGamal encryption (as considered by Abe), such a proof can be attained with 3 rounds by combining the Chaum-Pedersen protocol [CP92], which proves the equality of two discrete logarithms, with the protocol used in [CDS94], which proves two statements connected by OR, overall costing about four times as much computation as a single Chaum-Pedersen protocol execution.

However, lightweight protocols of this kind (inherently) provide only witness indistinguishability guarantees and/or *honest verifier* zero knowledge. Because of this, the mix-net of Abe was only proved to possess these properties as well.

1.3.3 Techniques and Ideas

To prove anonymity of our constructions, we must prove for any vector of votes $\vec{v} = (v_1, \ldots, v_n)$, and any permutation π of the honest parties, that the view of a (possibly *malicious*) verifier in the mix-net proof of correctness executed on votes $(v_1, \ldots, v_n)$ is indistinguishable from the analogous view on initial votes $(v_{\pi(1)}, \ldots, v_{\pi(n)})$. That is, intuitively, the verifier cannot distinguish which of the honest votes came from which honest party.

The semantic security of the encryption scheme directly allows us to "swap out" the starting honest-party vote encryptions themselves. So the core task is showing that interaction with an honest mix-server proving proper execution of random permutation σ on encryptions of $(v_1, \ldots, v_n)$ is indistinguishable from an analogous proof executing $\sigma \circ \pi^{-1}$ on encryptions of $(v_{\pi(1)}, \ldots, v_{\pi(n)})$. The main difficulty in doing so arises for adversaries who have partial control of the votes: specifically, when the adversary controls a subset $\{v_i\}_{i \in A}$ of votes for some arbitrary $A \subseteq [n]$ of his choice.

Recall that Abe's construction is composed of a collection of underlying proofs of knowledge, where each individual sub-protocol is WI. Consider the proof for a single switch gate. To leverage the WI property, we must arrive to a state where the corresponding gate-validity statement $((c_1, c_2), (c'_1, c'_2))$ has at least two witnesses. This aligns precisely with the case in which the two input ciphertexts (c_1, c_2) of the gate have the *same underlying plaintext*. In such case, one could have reached the output ciphertexts (c'_1, c'_2) either by simply rerandomizing directly, or by swapping first and then rerandomizing (with different randomness); conversely, if the input plaintexts differ then by the correctness of the encryption scheme there is a unique witness.

Now, suppose we are in the case of a gate where both input ciphertexts c_1, c_2 correspond to encrypted votes of *honest* users. Then although the underlying votes of the two users may disagree, by relying on the semantic security of the encryption scheme, we can argue that the adversary cannot distinguish this state from the one in which the votes *do* agree. Once in this modified version of the world, we can invoke the WI guarantee to argue that the proof hides the identity of the swap bit. A similar approach can further take care of the situation where a single input ciphertext to a gate is controlled by the adversary (by changing the honest ciphertext to agree with the adversary's fixed vote).

What poses an issue is when *both* input ciphertexts to a gate are under adversarial control. The adversary can then force the gate to have a single witness, by choosing different plaintext votes. (Note we cannot hope to invoke semantic security arguments as above, as the adversary generates the ciphertexts himself). In such a case, for all that is known, the underlying protocol may very well leak the control bit of this gate. Interestingly, we demonstrate that such leakage, while directly regarding only corrupt-party ciphertexts, would be fatal to anonymity of *honest* parties in Abe's mix-net (see Appendix B).

We address this issue via two alternative proposed modifications to Abe's protocol.

Using a lossy encryption scheme. In the first variant, we instantiate the encryption scheme within Abe's mix-net with a DDH-based *lossy* encryption scheme that admits a similar underlying WI gate-consistency proof. A lossy encryption scheme has the property that standard key generation is indistinguishable from a "lossy" version, such that encryption under a lossy key pk completely loses all information about the message. In particular, for a lossy pk, for *any* pair of ciphertexts c, c' (not necessarily formed by encrypting the same message), there exists a choice of re-randomization that takes c to c'. This means for a lossy key that for *any* switch gate tuple (c_1, c_2, c'_1, c'_2), there necessarily exist two witnesses.

The proof of anonymity then follows from four simple steps. First, the public key is replaced by a lossy version. Then, once we are under a lossy pk, we can directly use the WI of the underlying gate protocols to switch (gate by gate) from a Beneš representation of a starting permutation π to the representation of any other permutation σ. Additionally, by the guaranteed hiding, we can switch the plaintexts of honest users' votes to an arbitrary shuffle amongst themselves. Once we attain the desired permutation and plaintext settings, we simply return back to a standard (non-lossy) pk.

Injecting and removing "dummy" votes. In the second variant, we consider an arbitrary rerandomizable public-key encryption scheme (e.g., standard ElGamal), and instead modify the Abe mix protocol at a higher level. Interestingly, the design approach leverages the combinatorial structure of the Beneš network, without modifying the underlying building block proofs of knowledge (for which it is not known how to prove an analogous property).

The new mixing procedure begins by generating and injecting n "dummy" votes (i.e., encryptions of a fixed non-vote message $\perp$) into the list of n real encrypted votes. Abe's mix phase is performed (without modification) on the combined list of $2n$ ciphertexts (injecting the $\perp$ ciphertexts into the even-indexed positions). Then, Abe's Decrypt protocol is performed on *all* $2n$ resulting ciphertexts, and the $\perp$ plaintexts are identified and removed. Verification consists of Abe's standard verification, plus a process for verifying that $\perp$ ciphertexts were properly injected and removed in each mix step.

We remark that this modification of injecting non-adversarial ciphertexts into the even-indexed positions does not directly preclude gates within the Beneš execution whose input ciphertexts are both under adversarial control; indeed, this remains quite likely to occur in many locations within later levels of the Beneš network. However, leveraging the combinatorial Beneš structure, we prove that the power we gain by ensuring the *first-level* gates do not have this problem, is sufficient to hide all control bits used within the Beneš network.

Our proof takes an inductive approach, on the dimension d (i.e., number of users $n = 2^d$) of the Beneš network. Ultimately, we design a carefully ordered sequence of hybrids which enables us to step from honest input votes $\vec{u}_{\mathsf{honest}}$ and permutation $\pi \in S_n$ to an arbitrary other choice $\vec{u}'_{\mathsf{honest}}, \sigma$. In essence, for each gate g in the Beneš network whose control bit we would like to flip, we: (1) switch the control bits of relevant first-level gates to ensure at least one non-adversarial ciphertext c_i becomes directed to gate g; (2) rely on semantic security to change the plaintext underlying c_i to agree with its neighboring ciphertext c_j into g; and then (3) use the WI to flip the control bit of gate g, now that we have forced the existence of 2 witnesses. This procedure is performed on gates in a particular order to ensure progress is made in each step, while leaving sufficient flexibility to enable that the step (1) redirection can be performed.

2 Preliminaries

2.1 Standard Cryptographic Definitions

We denote by $[n]$ the set $\{1, \ldots, n\}$, and by S_n the set of all permutations over $[n]$. A *witness relation* for a language $\mathsf{L} \in \mathbf{NP}$ is a binary relation R_L such that $\mathsf{L} = \{x \mid \exists w \text{ s.t. } (x, w) \in \mathsf{R}_\mathsf{L}\}$.

Definition 1 (Interactive Proof System)**.** A pair of interactive machines $(\mathcal{P}, \mathcal{V})$ is called an *interactive proof system* for a language L if machine V is polynomial-time and the following two conditions hold:

- Completeness: For every $x \in \mathsf{L}$, there exists a string w such that for every $z \in \{0,1\}^*$,

$$\Pr[\langle \mathcal{P}(w), \mathcal{V}(z) \rangle(x) = 1] \geq 1 - \epsilon$$

- Soundness: For every $x \notin \mathsf{L}$, every interactive machine $\mathcal{P}^*$, and every $w, z \in \{0,1\}^*$,

$$\Pr[\langle \mathcal{P}(w), \mathcal{V}(z) \rangle(x) = 1] \leq \epsilon$$

Definition 2 (Σ-protocol, Special soundness)**.** A Σ-*protocol* is a 3-message interactive proof system with the following structure:

1. P sends a message a.
2. V sends a random t-bit string e.
3. P sends a reply z, and V decides to accept or reject based on x, a, e, z.

A Σ-protocol for relation R is said to satisfy *special soundness* if from any x and any pair of accepting conversations $(a, e, z), (a, e, z)$ on input x, where $e \neq e'$, one can efficiently compute a witness w such that $(x, w) \in \mathsf{R}$.

2.1.1 Public-Key Encryption

Definition 3 (Rerandomizable PKE)**.** A *Rerandomizable Public-Key Encryption* (PKE) scheme over a message space M is a tuple of PPT algorithms $\mathcal{E} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{ReRand})$, where $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ is a standard semantically secure PKE scheme, and

- $\mathsf{ReRand}_{\mathsf{pk}}(c)$ takes as input the public key pk and a ciphertext c and outputs a new ciphertext c_{out}.
- For every valid pk, every message $m \in M$, and every ciphertext $c = \mathsf{Enc}_{\mathsf{pk}}(m; r)$ (for arbitrary r), it holds that $\{c' \leftarrow \mathsf{ReRand}_{\mathsf{pk}}(c)\} \equiv \{c' \leftarrow \mathsf{Enc}_{\mathsf{pk}}(m)\}$.

We assume there is an efficient procedure given a key pair $(\mathsf{pk}, \mathsf{sk})$ for identifying whether sk is a valid secret key for pk, denoted $(\mathsf{pk}, \mathsf{sk}) \in \mathsf{KeyGen}(1^\lambda)$.

Definition 4 (ElGamal Encryption)**.** The *ElGamal Encryption Scheme* for a fixed (constant-size) message space $M = \{0, \ldots, \ell\}$ is defined as follows:

- $\mathsf{KeyGen}(1^\lambda)$: Generate the description of a cyclic group $\mathbb{G}$ of prime order q (with $\log_2 q \geq \lambda$) and generator g.

  Sample a random secret key $s \leftarrow [q-1]$ and compute $h = g^s$. Output $\mathsf{pk} = (\mathbb{G}, q, g, h)$ and $\mathsf{sk} = s$.

- $\mathsf{Enc}_{\mathsf{pk}}(m)$: Choose a random $r \leftarrow [q-1]$, and output the ciphertext $c = (g^r, h^r \cdot g^m)$.

- $\mathsf{Dec}_{\mathsf{sk}}(c = (a, b))$: Compute $u := b \cdot a^{-s}$, and output $m \in \{0, \ldots, \ell\}$ for which $u = g^m$ (recall $\ell \in O(1)$, so this step can be performed efficiently).
- $\mathsf{ReRand}_{\mathsf{pk}}(c = (a, b))$: Choose a random $r \leftarrow [q-1]$, and output $c_{out} = (a \cdot g^r, b \cdot h^r)$.

Definition 5 (Lossy PKE [PVW07])**.** A *lossy public-key encryption scheme* is a tuple of PPT algorithms $(\mathsf{KeyGen}, \mathsf{KeyGen}_{\mathsf{loss}}, \mathsf{Enc}, \mathsf{Dec})$ such that $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ is a semantically secure PKE scheme, and where $\mathsf{KeyGen}_{\mathsf{loss}}$ takes as input as security parameter and outputs a public key pk satisfying the following properties:

1. The distributions of public keys $\{\mathsf{pk} : (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda)\} \stackrel{c}{\cong} \{\mathsf{pk} : \mathsf{pk} \leftarrow \mathsf{KeyGen}_{\mathsf{loss}}(1^\lambda)\}$ are computationally indistinguishable.
2. Encryption under a "lossy" public key is information theoretically hiding. That is, with overwhelming probability in $\mathsf{pk} \leftarrow \mathsf{KeyGen}_{\mathsf{loss}}(1^\lambda)$, it holds for every pair $m_0, m_1 \in M$ that $\mathsf{Enc}_{\mathsf{pk}}(m_0) \equiv \mathsf{Enc}_{\mathsf{pk}}(m_1)$.

2.2 Mix-Net Definitions

Definition 6 (Mix-Net System)**.** An *n-user m-server Mix-Net System* with respect to a re-randomizable encryption scheme $\mathcal{E} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{ReRand})$ over message space M, has four phases: Setup, $\mathsf{SubmitCipher}$, $\mathsf{VrfblyMix}$, $\mathsf{Decrypt}$, with the following syntax.

- $\mathsf{Setup}(1^\lambda)$ is a distributed protocol that outputs an encryption public key pk, and m shares $\mathsf{sk}_1, \ldots \mathsf{sk}_m$ of the secret key sk to the m servers.
- $\mathsf{SubmitCipher}(\mathsf{pk}, u_i; r_i)$ is an algorithm run by each individual voter that takes as input a public key pk, a vote $u_i \in \{0, 1\}$, and randomness r_i. The output is a ciphertext $c_i = \mathsf{Enc}_{\mathsf{pk}}(v_i; r_i)$
- $\mathsf{VrfblyMix}\left(\mathsf{pk}, \vec{c}^0; (\mathsf{rnd}_j)_{j \in [m]}\right)$ is a sequential algorithm with m iterations that outputs a list of ciphertexts $\vec{c'} = \vec{c}^m$, where for each $j \in [m]$, the jth iteration $\mathsf{Mix}_j(\mathsf{pk}, \vec{c}^{j-1}; \mathsf{rnd}_j)$ is run by the server M_j.

  Mix_j is an algorithm that takes as input a public key pk and a list of n ciphertexts $\vec{c}^{j-1} = (c_1^{j-1}, \ldots, c_n^{j-1})$ under the public key pk. The server outputs a vector of ciphertexts $\vec{c}^j$ (permuted and rerandomized in some fashion).
- $\mathsf{Decrypt}(\vec{c'}, (\mathsf{sk}_i)_{i \in [m]})$ is a distributed protocol executed by the m servers, where each server M_i has input a share sk_i of a secret key sk, and the mth server M_m additionally holds a list of n ciphertexts $\vec{c'}$. As a result of the protocol, each server receives an output list of plaintexts $\vec{v}$ such that $\forall i \in [n]$, $v_i = \mathsf{Dec}_{\mathsf{sk}}(c'_i)$.

The mix-net is *correct* if for every choice of votes $(u_1, \ldots, u_n) \in \{0, 1\}^n$,

$$\Pr\left[\exists \pi \in S_n : u_i = v_{\pi(i)} \;\middle|\; \begin{array}{l} (\mathsf{pk}, (\mathsf{sk}_1, \ldots, \mathsf{sk}_m)) \leftarrow \mathsf{Setup}(1^\lambda); \\ c_i^0 \leftarrow \mathsf{SubmitCipher}(\mathsf{pk}, u_i)\ \forall i \in [n]; \\ \vec{c'} \leftarrow (\mathsf{pk}, \vec{c}^0); \\ \vec{v} \leftarrow \mathsf{Decrypt}(\vec{c'}, (\mathsf{sk}_i)_{i \in [m]}) \end{array}\right] = 1.$$

Remark. Note that the servers who execute the permutations via $\mathsf{VrfblyMix}$ and those who decrypt the final permuted votes via $\mathsf{Decrypt}$ need not be the same entities (nor have the same quantity); however, for simplicity, we consider a fixed set of m servers and assume they perform both operations.

We will be interested in mix-net systems which are *verifiable* and *anonymous*.

The verifiability of a mix-net system refers to a means for proving that the resulting list of plaintexts $\vec{v}$ truly is a permuted version of the input plaintexts.

Definition 7 (*Verifiable* Mix-Net System)**.** An n-user m-server mix-net system is said to be *verifiable* if there exists an interactive proof system $(\mathcal{P}, \mathcal{V})$ for correctness of the mix-net: i.e., for the NP language

$$\mathsf{L} = \{(\mathsf{pk}, \vec{c}^{\,0}, \vec{v}) \mid \exists w \text{ s.t. } ((\mathsf{pk}, \vec{c}^{\,0}, \vec{v}), w) \in \mathsf{R}_{\mathsf{L}}\}$$

where

$$\mathsf{R}_{\mathsf{L}} = \left\{ ((\mathsf{pk}, \vec{c}^{\,0}, \vec{v}), w) \;\middle|\; \begin{array}{ll} W(w) = (\pi, \mathsf{sk}): & \pi \in S_n, \\ & (\mathsf{pk}, \mathsf{sk}) \in \mathsf{KeyGen}(1^\lambda); \\ & \forall i \in [n], \mathsf{Dec}_{\mathsf{sk}}(c_i^0) = v_{\pi(i)} \end{array} \right\}. \tag{1}$$

and W is a deterministic polynomial time algorithm that takes as input a witness w and outputs a permutation π and secret key sk.

This notion of verifiability is sometimes referred to in the literature as *universal* verifiability, where verification does not require user-specific secrets.

In practice, the interactive proof system $(\mathcal{P}, \mathcal{V})$ typically decomposes into a collection of independent sub-proofs, each executed by an individual mix-server M_j, using its own mixing randomness as witness. This will be the case for all mix-net systems considered in the present work.

We discuss *anonymity* of a mix-net in Section 3.

2.3 Beneš Networks

Definition 8 (f-switch gate)**.** An *f-switch gate* is parameterized by a (possibly randomized) function $f : X \to Y$. The gate takes two inputs $x_0, x_1 \in X$, a control bit $b \in \{0, 1\}$ indicating whether inputs are swapped in the output, and possibly randomness r_0, r_1 (for randomized f). The gate produces two outputs $y_0, y_1 \in Y$, as specified by:

$$y_b = f(x_0; r_0) \quad y_{1-b} = f(x_1; r_1).$$

When f is not explicitly specified, a "switch gate" refers to an f-switch gate where f is the identity function.

Definition 9 (d-Dimensional Beneš Network)**.** For $d \in \mathbb{N}$, the *d-dimensional Beneš Network* is a network of switch gates which connects $N = 2^d$ inputs $(\mathsf{in}_1, \ldots, \mathsf{in}_N)$ to N outputs $(\mathsf{out}_1, \ldots, \mathsf{out}_N)$. The structure of the network is defined recursively.

- The 1-dimensional Beneš Network is a single switch gate, as per Definition 8.
- The d-dimensional Beneš Network consists of three stages (as depicted in Figure 1):

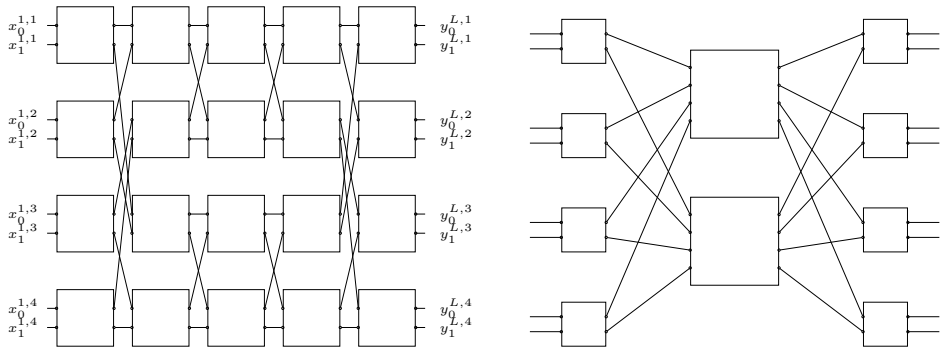

Figure 1: A 3-dimensional Beneš network

1. The ingress stage: The first level in the network consists of $\frac{N}{2}$ switch gates, where for each $i \in [\frac{N}{2}]$, the inputs to the ith switch gate are the corresponding adjacent Beneš network inputs $(\mathsf{in}_{2i-1}, \mathsf{in}_{2i})$.
2. The central stage: Consists of two $(d-1)$-dimensional Beneš Network: $\mathsf{B}_{\mathsf{up}}, \mathsf{B}_{\mathsf{down}}$. The $\frac{N}{2}$ inputs to B_{up} are the *odd* outputs from the first layer; the $\frac{N}{2}$ inputs to $\mathsf{B}_{\mathsf{down}}$ are the *even* outputs from the first layer.
3. The egress stage: The last level in the network consists of $\frac{N}{2}$ switch gates, where for each $i \in [\frac{N}{2}]$ the inputs to the ith switch gate in this gate are the ith outputs from B_{up} and $\mathsf{B}_{\mathsf{down}}$. The outputs of the gates in the level comprise the final outputs of the Beneš network $(\mathsf{out}_1, \ldots, \mathsf{out}_N)$.

Altogether (inductively), the d-dimensional Beneš network consists of $L = 2d-1$ levels of $\frac{N}{2}$ switch gates. The action of the Beneš network is specified by a $L \times \frac{N}{2}$ matrix of control bits $B \in \{0,1\}^{L \times \frac{N}{2}}$, which specifies the action of each switch gate.

Note that for any choice of control bits $B \in \{0,1\}^{L \times \frac{N}{2}}$, the Beneš network induces a permutation on its N inputs: i.e., there exists $\pi \in S_N$ for which $\forall i \in [N], \mathsf{out}_{\pi(i)} = \mathsf{in}_i$. It was shown by Waksman [Wak68] how to efficiently identify an appropriate matrix of control bits B_π for a given permutation π.[1]

Theorem 1 (Beneš Control Bit Sampling)**.** *[Wak68] There exists a deterministic polynomial-time algorithm that, on input an integer $d \in \mathbb{N}$ and description of a permutation $\pi \in S_N$ on $N = 2^d$ elements, outputs a control bit matrix $B \in \{0,1\}^{L \times \frac{N}{2}}$ (where $L = 2d-1$, as in Definition 9) which induce the permutation π.*

We will refer to the output of the Waksman algorithm with a fixed randomness as the *canonical* matrix of control bits for π, and sometimes use the abbreviated notation B_π.

[1]The algorithm presented by Waksman [Wak68] is randomized but satisfies perfect correctness, so that an arbitrary fixed choice of randomness can be used.

2.4 Abe's Mix-net Construction

Construction 1 (Abe's Mix-net system). Abe's Mix-net system is a $n = 2^d$-user m-server mix-net system with respect to the ElGamal encryption scheme $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{ReRand})$ over message space M, that uses a d-dimensional Beneš network with ReRand-switch gates (see definitions in Section 2.3).

$\mathsf{Setup}_{\mathsf{Abe}}, \mathsf{SubmitCipher}_{\mathsf{Abe}}$, and $\mathsf{Decrypt}_{\mathsf{Abe}}$ are the straightforward procedures for threshold ElGamal distributed key generation, encryption, and decryption. $\mathsf{VrfblyMix}_{\mathsf{Abe}}(\mathsf{pk}, \vec{c}^0)$ is a sequential algorithm with m iterations, where each iteration $j \in [m]$ is an execution of $\mathsf{Mix}^{\mathsf{Abe}}$ as given below.

$\mathsf{Mix}^{\mathsf{Abe}}(\mathsf{pk}, \vec{c}^{j-1})$: Let $L = 2d - 1$ (levels of d-dimension Beneš network). Perform the following:

1. Sample a random permutation $\pi_j \leftarrow S_n$.
2. Run $\mathsf{AbeMix}(\mathsf{pk}, \vec{c}^{j-1}, \pi_j)$, as specified here, to obtain $(\vec{c}^j, B_{\pi_j}, \hat{R}_0^j, \hat{R}_1^j)$ where $\vec{c}^j$ is a vector of n ElGamal ciphertexts, B_{π_j} is a matrix of Beneš control bits, and $\hat{R}_0^j, \hat{R}_1^j$ are matrices of randomness used by the ReRand algorithm:
   (a) Sample two matrices of uniform random values $\hat{R}_0^j, \hat{R}_1^j \leftarrow [q-1]^{L \times \frac{n}{2}}$, where each $\hat{R}_0^j[\ell, i]$ and $\hat{R}_1^j[\ell, i]$ will be used to rerandomize the two input ciphertexts at the ith switch gate of level ℓ in the Beneš network.
   (b) Let $B_{\pi_j} \in \{0,1\}^{L \times \frac{n}{2}}$ be the canonical induced matrix of control bits for which the Beneš network implements the permutation π_j on its input elements (see Theorem 1).
   (c) For every $i \in [n]$, let $\mathsf{in}_i = c_i$, and define $\hat{r}_i = \mathsf{AccumRand}(i, B_{\pi_j}, \hat{R}_0^j, \hat{R}_1^j)$ to be the accumulated rerandomization applied to the ith input ciphertext as a result of executing the Beneš network. (That is, the values $\hat{r}_i$ for which $\mathsf{out}_{\pi(i)} = \mathsf{ReRand}_{\mathsf{pk}}(\mathsf{in}_i; \hat{r}_i)$).
   (d) Compute the vector of permuted, rerandomized ciphertexts: $(c_1^j, \ldots, c_n^j)$, where $c_{\pi(i)}^j := \mathsf{ReRand}_{\mathsf{pk}}(c_i; \hat{r}_i)$.
   (e) Output $(\vec{c}^j, B_{\pi_j}, \hat{R}_0^j, \hat{R}_1^j)$.
3. Output the vector of permuted, rerandomized ciphertexts: $\vec{c}^j = (c_1^j, \ldots, c_n^j)$.

Construction 2 (Verification Proof System for Abe Mix-net). Abe's mix-net is verifiable (as per Definition 7), and the corresponding interactive proof system $(\mathcal{P}^{\mathsf{Abe}}, \mathcal{V}^{\mathsf{Abe}})$ can be expressed by the following collection of 3 steps:

1. **Submission of intermediate ciphertext vectors**: For every $j \in [m]$: $\mathcal{P}^{\mathsf{Abe}}$ generates and sends $\mathcal{V}^{\mathsf{Abe}}$ the list of ciphertexts $\vec{c}^j \leftarrow \mathsf{Mix}_j(\mathsf{pk}, \vec{c}^{j-1})$.
2. **Correctness Proof of** $\mathsf{VrfblyMix}$: For every $j \in [m]$: $\mathcal{P}^{\mathsf{Abe}}$ and $\mathcal{V}^{\mathsf{Abe}}$ interact in an execution of an interactive proof system $(\mathcal{P}^{\mathsf{Abe}}_{\mathsf{Mix}}, \mathcal{V}^{\mathsf{Abe}}_{\mathsf{Mix}})$ for the relation

$$\mathsf{R}_{\mathsf{Mix}} = \left\{ ((\mathsf{pk}, \vec{c}^{j-1}, \vec{c}^j), w_j) \;\middle|\; \begin{array}{l} W_{\mathsf{Mix}}(w_j) = (\pi_j, \vec{\hat{r}}^j) : \\ \quad \forall i \in [n],\ c_{\pi_j(i)}^j = \mathsf{ReRand}_{\mathsf{pk}}(c_i^{j-1}; \hat{r}_i^j) \end{array} \right\}, \tag{2}$$

where W_{Mix} is a deterministic polynomial time algorithm that takes as input a witness w_j and outputs a permutation π_j and vector of randomness $\vec{\hat{r}}^j$.

The interactive proof system $(\mathcal{P}^{\mathsf{Abe}}_{\mathsf{Mix}}, \mathcal{V}^{\mathsf{Abe}}_{\mathsf{Mix}})$ with the common input $(\mathsf{pk}, \vec{c}^{j-1}, \vec{c}^{j})$ and witness $(\pi_j, B_{\pi_j}, \hat{R}^j_0, \hat{R}^j_1)$ executes as follows:

(a) $\mathcal{P}^{\mathsf{Abe}}_{\mathsf{Mix}}$ executes the Beneš network on input ciphertext vector $\vec{c}$, with control bits B_π and randomness $\hat{R}_0, \hat{R}_1$. $\mathcal{P}_{\mathsf{Abe}}$ sends to $\mathcal{V}_{\mathsf{Abe}}$ all intermediate ciphertexts (i.e., a vector $\vec{c}^{(\ell)}$ of rerandomized, partially shuffled ciphertexts for the output of each level $\ell \in [L]$ of the Beneš network). Note that $\vec{c}^{(L)}$ is the final vector (i.e., $\vec{c}^{(L)} = \vec{c}^{j}$).

(b) For every level $\ell \in [L]$ and gate $i \in [\frac{n}{2}]$ within the ℓth level of the Beneš network, $\mathcal{P}^{\mathsf{Abe}}_{\mathsf{Mix}}$ and $\mathcal{V}^{\mathsf{Abe}}_{\mathsf{Mix}}$ execute (in parallel) an independent instance of the interactive proof $(\mathcal{P}_{\mathsf{Gate}}, \mathcal{V}_{\mathsf{Gate}})$ (as described in Appendix A) for the following gate-consistency relation:

$$\mathsf{R}_{\mathsf{Gate}} = \left\{ ((\mathsf{pk}, x_0, x_1, y_0, y_1), (b, \hat{r}_0, \hat{r}_1)) \;\middle|\; \begin{array}{l} y_b = \mathsf{ReRand}_{\mathsf{pk}}(x_0, \hat{r}_0) \\ y_{1-b} = \mathsf{ReRand}_{\mathsf{pk}}(x_1, \hat{r}_1) \end{array} \right\}, \tag{3}$$

with common input $(\mathsf{pk}, x_0, x_1, y_0, y_1)$ and witness $(B_\pi[\ell, i], \hat{R}_0[\ell, i], \hat{R}_1[\ell, i])$, where x_0, x_1, y_0, y_1 are the corresponding computed input and output ciphertexts to the gate, and $B_\pi[\ell, i], \hat{R}_0[\ell, i], \hat{R}_1[\ell, i]$ are the corresponding swap control bit and rerandomization values used within the gate.[2].

Note that each execution $j \in [m]$ of $\mathcal{P}^{\mathsf{Abe}}$ can be performed independently by the corresponding mix-server M_j.

3. **Correctness Proof of** Decrypt: $\mathcal{P}$ and $\mathcal{V}$ interact in an execution of an interactive proof system $(\mathcal{P}^{\mathsf{Abe}}_{\mathsf{Dec}}, \mathcal{V}^{\mathsf{Abe}}_{\mathsf{Dec}})$ for the relation

$$\mathsf{R}_{\mathsf{Dec}} = \left\{ ((\mathsf{pk}, \vec{c}^{m}, \vec{v}), \vec{w}) \;\middle|\; \begin{array}{l} W_{\mathsf{ReRand}}(w_1, \ldots, w_m) = \mathsf{sk} : \\ \quad \forall i \in [n]\ \mathsf{Dec}_{\mathsf{sk}}(c^m_i) = v_i \\ \quad (\mathsf{pk}, \mathsf{sk}) \in \mathsf{KeyGen}(1^\lambda) \end{array} \right\}. \tag{4}$$

where W_{Dec} is a deterministic polynomial time algorithm that takes as input a list of witnesses $(w_j)_{j \in [m]}$ and outputs a secret key sk. In $(\mathcal{P}^{\mathsf{Abe}}_{\mathsf{Dec}}, \mathcal{V}^{\mathsf{Abe}}_{\mathsf{Dec}})$ the witness is $\vec{w} = (\mathsf{sk}_1, \ldots, \mathsf{sk}_m)$. We assume this proof system $(\mathcal{P}^{\mathsf{Abe}}_{\mathsf{Dec}}, \mathcal{V}^{\mathsf{Abe}}_{\mathsf{Dec}})$ is *zero knowledge*. For the case of ElGamal, this can be achieved, e.g. via [CGJ+99].

3 Indistinguishability-Based Anonymity of Mix-Nets

We provide in Section 2.2 a complete definition of the syntax, correctness, and verifiability properties of a mix-net system $(\mathsf{Setup}, \mathsf{SubmitCipher}, \mathsf{VrfblyMix}, \mathsf{Decrypt}, (\mathcal{P}, \mathcal{V}))$ (as discussed informally in Section 1.3.1). Here we discuss the property of *anonymity*, most relevant to this work.

A wide range of anonymity notions have been considered within the mix-net literature, ranging from addressing specific anonymity attacks, to very strong notions of universally composable (UC) simulation.

In particular, the mix-net of Abe was proved in [Abe99, AH01, AI06] to satisfy the following anonymity notion: An efficient adversary who corrupts a subset of users, mix-servers, and decryption servers cannot gain noticeable advantage in predicting any single input-output pair $(i, j) \in [n]^2$

[2]Recall $B_\pi, \hat{R}_0, \hat{R}_1$ are $L \times [\frac{n}{2}]$ matrices which align with the gate structure of the Beneš network

for which honest user i's encrypted plaintext is permuted to position j in the output. Note that this definition protects the anonymity of each user, but is weaker than more general indistinguishability and simulation definitions, in that it could potentially reveal correlations between users (e.g., that users 2 and 3 voted in the same fashion).

We consider a stronger indistinguishability-based notion of anonymity, in the flavor of [NSK04]. Intuitively, our definition requires that for any permutation on the honest users' votes, the resulting views of the mix-net protocol and verification—including the view of a *possibly corrupt* verifier—are indistinguishable.[3] Note that this implies the anonymity definition of Abe [Abe99, AH01, AI06], as a successful (i, j)-predicting adversary would serve as a successful distinguisher between views for permutations σ, σ' which disagree on user i.

We formalize this notion via a notion of *distributional WI* (D-WI), a strengthening of WI we introduce that is related to strong-WI [Gol01], but parametrized by specific pairs of distributions.

Distributional Witness Indistinguishabilty. For ease of reading, we will make use of the following shorthand notation for the distribution over the view of a (potentially malicious) verifier $\mathcal{V}$ within an interactive proof $(\mathcal{P}, \mathcal{V})$ for a given distribution over statements (and witnesses).

Notation 1 ($\mathsf{View}_{\mathcal{V}^*}[D_\lambda]$). Let $(\mathcal{P}, \mathcal{V})$ be an interactive proof for a relation R. For a given ensemble of distributions D_λ over statements, witnesses, and auxiliary input $\{(X_\lambda, W_\lambda, Z_\lambda)\}_{\lambda\in\mathbb{N}}$ for which $(X_\lambda, W_\lambda) \in \mathsf{R}$ and $|X_\lambda| \geq \lambda$, and PPT interactive machine $\mathcal{V}^*$, we define the distribution

$$\mathsf{View}_{\mathcal{V}^*}[D_\lambda] := \{\langle \mathcal{P}(W_\lambda), \mathcal{V}^*(Z_\lambda)\rangle (X_\lambda) : (X_\lambda, W_\lambda, Z_\lambda) \leftarrow D_\lambda\}_{\lambda\in\mathbb{N}}\,.$$

Definition 10 (D-WI)**.** Let $(\mathcal{P}, \mathcal{V})$ be an interactive proof for a relation R, and let D_λ and D'_λ be two probability ensembles over statements, witnesses, and auxiliary inputs, as in Notation 1. We say that $(\mathcal{P}, \mathcal{V})$ is *distributional witness-indistinguishable (D-WI) with respect to D_λ, D'_λ* for relation R if for every PPT interactive machine $\mathcal{V}^*$, the following holds: $\mathsf{View}_{\mathcal{V}^*}[D_\lambda] \overset{c}{\approx} \mathsf{View}_{\mathcal{V}^*}[D'_\lambda]$.

Mix-net Anonymity. For a given mix-net protocol MixNet, adversarial entity A, and vector of honest user votes $(u_i)_{i\in\mathcal{U}_{\bar{A}}}$, the distribution $D_\lambda^{\mathsf{MixNet},\mathcal{A}}((u_i)_{i\in\mathcal{U}_{\bar{A}}})$ as given in Definition 11 denotes the induced distribution over statements, witnesses, and auxiliary input of correctness of the mix-net. Our notion of anonymity (Definition 12) requires the interactive proof system for correctness of the mix-net to be distributional witness indistinguishable (D-WI) with respect to any pair $D_\lambda^{\mathsf{MixNet},A}((u_i)_{i\in\mathcal{U}_{\bar{A}}})$ and $D_\lambda^{\mathsf{MixNet},A}((u_{\sigma(i)})_{i\in\mathcal{U}_{\bar{A}}})$, for any permutation σ on the ordering of the honest users.

Definition 11 ($D_\lambda^{\mathsf{MixNet},A}$ distribution)**.** Let $\mathsf{MixNet} = (\mathsf{Setup}, \mathsf{SubmitCipher}, \mathsf{VrfblyMix}, \mathsf{Decrypt})$ be a verifiable n-user m-server mix-net system with respect to a re-randomizable encryption scheme $\mathcal{E}$ over message space M (as given in Definition 6, Section 2.2).

Let $A = (\mathcal{U}_A, \mathcal{S}_A, \mathcal{A})$ be given, where $\mathcal{U}_A \subseteq [n]$, $\mathcal{S}_A \subset [m]$ are corrupted subsets of users and mix-servers, respectively, and $\mathcal{A}$ is an adversarial non-uniform PPT algorithm which has four modes $\mathsf{setup}, \mathsf{submit\ votes}, \mathsf{mix}$, and $\mathsf{decrypt}$ with the syntax as below. We define the distribution $D_\lambda^{\mathsf{MixNet},A}$ as follows (we denote $\mathcal{U}_{\bar{A}} = [n] \setminus \mathcal{U}_A$ and $\mathcal{S}_{\bar{A}} = [m] \setminus \mathcal{S}_A$):

[3] We remark, however, that [Abe99, AH01, AI06] directly consider non-malleability concerns, which we factor out and address separately; see Remark on Non-Malleability below. Note that Abe and Imai considered notions of anonymity against both static and *adaptive* adversaries [AI06]; however, anonymity of Abe's mix-net construction was proven only in the static setting [Abe99, AH01], and thus this is the notion we compare against.

$D_\lambda^{\mathsf{MixNet},A}((u_i)_{i\in\mathcal{U}_{\bar{A}}})$:

Input For each honest user $i \in \mathcal{U}_{\bar{A}}$, a vote $u_i \in \{0,1\}$.

– Let $\mathsf{state} := \emptyset$

– Sample $(\mathsf{pk}, (\mathsf{sk}_1, \ldots, \mathsf{sk}_m), \mathsf{state}) \leftarrow \mathsf{Setup}^{\mathcal{A}(\text{“}\mathsf{setup}\text{”},1^\lambda,\mathsf{state})}(1^\lambda)$,
i.e., simulate Setup protocol execution on honest party input 1^λ and (oracle access to) adversarial next-message function $\mathcal{A}(\text{“}\mathsf{setup}\text{”}, 1^\lambda, \mathsf{state})$, in each round with updated state. Output the induced values $\mathsf{pk}, (\mathsf{sk}_1, \ldots, \mathsf{sk}_m)$, and updated state.

– For each $i = 1, \ldots, n$: // Submit votes (n users)

if $i \in \mathcal{U}_A$
then Sample $(c_i^0, z_i) \leftarrow \mathcal{A}(\text{“}\mathsf{submit\ votes}\text{”}, \mathsf{pk}, i, \mathsf{state})$
Update $\mathsf{state} := \{z_i\} \cup \mathsf{state}$
else Sample $c_i^0 \leftarrow \mathsf{SubmitCipher}(\mathsf{pk}, u_i)$

– For $j = 1, \ldots, m$ do: // Mix phase (m mix servers)

if $j \in \mathcal{S}_A$
then Sample $(\vec{c}^j, w_j^\pi, z_j^\pi) \leftarrow \mathcal{A}(\text{“}\mathsf{mix}\text{”}, \mathsf{pk}, j, \vec{c}^{j-1}, \mathsf{state})$
Update $\mathsf{state} := \{z_j^\pi\} \cup \mathsf{state}$
else Sample $\mathsf{rnd}_j \leftarrow \$$, and set $\vec{c}^j = \mathsf{Mix}_j(\mathsf{pk}, \vec{c}^{j-1}; \mathsf{rnd}_j)$

– Run $(\vec{v}, (w_j^{\mathsf{sk}})_{j\in\mathcal{S}_A}, \mathsf{state}) \leftarrow \mathsf{Decrypt}^{\mathcal{A}(\text{“}\mathsf{decrypt}\text{”},\mathsf{state})}(\vec{c}^m, (\mathsf{sk}_j)_{j\in\mathcal{S}_{\bar{A}}})$,
i.e., simulate Decrypt protocol execution on input $(\vec{c}^m, \mathsf{sk}_j)$ for each honest mix-server j, and oracle access to adversarial next-message function $\mathcal{A}(\text{“}\mathsf{decrypt}\text{”}, \mathsf{state})$, in each round with updated state. Output the induced plaintext vector $\vec{v}$, adversarial witness information $(w_j^{\mathsf{sk}})_{j\in\mathcal{S}_A}$ for decryption, and updated state.

Output : $(X_\lambda, W_\lambda, Z_\lambda)$ where

– $X_\lambda = (\mathsf{pk}, \vec{c}^0, \vec{v})$

– $W_\lambda = ((\mathsf{rnd}_j)_{j\in\mathcal{S}_{\bar{A}}}, (w_j^\pi)_{j\in\mathcal{S}_A}, (\mathsf{sk}_j)_{j\in\mathcal{S}_{\bar{A}}}, (w_j^{\mathsf{sk}})_{j\in\mathcal{S}_A})$

– $Z_\lambda = (\mathsf{state})$

Definition 12 (*Anonymous* Mix-Net System)**.** We say that a verifiable n-user m-server mix-net system MixNet is *anonymous* if for every A (as in Definition 11), every choice of honest user votes $u_i \in \{0,1\}$ for $i \in \mathcal{U}_{\bar{A}}$, and every permutation σ over the honest users $\mathcal{U}_{\bar{A}}$ (i.e. $\sigma : \mathcal{U}_{\bar{A}} \hookrightarrow \mathcal{U}_{\bar{A}}$) the interactive proof system $(\mathcal{P}, \mathcal{V})$ for correctness of MixNet is D-WI with respect to the following two probability ensembles $D_\lambda = D_\lambda^{\mathsf{MixNet},A}((u_i)_{i\in\mathcal{U}_{\bar{A}}})$ and $D'_\lambda = D_\lambda^{\mathsf{MixNet},A}((u_{\sigma(i)})_{i\in\mathcal{U}_{\bar{A}}})$ where $D_\lambda^{\mathsf{MixNet},A}$ is as in Definition 11.

Remark. A few remarks on the $D_\lambda^{\mathsf{MixNet},A}$ distribution:

- **Sampling pk.** We remark that the somewhat-complex structure of $D_\lambda^{\mathsf{MixNet},A}$ is necessary, as opposed to fixing a public key and/or corrupt ciphertexts as parameters, since to achieve indistinguishability in such setting one must necessarily rely on semantic security of the encryption scheme, and thus on the public key being randomly sampled. In turn, we must allow the adversary's ciphertexts to be chosen (efficiently) as a function of pk.

- **Correctness / Non-Malleability of $\mathsf{SubmitCipher}$.** However, in effort to isolate a simple definition, within $D_\lambda^{\mathsf{MixNet},A}$ we assume the adversary chooses ciphertexts of corrupt users given the public key pk but *not* the ciphertexts of honest parties. To convert to security in the more general setting (that is, to address possible malleability attacks), one can apply generic techniques of accompanying the ciphertext output from $\mathsf{SubmitCipher}$ with a form of proof of knowledge of the underlying plaintext associated with the user index: i.e., making the encryption scheme *plaintext-aware* [BR94]. For example, for the case of ElGamal encryption (as used in this work), the $\mathsf{SubmitCipher}$ phase can be modified to have each user output a Chaum-Pederson-signed ElGamal encryption of his input vote u_i, which provides plaintext-awareness [Fis05, ST13].

  We refer the reader to e.g. [KMW12] for a thorough discussion of the subtleties arising when addressing non-malleability directly within anonymity definitions.

4 Abe's Mix-Net With Lossy Encryption

For our first mix-net construction, we consider an implementation of Abe with a modified *lossy ElGamal* encryption scheme.

A full description of Abe's original mix-net is given in Section 2.4. In Section 4.1 we present the additional necessary building blocks, and in Section 4.2 we provide our construction.

4.1 Building Blocks for Lossy Abe

A *lossy encryption scheme* [PVW07] $(\mathsf{KeyGen}, \mathsf{KeyGen}_{\mathsf{loss}}, \mathsf{Enc}, \mathsf{Dec})$ is a PKE scheme which possesses an alternative "lossy mode" key generation algorithm $\mathsf{KeyGen}_{\mathsf{loss}}$, whose output pk is computationally indistinguishable from an honestly generated pk, but for which the encryption of a message m information theoretically hides m. (See Definition 5 in Section 2.1 for formal definition.)

We make use of the following lossy variant of ElGamal.

Definition 13 (Lossy ElGamal [BHY09])**.** The *lossy ElGamal encryption* scheme for message space $M = \{0, 1\}$ is given by:

- $\mathsf{KeyGen}(1^\lambda)$: Generate the description of a cyclic group $\mathbb{G}$ of prime order q (with $\log_2 q \geq \lambda$) and generators g_0, g_1. Sample a random secret key $s \leftarrow [q-1]$ and compute $h_0 = g_0^s, h_1 = g_1^s$. Output $\mathsf{pk} = (\mathbb{G}, q, g_0, g_1, h_0, h_1)$ and $\mathsf{sk} = s$.

- $\mathsf{KeyGen}_{\mathsf{loss}}(1^\lambda)$: Generate the description of $\mathbb{G}$ and g_0, g_1 as above. Sample *two* random elements $s_0, s_1 \leftarrow [q-1]$, compute $h_0 = g_0^{s_0}, h_1 = g_1^{s_1}$. Output $\mathsf{pk} = (\mathbb{G}, q, g_0, g_1, h_0, h_1)$.

- $\mathsf{Enc}_{\mathsf{pk}}(m)$: Sample $r_0, r_1 \leftarrow [q-1]$. Output $(g_0^{r_0} g_1^{r_1}, h_0^{r_0} h_1^{r_1} \cdot g^m)$.

- $\mathsf{Dec}_{\mathsf{sk}}(c=(a,b))$: Compute $u := b \cdot a^{-s}$, and output $m \in \{0,1\}$ for which $u = g^m$.
- $\mathsf{ReRand}_{\mathsf{pk}}(c=(a,b))$: Choose random $r_0, r_1 \leftarrow [q-1]$, and output $c_{out} = (a \cdot g_0^{r_0} g_1^{r_1}, b \cdot h_0^{r_0} h_1^{r_1})$.

Theorem 2 ([BHY09])**.** *Based on the Decisional Diffie-Hellman assumption, the Lossy ElGamal scheme (Definition 13) is a rerandomizable (Definition 3) lossy PKE scheme (Definition 5).*

Note that ciphertexts are composed of two group elements, and conversely any pair of elements of $\mathbb{G}$ can be interpreted as a "valid" ciphertext under a given public key pk.

Proving correctness of the new switch gate can be achieved with WI via a similar approach as to standard ElGamal: Here, combining the protocol of Cramer *et al.* for proving OR [CDS94] instead with Okamoto's protocol [Oka93] for proving knowledge of Pedersen commitments (in the place of the Chaum-Pederson protocol [CP92] for proving equality of discrete logarithms). Further details of the resulting 3-round proof are given in Appendix A.2.

4.2 Lossy Abe Mix-Net

Construction 3 (Lossy Abe Mix-Net). We define the $n = 2^d$-user *lossy Abe mix-net system* $\mathsf{MixNet}^{\mathsf{loss}}$ to be identical to Abe's mix-net (as in Construction 1), with two exceptions:

- All mix-net procedures $\mathsf{Setup}, \mathsf{SubmitCipher}, \mathsf{VrfblyMix}, \mathsf{Decrypt}$ make use of the *Lossy ElGamal* algorithms $\mathsf{KeyGen}, \mathsf{Enc}$, and ReRand (Definition 13), in the place of ElGamal.
- Each gate-consistency proof execution $(\mathcal{P}_{\mathsf{Gate}}, \mathcal{V}_{\mathsf{Gate}})$ (which was specific to ElGamal) within Abe's $(\mathcal{P}^{\mathsf{Abe}}_{\mathsf{Mix}}, \mathcal{V}^{\mathsf{Abe}}_{\mathsf{Mix}})$ is replaced by a corresponding gate-consistency proof execution $(\mathcal{P}^{\mathsf{loss}}_{\mathsf{Gate}}, \mathcal{V}^{\mathsf{loss}}_{\mathsf{Gate}})$ for Lossy ElGamal, as specified in Appendix A.2 (this proof is formed as an OR (via Cramer *et al.* [CDS94]) of ANDs of Okamoto [Oka93]).

Note that while we use Lossy ElGamal for concreteness, a similar approach could be taken using amenable lossy encryption schemes based on, e.g., quadratic residosity, Paillier, or LWE (see e.g., [BHY09, PW11, FGK+13]).

Theorem 3 (Lossy Abe is Anonymous)**.** *The Lossy Abe Mix-Net, as described in Construction 3, is anonymous, as per Definition 12.*

Proof. Let $A = (\mathcal{U}_A, \mathcal{S}_A, \mathcal{A})$ be as in Definition 11, $u_i \in \{0,1\}$ for $i \in \mathcal{U}_{\bar{A}}$ a choice of honest user votes, and σ a permutation over the honest users $\mathcal{U}_{\bar{A}}$. We show that for any PPT interactive machine $\mathcal{V}^*$: $\mathsf{View}_{\mathcal{V}^*}[D_\lambda^{\mathsf{MixNet},A}((u_i)_{i\in\mathcal{U}_{\bar{A}}})] \overset{c}{\approx} \mathsf{View}_{\mathcal{V}^*}[D_\lambda^{\mathsf{MixNet},A}((u_{\sigma(i)})_{i\in\mathcal{U}_{\bar{A}}})]$, where $D_\lambda^{\mathsf{MixNet},A}$ is as in Definition 11), by a sequence of the hybrids which use also the following claim:

Claim 1 (Multiple Witnesses). With overwhelming probability over the choice of a *lossy* key $\mathsf{pk}^{\mathsf{loss}} \leftarrow \mathsf{KeyGen}_{\mathsf{loss}}(1^\lambda)$, the following holds. For any ciphertexts x_0, x_1, y_0, y_1 in the support of $\mathsf{Enc}_{\mathsf{pk}^{\mathsf{loss}}}(\cdot)$, there exists $(\hat{r}_0, \hat{r}_1), (\tilde{r}_0, \tilde{r}_1)$ for which $y_b = \mathsf{ReRand}_{\mathsf{pk}^{\mathsf{loss}}}(x_b; \hat{r}_b)$ for $b \in \{0,1\}$ *and* $y_b = \mathsf{ReRand}_{\mathsf{pk}^{\mathsf{loss}}}(x_{1-b}; \tilde{r}_{1-b})$ for $b \in \{0,1\}$.

Proof. Follows by the equivalence of distributions $\mathsf{Enc}_{\mathsf{pk}^{\mathsf{loss}}}(m_0) \equiv \mathsf{Enc}_{\mathsf{pk}^{\mathsf{loss}}}(m_1)$ for all messages $m_0, m_1 \in M$ under a lossy key. □

Recall the view of $\mathcal{V}^*$ consists of: honest user votes $(u_i)_{i\in\mathcal{U}_{\bar{A}}}$ (chosen by A), the view during the key setup phase $\mathsf{view}^{\mathsf{Setup}}$, the public key pk, secret shares $(\mathsf{sk}_j)_{j\in\mathcal{S}_A}$ of sk, vote ciphertexts of corrupt parties $(c_i^0)_{i\in\mathcal{U}_{\bar{A}}}$, the view of $\mathcal{V}^*$ within the mix phase $\left(\mathsf{view}^{\mathsf{Mix}_j}\right)_{j\in[m]}$, the view of $\mathcal{V}^*$ during the $\mathsf{Decrypt}$ joint decryption $\mathsf{view}^{\mathsf{Dec}}$, and the shuffled plaintext votes $\vec{v}$.

At a high level, the proof of Theorem 3 moves from

$$\mathsf{View}_{\mathcal{V}^*}[D_\lambda^{\mathsf{MixNet},A}((u_i)_{i\in\mathcal{U}_{\bar{A}}})] \text{ to } \mathsf{View}_{\mathcal{V}^*}[D_\lambda^{\mathsf{MixNet},A}((u_{\sigma(i)})_{i\in\mathcal{U}_{\bar{A}}})]$$

via the following sequence of hybrids. (1) First, $\mathsf{view}^{\mathsf{Setup}}$ and $\mathsf{view}^{\mathsf{Dec}}$ are replaced by simulated views, relying on zero knowledge simulation of the setup and joint decryption protocols. (2) The honest setup functionality is replaced by a modified one which samples a *lossy* system public key and outputs *random* secret key shares sk_j to the corrupt servers. (3) Using semantic security, the encryptions of honest user votes $(u_i)_{i\in\mathcal{U}_{\bar{A}}}$ are replaced by encryptions of the σ-permuted values $(u_{\sigma(i)})_{i\in\mathcal{U}_{\bar{A}}}$ (but the mix and decryption phases are still with respect to $(u_i)_{i\in\mathcal{U}_{\bar{A}}}$). (4) One uncorrupted mix-server modifies his permutation to "undo" the σ shuffle of honest votes. This step relies on the special-soundness property of the mix phase (in order to extract the permutations used by corrupt mix-servers), the WI of the gate-consistency proofs, and the existence of multiple witnesses for any ReRand-switch gate with respect to a lossy public key. (5) The setup procedure is returned to the honest (non-lossy) version. (6) Finally, the simulated $\mathsf{view}^{\mathsf{Setup}}, \mathsf{view}^{\mathsf{Dec}}$ are returned to the honestly generated versions.

We defer the full proof to Appendix C.1. □

5 Abe's Mix-net With Injected Dummy Votes

We demonstrate that an alternative simple tweak to the Abe mix-net system with comparable efficiency preserves verifiability, and further guarantees anonymity against a malicious verifier. At a high level, our construction is identical to the Abe mix-net (*without* changing the encryption scheme) on $2n$ votes, where n "dummy" ciphertexts of $\perp$ are introduced and removed at the beginning and end of each mix-server mix phase. To verify that this process was followed honestly, the injected ciphertexts will be decrypted at the end along with the shuffled votes (in a carefully chosen order).

Construction 4 (Injected Abe Mix-Net). The *injected Abe $n = 2^d$-user m-servers mix-net system* is identical to Abe's mix-net (as in Construction 1), with two exceptions: 1) $\mathsf{VrfblyMix}_{\mathsf{Abe}}^{\mathsf{inject}}(\mathsf{pk}, \vec{c}^0)$ is a sequential algorithm with m iterations, where each iteration $j \in [m]$ is an execution of $\mathsf{Mix}^{\mathsf{Inject}}$ as given below (instead of $\mathsf{Mix}^{\mathsf{Abe}}$), and 2) the verification proof system $(\mathcal{P}_{\mathsf{Abe}}^{\mathsf{inject}}, \mathcal{V}_{\mathsf{Abe}}^{\mathsf{inject}})$ has 4 steps as described below (instead of $(\mathcal{P}^{\mathsf{Abe}}, \mathcal{V}^{\mathsf{Abe}})$).

$\mathsf{Mix}^{\mathsf{Inject}}(\mathsf{pk}, \vec{c}^{j-1})$: Let $L = 2d - 1$. Perform the following:

1. (Inject Fake Votes). Generate n encryptions of the message $\perp$, and insert them into the even positions of a new $(N = 2n)$-length vector $\vec{C}^{j-1}$, with the real input ciphertexts in the odd positions. That is, for every $i \in [n]$,

$$C_{2i-1}^{j-1} := c_i^{j-1}, \qquad C_{2i}^{j-1} \leftarrow \mathsf{Enc}_{\mathsf{pk}}(\perp).$$

2. (Choose Permutation). Sample a random permutation $\pi_j \leftarrow S_n$ on n elements, and let $\pi_j^{\mathsf{new}} \in S_N$ be the permutation on N elements that acts as π on the odd positions and as the identity on the evens. That is:

$$\forall i \in [n]: \quad \pi^{\mathsf{new}}[2i-1] = 2 \cdot \pi[i] - 1, \qquad \pi^{\mathsf{new}}[2i] = 2i.$$

3. (AbeMix on $2n$ Inputs): Execute AbeMix (Step 2 of $\mathsf{Mix}^{\mathsf{Abe}}$, Construction 1) with input $(\mathsf{pk}, \vec{C}^{j-1}, \pi_j^{\mathsf{new}})$. Let $w^j = (\vec{C}^j, B_{\pi_j^{\mathsf{new}}}, \hat{R}_0^j, \hat{R}_1^j)$ be the resulting output.

4. (Remove Fake Votes). Output the length-n vector $\vec{c}^j$ corresponding to the odd locations of $\vec{C}^j$. That is, output

$$\forall i \in [n]: \quad c_i^j := {C'}_{2i-1}.$$

$(\mathcal{P}_{\mathsf{Abe}}^{\mathsf{inject}}, \mathcal{V}_{\mathsf{Abe}}^{\mathsf{inject}})$: The interactive proof system $(\mathcal{P}_{\mathsf{Abe}}^{\mathsf{inject}}, \mathcal{V}_{\mathsf{Abe}}^{\mathsf{inject}})$ with common input $(\mathsf{pk}, \vec{c}^0, \vec{v})$ and witness $(\mathsf{rnd}_j, \mathsf{sk}_j)_{j\in[m]}$ is

1. **Submission of intermediate ciphertext vectors**: For every $j \in [m]$: $\mathcal{P}$ generates and sends $\mathcal{V}$ the input and output lists of ciphertexts $(\vec{C}^{j-1}, \vec{C}^j)$ where $\vec{C}^{j-1}$ is generated as in step 1 above, and $\vec{C}^j$ is the list of ciphertexts output from AbeMix in step 3 above. $\mathcal{V}$ verifies that the output ciphertexts in odd locations for each mix-server $j-1$ are identical to the corresponding input ciphertexts to mix-server j: i.e., for every $j \in [m-1], i \in [n]$: $C_{2i-1}^j = C_{2i-1}^{j+1}$. Additionally, $\mathcal{V}$ verifies that the first set of ciphertexts in odd locations agree with the submitted vote ciphertexts: $c_i^0 = C_{2i-1}^0$, for every $i \in [n]$.
2. **Correctness Proof of** $\mathsf{VrfblyMix}$: For every $j \in [m]$, execute $(\mathcal{P}_{\mathsf{Mix}}^{\mathsf{Abe}}, \mathcal{V}_{\mathsf{Mix}}^{\mathsf{Abe}})$ with input $(\mathsf{pk}, \vec{C}^{j-1}, \vec{C}^j)$ and witness $(\pi^{\mathsf{new}}, B_{\pi_j^{\mathsf{new}}}, \hat{R}_0^j, \hat{R}_1^j)$
3. **Correctness Proof of Injected Fake Votes**: Let $\vec{v}^{\perp} = (\perp, \ldots, \perp)$ be an n-dimension vector of the message $\perp$, and $c^{j-1,\perp}$ be the vector of n ciphertexts such that $c_i^{j-1,\perp} = C_{2i}^{j-1}$ for every $i \in [n]$. Execute $(\mathcal{P}_{\mathsf{Dec}}^{\mathsf{Abe}}, \mathcal{V}_{\mathsf{Dec}}^{\mathsf{Abe}})$ with input $(\mathsf{pk}, \vec{c}^{j-1,\perp}, \vec{v}^{\perp})$, using witness $(\mathsf{sk}_j)_j \in [m]$. If the prover is rejected in this step, the proof system terminates, and no further steps take place.
4. **Correctness Proof of** $\mathsf{Decrypt}$: Let $\vec{c}^m$ be a list of n ciphertexts such that $c_i^m = C_{2i-1}^m$. Execute $(\mathcal{P}_{\mathsf{Dec}}^{\mathsf{Abe}}, \mathcal{V}_{\mathsf{Dec}}^{\mathsf{Abe}})$ with input $(\mathsf{pk}, \vec{c}^m, \vec{v})$ and witness $(\mathsf{sk}_j)_j \in [m]$. If the prover is rejected in this step, or if for any $i \in [n]$ it holds that $v_i = \perp$, the proof system terminates, and no further steps take place.
5. **Correctness Proof of Removed Fake Votes**: Let $\vec{v}^{\perp} = (\perp, \ldots, \perp)$ be an n-dimension vector of the message $\perp$, and $c^{j,\perp}$ be the vector of n ciphertexts such that $c_i^{j,\perp} = C_{2i}^j$ for every $i \in [n]$. Execute $(\mathcal{P}_{\mathsf{Dec}}^{\mathsf{Abe}}, \mathcal{V}_{\mathsf{Dec}}^{\mathsf{Abe}})$ with input $(\mathsf{pk}, \vec{c}^{j,\perp}, \vec{v}^{\perp})$, using witness $(\mathsf{sk}_j)_j \in [m]$. If the prover is rejected in this step, the proof system terminates, and no further steps take place.

Overall $(\mathcal{P}_{\mathsf{Abe}}^{\mathsf{inject}}, \mathcal{V}_{\mathsf{Abe}}^{\mathsf{inject}})$ proves that: (1) the submitted user ciphertexts are properly copied into the odd positions of the first mix input vector, and for every mix server j the ciphertexts in the odd locations of its input ciphertext vector are the same as those in the output of server $j-1$;[4] (2)

[4]Note that any pair of group elements can be interpreted as a "valid" ElGamal ciphertext under the public key pk.

every mix server permuted its input vector to its output vector; (3) the injected ciphertexts (in even positions) of each mix server are encryptions of $\perp$; (4) the final ciphertexts in the odd locations indeed decrypt to $\vec{v}$; and (5) the final ciphertexts in the even locations decrypt to $\perp$. Altogether, this ensures that the final vector $\vec{v}$ is indeed the permutation of the votes underlying $\vec{c}^0$. That is, soundness holds.

Theorem 4 (Injected Abe Mix-Net is Anonymous)**.** *The Injected Abe Mix-Net, as described in Construction 4, is anonymous (as per Definition 12).*

The proof uses the following core lemma, focusing on the proof of a single mix-phase. It states that for an honest mix-server who indeed injects ciphertexts of $\perp$ in even positions, then the view of a malicious verifier during the proof of correctness of the corresponding mix-phase is indistinguishable for *any* pair of implemented permutations which fix the even-location positions (but operate arbitrary $\pi_0, \pi \in S_n$ on the odd-location positions).

Lemma 5 (Replacing Permutation in Mix)**.** *For every (adversarial) non-uniform PPT $\mathcal{A}$, and every two permutations $\pi_0, \pi_1 \in S_n$, the interactive proof system $(\mathcal{P}^{\mathsf{Abe}}_{\mathsf{Mix}}, \mathcal{V}^{\mathsf{Abe}}_{\mathsf{Mix}})$ (for correctness of Abe mixing) for the relation $\mathsf{R}_{\mathsf{Mix}}$ (eq. (2)) in Abe Mix-net satisfies distributional witness-indistinguishability (D-WI) with respect to the following two distribution ensembles $D_\lambda = D^{\mathsf{Mix},\mathcal{A}}_\lambda(\pi_0)$ and $D'_\lambda = D^{\mathsf{Mix},\mathcal{A}}_\lambda(\pi_1)$ where $D^{\mathsf{Mix},\mathcal{A}}_\lambda$ is as in Definition 14 described below.*

Definition 14 ($D^{\mathsf{Mix},\mathcal{A}}_\lambda$)**.** For any (adversarial) non-uniform PPT algorithm $\mathcal{A}$, and security parameter $\lambda \in \mathbb{N}$, we define the following distribution $D^{\mathsf{Mix},\mathcal{A}}_\lambda$ as follows:

$D^{\mathsf{Mix},\mathcal{A}}_\lambda(\pi)$:

Input Permutation $\pi \in S_n$

- Sample $(\mathsf{pk}, (\mathsf{sk}_1, \ldots, \mathsf{sk}_m)) \leftarrow \mathsf{Setup}(1^\lambda)$
- For every $i \in [n]$: Obtain $c_i, z_i \leftarrow \mathcal{A}(\mathsf{pk}, i)$
- For every $i \in [n]$: Set $C_{2i-1} := c_i$ and $C_{2i} \leftarrow \mathsf{Enc}_{\mathsf{pk}}(\mathsf{Enc}_{\mathsf{pk}}(\perp)$
- Let π^{new} be such that $\forall i \in [n]$:
  $\pi^{\mathsf{new}}[2i-1] = 2 \cdot \pi[i] - 1, \quad \pi^{\mathsf{new}}[2i] = 2i.$
- Execute AbeMix (step 2 in Abe's Mix, on $2n$ votes):
  $(\vec{C}', B_{\pi^{\mathsf{new}}}, \hat{R}_0, \hat{R}_1) \leftarrow \mathsf{AbeMix}(\mathsf{pk}, \vec{C}, \pi^{\mathsf{new}})$

Output $(X_\lambda = (\mathsf{pk}, \vec{C}, \vec{C}'), W_\lambda = (B_{\pi^{\mathsf{new}}}, \hat{R}_0, \hat{R}_1), Z_\lambda = (z_1, \ldots, z_n))$

Proof. We change from the Beneš switch gate settings of $\pi^{\mathsf{new}}_0 \in S_{2n}$ to those of π^{new}_1 one gate at a time, in a particular order. This is achieved by a sequence of steps of the following two forms: (a) For any honest ciphertext (i.e., encrypting $\perp$), we can change the plaintext, by semantic security. (b) For any gate whose input ciphertexts encrypt the same plaintext (i.e., 2 witnesses to the switch gate), we can flip the switch bit from b to $1 - b$, by WI.

The order of gates is as follows.

We first target the last (output) level of the Beneš network, changing from the corresponding last-level bits of π_0^{new} to those of π_1^{new}. Since the mix-server is honest, in each even output position $2i$ in the last level is the (rerandomized) $\perp$ ciphertext that originated in input position $2i$. Using step type (a) (i.e., semantic security), convert each ciphertext $2i$ to encrypt the same value as its output-gate neighbor $2i-1$. This can be done by rerandomizing the neighbor ciphertext and using this as the original injected "$\perp$" $2i$th ciphertext. Note that changing the plaintext does not affect the permutation, meaning the same pairs of ciphertexts will appear together in the last level gates. Then, given the plaintext switch, we have that every gate in the final level has a pair of ciphertexts of the *same* plaintext. Then using step type (b) (i.e., WI), we may change each gate to agree with the Beneš settings for π_1^{new}.

Next we target the gates in the upper sub-Beneš (see Figure 1). Again we will use the power of the honest mix-server controlled dummy "$\perp$" ciphertexts to change from the corresponding permutation bits of π_0^{new} to those of π_1^{new}. First, we "direct" all the $\perp$ ciphertexts up to enter this sub-network by (temporarily) changing the switch settings of the *first* (input) level of the Beneš: Using (a) change all $\perp$ ciphertexts $2i$ to encrypt the same value as their *input*-gate neighbor $2i-1$, using (b) change all first-level gates to switch value 1, so that all ciphertexts entering the upper sub-Beneš are dummy, and then using (a) change them all back to encryptions of $\perp$. At this point, all gates in the upper sub-Beneš satisfy the conditions of step (b) (namely, all ciphertexts encrypt the same plaintext $\perp$), which means they can be changed one by one to agree with the Beneš settings for π_1^{new}.

Finally, the gates in the lower sub-Beneš and in the first-level (input) gates are changed in an analogous fashion. See Appendix C.2 for details. □

Given Lemma 5, the proof of anonymity follows essentially the same structure as in the case of the Lossy Abe Mix-net (where previously an analogous statement held by the multiple-witness guarantee of the lossy public key combined with WI). We note that the order of the executions of $(\mathcal{P}_{\mathsf{Dec}}^{\mathsf{Abe}}, \mathcal{V}_{\mathsf{Dec}}^{\mathsf{Abe}})$ (i.e., first the injected even-position ciphertexts, then the final shuffled user votes, then the post-shuffle even-position ciphertexts) is important in order to ensure that we can properly simulate the execution of these executions (i.e., Hybrid 1 in the Lossy Abe proof) without information on users' votes.

References

[Abe99] Masayuki Abe. Mix-networks on permutation networks. In *Advances in Cryptology - ASIACRYPT '99, International Conference on the Theory and Applications of Cryptology and Information Security, Singapore, November 14-18, 1999, Proceedings*, pages 258–273, 1999.

[AH01] Masayuki Abe and Fumitaka Hoshino. Remarks on mix-network based on permutation networks. In *Public Key Cryptography, 4th International Workshop on Practice and Theory in Public Key Cryptography, PKC 2001, Cheju Island, Korea, February 13-15, 2001, Proceedings*, pages 317–324, 2001.

[AI06] Masayuki Abe and Hideki Imai. Flaws in robust optimistic mix-nets and stronger security notions. *IEICE Transactions*, 89-A(1):99–105, 2006.

[BG12] Stephanie Bayer and Jens Groth. Efficient zero-knowledge argument for correctness of a shuffle. In *Advances in Cryptology - EUROCRYPT 2012 - 31st Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cambridge, UK, April 15-19, 2012. Proceedings*, pages 263–280, 2012.

[BHY09] Mihir Bellare, Dennis Hofheinz, and Scott Yilek. Possibility and impossibility results for encryption and commitment secure under selective opening. In *Advances in Cryptology - EUROCRYPT 2009, 28th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cologne, Germany, April 26-30, 2009. Proceedings*, pages 1–35, 2009.

[BR94] Mihir Bellare and Phillip Rogaway. Optimal asymmetric encryption. In *Advances in Cryptology - EUROCRYPT '94, Workshop on the Theory and Application of Cryptographic Techniques, Perugia, Italy, May 9-12, 1994, Proceedings*, pages 92–111, 1994.

[CDS94] Ronald Cramer, Ivan Damgård, and Berry Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. In *Advances in Cryptology - CRYPTO '94, 14th Annual International Cryptology Conference, Santa Barbara, California, USA, August 21-25, 1994, Proceedings*, pages 174–187, 1994.

[CGJ+99] Ran Canetti, Rosario Gennaro, Stanislaw Jarecki, Hugo Krawczyk, and Tal Rabin. Adaptive security for threshold cryptosystems. In *Advances in Cryptology - CRYPTO '99, 19th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 1999, Proceedings*, pages 98–115, 1999.

[Cha81] David Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Commun. ACM*, 24(2):84–88, 1981.

[CP92] David Chaum and Torben P. Pedersen. Wallet databases with observers. In *Advances in Cryptology - CRYPTO '92, 12th Annual International Cryptology Conference, Santa Barbara, California, USA, August 16-20, 1992, Proceedings*, pages 89–105, 1992.

[FGK+13] David Mandell Freeman, Oded Goldreich, Eike Kiltz, Alon Rosen, and Gil Segev. More constructions of lossy and correlation-secure trapdoor functions. *J. Cryptology*, 26(1):39–74, 2013.

[Fis05] Marc Fischlin. Completely non-malleable schemes. In *Automata, Languages and Programming, 32nd International Colloquium, ICALP 2005, Proceedings*, pages 779–790, 2005.

[FS86] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *Advances in Cryptology - CRYPTO '86, Santa Barbara, California, USA, 1986, Proceedings*, pages 186–194, 1986.

[FS90] Uriel Feige and Adi Shamir. Witness indistinguishable and witness hiding protocols. In *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing, May 13-17, 1990, Baltimore, Maryland, USA*, pages 416–426, 1990.

[GGMM97] Eran Gabber, Phillip B. Gibbons, Yossi Matias, and Alain J. Mayer. How to make personalized web browising simple, secure, and anonymous. In *Financial Cryptography, First International Conference, FC '97, Anguilla, British West Indies, February 24-28, 1997, Proceedings*, pages 17–32, 1997.

[GI08] Jens Groth and Yuval Ishai. Sub-linear zero-knowledge argument for correctness of a shuffle. In *Advances in Cryptology - EUROCRYPT 2008, 27th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Istanbul, Turkey, April 13-17, 2008. Proceedings*, pages 379–396, 2008.

[GK96] Oded Goldreich and Hugo Krawczyk. On the composition of zero-knowledge proof systems. *SIAM J. Comput.*, 25(1):169–192, 1996.

[GK03] Shafi Goldwasser and Yael Tauman Kalai. On the (in)security of the fiat-shamir paradigm. In *44th Symposium on Foundations of Computer Science (FOCS 2003), 11-14 October 2003, Cambridge, MA, USA, Proceedings*, pages 102–113, 2003.

[Gol01] Oded Goldreich. *The Foundations of Cryptography - Volume 1, Basic Techniques*. Cambridge University Press, 2001.

[Gro10] Jens Groth. A verifiable secret shuffle of homomorphic encryptions. *J. Cryptology*, 23(4):546–579, 2010.

[JJ00] Markus Jakobsson and Ari Juels. Mix and match: Secure function evaluation via ciphertexts. In *Advances in Cryptology - ASIACRYPT 2000, 6th International Conference on the Theory and Application of Cryptology and Information Security, Kyoto, Japan, December 3-7, 2000, Proceedings*, pages 162–177, 2000.

[JM98] Markus Jakobsson and David M'Raïhi. Mix-based electronic payments. In *Selected Areas in Cryptography '98, SAC'98, Kingston, Ontario, Canada, August 17-18, 1998, Proceedings*, pages 157–173, 1998.

[KMW12] Shahram Khazaei, Tal Moran, and Douglas Wikström. A mix-net from any CCA2 secure cryptosystem. In *Advances in Cryptology - ASIACRYPT 2012 - 18th International Conference on the Theory and Application of Cryptology and Information Security, Beijing, China, December 2-6, 2012. Proceedings*, pages 607–625, 2012.

[Nef01] C. Andrew Neff. A verifiable secret shuffle and its application to e-voting. In *CCS 2001, Proceedings of the 8th ACM Conference on Computer and Communications Security, Philadelphia, Pennsylvania, USA, November 6-8, 2001.*, pages 116–125, 2001.

[NSK04] Lan Nguyen, Reihaneh Safavi-Naini, and Kaoru Kurosawa. Verifiable shuffles: A formal model and a paillier-based efficient construction with provable security. In *Applied Cryptography and Network Security, Second International Conference, ACNS 2004, Proceedings*, pages 61–75, 2004.

[Oka93] Tatsuaki Okamoto. On the relationship among cryptographic physical assumptions. In *Algorithms and Computation, 4th International Symposium, ISAAC '93, Hong Kong, December 15-17, 1993, Proceedings*, pages 369–378, 1993.

[PVW07] Chris Peikert, Vinod Vaikuntanathan, and Brent Waters. A framework for efficient and composable oblivious transfer. *IACR Cryptology ePrint Archive*, 2007:348, 2007.

[PW11] Chris Peikert and Brent Waters. Lossy trapdoor functions and their applications. *SIAM J. Comput.*, 40(6):1803–1844, 2011.

[ST13] Yannick Seurin and Joana Treger. A robust and plaintext-aware variant of signed elgamal encryption. In *Topics in Cryptology - CT-RSA 2013 - The Cryptographers' Track at the RSA Conference 2013, Proceedings*, pages 68–83, 2013.

[TW10] Björn Terelius and Douglas Wikström. Proofs of restricted shuffles. In *Progress in Cryptology - AFRICACRYPT 2010, Third International Conference on Cryptology in Africa, Stellenbosch, South Africa, May 3-6, 2010. Proceedings*, pages 100–113, 2010.

[Wak68] Abraham Waksman. A permutation network. *J. ACM*, 15(1):159–163, 1968.

[Wik09] Douglas Wikström. A commitment-consistent proof of a shuffle. In *Information Security and Privacy, 14th Australasian Conference, ACISP 2009, Brisbane, Australia, July 1-3, 2009, Proceedings*, pages 407–421, 2009.

A Sub-Protocols for ReRand-Switch Gates

We briefly describe the 3-round sigma protocols for proving consistency of individual ReRand-switch gates, which serve as the core sub-protocols of Abe's mix-net.

A.1 Using ElGamal

Recall that a ReRand-switch gate with input x_0, x_1 outputs $y_0 = \mathsf{ReRand}_{\mathsf{pk}}(x_b; r_b)$ and $y_1 = \mathsf{ReRand}_{\mathsf{pk}}(x_{1-b}; r_{1-b})$ where $b \in \{0,1\}$ is a gate control bit, and $r_0, r_1 \leftarrow [q-1]$ are uniform randomness for ElGamal re-randomize algorithm.

Using the Chaum-Pedersen protocol [CP92] for equality of discrete logarithms, one can prove that a pair $(a,b) \in \mathbb{G}^2$ is an ElGamal encryption of zero (i.e., $\exists r : (a,b) = (g^r, h^r)$). Proving correctness of a ReRand-switch gate amounts to proving that Equation (5) holds for either $b = 0$ or $b = 1$.

$$y_0/x_b = (g^{r_b}, h^{r_b}) \quad \text{and} \quad y_1/x_{1-b} = (g^{r_{1-b}}, h^{r_{1-b}}) \tag{5}$$

This can be achieved by proving an OR (via Cramer *et al.* [CDS94]) of ANDs (via straightforward composition) of discrete log equalities. This yields a protocol with the following properties.

Theorem 6 ([CP92, CDS94])**.** *The protocol described above is a* witness indistinguishable*,* honest-verifier zero-knowledge *proof of ElGamal* ReRand*-switch gate correctness, satisfying* special soundness*.*

A.2 Using Lossy ElGamal

In our construction based on lossy ElGamal, correctness of each gate in the Beneš mix will be proved using an OR (via Cramer *et al.* [CDS94]) of ANDs of the following proof for equality of

Lossy ElGamal ciphertexts of Okamoto [Oka93]. (The proof described below actually proves that a Lossy ElGamal ciphertext encrypts 0, which suffices for our goal, simply by taking the new ciphertext formed by component-wise division of the 2 ciphertexts in question).

Statement: $\mathsf{pk} = (g_0, g_1, h_0, h_1), (a, b) \in \mathbb{G}^2$, where allegedly $a = g_0^{r_0} g_1^{r_1}$ and $b = h_0^{r_0} h_1^{r_1}$.
Witness: r_0, r_1.

1. $\mathcal{P}$ samples random $\alpha_0, \alpha_1 \leftarrow [q]$ and sends to $\mathcal{V}$ $c = g_0^{\alpha_0} g_1^{\alpha_1}, d = h_0^{\alpha_0} h_1^{\alpha_1}$.
2. $\mathcal{V}$ samples a random challenge $e \leftarrow [q]$ and sends to $\mathcal{P}$.
3. $\mathcal{P}$ responds with $u_0 = \alpha_0 + e \cdot r_0$ and $u_1 = \alpha_1 + e \cdot r_1$. The verifier accepts iff $g_0^{u_0} g_1^{u_0} = ca^e$ and $h_0^{u_0} h_1^{u_0} = db^e$.

Theorem 7 ([Oka93, CDS94])**.** *The protocol described above is a* witness indistinguishable*,* honest-verifier zero-knowledge *proof of Lossy ElGamal* ReRand*-switch gate correctness, satisfying* special soundness*.*

B Abe's Mix-Net Against Malicious Verifiers

The gate-consistency proof which serves as the core of Abe's mix-net system is known to be *honest-verifier zero knowledge* and *witness indistinguishable*. However, for the case of a malicious verifier, and where the statement has a *unique* witness, no hiding properties are known (indeed, it is known that it *cannot* be zero knowledge, at least with black-box simulation, as the protocol is only 3 rounds [GK96]). Given the state-of-the-art knowledge on this well-studied component, it is possible that an execution of the gate-consistency proof on a pair of ciphertexts of the adversary's choosing (with distinct plaintexts) may *reveal* the bit of whether these ciphertexts were swapped or not during an honest execution of VrfblyMix. A natural question is: Would this leakage inherently break anonymity of the *unmodified* Abe mix-net?

We show that indeed it would. We demonstrate an attack against even the (weaker) notion of anonymity considered in [Abe99, AH01, AI06], in the case that this swap-bit witness information is leaked in gates with two adversarially chosen ciphtertexts. In the attack, the adversary corrupts all but two users, and can predict with higher probability than he should where each honest user's vote was permuted.

In line with our indistinguishability-based notion of anonymity, we show a strategy in which the adversary needs to only corrupt *four* users out of an arbitrary total number n, and (given gate leakage information) can distinguish between the use of two different permutations which agree on corrupted parties.

Both attacks leverage the combinatorial structure of the Beneš network.

B.1 Gate Leakage Would Break [AI06] Anonymity

We demonstrate that such gate leakage in the case of a malicious verifier would result in a break of the (weaker) notion of anonymity considered in [Abe99, AH01, AI06].

Consider a setting with $n \geq 4$ users, with a single uncorrupted mix-server. The attack: Corrupt all but the last two users, $A = [n] \setminus \{n-1, n\}$. For each corrupt user i submit a unique vote value

i. Note that the uncorrupted users are neighbors in the first level of the Beneš network, and in particular that they are necessarily in the same Beneš cycle (as in Waksman [Wak68]).

Now, consider an execution of the Abe mix-net, in which a random permutation π is implemented. By the unique choice of votes of corrupted parties, the adversary can completely identify the permutation π except for the ordering of the two uncorrupted votes. Two different cases may occur:

- The uncorrupted users end in the same gate: i.e., there exists $i \in [n/2]$ for which $\pi(n-1) = 2i-1, \pi(n) = 2i$ or vice versa. In this case, there are 2 degrees of freedom (no attack). However, this occurs with probability less than $\frac{1}{n}$ (this is the probability that the uncorrupted users are neighbors in the permutation, not necessarily in the same output gate).

- The uncorrupted users end in different gates: in this case, the uncorrupted parties are on a Beneš cycle which includes corrupted parties (their neighbors in the output gates), and thus must *necessarily* include a gate with two corrupt parties (namely, there will be at least one additional input gate, which must have this property). This means the switch value of this gate is leaked. However, setting the switch value of any gate within a Beneš cycle, together with the corresponding known permutation information, determines the switch values of all other gates in the same cycle. In particular, this reveals the corresponding ordering of the two uncorrupted parties.

B.2 Gate Leakage Would Break Indistinguishability Anonymity

More in line with the indistinguishability-based notion of anonymity we consider, we next demonstrate a simple permutation distinguishing attack on Abe's mix-net given the above-described gate leakage.

Namely, for any choice of $n = 2^d \geq 8$ users, we present a choice of corruptions $A \subset [n]$, and a pair of permutations $\sigma, \pi \in S_n$ with the following properties:

- $A = \{1, 2, 5, 6\} \subset [n]$ consists of exactly 4 corrupted users. Note that $1, 2$ and $5, 6$, respectively, are neighbors in gate 1 and in gate 3 of the first level of the Beneš network on n users, so that both gates $B_\pi[1, 1]$ and $B_\pi[3, 1]$ would be subject to leakage in the above scenario.

- π and σ agree on the corrupted set A: i.e., $\forall i \in A$, $\pi(i) = \sigma(i)$.

- For *any* Beneš representation B_π of π and B_σ of σ, it holds that $(B_\pi[1, 1], B_\pi[3, 1]) \neq (B_\sigma[1, 1], B_\sigma[3, 1])$. That is, the Beneš control bits for gates 1 and 3 (in the first level of the Beneš network) for permutations π and σ cannot both agree.

This means that if the control bits of "adversarially controlled" Beneš mix gates *are* revealed, then an adversary can distinguish which of these two permutations was used for mixing, even though the permutations differ only on honest votes.

The counterexample permutations are as follows, depicted in Figure 2:

Permutation 1: $\pi \in S_8$ given by $(1)(3)(5)(7)(2864)$. All possible Beneš settings for π require the control bit $B_\pi[1]$ of the first gate in level 1 (dictating swap of inputs 1&2) to *agree* with the control bit $B_\pi[3]$ of the third gate in level 1 (dictating swap of inputs 5&6).

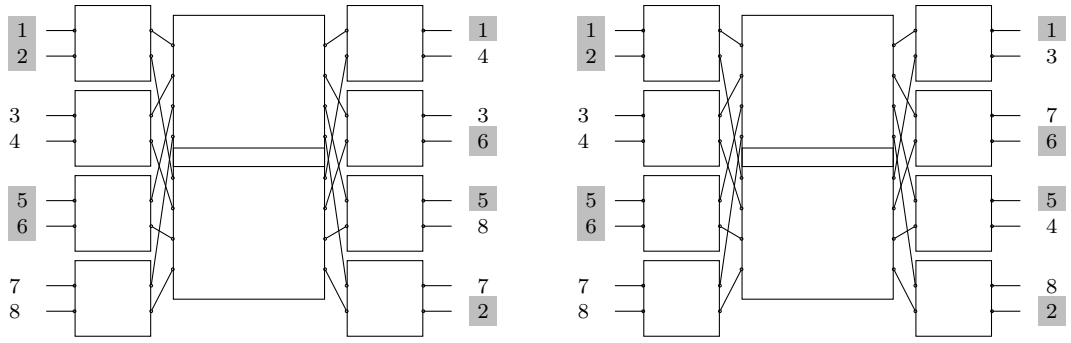

Figure 2: Pair of permutations π (left) and σ (right) for permutation distinguishing attack on Abe's mix-net [Abe99, AH01].

Permutation 2: $\sigma \in S_8$ given by (1)(5)(2873)(46). Here, all possible Beneš settings for π require the control bit $B_\sigma[1]$ to *disagree* with the control bit $B_\sigma[3]$.

Note that both permutations agree on the image of corrupt indices 1,2,5,6 (shaded in Figure 2). When determining possible Beneš control bit settings for a given permutation (as first presented by Waksman [Wak68]), one considers cycles within the Beneš network. For example, for permutation π, starting with input index 1 in gate $[1, 1]$, one first traces to the last-level gate of 1's image (gate $[L, 1]$), sees its output gate neighbor (here, 4), traces back to the input gate of this index 4 (gate $[1, 2]$), sees 4's input gate neighbor (3), tracing to 3's output gate (gate $[L, 2]$), etc.

For both of the chosen permutations π, σ, the Beneš settings result in a single cycle, so that the Beneš settings B_π and B_σ in the first and last level are completely determined by the choice of the first gate setting $B_\pi[1, 1]$ (and $B_\sigma[1, 1]$)

We remark that this conclusion holds even if we were to assume a large honest majority of users, as the same counterexample holds with 4 corrupted users even as the number of overall users n grows arbitrarily large (considering $\pi_d, \sigma_d \in S_8 \times S_{2^d-8}$).

C Omitted Proofs

C.1 Proof of Lossy Abe Anonymity (Theorem 3)

Proof of Theorem 3. The sequence of hybrids is as follows:

Hybrid 0 $\mathcal{H}_0$: We starts with $\mathsf{View}_{\mathcal{V}^*}[D_\lambda^{\mathsf{MixNet},A}((u_i)_{i\in\mathcal{U}_{\bar{A}}})]$. Informally, the relevant parts of the adversary $\mathcal{A}$ view consists of:

$$(u_i)_{i\in\mathcal{U}_{\bar{A}}},\ \mathsf{pk}, (\mathsf{sk}_j)_{j\in\mathcal{S}_A},\ (c_i^0)_{i\in\mathcal{U}_{\bar{A}}},\ \left(\mathsf{view}^{\mathsf{Mix}_j}\right)_{j\in[m]},\ \mathsf{view}^{\mathsf{Dec}},\ \vec{v},$$

corresponding to the (adversarially chosen) input vote vector, public key, secret shares of the secret key, honest-user encrypted votes, the view during the mix-phase of all m mix-servers, the view during the decryption ($\mathsf{Decrypt}$) phase, and the final vector of decrypted (shuffled) votes.

Hybrid 1 $\mathcal{H}_1$: In this hybrid we replace $\mathsf{view}^{\mathsf{Dec}}$ with a simulated view $\mathsf{Sim}^{\mathsf{Dec}}$. Namely, we change

$$\mathsf{View}^{\mathsf{Dec}}_{\mathcal{V}^*}\left[\left(\mathsf{pk}, \vec{c}^m, \vec{v}\right), \left(\mathsf{sk}_j\right)_{j\in[m]}, z\right].$$

with

$$\mathsf{Sim}^{\mathsf{loss}}_{\mathsf{Dec}}\left[\left(\mathsf{pk}, \vec{c}^m, \vec{v}\right), z\right]$$

By the zero-knowledge of $(\mathcal{P}^{\mathsf{loss}}_{\mathsf{Dec}}, \mathcal{V}^{\mathsf{loss}}_{\mathsf{Dec}})$, it holds that $\mathcal{H}_0 \overset{c}{\approx} \mathcal{H}_1$.

Hybrid 2 $\mathcal{H}_2$: In this hybrid we replace the setup phase Setup with a lossy setup $\mathsf{Setup}^{\mathcal{H}_2}$, where[5]

$\mathsf{Setup}(1^\lambda)$:
$(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$
$\mathsf{sk}_1, \ldots, \mathsf{sk}_m \leftarrow \mathsf{SecretShare}(\mathsf{sk})$

$\mathsf{Setup}^{\mathcal{H}_2}(1^\lambda)$:
$\mathsf{pk}_{\mathsf{loss}} \leftarrow \mathsf{KeyGen}_{\mathsf{loss}}(1^\lambda)$
$\mathsf{r}_1, \ldots, \mathsf{r}_m \leftarrow \{0,1\}^\lambda$

This change the view, so instead of $\mathsf{pk}, (\mathsf{sk}_j)_{j\in\mathcal{S}_A}$ we have $\mathsf{pk}_{\mathsf{loss}}, (\mathsf{r}_j)_{j\in\mathcal{S}_A}$.

By the keys indistinguishability of Lossy ElGamal and the pseudorandomness of shares of the secret sharing algorithm, it holds that $\mathcal{H}_1 \overset{c}{\approx} \mathcal{H}_2$.

Hybrid 3 $\mathcal{H}_3$: In this hybrid we use semantic security and change the encrypted user votes from $(c_i^0)_{i\in\mathcal{U}_{\bar{A}}}$ to encryption of the permuted user votes $(c_i^0)_{\sigma(i)\in\mathcal{U}_{\bar{A}}}$. Namely, we change

$$\forall i \in \mathcal{U}_{\bar{A}}: \qquad c_i^0 \leftarrow \mathsf{Enc}_{\mathsf{pk}}(u_i)$$

with

$$\forall i \in \mathcal{U}_{\bar{A}}: \qquad c_i^0 \leftarrow \mathsf{Enc}_{\mathsf{pk}}(u_{\sigma(i)})$$

By the semantic security of Lossy ElGamal, it holds that $\mathcal{H}_2 \overset{c}{\approx} \mathcal{H}_3$.

Remark. The simulator $\mathsf{Sim}^{\mathsf{loss}}_{\mathsf{Dec}}$ still runs given the input $((\mathsf{pk}_{\mathsf{loss}}, \vec{c}^m, \vec{v}), z)$ where $\vec{v}$ is a permutation π of the *original* votes.

Hybrid 4 $\mathcal{H}_4$: Let j^* be an honest mixing server, and let π_j be the selected permutation of server j. Let $\tilde{\pi}_{j^*-1}$ denote the cumulative permutations implemented by mix-servers $1, \ldots, j^*-1$. Note that these permutations are contained as part of the witness of the mix proof and thus can be extracted by the special soundness property of the underlying proof sub-protocols.

In this hybrid we use the WI property, and we change the permutation runs by the honest server j^* to undo the permutation σ. Namely, instead of the permutation π_{j^*}, the honest server j^* runs the permutation $\pi^* := \pi_{j^*} \circ \tilde{\pi}_{j^*-1} \circ \sigma^{-1} \circ \tilde{\pi}^{-1}_{j^*-1}$. Informally, the relevant parts of the adversary view consists of:

$$(u_i)_{i\in\mathcal{U}_{\bar{A}}},\ \mathsf{pk}_{\mathsf{loss}}, (\mathsf{r}_j)_{j\in\mathcal{S}_A},\ (c_i^0)_{\sigma(i)\in\mathcal{U}_{\bar{A}}},\ \mathsf{view}^{\mathsf{Mix}_1}, \ldots, \mathsf{view}^{\mathsf{Mix}_{j^*}}, \ldots, \mathsf{view}^{\mathsf{Mix}_m},\ \mathsf{Sim}^{\mathsf{loss}}_{\mathsf{Dec}},\ \vec{v}$$

[5]We assume that λ is the length of regular share of a key (if the length is $p(\lambda)$, $\mathsf{Setup}^{\mathcal{H}_2}$ samples $\mathsf{r}_i \leftarrow \{0,1\}^{p(\lambda)}$)

where in hybrid $\mathcal{H}_3$ the view $\mathsf{view}^{\mathsf{Mix}_{j^*}}$ is

$$\mathsf{View}_{\mathcal{V}^*}^{\mathsf{Mix}_{j^*}}\left[\left(\mathsf{pk}_{\mathsf{loss}}, \vec{c}^{j^*-1}, \vec{c}^{j^*}\right), \left(B_{\pi_{j^*}}, \hat{R}_0^{j^*}, \hat{R}_1^{j^*}\right), z\right].$$

and in $\mathcal{H}_4$

$$\mathsf{View}_{\mathcal{V}^*}^{\mathsf{Mix}_{j^*}}\left[\left(\mathsf{pk}_{\mathsf{loss}}, \vec{c}^{j^*-1}, \vec{c}^{*}\right), \left(B_{\pi_*}, \hat{R}_0^{*}, \hat{R}_1^{*}\right), z\right].$$

By Claim 1, for any fixed choice of intermediate ciphertext vectors $\vec{c}, (\vec{c}^{(\ell)})_{\ell\in[L]}$ for the Beneš execution, and for any Beneš setting $B_{\pi^*} \in \{0,1\}^{L\times\frac{n}{2}}$, there exist matrices R_0^*, R_1^* of randomness such that R_0^*, R_1^* are consistent re-randomization for B_{π^*}.

By the WI of $(\mathcal{P}_{\mathsf{Gate}}^{\mathsf{loss}}, \mathcal{V}_{\mathsf{Gate}}^{\mathsf{loss}})$ and Claim 1, it holds that $\mathcal{H}_3 \overset{c}{\approx} \mathcal{H}_4$.

Hybrid 5 $\mathcal{H}_5$: In this hybrid we replace the lossy setup phase $\mathsf{Setup}^{\mathcal{H}_2}$ with Setup (i.e., we undo the change made in hybrid 2).

By the keys indistinguishability of Lossy ElGamal and the pseudorandomness of shares of the secret sharing algorithm, it holds that $\mathcal{H}_4 \overset{c}{\approx} \mathcal{H}_5$.

Hybrid 6 $\mathcal{H}_6$: In this hybrid we replace the simulated view $\mathsf{Sim}_{\mathsf{Dec}}^{\mathsf{loss}}$ with an $\mathsf{view}^{\mathsf{Dec}}$ (i.e, we undo the change made in hybrid 1).

By the zero-knowledge of $(\mathcal{P}_{\mathsf{Dec}}^{\mathsf{loss}}, \mathcal{V}_{\mathsf{Dec}}^{\mathsf{loss}})$, it holds that $\mathcal{H}_5 \overset{c}{\approx} \mathcal{H}_6$.

□

C.2 Proof of Injected Abe Anonymity (Lemma 5)

The proof will make repeated use of the following two useful lemmata.

Observe that if the two input ciphertexts x_0, x_1 to a Beneš switch gate encrypt the same plaintext, then any corresponding common input $x = (\mathsf{pk}, x_0, x_1, y_0, y_1)$ for the relation $\mathsf{R}_{\mathsf{Gate}}$ has exactly *two* witnesses of the form (b, r_0, r_1) as in Eq (3). We next define notation for the set of all such instances and witness pairs.

Definition 15 ($\mathsf{SamePlaintext}_\lambda$)**.** For ElGamal encryption $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{ReRand})$, and for security parameter λ, we define the following set

$$\mathsf{SamePlaintext}_\lambda = \left\{ \left(\begin{array}{l} x = (\mathsf{pk}, x_0, x_1, y_0, y_1), \\ w_0 = (0, r_0, r_1), \\ w_1 = (1, r_0 - r, r_1 + r) \end{array} \right) \middle| \begin{array}{ll} \exists u \in M: & \mathsf{pk} \in \mathsf{KeyGen}(1^\lambda)_1, \\ & x_0 \in \mathsf{Enc}_{\mathsf{pk}}(u), \\ & x_1 = \mathsf{ReRand}_{\mathsf{pk}}(x_0; r), \\ & y_0 = \mathsf{ReRand}_{\mathsf{pk}}(x_0; r_0), \\ & y_1 = \mathsf{ReRand}_{\mathsf{pk}}(x_1; r_1) \end{array} \right\}.$$

Lemma 8 ($(\mathcal{P}_{\mathsf{Gate}}, \mathcal{V}_{\mathsf{Gate}})$ Same Plaintext)**.** *For every $(x, w_0, w_1) \in \mathsf{SamePlaintext}_\lambda$, and for any auxiliary input z,*

$$\mathsf{View}_{\mathcal{V}_{\mathsf{Gate}}}[(x, w_0, z)] \overset{c}{\approx} \mathsf{View}_{\mathcal{V}_{\mathsf{Gate}}}[(x, w_1, z)].$$

Proof. Follows by the WI property of the interactive proof system $(\mathcal{P}_{\mathsf{Gate}}, \mathcal{V}_{\mathsf{Gate}})$, as per Theorem 6. □

Lemma 9 (($\mathcal{P}_{\mathsf{Gate}}, \mathcal{V}_{\mathsf{Gate}}$) Same Permutation)**.** *For every $w \in \{0,1\} \times [q-1]^2$ and non-uniform PPT $\mathcal{A}$, the interactive proof system ($\mathcal{P}_{\mathsf{Gate}}, \mathcal{V}_{\mathsf{Gate}}$) for the gate-consistency relation $\mathsf{R}_{\mathsf{Gate}}$ (eq. (3)) in Abe Mix-net is distributional witness-indistinguishability (D-WI) with respect to the following two distribution ensembles*

$$D_\lambda = (D_\lambda^{\mathsf{Gate},w,\mathcal{A}}(0)) \qquad D'_\lambda = (D_\lambda^{\mathsf{Gate},w,\mathcal{A}}(1)),$$

where $D_\lambda^{\mathsf{Gate},w,\mathcal{A}}$ is as in Definition 16.

Definition 16 ($D_\lambda^{\mathsf{Gate},w,\mathcal{A}}$)**.** For any $w = (b, r_0, r_1) \in \{0,1\} \times [q-1]^2$, (adversarial) non-uniform PPT algorithm $\mathcal{A}$, and security parameter $\lambda \in \mathbb{N}$, we define the following distribution $D_\lambda^{\mathsf{Gate},w,\mathcal{A}}$ as follows:

$D_\lambda^{\mathsf{Gate},w,\mathcal{A}}(u)$:

Input Plaintext $u \in M$

- Sample $\mathsf{pk} \leftarrow \mathsf{KeyGen}(1^\lambda)$
- Sample $c_0 \leftarrow \mathsf{Enc}_{\mathsf{pk}}(u)$
- Obtain $(c_1, \beta, z) \leftarrow \mathcal{A}(\mathsf{pk}, c_0)$
- Set $x_0 = c_\beta$ and $x_1 = c_{\neg\beta}$
- Set $y_b = \mathsf{ReRand}_{\mathsf{pk}}(x_0; r_0)$ and $y_{\neg b} = \mathsf{ReRand}_{\mathsf{pk}}(x_1; r_1)$

Output $(X_\lambda = (\mathsf{pk}, x_0, x_1, y_0, y_1), W_\lambda = w, Z_\lambda = z)$

Proof. Suppose for contradiction there exists $w = (b, r_0, r_1)$, non-uniform PPT algorithm $\mathcal{A}$, non-uniform (adversarial) verifier $\mathcal{V}^*$, and non-uniform PPT distinguisher Adv who successfully distinguishes $\mathsf{View}_{\mathcal{V}^*}[D_\lambda]$ and $\mathsf{View}_{\mathcal{V}^*}[D'_\lambda]$ with non-negligible advantage ϵ. We construct an adversary Adv' who breaks the semantic security of ElGamal.

Adv' has $((b, r_0, r_1), \mathcal{A}, \mathcal{V}^*, Adv)$ hardcoded. In the semantic security challenge, Adv' receives a public key pk and a challenge ciphertext c^* which encrypts either 0 or 1. Adv' executes $(c_1, \beta, z) \leftarrow \mathcal{A}(\mathsf{pk}, c^*)$, sets $x := (\mathsf{pk}, c_\beta, c_{\neg\beta}, \mathsf{ReRand}_{\mathsf{pk}}(x_b; r_b), \mathsf{ReRand}_{\mathsf{pk}}(x_{\neg b}; r_{\neg b}))$ and simulates the interactive proof
($\mathcal{P}_{\mathsf{Gate}}, \mathcal{V}_{\mathsf{Gate}}$) with $\mathcal{V}^*$ to generate $\mathsf{view} \leftarrow \mathsf{View}_{\mathcal{V}^*}[(x, w, z)]$. Finally, Adv' outputs $Adv(\mathsf{view})$.

If c^* is an encryption of 0, then $\mathsf{View}_{\mathcal{V}^*}[(x, w, z)]$ is identically distributed to D_λ, whereas if c^* is an encryption of 1, then it is identically distributed to D'_λ. The lemma follows.

□

Using the lemmata 8 and 9, we now prove Lemma 5.

Proof. Let $\mathcal{A}$ be a non-uniform PPT ciphertexts generator, and let π_0 and π_1 be two permutations in S_n. We show that for any PPT interactive machine $\mathcal{V}^*$: $\mathsf{View}_{\mathcal{V}^*}[D_\lambda^{\mathsf{Mix},\mathcal{A}}(\pi_0)] \stackrel{c}{\approx} \mathsf{View}_{\mathcal{V}^*}[D_\lambda^{\mathsf{Mix},\mathcal{A}}(\pi_1)]$

(where $D_\lambda^{\mathsf{Mix},\mathcal{A}}$ is as in Definition 14), by a sequence of the following hybrids. We denote by $L = N\log(N)$ (number of levels), $N = 2n$ (number of users),

$$\left(X_\lambda^0 = \left(\mathsf{pk}, \vec{C}, \vec{C'}\right), W_\lambda^0 = \left(B_{\pi_0^{\mathsf{new}}}, \hat{R}_0^0, \hat{R}_1^0\right), Z_\lambda^0 = \vec{z}\right) \leftarrow D_\lambda^{\mathsf{Mix},\mathcal{A}}(\pi_0)$$

and similarly

$$\left(X_\lambda^1 = \left(\mathsf{pk}, \vec{C}, \vec{C''}\right), W_\lambda^1 = \left(B_{\pi_0^{\mathsf{new}}}, \hat{R}_0^1, \hat{R}_1^1\right), Z_\lambda^1 = \vec{z}\right) \leftarrow D_\lambda^{\mathsf{Mix},\mathcal{A}}(\pi_1)$$

Hybrid 0 $\mathcal{H}_0$: $\mathsf{View}_{\mathcal{V}^*}[D_\lambda^{\mathsf{Mix},\mathcal{A}}(\pi_0)]$

Hybrid 1 $\mathcal{H}_1$: In this hybrid we change the input ciphertexts at the even positions to be

$$\forall i \in [n]: \quad C_{2i} \leftarrow \mathsf{ReRand}_{\mathsf{pk}}(C'_{2i-1})$$

instead of an encryptions of $\perp$. Namely, $\forall i \in [n] : C_{2i} \leftarrow \mathsf{Enc}_{\mathsf{pk}}(\perp)$. After this change, the two input ciphertexts $C_{2i-1}^{L-1}, C_{2i}^{L-1}$ (i.e. two input ciphertexts of every gate at the *last level* in the Beneš network) have the same plaintexts. This is because π_0^{new} acts as the identity on the even positions. By Lemma 9 (same permutation):

$$\mathcal{H}_0 \overset{c}{\approx} \mathcal{H}_1$$

Hybrid 2 $\mathcal{H}_2$: In this hybrid we change the last level control bits from $B_{\pi_0^{\mathsf{new}}}[L]$ to $B_{\pi_1^{\mathsf{new}}}[L]$. By Lemma 8 (same plaintext):

$$\mathcal{H}_1 \overset{c}{\approx} \mathcal{H}_2$$

Hybrid 3 $\mathcal{H}_3$: In this hybrid we change the input ciphertexts at the even positions to be

$$\forall i \in [n]: \quad C_{2i} \leftarrow \mathsf{ReRand}_{\mathsf{pk}}(C_{2i-1})$$

instead of $C_{2i} \leftarrow \mathsf{ReRand}_{\mathsf{pk}}(C'_{2i-1})$ for all $i \in [n]$. After this change, the two input ciphertexts C_{2i-1}, C_{2i} (i.e. two input ciphertexts of every gate at the *first level* in the Beneš network) have the same plaintexts. By Lemma 9 (same permutation):

$$\mathcal{H}_2 \overset{c}{\approx} \mathcal{H}_3$$

Hybrid 4 $\mathcal{H}_4$: In this hybrid we change the first level control bits from $B_{\pi_0^{\mathsf{new}}}[1]$ to all be one (i.e., all even positions are going to the $(d-1)$-dimensional Beneš Network B_{up}) . By Lemma 8 (same plaintext):

$$\mathcal{H}_3 \overset{c}{\approx} \mathcal{H}_4$$

Hybrid 5 $\mathcal{H}_5$: In this hybrid we change the input ciphertexts at the even positions to be

$$\forall i \in [n]: \quad C_{2i} \leftarrow \mathsf{Enc}_{\mathsf{pk}}(\perp)$$

instead of $C_{2i} \leftarrow \mathsf{ReRand}_{\mathsf{pk}}(C_{2i-1})$ for all $i \in [n]$. After this change, all the ciphertexts in the $(d-1)$-dimensional Beneš Network B_{up} have the same plaintext (which is $\perp$). By Lemma 9 (same permutation):

$$\mathcal{H}_4 \overset{c}{\approx} \mathcal{H}_5$$

Hybrid 6 $\mathcal{H}_6$: In this hybrid we change the $(d-1)$-dimensional Beneš Network B_{up} control bits from $B_{\pi_0^{\mathsf{new}}}[i][\ell]$ to $B_{\pi_1^{\mathsf{new}}}[i][\ell]$ (for every $i \in [n/4], \ell \in \{2, \ldots, L-1\}$). By Lemma 8 (same plaintext):

$$\mathcal{H}_5 \overset{c}{\approx} \mathcal{H}_6$$

Hybrid 7 $\mathcal{H}_7$: In this hybrid we repeat hybrid 3 and change injected ciphertexts in such a way that the two input ciphertexts of every gate at the first level in the Beneš network have the same plaintexts. By Lemma 9 (same permutation):

$$\mathcal{H}_6 \overset{c}{\approx} \mathcal{H}_7$$

Hybrid 8 $\mathcal{H}_8$: In this hybrid we change the first level control bits from $B_{\pi_0^{\mathsf{new}}}[1]$ to all be zero (i.e., all even positions are going to the $(d-1)$-dimensional Beneš Network $\mathsf{B}_{\mathsf{down}}$) . By Lemma 8 (same plaintext):

$$\mathcal{H}_7 \overset{c}{\approx} \mathcal{H}_8$$

Hybrid 9 $\mathcal{H}_9$: In this hybrid we repeat hybrid 5 and change the input ciphertexts at the even positions to be

$$\forall i \in [n]: \quad C_{2i} \leftarrow \mathsf{Enc}_{\mathsf{pk}}(\perp)$$

instead of $C_{2i} \leftarrow \mathsf{ReRand}_{\mathsf{pk}}(C_{2i-1})$ for all $i \in [n]$. After this change, all the ciphertexts in the $(d-1)$-dimensional Beneš Network $\mathsf{B}_{\mathsf{down}}$ have the same plaintext (which is $\perp$). By Lemma 9 (same permutation):

$$\mathcal{H}_8 \overset{c}{\approx} \mathcal{H}_9$$

Hybrid 10 $\mathcal{H}_{10}$: In this hybrid we change the $(d-1)$-dimensional Beneš Network $\mathsf{B}_{\mathsf{down}}$ control bits from $B_{\pi_0^{\mathsf{new}}}[i][\ell]$ to $B_{\pi_1^{\mathsf{new}}}[i][\ell]$ (for every $i \in \{n/4+1, \ldots n/2\}, \ell \in \{2, \ldots, L-1\}$). By Lemma 8 (same plaintext):

$$\mathcal{H}_9 \overset{c}{\approx} \mathcal{H}_{10}$$

Hybrid 11 $\mathcal{H}_{11}$: In this hybrid we repeat hybrid 3 (and 7) and change injected ciphertexts in such a way that the two input ciphertexts of every gate at the *first level* in the Beneš network have the same plaintexts. By Lemma 9 (same permutation):

$$\mathcal{H}_{10} \overset{c}{\approx} \mathcal{H}_{11}$$

Hybrid 12 $\mathcal{H}_{12}$: In this hybrid we change the first level control bits from $B_{\pi_0^{\mathsf{new}}}[1]$ to $B_{\pi_1^{\mathsf{new}}}[1]$. By Lemma 8 (same plaintext):

$$\mathcal{H}_{11} \overset{c}{\approx} \mathcal{H}_{12}$$

Hybrid 13 $\mathcal{H}_{13}$: In this hybrid we repeat hybrid 5 (and 9) and change the input ciphertexts at the even positions to be

$$\forall i \in [n]: \quad C_{2i} \leftarrow \mathsf{Enc}_{\mathsf{pk}}(\perp)$$

instead of $C_{2i} \leftarrow \mathsf{ReRand}_{\mathsf{pk}}(C_{2i-1})$ for all $i \in [n]$. By Lemma 9 (same permutation):

$$\mathcal{H}_{12} \overset{c}{\approx} \mathcal{H}_{13}$$

□