Indifferentiability of Confusion-Diffusion Networks

Yevgeniy Dodis[1]*, Martijn Stam[2], John Steinberger[3]**, and Liu Tianren[4]

dodis@cs.nyu.edu, csxms@bristol.ac.uk, jpsteinb@gmail.com, liutianren@gmail.com

Abstract. We show the first positive results for the indifferentiability security of the confusion-diffusion networks (which are extensively used in the design of block ciphers and hash functions). In particular, our result shows that a constant number of confusion-diffusion rounds is sufficient to extend the domain of a public random permutation.

1 Introduction

In this work we simultaneously address the following two questions:

– **Question 1**: secure domain extension of a *public* random permutation.
– **Question 2**: theoretical soundness of Shannon's (or Feistel's) confusion-diffusion paradigm.

Domain Extension of RPs. The question of domain extension of various cryptographic primitives, such as encryption, signatures, message authentication codes, pseudorandom functions (PRFs), pseudorandom permutations (PRPs), etc., is one of the fundamental questions in cryptography.

In this paper we address a similar question for a *public* random permutation. Namely, given one (or a constant number of) n-bit random permutation(s) $P : \{0,1\}^n \to \{0,1\}^n$, and a number $w \geq 2$, build a wn-bit random permutation $\Pi : \{0,1\}^{wn} \to \{0,1\}^{wn}$. This question is clearly natural and interesting it is own right, but also seems extremely relevant in practice. Indeed, the random permutation model (RPM) has recently received a lot of attention [2,9,23,26], starting to "compete with" and perhaps even "overtake" the more well known random oracle model (ROM) and the ideal cipher model (ICM). Aside from elegance, one of the reasons for this renewed attention comes from the fact that one can abstract the design of both the block-cipher standard AES and the new SHA-3 standard Keccak as being in the RPM. Namely, AES can be viewed as a 10-round *key-alternating* cipher applied to a concrete ("random-looking") permutation, while SHA-3 can be viewed as applying a "sponge" mode of operation [2] to a similarly "random-looking" permutation. In fact, in his invited talk at Eurocrypt'13, the designer of both AES and SHA-3 Joan Daemen claimed that the RPM is much closer to the existing practice of designing hash functions and block ciphers than either the ROM or ICM, challenging the cryptographic community to switch to the RPM!

Of course, one must now build those "random looking permutations" Π on relatively large domains (perhaps from 128 bits, like AES-128, to 1600 bits, like Keccak, or even longer). In practice, we have two well-known methods for accomplishing such a goal. The first method is based on applying several rounds of the Feistel network to some (not necessarily invertible) round functions. In our (public) setting, this method was theoretically analyzed only recently by Holenstein et al. [15] (building on an earlier work of [7]), who showed that a 14-round Feistel network is indeed

* Partially supported by gifts from VMware Labs and Google, and NSF grants 1319051, 1314568, 1065288, 1017471.
** Supported by National Basic Research Program of China Grant 2011CBA00300, 2011CBA00301, the National Natural Science Foundation of China Grant 61033001, 61361136003, and by the China Ministry of Education grant number 20121088050.

sufficient for building a random permutation (RP), provided the round functions are modeled as (easily made) independent random oracles (ROs). Although very important in theory (i.e., showing the equivalence between ROM and RPM), this method does not seem to be used in practice, as it appears almost as hard, — if not *harder*, — to design "random-looking" non-invertible round functions on large domains as it is to design the desired random-looking permutation Π.

Confusion-Diffusion Paradigm. Instead, practitioners use the second method, — the *confusion-diffusion* (CD) paradigm,[1] — which directly connects our motivating Questions 1 and 2. The idea of CD goes back to the seminal paper of Feistel [11] and even[2] back to Shannon [25]. Abstractly, one splits the input x to Π into several shorter blocks $x_1 \ldots x_w$, and then alternates the following two steps for several rounds: (a) *Confusion*, which consists of applying some fixed short permutations $P_1 \ldots P_w$ (called *S-boxes*) to $x_1, \ldots x_w$; and (b) *Diffusion*, which consists of applying some "mixing" non-cryptographic permutation $\pi(y_1 \ldots y_w)$ (typically, carefully chosen linear function, sometimes also called *D-box*) to the results $y_1 \ldots y_w$ of step (a).

Despite its extensive use in practice, the CD paradigm received extremely little attention from the cryptographic community. A notable exception is a beautiful work of Miles and Viola [20], who only looked at the secret-key setting — where the permutations $P_1, \ldots, P_w$ are secret — and also primarily considered the "weaker-than-indistinguishability" properties which can be proven about CD (and, more generally, SPN networks). In contrast, we are interested in the public setting, where the permutations P_i are modeled as RPs, and seek to examine the *indifferentiability* properties [5,18] of the CD paradigm. This leads us to the following more precise reformulation of our motivating Questions 1 and 2:

– **Main question:** *Analyze indifferentiability of the confusion-diffusion paradigm as a way to extend the domain of a (constant number of) random permutation(s).* More precisely, for how many rounds r, and under what conditions on the D-boxes $\pi_1 \ldots \pi_r$, is the r-round CD paradigm indifferentiable from an nw-bit random permutation Π?

Before presenting our results, we make a few remarks. First, we will model the "small permutations" P_i as both random and independent. The independence assumption is crucially used in our current proofs, but does not appear necessary. Unfortunately, the proofs we have are already extremely involved, so we feel this initial simplification is justified. We notice similar abstractions are made by other papers in the area (including the seminal Luby-Rackoff paper [17]), though one hopes it might be lifted in future work.

As for modeling P_i as random, it seems inherent if we want to build a random permutation Π; e.g., we cannot build it from "nothing", and it seems unlikely that any weaker assumption on the P_i will work. However, it does come with an important caveat: the best security bound ε we can naturally get with this approach will certainly be $\varepsilon \gg 2^{-n}$, where n is the domain of the S-boxes P_i. In practice, however, the S-boxes use a very small value of n (e.g., $n = 8$ for the AES), partly so that S-boxes can be easily and efficiently implemented as lookup tables. With such a small value of n, however, our bounds appear "practically meaningless", irrespective of the number of

[1] This is closely related to the substitution-permutation network (SPN) paradigm. Historically, though, the term SPN usually refers to the design of block ciphers as opposed to a single permutation, where one also XORs some key material in between successive CD rounds. To avoid confusion, we will stick with the term CD and not use the term SPN.

[2] Shannon [25] introduces "confusion" and "diffusion" into the cryptographic lexicon while Feistel [11] articulates the modern notion of a confusion-diffusion network, crediting Shannon with inspiration. There are some notable gaps between Shannon and the modern viewpoint. In particular Shannon does not seem to view confusion as a local operation, nor does he advocate repeatedly alternating steps of "confusion" and "diffusion". Instead, Shannon seems to view confusion and diffusion as globally desirable attributes of a cryptographic mixing operation.

queries q made by the attacker. This means that none of our results would be directly applicable to any of the "practical" permutations Π used in the existing hash functions and block ciphers. Still, we believe establishing "structural soundness" of the CD paradigm is an important conceptual contribution—and an overdue sanity check—even with this serious (and inherent) limitation.

Our Results. We give a sequence of results establishing the soundness of the CD paradigm as a method for domain-extension of random permutations. These results are discussed in detail in Section 3, and summarized in Theorem 1. Here we only mention a few highlights. We establish different indifferentiability bounds for various number of rounds r, from as low as $r = 5$ and as high as $r = 11$. All these bounds critically depend on some *purely combinatorial* properties of the diffusion permutations π_i (which are described in Section 2), and unsurprisingly become potentially better if we allow more rounds. In particular, while our 5-round result inherently contains terms of the order $q^{w^2}/2^n$ and also requires nonlinear diffusion permutations, our 11-round result achieves (modulo the problem of giving explicit constructions of diffusion permutations that achieve certain combinatorial properties that are provably achieved by random permutations) "birthday" security $O(q^2/2^n)$ and even use all linear diffusion permutations π_i (although this "convenience" seems to degrade the security to about $O(q^4/2^n)$).

We also believe that some of the combinatorial properties that we discovered in the course of our analysis (e.g., something we call "conductance") are very natural and interesting in their own right, and could be fruitfully used by the future work in the area.

Other Related Work. The question of domain extension ideal primitives was considered by [5, 19] for the setting of public random functions (ROM), and by [6] for the setting of block ciphers (ICM). While none if these domain extensions directly apply to the RPM (e.g., the result of [6] crucially relies of the existence of a key for the "small ideal cipher"), they can be composed with the series of results showing the equivalence between RPM, ICM and ROM [1,5,7–10,15] to achieve various theoretical domain extension methods in the RPM. For example, one can get a "small RO" from "small RP" [8,9], extend domain of RO [5], and apply the 14-round Feistel construction to get a large-domain RP [15] (many other combinations of prior results will suffice as well). However, all such combinations of prior work will be *much* less efficient (and elegant) than our natural construction, and, more importantly, such results will not correspond to the way random permutations are built in real life.

Finally, domain extension of *secret-key* random permutations is well studied: examples include PEP [3], XCB [13], HCTR [29], HCH [4] and TET [14] (and even the original Feistel constructions [17,21] could be viewed as domain doubling techniques in this setting). However, it is easy to see that none of those constructions provide the indifferentiability property in the public permutation setting.

2 Definitions

Basic Notations. We write $[w]$ for the set of integers $\{1, \dots, w\}$. Vectors in $\mathbb{F}^w$, where $\mathbb{F}$ denotes a finite field, are written in bold letters such as $\mathbf{x}$, $\mathbf{y}$. The i-th entry of $\mathbf{x}$, $i \in [w]$, is written $\mathbf{x}[i]$.

Except in Appendices A and B, where for the sake of conceptual correctness we work with an arbitrary field $\mathbb{F}$, we will set $\mathbb{F} = \mathrm{GF}(2^n)$. Then elements in $\mathbb{F}$ can (under some representation) be identified with n-bit strings, and moreover a string in $\{0,1\}^{wn}$ can be identified with an element $\mathbf{x} \in \mathbb{F}^w$.

Random Permutations and Ideal Primitives. In this paper, the only ideal primitives under consideration are random permutations. A random permutation from $\{0,1\}^k$ to $\{0,1\}^k$ is a permutation drawn uniformly at random from the set of all permutations from $\{0,1\}^k$ to $\{0,1\}^k$. Such a permutation is implemented as an oracle accessible to a specified subset of the parties performing a security experiment. In our setting all parties also have access to the inverse oracle (i.e., implementing the inverse of the random permutation).

Confusion-Diffusion Networks. Fix integers $w, n, r \in \mathbb{N}$. Let

$$\mathcal{P} = \{P_{i,j} : (i,j) \in [r] \times [w]\}$$

be an array of rw permutations from $\{0,1\}^n$ to $\{0,1\}^n$, i.e., $P_{i,j}$ is a permutation from $\{0,1\}^n$ to $\{0,1\}^n$ for each $i \in [r]$ and each $j \in [w]$. Also let

$$\overline{\pi} = (\pi_1, \ldots, \pi_{r-1})$$

be an arbitrary sequence of $r-1$ permutations, each from $\{0,1\}^{wn}$ to $\{0,1\}^{wn}$.

Given $\mathcal{P}$ and $\mathbf{x} \in \{0,1\}^{wn}$ we let

$$P_i(\mathbf{x})$$

denote the value in $\{0,1\}^{wn}$ obtained by applying the permutations $P_{i,1}, \ldots, P_{i,w}$ blockwise to $\mathbf{x}$. In other words, $P_i : \{0,1\}^{wn} \to \{0,1\}^{wn}$ is defined by setting

$$P_i(\mathbf{x})[j] = P_{i,j}(\mathbf{x}[j])$$

for all $j \in [w]$. It is obvious that P_i is a permutation of $\{0,1\}^{wn}$.

Given $\mathcal{P}$ and $\overline{\pi}$, we define the permutation $P = P[\mathcal{P}, \overline{\pi}]$ from $\{0,1\}^{wn}$ to $\{0,1\}^{wn}$ as the composition

$$P[\mathcal{P}, \overline{\pi}] = P_r \circ \pi_{r-1} \circ \ldots \circ P_2 \circ \pi_1 \circ P_1.$$

I.e.,

$$P[\mathcal{P}, \overline{\pi}](\mathbf{x}) = P_r(\pi_{r-1}(\ldots P_2(\pi_1(P_1(\mathbf{x})))\ldots))$$

for $\mathbf{x} \in \{0,1\}^{wn}$. We call $P[\mathcal{P}, \overline{\pi}]$ the *confusion-diffusion network* built from $\mathcal{P}$ and $\overline{\pi}$. The permutations in $\mathcal{P}$ are variously called the *confusion permutations* or *S-boxes*. The permutations in $\overline{\pi}$ are variously called the *diffusion permutations* or *D-boxes*.

The values n, w and r will be called the *wire length*, the *width* and the *number of rounds* respectively.

In practice, the S-boxes are implemented by "convoluted" or "random-like" permutations while the D-boxes are implemented by "easy" (typically linear) permutations that are cryptographically weak. In our indifferentiability model, described next, the S-boxes are modeled as random permutations while the D-boxes are publically fixed parameters of the network.

Indifferentiability. Let C be a construction making calls to an ideal set of primitives $\mathcal{P}$, which we notate as $C^{\mathcal{P}}$. Let $\mathcal{Z}$ be an ideal primitive with the same interface as $C^{\mathcal{P}}$ (e.g., $\mathcal{Z}$ is a random permutation if $C^{\mathcal{P}}$ implements a permutation). Indifferentiability is meant to capture the intuitive notion that the construction $C^{\mathcal{P}}$ is "just as good" as $\mathcal{Z}$, in some precise sense. The definition involves a simulator:

Definition 1. *An (oracle) circuit C with access to a set of ideal primitives $\mathcal{P}$ is (t_S, q_S, ε)-indifferentiable from an ideal primitive $\mathcal{Z}$ if there exists a simulator S such that*

$$\Pr\left[D^{C^{\mathcal{P}},\mathcal{P}} = 1\right] - \Pr\left[D^{\mathcal{Z},S^{\mathcal{Z}}}\right] \leq \varepsilon$$

for every distinguisher D making at most q_0 queries to its oracles, and such that S runs in total time t_S and makes at most q_S queries to $\mathcal{Z}$. Here t_S, q_S and ε are functions of q_0.

We note that in the "real world" D has oracle access to the construction $C^{\mathcal{P}}$ as well as to the primitives $\mathcal{P}$; in the "ideal world" $C^{\mathcal{P}}$ is replaced by the ideal primitive $\mathcal{Z}$ and the ideal primitives $\mathcal{P}$ are replaced by the simulator S. Thus, S's job is to make $\mathcal{Z}$ look like $C^{\mathcal{P}}$ by inventing "answers that fit" for D's queries to the primitives in $\mathcal{P}$. For this, S requires query access to $\mathcal{Z}$ (notated as $S^{\mathcal{Z}}$); on the other hand, S does not get to see which queries D is making to $\mathcal{Z}$.

We emphasize that D is information-theoretic: there is no hardness assumption. What is at stake is the statistical distance between two worlds with two different underlying sources of randomness ($\mathcal{P}$ versus $\mathcal{Z}$).

Informally, $C^{\mathcal{P}}$ is *indifferentiable* from $\mathcal{Z}$ if it is (t_S, q_S, ε)-indifferentiable for "reasonable" values of (t_S, q_S, ε). An essential composition theorem [5,18] states that any cryptosystem that is secure when implemented with $\mathcal{Z}$ remains secure if $\mathcal{Z}$ is replaced with $C^{\mathcal{P}}$, if $C^{\mathcal{P}}$ is indifferentiable from $\mathcal{Z}$. However, the class of adversaries with respect to which the cryptosystem's security is defined must be a class that is large enough to accomodate the simulator S from Definition 1. See, e.g., [22] for a dramatic example in which indifferentiability fails completely.

The value q_S in Definition 1 is called the *query complexity* of the simulator. Having a small value of q_S is important for "quality of composition", i.e., to avoid an unacceptable watering-down of the security bounds when the afore-mentioned composition theorem is applied; ideally q_S should be as close to q_0 as possible.

In our setting "$\mathcal{P}$ will be $\mathcal{P}$" (i.e., the set of ideal primitives $\mathcal{P}$ will be the set of wr independent random permutations discussed in the previous subsection), while $C^{\mathcal{P}}$ will be $P[\mathcal{P}, \overline{\pi}]$. (As explained, the diffusion permutations $\overline{\pi}$ are a fixed, publically known parameter of the construction.) Consequently, $\mathcal{Z}$ (matching $C^{\mathcal{P}}$'s syntax) will be a random permutation from $\{0,1\}^{wn}$ to $\{0,1\}^{wn}$. Like all permutation oracles, $\mathcal{Z}$ can be queried in both forward and backward directions.

Combinatorial Properties of the Diffusion Permutations. Our main result is of the type "for a certain number of rounds and for diffusion permutations $\overline{\pi}$ with certain combinatorial properties for certain π_i's, $P[\mathcal{P}, \overline{\pi}]$ is xyz-indifferentiable from a random permutation $\mathcal{Z}$". Here we define some of the "combinatorial properties" involved.

Properties will be defined unidirectionally: π might satisfy a property while π^{-1} doesn't.

Given $\pi : \{0,1\}^{wn} \to \{0,1\}^{wn}$, a vector $\mathbf{x} \in \{0,1\}^{wn}$ and indices $j, j' \in [w]$, we let

$$\pi^{\mathbf{x}}_{j,j'} : \{0,1\}^n \to \{0,1\}^n$$

be the function from $\{0,1\}^n$ to $\{0,1\}^n$ obtained by restricting the i-th block of input of π, $i \neq j$, to $\mathbf{x}[i]$, by replacing $\mathbf{x}[j]$ with the input $x \in \{0,1\}^n$, and by considering only the j'-th block of output. (The value $\mathbf{x}[j]$ being, thus, immaterial to $\pi^{\mathbf{x}}_{j,j'}$, since it is replaced by the input.)

We define

$$\mathsf{MaxPreim}(\pi) = \max_{\mathbf{x},j,h,y} |\{x \in \{0,1\}^n : \pi^{\mathbf{x}}_{j,h}(x) = y\}|$$

and

$$\mathsf{MaxColl}(\pi) = \max_{\mathbf{x},\mathbf{x}',j,h} |\{x \in \{0,1\}^n : \pi^{\mathbf{x}}_{j,h}(x) = \pi^{\mathbf{x}'}_{j,h}(x)\}|$$

where the latter maximum is taken over all tuples $\mathbf{x}$, $\mathbf{x}'$, j, h such that $\mathbf{x}[j'] \neq \mathbf{x}'[j']$ for some $j' \neq j$. Then by definition

$$\Pr_x[\pi^{\mathbf{x}}_{j,h}(x) = y] \leq \frac{\mathsf{MaxPreim}(\pi)}{2^n}$$

for all $\mathbf{x} \in \{0,1\}^{wn}$, $y \in \{0,1\}^n$, and $j, h \in [w]$, where the probability is computed over a uniform choice of $x \in \{0,1\}^n$, and

$$\Pr_x[\pi^{\mathbf{x}}_{j,h}(x) = \pi^{\mathbf{x}'}_{j,h}(x)] \leq \frac{\mathsf{MaxColl}(\pi)}{2^n}$$

for all $\mathbf{x}$, $\mathbf{x}' \in \{0,1\}^{wn}$, $j, h \in [w]$, such that $\mathbf{x}[j'] \neq \mathbf{x}'[j']$ for at least one $j' \neq j$. For lack of better terminology, we refer to $\mathsf{MaxPreim}(\pi)$ as the *entry-wise randomized preimage resistance* of π and to $\mathsf{MaxColl}(\pi)$ as the *entry-wise randomized collision resistance* of π.

Small values of $\mathsf{MaxPreim}(\pi)$ and of $\mathsf{MaxColl}(\pi)$ are better. It is easy to construct permutations with $\mathsf{MaxPreim}(\pi) = 1$ (which is optimal): simply use a linear permutation $\pi : \mathrm{GF}(2^n)^w \to \mathrm{GF}(2^n)^w$ whose associated matrix (a $w \times w$ matrix with entries in $\mathrm{GF}(2^n)$) has all nonzero entries. Constructing permutations with small values of $\mathsf{MaxColl}(\pi)$ is more tricky. In Appendix C we show how to construct permutations that simultaneously achieve small entry-wise randomized preimage resistance and small entry-wise randomized collision resistance (of order $O(w)$ each).

Conductance. Another combinatorial metric that we identify for a permutation $\pi : \{0,1\}^{wn} \to \{0,1\}^{wn}$ is the so-called "conductance" of the permutation. This metric plays a key role in some of our simulators for improving the security bound and reducing the query complexity.

The conductance Cond_π of a permutation $\pi : (\{0,1\}^n)^w \to (\{0,1\}^n)^w$ is a function of q defined by

$$\mathsf{Cond}_\pi(q) = \max_{\substack{U_1,\ldots,U_w,V_1,\ldots,V_w \subseteq \{0,1\}^n \\ |U_1|=\cdots=|V_w|=q}} |\{(\mathbf{x},\mathbf{y}) : \mathbf{y} = \pi(\mathbf{x}), \mathbf{x}[j] \in U_j, \mathbf{y}[j] \in V_j, 1 \le j \le w\}|.$$

One can observe that w is a hidden parameter of the conductance (e.g., the conductance of a permutation $\pi : \{0,1\}^{1024} \to \{0,1\}^{1024}$ isn't defined on its own without knowing w) and that

$$q \le \mathsf{Cond}_\pi(q) \le q^w$$

for any permutation π, $0 \le q \le 2^n$. Indeed, $q \le \mathsf{Cond}_\pi(q)$ since we can always choose q distinct vectors $\mathbf{x}_1, \ldots, \mathbf{x}_q \in \{0,1\}^{wn}$ and set

$$U_j = \{\mathbf{x}_k[j] : 1 \le k \le q\}$$
$$V_j = \{\mathbf{y}_k[j] : 1 \le k \le q\}$$

where $\mathbf{y}_k = \pi(\mathbf{x}_k)$. On the other hand $\mathsf{Cond}_\pi(q) \le q^w$ since the number of vectors $\mathbf{x}$ such that $\mathbf{x}[j] \in U_j$ for $1 \le j \le w$ is at most $|U_1 \times \cdots \times U_w| = q^w$.

For technical reasons we also define a notion of *all-but-one conductance*, which is essentially the same as conductance but where one coordinate position in either the input or output is ignored. The all-but-one conductance of a permutation π at q queries is denoted $\mathsf{aboCond}_\pi(q)$.

Formally, given a permutation $\pi : (\{0,1\}^n)^w \to (\{0,1\}^n)^w$, we define

$$\mathsf{Cond}^{h,+}_\pi(q) = \max_{\substack{U_1,\ldots,U_w,V_1,\ldots,V_w \subseteq \{0,1\}^n \\ |U_1|=\cdots=|V_w|=q}} |\{(\mathbf{x},\mathbf{y}) : \mathbf{y} = \pi(\mathbf{x}), \mathbf{x}[j] \in U_j \, \forall j \in [w], \mathbf{y}[j] \in V_j \, \forall j \in [w] \backslash h\}|,$$

$$\mathsf{Cond}^{h,-}_\pi(q) = \max_{\substack{U_1,\ldots,U_w,V_1,\ldots,V_w \subseteq \{0,1\}^n \\ |U_1|=\cdots=|V_w|=q}} |\{(\mathbf{x},\mathbf{y}) : \mathbf{y} = \pi(\mathbf{x}), \mathbf{x}[j] \in U_j \, \forall j \in [w] \backslash h, \mathbf{y}[j] \in V_j \, \forall j \in [w]\}|$$

$$\mathsf{aboCond}_\pi(q) = \max(\max_{h \in [w]}(\mathsf{Cond}^{h,+}(\pi, q)), \max_{h \in [w]}(\mathsf{Cond}^{h,-}(\pi, q)))$$

(Here the first two definitions are for all $h \in [w]$, and we use $\forall$ in postfix notation.) Thus the set V_h is immaterial in the definition of $\mathsf{Cond}^{h,+}$, while the set U_h is immaterial in the definition of $\mathsf{Cond}^{h,-}$. We call $\mathsf{aboCond}_\pi(q)$ as the *all-but-one conductance* of π (at q queries).

In Appendix A we show that the conductance and all-but-one conductance of a random permutation π are both roughly qwn, which is essentially $q \log(q)$ since q is exponential in n. Constructing explicit permutations with low conductance is an interesting open problem.

More intuition regarding the purpose of conductance and all-but-one conductance is given in Section 4.

3 Network Nomenclature and Main Result

In this section we give a syntax-oriented description of the confusion-diffusion networks for which our main results are obtained. This material is intended to cover the bare minimum that is necessary to state the main result. In the next section we explain the design principles of the simulator(s) that attempt to emulate these confusion-diffusion networks.

A *round* of a confusion-diffusion network refers to a round of S-boxes. More precisely, all S-box permutations $P_{i,j}$ with the same value of i lie in the same *round* of the network.

As mentioned, our results concern eight different confusion-diffusion networks with between 5 and 11 rounds. Since, say, the middle round of the 5-round confusion-diffusion network plays the same structural role (with respect to our simulator) as the middle round in the 9-round confusion-diffusion network, it makes more sense to designate rounds according to their structural role instead of by their round number (as the latter will keep changing from network to network, even while the structural purpose of the round stays the same).

For this purpose, we replace the array $\mathcal{P} = \{P_{i,j}\}$ of $r \times w$ random permutations with an array $\mathcal{Q}$ of $12 \times w$ random permutations where each "round" (value of the index i) is designated by a different alphabet letter. Specifically, we let

$$\mathcal{Q} = \{F_j, G_j, I_j, D_j, J_j, B_j, A_j, C_j, K_j, E_j, L_j, H_j : j \in [w]\} \tag{1}$$

be a set of $12w$ random permutations, where each permutation is thus indexed by an alphabet letter from the set $\{A, \ldots, L\}$ as well as by an index $j \in [w]$.

Having traded the set of indices $\{i : i \in [r]\}$ (the possible round numbers) for the set of letters $\{A, \ldots, L\}$, a "round" will henceforth mean, for us, a member of the latter set, i.e., a "round" means one of the letters $A, \ldots, L$.

Not all rounds will be used for all confusion-diffusion networks. For example, our 5-round confusion-diffusion network uses the rounds

$$G, D, A, E, H$$

and no others. However the order in which rounds appear, if they appear, is invariant and is the same as the order in which we listed the elements of $\mathcal{Q}$, cf. (1). (Thus, for example, the permutation $P_{1,1}$ according to our old naming scheme becomes the permutation G_1 in the 5-round network, in our new naming scheme.)

Our eight different confusion-diffusion networks correspond to the eight different possible settings of three independent boolean flags called $\mathsf{XtraMiddleRnd}$, $\mathsf{XtraOuterRnd}$ and $\mathsf{XtraUntglRnds}$. The rounds that appear in each network, as a function of these boolean flags, are as follows:

$$\begin{cases} A & \text{if } \mathsf{XtraMiddleRnd} \text{ is off} \\ B, C & \text{if } \mathsf{XtraMiddleRnd} \text{ is on} \end{cases}$$

$$\begin{cases} G, H & \text{if } \mathsf{XtraOuterRnd} \text{ is off} \\ F, G, H & \text{if } \mathsf{XtraOuterRnd} \text{ is on} \end{cases}$$

$$\begin{cases} D, E & \text{if } \mathsf{XtraUntglRnds} \text{ is off} \\ I, D, J, K, E, L & \text{if } \mathsf{XtraUntglRnds} \text{ is on} \end{cases}$$

As can be seen, toggling either of $\mathsf{XtraMiddleRnd}$ or $\mathsf{XtraOuterRnd}$ "costs" one extra round, whereas toggling $\mathsf{XtraUntglRnds}$ four extra rounds respectively. Hence the number of rounds in the network will be

$$5 + \mathsf{XtraMiddleRnd} + \mathsf{XtraOuterRnd} + 4 \cdot \mathsf{XtraUntglRnds}$$

which spans the integers 5, 6, 6, 7, 9, 10, 10, 11.

For example, our 11-round network consists of the rounds

$$F, G, I, D, J, B, C, K, E, L, H$$

in this order. (See also Fig. 1 in Section 4 for the following discussion.) The 10-round network with XtraMiddleRnd= **false** consists of the rounds

$$F, G, I, D, J, A, K, E, L, H$$

in this order as well. All other networks can be obtained by removing rounds from one of these two sequences. In more detail, round F is removed to un-toggle XtraOuterRnd and rounds I, J, K, L are removed to un-toggle XtraUntglRnds.

We will also rename the diffusion permutations $\overline{\pi} = (\pi_1, \ldots, \pi_r)$ according to their structural roles in the diffusion network. This time, however, we will find it convenient to reuse the name $\overline{\pi}$ for the set of diffusion permutations (as opposed to above, where we switched from $\mathcal{P}$ to $\mathcal{Q}$). We let

$$\overline{\pi} = (\nu, \pi_G, \pi_I, \pi_J, \pi_B, \tau, \pi_C, \pi_K, \pi_L, \pi_H)$$

where each element in the sequence $\overline{\pi}$ is a permutation from $\{0,1\}^{wn}$ to $\{0,1\}^{wn}$. In the 11-round confusion-diffusion network, diffusion permutations appear interleaved with the S-box rounds in the order

$$F\text{–}\nu\text{–}G\text{–}\pi_G\text{–}I\text{–}\pi_I\text{–}D\text{–}\pi_J\text{–}J\text{–}\pi_B\text{–}B\text{–}\tau\text{–}C\text{–}\pi_C\text{–}K\text{–}\pi_K\text{–}E\text{–}\pi_L\text{–}L\text{–}\pi_H\text{–}H$$

(i.e., the S-box round consisting of the parallel application of the permutations F_j is followed by the diffusion permutation ν, and so on), whereas in the 10-round network with XtraMiddleRnd= **false** the diffusion permutations appear in the order

$$F\text{–}\nu\text{–}G\text{–}\pi_G\text{–}I\text{–}\pi_I\text{–}D\text{–}\pi_J\text{–}J\text{–}\pi_B\text{–}A\text{–}\pi_C\text{–}K\text{–}\pi_K\text{–}E\text{–}\pi_L\text{–}L\text{–}\pi_H\text{–}H$$

with τ dropped. From either of these configurations one can un-toggle XtraOuterRnd by dropping F–ν– and one can un-toggle XtraUntglRnds by dropping I–π_I–, –π_J–J, K–π_K– and –π_L–L. For example, our 9-round confusion-diffusion network has the order

$$G\text{–}\pi_G\text{–}I\text{–}\pi_I\text{–}D\text{–}\pi_J\text{–}J\text{–}\pi_B\text{–}A\text{–}\pi_C\text{–}K\text{–}\pi_K\text{–}E\text{–}\pi_L\text{–}L\text{–}\pi_H\text{–}H$$

whereas the 5-round and 6-round network with XtraMiddleRnd toggled respectively have order

$$G\text{–}\pi_G\text{–}D\text{–}\pi_B\text{–}A\text{–}\pi_C\text{–}E\text{–}\pi_H\text{–}H$$
$$G\text{–}\pi_G\text{–}D\text{–}\pi_B\text{–}B\text{–}\tau\text{–}C\text{–}\pi_C\text{–}E\text{–}\pi_H\text{–}H$$

and so on.

Altogether the confusion-diffusion network is a function of the confusion permutations $\mathcal{Q}$, of the diffusion permutation vector $\overline{\pi}$ and of the three boolean flags XtraMiddleRnd, XtraOuterRnd and XtraUntglRnds. For brevity we write this network as

$$P[\mathcal{Q}, \overline{\pi}]$$

keeping the three boolean flags implicit. Depending on the value of the flags some permutations in $\mathcal{Q}$ and/or $\overline{\pi}$ are of course unused. In particular, we assume that unused permutations in $\mathcal{Q}$ are simply ignored for the purpose of the indifferentiability experiment (i.e., these unused permutations are not accessible as oracles).

In order to more succinctly state the main result, we define

$$\begin{aligned}
\mathsf{MaxColl}(\overline{\pi}) &= \max(\mathsf{MaxColl}(\pi_G), \mathsf{MaxColl}(\pi_B^-), \mathsf{MaxColl}(\pi_C), \mathsf{MaxColl}(\pi_H^-))\\
\mathsf{MaxPreim}(\overline{\pi}) &= \max(\mathsf{MaxPreim}(\pi_G), \mathsf{MaxPreim}(\pi_B^-), \mathsf{MaxPreim}(\pi_C), \mathsf{MaxPreim}(\pi_H^-),\\
&\qquad \mathsf{MaxPreim}(\pi_I), \mathsf{MaxPreim}(\pi_J^-), \mathsf{MaxPreim}(\pi_K), \mathsf{MaxPreim}(\pi_L^-))\\
\mathsf{MaxCoPr}(\overline{\pi}) &= \max(\mathsf{MaxColl}(\overline{\pi}), \mathsf{MaxPreim}(\overline{\pi}))
\end{aligned}$$

where π^- denotes the inverse of π.

Moreover we define

$$N = 2^n$$

and

$$\alpha(q) = \begin{cases} (2q)^w & \text{if } \mathsf{XtraMiddleRnd} \text{ is off,}\\ \mathsf{Cond}_\tau(2q) & \text{if } \mathsf{XtraMiddleRnd} \text{ is on,}\end{cases}$$

$$\beta(q) = \begin{cases} (q+\alpha(q))^w & \text{if } \mathsf{XtraOuterRnd} \text{ is off,}\\ \mathsf{Cond}_\nu(q+\alpha(q)) & \text{if } \mathsf{XtraOuterRnd} \text{ is on.}\end{cases}$$

The definitions of $\alpha(q)$ and $\beta(q)$ might seem annoyingly technical right now. In Section 4 we will provide more digestible semantic explanations for $\alpha(q)$ and $\beta(q)$.

Theorem 1. *Let $N = 2^n$. The confusion-diffusion network $P[\mathcal{Q}, \overline{\pi}]$ achieves (t_S, q_S, ε)-indifferentiability from a random permutation $\mathcal{Z} : \{0,1\}^{wn} \to \{0,1\}^{wn}$ for ε equal to*

$$\begin{aligned}
&\frac{\beta(q)(q+\alpha(q))^w}{N^w - q - \alpha(q)} + \frac{1}{N^w} + \frac{4w(q+\alpha(q))^2}{N-q-\alpha(q)}\\
&+ \frac{4wq\,\mathsf{aboCond}_\tau(2q)}{N-2q} && \textit{if } \mathsf{XtraMiddleRnd} \textit{ is on}\\
&+ \frac{2w(q+\alpha(q))\,\mathsf{aboCond}_\nu(q+\alpha(q))}{N-q-\alpha(q)} && \textit{if } \mathsf{XtraOuterRnd} \textit{ is on}\\
&+ \frac{4w\alpha(q)(q+\alpha(q))\mathsf{MaxCoPr}(\overline{\pi})}{N-q-\alpha(q)} && \textit{if } \mathsf{XtraUntglRnds} \textit{ is off}\\
&+ \frac{6w(q+\alpha(q))^2\,\mathsf{MaxPreim}(\overline{\pi})}{N-q-\alpha(q)} && \textit{if } \mathsf{XtraUntglRnds} \textit{ is on}
\end{aligned}$$

and for $q_S = \beta(q)$, $t_S = O(w(q+\alpha(q))^w)$. Here $q = q_0(1+rw)$ where q_0 is the number of distinguisher queries and $r \in \{5, 6, 7, 9, 10, 11\}$ is the number of rounds in the confusion-diffusion network.

Interpretation. In order to get a rough feel for the bound of Theorem 1 it is helpful to make the order-of-magnitude approximations

$$\begin{aligned}
\mathsf{MaxPreim}(\overline{\pi}) = \mathsf{MaxColl}(\overline{\pi}) &\approx O(1)\\
\mathsf{Cond}_\tau(2q) = \mathsf{aboCond}_\tau(2q) &\approx q\\
\mathsf{Cond}_\nu(q+\alpha(q)) = \mathsf{aboCond}_\nu(q+\alpha(q)) &\approx \alpha(q).
\end{aligned}$$

With these approximations in place, and given $q \ll N$ (in fact we can assume $q \le N^{1/2}$, since the security bound is void otherwise) it easy to verify that the largest terms in Theorem 1 are of the order

$$\frac{\alpha(q)^2}{N}$$

and which is, therefore, a first approximation to the security ε that we achieve. Unfolding the definition of $\alpha(q)$, we thus find

$$\varepsilon \approx \begin{cases} (2q)^{2w}/N & \text{if } \mathsf{XtraMiddleRnd} \text{ is off,} \\ q^2/N & \text{if } \mathsf{XtraMiddleRnd} \text{ is on} \end{cases}$$

for the security, to a first approximation. On the other hand we find

$$q_S = \beta(q) \approx \begin{cases} (2q)^{w^2} & \text{if } \mathsf{XtraMiddleRnd}/\mathsf{XtraOuterRnd} \text{ are off/off} \\ (2q)^{w} & \text{if } \mathsf{XtraMiddleRnd}/\mathsf{XtraOuterRnd} \text{ are off/on} \\ (2q)^{w} & \text{if } \mathsf{XtraMiddleRnd}/\mathsf{XtraOuterRnd} \text{ are on/off} \\ q & \text{if } \mathsf{XtraMiddleRnd}/\mathsf{XtraOuterRnd} \text{ are on/on} \end{cases}$$

for the query complexity, again to a first approximation.

Digging a little deeper into lower-order factors, if we let

$$\mu(\overline{\pi}) = \begin{cases} \mathsf{MaxCoPr}(\overline{\pi})) & \text{if } \mathsf{XtraUntglRnds} \text{ is off} \\ \mathsf{MaxPreim}(\overline{\pi})) & \text{if } \mathsf{XtraUntglRnds} \text{ is on} \end{cases}$$

then it is easy to verify that

$$\varepsilon = \begin{cases} O(w)\alpha(q)^2\mu(\overline{\pi})/N & \text{if } \mathsf{XtraOuterRnd} \text{ is off,} \\ O(w)\alpha(q)\mathsf{aboCond}_\nu(q+\alpha(q))/N & \text{if } \mathsf{XtraOuterRnd} \text{ is on} \end{cases} \tag{2}$$

by keeping only the biggest term(s) in the security bound and by folding lower-order terms—as well as factors of the type $N/(N-q-\alpha(q))$, which are dominated by constants—into the big-Oh constant.

In Appendix C we discuss the explicit construction of diffusion permutations with low values of $\mathsf{MaxColl}$, and also unconditional lower bounds on $\mathsf{MaxColl}$. In that appendix, we show that

$$\mu(\overline{\pi}) = \begin{cases} O(w) & \text{if } \mathsf{XtraUntglRnds} \text{ is off,} \\ 1 & \text{if } \mathsf{XtraUntglRnds} \text{ is on,} \end{cases}$$

in the best case, and with these bounds being matched by explicit constructions. With this value of $\mu(\overline{\pi})$ the first line of (2) (i.e., with $\mathsf{XtraOuterRnd} = \textbf{false}$) becomes

$$\begin{cases} O(w^2)\alpha(q)^2/N & \text{if } \mathsf{XtraUntglRnds} \text{ is off} \\ O(w)\alpha(q)^2/N & \text{if } \mathsf{XtraUntglRnds} \text{ is on} \end{cases}$$

where $\alpha(q)$ can be further unfolded in each case according to the value of $\mathsf{XtraMiddleRnd}$.

In summary, the security bound is essentially a function of the flag $\mathsf{XtraMiddleRnd}$ with secondary influences from the flags $\mathsf{XtraOuterRnd}$ and $\mathsf{XtraUntglRnds}$. The query complexity, for its part, is highly sensitive to both $\mathsf{XtraMiddleRnd}$ and $\mathsf{XtraOuterRnd}$, while the simulator's time complexity is essentially q^w in all cases.

It should be noted that while security of the form, e.g., q^2/N, is hardly relevant for practical values of n such as $n = 8$, security in the current model can actually not exceed $q = wrN$ distinguisher queries *regardless of the simulator* because the distinguisher can learn the entire content of the S-boxes with this many queries. In other words, no simulator can hope to handle practical parameters. Nonetheless, an improved "beyond birthday" security bound, if it could be attained, might shed additional theoretical insight.

The effect of linear permutations. It is relatively easy to see that $\mathsf{MaxColl}(\pi) = 2^n = N$ for any linear permutation $\pi : \mathrm{GF}(2^n)^w \to \mathrm{GF}(2^n)^w$ as long as $w > 2$. In this case $\mathsf{MaxCoPr}(\pi) = N$, and Theorem 1 becomes void if XtraUntglRnds is on. Thus one of the main reasons for toggling XtraUntglRnds would be to enable the use of linear diffusion permutations, or any other[3] family of permutations that have small entry-wise randomized preimage resistance (but potentially large entry-wise randomized collision resistance).

It should be emphasized that the *only* properties required of the permutations ν and τ are low conductance and low all-but-one conductance—no other combinatorial property of these permutations is ever considered. If these permutations are required to be linear over[4] $\mathrm{GF}(2^n)$ however, we know from Appendix B that these conductances will be adverserly affected, and can no longer approach the theoretical minimums forecast by random permutations.

Even though we do not know the actual q-query conductance of a "generic" $\mathrm{GF}(2^n)$-linear permutation, we can take an educated guess and assume that both the conductance and all-but-one conductance are in the vicinity of q^2, which is compatible with the bounds in Appendix B.[5] Then, assuming that both of the flags XtraMiddleRnd and XtraOuterRnd are set, we have

$$\alpha(q) = \mathsf{Cond}_\tau(2q) = \mathsf{aboCond}_\tau(2q) = O(q^2)$$

and

$$\beta(q) = \mathsf{Cond}_\nu(q + \alpha(q)) = \mathsf{aboCond}_\nu(q + \alpha(q)) = O(q^4)$$

leading to security $\approx q^4/N$ and to query complexity $\approx q^4$.

It is also conceivable, however, to use $\mathrm{GF}(2^n)$-linear permutations for all permutations in $\overline{\pi}$ *except* ν and τ. This would especially make sense if one could devise a fast diffusion permutation with provably low conductance (or even with conductance that is believed to be low according to some heuristic criteria).

Further discussion—including an extension of our main result—is given in Section 7.

Space complexity of the simulator, and the potential benefit of adding an extra round to the right-hand side detect zone. As will be clarified by the description of our simulator given in Section 4, adding an extra round to the right-hand outer detect zone (assuming XtraOuterRnd is set) does not further decrease the simulator's query complexity, nor improve security. Nonetheless, such an extra zone can be used to reduce the simulator's space complexity while maintaining essentially the same query complexity.

Without going into details here (see more discussion after Lemma 52, at the end of Section 5), the simulator's space complexity can be reduced from $O(\beta(q))$ to $O(q)$ by keeping the same number of rounds in both the outer left and outer right detect zones (i.e., either 1 for each or 2 for each); taking advantage of this space savings, though, means slightly increasing the query complexity, to $4\beta(q)$ from $\beta(q)$.

[3] Indeed, an interesting research direction would be the design of faster-than-linear diffusion permutations with small entry-wise randomized preimage resistance.

[4] Indeed, the result of Appendix B says nothing about the conductance of, say, GF(2)-linear permutations, i.e., in which each output bit is an xor of some specified set of input bits. Such permutations might well have low conductance, as far as we know!

[5] The purpose of taking such an educated guess is just for the sake of curiosity. We do not aim or claim to draw any hard conclusions here!

4 Simulator Overview

Context. We start with some very high-level description and reminder-of-purpose of our simulator. For this discussion it will be more convenient if we momentarily revert to indexing the S-boxes by coordinate pairs (i, j) where $i \in [r]$ the round number and $j \in [w]$ the layer number, with r being the number of rounds and w being the width. The diffusion permutation between the i-th and $(i+1)$-th rounds will again be denoted π_i as well.

The simulator is responsible for answering queries to the S-boxes, and has access to a random permutation oracle $\mathcal{Z} : \{0,1\}^{wn} \to \{0,1\}^{wn}$ that is being independently accessed by the distinguisher. The simulator's job is to keep the S-box answers compatible with $\mathcal{Z}$ in the sense that it looks to the distinguisher as if $\mathcal{Z}$ is implemented by the confusion-diffusion network.

For each pair $(i, j) \in [r] \times [w]$ the simulator maintains a pair of tables $P_{i,j}$ and $P_{i,j}^{-1}$, each containing 2^n entries of n bits each, in which the simulator keeps a record of "what it has already decided" about the (i, j)-th S-box. Initially the tables are blank, meaning that $P_{i,j}(x) = P_{i,j}^{-1}(y) = \perp$ for all $x, y \in \{0,1\}^n$. The simulator sets $P_{i,j}(x) = y$, $P_{i,j}^{-1}(y) = x$ to indicate that the (i, j)-th S-box maps x to y. The simulator never overwrites values in $P_{i,j}$ or in $P_{i,j}^{-1}$ and always keeps these two tables consistent. Hence $P_{i,j}$ encodes a partial matching (or "partial permutation") from $\{0,1\}^n$ to $\{0,1\}^n$ from which edges are never subtracted. We also note that the edges in $P_{i,j}$ are a *superset* of those queries that the distinguisher has made to the (i, j)-th S-box or to its inverse (i.e., $P_{i,j}$ contains the answers to those queries, and possibly more).

By analogy with the notation of Section 2 we write

$$P_i(\mathbf{x}) = \mathbf{y} \tag{3}$$

if $\mathbf{x}, \mathbf{y} \in \{0,1\}^{wn}$ are vectors such that $P_{i,j}(\mathbf{x}[j]) = \mathbf{y}[j]$ for all $j \in [w]$. Note that (3) is a time-dependent statement, in the sense that the tables $P_{i,j}$ keep accruing entries as the distinguishing experiment proceeds. For example, (3) is initially false for all i and all vectors $\mathbf{x}$, $\mathbf{y}$. Moreover P_i is not an actual table maintained by the simulator—i.e., (3) is "just notation".

A sequence of vectors $(\mathbf{x}^1, \mathbf{y}^1, \ldots, \mathbf{x}^r, \mathbf{y}^r)$ is called a *completed path*[6] if $P_i(\mathbf{x}^i) = \mathbf{y}^i$ for $i = 1, \ldots, r$ and if $\pi_i(\mathbf{y}^i) = \mathbf{x}^{i+1}$ for $i = 1, \ldots, r-1$. The set of completed paths is also time-dependent. The vectors $\mathbf{x}^1$ and $\mathbf{y}^r$ are called the *endpoints* of the path.

We informally say that the distinguisher *completes a path* if it makes queries to the simulator that form a completed path. (There are many different possible ways to order such a set of queries, obviously.) One can picture the distinguisher as trying to complete paths in various devious ways (typically, reusing the same queries as part of different paths), and checking that the path endpoints are each time compatible with $\mathcal{Z}$.

The simulator's job, in response, is to run ahead of the distinguisher and pre-emptively complete paths that it thinks the distinguisher is interested in, such as to make these paths compatible with $\mathcal{Z}$. The simulator's dilemna is that it must choose under which conditions to complete a path; if it waits too long, or completes paths in only highly specialized cases, it may find itself trapped in a contradiction (typically, while trying to complete several paths at once); but if it is too trigger-happy, having a very large number of conditions under which it will choose to complete a path, the simulator runs the risk creating[7] an out-of-control chain reaction of path completions.

[6] This definition, made for the sake of expository convenience, is superceded further down, where we redefine "completed path" by adding the requirement that the endpoints be compatible with $\mathcal{Z}$, i.e., that $\mathcal{Z}(\mathbf{x}^1) = \mathbf{y}^r$.

[7] Indeed, the simulator makes no distinction between those entries in its tables $P_{i,j}$ that are the direct result of an distinguisher query, and those which it created on its own while pre-emptively completing paths. It seems very hard to leverage such a distinction. Note for example that the distinguisher may know values in $P_{i,j}$ without having made the relevant queries, simply by virtue of knowing how the simulator works.

Essentially the simulator must be safe, but in a smart enough way that it avoids (out-of-control) chain reactions. We will informally refer to the problem of showing that no out-of-control chain reactions occur as the problem of *simulator termination*.

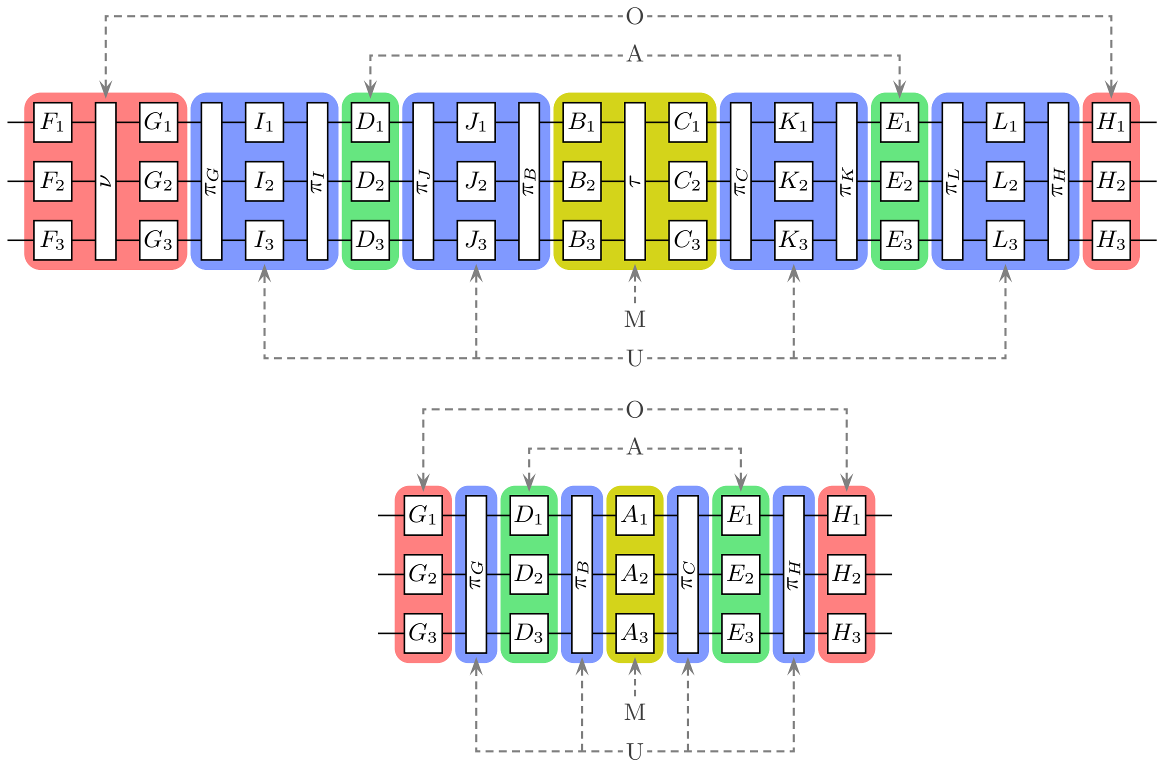
O
A
F_1 F_2 F_3 ν G_1 G_2 G_3 π_G I_1 I_2 I_3 π_I D_1 D_2 D_3 π_J J_1 J_2 J_3 π_B B_1 B_2 B_3 τ C_1 C_2 C_3 π_C K_1 K_2 K_3 π_K E_1 E_2 E_3 π_L L_1 L_2 L_3 π_H H_1 H_2 H_3
M
U
O
A
G_1 G_2 G_3 π_G D_1 D_2 D_3 π_B A_1 A_2 A_3 π_C E_1 E_2 E_3 π_H H_1 H_2 H_3
M
U

Fig. 1. Emplacement of the outer detect (O), adapt (A), middle detect (M) and untangle (U) zones for the 11- and 5-round simulators. The adapt zones always consist of rounds D and E.

SIMULATOR ZONES. Conceptually our simulator divides the confusion rounds and diffusion permutations into nine zones of four different types, to wit, one *middle detect* zone (M), left and right *outer detect* zones (O), four *untangle* zones (U) and two *adapt* zones (A). Each zone consists of one or more contiguous rounds[8] and/or diffusion permutations with every round and every diffusion permutation belonging to exactly one zone.

Fig. 1 shows how zones are divided for the 11- and 5-round networks. The zone division for networks with other numbers of rounds can easily be extrapolated from these. For example, the four untangle zones will be

$$\begin{cases} \pi_G,\ \pi_B,\ \pi_C,\ \pi_H & \text{if } \mathsf{XtraUntglRnds} \text{ is off} \\ \pi_G\text{–}I\text{–}\pi_I, \pi_J\text{–}J\text{–}\pi_B, \pi_C\text{–}K\text{–}\pi_K, \pi_L\text{–}L\text{–}\pi_H & \text{if } \mathsf{XtraUntglRnds} \text{ is on} \end{cases}$$

depending only on the value of XtraUntglRnds, while the outer detect zones will be

$$\begin{cases} G, H & \text{if } \mathsf{XtraOuterRnd} \text{ is off} \\ F\text{–}\nu\text{–}G, H & \text{if } \mathsf{XtraOuterRnd} \text{ is on} \end{cases}$$

[8] We recall from Section 3 that a round refers to an S-box round.

depending only the value of $\mathsf{XtraOuterRnd}$, and so on. We observe that zones (specifically, untangle zones with $\mathsf{XtraUntglRnds} = \mathbf{false}$) might consist solely of diffusion permutations. This contrasts with semantically similar zoning systems for Feistel network simulators and key-alternating simulators [15, 16, 24] for which the zones always contain at least one ideal component. We also observe that in the minimalist 5-round simulator, each round and each diffusion permutation corresponds to an individual zone.

Table Notation, Columns and Matching Query Sets. We revert to identifying rounds with letters in $\{A, \dots, L\}$. Under this notation tables $P_{i,j}$, $P_{i,j}^{-1}$ described above become, e.g., tables A_j and A_j^{-1}. Thus for each round $T \in \{A, \dots, L\}$ that is "operational" (as will depend on the flag settings) the simulator maintains tables T_j, T_j^{-1} for $1 \leq j \leq w$, as already described above under the notation $P_{i,j}, P_{i,j}^{-1}$.

We write $T(\mathbf{x}) = \mathbf{y}$ if $T_j(\mathbf{x}[j]) = \mathbf{y}[j]$ for each $j \in [w]$, and for all $T \in \{A, \dots, L\}$. We also write $T(\mathbf{x}) = \perp$ if $T_j(\mathbf{x}[j]) = \perp$ for at least one value of j. The notation $T^{-1}(\mathbf{y})$ is analogous, with T^{-1} being the inverse of T. As emphasized above this notation is really "notation only" in the sense that the simulator does not maintain such things as tables T or T^{-1}.

A non-null entry in table T_j will be called an (S-box) *query*. More formally, an S-box query is a quadruple (T, j, x, y) where $T \in \{A, \dots, L\}$, $j \in [w]$, $x, y \in \{0,1\}^n$, such that $T_j(x) = y$.

A set of w queries with the same value of T but with different values of j will be called a *column* or a T*-column* when we wish to emphasize the round. The (unique) vectors $\mathbf{x}$, $\mathbf{y}$ such that $(T, j, \mathbf{x}[j], \mathbf{y}[j])$ is a query in the column for each $j \in [w]$ are called the *input* and *output* of the column respectively. We note that a column is uniquely determined by either its input or output.

Two columns are *adjacent* if their rounds are adjacent. (E.g., a B-column and a C-column are adjacent.) Two adjacent columns are *matching* if $\pi(\mathbf{y}) = \mathbf{x}$, where $\mathbf{y}$ is the output of the first column, where $\mathbf{x}$ the input of the second column, and where π is the diffusion permutation between the two rounds.

A pair of columns from the first and last round of the confusion-diffusion network are likewise *matching* if $\mathcal{Z}(\mathbf{x}) = \mathbf{y}$, where $\mathbf{x}$ is the input to the first-round column (either an F- or G-column) and $\mathbf{y}$ is the output of the last-round column (the H-column).

The notion of matching columns naturally extends to sequences of columns from consecutive rounds of length greater than two. (The first and last round of the network are always considered adjacent.) If a set of matching columns encompasses all rounds we call the set a *completed path.* Thus, since the first and last column of a network are considered adjacent, completed paths are compatible with $\mathcal{Z}$ by definition.

The set of queries in a matching sequence of columns of any length is called a *matching set* of queries. We will also refer to the queries of a single column as a matching set, considering that column as a matching sequence of columns of length 1.

One can also observe that two different completed paths cannot contain the same column. Indeed, each D-box is a permutation and each round implements a partial permutation as well.

Simulator Operation and Termination Argument. Our simulator applies a design paradigm pioneered by Yannick Seurin [24] that has also been used in other simulators for Feistel networks or key-alternating ciphers [15, 16]. As for all such simulators, our simulator completes two types of paths, where one type of path is triggered by a "middle detect zone" and the other is triggered by an "outer detect zone".

In more detail, our simulator pre-emptively completes a path[9] for every matching set of queries in the middle detect zone (such a matching set will consist of either one or two columns, depending on whether $\mathsf{XtraMiddleRnd}$ is toggled) and also for every matching set of queries in the two outer detect zones, considered as a single consecutive set of columns (the latter kind of matching set will thus consist of either two or three columns, depending on whether $\mathsf{XtraOuterRnd}$ is toggled).

Crucially, one can show that, unless some bad event of negligible probability occurs, each outer-triggered path completion is associated to a unique pre-existing query made by the distinguisher to $\mathcal{Z}$. Since the distinguisher makes only q queries in total, this means that at most q outer-triggered path completions occur, with high probability. In fact our simulator aborts if it counts[10] more than q outer-triggered path completions, so in our case at most q outer-triggered path completions occur with probability 1.

Moreover, a middle-triggered path completion does not add any new queries to the middle detect zone. This means that all queries in the middle detect zone can be chalked up to one of two causes: (1) queries directly made by the distinguisher, (2) queries made by the simulator during outer-triggered path completions.

Cause (1) accounts for at most q queries to each S-box and, as just seen, cause (2) accounts for at most q queries as well. Hence a middle detect zone S-box is never queried at more then $2q$ points, i.e., the table T_j for $T \in \{A, B, C\}$ never has more than $2q$ non-$\perp$ entries.

In particular, if $\mathsf{XtraMiddleRnd}$ is off, the latter implies that no more than $(2q)^w$ middle-triggered path completions occur, or one for every possible combination of an entry from each of the tables $A_1, \ldots, A_w$. If $\mathsf{XtraMiddleRnd}$ is on, on the other hand, than at most $\mathsf{Cond}_\tau(2q)$ middle-triggered path completions occur, as is easy to see per the definition[11] of conductance. In other words, $\alpha(q)$ (cf. Section 3) is an upper bound on the number of middle-triggered path completions. In fact, $\alpha(q)$ is an upper bound for the *total* number of path completions performed (including also outer path completions), since each completed path is also associated to a unique set of matching queries from the middle detect zone.

As for all S-boxes outside the middle detect zone, their queries can also be chalked up to one of two sources, namely direct distinguisher queries and path completions. There are at most q direct distinguisher queries, and each completed path contributes at most 1 query to each S-box, so each S-box outside the middle detect zone ends up with at most $q + \alpha(q)$ queries.

The simulator, moreover, only queries $\mathcal{Z}$ in order to either complete paths or else in order to detect outer-triggered path completions. This implies the number of distinct simulator queries to $\mathcal{Z}$ is upper bounded by the number of matching sets of queries in the left-hand outer detect zone. Indeed each completed path is obviously associated to a matching set of queries in the left-hand outer detect zone; and for the purpose of outer-triggered path detection, it is easy to see that the simulator only needs to query $\mathcal{Z}$ at most once for each such matching set as well by maintaining a

[9] The phrase "completes a path" is informal at this point, as there are generally many different ways to complete a path. (E.g., where to "adapt" a path to make it compatible with $\mathcal{Z}$, etc.) More details follow below.

[10] This means the simulator knows the value of q beforehand, which introduces a small amount of non-uniformity into the simulator. One could remove this non-uniformity—i.e., not tell the simulator the value of q beforehand—at the cost of a more complicated theorem statement and proof. But the fact that essentially all security games allow adversaries that know the value of q for which they want to carry out an attack makes the issue a bit moot.

[11] More precisely, let $U_j = \{y \in \{0,1\}^n : B_j^{-1}(y) \neq \perp$, let $V_j = \{x \in \{0,1\}^n : C_j(x) \neq \perp\}$ at the end of the distinguishing experiment. Then $|U_j|, |V_j| \leq 2q$ for each j, and the number of middle-triggered path completions that have occurred is at most

$$\{(\mathbf{x}, \mathbf{y}) : \tau(\mathbf{x}) = \mathbf{y}, \mathbf{x} \in \prod_j U_j, \mathbf{y} \in \prod_j V_j\} \leq \mathsf{Cond}_\tau(2q).$$

table[12] of queries already made to $\mathcal{Z}$. If $\mathsf{XtraOuterRnd}$ is off, thus, the number of simulator queries to $\mathcal{Z}$ will be at most $(q + \alpha(q))^w$; whereas if $\mathsf{XtraOuterRnd}$ is on, the same number will be at most $\mathsf{Cond}_\nu(q + \alpha(q))$. The simulator query complexity is thus at most $\beta(q)$ (cf. Section 3).

Procedure Names and Aliases. The procedures implementing the oracles maintained by the simulator are denoted using uppercase plaintext letters, where the letter matches the round of the S-box in question. The first argument to the oracle denotes the index of the S-box within the round. Thus $\mathrm{A}(j, x)$ denotes a query to the j-th S-box in round A in input $x \in \{0,1\}^n$. Likewise $\mathrm{A}^{-1}(j, y)$ denotes a query to the inverse of the j-th S-box in round A on input $y \in \{0,1\}^n$, and so on. These are the oracles available to the distinguisher, ranging over all rounds that are in use as determined by the boolean flags.

Moreover, it will be convenient to have a single name, e.g., for "the last round of the middle detect zone" or "the first round of the left-hand outer detect zone" irrespective of the boolean flag values. For this we implement the following three "round aliases" U, V and X:

- U denotes A if $\mathsf{XtraMiddleRnd}$ is off, denotes B otherwise
- V denotes A if $\mathsf{XtraMiddleRnd}$ is off, denotes C otherwise
- X denotes G if $\mathsf{XtraOuterRnd}$ is off, denotes F otherwise

For example, the table U_j denotes the table A_j if $\mathsf{XtraMiddleRnd}$ is off, the table B_j otherwise, and so on.

Further Details: Query Handling. S-box queries from the adversary to the simulator, and from the simulator to itself, are handled differently according to whether or not they occur in a position and in a direction that might trigger a path completion. Queries outside the detect zones—i.e., queries in the adapt or untangle zones—cannot trigger path completions and are answered by lazy permutation sampling, if they are not already present in the relevant table T_j. Likewise, new queries to B, C^{-1}, G^{-1}, H as well as to F and F^{-1} are unlikely to trigger new path completions and are answered by lazy permutation sampling. (For example, a new query to C^{-1} is unlikely to form a new pair of matching B and C columns together with some other $2w - 1$ pre-existing queries. The degree to which this is unlikely, however, depends on the all-but-one conductance of τ, which is precisely how all-but-one conductance enters into the proof. As another example, a new query to F^{-1} or H is unlikely to trigger a new outer path completion, because it is unlikely that the random answer to this query will "meet up" with a previous query to $\mathcal{Z}$.)

All other queries—namely queries to A, A^{-1}, B^{-1}, C, G and H^{-1}—can reasonably be expected to result in new path completions. Instead of being lazy-sampled "on the spot", these queries are handled by a system of *requests*. In more detail, when the simulator fields a *request* to A, A^{-1}, B^{-1}, C, G or H^{-1}, the simulator makes note of the request and determines what path completions the request will trigger. Instead of answering the request, the simulator proceeds to begin these path completions in "the other direction" (i.e., the direction for which the answer to the current request isn't needed); these tentative path completions eventually engender new *requests* of their own, which engender new path completions of their own, and so on. The recursive process eventually stops with a list of pending requests and a list of pending path completions. At this point (and we give more details below) the simulator simultaneously lazy samples all the pending requests, and simultaneously completes all the pending path completions.

[12] Thus, in particular, adding an extra round to the right-hand detect zone would not further reduce the query complexity, since the query complexity is only proportional to the number of matching query sets for the left-hand outer detect zone.

Crucially for this idea, requests to U^{-1} (i.e., requests to A^{-1} or B^{-1}, depending) can only give rise to new requests to G, as is easy to see[13], and likewise requests to G can only engender new requests to U^{-1}. Symmetrically, requests to V only engender new requests to H^{-1} and vice-versa. Thus, there exist only two possible "closed systems" of recursive request calls: the $\mathrm{G}/\mathrm{U}^{-1}$ system and the $\mathrm{V}/\mathrm{H}^{-1}$ system. At most one of the two possible systems is triggered by any given distinguisher query. (To be precise, a G or U^{-1} query will trigger the $\mathrm{G}/\mathrm{U}^{-1}$ system; a V or H^{-1} query will trigger the $\mathrm{V}/\mathrm{H}^{-1}$ system; other queries trigger neither system.)

Unfolding of Recursive Calls. We now describe in more detail how a recursive system of $\mathrm{G}/\mathrm{U}^{-1}$ requests unfolds. The simulator maintains sets $\mathrm{ToBeAssigned}_{G_j}$ and $\mathrm{ToBeAssigned}^{-}_{U_j}$ for $1 \leq j \leq w$ that hold the requests to G and U^{-1} respectively. At the beginning of the query cycle (a *query cycle* is the portion of game execution between when the distinguisher makes a query and when that query is answered) these sets are empty.

Assume, say, that the system of recursive calls starts with a distinguisher query $\mathrm{G}(j, x)$. At this point the simulator *requests* the value $\mathrm{G}(j, x)$ by an internal call $\mathrm{RequestG}(j, x)$. If $G_j(x) \neq \perp$ the request has no effect and the call returns with no value.

Otherwise, if $G_j(x) = \perp$, the simulator adds the value x to the set $\mathrm{ToBeAssigned}_{G_j}$. The simulator then enumerates outer path completions, if any, that will be triggered by the new query $(G, j, x, \cdot)$. In more detail, the simulator enumerates all vectors $\mathbf{x}^G$ such that $\mathbf{x}^G[j] = x$ and such that either $G_{j'}(\mathbf{x}^G[j']) \neq \perp$ or $\mathbf{x}^G[j'] \in \mathrm{ToBeAssigned}_{G_{j'}}$ for all $j' \neq j$. If $\mathsf{XtraOuterRnd}$ is off the simulator defines $\mathbf{y}^H = \mathcal{Z}(\mathbf{x}^G)$ and checks that $H^{-1}(\mathbf{y}^H) \neq \perp$; if $\mathsf{XtraOuterRnd}$ is on, the simulator checks that $\perp \neq \mathbf{x}^F := F^{-1}(\nu^-(\mathbf{x}^G))$, evaluates $\mathbf{y}^H = \mathcal{Z}(\mathbf{x}^F)$, and checks again that $H^{-1}(\mathbf{y}^H) \neq \perp$. (As soon as the first check fails, the simulator moves on to the next candidate for $\mathbf{x}^G$. In particular, the simulator does not define any new queries via lazy sampling at this stage.)

For each $\mathbf{x}^G$ for which all of the above checks succeed, the simulator starts completing a path backward starting from the vector $\mathbf{x}^H = H^{-1}(\mathbf{y}^H)$. For this the simulator fills in missing queries to the path (working backward through the rounds) by lazy sampling until it reaches round U; here, instead of filling in the missing queries, it calls $\mathrm{RequestU}^{-1}(j, \mathbf{y}^U[j])$ for $1 \leq j \leq w$, where $\mathbf{y}^U$ is the input to U^{-1} for the path in question. It also adds the pair $(\mathbf{x}^G, \mathbf{y}^U)$ to a list of such pairs named $\mathrm{ToBeAdapted}_D$ (initially empty at the start of the query cycle).

A description of how requests to U^{-1} are handled will finish the description of the recursive procedure. Firstly, as for G-requests, a U^{-1}-request (j, y) returns immediately if $U_j^{-1}(y) \neq \perp$ of if $y \in \mathrm{ToBeAssigned}^{-}_{U_j}$ already. Otherwise, the value y is added to $\mathrm{ToBeAssigned}^{-}_{U_j}$ and the simulator enumerates all middle path completions, if any, that will be triggered by the new request. In more detail the simulator enumerates all vectors $\mathbf{y}^U$ such that $\mathbf{y}^U[j] = y$ and such that either $U_j(\mathbf{y}^U[j']) \neq \perp$ or $\mathbf{y}^U[j'] \in \mathrm{ToBeAssigned}^{-}_{U_{j'}}$ for all $j \neq j'$; moreover if $\mathsf{XtraMiddleRnd}$ is on the simulator checks that $\perp \neq \mathbf{y}^V := C(\tau(\mathbf{y}^U))$; otherwise (if $\mathsf{XtraMiddleRnd}$ is off) we have $\mathbf{y}^V = \mathbf{y}^U$.

For each such $\mathbf{y}^U$ for which $\mathbf{y}^V$ is defined, the simulator starts completing a path forward from $\mathbf{y}^V$. The simulator fills missing queries to the path by lazy sampling until it reaches round G (after calling $\mathcal{Z}^{-1}$). At this point the simulator adds the pair $(\mathbf{x}^G, \mathbf{y}^U)$ for the corresponding $\mathbf{x}^G$ to the list $\mathrm{ToBeAdapted}_D$ and calls $\mathrm{RequestG}(j, \mathbf{x}^G[j])$ for all $1 \leq j \leq w$, so the recursion continues.

When the recursion terminates control reverts to the original procedure $\mathrm{G}(j, x)$ (queried by the distinguisher) that placed the first request. So far, no requests have been answered, though all

[13] The path completion(s) triggered by the request to U^{-1} will "pass through" rounds H and F (if present) without triggering new path completions, due to the fact, argued above, that (forward) queries to H and F are unlikely to cause new outer-triggered path completions.

requests have been recorded, and all partially-built, yet-to-be-completed paths have been recorded as well by a pair $(\mathbf{x}^G, \mathbf{y}^U)$ in the set ToBeAdapted_D.

At this point $\mathrm{G}(j, x)$ will calls a procedure AdaptLeft which simultaneously samples all the pending requests and completes all pending paths, "adapting" each such path at round D. This procedure is discussed below.

A recursive system of $\mathrm{G}/\mathrm{U}^{-1}$ calls can also be triggered by a distinguisher query to U^{-1} (A^{-1} or B^{-1} depending), but this case is exactly analogous. Moreover a recursive system of $\mathrm{V}/\mathrm{H}^{-1}$ calls is analogous as well.

Path Completion and the Purpose of the Untangle Zones. When AdaptLeft is called it has the sets $\text{ToBeAssigned}_{G_j}$ and $\text{ToBeAssigned}^-_{U_j}$ for $1 \le j \le w$, as well as the list ToBeAdapted_D consisting of pairs of the form $(\mathbf{x}^G, \mathbf{y}^U)$ for paths to be completed. We will assume that ToBeAdapted_D has k elements, and write the pairs in ToBeAdapted_D as

$$(\mathbf{x}_1^G, \mathbf{y}_1^U), \ldots, (\mathbf{x}_k^G, \mathbf{y}_k^U).$$

It is easy to see that $G_j(x) = \perp$ and $U_j^{-1}(y) = \perp$ for each $x \in \text{ToBeAssigned}_{G_j}$ and each $y \in \text{ToBeAssigned}^-_{U_j}$. Moreover, and as we argue in the proof, one can show that for each pair $(\mathbf{x}_i^G, \mathbf{y}_i^U)$ there is some $j \in [w]$ such that $\mathbf{x}_i^G[j] \in \text{ToBeAdapted}_{G_j}$ and some $h \in [w]$ such that $\mathbf{y}_i^U[h] \in \text{ToBeAdapted}^-_{U_h}$. Moreover, one can also show that $\mathbf{x}_i^G \neq \mathbf{x}_{i'}^G$, $\mathbf{y}_i^U \neq \mathbf{y}_{i'}^U$ for all $i \neq i'$.

AdaptLeft consists of three stages:

(1) Values of $G_j(x)$, $U_j^{-1}(y)$ are (finally) lazy sampled for each $x \in \text{ToBeAssigned}_{G_j}$, $y \in \text{ToBeAssigned}^-_{U_j}$, for each $j \in [w]$, whence vectors $\mathbf{y}_i^G := G(\mathbf{x}_i^G)$, $\mathbf{x}_i^U := U^{-1}(\mathbf{y}_i^U)$ become defined for $1 \le i \le k$.

(2) The vector $\mathbf{y}_i^G$ is "evaluated forward" to a vector $\mathbf{x}_i^D$ that is an input to round D for $1 \le i \le k$, while the vector $\mathbf{x}_i^U$ is "evaluated backward" to a vector $\mathbf{y}_i^D$ that is an output to round D (in a little more detail, if $\mathsf{XtraUntglRnds}$ is off, we have $\mathbf{x}_i^D = \pi_G(\mathbf{y}^G)$, $\mathbf{y}_i^D = \pi_B^-(\mathbf{x}^U)$; if $\mathsf{XtraUntglRnds}$ is on, lazy sampling of rounds I and J is required to define $\mathbf{x}_i^D$, $\mathbf{y}_i^D$).

(3) AdaptLeft (tries to) set $D_j(\mathbf{x}_i^D[j]) = \mathbf{y}_i^D[j]$ for each $i \in [k]$ and each $j \in [w]$.

The only step that can run into problems is step (3). Namely, the step (3) fails if either (i) there exists values $i \in [k]$, $j \in [w]$ such that either $D_j(\mathbf{x}_i^D[j]) \neq \perp$ or $D_j^{-1}(\mathbf{y}_i^D[j]) \neq \perp$ before the start of step (3), or (ii) there exists values $i, i' \in [k]$, $i \neq i'$, $j \in [w]$, such that either $\mathbf{x}_i^D[j] = \mathbf{x}_{i'}^D[j]$ or $\mathbf{y}_i^D[j] = \mathbf{y}_{i'}^D[j]$ before the start of step (3). The purpose of the untangle zones is to make these two types of bad events unlikely.

If $\mathsf{XtraUntglRnds}$ is off, a bit of thought reveals that small values of $\mathsf{MaxPreim}(\pi_G)$, $\mathsf{MaxPreim}(\pi_C^-)$ are sufficient and necessary to ensure that (i) only occurs with low probability whereas small values of $\mathsf{MaxColl}(\pi_G)$, $\mathsf{MaxColl}(\pi_C^-)$ are sufficient and necessary to ensure (ii) only occurs with low probability. The argument is simple (involving a union bound) but the details are notationally tedious, and we refer to the proof of Lemma 38 in Section 5 for the details.

If $\mathsf{XtraUntglRnds}$ is on, a little bit more thought[14] reveals that small values of $\mathsf{MaxPreim}(\pi_G)$, $\mathsf{MaxPreim}(\pi_I)$ as well as small values of $\mathsf{MaxPreim}(\pi_J^-)$, $\mathsf{MaxPreim}(\pi_C^-)$ ensure that both (i) and (ii) are unlikely. Here too we refer to the proof of Lemma 38 in Section 5 for more details.

[14] Focusing on the first untangle zone, the intuition is, firstly, that small $\mathsf{MaxPreim}(\pi_G)$ implies that all queries to I will be fresh, i.e., will be lazy sampled during AdaptLeft, even if there are some indices $i \neq i'$ for which (some of) those queries overlap; then because π_G is a permutation, for each $i \neq i'$, there is at least one $j \in [w]$ such that $\mathbf{x}_i^I[j] \neq \mathbf{x}_{i'}^I[j]$ and, hence, such that $I_j(\mathbf{x}_i^I[j])$ is independently lazy sampled from $I_j(\mathbf{x}_{i'}^I[j])$; focusing on the last of these two value to be lazy sampled, and coupled with the fact that $\mathsf{MaxPreim}(\pi_I)$ is small, we can conclude that all entries in $\mathbf{x}_i^D$, $\mathbf{x}_{i'}^D$ are distinct with high probability.

Interestingly, one could imagine relying on even weaker combinatorial properties of the diffusion permutations in the untangle zones, if the untangle zones had more rounds. We refer to Section 7 for more ideas in this direction.

THE SIMULATOR IN PSEUDOCODE. Our "official" simulator is given by the pseudocode in Figures 2–4 in Section 5, more specifically by all procedures in game G_1 of that pseudocode, except for procedures Z and Z^{-1} which implement the random permutation $\mathcal{Z}$, and which are not part of the simulator properly speaking.

We note that following [15] our simulator uses *explicit random tapes* for its lazy sampling. More precisely, for each $T \in \{A, \ldots, L\}$ and for each $j \in [w]$ there is a tape p_{T_j} (and, implicitly, an inverse $p_{T_j}^{-}$) that encodes a random permutation of $\{0,1\}^n$ in the form of a table: $p_{T_j}(x)$ is the image of x for all $x \in \{0,1\}^n$; moreover $p_{T_j}^{-}(y)$ is the preimage of y for all $y \in \{0,1\}^n$. Likewise the random permutation $\mathcal{Z}$ (implemented by the procedures Z and Z^{-1}) relies on a random permutation tape p_Z.

Of course, in a practical implementation of the simulator such random tapes would be too large to keep in memory, and an independent lazy sampling process would substitute for each tape access.

5 Proof

In this section we provide the proof of Theorem 1. More precisely, we show that the simulator discussed in Section 4 and formally given by the pseudocode in Figures 2–6 fulfills the claims of Theorem 1.

On a high level, the formal structure of our indifferentiability proof(s) follows [1] rather closely. On the other hand our simulator shares more in common with the simulator of [15] (which also employs Seurin's termination paradigm) so our proof is also descended from [15]. Moreover our proof utilizes (with some refinements) the *randomness mapping* technique pioneered by [15], and described in more detail below.

5.1 Overview

An indifferentiability theorem encompasses at least three claims: an upper bound on the simulator running time, an upper bound on the simulator query complexity, and lastly the indistinguishability of the real and simulated worlds. Here (in the proof overview) we focus on the latter aspect. Lemmas 51, 52 in subsection 5.5 establish upper bounds on the simulator efficiency and query complexity respectively.

The indistinguishability argument uses a sequence of five games G_1, G_2, …, G_5. The games are specified by the pseudocode in Figs. 2–8. More precisely, the pseudocode in Figs. 2–6 collectively describes games G_1, G_2 and G_3 (the differences between the various games being highlighted in red, and also by comment lines), whereas Fig. 7–8 depicts games G_4 and G_5. Game G_1 is the simulated world and G_5 is the real world.

Following [1] the simulator explicitly aborts ('**abort**') if runs into an unexpected state. The distinguisher is consequently notified and, in such a case, knows that it is in the simulated world (since game G_5, the real world, doesn't abort).

As stated, game G_1 is the simulated world. The interface available to the distinguisher consists of the **public** procedures, of which the simulator controls all but Z and Z^{-1} (which implement $\mathcal{Z}$ and $\mathcal{Z}^{-1}$). The procedures $\mathrm{Z}/\mathrm{Z}^{-1}$ maintain a partial permutation table P_Z that records which queries have already been made to these procedures either by the adversary or by the simulator. In game G_1 the table P_Z serves no effective purpose, but this changes in later games. Moreover Z and Z^{-1} are the only procedures to access the permutation table P_Z in game G_1 (as should be, since the simulator shouldn't know which queries to Z/Z^{-1} the distinguisher has already made).

Game G_2 contains only one change from game G_1, in the procedure CheckZ used by the simulator (see Fig. 2). In G_1 CheckZ$(\mathbf{x}, \mathbf{y})$ calls Z$(\mathbf{x})$ to see if Z$(\mathbf{x}) = \mathbf{y}$. In game G_2, however, CheckZ$(\mathbf{x}, \mathbf{y})$ answers instead according to the table P_Z of values already queried to Z/Z^{-1} by either the distinguisher or the simulator, and returns **false** if the relevant table entry is blank, without calling Z. Thus CheckZ may return a "false negative" in game G_2. The G_1–G_2 game transition is inspired by a similar transition in [15]. Our transition is substantially simpler, however, since (and following [1]) we keep Z as a permutation instead of replacing it by a "two-way random function" [15].

The fact that G_1 and G_2 are hard to distinguish for a q-query distinguisher is proven in Lemma 40, Section 5.4. The proof is relatively straightforward but relies on having an upper bound on the number of times that CheckZ is called in G_1 and G_2. This upper bound is given by Lemma 23 at the end of Section 5.2.

Game G_3 adds a number of **abort** conditions over game G_2. (These abort conditions could not be present in G_2 because they involve the simulator reading values in P_Z, or because they involve abort conditions placed in Z/Z^{-1}, which should of course not abort in the real simulated world.) In fact, one doesn't need to upper bound the distinguishability of G_2 from G_3: one simply observes that the pair (G_3, G_5) is strictly *easier* to distinguish than the pair (G_2, G_5) (as abortions never occur in G_5—we refer to Lemma 41 for a formal statement and proof). Hence it suffices to show that G_3 and G_5 are hard to distinguish.

The transition from G_3 to G_5 is analyzed via G_4. In G_4 the procedures Z and Z^{-1} are changed: instead of using the random tape p_Z these procedures now use (some subset of) the random tapes $p_{A_1}, \ldots, p_{L_w}$ and the "real" substitution-permutation network; see Figs. 7–8. There is another minor technicality: the random tape p_{T_j} is renamed q_{T_j} in G_4, in order to clarify subsequent simultaneous discussion of G_3 and G_4.

The proof's key transition is the transition from G_3 to G_4. For this a *randomness mapping* argument is used, following [15]. We borrow the idea of *footprints* from [1], but we manage to eschew their "execution trees".

The randomness mapping argument is sketched in some detail after Lemma 41 in Section 5.4. For completeness, however, we provide a briefer sketch here.

A *footprint* is that subset of the random tapes (plus its contents) which is actually accessed during an execution[15]. If we change portions of a random tape outside of its footprint, the execution doesn't change. The set of possible footprints is a function of the distinguisher (which is presumed fixed and deterministic, wlog). A moment's thought reveals that two distinct footprints (with respect to the same distinguisher) are never compatible: they always overlap in at least one conflicting entry.

A footprint is said to be *good* if the execution doesn't **abort** for that footprint. Essentially, the randomness mapping argument maps good footprints of G_3 to good footprints of G_4 by the following rule: the content of the table T_j at the end of the G_3-execution becomes the G_4-footprint for p_{T_j} (actually, q_{T_j} in G_4). One must prove that points in the image of this mapping are really footprints for G_4, that the mapping is injective (maps distinct footprints to distinct footprints), and that a G_3-execution with a good footprint looks identical from the distinguisher's perspective to a G_4-execution run with the image footprint. For all this to be useful one must also show that the "probability mass" of good footprints in G_3 (i.e., the probability of obtaining a good footprints, taken over the uniform choice of random tapes) is high, and that a good G_3 footprint has roughly

[15] An *execution* consists of one run of the distinguishing experiment from beginning to end.

the same probability of occuring as its image in G_4. Once all these elements in place it is easily seen that G_3 are G_4 are hard to distinguish.

We note that the probability of obtaining a good footprint in G_3 is exactly the probability of not aborting in G_3. Hence for the randomness mapping argument to be effective one must, in particular, show that G_3 aborts with low probability. Section 5.3 is devoted to upper bounding the latter probability. For more details we refer to the afore-mentioned material in Section 5.4.

For the transition from G_4 to G_5, we first argue that queries in G_4 are all answered by the value found in the underlying random tape (i.e., query $\mathrm{T}(j, x)$ is answered by $q_{T_j}(x)$, and so on) as long as **abort** does not occur. (See Lemma 49.) Hence the only effective difference between G_4 and G_5 is the possibility that G_4 aborts. The probability that G_4 aborts can be upper bounded from the probability that G_3 aborts and from the randomness map. This is the argument adopted by [1]. However, we note that a more careful additive accounting of the transition probabilities causes the probability of abortion in G_4 to cancel out entirely from the final distinguishing upper bound (see the computation in the proof of Lemma 50) which makes the latter argument superfluous, and which also saves us a factor of two[16] in the final indistinguishability bound over [1,15,16]. This is another (small) technical contribution of our proof.

Proof Organization. A prerequisite for much of the analysis is to have upper bounds on the operations of various types (e.g., path completions, queries) carried out by the simulator. The proof starts by subsection 5.2, where we establish the most important such upper bounds.

Another crucial component of the analysis is to upper bound the probability of bad executions (i.e., executions in which the simulator aborts) in game G_3. This is the topic of subsection 5.3.

The high-level assembly of these prior low-level results occurs in subsection 5.4, where we separately analyze each of the game transitions G_1–G_2, G_3–G_4 and G_4–G_5 and, in particular, carry out the randomness mapping argument (for the transition from G_3 to G_4).

5.2 Simulator Efficiency

Last Minute Reminders. We have $\alpha(q) = (2q)^w$ if XtraMiddleRnd is off, $\alpha(q) = \mathsf{Cond}_\tau(2q)$ if XtraMiddleRnd is on; $\beta(q) = (q+\alpha(q))^w$ if XtraOuterRnd is off, $\beta(q) = \mathsf{Cond}_\nu(q+\alpha(q))$ if XtraOuter-Rnd is on. We recall that p_T, p_T^- are a pair of random tapes encoding a random permutation from $\{0,1\}^n$ to $\{0,1\}^n$ for all $T \in \{A, \ldots, L\}$ whereas p_Z, p_Z^- are a pair of random tapes encoding a random permutation from $\{0,1\}^{wn}$ to $\{0,1\}^{wn}$.

An *execution* refers to the entire run of the distinguishing experiment; a *q-query execution* is an execution in which the distinguisher makes at most q queries; a *query cycle* is that time portion of an execution between when the distinguisher makes a query and when that query is answered.

For $T \in \{A, \ldots, L\}$ we write $T(\mathbf{x})$ to denote the vector whose j-th entry is $T_j(\mathbf{x}[j])$, if those w values are all defined, where T_j refers to the table maintained by the simulator; $T(\mathbf{x}) = \perp$ if $T_j(\mathbf{x}[j]) = \perp$ for any j. Similar conventions hold for $T^{-1}(\mathbf{y})$ with respect to the tables T_j^{-1}, $j \in [w]$.

For the time being, q is the number if distinguisher queries. The distinction between q and q_0 is elucidated in subsection 5.4.

Lemma 1. *Each pair of tables T_j, T_j^{-1}, $T \in \{F, G, I, D, J, B, A, C, K, E, L, H\}$, $1 \leq j \leq w$, represents a partial permutation at all points in the execution of* G_1, G_2 *and* G_3 *outside of calls to* SetTable. *Moreover, values in these tables are never overwritten.*

[16] In fact, Andreeva et al. [1], thanks to their use of footprints, save a factor of two in the transition from G_3 to G_4 over the traditional approach of [15,16]. But Andreeva et al. lose this factor again in the transition from G_4 to G_5 by double-counting abort probabilities. In our approach the saved factor of two is conserved all the way through.

Proof. The lemma transparently follows from the fact that these tables are initialized to $\perp$ and from the fact that the only function to write to the tables T_j, T_j^{-1} is SetTable, which is set to abort the game rather than overwrite. $\square$

Lemma 2. *The pair of tables P_Z, P_Z^{-1} represents a partial permutation at all points in the execution of* G_1*,* G_2 *and* G_3 *outside of calls to* SetTable*. Moreover P_Z remains consistent with p_Z.*

Proof. The lemma directly follows from the fact that table P_Z, P_Z^{-1} are presumed initialized to $\perp$ and from the fact that the tables P_Z, P_Z^{-1} are only modified by the ReadTape calls in Z and Z^{-1}. $\square$

Definition 2. *For* G_1*,* G_2 *and* G_3 *we define*

$$\begin{aligned}
\mathrm{Index}_{G_j} &= \mathrm{ToBeAssigned}_{G_j} \cup \{x : G_j(x) \neq \perp\}, \\
\mathrm{Index}^-_{H_j} &= \mathrm{ToBeAssigned}^-_{H_j} \cup \{y : H_j^{-1}(y) \neq \perp\}, \\
\mathrm{Index}_{A_j} &= \mathrm{ToBeAssigned}_{A_j} \cup \{x : A_j(x) \neq \perp\}, \\
\mathrm{Index}^-_{A_j} &= \mathrm{ToBeAssigned}^-_{A_j} \cup \{y : A_j^{-1}(y) \neq \perp\}, \\
\mathrm{Index}^-_{B_j} &= \mathrm{ToBeAssigned}^-_{B_j} \cup \{y : B_j^{-1}(y) \neq \perp\}, \\
\mathrm{Index}_{C_j} &= \mathrm{ToBeAssigned}_{C_j} \cup \{x : C_j(x) \neq \perp\},
\end{aligned}$$

at all time points in the execution.

As per the aliases introduced in Section 3 $\mathrm{Index}^-_{U_j}$ will stand for $\mathrm{Index}^-_{A_j}$ if XtraOuterRnd is off, for $\mathrm{Index}^-_{B_j}$ otherwise, and Index_{V_j} will stand for Index_{A_j} if XtraOuterRnd is off, for Index_{C_j} otherwise.

Lemma 3. *The sets* $\mathrm{Index}^{(-)}_{T_j}$ *are monotone increasing in* G_1*,* G_2 *and* G_3*: elements are never removed from these sets.*

Proof. The set $\{x : G_j(x) \neq \perp\}$ is monotone increasing by Lemma 1. Moreover x is only removed from $\mathrm{ToBeAssigned}_{G_j}$ in AdaptLeft after $G_j(x)$ has been read from p_{G_j} (see Fig. 5). Similar statements apply to $\mathrm{Index}^-_{H_j}$, $\mathrm{Index}^-_{U_j}$, and Index_{V_j}. $\square$

We will refer to the procedures RequestG, $\mathrm{RequestU}^{-1}$, AdaptLeft as well as the public procedures G and U^{-1} (A^{-1} or B^{-1} depending on XtraMiddleRnd) as *left-hand* procedures and to the procedures $\mathrm{RequestH}^{-1}$, RequestV, AdaptRight as well as the public procedures H^{-1} and V (A or C depending on XtraMiddleRnd) as *right-hand* procedures. Moreover, the data structures $\mathrm{ToBeAssigned}_{G_j}$, $\mathrm{ToBeAssigned}^-_{U_j}$, $\mathrm{ToBeAdapted}_D$ are *left-hand* (data structures), while $\mathrm{ToBeAssigned}_{V_j}$, $\mathrm{ToBeAssigned}^-_{H_j}$, $\mathrm{ToBeAdapted}_E$ are *right-hand* (data structures).

A query made by the distinguisher is *left-hand* if it is a query to G or U^{-1} it is *right-hand* if it is a query to H^{-1} or V.[17] Other adversarial queries are *neutral.* Finally a left-hand distinguisher query gives rise to a “left-hand” query cycle, a right-hand distinguisher query gives rise to a “right-hand” query cycle, etc.

The following lemma holds for G_1, G_2 and G_3.

[17] We observe one last time that U^{-1} and V aren't actual procedures, but stand for A^{-1} or B^{-1} in the first case and for A or C in the second case. On the other hand, $\mathrm{RequestU}^{-1}$, RequestV as well as $\mathrm{BlockRequestU}^{-1}$, BlockRequestV, BlockU^{-1} and BlockV are not aliases, but the actual names of pseudocode procedures.

Lemma 4. *Left-hand data structures are only modified by left-hand procedures, right-hand data structures are only modified by right-hand procedures. Right-hand procedures are never evaluated during left-hand or neutral query cycles and left-hand procedures are never evaluated during right-hand or neutral query cycles.*

Proof. These claims are all syntatically clear from the code. In particular, one can observe that the only closed systems or recursive calls that occurs are between G or U^{-1}, which is a left-hand closed system, and between H^{-1} or V, which is a right-hand closed system. □

Lemma 5. *Outside a left-hand query cycle, the left-hand data sets* $\mathrm{ToBeAssigned}_{G_j}$, $\mathrm{ToBeAssigned}^{-}_{U_j}$, $\mathrm{ToBeAdapted}_D$ *are empty in the execution of* G_1, G_2 *and* G_3*. A symmetric statement holds for right-hand data.*

Proof. $\mathrm{ToBeAssigned}_{G_j}$, $\mathrm{ToBeAssigned}^{-}_{U_j}$, $\mathrm{ToBeAdapted}_D$ are empty when AdaptLeft returns without abort. Moreover, these sets are initially empty and never modified in a right-hand or neutral query cycle by Lemma 4. □

Lemma 6. *For any $j \in [w]$, the sets $\{y : G_j(y) \neq \bot\}$ and* $\mathrm{ToBeAssigned}_{G_j}$ *are disjoint at all points in the execution of* G_1, G_2 *and* G_3 *except for inside the first* **forall** *loop in* AdaptLeft.

Similar statements hold for $\mathrm{ToBeAssigned}^{-}_{U_j}$, $\mathrm{ToBeAssigned}_{V_j}$ *and* $\mathrm{ToBeAssigned}^{-}_{H_j}$ *(for the latter two, with respect to* AdaptRight*).*

Proof. We focus on the table G_j. Other tables are treated similarly.

Outside a left-hand query cycle $\mathrm{ToBeAssigned}_{G_j}$ is empty by Lemma 5.

Inside a left-hand query, before AdaptLeft is called, G_j is unmodified and x is added to $\mathrm{ToBeAssigned}_{U_j}$ only if $G_j(x) = \bot$.

Finally, $\mathrm{ToBeAssigned}_{G_j}$ is empty after the first **forall** loop in AdaptLeft. □

Lemma 7. *For any q-query execution of* G_1, G_2 *or* G_3*, the number of times lines 17–18 of* RequestG *are executed plus the number of times lines 17–18 of* $\mathrm{RequestH}^{-1}$ *are executed is at most q. Moreover, the number of times line 20 of* RequestG *is executed plus the number of times line 20 of* $\mathrm{RequestH}^{-1}$ *is executed is at most q. (Nb: line 20 is the last line of* RequestG, $\mathrm{RequestH}^{-1}$*.)*

Proof. Both statements easily from the fact that when line 16 of $\mathrm{RequestG}^{-1}$ or RequestH is executed, NumOuter is increased by 1 and from the fact that the simulator aborts when NumOuter exceeds q. □

Lemma 8. *For any $j \in [w]$ and for any q-query execution of* G_1, G_2 *or* G_3 *the number of calls made to* $\mathrm{RequestU}^{-1}(j, \cdot)$ *plus the number of calls made to* $\mathrm{B}(j, \cdot)$ *is at most $2q$ if* XtraMiddleRnd *is on. A symmetric statement holds for* $\mathrm{C}^{-1}(j, \cdot)$ *and* $\mathrm{RequestV}(j, \cdot)$.

If XtraMiddleRnd *is off, the number of calls made to* $\mathrm{RequestU}^{-1}(j, \cdot)$ *plus the number of calls made to* $\mathrm{RequestV}(j, \cdot)$ *is at most $2q$.*

Proof. Note that $\mathrm{RequestU}^{-1}(j, \cdot)$ is called only by $\mathrm{U}^{-1}(j, \cdot)$ or by $\mathrm{BlockRequestU}^{-1}$. Moreover $\mathrm{U}^{-1}(j, \cdot)$ (which is $\mathrm{A}^{-1}(j, \cdot)$ or $\mathrm{B}^{-1}(j, \cdot)$, depending) is only called by the distinguisher.

If XtraMiddleRnd is on: The number of calls to $\mathrm{B}(j, \cdot)$ made by the distinguisher plus the number of calls to $\mathrm{B}^{-1}(j, \cdot)$ is at most q (as just noted, $\mathrm{B}^{-1}(j, \cdot)$ is only called by the distinguisher). Hence, the number of calls to $\mathrm{B}(j, \cdot)$ made by the distinguisher plus the number of times $\mathrm{RequestU}^{-1}(j, \cdot)$ is called by $\mathrm{B}^{-1}(j, \cdot)$ is at most q.

It remains to account for non-distinguisher calls to $\mathrm{B}(j,\cdot)$ as well as for the number of times $\mathrm{BlockRequestU}^{-1}$ calls $\mathrm{RequestU}^{-1}(j,\cdot)$, i.e., the number of times $\mathrm{BlockRequestU}^{-1}$ is called. However non-distinguisher calls to $\mathrm{B}(j,\cdot)$ only occur in BlockB, called only by the function LeftToMiddle, itself called only on line 17 of $\mathrm{RequestH}^{-1}$. Moreover $\mathrm{BlockRequestU}^{-1}$ is only called on line 20 of RequestG. Hence the sum of these two types of calls is at most q by Lemma 7.

If XtraMiddleRnd is off: The number of times $\mathrm{RequestU}^{-1}(j,\cdot)$ is called by $\mathrm{A}^{-1}(j,\cdot)$ plus the number of times $\mathrm{RequestV}(j,\cdot)$ is called by $\mathrm{A}(j,\cdot)$ is at most q, the number of distinguisher queries. Indeed, A and A^{-1} are not called by the simulator.

It remains again to account for calls to $\mathrm{RequestU}^{-1}(j,\cdot)$ made by $\mathrm{BlockRequestU}^{-1}$ and for calls to $\mathrm{RequestV}(j,\cdot)$ made by BlockRequestV. But this number of calls is at most q by Lemma 7, as $\mathrm{BlockRequestU}^{-1}$ only appears on line 20 of RequestG and BlockRequestV only appears on line 20 of RequestV. □

Lemma 9. *For any $j \in \{1,\ldots,w\}$ and for any q-query execution of G_1, G_2 or G_3 the size of $\mathrm{Index}_{U_j}^{-1}$ is at most $2q$ and the size of U_j is at most $2q$.*

The same holds for Index_{V_j} and V_j.

Proof. The size of U_j is no greater than the size of $\mathrm{Index}_{U_j}^{-1}$, so it suffices to prove the statement for Index_{U_j}.

If XtraMiddleRnd is on: The only functions to possibly increase the size of $\mathrm{Index}_{U_j}^{-1}$ are $\mathrm{B}(j,\cdot)$ and $\mathrm{RequestU}^{-1}(j,\cdot)$. Moreover the size of $\mathrm{Index}_{U_j}^{-1}$ increases by at most 1 during calls to either of these functions. The conclusion thus follows by Lemma 8.

If XtraMiddleRnd is off: The only functions to possibly increase the size of $\mathrm{Index}_{U_j}^{-1} = \mathrm{Index}_{A_j}^{-1}$ are $\mathrm{RequestU}^{-1}(j,\cdot)$ and AdaptRight. (Indeed, the latter adds entries to A_j^{-1}; AdaptLeft also adds entries to A_j^{-1}, but the y's for which AdaptLeft sets $A_j^{-1}(y)$ are already in $\mathrm{ToBeAssigned}_{U_j}^{-}$ at the start of AdaptLeft.) Moreover, for each value x such that $A_j(x)$ is assigned in AdaptRight there obviously exists a unique corresponding call to $\mathrm{RequestV}(j,\cdot)$, namely the call which added x to $\mathrm{ToBeAssigned}_{V_j}$. Hence the conclusion also follows by Lemma 8. □

Lemma 10. *In G_1, G_2 and G_3, when a pair $(\mathbf{x}^G, \mathbf{y}^U)$ is added to $\mathrm{ToBeAdapted}_D$ in $\mathrm{RequestU}^{-1}$, $\mathbf{y}^U[k] \in \mathrm{Index}_{U_k}^{-}$ for every $k \in [w]$, and there exists $j \in [w]$ such that $\mathbf{y}^U[j] \in \mathrm{ToBeAssigned}_{G_j}^{-}$.*

Moreover, if XtraMiddleRnd is on, let $\mathbf{x}^V = \tau(\mathbf{y}^U)$. Then $\mathbf{x}^V[k] \in \mathrm{Index}_{V_k}$ for every $k \in [w]$ when $(\mathbf{x}^G, \mathbf{y}^U)$ is added to $\mathrm{ToBeAdapted}_D$.

A symmetric statement holds for RequestV.

Proof. If $(\mathbf{x}^G, \mathbf{y}^U)$ is added to $\mathrm{ToBeAdapted}_D$ in $\mathrm{RequestU}^{-1}$ then $\mathrm{BlockDefined}(U, \mathbf{y}^U, -) = \mathbf{true}$, so $\mathbf{y}^U[k] \in \mathrm{Index}_{U_k}^{-}$ for all $k \in [w]$. Moreover if j and y are the arguments to $\mathrm{RequestU}^{-1}$ then it's clear that $\mathbf{y}^U[j] = y$ and that $y \in \mathrm{ToBeAssigned}_{U_j}^{-}$. (See the pseudocode of $\mathrm{RequestU}^{-1}$.)

If XtraMiddleRnd is on, moreover, then it's necessary (see the pseudocode again) that $\mathrm{BlockDefined}(C, \tau(\mathbf{y}^U),$ **true** in order for a pair $(\mathbf{x}^G, \mathbf{y}^U)$ to be added to $\mathrm{ToBeAdapted}_D$ in $\mathrm{RequestU}^{-1}$. So $\mathbf{x}^V[k] \in \mathrm{Index}_{V_k}$ for all $k \in [w]$. □

Lemma 11. *For every pair $(\mathbf{x}^G, \mathbf{y}^U)$ ever added to $\mathrm{ToBeAdapted}_D$,*

$$\mathbf{y}^U \in \mathrm{Index}_{U_1}^{-} \times \ldots \times \mathrm{Index}_{U_w}^{-}$$

Moreover, if XtraMiddleRnd is on, then

$$\tau(\mathbf{y}^U) \in \mathrm{Index}_{V_1} \times \ldots \times \mathrm{Index}_{V_w}.$$

A symmetric statement holds for every pair $(\mathbf{x}^V, \mathbf{y}^H)$ *added to* $\mathrm{ToBeAdapted}_E$. *(This for* G_1, G_2 *and* G_3.*)*

Proof. This is a simple corollary of Lemma 10. □

Lemma 12. *The same pair* $(\mathbf{x}^G, \mathbf{y}^U)$ *is never added twice to* $\mathrm{ToBeAdapted}_D$. *In fact, if* $(\mathbf{x}^G, \mathbf{y}^U)$ *and* $(\mathbf{x}'^G, \mathbf{y}'^U)$ *are two pairs added to* $\mathrm{ToBeAdapted}_D$ *at different points in time, then both* $\mathbf{x}^G \neq \mathbf{x}'^G$ *and* $\mathbf{y}^U \neq \mathbf{y}'^U$. *A symmetric statement holds for pairs* $(\mathbf{x}^V, \mathbf{y}^H)$ *and* $\mathrm{ToBeAdapted}_E$. *(This for* G_1, G_2 *and* G_3.*)*

Proof. We start by noting that the second statement easily follows from the first: when $(\mathbf{x}^G, \mathbf{y}^U)$ is added to $\mathrm{ToBeAdapted}_D$ in $\mathrm{RequestU}^{-1}$, $\mathbf{y}^U$ is associated to a unique vector $\mathbf{y}^H$, which association remains unchanged for the rest of the execution by virtue of the fact that the tables P_{T_j} are never overwritten (Lemma 1); moreover the mapping from $\mathbf{x}^X$ to $\mathbf{y}^H$ is also invariant given that values in P_Z are never overwritten either (Lemma 2); finally the mapping from $\mathbf{x}^X$ to $\mathbf{x}^G$ is invariant as P_{F_j} is never overwritten. Hence $\mathbf{y}^U$ "non-malleably" determines $\mathbf{x}^G$ and vice-versa.

We now argue that the same pair $(\mathbf{x}^G, \mathbf{y}^U)$ is never added twice to $\mathrm{ToBeAdapted}_D$ in $\mathrm{RequestU}^{-1}$. For the following argument it helps to picture a modified (but equivalent) version of the code in which the last **forall** loop of $\mathrm{RequestU}^{-1}$ is detached from the main body of that function, and executed separately by whatever function called $\mathrm{RequestU}^{-1}$ in the first place, right after the call to $\mathrm{RequestU}^{-1}$. With this (cosmetic) change in place the execution stack only ever contains one active copy of $\mathrm{RequestU}^{-1}$. (In other words, $\mathrm{RequestU}^{-1}$ is never called while another instance of $\mathrm{RequestU}^{-1}$ is still waiting to terminate.)

Firstly, now, a given pair $(\mathbf{x}^G, \mathbf{y}^U)$ cannot be added twice to $\mathrm{ToBeAdapted}_D$ by the same call to $\mathrm{RequestU}^{-1}$ because the **forall** loop of $\mathrm{RequestU}^{-1}$ only considers each value of $\mathbf{y}^U$ once. Thus if a pair $(\mathbf{x}^G, \mathbf{y}^U)$ is added twice to $\mathbf{y}^U$ it is added during two distinct calls to $\mathrm{RequestU}^{-1}$.

Let the first such call be $\mathrm{RequestU}^{-1}(j_1, y_1)$ and the second call be $\mathrm{RequestU}^{-1}(j_2, y_2)$. (We do not assume $(j_1, y_1) \neq (j_2, y_2)$.) By the remarks above we can assume—via our mental experiment—that $\mathrm{RequestU}^{-1}(j_1, y_1)$ has finished executing when $\mathrm{RequestU}^{-1}(j_2, y_2)$ is called. Also $\mathbf{y}^U[j_1] = y_1$ and $\mathbf{y}^U[j_2] = y_2$ since the **forall** loop of $\mathrm{RequestU}^{-1}(j, y)$ only iterates over vectors $\mathbf{y}^U$ such that $\mathbf{y}^U[j] = y$. For $\mathrm{RequestU}^{-1}(j_2, y_2)$ to not immediately return we must have $U_{j_2}^{-1}(y_2) = \bot$ and $y_2 \notin \mathrm{ToBeAssigned}_{U_{j_2}}^{-1}$ when the call $\mathrm{RequestU}^{-1}(j_2, y_2)$ occurs. But this implies $\mathbf{y}^U[j_1] \notin \mathrm{Index}_{U_{j_1}}$ before the second call, which contradicts the fact that $\mathbf{y}^U[j_1] \in \mathrm{Index}_{U_{j_1}}$ after the first call by Lemma 3. □

Lemma 13. *Let* $(\mathbf{x}^G, \mathbf{y}^U)$ *be a pair added to* $\mathrm{ToBeAdapted}_D$ *at some point and let* $(\mathbf{x}^V, \mathbf{y}^H)$ *be a pair added to* $\mathrm{ToBeAdapted}_E$ *at some other point of an execution of* G_1, G_2 *or* G_3. *Then* $\tau(\mathbf{y}^U) \neq \mathbf{x}^V$ *if* XtraMiddleRnd *is on. Moreover if* XtraMiddleRnd *is off,* $A(\mathbf{x}^V) \neq \mathbf{y}^U$ *(and* $A^{-1}(\mathbf{y}^U) \neq \mathbf{x}^V$*) for all* $j \in [w]$ *if* $A(\mathbf{x}^V)$ *(or* $A^{-}(\mathbf{y}^U)$*) ever becomes defined.*

Proof. Without loss of generality, we can assume the addition of $(\mathbf{x}^V, \mathbf{y}^H)$ to $\mathrm{ToBeAdapted}_E$ occurs after the addition of $(\mathbf{x}^G, \mathbf{y}^U)$ to $\mathrm{ToBeAdapted}_D$. Note moreover these additions will occur in distinct query cycles, since $\mathrm{RequestU}^{-1}$ is a left-hand procedure and RequestV is a right-hand procedure. For the second query cycle to occur at all, naturally, the first query cycle must be successful (i.e., not abort).

If XtraMiddleRnd is on: Lemma 11 implies that $\tau(\mathbf{y}^U) \in \mathrm{Index}_{V_1} \times \ldots \times \mathrm{Index}_{V_w}$ when $(\mathbf{x}^G, \mathbf{y}^U)$ is added to $\mathrm{ToBeAdapted}_D$. Hence any subsequent call $\mathrm{RequestV}(j, x)$ that doesn't immediately return must have $x \neq \tau(\mathbf{y}^U)[j]$. (More precisely, $C_j(\tau(\mathbf{y}^U)[j])$ will be defined for all $j \in [w]$ if the query cycle that adds $(\mathbf{x}^G, \mathbf{y}^U)$ to ToBeAdapted_D completes without aborting.) So any pair $(\mathbf{x}^V, \mathbf{y}^H)$ added to ToBeAdapted_E by RequestV must have $\mathbf{x}^V \neq \tau(\mathbf{y}^U)$.

If $\mathsf{XtraMiddleRnd}$ is off: Lemma 11 implies that $\mathbf{y}^U \in \text{Index}^-_{U_1} \times \cdots \times \text{Index}^-_{U_w}$ when $(\mathbf{x}^G, \mathbf{y}^U)$ is added to ToBeAdapted_D, and Lemma 3 implies that $A^-(\mathbf{y}^U) \neq \bot$ after that query cycle completes (since $\text{ToBeAssigned}^-_{U_j} = \emptyset$ for all j at the end of a non-aborted query cycle). Thus any subsequent call $\text{RequestV}(j, x)$ such that $\mathbf{x}^G[j] = x$ will return immediately. So any pair $(\mathbf{x}^V, \mathbf{y}^H)$ added later to ToBeAdapted_E by RequestV must have $\mathbf{x}^V \neq A^-(\mathbf{y}^U)$ or, equivalently (cf. Lemma 1) $A(\mathbf{x}^V) \neq \mathbf{y}^U$. □

For the remaining lemmas of this section, we define

$$\begin{aligned}\text{Index}^-_{U_j} \wedge \text{Index}_{V_j} = \{(x, y) : A_j(x) = y\} \; \cup \\ \{(x, \bot) : A_j(x) = \bot, x \in \text{Index}_{V_j}\} \; \cup \\ \{(\bot, y) : A^-_j(y) = \bot, y \in \text{Index}^-_{U_j}\}\end{aligned}$$

if $\mathsf{XtraMiddleRnd}$ is off. We note that since $\mathsf{XtraMiddleRnd}$ is off, and by definition of $\text{Index}^-_{U_j}$ and Index_{V_j}, this definition can also be written

$$\begin{aligned}\text{Index}^-_{U_j} \wedge \text{Index}_{V_j} = \{(x, y) : A_j(x) = y\} \; \cup \\ \{(x, \bot) : A_j(x) = \bot, x \in \text{ToBeAssigned}_{A_j}\} \; \cup \\ \{(\bot, y) : A^-_j(y) = \bot, y \in \text{ToBeAssigned}^-_{A_j}\}.\end{aligned}$$

We note that by definition, a pair (x, y) cannot exist in $\text{Index}^-_{U_j} \wedge \text{Index}_{V_j}$ if either of the pairs $(x, \bot)$ or $(\bot, y)$ is in $\text{Index}^-_{U_j} \wedge \text{Index}_{V_j}$. Moreover pairs of the form $(x, \bot)$ can only exist during right-hand query cycles, and pairs of the form $(\bot, y)$ can only exist during left-hand query cycles, as is clear from the definition and from Lemma 5.

Each pair in $\text{Index}^-_{U_j} \wedge \text{Index}_{V_j}$ will be called a *slot*. A slot of the form (x, y) is permanent by Lemma 1. Thus, two distinct slots cannot have the same non-null first coordinate, nor can they share the same non-null second coordinate. (Case analysis: if (x, y) and (x, y') are two slots then $A_j(x) = y$ and $A_j(x) = y'$, so $y = y'$ by Lemma 1; secondly, by the definition of $\text{Index}^-_{U_j} \wedge \text{Index}_{V_j}$, (x, y) and $(x, \bot)$ cannot both be slots, as noted above.)

Lemma 14. *If* $\mathsf{XtraMiddleRnd}$ *is off,* $\text{Index}^-_{U_j} \wedge \text{Index}_{V_j}$ *never contains more than* $2q$ *elements at any point in an execution of* G_1*,* G_2 *or* G_3*.*

Proof. AdaptLeft and AdaptRight do not add entries to $\text{Index}^-_{U_j} \wedge \text{Index}_{V_j}$, since they simply turns pairs of the form $(\bot, y)$ into pairs of the form (x, y) (in the case of AdaptLeft) or turn pairs of the form $(x, \bot)$ into pairs of the form (x, y) (in the case of AdaptRight).

Thus, the only functions to add pairs to $\text{Index}^-_{U_j} \wedge \text{Index}_{V_j}$ are calls to $\text{RequestU}^{-1}(j, \cdot)$ and calls to $\text{RequestV}(j, \cdot)$. However each such call adds at most one pair (since these call add at most one element to $\text{ToBeAssigned}^-_{A_j}$ and $\text{ToBeAssigned}_{A_j}$, respectively), and the total number of such calls is at most $2q$ by Lemma 8. □

We note that when $\mathsf{XtraMiddleRnd}$ is off, the value of the function ϕ—defined in the following lemma—on a given pair $(\mathbf{x}^G, \mathbf{y}^U)$ or $(\mathbf{x}^V, \mathbf{y}^H)$ is time-dependent, as the table A_j is time-dependent.

Lemma 15. *If* XtraMiddleRnd *is on: Let* $\phi(\mathbf{x}^G, \mathbf{y}^U) = \mathbf{y}^U$ *for each pair* $(\mathbf{x}^G, \mathbf{y}^U)$ *added to* $\mathrm{ToBeAdapted}_D$*, and let* $\phi(\mathbf{x}^V, \mathbf{y}^H) = \tau^{-1}(\mathbf{x}^V)$ *for each pair* $(\mathbf{x}^V, \mathbf{y}^H)$ *added to* $\mathrm{ToBeAdapted}_E$*. Then* ϕ *is injective, and the range of* ϕ *always lies in the set*

$$\{\mathbf{y} : \mathbf{y} \in \mathrm{Index}^-_{U_1} \times \cdots \times \mathrm{Index}^-_{U_w} \wedge \tau(\mathbf{y}) \in \mathrm{Index}_{V_1} \times \cdots \times \mathrm{Index}_{V_w}\}.$$

If XtraMiddleRnd *is off: Let* $\phi(\mathbf{x}^G, \mathbf{y}^U)$ *be the* w*-tuple whose* j*-th element is the pair* $(A^-_j(\mathbf{y}^U[j]),$ $\mathbf{y}^U[j])$ *for each* j*, for each* $(\mathbf{x}^G, \mathbf{y}^U)$ *added to* $\mathrm{ToBeAdapted}_D$*, and let* $\phi(\mathbf{x}^V, \mathbf{y}^H)$ *be the* w*-tuple whose* j*-th element is the pair* $(\mathbf{x}^V[j],\, A_j(\mathbf{x}^V[j]))$ *for each* j*, for each* $(\mathbf{x}^V, \mathbf{y}^U)$ *added to* $\mathrm{ToBeAdapted}_E$*. Then* ϕ *is injective, and the range of* ϕ *remains in the set* $(\mathrm{Index}^-_{U_1} \wedge \mathrm{Index}_{V_1}) \times \cdots \times (\mathrm{Index}^-_{U_w} \wedge \mathrm{Index}_{V_w})$*.*

Proof. Whether XtraMiddleRnd is on or off ϕ has the claimed range by Lemma 10 and is injective by lemmas 12 and 13. □

Lemma 16. *At the end of each successful query cycle in* G_1*,* G_2 *and* G_3*, every pair* $(\mathbf{y}^U, \mathbf{x}^V)$ *in*

$$\varGamma = \begin{cases} \left\{(\mathbf{y}^B, \mathbf{x}^C) : \tau(\mathbf{y}^B) = \mathbf{x}^C, \forall j\ \mathbf{y}^B[j] \in \mathrm{Index}^-_{B_j} \wedge \mathbf{x}^C[j] \in \mathrm{Index}_{C_j}\right\} & \text{if XtraMiddleRnd is on} \\ \left\{(\mathbf{y}^A, \mathbf{x}^A) : \forall j\ A_j(\mathbf{x}^A[j]) = \mathbf{y}^A[j]\right\} & \text{if XtraMiddleRnd is off} \end{cases}$$

is in a completed path.

Proof. If XtraMiddleRnd is on: For any such pair $(\mathbf{y}^U, \mathbf{x}^V)$, consider the query cycle during which $(\mathbf{y}^U, \mathbf{x}^V)$ becomes contained by the above set. Without loss of generality, has that $j' \in [w]$ has the property that is $\mathbf{y}^U[j']$ added to $\mathrm{Index}^-_{B'_j}$ after $\mathbf{y}^U[j]$ is added to $\mathrm{Index}^-_{B_j}$ for all $j \neq j'$, and also after $\mathbf{x}^V[j]$ is added to Index_{C_j} for all j.

As already noted in Lemma 9, a value y is added to $\mathrm{Index}^-_{B'_j}$ only by calls to $\mathrm{B}(j', \cdot)$ or to $\mathrm{RequestU}^{-1}(j', y)$. If $\mathbf{y}^U[j']$ is added to $\mathrm{Index}^{-1}_{B'_j}$ via B, then the query cycle is neutral or a right-hand query cycle, so $\mathrm{B}^-_j(\mathbf{y}^U[j]) \neq \perp$ for all $j \neq j'$ and $\mathbf{x}^V[j] \in \mathrm{Index}_{V_j}$ for all $j \in [w]$ by assumption, so the simulator will abort inside of B. (See the pseudocode for procedure $\mathrm{B}(j, x)$.)

Therefore we can assume that $\mathbf{y}^U[j']$ is added to $\mathrm{Index}^-_{B'_j}$ by a function call of the form $\mathrm{RequestU}^{-1}(j', y)$, which can only occur in a left-hand query cycle. By assumption, then $\mathbf{y}^U[j] \in \mathrm{Index}^{-1}_{U_j}$ and $\mathbf{x}^V[j] \in \mathrm{Index}_{V_j}$ for all $1 \leq j \leq w$ after $\mathbf{y}^U[j']$ is added to $\mathrm{ToBeAssigned}^{-1}_{B'_j}$ inside of $\mathrm{RequestU}^{-1}(j', \mathbf{y}^U[j'])$. The **forall** loop of $\mathrm{RequestU}^{-1}(j', \mathbf{y}^U[j'])$ will iterate with $\mathbf{y}^U$ and both $\mathrm{BlockDefined}(U, \mathbf{y}^U, -)$ and $\mathrm{BlockDefined}(V, \mathbf{x}^V = \tau(\mathbf{y}^U), +)$ will return **true**. A corresponding pair $(\mathbf{x}^G, \mathbf{y}^U)$ is then added to $\mathrm{ToBeAdapted}_D$ and, assuming successful completion of the query cycle, a completed chain containing $\mathbf{y}^U, \mathbf{x}^V$ is created by AdaptLeft.

If XtraMiddleRnd is off: We assume without loss of generality that $(\mathbf{y}^A, \mathbf{x}^A)$ enters $\varGamma$ for the first time during a left-hand query cycle, as neutral query cycles don't modify the tables A_j. Then there exists some $j' \in [w]$ such that $\mathbf{y}^A[j']$ is added to $\mathrm{ToBeAssigned}^-_{A_{j'}}$ during that query cycle. We can moreover define j' such that $\mathbf{y}^A[j']$ is added to $\mathrm{ToBeAssigned}^-_{A_{j'}}$ after any other value $\mathbf{y}^A[j]$ is added to $\mathrm{ToBeAssigned}^-_{A_j}$ during that query cycle. Then we have $\mathbf{y}^A[j] \in \mathrm{Index}^-_{A_j}$ for all j when $\mathbf{y}^A[j']$ is added to $\mathrm{ToBeAssigned}^-_{A_{j'}}$, since $\mathbf{y}^A[j] \in \mathrm{Index}^-_{A_j}$ for all j at the end of the query cycle (and being added to $\mathrm{Index}^-_{A_j}$ can only happen by being added to $\mathrm{ToBeAssigned}^-_{A_j}$, for a left-hand query cycle). Thus the call to $\mathrm{RequestU}^{-1}$ that adds $\mathbf{y}^A[j']$ to $\mathrm{ToBeAssigned}^-_{A_j}$ will add a pair $(\mathbf{x}^G, \mathbf{y}^U)$ to $\mathrm{ToBeAdapted}_D$ with $\mathbf{y}^U = \mathbf{y}^A$. Assuming successful completion of the query cycle, a completed chain containing $\mathbf{y}^A$ is thus created by AdaptLeft. □

Lemma 17. *The total number of pairs* $(\mathbf{x}^G, \mathbf{y}^U)$ *ever added to* ToBeAdapted_D *plus the total number of pairs* $(\mathbf{x}^V, \mathbf{y}^H)$ *ever added to* ToBeAdapted_E *in* G_1*,* G_2 *or* G_3 *is at most* $(2q)^w$ *if* $\mathsf{XtraMiddleRnd}$ *is off and is at most* $\mathsf{Cond}_\tau(2q)$ *is* $\mathsf{XtraMiddleRnd}$ *is on.*

Proof. When $\mathsf{XtraMiddleRnd}$ is off this is a direct corollary of lemmas 14 and 15.

When $\mathsf{XtraMiddleRnd}$ is on this follows from Lemma 15 and from the fact that

$$\left\{\mathbf{y}^U : \mathbf{y}^U \in \text{Index}^-_{U_1} \times \ldots \times \text{Index}^-_{U_w} \wedge \tau(\mathbf{y}^U) \in \text{Index}_{V_1} \times \ldots \times \text{Index}_{V_w}\right\}$$

has size at most $\mathsf{Cond}_\tau(2q)$ by the definition of conductance. □

In other words, the total number of pairs $(\mathbf{x}^G, \mathbf{y}^U)$ added to ToBeAdapted_D plus the total number of pairs $(\mathbf{x}^V, \mathbf{y}^H)$ added to ToBeAdapted_E is at most $\alpha(q)$.

Lemma 18. *The number of times lines 8–11 of* RequestU^{-1} *plus the number of times lines 8–9 of* RequestV *are executed is at most* $\alpha(q)$*.*

Proof. This is a corollary of lemmas 17 and 12, since lines 8–11 of RequestU^{-1} and lines 8–9 of RequestV are executed each time before an element is added respectively to ToBeAdapted_D or ToBeAdapted_E. □

For the next two lemmas, let AllAdapted be the set of all pairs $(\mathbf{x}^G, \mathbf{y}^U)$ ever added to ToBeAdapted_D unioned with the set of all pairs $(\mathbf{x}^V, \mathbf{y}^H)$ ever added to ToBeAdapted_E.

Lemma 19. *For every* $T \in \{D, \ldots, L\}$ *and for every* $j \in [w]$ *there is an injection from the set of pairs* (x, y) *such that* $T_j(x) = y$ *to* $\text{AllAdapted} \cup [q]$*, at any point in time of any execution of* G_1*,* G_2 *or* G_3*.*

Proof. If the distinguisher previously queried $\text{T}(j, x)$ or $\text{T}^{-1}(y)$ with (say) his h-th query, we map the pair (x, y) to the value $h \in [q]$.

Otherwise the distinguisher did not query $\text{T}(j, x)$ or $\text{T}^{-1}(y)$, and the entry was added to T_j either by line 17 of RequestG, by line 17 of RequestH^{-1}, by line 8 of RequestU^{-1}, by line 8 of RequestV, by line 11 of RequestU^{-1}, by line 11 of RequestV, or else inside the second **forall** loop of AdaptLeft or AdaptRight. (Recall that $T \neq A, B, C$.) However, for each of these cases it is straightforward (by direct examination of the pseudocode) to associate either some pair $(\mathbf{x}^G, \mathbf{y}^U)$ added to ToBeAdapted_D or some pair $(\mathbf{x}^V, \mathbf{y}^H)$ added to ToBeAdapted_E. Moreover it is equally easy to see that under this natural mapping, no element of AllAdaptedis the image of more than one pair (x, y), for a given T and j (in a nutshell, because the query $T_j(x) = y$ lies on the path corresponding to the element in AllAdapted, which limits x and y to one possible combination). □

Lemma 20. *Let* $T \in \{D, \ldots, L\}$*,* $j \in [w]$*. Then the number of* x *for which* $T_j(x) \neq \perp$ *is at most* $q + \alpha(q)$ *at any point of any execution of* G_1*,* G_2 *and* G_3*. Moreover if* $T \in \{A, B, C\}$ *then the same number is at most* $2q$*.*

Proof. The first statement follows by Lemma 19 coupled with Lemma 17. The second statement is a restatement of Lemma 9. □

Lemma 21. *In any execution of* G_1*,* G_2 *or* G_3 *the total number of distinct*[18] *calls from* RequestU^{-1} *to* BlockRequestG *and from* RequestV *to* $\text{BlockRequestH}^{-1}$ *is at most* $\alpha(q)$*.*

[18] Distinct means with distinct arguments. The total number of calls may be greater.

Proof. This follows from Lemma 17 since for each call BlockRequestG($\mathbf{x}^G$) there is a pair $(\mathbf{x}^G, \mathbf{y}^U) \in$ ToBeAdapted$_D$ for some $\mathbf{y}^U$ and for each call BlockRequestH^{-1}($\mathbf{y}^H$) there is a pair $(\mathbf{x}^V, \mathbf{y}^H) \in$ ToBeAdapted$_E$ for some $\mathbf{x}^V$. □

Lemma 22. *For any $j \in [w]$ and at any point in an execution of* G_1*,* G_2 *or* G_3*,* Index$_{G_j}$ *and* Index$^{-}_{H_j}$ *have size at most $q + \alpha(q)$.*

Proof. The size of Index$_{G_j}$ is upper bounded by the number of distinguisher calls to G and G^{-1} plus the number of distinct values x for which a call RequestG(j, x) occurs plus the number of times BlockG^{-1} is called. The number of distinguisher calls to G and G^{-1} is at most q in total. Moreover, since RequestG is otherwise (that is, when it isn't being called by G because of a direct distinguisher query to G) only called by BlockRequestG and the end of RequestU^{-1}, and since BlockG^{-1} is only called from within MiddleToLeft from within RequestV, the conclusion follows by Lemma 21. □

Lemma 23. *In any execution of* G_1*,* G_2 *or* G_3*, the total number of distinct pairs $(\mathbf{x}^X, \mathbf{y}^H)$ such that a call CheckZ$(\mathbf{x}^X, \mathbf{y}^H)$ occurs is at most $\beta(q)(q + \alpha(q))^w$.* □

Proof. CheckZ is only called from within on arguments of the type $(\mathbf{x}^X, \mathbf{y}^H)$, where $\mathbf{y}^H$ satisfies BlockDefined$(H, \mathbf{y}^H, -) =$ **true**. There are at most $(q + \alpha(q))^w$ distinct such $\mathbf{x}^X$ over the course of an execution by Lemma 20.

Moreover, if XtraOuterRnd is off then BlockDefined$(G, \mathbf{x}^X, +) =$ **true** and there are at most $\beta(q) = (q + \alpha(q))^w$ distinct such $\mathbf{x}^X$ over the course of an execution by the same lemma.

If XtraOuterRnd is on, finally, CheckZ is only called on arguments of then $\mathbf{x}^X$ satisfies BlockDefined$(F, \mathbf{x}^X, +)$ **true** and BlockDefined$(G, \nu(\mathrm{BlockF}(\mathbf{x}^X)), +) =$ **true**. Obviously there are at most $\beta(q) = \mathsf{Cond}_\nu(\alpha(q)+ q)$ distinct such $\mathbf{x}^X$ by Lemma 20 and by the definition of conductance. □

5.3 Bounding the Abort Probability in $\mathbf{G_3}$

Lemma 24. *An execution of* G_3 *never aborts inside* ReadTape *unless the function calling* ReadTape *is* D*,* D^{-1}*,* E *or* E^{-1}.

Proof. Note that values in $A_j, B_j, C_j, F_j, G_j, H_j, I_j, J_j, K_j, L_j$ and P_Z are always set via ReadTape; hence, ReadTape never aborts except the second argument to ReadTape is D_j or E_j, and such calls to ReadTape only occur in procedures eD, D^{-1}, E and E^{-1}. □

Lemma 25. *The number of times* Z/Z^{-1} *is called during an execution of* G_2 *or* G_3 *is at most $q + \alpha(q)$.*

Proof. Note that Z is no longer called by CheckZ in game G_2 and G_3. Hence, the only calls to Z are those which occur in RequestU^{-1} and RequestV, or else are distinguisher calls. The number of former calls is at most $\alpha(q)$ by Lemma 17 while the latter is at most q. □

Lemma 26. *The probability that* G_2 *or* G_3 *aborts inside of* Z *or* Z^{-1} *is at most $w(q+\alpha(q))^2/(N - q - \alpha(q))$.*

Proof. Note that values in P_Z are always set via ReadTape and never via SetTable; hence, calls to ReadTape never abort in Z or Z^{-1}.

The number of entries in any of the tables X_j, Y_j is at most $q + \alpha(q)$ by Lemma 20. Moreover, a lookup in p_Z produces a value $z_1 \dots z_w \in \{0, 1\}^{wn}$ where each z_i is uniform in a set of size at least

$N - q - \alpha(q)$ by Lemma 25. Thus, by a union bound over the entries in X_j (for queries to Z^{-1}) or over the entries in Y_j (for queries to Z) as well as over the value of j the probability that a query to Z or Z^{-1} causes an abort is at most $w(q + \alpha(q))/(N - q - \alpha(q))$. The total number of queries to $\mathrm{Z}/\mathrm{Z}^{-1}$ being at most $q + \alpha(q)$, the lemma follows by a union bound over all these queries. □

Lemma 27. *The procedure* CheckZ *is never called twice on the same input pair* $(\mathbf{x}, \mathbf{y})$ *over the course of an execution of* G_3*.*

Proof. Calls to CheckZ occur only in RequestG and $\mathrm{RequestH}^{-1}$. Oviously, the given call to RequestG or $\mathrm{RequestH}^{-1}$ will not call CheckZ twice on the same pair $(\mathbf{x}^X, \mathbf{y}^H)$, as is clear by inspection of the pseudocode. Therefore if CheckZ is called twice on the same pair $(\mathbf{x}^X, \mathbf{y}^H)$, the two calls must occur from within distinct instances of RequestG and/or $\mathrm{RequestH}^{-1}$. For concreteness, assume to start with that the two calls are issued by distinct instances of RequestG. (This is in fact the more interesting case, since if one call occurs within RequestG and the other occurs within $\mathrm{RequestH}^{-1}$, the calls are happening in different query cycles.)

Let the two calls to RequestG in question be $\mathrm{RequestG}(j_1, x_1)$ and $\mathrm{RequestG}(j_2, x_2)$, where $\mathrm{RequestG}(j_1, x_1)$ starts executing before $\mathrm{RequestG}(j_2, x_2)$; we do not assume $(j_1, x_1) \neq (j_2, x_2)$.

Since $\mathrm{RequestG}(j_1, x_1)$ does not start making recursive calls until its last **forall** loop, the call $\mathrm{CheckZ}(\mathbf{x}^X, \mathbf{y}^H)$ that occurs in $\mathrm{RequestG}(j_1, x_1)$ occurs before $\mathrm{RequestG}(j_2, x_2)$ starts executing. At the point when the call $\mathrm{CheckZ}(\mathbf{x}^X, \mathbf{y}^H)$ is made (or even when $\mathrm{RequestG}(j_1, x_1)$ is called) $\mathbf{x}^X$ is associated to a unique $\mathbf{x}^G$, which is namely $\mathbf{x}^G = \nu(F(\mathbf{x}^X))$ if $\mathsf{XtraOuterRnd}$ is on and which is $\mathbf{x}^G = \mathbf{x}^X$ if $\mathsf{XtraOuterRnd}$ is off. Moreover $\mathbf{x}^G[j_1] = x_1$ by design of RequestG, and $\mathbf{x}^G[j_2] = x_2$ because $\mathrm{RequestG}(j_2, x_2)$ also makes the call $\mathrm{CheckZ}(\mathbf{x}^X, \mathbf{y}^H)$ by assumption. But since $\mathbf{x}^G[j] \in \mathrm{Index}_{G_j}$ for all j when $\mathrm{RequestG}(j_1, x_1)$ calls $\mathrm{CheckZ}(\mathbf{x}^X, \mathbf{y}^H)$, this implies that $x_2 \in \mathrm{Index}_{G_{j_2}}$ when $\mathrm{RequestG}(j_2, x_2)$ is called, since the latter call occurs after $\mathrm{RequestG}(j_1, x_1)$ calls $\mathrm{CheckZ}(\mathbf{x}^X, \mathbf{y}^H)$, and hence $\mathrm{RequestG}(j_2, x_2)$ would return immediately, a contradiction.

Other cases (e.g., when one call to CheckZ is caused by RequestG, and the other by $\mathrm{RequestH}^{-1}$) can be handled by a similar analysis. □

Lemma 28. *The probability that* G_3 *aborts inside of* ForwardOutside *or* BackwardOutside *is at most* $(q + w\alpha(q))(q + \alpha(q))/(N - q - \alpha(q))$.

Proof. $\mathrm{BackwardOutside}(T, j, y)$ is called with $T \in \{F, G\}$ while $\mathrm{ForwardOutside}(T, j, x)$ is called with $T = H$. By Lemma 20, values read from $p^-_{F_j}$, $p^-_{G_j}$ or p_{H_j} come uniformly at random from a set of size at least $N - q - \alpha(q)$. Since P_Z contains at most $q + \alpha(q)$ entries by Lemma 25, the abort probability in a query to ForwardOutside or BackwardOutside is at most $(q + \alpha(q))/(N - q - \alpha(q))$.

The total number of queries to ForwardOutside and BackwardOutside is no more than $q + w\alpha(q)$, including no more than q queries directly issued by the distinguisher and, by Lemma 18, no more than $w\alpha(q)$ queries made by MiddleToLeft and MiddleToRight. The lemma thus follows by a union bound. □

Lemma 29. *When* $\mathsf{XtraOuterRnd}$ *is on, the probability that* G_3 *aborts inside of* F *or* G^{-1} *is at most*

$$\frac{2w(q + \alpha(q))\mathit{aboCond}_\nu(q + \alpha(q))}{N - q - \alpha(q)}. \tag{4}$$

Proof. We focus on a call $\mathrm{F}(j, x)$. Calling $\mathrm{F}(j, x)$ causes abort if and only if $F_j(x) = \bot$ and if once we set $F_j(x) = y := p_{F_j}(x)$, there exist $\mathbf{y}^F, \mathbf{x}^G$ such that $\mathbf{y}^F[j] = y$, $\mathbf{x}^G = \nu(\mathbf{y}^F)$ and

BlockDefined$(F, \mathbf{y}^F, -)$ = BlockDefined$(G, \mathbf{x}^G, +)$ = **true**. Since there is no such thing as a set ToBeAssigned$^-_{F_j}$, the set of values y that will cause an abort is

$$\left\{y \in \{0,1\}^n : \exists \mathbf{y}^F, \mathbf{x}^G \text{ s.t. } \mathbf{y}^F[j] = y, \nu(\mathbf{y}^F) = \mathbf{x}^G, \forall k \neq j \; F_k^-(\mathbf{y}^F[k]) \neq \bot, \forall k \; \mathbf{x}^G[k] \in \text{Index}_{G_k}\right\}.$$

The size of this set can be upper bounded by the size of

$$\left\{(\mathbf{y}^F, \mathbf{x}^G) : \nu(\mathbf{y}^F) = \mathbf{x}^G, \forall k \neq j \; F_k^-(\mathbf{y}^F[k]) \neq \bot, \forall k \; \mathbf{x}^G[k] \in \text{Index}_{G_k}\right\}$$

which is upper bounded by $\mathsf{aboCond}_\nu(q + \alpha(q))$ by the definition of all-but-one conductance and by lemmas 20, 22. Moreover y comes uniformly at random from a set of size at least $N - q - \alpha(q)$ by Lemma 20. Thus the probability of abort for a call to $\mathrm{F}(j, \cdot)$ is no more than $\mathsf{Cond}_\nu(q + \alpha(q))/(N - q - \alpha(q))$, and it is easy to see that the same bound holds for a call to $\mathrm{G}^{-1}(j, \cdot)$. Multiplying by $q + \alpha(q)$ to account for all calls to $\mathrm{F}(j, \cdot)$ and by a further factor of $2w$ to account for all values of j as well as for calls to G^{-1} yields the final bound. □

Lemma 30. *The probability that* $\mathsf{NumOuter}$ *exceeds* q *in* G_3 *is zero.*

Proof. Each increment of $\mathsf{NumOuter}$ is associated to a unique pair $(\mathbf{x}^X, \mathbf{y}^H)$ such that CheckZ$(\mathbf{x}^X, \mathbf{y}^H)$ = **true**. Moreover a given pair $(\mathbf{x}^X, \mathbf{y}^H)$ only contributes at most once to the increase in $\mathsf{NumOuter}$ by Lemma 27. We claim that before the call CheckZ$(\mathbf{x}^X, \mathbf{y}^H)$ occurs the distinguisher has already made the call $\mathrm{Z}(\mathbf{x}^X) \to \mathbf{y}^H$ or $\mathrm{Z}^{-1}(\mathbf{y}^H) \to \mathbf{x}^X$. This would complete the proof since the distinguisher has at most q queries.

To prove the latter claim, consider wlog the case where $\mathsf{NumOuter}$ is incremented within a left-hand query cycle, so that the call to CheckZ occurs from within RequestG. Assume by contradiction that the distinguisher did not make the call $\mathrm{Z}(\mathbf{x}^X) \to \mathbf{y}^H$ or $\mathrm{Z}^{-1}(\mathbf{y}^H) \to \mathbf{x}^X$, so that the entry $P_Z(\mathbf{x}^X) = \mathbf{y}^H$ which results in CheckZ$(\mathbf{x}^X, \mathbf{y}^H)$ returning **true** (see the pseudocode of procedure CheckZ in G_3, Fig. 2) was created by a call to Z^{-1} in RequestU^{-1} or by a call to Z in RequestV, as no other simulator functions call $\mathrm{Z}/\mathrm{Z}^{-1}$ in G_3. In the latter case the call to Z must have occured during a previous query cycle because RequestV is a right-hand function while RequestG is a left-hand function and, thus, the pair $(\mathbf{x}^X, \mathbf{y}^H)$ was in a completed path already at the beginning of the current query cycle, a contradiction to the first line of RequestG since values are never removed from the tables T_j and since the vector $\mathbf{x}^X$ would have already been associated to a unique vector $\mathbf{x}^G$ such that $G(\mathbf{x}^G) \neq \bot$ at the beginning of the query cycle. (I.e., the vector $\mathbf{x}^G$ on the same completed path as $\mathbf{x}^X$.) In the other case, when the entry $P_Z(\mathbf{x}^X) = \mathbf{y}^H$ was created by a call to Z^{-1} in RequestU^{-1}, then either that query to Z^{-1} occured in a previous query cycle, in which case the same argument applies, or else $(\mathbf{x}^X, \mathbf{y}^U) \in \text{ToBeAdapted}_D$ for some vector $\mathbf{y}^U$, contradicting the fact that RequestG made it to the increment of $\mathsf{NumOuter}$. This completes the proof. □

Lemma 31. *The probability that an execution of* G_3 *aborts inside either of* D, D^{-1}, E *or* E^{-1} *is, altogether, at most* $2w\alpha(q)(q + \alpha(q))/(N - q - \alpha(q))$.

Proof. We fix a value of j and focus on calls $\mathrm{D}(j, \cdot)$, $\mathrm{D}^{-1}(j, \cdot)$, and later multiply by $2w$ (union bound) to obtain the final bound.

Note that at most $\alpha(q)$ entries of D_j are fixed by SetTable, since at most $\alpha(q)$ paths are completed by Lemma 17. Moreover, every lookup in the table p_{D_j} or $p^{-1}_{D_j}$ gives a value drawn uniformly at random from a set of size at least $N - q - \alpha(q)$ by Lemma 20. Hence the probability that an abort occurs because a drawn value "hits" a predefined value within D or D^{-1} is at most $\alpha(q)/(N - q - \alpha(q))$ per query. Furthermore the number of queries to $\mathrm{D}(j, \cdot)$ or $\mathrm{D}^{-1}(j, \cdot)$ that result in a call ReadTable is at most $q + \alpha(q)$ again by Lemma 20. Multiplying by $2w$ to account for the different values of j and for queries to D gives the union bound its final form. □

Lemma 32. *When* AdaptLeft *is called, for every pair* $(\mathbf{x}^G, \mathbf{y}^U) \in \text{ToBeAdapted}_D$ *and for every* $1 \leq j \leq w$*, either* $\mathbf{x}^G[j] \in \text{ToBeAssigned}_{G_j}$ *or* $G_j(\mathbf{x}^1[j]) \neq \perp$*, but not both, and, similarly, either* $\mathbf{y}^U[j] \in \text{ToBeAssigned}_{U_j}^{-1}$ *or* $U_j^{-1}(\mathbf{y}^U[j]) \neq \perp$*, but not both. A symmetric statement holds for calls to* AdaptRight *and pairs* $(\mathbf{x}^V, \mathbf{y}^H) \in \text{ToBeAdapted}_E$*.*

Proof. The "but not both" clauses follows from Lemma 6. Subsequently, the statement is obvious for $\mathbf{y}^U$, given that the **forall** loop of RequestU^{-1} doesn't proceed unless BlockDefined$(U, \mathbf{y}^U, -) =$ **true**. Moreover, once a pair $(\mathbf{x}^G, \mathbf{y}^U)$ is added to ToBeAdapted$_D$ the call BlockRequestG$(\mathbf{x}^G)$ occurs, which implies (by direct inspection of BlockRequestG and of RequestG) that the statement also hold for $\mathbf{x}^G$. □

Lemma 33. *The probability that an execution of* G$_3$ *aborts inside of* BlockU^{-1}*,* BlockV*,* BlockG *or* BlockH^{-1} *is zero.*

Proof. First we consider the call to BlockG in RequestH^{-1} via LeftToMiddle. Since RequestH^{-1} is only called during a right-hand query cycle, the tables ToBeAssigned$_{G_j}$ remain empty during a right-hand query cycle. Hence if BlockDefined$(G, \mathbf{x}^G, +) =$ **true** in RequestH^{-1} it implies that $G_j(\mathbf{x}^G[j]) \neq \perp$ for $1 \leq j \leq w$. This implies that the call to BlockG doesn't abort within RequestH^{-1}.

When XtraMiddleRnd is on, BlockU^{-1} can be called in RequestV via MiddleToLeft. Since RequestV is only called during a right-hand query cycle, the tables ToBeAssigned$_{U_j}^{-1}$ remain empty during a right-hand query cycle. Hence if BlockDefined$(B, \mathbf{y}^B, -) =$ **true** in RequestV it implies that $B_j(\mathbf{y}^B[j]) \neq \perp$ for $1 \leq j \leq w$. This implies that the call to BlockU^{-1} never aborts within RequestV.

Otherwise, BlockU^{-1}, BlockG are only called in AdaptLeft. The claim thus follows by Lemma 32 and by direct inspection of AdaptLeft.

Calls to BlockV, BlockH^{-1} are symmetrically analyzed. □

Lemma 34. *When* XtraMiddleRnd *is on, the probability that* G$_3$ *aborts inside of* B *or* C^{-1} *is at most*

$$\frac{4wq \cdot \mathsf{aboCond}_\tau(2q)}{N - 2q}.$$

Proof. By Lemma 24, ReadTape does not abort when called by B or C^{-1}. By Lemma 9, values read from the tape p_{B_j} or from the tape $p_{C_j}^{-1}$ come uniformly at random from a set of size at least $N - 2q$.

Calling B(j, x) causes abort if and only $B_j(x) = \perp$ and if once we set $B_j(x) = y := p_{B_j}(x)$, there exists a pair $\mathbf{y}^B$, $\mathbf{x}^C$ such that $\mathbf{y}^B[j] = y$, $\mathbf{x}^C = \tau(\mathbf{y}^B)$ and BlockDefined$(B, \mathbf{y}^B, -) =$ BlockDefined$(C, \mathbf{x}^C, +) =$ **true**. As in Lemma 29, the number of such y's is upper bounded by the size of the set

$$\left\{(\mathbf{y}^B, \mathbf{x}^C) : \tau(\mathbf{y}^B) = \mathbf{x}^C, \forall k \neq j\ \mathbf{y}^B[k] \in \text{Index}_{B_k}^{-1}, \forall k\ \mathbf{x}^C[k] \in \text{Index}_{C_k}\right\}$$

which is upper bounded by $\mathsf{aboCond}_\tau(2q)$ by the definition of all-but-one conductance and by Lemma 9. Hence the probability of abort for a single call to B is no more than $\mathsf{aboCond}_\tau(2q)/(N - 2q)$. Moreover at most $2q$ distinct queries to B$(j, \cdot)$ occur by Lemma 9 again, so the final bound follows by symmetry between B and C^{-1} and by a straightforward union bound. □

Lemma 35. *During a left-hand query cycle, entries are not added to tables* G_j *or* U_j *until* AdaptLeft *is called. Moreover, the only entries added to* H_j *are via forward calls to* H*. A symmetric statement holds for right-hand query cycles.*

Proof. Focusing on left-hand query cycles, the first statement follows by inspection of the procedures RequestG and RequestU^{-1}. The second statement follows by inspection of the same procedures and by noting that the procedure BlockH^{-1} (called in RequestG via RightToMiddle) does not actually call H^{-1}, but performs look-ups in the tables H_j^{-1} instead. □

Lemma 36. *At the end of each non-aborted query cycle in* G_3*, every pair* $(\mathbf{x}^G, \mathbf{y}^H)$ *such that*

1. $G_j(\mathbf{x}^G[j]) \neq \perp$ *and* $H_j^{-1}(\mathbf{y}^H[j]) \neq \perp$ *for all* $1 \leq j \leq w$;
2. *If* XtraOuterRnd *is on and* $\mathbf{y}^F = \nu^{-1}(\mathbf{x}^G)$ *then* $F_j^{-1}(\mathbf{y}^F[j]) \neq \perp$ *for all* $1 \leq j \leq w$;
3. $P_Z(\mathbf{x}^X) = \mathbf{y}^H$, *where* $\mathbf{x}^X = F^{-1}(\mathbf{y}^F)$ *if* XtraOuterRnd *is on,* $\mathbf{x}^X = \mathbf{x}^G$ *otherwise*

is in a completed path.

Proof. We prove the statement by induction on the number of distinguisher queries. We refer to the lemma's assertion as "the invariant" in the proof below. Obviously, the invariant holds before the distinguisher starts making queries.

Distinguisher queries to Z and Z^{-1} do not affect the invariant, thanks to the abort conditions installed in those functions in game G_3. Nor do distinguisher queries to G^{-1}, H and F, F^{-1} thanks to similar abort conditions installed in those functions.

Other neutral calls obviously leave the lemma's invariant unaffected as well, since such calls don't affect any of the tables on which the invariant depends.

For the rest we focus on a left-hand query (i.e., to G or U^{-1}), the case of a right-hand query being symmetric.

Let $(\mathbf{x}_*^G, \mathbf{y}_*^H)$ be a putative pair that satisfies the lemma's constraints at the end of the query cycle, but such that $\mathbf{x}_*^G$ and $\mathbf{y}_*^H$ don't lie in a (common) completed path at the end of the query cycle.

At the end of a non-aborted query cycle the condition $G_j(\mathbf{x}_*^G[j]) \neq \perp$ is equivalent to $\mathbf{x}_*^G[j] \in \mathrm{Index}_{G_j}$. So $\mathbf{x}_*^G$ and $\mathbf{y}_*^H$ satisfy

1. $\mathbf{x}_*^G[j] \in \mathrm{Index}_{G_j}$ for all $j \in [w]$;
2. $H_j^{-1}(\mathbf{y}_*^H[j]) \neq \perp$ for all $j \in [w]$;
3. if XtraOuterRnd is on then $F^{-1}(\mathbf{y}_*^F) \neq \perp$ where and $\mathbf{y}_*^F = \nu^{-1}(\mathbf{x}_*^G)$;
4. $P_Z(\mathbf{x}_*^X) = \mathbf{y}_*^H$, where $\mathbf{x}_*^X = F^{-1}(\mathbf{y}_*^F)$ if XtraOuterRnd is on, and where $\mathbf{x}_*^X = \mathbf{x}_*^G$ otherwise

at the end of the query cycle. Moreover, by the hypothesis that the invariant holds at the start of the query cycle, one or more of these constraints doesn't hold at the start of the query cycle.

Consider the first time the pair $(\mathbf{x}_*^G, \mathbf{y}_*^H)$ satifies constraints 1–4. Note this cannot occur in H, F, F^{-1}, Z and Z^{-1} because of the abort conditions in those functions. (As far as the last claim is concerned, in fact, constraint 1 could be replaced with "$G_j(\mathbf{x}_*^G[j]) \neq \perp$" as well, which would only make the constraints harder to satisfy. One can also note that F^{-1} is never actually called during a left-hand query cycle.)

As argued in Lemma 35, H^{-1} and RequestH^{-1} are never called during a left-hand query cycle. Moreover AdaptLeft leaves Index_{G_j} and H_j, F_j and P_Z invariant, so $(\mathbf{x}_*^G, \mathbf{y}_*^H)$ must satisfy the constraints 1–4 before AdaptLeft is called.

We have ruled out all the functions that affect the tables F_j, H_j and P_Z (being respectively F, H and Z/Z^{-1}) during a left-hand query cycle as being the point where constraints 1–4 become satisfied for the first time, and we know the constraints become satisfied before AdaptLeft. The only remaining possibility is that the constraints become satisfied when $\mathbf{x}_*^G[k]$ is added to $\mathrm{ToBeAssigned}_{G_k}$

for some k. But when $\mathbf{x}_*^G[k]$ is added to ToBeAssigned$_{G_k}$ inside RequestG$(k, \mathbf{x}_*^G[k])$, and since by assumption all the constraints 1–4 are satisfied at that moment, the pair $(\mathbf{x}_*^G, \mathbf{y}_*^U)$ would be added to ToBeAdapted$_D$ for some $\mathbf{y}_*^U$ and, if abort doesn't occur, $\mathbf{x}_*^G$ must be in a completed path at the end of the query cycle, which would perforce include $\mathbf{y}_*^H$. □

Lemma 37. *When* AdaptLeft *is called in* G$_3$*, for every pair* $(\mathbf{x}^G, \mathbf{y}^U) \in$ ToBeAdapted$_D$ *there exists a* $j \in [w]$ *such that* $\mathbf{x}^G[j] \in$ ToBeAssigned$_{G_j}$ *and there exists* $k \in [w]$ *such that* $\mathbf{y}^U[k] \in$ ToBeAssigned$^{-1}_{U_k}$*. A symmetric statement holds for calls to* AdaptRight *and pairs* $(\mathbf{x}^V, \mathbf{y}^H) \in$ ToBeAdapted$_E$*.*

Proof. The claim is clear for vector $\mathbf{y}^U$ by inspection of RequestU^{-1}, where pairs are added to ToBeAdapted$_D$. Indeed every addition of a pair $(\mathbf{x}^G, \mathbf{y}^U)$ to ToBeAdapted$_D$ is preceded by adding $\mathbf{y}^U[j]$ to ToBeAssigned$^-_{U_j}$ for some j, and $\mathbf{y}^U[j]$ is not removed from ToBeAssigned$^-_{U_j}$ until AdaptLeft is called. We also note that in this case the pair $(\mathbf{x}^G, \mathbf{y}^U)$ is added to ToBeAdapted$_D$ during the call RequestU$^{-1}(j, \mathbf{y}^U[j])$.

In order to argue a similar conclusion for the vector $\mathbf{x}^G$ of the pair $(\mathbf{x}^G, \mathbf{y}^U)$ in question we can consider the point in time after $\mathbf{x}^G$ has been computed in RequestU$^{-1}(j, \mathbf{y}^U[j])$ but before BlockRequestG$(\mathbf{x}^G)$ is called.

If at this point $G_h(\mathbf{x}^G[h]) = \perp$ for some $h \in [w]$ then BlockRequestG$(\mathbf{x}^G)$ will add $\mathbf{x}^G[h]$ to ToBeAssigned$_{G_h}$, and the conclusion follows as well for $\mathbf{x}^G$. Therefore we can assume that $G_h(\mathbf{x}^G[h]) \neq \perp$ for all $h \in [w]$ at this point, i.e., that $G(\mathbf{x}^G) \neq \perp$ at this point. Since new entries aren't added to the tables G_h before AdaptLeft is called, this also implies that $G(\mathbf{x}^G) \neq \perp$ already at the start of the query cycle.

However, RequestU$^{-1}(j, \mathbf{y}^U[j])$ has computed a (forward) path going all the way from $\mathbf{y}^U$ to $\mathbf{x}^G$ via a vector $\mathbf{y}^H$ (computed in MiddleToRight), and one can check that the pair $(\mathbf{x}^G, \mathbf{y}^H)$ now meets the conditions 1–4 listed in the previous lemma. But following the same argument as in the proof of that lemma (i.e., seeking the first point in time at which $(\mathbf{x}^G, \mathbf{y}^H)$ meets conditions 1–4) leads us to a contradiction, for we concluded there the existence of some value k such that $\mathbf{x}^G[k]$ is added to ToBeAssigned$_{G_k}$ during the query cycle, which cannot be the case here if $G(\mathbf{x}^G) \neq \perp$ at the start of the query cycle. □

Lemma 38. *The probability that abort occurs in* AdaptLeft *or in* AdaptRight *in an execution of* G$_3$ *is no more than*

$$\frac{2w\alpha(q)(q+\alpha(q)) \cdot \mathsf{MaxPreim}(\overline{\pi})}{N-q-\alpha(q)} + \frac{2w\alpha(q)^2 \cdot \mathsf{MaxCoPr}(\overline{\pi})}{N-q-\alpha(q)}$$

if $\mathsf{XtraUntglRnds}$ *is off, and no more than*

$$\frac{(4w\alpha(q) + 2w\alpha(q)^2) \cdot \mathsf{MaxPreim}(\overline{\pi})(q+\alpha(q))}{N-q-\alpha(q)}$$

if $\mathsf{XtraUntglRnds}$ *is on.*

Proof. As all values of G_j, U_j, I_j, J_j are set by ReadTape, the calls to ReadTape that occur in AdaptLeft cannot cause abort (Lemma 24).

Moreover, by Lemma 33, the calls to BlockG and BlockU^{-1} that occur in AdaptLeft cannot cause abort either. Therefore, the only calls that might cause AdaptLeft to abort are the calls to SetTable. Specifically, a call SetTable(D_j, x, y) will abort if either the x- or y-argument already appears in D_j. Arguing by symmetry, we will bound the probability that $D_j(x) \neq \perp$.

For concreteness, assume that ToBeAdapted_D contains k pairs

$$(\mathbf{x}_1^G, \mathbf{y}_1^U), \ldots, (\mathbf{x}_k^G, \mathbf{y}_k^U)$$

when AdaptLeft is called. Let

$$\mathcal{S}_i = \{j : 1 \le j \le w, G_j(\mathbf{x}_i^G[j]) = \bot\}$$

Note that $\mathcal{S}_i \neq \emptyset$ for $1 \le i \le k$ by Lemma 37.

Let $\mathcal{D} \subseteq \{0,1\}^n$ be the domain of D_j when AdaptLeft is called and, if XtraUntglRnds is on, let $\mathrm{I} \subseteq \{0,1\}^n$ be the domain of I_j when AdaptLeft is called. In other words,

$$\mathcal{D}_j = \{x \in \{0,1\}^n : D_j(x) \neq \bot\}$$
$$\mathcal{I}_j = \{x \in \{0,1\}^n : I_j(x) \neq \bot\}$$

with these "snapshots" of D_j and I_j being taken when AdaptLeft is called. (Thus subsequent modifications to D_j, I_j do not affect $\mathcal{D}_j, \mathrm{I}_j$.)

Define

$$\begin{aligned} \mathbf{y}_i^G &= G(\mathbf{x}_i^G) & \\ \mathbf{x}_i^D &= \pi_G(\mathbf{y}_i^G) & \text{if XtraUntglRnds is off} \\ \mathbf{x}_i^I &= \pi_G(\mathbf{y}_i^G) & \text{if XtraUntglRnds is on} \\ \mathbf{y}_i^I &= I(\mathbf{x}_i^I) & \text{if XtraUntglRnds is on} \\ \mathbf{x}_i^D &= \pi_I(\mathbf{y}_i^I) & \text{if XtraUntglRnds is on} \end{aligned}$$

If XtraUntglRnds is off, $\mathbf{y}_i^G$ and $\mathbf{x}_i^D$ become defined after the first **forall** loop of AdaptLeft has completed. If XtraUntglRnds is on, $\mathbf{y}_i^G$ and $\mathbf{x}_i^I$ become defined after the **forall** loop has completed, while $\mathbf{y}_i^I$ and $\mathbf{x}_i^D$ only become defined within the second (outermost) **forall** loop of AdaptLeft.

In either case, the underlying probability space for the following argument is all the coins thrown during the AdaptLeft call. One can think of $\mathbf{y}_i^G$, $\mathbf{x}_i^I$, $\mathbf{y}_i^I$, $\mathbf{x}_i^D$ as random variables over this probability space. Moreover, for the sake of having these random values always be well-defined, we will assume that $\text{SetTable}(D_j, \cdot, \cdot)$ doesn't actually stop the execution of AdaptLeft if **abort** occurs, but throws some kind of uncaught exception instead. This way both **forall** loops fully complete and all afore-mentioned values are defined, modulo the flag XtraUntglRnds.

We define the following bad events:

- $\mathsf{PreExisting}_{i,j}$ occurs if and only if $\mathbf{x}_i^D[j] \in \mathcal{D}_j$ $(i \in [k], j \in [w])$
- $\mathsf{Colliding}_{i,i',j}$ occurs if and only if $\mathbf{x}_i^D[j] = \mathbf{x}_{i'}^D[j]$ $(1 \le i < i' \le k,\ j \in [w])$

It is easy to see that some call $\text{SetTable}(D_j, x, y)$ aborts because $D_j(x) \neq \bot$ if and only if an event from these two families occurs.

First we bound the probability of one of these events occuring when XtraUntglRnds is off. We have $|\mathcal{D}_j| \le q + \alpha(q)$ by Lemma 20. Note that $\mathbf{x}_i^D[j]$ only becomes defined (and for all j at once) once $\text{ReadTape}(G_j, \mathbf{x}_i^G[j], p_{G_j})$ has been called for all $j \in \mathcal{S}_i$. The last such call to ReadTape, for a given value of i, will be called the *defining* call for $\mathbf{x}_i^D$.

As in Section 2, let

$$\pi_{j,j'}^{\mathbf{x}}$$

be the function from $\{0,1\}^n$ to $\{0,1\}^n$ obtained by restricting the i-th block of input of π, $i \neq j$, to $\mathbf{x}[i]$, an by considering only the j'-th block of input. Recall that

$$\mathsf{MaxPreim}(\pi) = \max_{\mathbf{x},j,h,y} |\{x \in \{0,1\}^n : \pi_{j,h}^{\mathbf{x}}(x) = y\}|.$$

Since the random value accessed by ReadTape during the defining call for $\mathbf{x}_i^D$ comes uniformly at random from a set of size at least $N - q - \alpha(q)$ by Lemma 20, the defining call to ReadTape for $\mathbf{x}_i^D$ has probability at most

$$\frac{\mathsf{MaxPreim}(\pi_G)|\mathcal{D}_j|}{N - q - \alpha(q)} \tag{5}$$

of causing $\mathsf{PreExisting}_{i,j}$ to occur for each $1 \leq j \leq w$, by a union bound over $\mathcal{D}_j$. Thus, given that $|\mathcal{D}_j| \leq q + \alpha(q)$,

$$\Pr\left[\bigvee_{1\leq i\leq k, 1\leq j\leq w} \mathsf{PreExisting}_{i,j}\right] \leq \frac{wk \cdot \mathsf{MaxPreim}(\pi_G)(q + \alpha(q))}{N - q - \alpha(q)}. \tag{6}$$

If the defining calls for vectors $\mathbf{x}_i^D$, $\mathbf{x}_{i'}^D$ are distinct calls, moreover, then

$$\Pr\left[\mathsf{Colliding}_{i,i',j}\right] \leq \frac{\mathsf{MaxPreim}(\pi_G)}{N - q - \alpha(q)}.$$

(The right-hand side is the same as (5), but with $|\mathcal{D}_j|$ replaced by 1, as there is only one bad value for the second of the two defining calls to land on.) If $\mathbf{x}_i^D$, $\mathbf{x}_{i'}^D$ share the same defining call, however, then

$$\Pr\left[\mathsf{Colliding}_{i,i',j}\right] \leq \frac{\mathsf{MaxColl}(\pi_G)}{N - q - \alpha(q)}$$

where we recall that

$$\mathsf{MaxColl}(\pi) = \max_{\mathbf{x}\neq\mathbf{x}', j, h} |\{x \in \{0,1\}^n : \pi_{j,h}^{\mathbf{x}}(x) = \pi_{j,h}^{\mathbf{x}'}(x)\}|.$$

Thus,

$$\Pr\left[\bigvee_{i\neq i', j} \mathsf{Colliding}_{i,i',j}\right] \leq \frac{wk^2 \cdot \mathsf{MaxCoPr}(\pi_G)}{N - q - \alpha(q)} \tag{7}$$

where $\mathsf{MaxCoPr}(\pi) = \max(\mathsf{MaxPreim}(\pi), \mathsf{MaxColl}(\pi))$. Thus, the probability that AdaptLeft aborts during this call because SetTable(D_j, x, y) is call with $D_j(x) \neq \perp$ is at most

$$\frac{wk \cdot \mathsf{MaxPreim}(\pi_G)(q + \alpha(q))}{N - q - \alpha(q)} + \frac{wk^2 \cdot \mathsf{MaxCoPr}(\pi_G)}{N - q - \alpha(q)} \tag{8}$$

and, by symmetry and by a union bound, the overall probability that AdaptLeft aborts (thus also including bad events of type $D_j^-(y) \neq \perp$) is upper bounded by

$$\frac{2wk \cdot \mathsf{MaxPreim}(\overline{\pi})(q + \alpha(q))}{N - q - \alpha(q)} + \frac{2wk^2 \cdot \mathsf{MaxCoPr}(\overline{\pi})}{N - q - \alpha(q)} \tag{9}$$

as per the definitions of $\mathsf{MaxPreim}(\overline{\pi})$, $\mathsf{MaxCoPr}(\overline{\pi})$ given in Section 3.

By symmetry, (9) holds equally well for a call to AdaptRight, provided ToBeAdapted$_E$ has k pairs when AdaptRight is called.

If the distinguisher's i-th query is a left-hand query let k_i be the size of ToBeAdapted$_D$ when AdaptLeft is called; if the distinguisher's i-th query is a right-hand query, let k_i be the size of ToBeAdapted$_E$ when AdaptRight is called; and if the distinguisher's i-th query is a neutral query let $k_i = 0$. Then $k_1, \ldots, k_q$ are random variables (functions of the execution) such that

$$k_1 + \cdots + k_q \leq \alpha(q) \tag{10}$$

with probability 1 by Lemma 17 and, by extension, such that

$$k_1^2 + \cdots + k_q^2 \leq \alpha^2(q) \tag{11}$$

with probability 1. Even while the k_i's are random variables, a little thought reveals that the total probability of aborting in AdaptLeft or AdaptRight is upper bounded by replacing respectively k and k^2 in (9) by the upper bounds $\alpha(q)$ and $\alpha^2(q)$ on these sums, cf. (10), (11). The total probability that G$_3$ aborts in AdaptLeft or AdaptRight is thus upper bounded by

$$\frac{2w\alpha(q)(q+\alpha(q)) \cdot \mathsf{MaxPreim}(\overline{\pi})}{N-q-\alpha(q)} + \frac{2w\alpha(q)^2 \cdot \mathsf{MaxCoPr}(\overline{\pi})}{N-q-\alpha(q)} \tag{12}$$

when $\mathsf{XtraUntglRnds}$ is off.

When $\mathsf{XtraUntglRnds}$ is on, we define an extra bad event:

– $\mathsf{PreHit}_{i,j}$ occurs if and only if $\mathbf{x}_i^I[j] \in \mathcal{I}_j$ $(1 \leq i \leq k,\ 1 \leq j \leq w)$

Similarly to (6), we have

$$\Pr\left[\bigvee_{1\leq i\leq k, 1\leq j\leq w} \mathsf{PreHit}_{i,j}\right] \leq \frac{wk \cdot \mathsf{MaxPreim}(\pi_G)(q+\alpha(q))}{N-q-\alpha(q)}. \tag{13}$$

If none of the events $\mathsf{PreHit}_{i,j}$ occur each $\mathbf{y}_i^I[j]$ is randomly chosen from a set of at least $N-q-\alpha(q)$ values (Lemma 20) during the second **forall** loop of AdaptLeft. We note this does not preclude the possibility that $\mathbf{y}_i^I[j] = \mathbf{y}_{i'}^I[j]$ for certain pairs $i \neq i'$ and certain values of j.

We proceed to upper bound the probability that some event $\mathsf{PreExisting}_{i,j}$ or $\mathsf{Colliding}_{i,i',j}$ occurs given that no event $\mathsf{PreHit}_{i,j}$ occurs. For the remainder of the argument the underlying randomness is only the randomness remaining in I. (We note, indeed, that this randomness isn't skewed by assuming none of the events $\mathsf{PreHit}_{i,j}$ occurs, since these events depend only the randomness in G.)

We can define the "defining call" of $\mathbf{x}_i^D$ similarly to the case $\mathsf{XtraUntglRnds} = \mathbf{false}$; the only difference is that we are now conditioning on G-values, and that the defining call is a call to I.

Once again, the random value accessed by ReadTape during the defining call for $\mathbf{x}_i^D$ comes uniformly at random from a set of size at least $N-q-\alpha(q)$ by Lemma 20. Thus, the defining call for $\mathbf{x}_i^D$ has probability at most

$$\frac{\mathsf{MaxPreim}(\pi_I)|\mathcal{D}_j|}{N-q-\alpha(q)} \tag{14}$$

of causing $\mathsf{PreExisting}_{i,j}$ to occur for each $1 \leq j \leq w$, by a union bound over $\mathcal{D}_j$. Thus, given that $|\mathcal{D}_j| \leq q+\alpha(q)$,

$$\Pr\left[\bigvee_{1\leq i\leq k, 1\leq j\leq w} \mathsf{PreExisting}_{i,j}\right] \leq \frac{wk \cdot \mathsf{MaxPreim}(\pi_I)(q+\alpha(q))}{N-q-\alpha(q)}. \tag{15}$$

If the defining calls for vectors $\mathbf{x}_i^D$, $\mathbf{x}_{i'}^D$ are distinct calls, moreover, then

$$\Pr\left[\mathsf{Colliding}_{i,i',j}\right] \leq \frac{\mathsf{MaxPreim}(\pi_I)}{N-q-\alpha(q)} \tag{16}$$

as before. Note that $\mathbf{x}_i^G \neq \mathbf{x}_{i'}^G$ implies $\mathbf{x}_i^I \neq \mathbf{x}_{i'}^I$, so there exists $1 \leq j' \leq w$ that $\mathbf{x}_i^I[j'] \neq \mathbf{x}_{i'}^I[j']$. In fact we can assume without loss of generality that ReadTape$(I_{j'}, \mathbf{x}_{i'}^I[j'], p_{I_{j'}})$ is the defining call of

$\mathbf{x}_i^D$. (This is because changing the order in which values are sampled does not change the sampled values; indeed, the underlying random tapes are the same.) Thus

$$\Pr\left[\bigvee_{i\neq i',j} \mathsf{Colliding}_{i,i',j}\right] \leq \frac{wk^2 \cdot \mathsf{MaxPreim}(\pi_I)}{N-q-\alpha(q)}. \tag{17}$$

because (16) holds for each i, i', j individually.

The probability that AdaptLeft aborts during this call because SetTable(D_j, x, y) is called with $D_j(x) \neq \perp$ is thus at most

$$\frac{wk \cdot \mathsf{MaxPreim}(\pi_G)(q+\alpha(q))}{N-q-\alpha(q)} + \frac{wk \cdot \mathsf{MaxPreim}(\pi_I)(q+\alpha(q))}{N-q-\alpha(q)} + \frac{wk^2 \cdot \mathsf{MaxPreim}(\pi_I)}{N-q-\alpha(q)}. \tag{18}$$

as obtained by summing (13), (15), (17). The overall probability that AdaptLeft aborts during this call is thus upper bounded by

$$\frac{(4wk(q+\alpha(q)) + 2wk^2) \cdot \mathsf{MaxPreim}(\overline{\pi})}{N-q-\alpha(q)} \tag{19}$$

by taking bad events of the type $D_j^-(y) \neq \perp$ into account. The same upper bound also holds for AdaptRight by symmetry.

Similarly to when $\mathsf{XtraUntglRnds}$ is off, one can then argue that the total probability that G_3 aborts in AdaptLeft or AdaptRight is upper bounded by

$$\frac{(4w\alpha(q)(q+\alpha(q)) + 2w\alpha(q)^2) \cdot \mathsf{MaxPreim}(\overline{\pi})}{N-q-\alpha(q)} \tag{20}$$

as obtained by replacing k with $\alpha(q)$ in (19). □

Lemma 39. *The probability that* G_3 *aborts for a q-query distinguisher is at most*

$$\begin{aligned}
&\frac{w(q+\alpha(q))^2}{N-q-\alpha(q)} + \frac{(q+w\alpha(q))(q+\alpha(q))}{N-q-\alpha(q)} + \frac{2w\alpha(q)(q+\alpha(q))}{N-q-\alpha(q)} \\
&+ \frac{4wq \cdot \mathsf{aboCond}_\tau(2q)}{N-2q} && \textit{if } \mathsf{XtraMiddleRnd} \textit{ is on} \\
&+ \frac{4w(q+\alpha(q))\big(\mathsf{aboCond}_\nu(q+\alpha(q))\big)}{N-q-\alpha(q)} && \textit{if } \mathsf{XtraOuterRnd} \textit{ is on} \\
&+ \frac{2w\alpha(q)(q+\alpha(q)) \cdot \mathsf{MaxPreim}(\overline{\pi})}{N-q-\alpha(q)} + \frac{2w\alpha(q)^2 \cdot \mathsf{MaxCoPr}(\overline{\pi})}{N-q-\alpha(q)} && \textit{if } \mathsf{XtraUntglRnds} \textit{ is off} \\
&+ \frac{(4w\alpha(q)(q+\alpha(q)) + 2w\alpha(q)^2) \cdot \mathsf{MaxPreim}(\overline{\pi})}{N-q-\alpha(q)} && \textit{if } \mathsf{XtraUntglRnds} \textit{ is on}
\end{aligned}$$

Proof. This follows by summing the bounds of Lemmas 24, 26, 28, 29, 30, 31, 33, 34 and 38. □

5.4 Proof of Indifferentiability

We say that a distinguisher D *completes all paths* if, for every query $\mathrm{Z}(\mathbf{x}^X) \leftarrow \mathbf{y}^H$ or query $\mathrm{Z}^{-1}(\mathbf{y}^H) \leftarrow \mathbf{x}^X$ made by D, D subsequently and eventually makes the (possibly redudant) *rwq* queries

$$\begin{array}{rl}
\mathrm{F}(1, \mathbf{x}^F[1]) \to \mathbf{y}^F[1], \ldots, \mathrm{F}(w, \mathbf{x}^F[w]) \to \mathbf{y}^F[w], & \text{if } \mathsf{XtraOuterRnd} \text{ is on} \\
\mathrm{G}(1, \mathbf{x}^G[1]) \to \mathbf{y}^G[1], \ldots, \mathrm{G}(w, \mathbf{x}^G[w]) \to \mathbf{y}^G[w], & \\
\mathrm{I}(1, \mathbf{x}^I[1]) \to \mathbf{y}^I[1], \ldots, \mathrm{I}(w, \mathbf{x}^I[w]) \to \mathbf{y}^I[w], & \text{if } \mathsf{XtraMiddleRnd} \text{ is on} \\
\vdots \qquad \ddots \qquad \vdots & \\
\mathrm{H}(1, \mathbf{x}^H[1]) \to \mathbf{y}'^H[1], \ldots, \mathrm{H}(w, \mathbf{x}^H[w]) \to \mathbf{y}'^H[w] &
\end{array}$$

unless the game aborts, where $\mathbf{x}^F = \mathbf{x}^X, \mathbf{x}^G = \nu(\mathbf{y}^F)$ if XtraOuterRnd is on, $\mathbf{x}^G = \mathbf{x}^X$ otherwise, etc. (Here we don't presume $\mathbf{y}'^H = \mathbf{y}^H$, though obviously D will be able to distinguish the simulated world from the real world if $\mathbf{y}'^H \neq \mathbf{y}^H$.)

For this section we presume a fixed q-query (information-theoretic) distinguisher D that completes all paths. The object is to upper bound

$$\Delta_D(\mathrm{G}_1, \mathrm{G}_5) = \Pr[D^{\mathrm{G}_1} = 1] - \Pr[D^{\mathrm{G}_5} = 1]. \tag{21}$$

The upper bounds we obtain for D will imply appropriately modified upper bounds for arbitrary distinguishers by noting that if D' is an arbitrary q_0-query distinguisher there exists a $(q_0 + rwq_0)$-query distinguisher D^* that completes all paths and that achieves advantage at least as good as D'.

Since D is information-theoretic we can assume without loss of generality that D is deterministic. Moreover since G_5 never aborts we can assume that if the environment aborts then D outputs 1, which can only serve to maximize (21).

Lemma 40. *Let D be as described above. Then $\Delta_D(\mathrm{G}_1, \mathrm{G}_2) \leq \beta(q)(q + \alpha(q))^w/(N^w - q - \alpha(q))$.*

Proof. The only difference between G_1 and G_2 is in the procedure CheckZ. Specifically, CheckZ will call Z in G_1 whereas CheckZ consults P_Z in G_2. Hence CheckZ is answered according to p_Z in G_1 and according to P_Z in G_2. Also, a call to CheckZ might modify P_Z in G_1 whereas a call to CheckZ never modifies P_Z in G_2. (We note, however, that P_Z always remains consistent with p_Z in either G_1 or G_2: P_Z is a subset of p_Z.)

Consider two concurrent executions of G_1 and G_2, executed on the same random tapes $p_{A_1}, \ldots,$ p_{L_w}, p_Z. We say the two executions *diverge* if CheckZ is ever called with arguments $(\mathbf{x}, \mathbf{y})$ such that $p_Z(\mathbf{x}) = \mathbf{y}$ and such that $P_Z(\mathbf{x}) = \perp$ in the G_2-execution. We claim that as long as the two executions don't diverge in this sense, every function call is answered the same in each execution. Indeed, calls to CheckZ are answered the same (by the definition of divergence) and calls of the form $\mathrm{Z}(\mathbf{x})$, $\mathrm{Z}^{-1}(\mathbf{y})$ are also answered the same, being answered by $p_Z(\mathbf{x})$ and $p_Z^{-1}(\mathbf{y})$ (respectively) in either game regardless of the state of P_Z. Furthermore, while P_Z may contain more defined entries in G_1 than in G_2, CheckZ, Zand Z^{-1} are the only functions to use P_Z in either game, so the differences in P_Z don't propagate to elsewhere, as we have just seen that CheckZ, Z and Z^{-1} answer function calls the same.

Next, to upper bound the probability of divergence, we start by observing that divergence is well-defined for an execution of G_2 on its own. Indeed we can say that an execution of G_2 "diverges" if CheckZ is ever called with a pair $(\mathbf{x}, \mathbf{y})$ such that $p_Z(\mathbf{x}) = \mathbf{y}$ and $P_Z(\mathbf{x}) = \perp$; then G_2 "diverges" on a given set of random tapes if and only if the concurrent executions of G_1 and G_2 "diverge" in the sense above on the same set of random tapes. We can therefore focus on the probability that an execution of G_2 diverges, without reference to G_1.

To upper bound the probability of divergence in G_2 note that if CheckZ is called on arguments $(\mathbf{x}, \mathbf{y})$ such that $P_Z(\mathbf{x}) = \perp$ then $p_Z(\mathbf{x})$ hasn't been read yet. Indeed $p_Z(\mathbf{x})$ is only read in Z and Z^{-1}, at which point P_Z is updated with the read value. Hence, the value $p_Z(\mathbf{x})$ comes uniformly at random from a set of size at least $N^w - q - \alpha(q)$ by Lemma 25. (A subtlety here is that we are *not* assuming that divergence hasn't already occurred. Indeed such an assumption would skew the conditional distribution of the *unread* values in p_Z, as the truth of this assumption depends on unread values. Instead we are simply doing a union bound over all calls to CheckZ.) As CheckZ is called with at most $\beta(q)(q+\alpha(q))^w$ distinct arguments pairs during a G_2-execution by Lemma 23, the probability of divergence in G_2 (and hence between G_1 and G_2) is at most $\beta(q)(q+\alpha(q))^w/(N^w - q - \alpha(q))$. □

Lemma 41. *Let D be as described above. Then $\Delta_D(\mathrm{G}_2, \mathrm{G}_5) \leq \Delta_D(\mathrm{G}_3, \mathrm{G}_5)$.*

Proof. Since G_5 never aborts we can assume that D outputs 1 when the environment aborts. However the only difference between G_2 and G_3 is that more abort conditions are added in G_3 (specifically, two executions with the same random tapes will unfold exactly the same in G_2 and G_3 except for the eventuality that G_3 might abort while G_2 does not) so D's probability of outputting 1 is at least as large as D's probability of outputting 1 in G_1, which proves the claim. □

To upper bound $\Delta_D(\mathrm{G}_3, \mathrm{G}_5)$ we upper bound $\Delta_D(\mathrm{G}_3, \mathrm{G}_4)$ and $\Delta_D(\mathrm{G}_4, \mathrm{G}_5)$ separately and apply the triangle inequality. The crucial transition is the transition from G_3 to G_4, for which we apply a randomness mapping argument. This requires some extra definitions.

Given an execution of G_3 the *footprint* of that execution is that subset of the random tapes actually read during that execution. More precisely, identify a permutation $p : \{0,1\}^n \to \{0,1\}^n$ with the set of pairs $(x, y) \in \{0,1\}^n \times \{0,1\}^n$ such that $p(x) = y$. A *partial permutation* is obtained by dropping pairs from p. Then the *footprint* of an execution of G_3 (with our fixed distinguisher D) on random tapes p_Z, $p_{A_1}, \ldots, p_{L_w}$ is the set of *partial* permutations $\tilde{p}_Z, \tilde{p}_{A_1}, \ldots, \tilde{p}_{L_w}$ where

$$\tilde{p}_Z \subseteq p_Z, \ \tilde{p}_{A_1} \subseteq p_{A_1}, \ \ldots, \ \tilde{p}_{L_w} \subseteq p_{L_w}$$

and where $\tilde{p}_Z$ and $\tilde{p}_{T_j}$ contain only those entries of respectively p_Z and p_{T_j} that were actually read from (respectively again) p_Z and p_{T_j} during the execution. (Here an entry (x, y) is "read" from p if and only if either $p(x)$ or $p^{-1}(y)$ was accessed during the execution. In G_3 such reads occur only in the function ReadTape.)

Note that since D is deterministic, a footprint contains all the information necessary to re-create a particular execution (and in particular D's output). In particular if we run D on a footprint $\tilde{p}_Z, \tilde{p}_{A_1}, \ldots, \tilde{p}_{L_w}$ then (i) the execution will never "leave" that footprint (i.e., request values from the random tables that are outside the footprint), and (ii) all values included in the footprint will be read at some point during the execution. Also note that two distinct footprints are always incompatible in the sense that they don't have a common extension $p_Z, p_{A_1}, \ldots, p_{L_w}$. This follows by points (i) and (ii) just mentioned. We also emphasize that the set of possible footprints is a function of D.

We say that a footprint is *accepting* for G_3 if D outputs 1 for the execution corresponding to that footprint. Moreover we say that a footprint is *non-aborting* for G_3 if the corresponding G_3 execution doesn't abort.

Letting $|\tilde{p}|$ denote the number of input-output pairs in the partial permutation $\tilde{p}$, the probability of obtaining a given footprint $\tilde{p}_Z, \tilde{p}_{A_1}, \ldots, \tilde{p}_{L_w}$, taken over the random choice of $p_Z, p_{A_1}, \ldots, p_{L_w}$, is obviously

$$\left(\prod_{\ell=0}^{|\tilde{p}_Z|-1} \frac{1}{N^w - \ell} \right) \left(\prod_{T} \prod_{j=1}^{w} \prod_{\ell=0}^{|\tilde{p}_{T_j}|-1} \frac{1}{N - \ell} \right) \tag{22}$$

since this is the probability that a randomly chosen $p_Z, p_{A_1}, \ldots, p_{L_w}$ is an extension of $\tilde{p}_Z, \tilde{p}_{A_1}, \ldots,$ $\tilde{p}_{L_w}$. (Of course, this presumes that $\tilde{p}_Z, \tilde{p}_{A_1}, \ldots, \tilde{p}_{L_w}$ is a real footprint and not just some set of arbitrarily restricted permutations.) In particular, for instance, the probability that D accepts is the sum of (22) taken over all accepting footprints.

We likewise define the footprint of an execution in G_4 with respect to the random tapes $q_{A_1}, \ldots, q_{L_w}$ as the set of entries read from those tapes. Thus a footprint in G_4 is again a set of partially defined permutations. The probability of obtaining a given footprint $\tilde{q}_{A_1}, \ldots, \tilde{q}_{L_w}$ in G_4 is

$$\prod_T \prod_{j=1}^{w} \prod_{\ell=0}^{|\tilde{q}_{T_j}|-1} \frac{1}{N-\ell}. \tag{23}$$

since this is the probability that randomly chosen tapes $q_{A_1}, \ldots, q_{L_w}$ are an extension of $\tilde{q}_{A_1}, \ldots, \tilde{q}_{L_w}$. We likewise talk of *accepting* and *non-aborting* footprints in G_4.

For $i = 3, 4$ let FP_i denote the set of all G_i footprints and $\mathsf{FP}_i^\star \subseteq \mathsf{FP}_i$ the set of all non-aborting G_i footprints. We write

$$\Pr_{\mathrm{G}_i}[\omega]$$

for the probability of obtaining a given footprint $\omega \in \mathsf{FP}_i$, $i = 3, 4$ (cf. (22), (23)), and

$$\Pr_{\mathrm{G}_i}[\mathcal{A}] = \sum_{\omega \in \mathcal{A}} \Pr_{\mathrm{G}_i}[\omega]$$

for any $\mathcal{A} \subseteq \mathsf{FP}_i$. Note that $\Pr_{\mathrm{G}_i}[\mathsf{FP}_i \backslash \mathsf{FP}_i^\star]$ is the probability of abortion in game G_i for $i = 3, 4$ since $\mathsf{FP}_i^\star$ is the set of non-aborting footprints.

We also let $\mathcal{A}_i \subseteq \mathsf{FP}_i$ be the set of accepting footprints in game G_i and let $\mathcal{A}_i^\star = \mathcal{A}_i \cap \mathsf{FP}_i^\star$ be the set of accepting footprints that are also non-aborting. Note that $\mathsf{FP}_i \backslash \mathsf{FP}_i^\star \subseteq \mathcal{A}_i$ by our assumption that D accepts whenever the game aborts and, by the same token, $\mathsf{FP}_i \backslash \mathsf{FP}_i^\star = \mathcal{A}_i \backslash \mathcal{A}_i^\star$. (In other words, the set of non-good accepting footprints is also the set of all non-good footprints.)

The key to the randomness mapping argument is an injection $\zeta : \mathsf{FP}_3^\star \leftarrow \mathsf{FP}_4^\star$ such that $\omega \in \mathsf{FP}_3^\star$ is accepting if and only if $\zeta(\omega) \in \mathsf{FP}_4^\star$ is accepting and such that

$$\Pr_{\mathrm{G}_4}[\zeta(\omega)] \geq \Pr_{\mathrm{G}_3}[\omega]\,(1 - 1/N^w) \tag{24}$$

for all $\omega = (\tilde{p}_Z, \tilde{p}_{A_1}, \ldots, \tilde{p}_{L_w}) \in \mathsf{FP}_3^\star$. Then

$$\begin{aligned}
\Delta_D(\mathrm{G}_3, \mathrm{G}_4) &= \Pr[D^{\mathrm{G}_3} \leftarrow 1] - \Pr[D^{\mathrm{G}_4} \leftarrow 1] \\
&= \Pr_{\mathrm{G}_3}[\mathcal{A}_3] - \Pr_{\mathrm{G}_4}[\mathcal{A}_4], \\
&= \Pr_{\mathrm{G}_3}[\mathcal{A}_3 \backslash \mathcal{A}_3^\star] + \Pr_{\mathrm{G}_3}[\mathcal{A}_3^\star] - \Pr_{\mathrm{G}_4}[\mathcal{A}_4 \backslash \mathcal{A}_4^\star] - \Pr_{\mathrm{G}_4}[\mathcal{A}_4^\star] \\
&= \Pr_{\mathrm{G}_3}[\mathsf{FP}_3 \backslash \mathsf{FP}_3^\star] + \Pr_{\mathrm{G}_3}[\mathcal{A}_3^\star] - \Pr_{\mathrm{G}_4}[\mathsf{FP}_4 \backslash \mathsf{FP}_4^\star] - \Pr_{\mathrm{G}_4}[\mathcal{A}_4^\star] \\
&\leq \Pr_{\mathrm{G}_3}[\mathsf{FP}_3 \backslash \mathsf{FP}_3^\star] + \frac{1}{N^w} - \Pr_{\mathrm{G}_4}[\mathsf{FP}_4 \backslash \mathsf{FP}_4^\star]
\end{aligned} \tag{25}$$

where the inequality holds by (24) (recall that (24) holds for all $\omega \in \mathcal{A}_3^\star$ and that $\zeta : \mathcal{A}_3^\star \leftarrow \mathcal{A}_4^\star$ is injective—see Lemma 47 below for more details).

The definition of $\zeta : \mathsf{FP}_3^\star \rightarrow \mathsf{FP}_4^\star$ is fairly simple: for $\omega = (\tilde{p}_Z, \tilde{p}_{A_1}, \ldots, \tilde{p}_{L_w}) \in \mathsf{FP}_3^\star$ we define $\zeta(\omega) = (\tilde{q}_{A_1}, \ldots, \tilde{q}_{L_w})$ by

$$\tilde{q}_{T_j} = \{(x, y) \in \{0,1\}^n \times \{0,1\}^n : T_j(x) = y\}$$

where T_j refers to the table T_j as it stands at the end of the G_3-execution. (In a nutshell, thus, $\tilde{q}_{T_j} = T_j$.)

The map ζ is obviously well-defined because ω is a footprint. But, to recapitulate, the following properties of ζ must still be proved:

- that $\zeta(\omega) \in \mathsf{FP}_4^\star$ for all $\omega \in \mathsf{FP}_3^\star$ (i.e., that the image of a good footprint is a good footprint)
- that ζ is injective
- that $\zeta(\omega)$ is accepting in G_4 if and only if ω is accepting for G_3
- that $\Pr_{\mathrm{G}_4}[\zeta(\omega)] \geq \Pr_{\mathrm{G}_3}[\omega](1 - 1/N^w)$ for all $\omega \in \mathsf{FP}_3^\star$

We build up to these four points in a series of lemmas. We recall the terminology of *T-columns*, *matching columns* and *completed paths* from Section 4. We note that Lemma 1 implies that both completed paths and T-columns are "persistent": once a column or completed path appears, it remains unchanged until the end of the execution. Moreover, because the elements of $\overline{\pi}$ are permutations, each column is part of at most one completed path, and matching columns are in the same (if any) completed path.

Lemma 42. *At the end of each non-aborted query cycle in* G_3 *(and, in particular, and the end of each non-aborted* G_3*-execution) every pair of matching B- and C-columns is part of a completed path if* $\mathsf{XtraMiddleRnd}$ *is on, every A-column is part of a completed path if* $\mathsf{XtraMiddleRnd}$ *is off.*

Proof. This is a corollary of Lemma 16. □

Lemma 43. *At the end of each non-aborted execution of* G_3 *there exists a completed path with endpoints* $(\mathbf{x}, \mathbf{y})$ *for each pair* $(\mathbf{x}, \mathbf{y})$ *such that either of the calls* $\mathrm{Z}(\mathbf{x}) \leftarrow \mathbf{y}$ *or* $\mathrm{Z}^{-1}(\mathbf{y}) \leftarrow \mathbf{x}$ *has occurred at some point in the execution.*

Proof. For calls to Z, Z^{-1} made by the simulator this follows from the fact that the simulator only calls Z or Z^{-1} when it has already decided to complete a path in G_3 (cf. RequestG, $\mathrm{RequestU}^{-1}$ and AdaptLeft, AdaptRight). For calls to Z, Z^{-1} made by the distinguisher this follows from the assumption that the distinguisher completes all paths and from Lemma 42 (since the "path" built by the distinguisher will in particular go through the middle detect zone and hence, by Lemma 42, have endpoints that are compatible with Z). □

Lemma 44. *Let the map ζ be defined on $\mathsf{FP}_3^\star$ as above. Then $\zeta(\omega) \in \mathsf{FP}_4^\star$ for each footprint $\omega \in \mathsf{FP}_3^\star$ and $\zeta(\omega)$ is accepting for* G_4 *if and only if ω is accepting for* G_3*. Finally, ζ is injective.*

Proof. Let $\omega = (\tilde{p}_Z, \tilde{p}_{A_1}, \ldots, \tilde{p}_{L_w}) \in \mathsf{FP}_3^\star$ and let $\zeta(\omega) = (\tilde{q}_{A_1}, \ldots, \tilde{q}_{L_w})$.

For syntactical convenience, let $\lambda = (q_{A_1}, \ldots, q_{L_w})$ be some extension of $\zeta(\omega)$; we will first prove that executing G_4 on random tape λ gives a non-aborting execution, and then argue that this execution's footprint is in fact $\zeta(\omega)$.

Consider concurrent executions of G_3 and G_4 with random tapes ω and λ respectively. (Note that ω is a partial random tape for G_3 whereas λ is a complete random tape G_4, but since ω is a G_3-footprint by assumption this poses no problem.)

Note that only two differences occur between G_3 and G_4: (i) the internals of the functions Z, Z^{-1}; (ii) the fact that G_4 reads from the random tapes $q_{A_1}, \ldots, q_{L_w}$, whereas G_3 reads from the tapes $p_Z, p_{A_1}, \ldots, p_{F_w}$. Moreover, while calls to $\mathrm{Z}/\mathrm{Z}^{-1}$ affect the table P_Z—and only this table—P_Z is not actually used anywhere outside of Z, Z^{-1}. Consequently we say that the two executions *diverge* if (a) there either occurs a call Z to Z^{-1} that is not answered the same in either game (where the call could be made by the simulator or by the distinguisher either) or if (b) a call to $\mathrm{ReadTape}(T_j, x, p_{T_j})$ or $\mathrm{ReadTape}(T_j^{-1}, y, p_{T_j}^{-1})$ occurs in G_3 such that, respectively, $\tilde{p}_{T_j}(x) \neq q_{T_j}(x)$

or $\tilde{p}_{T_j}^{-1}(y) \neq q_{T_j}^{-1}(y)$. Thus if divergence doesn't occur in this technical sense, the two executions proceed identically up to the internals of Z, Z^{-1}.

Firstly divergence of type (b) cannot occur because every time a value is read from p_{T_j} in G_3 this value ends up in T_j and hence, by definition of ζ, in $\tilde{q}_{T_j}$ (which is a subset of q_{T_j}). Secondly divergence of type (a) cannot occur by Lemma 43. Indeed this lemma implies that for each call $\mathrm{Z}(\mathbf{x}) \to \mathbf{y}$ or $\mathrm{Z}^{-1}(\mathbf{y}) \to \mathbf{x}$ occurring in G_3 there eventually exists a completed path with endpoints $(\mathbf{x}, \mathbf{y})$ in the tables T_j; by definition of ζ the relevant entries of T_j appear in $\tilde{q}_{T_j}$.

Since the two executions don't diverge it follows that G_4 does not abort on λ (as G_3 doesn't abort on ω), and that G_4 accepts if and only if G_3 accepts.

We next show that $\zeta(\omega)$ is a G_4-footprint. To see that G_4 never reads values *outside* of $\zeta(\omega)$ note that all values read by G_4 in the tables q_{T_j} were either: (i) simultaneously read by G_3 in p_{T_j} or (ii) read as part of a call to Z or Z^{-1}; by Lemma 42 it follows that the correponding entry was in T_j at the end of the G_3-execution. Hence, G_4 never reads outside of $\zeta(\omega)$ when executed on a random tape λ. Then to see that G_4 reads *every* value in $\zeta(\omega)$ note that each entry entered into a T_j table in G_3 is either entered by a call to ReadTape (in which case the same entry is read from $\zeta(\omega)$ in G_4, as the two executions don't diverge) or else, in the case of D_j and E_j, placed there while completing a path. But each completed path is completed only after a corresponding call to $\mathrm{Z}/\mathrm{Z}^{-1}$, and the matching entry from q_{D_j} or q_{E_j} will be read during the corresponding call to $\mathrm{Z}/\mathrm{Z}^{-1}$ that occurs in G_4.

Finally we show that ζ is injective. For this we build an inverse map. Given $\zeta(\omega) \in \mathsf{FP}_4^\star$, we define a footprint ω' for G_3 like so: we run G_3 with "blank" random tapes concurrently to G_4 with random tape $\zeta(\omega)$; when a call $\mathrm{ReadTape}(T_j^{\pm}, \cdot, \cdot)$ occurs we set the corresponding entry of p_{T_j} according to the same entry in $\tilde{q}_{T_j}$, and when a call to Z or Z^{-1} occurs such that P_Z is undefined at that point, we set the corresponding entry of p_Z according to the answer of that call in G_4. Since the execution of G_4 is "the same" as when G_4 and G_3 were executed concurrently with random tapes λ and ω respectively, it's easy to see that the partial random tape obtained for G_3 by this inverse map must be compatible[19] with ω and hence must be ω (since two distinct footprints are never compatible), and hence that ζ is injective. □

Lemma 45. *Let* $\omega = (\tilde{p}_Z, \tilde{p}_{A_1}, \ldots, \tilde{p}_{L_w}) \in \mathsf{FP}_3^\star$ *and let* $\zeta(\omega) = (\tilde{q}_{A_1}, \ldots, \tilde{q}_{L_w}) \in \mathsf{FP}_4^\star$*. Then* $|\tilde{q}_{T_j}| = |\tilde{p}_{T_j}|$ *for* $T \notin \{D, E\}$ *and there exist nonnegative integers* a, b *such that* $a + b = |\tilde{p}_Z|$ *and such that* $|\tilde{q}_{D_j}| = |\tilde{p}_{D_j}| + a$, $|\tilde{q}_{E_j}| = |\tilde{p}_{E_j}| + b$ *for* $1 \leq j \leq w$.

Proof. Recall that $|\tilde{q}_{T_j}| = |T_j|$ where $|T_j|$ is the number of entries in T_j at the end of the G_3-execution on tape (or footprint) ω. Thus, the fact that $|\tilde{q}_{T_j}| = |\tilde{p}_{T_j}|$ for $T \notin \{D, E\}$ follows from the fact that $\mathrm{SetTable}(T_j, \cdot, \cdot)$ is never called for $T \notin \{D, E\}$ and, hence, that the number of entries in T_j is the same as the number of calls to $\mathrm{ReadTape}(T_j^{\pm 1}, \cdot, \cdot)$ for $T \notin \{D, E\}$.

To prove the second part of the claim let a be the number of times that a pair $(\mathbf{x}^G, \mathbf{y}^U)$ is added to $\mathrm{ToBeAdapted}_D$ in $\mathrm{RequestU}^{-1}$ and let b be the number of times that a pair $(\mathbf{x}^V, \mathbf{y}^H)$ is added to $\mathrm{ToBeAdapted}_E$ in RequestV, in the G_3-execution on random tape ω. Note there is a bijection between the final set of completed paths and the entries in p_Z (in more detail, each A-column or each matching pair of B- and C-columns causes the simulator to complete a path in which it calls Z or Z^{-1}, and each call to Z or Z^{-1} has an associated completed path (containing a unique A-column or a unique matching pair of B- and C-columns) because the distinguisher completes all paths). Moreover, as noted in the proof of Lemma 42, there is a bijection between all A-columns (if $\mathsf{XtraMiddleRnd}$ is off) or all matching pairs of B- and C-columns (if $\mathsf{XtraMiddleRnd}$ is on) and

[19] Two partial random random tapes are "compatible" if they have a common extension.

the set of all pairs $(\mathbf{x}^G, \mathbf{y}^U)$ or $(\mathbf{x}^V, \mathbf{y}^H)$ added respectively to ToBeAdapted_D and ToBeAdapted_E; hence $a+b = |p_Z|$. But a is also the number of times $\text{SetTable}(D_j, \cdot, \cdot)$ is called for each $1 \le j \le w$, while b is the number of times $\text{SetTable}(E_j, \cdot, \cdot)$ is called for each $1 \le j \le w$. Since SetTable and ReadTape both abort rather than overwrite an entry, this completes the proof. □

Lemma 46. *We have* $\Pr_{G_4}[\omega] \ge \Pr_{G_3}[\zeta(\omega)](1 - 1/N^w)$ *for every* $\omega \in \mathsf{FP}_3^\star$.

Proof. Let a and b be as in Lemma 45. Then by (22), (23), and keeping other notations of Lemma 45,

$$\begin{aligned}
\Pr_{G_4}[\zeta(\omega)] / \Pr_{G_3}[\omega] &= \prod_{j=0}^{w} \left(\prod_{h=0}^{a-1} \frac{1}{N - |\tilde{p}_{D_j}| - h} \cdot \prod_{h=0}^{b-1} \frac{1}{N - |\tilde{p}_{E_j}| - h} \right) \Big/ \prod_{\ell=0}^{a+b-1} \frac{1}{N^w - \ell} \\
&\ge \prod_{j=0}^{w} \left(\prod_{h=0}^{a-1} \frac{1}{N - h} \cdot \prod_{h=0}^{b-1} \frac{1}{N - h} \right) \Big/ \prod_{\ell=0}^{a+b-1} \frac{1}{N^w - \ell} \\
&= \prod_{h=0}^{a-1} \frac{N^w}{(N-h)^w} \cdot \prod_{h=0}^{b-1} \frac{N^w}{(N-h)^w} \cdot \prod_{\ell=0}^{a+b-1} \frac{N^w - \ell}{N^w} \qquad (26)
\end{aligned}$$

As $a + b = |p_Z|$, we have

$$a + b \le q + \alpha(q) \le N^{w-1}$$

by Lemma 25, where the second inequality holds without loss of generality. (Indeed, the security bound of Theorem 1 is void otherwise.) Thus

$$(N-1)^w = N^w(1 - 1/N)^w \le N^w(1 - 1/N) = N^w - N^{w-1} \le N - a - b$$

and, more generally,

$$(N-1)^w \le N - \ell \qquad (27)$$

for all $0 \le \ell \le a + b$. In particular, we can assume $a, b > 0$ since if $a = 0$ or $b = 0$ we have that (26) is lower bounded by 1, by (27). Thus, continuing from (26),

$$\begin{aligned}
\Pr_{G_4}[\zeta(\omega)] / \Pr_{G_3}[\omega] &\ge \frac{N^w - 1}{N^w} \cdot \prod_{h=1}^{a-1} \frac{N^w}{(N-h)^w} \cdot \prod_{h=1}^{b-1} \frac{N^w}{(N-h)^w} \cdot \prod_{\ell=2}^{a+b-1} \frac{N^w - \ell}{N^w} \\
&\ge \frac{N^w - 1}{N^w} \cdot \prod_{h=1}^{a-1} \frac{N^w}{(N-h)^w} \cdot \prod_{h=1}^{b-1} \frac{N^w}{(N-h)^w} \cdot \prod_{\ell=2}^{a+b-1} \frac{(N-1)^w}{N^w} \\
&= \frac{N^w - 1}{N^w} \prod_{h=1}^{a-1} \frac{(N-1)^w}{(N-h)^w} \cdot \prod_{h=1}^{b-1} \frac{(N-1)^w}{(N-h)^w} \\
&\ge \frac{N^w - 1}{N^w} = 1 - 1/N^w \qquad (28)
\end{aligned}$$

as claimed. □

Lemma 47. *We have*

$$\Pr_{G_3}[\mathcal{A}_3^\star] - \Pr_{G_4}[\mathcal{A}_4^\star] \le \frac{1}{N^w}$$

where $\mathcal{A}_i^\star \subseteq \mathsf{FP}_i^\star$ *denotes the set of accepting, non-aborting footprints in* G_i.

Proof. This follows directly by Lemma 46 since ζ maps elements of $\mathcal{A}_3^\star$ to elements of $\mathcal{A}_4^\star$, and is injective (Lemma 44). More precisely, the fact that

$$\Pr_{G_4}[\mathcal{A}_4^\star] \geq \Pr_{G_3}[\mathcal{A}_3^\star]\left(1 - \frac{1}{N^w}\right)$$

implies

$$\Pr_{G_3}[\mathcal{A}_3^\star] - \Pr_{G_4}[\mathcal{A}_4^\star] \leq \frac{1}{N^w}\Pr_{G_3}[\mathcal{A}_3^\star] \leq \frac{1}{N^w}.$$

□

Lemma 48. *We have*

$$\varDelta_D(G_3, G_4) \leq \Pr_{G_3}[\mathsf{FP}_3 \backslash \mathsf{FP}_3^\star] + \frac{1}{N^w} - \Pr_{G_4}[\mathsf{FP}_4 \backslash \mathsf{FP}_4^\star]$$

for any q-query distinguisher D that completes all paths.

Proof. This was shown in (25), above, where we used the fact that $\Pr_{G_3}[\mathcal{A}_3^\star] - \Pr_{G_4}[\mathcal{A}_4^\star] \leq \frac{1}{N^w}$, as shown in Lemma 47. □

Lemma 49. *We have*

$$\varDelta_D(G_4, G_5) \leq \Pr_{G_4}[\mathsf{FP}_4 \backslash \mathsf{FP}_4^\star]$$

for any q-query distinguisher D that completes all paths.

Proof. Consider a non-aborting execution of G_4. In such an execution queries to Z/Z^{-1} are answered entirely according to the random tapes $q_{A_1}, \ldots, q_{L_w}$, as in G_5. Also, since entries in T_j for $T \notin \{D, E\}$ are never adapted, queries to $T^\pm$ are answered according to the random tapes q_{T_j}. Lastly we argue that queries to $D^\pm(\cdot, j)$ and $E^\pm(\cdot, j)$ are answered according to the tables q_{D_j} and q_{E_j}. This is equivalent to showing that entries written in D_j and E_j always agree with q_{D_j} and q_{E_j}, respectively. We argue the latter by induction (on the number of entries added to D_j and E_j). For entries in D_j, E_j that are directly read from the random tapes the claim is obvious. For an entry in (say) D_j that is adapted, the claim follows from the fact that every adapted value in D_j is part of a completed path, from the induction hypothesis, from the fact that queries to Z/Z^{-1} are answered according to the random tapes q_{T_j}, and from the fact that when the D_j query is adapted all queries in columns $T \neq D$ from the completed path are already made.

It follows that queries in both G_4 and G_5 are answered according the tables q_{T_j}, with the only difference that G_4 might abort. So $\varDelta_D(G_4, G_5) \leq \Pr[\mathsf{FP}_4 \backslash \mathsf{FP}_4^\star]$. □

(One can, in fact, argue that $\varDelta_D(G_4, G_5) = \Pr[\mathsf{FP}_4 \backslash \mathsf{FP}_4^\star]$ from the fact that D outputs 1 if abort occurs. But we only need the inequality.)

Lemma 50. *We have*

$$\begin{aligned}
&\Delta_D(\mathrm{G}_1,\mathrm{G}_5)\\
&\le \frac{\beta(q)(q+\alpha(q))^w}{N^w-q-\alpha(q)}+\frac{1}{N^w}\\
&\quad+\frac{w(q+\alpha(q))^2}{N-q-\alpha(q)}+\frac{(q+w\alpha(q))(q+\alpha(q))}{N-q-\alpha(q)}+\frac{2w\alpha(q)(q+\alpha(q))}{N-q-\alpha(q)}\\
&\quad+\frac{4wq\cdot\mathsf{aboCond}_\tau(2q)}{N-2q} && \textit{if } \mathsf{XtraMiddleRnd} \textit{ is on}\\
&\quad+\frac{2w(q+\alpha(q))\big(\mathsf{aboCond}_\nu(q+\alpha(q))\big)}{N-q-\alpha(q)} && \textit{if } \mathsf{XtraOuterRnd} \textit{ is on}\\
&\quad+\frac{2w\alpha(q)(q+\alpha(q))\cdot\mathsf{MaxPreim}(\overline{\pi})}{N-q-\alpha(q)}+\frac{2w\alpha(q)^2\cdot\mathsf{MaxCoPr}(\overline{\pi})}{N-q-\alpha(q)} && \textit{if } \mathsf{XtraUntglRnds} \textit{ is off}\\
&\quad+\frac{(4w\alpha(q)(q+\alpha(q))+2w\alpha(q)^2)\cdot\mathsf{MaxPreim}(\overline{\pi})}{N-q-\alpha(q)} && \textit{if } \mathsf{XtraUntglRnds} \textit{ is on}
\end{aligned}$$

for any q-query distinguisher D that complete all paths.

Proof. We have, using lemmas 40, 41, 48 and 49,

$$\begin{aligned}
\Delta_D(\mathrm{G}_1,\mathrm{G}_5) &= \Delta_D(\mathrm{G}_1,\mathrm{G}_2)+\Delta_D(\mathrm{G}_2,\mathrm{G}_5)\\
&\le \Delta_D(\mathrm{G}_1,\mathrm{G}_2)+\Delta_D(\mathrm{G}_3,\mathrm{G}_5)\\
&= \Delta_D(\mathrm{G}_1,\mathrm{G}_2)+\Delta_D(\mathrm{G}_3,\mathrm{G}_4)+\Delta_D(\mathrm{G}_4,\mathrm{G}_5)\\
&\le \frac{\beta(q)(q+\alpha(q))^w}{N^w-q-\alpha(q)}+\left(\Pr_{\mathrm{G}_3}[\mathsf{FP}_3\backslash\mathsf{FP}_3^\star]+\frac{1}{N^w}-\Pr_{\mathrm{G}_4}[\mathsf{FP}_4\backslash\mathsf{FP}_4^\star]\right)+\Pr_{\mathrm{G}_4}[\mathsf{FP}_4\backslash\mathsf{FP}_4^\star]\\
&= \frac{\beta(q)(q+\alpha(q))^w}{N^w-q-\alpha(q)}+\Pr_{\mathrm{G}_3}[\mathsf{FP}_3\backslash\mathsf{FP}_3^\star]+\frac{1}{N^w}.
\end{aligned}$$

The bound then follows by Lemma 39, since $\Pr_{\mathrm{G}_3}[\mathsf{FP}_3\backslash\mathsf{FP}_3^\star]$ is the probability of abortion in G_3. □

5.5 Leftovers

The security bound in Theorem 1 is easily seen to be the bound of Lemma 50 modulo some simplifications to make the bound more compact. In order to prove Theorem 1 it remains to argue the simulator's query complexity and running time.

Lemma 51. *The simulator described in game G_1 of Figs. 2–6 can be implemented in running time $O(w(q+\alpha(q))^w)$.*

Proof. Once glaring inefficiencies are removed from the simulator (many of which are caused by our efforts to avert nested loops), one can see that the simulator's running time is dominated by steps in which it "mix-and-matches" all queries within a given column, either to check for abort conditions or (more crucially) for path completion triggers.

A little more precisely, such "mix-and-matching" can be arranged to always take place such that at least one query of the column is fixed (being in fact the last query created for that column); this means that each mix-and-match enumeration takes time $O((q+\alpha(q))^{w-1})$ in the worst case by Lemma 20. Moreover there will be at most $5w(q+\alpha(q))$ mix-and-match enumerations over the whole course of the experiment, given that at most 5 columns (F, G, B, C and H) trigger mix-and-match enumerations. This leads to the stated bound. □

Lemma 52. *The simulator described in game* G_1 *of Figs. 2–6 can be implemented to have query complexity at most* $\beta(q)$.

Proof. Note the simulator in game G_1 of Figs. 2–6 queries $\mathrm{Z}/\mathrm{Z}^{-1}$ without checking to see if it possibly knows the answers to these queries from its previous queries to $\mathrm{Z}/\mathrm{Z}^{-1}$. For the following argument, we imagine that the simulator maintains a global table of its queries to Z, together with their inverses, such as to never make a redundant query to Z. (This obviously does not change anything from the distinguisher's perspective, nor for the simulator, except for the change itself.)

Each query to Z or Z^{-1} either occurs in CheckZ or else because the simulator is completing a middle-triggered path. In either case, the query can thus uniquely be associated with a set of matching queries in the left-hand outer detect zone. But there are at most $\beta(q)$ such matching query sets by Lemma 20. □

Further remarks. Space complexity of the simulator. The table mentioned in the proof of Lemma 52 is in fact our simulator's dominating space requirement; the space complexity of our simulator is thus

$$O(\beta(q))$$

if we assume that some efficient data structure is used to implement the table. (We omit a factor of nw for simplicity.)

If we remove the table, the space complexity drops down to $O(q)$ (factor of nwr omittted), i.e., the space needed to store the S-box permutation tables. On the other hand one can check that the query complexity climbs to

$$2(\beta(q) + (2q)^w) \tag{29}$$

because one can establish a 2-to-1 surjection from the set of queries to the union of (i) matching query sets in the left-hand detect zone and (ii) matching query sets in the right-hand detect zone. (The factor of 2 occurs because the same query can occur once in RequestF and once in $\mathrm{RequestU}^{-1}$, during a left-hand query cycle, or once in $\mathrm{RequestG}^{-1}$ and once in RequestV, during a right-hand query cycle.)

However, if we change the design of the network by keeping the left- and right-hand detect zones symmetric the term $(2q)^w$ in (29) is replaced by $\beta(q)$, making the query complexity

$$4\beta(q)$$

instead of $2(\beta(q) + (2q)^w)$.

In summary, one can make potentially significant savings in space complexity (from $O(\beta(q))$ to $O(q)$) at the cost of only moderately worse query complexity (from $\beta(q)$ to $4\beta(q)$) by maintaining symmetry between the two left- and right-hand outer detect zones. On the other hand, maintaining symmetry does not improve the security of our simulator (and obviously, costs an extra round).

6 An Attack on Two-round Confusion-Diffusion Networks

In this section we outline a simple distinguishing attack that shows confusion-diffusion networks of two rounds or less cannot be indifferentiable from a random permutation. Unfortunately we could not find a similarly general attack for networks with three rounds, which leaves open the possibility that 3- or 4-round confusion-diffusion network might already be indifferentiable.

The attack on 2-round networks requires $w \geq 2$, which is indeed a trivial requirement since if $w = 1$ then a 1-round network is already indifferentiable from a random permutation.

For concreteness we sketch the attack with $w = 2$. The confusion-diffusion network then has four S-boxes labeled $P_{i,j}$ for $(i, j) \in [2] \times [2]$ and one diffusion permutation $\pi : \{0,1\}^{2n} \to \{0,1\}^{2n}$. The S-boxes in the first round are $P_{1,j}$, $j \in [2]$, the S-boxes in the second round are $P_{2,j}$, $j \in [2]$.

We will say the distinguisher "rejects" if it believes that it is in the simulated world; "accepts" if it believes it is in the real world.

The distinguishing attack is as follows:

1. The distinguisher randomly chooses $\mathbf{x} \in \{0,1\}^{2n}$ and queries $\mathcal{Z}(\mathbf{x})$, where $\mathcal{Z} : \{0,1\}^{2n} \to \{0,1\}^{2n}$ is the random permutation, obtaining $\mathbf{y} \in \{0,1\}^{2n}$ as answer.
2. The distinguisher make the two S-box queries $P_{1,1}(\mathbf{x}[1])$ and $P_{2,1}^{-1}(\mathbf{y}[1])$ receiving answers $a \in \{0,1\}^n$ and $b \in \{0,1\}^n$ respectively.
3. If there exists no pair of values (c, d) such that $\pi(a\|c) = (b\|d)$, the distinguisher rejects.
4. If there exists a pair of values (c, d) such that $\pi(a\|c) = (b\|d)$, the distinguisher chooses any such pair, queries $P_{1,2}^{-1}(c)$ obtaining answer t, and accepts if and only if $\mathcal{Z}(\mathbf{x}[1]\|t)[1] = \mathbf{y}[1]$.

It is clear that the distinguisher always accepts in the real world. We now argue that the simulator has negligible chance of making the distinguisher accept.

It is helpful to picture the simulator as knowing the distinguisher's attack. Moreover, we can be generous to the simulator and give both $\mathbf{x}[1]$ and $\mathbf{y}[1]$ to the simulator before requesting the answers a and b from the simulator.

By choosing a and b, the simulator knows which of options 3 and 4 the distinguisher will execute, so the simulator is essentially choosing between these options when it chooses a and b.

Obviously, case 3 is no good for the simulator; moreover, the simulator has no further information on $\mathbf{x}$ and $\mathbf{y}$ besides $\mathbf{x}[1]$ and $\mathbf{y}[1]$, from which it is computationally infeasible, if $\mathcal{Z}$ is a random permutation, to locate a value t such that $\mathcal{Z}(\mathbf{x}[1]\|t)[1] = \mathbf{y}[1]$, and which rules out case 4. The simulator is therefore doomed.

7 Extensions and Future Work

In this section we make some speculative remarks about possible extensions of our work. In so doing, moreover, we will outline a broader perspective for understanding our simulator.

A *network segment* consists of an alternation of D-boxes and of S-box rounds, starting either with a D-box or with an S-box round and ending either with a D-box or with an S-box round. (Thus a standalone D-box is a "network segment", just like a standalone round of S-boxes is a "network segment".) The network segment is *even* if it starts and ends with S-box rounds; it is *odd* if starts and ends with D-box rounds.

For example, the untangle zones in our simulator are odd network segments regardless of XtraUntglRnds; all other zones (considering the left outer detect and right outer detect zones as separate zones) are even.

One of the main ingredients of the present "broader perspective" is the fact that all combinatorial properties of D-boxes that we have defined—conductance, all-but-one conductance, randomized preimage resistance, randomized collision resistance—are special cases of more general definitions that apply to network segments.

We will start by describing these generalized definitions. Formally, each property we define is a predicate of a sequence of D-boxes $\overline{\pi} = (\pi_1, \ldots, \pi_t)$. Each definition either targets even or odd network segments. In the case of an even network segment the case $t = 0$ is allowed, corresponding to a network segment that consists of a single S-box round.

For definitions that apply to even network segments we view $\overline{\pi}$ as part of a $(t+1)$-round confusion-difusion network $P[\mathcal{P}, \overline{\pi}]$ where $\mathcal{P} = \{P_{i,j} : i \in [t+1], j \in [w]\}$ and where π_i is the D-box between the S-box rounds $\{P_{i,j} : j \in [w]\}$ and $\{P_{i+1,j} : j \in [w]\}$. For definitions that apply to odd network segments we view $\overline{\pi}$ as part of a $(t-1)$-round confusion-diffusion network $P[\mathcal{P}, \overline{\pi}]$ where $\mathcal{P} = \{P_{i,j} : i \in \{2, \ldots, t\}, j \in [w]\}$, where π_i is again the D-box between rounds $\{P_{i,j} : j \in [w]\}$ and $\{P_{i+1,j} : j \in [w]\}$. In either case the permutations $P_{i,j} \in \mathcal{P}$ will be modeled as a set of *random* independent permutations accessible as oracles to an adversary.

Conductance. The *generalized conductance* $\mathsf{Cond}^*_{\overline{\pi}}$ of $\overline{\pi} = (\pi_1, \ldots, \pi_t)$ is defined with respect to even network segments. Formally $\mathsf{Cond}^*_{\overline{\pi}}$ is a function

$$\mathsf{Cond}^*_{\overline{\pi}} : \mathbb{Z} \times \mathbb{Z} \to [0,1]$$

defined by the following game. Let q and Q be integers. Let $\mathcal{P} = \{P_{i,j} : i \in [t+1], j \in [w]\}$ be the array of random permutation oracles for the even network, as described above. We give an adversary q queries (forward or backward) to each of the oracles in $\mathcal{P}$. We the adversary is done, we count the number of completed paths that can be assembled from these queries (i.e., inputs $\mathbf{x} \in \{0,1\}^{wn}$ to $P[\mathcal{P}, \overline{\pi}]$ that can be evaluated to outputs using only the adversary's queries). We say the adversary "wins" if this number of paths is more than Q. We let

$$\mathsf{Cond}^*_{\overline{\pi}}(q, Q)$$

be the supremum[20] of the probability of winning this game, taken over all possible adversaries.

Relation to "ordinary" conductance. If $t = 1$, i.e., $\overline{\pi} = (\pi_1)$, then $\mathsf{Cond}^*_{\overline{\pi}}$ is characterized by Cond_{π_1} in the sense that

$$\mathsf{Cond}^*_{\overline{\pi}}(q, Q) = \begin{cases} 0 & \text{if } Q > \mathsf{Cond}_{\pi_1}(q), \\ 1 & \text{if } Q \leq \mathsf{Cond}_{\pi_1}(q). \end{cases}$$

This is easy to see from the definitions. (Indeed, the number of paths assembled is easily seen to be purely a function of the sets $U_j := \{y : P^{-1}_{1,j}(y) \neq \bot\}$, $1 \leq j \leq w$, and $V_j := \{x : P_{2,j}(x) \neq \bot\}$, $1 \leq j \leq w$, where we write $P^{-1}_{1,j}(y) \neq \bot$ to mean that the adversary either made the query $P^{-1}_{1,j}(y)$ or made a query $P_{1,j}(x)$ whose answer resulted in y. Hence, the best possible adversary will simply choose the sets $U_1, \ldots, V_w$ beforehand, and make the relevant inverse queries in the first round and the relevant forward queries in the second round.)

Moreover when $t = 0$ then $\overline{\pi} = ()$ where $()$ denotes an empty sequence of S-boxes, and one has

$$\mathsf{Cond}^*_{\overline{\pi}}(q, Q) = \begin{cases} 0 & \text{if } Q > q^w, \\ 1 & \text{if } Q \leq q^w. \end{cases}$$

Thus one can observe that $\alpha(q)$, as defined in our main proof, is simply the threshold Q at which the generalized conductance $\mathsf{Cond}^*_{\overline{\pi}}(2q, \cdot)$ of the middle round goes from 1 to 0, whether or not $\mathsf{XtraMiddleRnd}$ is on.

Likewise, $\beta(q)$ is the same threshold, only with respect to the generalized conductance $\mathsf{Cond}^*_{\overline{\pi}}(q + \alpha(q), \ldots)$ and with respect to the sequence of D-boxes $\overline{\pi}$ in the outer left detect zone.

All-but-one conductance. The *generalized all-but-one conductance* $\mathsf{aboCond}^*_{\overline{\pi}}$ of $\overline{\pi}$ is also defined with respect to even network segments. Formally, $\mathsf{aboCond}^*_{\overline{\pi}}$ is a function

$$\mathsf{aboCond}^*_{\overline{\pi}} : \mathbb{Z} \to [0,1]$$

[20] Given the finite amount of randomness involved, it is easy to easy to see that the supremum is achieved by some adversary. Hence we might also say *maximum*.

defined with respect to the following game. Let q be an integer. An adversary is given q queries to each of the permutations in $\mathcal{P} = \{P_{i,j} : i \in [t+1], j \in [w]\}$. We say that a given adversarial query is *internal* if it has the form $P_{i,j}(x)$ with $i < t+1$ or $P_{i,j}^{-1}(y)$ with $i > 1$. Moreover we assume that the adversary does not make redundant queries. Then the adversary wins if, at any point, it makes an internal query such that a completed path can be assembled using (the result of) this internal query together with queries that were made prior to this internal query. We let

$$\mathsf{aboCond}^*_{\overline{\pi}}(q)$$

be the maximum probability of an adversary winning this game, taken over all adversaries.

Relation to "ordinary" all-but-one conductance. If $t = 1$ then one can verify that

$$\mathsf{aboCond}^*_{\overline{\pi}}(q) \leq \frac{2wq \cdot \mathsf{aboCond}_{\pi_1}(q)}{N - q}$$

by a straightforward union bound. On the other hand if $t = 0$ then one can verify that

$$\mathsf{aboCond}^*_{\overline{\pi}}(q) = 0$$

regardless of q, since there is no such thing as an internal query in this case.

Entry-wise randomized preimage resistance. The *generalized entry-wise randomized preimage resistance* $\mathsf{MaxPreim}^*_{\overline{\pi}}$ of $\overline{\pi}$ is defined with respect to odd network segments. Formally $\mathsf{MaxPreim}^*_{\overline{\pi}}$ is a function

$$\mathsf{MaxPreim}^*_{\overline{\pi}} : \mathbb{Z} \to [0,1]$$

defined by the following game. Let q be an integer. An adversary is allowed q queries to each of the permutations in $\mathcal{P} = \{P_{i,j} : i \in \{2, \dots, t\}, j \in [w]\}$ in the odd network. After making its queries, the adversary names an index $j \in [w]$ and a set of values $\{x_{j'} : j' \neq j\}$ as well as an index $h \in [w]$ and a value $y_h \in \{0,1\}^n$. A value x_j is sampled uniformly at random from $\{0,1\}^n$ and the (odd) network is evaluated on the input vector $\mathbf{x} = (x_1, \dots, x_w)$, making additional queries to the $P_{i,j}$'s if necessary, until an output vector $\mathbf{y}$ is obtained. (Thus $\mathbf{x}$ is the input to the first D-box, $\mathbf{y}$ is the output of the last D-box.) The adversary wins if $\mathbf{y}[h] = y_h$. We define

$$\mathsf{MaxPreim}^*_{\overline{\pi}}(q)$$

to be the maximum probability of an adversary winning this game, taken over all adversaries.

Relation to "ordinary" entry-wise randomized preimage resistance. If $t = 1$ then there are no S-boxes to query, so the value q is immaterial, and it is easily verified that

$$\mathsf{MaxPreim}^*_{\overline{\pi}}(q) = \frac{\mathsf{MaxPreim}(\pi_1)}{N}.$$

If $t = 2$ one can also check that

$$\mathsf{MaxPreim}^*_{\overline{\pi}}(q) \leq \frac{q \cdot \mathsf{MaxPreim}(\pi_1)}{N} + \frac{q}{N} + \frac{\mathsf{MaxPreim}(\pi_2)}{N}$$

by using an argument somewhat similar to the case $\mathsf{XtraUntglRnds} = \textbf{true}$ of Lemma 38 in Section 5.3. (A factor w is missing from the first fraction due to the fact that it is only necessary for, say, the query $P_{2,1}(\pi_1(\mathbf{x})[1])$ to be "fresh", among all the queries $P_{2,j}(\pi_1(\mathbf{x})[j])$, $1 \leq j \leq w$. The fraction q/N accounts for the difference between sampling uniformly at random from $\{0,1\}^n$ and sampling uniformly at random from a subset of $\{0,1\}^n$ of size $N-q$; specifically, the latter kind of sampling can be implemented by sampling uniformly at random, and giving the adversary a win if the sample falls in the forbidden subset of q elements.)

Entry-wise randomized collision resistance. The *generalized entry-wise randomized preimage resistance* $\mathsf{MaxColl}^*_{\overline{\pi}}(q)$ of $\overline{\pi}$ is also defined with respect to odd network segments. Formally

$$\mathsf{MaxColl}^*_{\overline{\pi}} : \mathbb{Z} \to [0,1]$$

is a function defined via the following game. Let q be an integer. An adversary is allowed q queries to each of the odd network permutations $\mathcal{P} = \{P_{i,j} : i \in \{2,\dots,t\}, j \in [w]\}$. After making its queries, the adversary names an index $j \in [w]$ and two set of values $\{x_{j'} : j' \neq j\}$, $\{x'_{j'} : j' \neq j\}$ such that $x_{j'} \neq x'_{j'}$ for at least one $j' \neq j$. A uniform random value $x_j \in \{0,1\}^n$ is revealed, and the network is evaluated (requiring new queries if necessary) on the inputs $\mathbf{x} = (x_1,\dots,x_w)$ and $\mathbf{x}' = (x'_1,\dots,x'_w)$ where $x'_j := x_j$, resulting in outputs $\mathbf{y}$ and $\mathbf{y}'$. The adversary wins if $\mathbf{y}[h] = \mathbf{y}'[h]$ for some $h \in [w]$. We define

$$\mathsf{MaxColl}^*_{\overline{\pi}}(q)$$

to be the maximum probability of an adversary winning this game, taken over all adversaries.

Relation to "ordinary" entry-wise randomized collision resistance. If $t = 1$ then q is extraneous and it is easy to see that

$$\mathsf{MaxColl}^*_{\overline{\pi}}(q) = \frac{w\mathsf{MaxColl}(\pi_1)}{N}$$

whereas if $t = 2$ then one can see that

$$\mathsf{MaxColl}^*_{\overline{\pi}}(q) \leq \frac{wq \cdot \mathsf{MaxPreim}(\pi_1)}{N} + \frac{q+1}{N} + \frac{w \cdot \mathsf{MaxPreim}(\pi_2)}{N}. \tag{30}$$

similarly to the argument in the case $\mathsf{XtraUntglRnds} = \textbf{true}$ proof of Lemma 38, Section 5.3.

A more general theorem. One can easily state a more general version of our main theorem. The parameters of the network would be sequences of D-boxes $\overline{\nu}$, $\overline{\pi}_I$, $\overline{\pi}_J$, $\overline{\tau}$, $\overline{\pi}_K$ and $\overline{\pi}_L$ that appear in respectively the left outer detect zone, the first untangle zone, the second untangle zone, the middle detect zone, the third untangle zone and the fourth untangle zone. Since the untangle zones are implemented by odd network segments, each of $\overline{\pi}_I$, $\overline{\pi}_J$, $\overline{\pi}_K$ and $\overline{\pi}_L$ contain at least one D-box each.

To state such a generalized theorem, let $\eta = (\overline{\nu}, \overline{\pi}_I, \overline{\pi}_J, \overline{\tau}, \overline{\pi}_K, \overline{\pi}_L)$ (a sequence-of-sequences, not a concatenation of sequences) and let

$$\begin{aligned}\mathsf{MaxPreim}^*_{\eta}(q) &:= \max(\mathsf{MaxPreim}^*_{\overline{\pi}_I}(q), \mathsf{MaxPreim}^*_{\overline{\pi}_J^-}(q), \mathsf{MaxPreim}^*_{\overline{\pi}_K}(q), \mathsf{MaxPreim}^*_{\overline{\pi}_L^-}(q))\\ \mathsf{MaxColl}^*_{\eta}(q) &:= \max(\mathsf{MaxColl}^*_{\overline{\pi}_I}(q), \mathsf{MaxColl}^*_{\overline{\pi}_J^-}(q), \mathsf{MaxColl}^*_{\overline{\pi}_K}(q), \mathsf{MaxColl}^*_{\overline{\pi}_L^-}(q))\end{aligned}$$

where $\overline{\pi}^-$ stands for the sequence of inverses of permutations in $\overline{\pi}$, in reverse order than $\overline{\pi}$. Moreover let $|\overline{\pi}|$ be the length (number of sequences) in $\overline{\pi}$, and let

$$r_\nu = |\overline{\nu}| + 1, r_I = |\overline{\pi}_I| - 1, r_J = |\overline{\pi}_J| - 1, r_\tau = |\overline{\tau}| + 1, r_K = |\overline{\pi}_K| - 1, r_L = |\overline{\pi}_L| - 1$$

be the number of (S-box) rounds in the outer left detect zone, etc, and let

$$r = r_\nu + r_I + r_J + r_\tau + r_K + r_L + 3.$$

be the total number of rounds in the network. (The '3' accounts for rounds D, E and H in our original nomenclature.) Finally let

$$\overline{\eta} = \overline{\nu} \| \overline{\pi}_I \| \overline{\pi}_J \| \overline{\tau} \| \overline{\pi}_K \| \overline{\pi}_L$$

be the concatenation of the D-boxes sequences $\overline{\nu}, \ldots, \overline{\pi}_L$ into a single sequence, and let $\mathcal{Q} = \{P_{i,j} : i \in [r], j \in [w]\}$ denote a set of random permutations.

Then we have the following result (stated without proof):

Theorem 2. *For all integers $Q_{\overline{\nu}}$ and $Q_{\overline{\tau}}$, the confusion-diffusion network $P[\mathcal{Q}, \overline{\eta}]$ achieves (t_S, q_S, ε)-indifferentiability from a random permutation $\mathcal{Z} : \{0,1\}^{wn} \to \{0,1\}^{wn}$, for ε equal to*

$$\begin{aligned}
&\mathsf{Cond}^*_{\overline{\tau}}(2q, Q_{\overline{\tau}}) + \mathsf{Cond}^*_{\overline{\nu}}(q + Q_{\overline{\tau}}, Q_{\overline{\nu}}) && (31)\\
&+ \frac{w(q + Q_{\overline{\tau}})^2}{N - q - Q_{\overline{\tau}}} + \frac{(q + wQ_{\overline{\tau}})(q + \mathcal{Q}_{\overline{\tau}})}{N - q - Q_{\overline{\tau}}} + \frac{2w(q + Q_{\overline{\tau}})Q_{\overline{\tau}}}{N - q - Q_{\overline{\tau}}} && (32)\\
&+ \mathsf{aboCond}^*_{\overline{\tau}}(2q) + \mathsf{aboCond}^*_{\overline{\pi}}(q + Q_{\overline{\tau}}) && (33)\\
&+ 2wQ_{\overline{\tau}} \left[(q + Q_{\overline{\tau}}) \cdot \mathsf{MaxPreim}^*_{\eta}(q + Q_{\overline{\tau}}) + (q + Q_{\overline{\tau}})/N\right] && (34)\\
&+ 2Q_{\overline{\tau}} \left[Q_{\overline{\tau}} \cdot \mathsf{MaxColl}^*_{\eta}(q + Q_{\overline{\tau}}) + (q + Q_{\overline{\tau}})/N\right] && (35)\\
&+ \frac{Q_{\overline{\nu}}(q + Q_{\overline{\tau}})^w}{N^w - q - Q_{\overline{\tau}}} + \frac{1}{N^w} && (36)
\end{aligned}$$

and for $q_S = Q_{\overline{\tau}}$, $t_S = O(w(q + Q_{\overline{\tau}})^w)$. Here $q = q_0(1 + rw)$ where r is the number of S-box rounds.

In this statement, $Q_{\overline{\tau}}$ is the (expected) maximum number of matching query sets for the middle detect zone, given $2q$ queries to each S-box in that detect zone; $Q_{\overline{\nu}}$ is the (expected) maximum number of matching query sets for the outer left detect zone, given $q + Q_{\overline{\tau}}$ queries to each S-box in that zone. The first two terms in the security bounds account for the distinguisher's probability of beating these "expected" conductances, and $Q_{\overline{\tau}}$, $Q_{\overline{\nu}}$ should be set high enough to make the first two terms negligible.

The remainder of the security bound affords the following term-by-term interpretation (we slightly abusively refer to "game G_3", having the natural extrapolation of our previous proof in mind):

- the first term of line (32) upper bounds the probability that a fresh query to Z or Z^{-1} in game G_3 accidentally hits a predefined value in one of the S-boxes of the last or first round, respectively (compare with Lemma 26)

- the second term of line (32) upper bounds the probability that a forward query in the last round or a backward query in the first round causes abort in game G_3 by "hitting" (the coordinate of) a pre-existing query to Z or Z^{-1} (compare with Lemma 28)

- the third term of line (32) upper bounds the probability that procedures $\mathrm{D}/\mathrm{D}^{-1}$ or $\mathrm{E}/\mathrm{E}^{-1}$ abort in game G_3 (compare with Lemma 31)

- the first term of line (33) upper bounds the probability that a middle path completion is ever triggered in game G_3 by a query that is neither a forward query to the last round of the middle detect zone nor a backward query in the first round of the middle detect zone (compare with Lemma 34)

- the second term of line (33) is similar, but for the left outer detect zone (compare with Lemma 29)

- line (34) upper bounds the probability that LeftAdapt or RightAdapt ever abort in game G_3 because an input or output to D (respectively E) hits a predefined value in D (respectively E);

the extra additive term $(q + Q_{\overline{\tau}})/N$ inside the square brackets accounts for the difference between uniform sampling and lazy sampling

- line (35) upper bounds the probability that LeftAdapt or RightAdapt ever abort in game G_3 because inputs or outputs to D (respectively E) for two different adapted paths collide; the extra additive term $(q + Q_{\overline{\tau}})/N$ again accounts for the difference between uniform sampling and lazy sampling

- the first term of line (36) upper bounds the probability that a fresh query to Z or Z^{-1} by the simulator in game G_1 (or G_2 or G_3) accidentally hits a column of defined queries (compare with Lemma 40)

- the second term of line (36) is incurred by probability ratios in the randomness map from G_3 to G_4, cf. Lemma 46

One can observe that $\mathsf{MaxPreim}^*_{\eta}$ and $\mathsf{MaxColl}^*_{\eta}$ appear with the same arguments and with essentially the same coefficients. Hence it makes sense to define

$$\mathsf{MaxCoPr}^*_{\overline{\pi}}(q) := \max(\mathsf{MaxPreim}^*_{\overline{\pi}}(q), \mathsf{MaxColl}^*_{\overline{\pi}}(q))$$

and to characterize the quality of an untangle zone by the single function $\mathsf{MaxCoPr}^*(\cdot)$. Typically we expect $\mathsf{MaxCoPr}^*_{\overline{\pi}} = \mathsf{MaxColl}^*_{\overline{\pi}}$, though it is possible to devise pathological examples for which $\mathsf{MaxPreim}^*_{\overline{\pi}} \gg \mathsf{MaxColl}^*_{\overline{\pi}}$.

Making it Fast. Given these general observations, a basic class of questions that can be pursued is the following:

how can one build fast networks[21] *with low conductance/randomized preimage resistance/randomized collision resistance?*

As a special case of this question, we have:

how can one build fast diffusion permutations with low conductance/randomized preimage resistance/randomized collision resistance?

As seen by the form of our main theorem (or equivalently, as evidenced by (30)), the second question can also potentially serve as a building block for the first.

Importantly, one should not necessarily equate efficiency with the smallest possible number of rounds. For example, a $\mathrm{GF}(2^n)$-linear diffusion permutation whose matrix has no zero entries has low randomized preimage resistance on its own. But arbitrary field operations are slow, and one could easily imagine there might exist a sequence $\overline{\pi} = (\pi_1, \pi_2)$ with good randomized preimage resistance that outperforms—that is, even taking into account the S-box round between π_1 and π_2—a standalone linear diffusion permutation. The question is whether the use of an extra round makes it possible to sufficiently simplify the diffusion permutations.

Similarly, a network zone with low conductance might be more quickly achieved by using three S-box rounds and two "cheap" diffusion permutations than by using one S-box round and one "expensive" diffusion permutation that controls conductance on its own.

As a concrete idea, consider the D-boxes $\oplus_{\uparrow} : \{0,1\}^{wn} \to \{0,1\}^{wn}$, $\oplus_{\downarrow} : \{0,1\}^{wn} \to \{0,1\}^{wn}$ defined by

$$\oplus_{\uparrow}(\mathbf{x})[j] = \bigoplus_{1 \le h \le j} \mathbf{x}[h],$$

$$\oplus_{\downarrow}(\mathbf{x})[j] = \bigoplus_{j \le h \le w} \mathbf{x}[h].$$

[21] By "networks" we mean "network segments" or, even more precisely, "D-box sequence". In this discussion we write *conductance* instead of *generalized conductance*, etc.

Then $\oplus_\uparrow$ and $\oplus_\downarrow$ can be efficiently implemented in software or in hardware. While $\oplus_\uparrow$, $\oplus_\downarrow$ both have worst-possible randomized preimage and collision resistance on their own, one can check that

$$\mathsf{MaxPreim}^*_{(\oplus_\downarrow,\oplus_\uparrow)}(q) \le \frac{q}{N} + \frac{1}{N-q}$$

and moreover that

$$\mathsf{MaxPreim}^*_{(\oplus_\downarrow,\oplus_\uparrow,\oplus_\downarrow,\oplus_\uparrow)}(q) \le \frac{q}{N} + \frac{2q}{N-q} + \frac{1}{N-q}$$
$$\mathsf{MaxColl}^*_{(\oplus_\downarrow,\oplus_\uparrow,\oplus_\downarrow,\oplus_\uparrow)}(q) \le \frac{q}{N} + \frac{2q}{N-q} + \frac{w}{N-q}$$

which suggests that the sequence $\overline{\oplus}_4 := (\oplus_\downarrow,\oplus_\uparrow,\oplus_\downarrow,\oplus_\uparrow)$ could serve as a very efficient[22] untangle zone. A confusion-diffusion network that used this untangle zone four times over[23] and that used, say, two S-box rounds for each of the outer left and middle detect zones would have $2{\cdot}2+4{\cdot}3+2+1 = 19$ rounds. The main "takeaway lesson" being that by targeting "the right combinatorial property at the right time" one can potentially arrive at sequences of very simple D-boxes.

Tantalizingly, however—and as far as we know!—the above upper bounds on $\mathsf{MaxPreim}^*_{\overline{\oplus}_4}$, $\mathsf{MaxColl}^*_{\overline{\oplus}_4}$ crucially require the S-boxes in each round to be independent. Another potential research direction, therefore, would be to devise D-box sequences that are "S-box reuse resistant", or more precisely, that preserve low randomized preimage and collision resistance even when the S-boxes in a column are all the same, or even when all the S-boxes in the network segment are the same.

In parallel, one could envision extending Theorem 1 or Theorem 2 to cover the possibility of non-independent S-boxes. (Within each round, within each zone, or—most ambitiously but also most messily—within the entire network.)

Of course, and even with extensions to cover the case of non-independent S-boxes, one should bear in mind that security guarantees of the form $q^{O(1)}/N$ are meaningless for, say, $N = 2^8$, not to mention the fact that our results are obtained with respect to an ideal model, while S-boxes must be concretely implemented. Still, we tend to believe that indifferentiability represents a nontrivial and potentially important "sanity check" for the high-level design of a confusion-diffusion network, and which has been absent so far. It could therefore be worthwhile to pursue more concrete instantiations (accompanied by "real" cryptanalysis) for constructions that fall under the umbrella of Theorem 2, or some even more general extension thereof.

References

1. Elena Andreeva, Andrey Bogdanov, Yevgeniy Dodis, Bart Mennink, and John P. Steinberger. On the indifferentiability of key-alternating ciphers. In *Advances in Cryptology—CRYPTO 2013* (volume 1), pages 531–550.
2. Guido Bertoni, Joan Daemen, Michael Peeters, and Gilles Van Assche. On the indifferentiability of the sponge construction. In Smart [27], pages 181–197.

[22] In fact one could even use the simpler transformation

$$\oplus'_\downarrow(\mathbf{x})[j] = \begin{cases} \mathbf{x}[j] & \text{if } j \neq w \\ \bigoplus_{1\le h\le w} \mathbf{x}[h] & \text{if } j = w. \end{cases}$$

in place of $\oplus_\downarrow$, though one's cryptographic instincts recoil in the face of such weak diffusion. Interestingly, too, $(\oplus_\uparrow,\oplus'_\downarrow)$ has bad randomized preimage resistance, even while $(\oplus'_\downarrow,\oplus_\uparrow)$ has randomized preimage resistance similar to $(\oplus_\downarrow,\oplus_\uparrow)$. Similarly $(\oplus_\uparrow,\oplus'_\downarrow,\oplus_\uparrow,\oplus'_\downarrow)$ has bad randomized collision resistance while $(\oplus'_\downarrow,\oplus_\uparrow,\oplus'_\downarrow,\oplus_\uparrow)$ has good randomized collision resistance.

[23] Indeed, our bounds don't suggest that different untangle zones should be implemented any differently apart from the obvious left-right mirror symmetry, i.e., $\overline{\pi}_J = \overline{\pi}_I^-$.

3. D. Chakraborty and P. Sarkar. A new mode of encryption providing a tweakable strong pseudo-random permutation. In *Proceedings of FSE '06*, LNCS 4047, pp. 293–309, 2006.
4. D. Chakraborty and P. Sarkar. HCH: A new tweakable enciphering scheme using the hash-encrypt-hash approach. In *Proceedings of Indocrypt '06*, LNCS 4329, pp. 287–302, 2006.
5. Jean-Sébastien Coron, Yevgeniy Dodis, Cécile Malinaud, and Prashant Puniya. Merkle-damgård revisited: How to construct a hash function. In Victor Shoup, editor, *Advances in Cryptology—CRYPTO 2005*, volume 3621 of *LNCS*, pages 430–448. Springer-Verlag, 14–18 August 2005.
6. Jean-Sébastien Coron, Yevgeniy Dodis, Avradip Mandal, and Yannick Seurin. A domain extender for the ideal cipher. To appear in the *Theory of Cryptography Conference* (TCC), February 2010.
7. Jean-Sébastien Coron, Jacques Patarin, and Yannick Seurin. The random oracle model and the ideal cipher model are equivalent. In Wagner [28], pages 1–20.
8. Yevgeniy Dodis, Krzysztof Pietrzak, and Prashant Puniya. A new mode of operation for block ciphers and length-preserving macs. In Smart [27], pages 198–219.
9. Yevgeniy Dodis, Leonid Reyzin, Ronald L. Rivest, and Emily Shen. Indifferentiability of permutation-based compression functions and tree-based modes of operation, with applications to md6. In Orr Dunkelman, editor, *Fast Software Encryption: 16th International Workshop, FSE 2009*, volume 5665 of *Lecture Notes in Computer Science*, pages 104–121. Springer-Verlag, 22–25 February 2009.
10. Yevgeniy Dodis, Thomas Ristenpart, and Thomas Shrimpton. Salvaging merkle-damgård for practical applications. In Antoine Joux, editor, *Advances in Cryptology - EUROCRYPT 2009*, volume 5479 of *LNCS*, pages 371–388. Springer-Verlag, 2009.
11. Hörst Feistel. Cryptographic coding for data-bank privacy. IBM Technical Report RC-2827, March 18 1970.
12. Hörst Feistel, William A. Notz, J. Lynn Smith. Some Cryptographic Techniques for Machine-to-Machine Data Communications. IEEE proceedings, **63**(11), pages 1545–1554, 1975.
13. S.R. Fluhrer and D.A. McGrew. The extended codebook (XCB) mode of operation. Technical Report 2004/078, IACR eprint archive, 2004.
14. S. Halevi. Invertible Universal hashing and the TET Encryption Mode. In *Proceedings of CRYPTO '07*, LNCS 4622, pp. 412–429, 2007.
15. Thomas Holenstein, Robin Künzler, and Stefano Tessaro. The equivalence of the random oracle model and the ideal cipher model, revisited. In Lance Fortnow and Salil P. Vadhan, editors, *Proceedings of the 43rd ACM Symposium on Theory of Computing, STOC 2011, San Jose, CA, USA, 6-8 June 2011*, pages 89–98. ACM, 2011.
16. Rodolphe Lampe and Yannick Seurin. How to construct an ideal cipher from a small set of public permutations. In Kazue Sako and Palash Sarkar, editors, *Advances in Cryptology - ASIACRYPT 2013 - 19th International Conference on the Theory and Application of Cryptology and Information Security, Bengaluru, India, December 1-5, 2013, Proceedings, Part I*, volume 8269 of *Lecture Notes in Computer Science*, pages 444–463. Springer, 2013.
17. M. Luby and C. Rackoff. How to construct pseudorandom permutations and pseudorandom functions. *SIAM Journal on Computing*, 17(2):373–386, April 1988.
18. Ueli M. Maurer, Renato Renner, and Clemens Holenstein. Indifferentiability, impossibility results on reductions, and applications to the random oracle methodology. In Moni Naor, editor, *First Theory of Cryptography Conference — TCC 2004*, volume 2951 of *LNCS*, pages 21–39. Springer-Verlag, February 19–21 2004.
19. Ueli M. Maurer, and Stefano Tessaro. Domain Extension of Public Random Functions: Beyond the Birthday Barrier In *Advances in Cryptology—CRYPTO 2007*, August 2007.
20. Eric Miles and Emanuele Viola. Substitution-permutation networks, pseudorandom functions, and natural proofs. In *Advances in Cryptology - CRYPTO 2012 - 32nd Annual Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2012. Proceedings*, pages 68–85, 2012.
21. M. Naor and O. Reingold, *On the construction of pseudorandom permutations: Luby-Rackoff revisited*, J. of Cryptology, 1999. Preliminary Version: STOC 1997.
22. Thomas Ristenpart and Hovav Shacham and Thomas Shrimpton, Careful with Composition: Limitations of the Indifferentiability Framework. EUROCRYPT 2011, LNCS 6632, 487–506.
23. Phillip Rogaway and John P. Steinberger. Constructing cryptographic hash functions from fixed-key blockciphers. In Wagner [28], pages 433–450.
24. Yannick Seurin, *Primitives et protocoles cryptographiques à sécurité prouvée.* PhD thesis, Université de Versailles Saint-Quentin-en-Yvelines, France, 2009.
25. Claude E. Shannon. Communication theory of secrecy systems. *Bell System Technical Journal*, 28(4):656–715, October 1949. Available at http://www.cs.ucla.edu/ jkong/research/security/shannon.html and http://www3.edgenet.net/dcowley/docs.html.
26. Thomas Shrimpton and Martijn Stam. Building a collision-resistant compression function from non-compressing primitives. In Luca Aceto, Ivan Damgård, Leslie Ann Goldberg, Magnús M. Halldórsson, Anna Ingólfsdóttir, and Igor Walukiewicz, editors, *ICALP (2)*, volume 5126 of *Lecture Notes in Computer Science*, pages 643–654. Springer, July 7-11 2008.

27. Nigel P. Smart, editor. *Advances in Cryptology - EUROCRYPT 2008*, volume 4965 of *LNCS*. Springer-Verlag, 2008.
28. David Wagner, editor. *Advances in Cryptology - CRYPTO 2008*, volume 5157 of *LNCS*. Springer-Verlag, 2008.
29. P. Wang, D. Feng, and W. Wu. HCTR: A variable-input-length enciphering mode. In *Proceedings of CISC '05*, LNCS 3822, pp. 175–188, 2005.

A Probabilistic constructions of low-conductance permutations

In this section we show the probabilistic construction of permutations $\tau : \{0,1\}^{wn} \to \{0,1\}^{wn}$ with low conductance and low all-but-one conductance.

Let $U_1, \ldots, U_w, V_1, \ldots, V_w \subseteq \{0,1\}^n$ such that $|U_j| = |V_j| = q$, $1 \le j \le w$. Let $a \in \mathbb{N}$. We start by upper bounding the probability that when a permutation $\tau : \{0,1\}^{wn} \to \{0,1\}^{wn}$ is chosen uniformly at random,

$$|\{\mathbf{x} \in U_1 \times \cdots \times U_w : \tau(\mathbf{x}) \in V_1 \times \cdots \times V_w\}| \ge a. \tag{37}$$

We assume that $q^w \le N^w/2$ (where $N = 2^n$ as usual), which means that q cannot be too close to N. (In our applications we have $q \le \sqrt{N}$, which is much smaller.)

Let $\mathbf{x}_1, \ldots, \mathbf{x}_{q^w}$ be an enumeration of the elements in $U_1 \times \cdots \times U_w$. If $\tau : \{0,1\}^{wn} \to \{0,1\}^{wn}$ is a random permutation and the values $\tau(\mathbf{x}_1), \ldots, \tau(\mathbf{x}_\ell)$ have already been revealed, then the probability that $\tau(\mathbf{x}_{\ell+1}) \in V_1 \times \cdots \times V_w$ is at most

$$\frac{q^w}{N^w - \ell} \le \frac{q^w}{N^w - q^w} \le \frac{2q^w}{N^w}$$

by the assumption $q^w \le N^w/2$. That is,

$$\Pr_\tau [\tau(\mathbf{x}_{\ell+1}) \in V_1 \times \cdots \times V_w \,|\, \tau(\mathbf{x}_1), \ldots, \tau(\mathbf{x}_\ell)] \le \frac{2q^w}{N^w}$$

for any values of $\tau(\mathbf{x}_1), \ldots, \tau(\mathbf{x}_\ell)$, $\ell \le q^w - 1$.

Now the probability that (37) occurs is upper bounded by

$$\binom{q^w}{a}\left(\frac{2q^w}{N^w}\right)^a \le q^{wa}\left(\frac{2q^w}{N^w}\right)^a = \left(\frac{2q^{2w}}{N^w}\right)^a.$$

By a union bound, then, the probability (taken over the choice of a random τ) that there *exist* $U_1, \ldots, U_w, V_1, \ldots, V_w$ such that (37) occurs is upper bounded by

$$\binom{N}{q}^{2w}\left(\frac{2q^{2w}}{N^w}\right)^a \le N^{2qw}\left(\frac{2q^{2w}}{N^w}\right)^a. \tag{38}$$

This probability is less than 1, thus, as long as $q^{2w} \le N^w/4$ (which essentially translates to $q < \sqrt{N}$) and as long as $a > 2nqw$. We have thus proved:

Theorem 3. *Let $q, n, w \in \mathbb{N}$ be such that $q^{2w} \le N^w/4$, $N = 2^n$. Then there exists a permutation $\tau : \{0,1\}^{wn} \to \{0,1\}^{wn}$ such that*

$$\mathsf{Cond}(\tau) \le 2nqw + 1.$$

The conductance of the permutation τ constructed by Theorem 3 is thus quite close to q, being only a factor $2nw = O(\log(N^w))$ away from q. One can also that observe that (38) does not deliver much better conductance even for $q^{2w} \ll N^w$, since

$$N^{2qw}\left(\frac{2q^{2w}}{N^w}\right)^a \geq N^{2qw}\cdot\frac{1}{N^{wa}}$$

means that we at least need $a > 2q$ in order to strictly upper bound the union bound by 1.

All-but-one conductance. For the case of all-but-one-conductance, start by fixing sets $U_1, \ldots, U_w$, $V_2, \ldots, V_w \subseteq \{0,1\}^n$ and a value $a \in \mathbb{N}$. This time we focus on the probability (over random choice of τ) that

$$|\{\mathbf{x} \in U_1 \times \cdots \times U_w : \tau(\mathbf{x}) \in \{0,1\}^n \times V_2 \times \cdots \times V_w\}| \geq a \tag{39}$$

occurs.

As before let $\mathbf{x}_1, \ldots, \mathbf{x}_{q^w}$ be an enumeration of $U_1 \times \cdots \times U_w$. Then for $0 \leq \ell \leq q^w - 1$, we have

$$\Pr\left[\tau(\mathbf{x}_{\ell+1}) \in \{0,1\}^n \times V_2 \times \cdots \times V_w \,|\, \tau(\mathbf{x}_1), \ldots, \tau(\mathbf{x}_\ell)\right] \leq \frac{Nq^{w-1}}{N - q^w} \leq \frac{Nq^{w-1}}{N^w/2} = \frac{2q^{w-1}}{N^{w-1}}$$

for any values of $\tau(\mathbf{x}_1), \ldots, \tau(\mathbf{x}_\ell)$.

The probability that (39) occurs is thus upper bounded by

$$\binom{q^w}{a}\left(\frac{2q^{w-1}}{N^{w-1}}\right)^a \leq q^{wa}\left(\frac{2q^{w-1}}{N^{w-1}}\right)^a = \left(\frac{2q^{2w-1}}{N^{w-1}}\right)^a.$$

By a union bound, the probability that there *exist* sets $U_1, \ldots, U_w, V_2, \ldots, V_w$ such that (39) occurs is at most

$$\binom{N}{q}^{2w-1}\left(\frac{2q^{2w-1}}{N^{w-1}}\right)^a \leq N^{q(2w-1)}\left(\frac{2q^{2w-1}}{N^{w-1}}\right)^a.$$

Unfortunately, for this bound to give a non-void result we need

$$\frac{2q^{2w-1}}{N^{w-1}} < 1$$

which is too constraining to be useful, since in our setting we need to let q approach $N^{1/2}$. Hence, a different approach is needed.

Multiplicative Chernoff bound. As above, start by fixing sets $U_1, \ldots, U_w, V_2, \ldots, V_w$ and let $\mathbf{x}_1, \ldots, \mathbf{x}_{q^w}$ be an enumeration of $U_1 \times \cdots \times U_w$. Also assume that $q \leq \sqrt{N}$, since this is the regime which interests us. Let X_i be the indicator random variable for the event $\tau(\mathbf{x}_i) \in \{0,1\}^n \times V_2 \times \cdots \times V_w$, $1 \leq i \leq q^w$. Then

$$\Pr[X_i = 1 \,|\, X_1 \ldots X_{i-1}] \leq \frac{Nq^{w-1}}{N^w - q^w} \leq \frac{2Nq^{w-1}}{N^w} = \frac{2q^{w-1}}{N^{w-1}} \tag{40}$$

for all outcomes $X_1 \ldots X_{i-1}$. Let $Y_1, \ldots, Y_{q^w}$ be *independent* indicator random variables such that

$$\Pr[Y_i = 1] = \frac{2q^w}{N^w}$$

for all i. Let $X = \sum_i X_i$, $Y = \sum_i Y_i$. An easy coupling argument shows that

$$\Pr[X > t] \leq \Pr[Y > t]$$

for all $t > 0$, by (40).

Let $\mu = E[Y] = q^w(2q^{w-1}/N^{w-1}) = 2q^{2w-1}/N^{w-1}$. Since $N \geq q^2$ we have

$$\mu \leq 2q^{2w-1}/(q^2)^{w-1} = 2q \leq qw\ln(N) \tag{41}$$

(using $w \geq 2$) and we also have

$$\mu \leq 2qw\ln(N)/(e^2+1) \tag{42}$$

and

$$\mu \leq qw\ln(N) - \frac{1}{2}\ln(2w) \tag{43}$$

for sufficiently large n (e.g., $n \geq 3$, which implies $\ln(N) \geq 2$). These inequalities will play a role later.

A well-known multiplicative Chernoff bound states that

$$\Pr[Y > (1+\delta)\mu] < \left(\frac{e^\delta}{(1+\delta)^{1+\delta}}\right)^\mu$$

for all $\delta > 0$, since the Y_i's are independent. As

$$\left(\frac{e^\delta}{(1+\delta)^{1+\delta}}\right)^\mu \leq \left(\frac{e^\delta}{\delta^\delta}\right)^\mu = e^{(1-\ln(\delta))\delta\mu} \leq e^{-\delta\mu}$$

for $\delta \geq e^2$, we thus have

$$\Pr[X > (1+\delta)\mu] < e^{-\delta\mu}$$

for all $\delta \geq e^2$.

Thus

$$N^{2qw}\Pr[X > (1+\delta)\mu] < e^{2qw\ln(N)-2\delta\mu}$$

for $\delta \geq e^2$. Note that

$$\begin{aligned} & 2qw\ln(N) - 2\delta\mu < 0 \\ \iff\ & \delta\mu > qw\ln(N) \\ \Longleftarrow\ & (1+\delta)\mu \geq 2qw\ln(N) \end{aligned}$$

where the last implication uses (41). Thus letting

$$\delta = 2qw\ln(N)/\mu - 1 \geq 2qw\ln(N) - 1$$

we have $\delta \geq e^2$ by (42) and

$$N^{2qw}\Pr[X > 2qw\ln(N)] = N^{2qw}\Pr[X > (1+\delta)\mu] < 1$$

which implies, by a union bound over all

$$\binom{N}{q}^{2w} \leq N^{2qw}$$

possible choices of $U_1, \ldots, U_w, V_2, \ldots, V_w$ the existence of a permutation $\tau : \{0,1\}^{wn} \to \{0,1\}^{wn}$ such that

$$\mathsf{Cond}^{1,+}(\tau, q) \leq 2qw\ln(N).$$

In fact, moreover, a quick revisitation of the computations above shows that our choice of δ not only gives

$$2qw\ln(N) - 2\delta\mu < 0$$

but more strongly gives (using (43))

$$2qw\ln(N) - 2\delta\mu < -\ln(2w)$$

which thus implies

$$N^{2qw}\Pr[X > 2qw\ln(N)] = N^{2qw}\Pr[X > (1+\delta)\mu] < \frac{1}{2w}$$

and, by an additional union bound over $h \in [w]$ and in $\sigma \in \{+,-\}$, the existence of a permutation $\tau : \{0,1\}^{wn} \to \{0,1\}^{wn}$ such that

$$\mathsf{Cond}^{h,\sigma}(\tau, q) \leq 2qw\ln(N)$$

for all $h \in [w]$, $\sigma \in \{+,-\}$, i.e., such that

$$\mathsf{aboCond}(\tau, q) \leq 2qw\ln(N).$$

We have proved:

Theorem 4. *Let $q, n, w \in \mathbb{N}$ be such that $n \geq 3$, $q \leq \sqrt{N}$. Then there exists a permutation $\tau : \{0,1\}^{wn} \to \{0,1\}^{wn}$ such that*

$$\mathsf{aboCond}(\tau) \leq 2qw\ln(N).$$

B Lower Bounds on the Conductance of Linear Permutations

In this section we show that linear permutations cannot achieve conductance as low as random permutations. Specifically, we show that

$$\mathsf{Cond}_\tau(q) \geq \Omega(q^{2-\frac{1}{2w-1}})$$

for a "generic" $\mathbb{F}$-linear transformation $\tau : \mathbb{F}^w \to \mathbb{F}^w$, where $\mathbb{F}$ is a finite field. This lower bound is valid for any $q = o(|\mathbb{F}|)$, and hides a constant factor dependent on w. (The constant factor is roughly 2^{-w^3}.) The linear permutation is "generic" in the sense that our lower bound holds with probability[24] $1 - o(1)$ over the choice of a random matrix $A \in \mathbb{F}^{w\times w}$. We conjecture that our lower bound should carry over to non-generic (i.e., arbitrary) linear permutations as well, but our analysis does not cover this case.

We start by describing the result for $w = 2$, as the general case is a bit unwieldy. When $w = 2$ the linear transformation τ takes the form of a 2×2 matrix

$$A = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix}$$

where $\tau(\mathbf{x}) = A\mathbf{x}$. Or, if $\mathbf{x} = (x_1, x_1)$, $\tau(\mathbf{x}) = \mathbf{y} = (y_1, y_2)$,

$$a_{11}x_1 + a_{12}x_2 = y_1,$$
$$a_{21}x_1 + a_{22}x_2 = y_2.$$

[24] The underlying asymptotic parameter is $|\mathbb{F}|$. I.e., the result has the form "for all functions $q(|\mathbb{F}|) = o(|\mathbb{F}|)$, ...".

The object is to build sets $U_1, U_2, V_1, V_2 \subseteq \mathbb{F}$ such that $\tau(\mathbf{x}) \in V_1 \times V_2$ for many $\mathbf{x} \in U_1 \times U_2$.

To begin with, define

$$\begin{aligned} U_1^* &= \{ia_{12} + ja_{22} : 1 \leq i, j \leq q^{1/3}\}, \\ U_2^* &= \{ka_{11} + ha_{21} : 1 \leq k, h \leq q^{1/3}\}. \end{aligned}$$

(These sets only have size $q^{2/3}$, but later we will extend them to sets of size q.) Note that if

$$ia_{12} + ja_{22} = i'a_{12} + j'a_{22}$$

has a nontrivial solution for some coefficients i, j, i', j' between 1 and $q^{1/3}$ then

$$ia_{12} + ja_{22} = 0$$

has a nontrivial solution for some i and j between $-q^{1/3}$ and $q^{1/3}$. When a_{12} and a_{22} are chosen randomly, the probability of such a solution occurring is upper bounded by $4q^{2/3}/|\mathbb{F}| \ll 1$, since $q = o(|\mathbb{F}|)$. Hence the "syntactically distinct" elements of U_1^* are also distinct elements of $\mathbb{F}$ with high probability (and likewise for U_2^*) with high probability over the choice of a random matrix A.

Note that

$$\begin{aligned} &\{a_{11}x_1 + a_{12}x_2 : \mathbf{x} \in U_1^* \times U_2^*\} \\ &= \{a_{11}a_{12}(i + k) + a_{11}a_{22}j + a_{12}a_{21}h : 1 \leq i, j, k, h \leq q^{1/3}\} \\ &\supseteq \{a_{11}a_{12}i + a_{11}a_{22}j + a_{12}a_{21}h : 1 \leq i, j, h \leq q^{1/3}\}. \end{aligned}$$

Likewise

$$\begin{aligned} &\{a_{21}x_1 + a_{22}x_2 : \mathbf{x} \in U_1 * \times U_2^*\} \\ &= \{a_{21}a_{22}(j + h) + a_{21}a_{12}i + a_{11}a_{22}k : 1 \leq i, j, k, h \leq q^{1/3}\} \\ &\supseteq \{a_{21}a_{22}j + a_{21}a_{12}i + a_{11}a_{22}k : 1 \leq i, j, k \leq q^{1/3}\}. \end{aligned}$$

This motivates defining

$$\begin{aligned} V_1 &= \{a_{11}a_{12}i + a_{11}a_{22}j + a_{12}a_{21}h : 1 \leq i, j, h \leq q^{1/3}\}, \\ V_2 &= \{a_{21}a_{22}j + a_{21}a_{12}i + a_{11}a_{22}k : 1 \leq i, j, k \leq q^{1/3}\}. \end{aligned}$$

Then if $x_1 = ia_{12} + ja_{22} \in U_1^*$, $x_2 = ka_{11} + ha_{21} \in U_2^*$, we have

$$(a_{11}x_1 + a_{12}x_2, a_{21}x_1 + a_{22}x_2) \in V_1 \times V_2$$

if $i + k \leq q^{1/3}$ and $j + h \leq q^{1/3}$. The latter conditions are both implied if $i, j, k, h \leq (q^{1/3})/2$, so

$$|\{\mathbf{x} \in U_1^* \times U_2^* : \tau(\mathbf{x}) \in V_1 \times V_2\}| \geq (q^{1/3}/2)^4 = \Omega(q^{\frac{4}{3}}).$$

I.e., $\mathsf{Cond}_\tau(q) \geq \Omega(q^{\frac{4}{3}})$.

As already pointed out, however, this construction has $|U_1^*| = |U_2^*| = q^{2/3} \ll q$, which suggests that even higher conductance could conceivably be shown. For this, our idea is to replace U_1^* and U_2^* by sets of the form $U_1^* + R$, $U_2^* + S$ where R and S are arithmetic progressions of length $q^{1/3}$. The step sizes for R and S will be different. Details follow.

Let $\mathbf{b} = (b_1, b_2)$ be the first column of A's inverse. In other words

$$\begin{aligned} a_{11}b_1 + a_{12}b_2 &= 1, \\ a_{21}b_1 + a_{22}b_2 &= 0. \end{aligned}$$

We let

$$U_1 = \{rb_1a_{11}a_{22} + ia_{12} + ja_{22} : 1 \le r, i, j \le q^{1/3}\},$$
$$U_2 = \{rb_2a_{11}a_{22} + ka_{11} + ha_{21} : 1 \le r, k, h \le q^{1/3}\}.$$

A similar argument as above shows (using $q = o(|\mathbb{F}|)$) that syntactically distinct elements of U_i are also distinct as field elements with high probability over the choice of A.

Let

$$r, i, j, k, h$$

be arbitrary values between 1 and $(q^{1/3})/2$ and set

$$x_1 = rb_1a_{11}a_{22} + ia_{12} + ja_{22}$$
$$x_2 = rb_2a_{11}a_{22} + ia_{11} + ka_{21}$$

so that $\mathbf{x} = (x_1, x_2) \in U_1 \times U_2$. (Note that we are using the "same r" for both x_1 and x_2; hence, only a negligible fraction of vectors in $U_1 \times U_2$ have this form.)

If we let $\tau(\mathbf{x}) = \mathbf{y} = (y_1, y_2)$ then it is easy to see from the fact that $\mathbf{b}$ is the first column of A's inverse that

$$y_1 = a_{11}a_{12}(i + k) + a_{11}a_{22}(j + r) + a_{12}a_{21}h,$$
$$y_2 = a_{21}a_{22}(j + h) + a_{21}a_{12}i + a_{11}a_{22}k.$$

Hence $\mathbf{y} \in V_1 \times V_2$ (the definition of the sets V_1, V_2 hasn't changed), by our stipulation that $1 \le r, i, j, k, h \le (q^{1/3})/2$. Hence

$$\mathsf{Cond}_\tau(q) \ge ((q^{1/3})/2)^5 = \Omega(q^{5/3}) = \Omega(q^{2-1/(2w-1)})$$

as claimed.

We note that the constant $a_{11}a_{22}$, appearing in the definition of U_1, U_2, is somewhat arbitrary and could be replaced with $a_{12}a_{21}$ or even $a_{11}a_{12}$ or $a_{21}a_{22}$ without (much) damaging the bound. (In more detail, choosing $a_{11}a_{12}$ or $a_{21}a_{22}$ means that r, i, j, k, h should be restricted to be less than $(q^{1/3})/3$ instead of $(q^{1/3})/2$, which is obviously a small loss.)

In the general case, the constant $a_{11}a_{22}$ is replaced by a collection of constants each of the form $a_{1j}a_{xy}$ with $x \neq 1$ and $y \neq j$, with the arithmetic progression $\{rb_ia_{11}a_{22} : 1 \le r \le q^{1/3}\}$ being replaced by a higher-dimensional arithmetic progression, with each such constant used for one dimension of the arithmetic progression.

General case. Let $\tau(\mathbf{x}) = A\mathbf{x}$ where $A \in \mathbb{F}^{w \times w}$ is a $w \times w$ matrix with entries in $\mathbb{F}$, and let a_{ij} be the (i, j)-th entry of A.

Let $m = w(w-1)^2 + w(w-1)/2$. Let $B_1, \dots, B_w$ be subsets of $[w]\backslash\{1\} \times [w]$ such that $B_j \subseteq [w]\backslash\{1\} \times [w]\backslash\{j\}$ for each $1 \le j \le w$ and such that

$$\sum_j |B_j| = m - w(w-1) = w(w-1)^2 - w(w-1)/2.$$

We write $xy \in B_j$ as a shorthand for $(x, y) \in B_j$. [For $w = 2$ we had $B_1 = (2, 2)$, $B_2 = \emptyset$.]

Let $\mathbf{b} = (b_1, \dots, b_w)$ be the first column of A's inverse.

We define

$$U_i = \{\sum_j \sum_{xy \in B_j} r_{jxy}b_ia_{1j}a_{xy} + \sum_{i'} \sum_{j \neq i} h_{i'j}a_{i'j} : 1 \le r_{jxy} \le q^{1/m}, 1 \le h_{i'j} \le q^{1/m}\}$$

for $1 \le i \le w$. Then $|U_i| \le q$. To argue distinctness of elements in U_i, note that the equation

$$\sum_j \sum_{xy \in B_j} r_{jxy} b_i a_{1j} a_{xy} + \sum_{i'} \sum_{j \ne i} h_{i'j} a_{i'j} = \sum_j \sum_{xy \in B_j} r^*_{jxy} b_i a_{1j} a_{xy} + \sum_{i'} \sum_{j \ne i} h^*_{i'j} a_{i'j}$$

has a nontrivial solution with coefficients r_{jxy}, r^*_{jxy}, $h_{i'j}$, $h^*_{i'j}$ between 1 and $q^{1/m}$ if and only if the equation

$$\sum_j \sum_{xy \in B_j} r_{jxy} b_i a_{1j} a_{xy} + \sum_{i'} \sum_{j \ne i} h_{i',j} a_{i'j} = 0$$

has a nontrivial solution with coefficients between $-q^{1/m}$ and $q^{1/m}$. By a union bound, the probability of such a solution existing for a randomly chosen matrix A is at most $(2q^{1/m})^{(m-w(w-1))+w(w-1)}/|\mathbb{F}| = 2^m q/|\mathbb{F}| = o(1)$ if $q = o(|\mathbb{F}|)$. Hence $|U_i| = q$ for all i with high probability over A.

We also define

$$V_i = \{\sum_{j<j'} g_{jj'} a_{ij} a_{ij'} + \sum_{i' \ne i} \sum_{j \ne j'} h_{i'jj'} a_{ij} a_{i'j'} : 1 \le g_{jj'}, h_{i'jj'} \le q^{1/m}\}.$$

Since the number of distinct choices for the $g_{jj'}$'s is $(q^{1/m})^{w(w-1)/2}$ and the number of distinct choices for the $h_{i'jj'}$'s is $(q^{1/m})^{(w-1)w(w-1)}$, we have $|V_i| \le q$ by the fact that $m = w(w-1)/2 + w(w-1)^2$.

Let

$$\{r_{zxy} : xy \in B_z\}, \{h^i_{i'j} : j \ne i\}$$

be values such that $1 \le r_{zxy} \le (q^{1/m})/2$ for all z, x, y such that $xy \in B_z$ and such that $1 \le h^i_{i'j} \le (q^{1/m})/2$ for all i, i', j such that $j \ne i$. We note there are

$$\begin{aligned}
(q^{1/m}/2)^{m-w(w-1)} \cdot (q^{1/m}/2)^{w^2(w-1)} &= \Omega(1) \cdot q^{[m-w(w-1)+w^2(w-1)]/m} \\
&= \Omega(1) \cdot q^{1+w(w-1)^2/m} \\
&= \Omega(1) \cdot q^{1+[m-(m-w(w-1)^2)]/m} \\
&= \Omega(1) \cdot q^{2-w(w-1)/2m} \\
&= \Omega(1) \cdot q^{2-1/(2w-1)}
\end{aligned}$$

such tuples. Moreover, each such tuple naturally corresponds to a distinct element $\mathbf{x}$ in the direct product $U_1 \times \cdots \times U_w$, which is namely the vector $\mathbf{x}$ such that

$$\mathbf{x}[i] = \sum_z \sum_{xy \in B_z} r_{zxy} b_i a_{1z} a_{xy} + \sum_{i'} \sum_{j \ne i} h^i_{i'j} a_{i'j}.$$

Then

$$\begin{aligned}
\tau(\mathbf{x})[i] \ = \ (A\mathbf{x})[i] &= \sum_k a_{ik} \mathbf{x}[k] \\
&= \sum_k a_{ik} \left(\sum_z \sum_{xy \in B_z} r_{zxy} b_k a_{1z} a_{xy} + \sum_{i'} \sum_{j \ne k} h^k_{i'j} a_{i'j} \right) \\
&= \sum_z \sum_{xy \in B_z} r_{zxy} a_{1z} a_{xy} \sum_k a_{ik} b_k + \sum_{i'} \sum_{j \ne k} h^k_{i'j} a_{ik} a_{i'j}
\end{aligned}$$

(Note the symbol $\sum_{j \ne k}$ in the second line does not mean the same as the symbol $\sum_{j \ne k}$ in the third line, since in the former sum k is already quantified, and the sum only takes place over j, whereas in the latter sum summation takes place over both j and k.) Here one can note that for each z, $xy \in B_z$,

$$r_{zxy}a_{1z}a_{xy}\sum_k a_{ik}b_k = \begin{cases} r_{zxy}a_{1z}a_{xy} & \text{if } i = 1 \\ 0 & \text{otherwise} \end{cases}$$

since $\mathbf{b} = (b_1, \dots, b_w)$ is the first column of A's inverse. Thus we can write $\tau(\mathbf{x})[i]$ as a sum

$$\tau(\mathbf{x})[i] = \sum_{i' \neq i}\sum_{j \neq k} c_{iji'k}a_{ik}a_{i'j} + \sum_{k<j} c_{ikj}a_{ik}a_{ij}$$

for some coefficients $c_{iki'j}$, $i' \neq i$, $j \neq k$, as well as some coefficients c_{ikj}, $k < j$. More precisely, one has

$$c_{ikj} = h^k_{ij} + h^j_{ik}$$

for all $k < j$, and

$$c_{iki'j} = \begin{cases} h^k_{i'j} & \text{if } i \neq 1 \\ h^k_{i'j} & \text{if } i = 1, i'j \notin B_k \\ h^k_{i'j} + r_{ki'j} & \text{if } i = 1, i'j \in B_k \end{cases}$$

for all $i' \neq i$ and all $j \neq k$.

Since all the indices r_{zxy} and $h^i_{i'j}$ are between 1 and $(q^{1/m})/2$, we thus find that all the coefficients $c_{iki'j}$ as well as c_{ikj} are between 1 and $q^{1/m}$, and hence that $\tau(\mathbf{x})[i] \in V_i$ and that $\tau(\mathbf{x}) \in V_1 \times \cdots \times V_w$. The claim on the conductance of τ then follows from the fact (argued above) that there are $\Omega(1) \cdot q^{2-1/(2w-1)}$ such $\mathbf{x}$'s.

All-but-one conductance. In the same vein, we could also show that all-but-one conductance is at least $q^{2+(w-2)/(w-1)(2w-3)}$ for generic linear permutations. For the sake of brevity, however, and since all-but-one conductance does not seem as intrinsically interesting as conductance, we skip this construction.

C Explicit constructions of permutations with low values of MaxColl

In this appendix we show an explicit construction of a permutation with small entry-wise random collision resistance of $O(w)$. Moreover we show this is essentially tight by proving entry-wise random collision resistance cannot is at least $\Omega(w)$ for any permutation π. (For the latter see Theorem 7 below.)

More specifically, our construction consists of an explicit permutation $\pi : \mathbb{F}^w \to \mathbb{F}^w$ such that $\mathsf{MaxColl}(\pi) \leq 2w$, for a sufficiently large finite field $\mathbb{F}$. (The “sufficiently large” is needed in order for a matrix with certain generic features to exist. In fact a field of size $O(w^2)$ is already sufficient.) Note that if $\pi : \mathbb{F}^w \to \mathbb{F}^w$ is a linear permutation whose associated matrix has no zero entries, then $\mathsf{MaxPreim}(\pi) = 1$, which is optimal. Hence, achieving small entry-wise random preimage resistance is trivial. Nonetheless, since we are generally interested in permutations that *simultaneously* achieve small entry-wise random preimage resistance and small entry-wise random collision resistance, we provide an analysis of the entry-wise random preimage resistance as well.

The Construction. Let A be a $w \times w$ invertible matrix over a finite field $\mathbb{F}$ such that A has no zero entries and such that such that A^{-1} has no zero entries in its first column. By Cramer's rule for the computation of the inverse, the latter condition is equivalent to $\det(A_{1k}) \neq 0$ for all $k \in [w]$, where A_{1k} denote the $(w-1) \times (w-1)$ matrix obtained by deleting the first row and j-th column of A. Clearly a random $w \times w$ matrix A over $\mathbb{F}$ satisfies these requirements with high probability if $\mathbb{F}$ is sufficiently large.

We also let

$$D = (d_2, \ldots, d_w)$$

be an increasing sequence of positive integers, none of which is divisible by the characteristic p of $\mathbb{F}$, and such that $d_2 \geq 2$. (E.g., $(d_2, \ldots, d_w) = (3, 5, \ldots, 2w-1)$ if $p = 2$.) The matrix A and the sequence D will be the parameters of the construction.[25]

Let $\sigma : \mathbb{F}^w \to \mathbb{F}^w$ be defined by

$$\sigma(\mathbf{x}) = A\mathbf{x}$$

for $\mathbf{x} = (x_1, \ldots, x_w) \in \mathbb{F}^w$. Also let $\eta : \mathbb{F}^2 \to \mathbb{F}^w$ be the Feistel-round-like transformation given by

$$\eta(\mathbf{x})[i] = \begin{cases} x_i + \sum_{j=2}^{w} x_j^{d_j} & \text{if } i = 1, \\ x_i & \text{if } i \neq 1. \end{cases}$$

We note that η has a straightforward inverse:

$$\eta^{-1}(\mathbf{x})[i] = \begin{cases} x_i - \sum_{j=2}^{w} x_j^{d_j} & \text{if } i = 1, \\ x_i & \text{if } i \neq 1. \end{cases}$$

We define the permutation $\pi[A, D] : \mathbb{F}^w \to \mathbb{F}^w$ as

$$\pi[A, D] = \sigma^{-1} \circ \eta \circ \sigma$$

with inverse

$$\pi[A, D]^{-1} = \sigma^{-1} \circ \eta^{-1} \circ \sigma.$$

Let

$$z(\mathbf{x}) = \sum_{j=2}^{w} x_j^{d_j}$$

be the polynomial appearing in the definition of η, and let $Z : \mathbb{F}^w \to \mathbb{F}^w$ be defined by

$$Z(\mathbf{x}) = \eta(\mathbf{x}) - \mathbf{x}.$$

Then $Z(\mathbf{x})[1] = z(\mathbf{x})$ and $Z(\mathbf{x})[i] = 0$ for $i \geq 2$. Moreover we can write $\eta(\mathbf{x})$ as $Z(\mathbf{x}) + \mathbf{x}$. Thus

$$(\eta \circ \sigma)(\mathbf{x}) = A\mathbf{x} + Z(A\mathbf{x}).$$

Thus if $\mathbf{a}_i^T$ denotes the i-th row of A and $\mathbf{b}_i^T$ denotes the i-th row of A^{-1}, we have

$$\begin{aligned}(\sigma^{-1} \circ \eta \circ \sigma)(\mathbf{x})[i] &= (A^{-1}A\mathbf{x})[i] + A^{-1}Z(A\mathbf{x})[i] \\ &= x_i + \mathbf{b}_i^T Z(A\mathbf{x}) \\ &= x_i + \mathbf{b}_i[1] z(A\mathbf{x}) \\ &= x_i + \mathbf{b}_i[1] \sum_{j=2}^{w} (\mathbf{a}_j^T \mathbf{x})^{d_j}\end{aligned}$$

[25] We believe that the restriction $d_2 \geq 2$ can be replaced with $d_2 \geq 1$ with some extra technical (but annoying) legwork. If this restriction were lifted one could use $(d_2, \ldots, d_w) = (1, 3, \ldots, 2w-3)$ for $p = 2$.

for all $1 \leq i \leq w$.

The analysis. We start with entry-wise random collision resistance (the more interesting of the two properties), and follow with entry-wise random preimage resistance.

Theorem 5. *$\mathsf{MaxColl}_{\pi[A,D]} \leq d_w$ for A, D and $\pi[A, D]$ as described above.*

Proof. Fix values $k, \ell \in [w]$. Let

$$\begin{aligned} \mathbf{x}' &= (x'_1, \ldots, x'_{k-1}, x'_{k+1}, \ldots x'_k) \\ \mathbf{x}'' &= (x''_1, \ldots, x''_{k-1}, x''_{k+1}, \ldots, x''_k) \end{aligned}$$

be two arbitrary $(w-1)$-dimensional vectors such that $\mathbf{x}' \neq \mathbf{x}''$. We will also let $\mathbf{x}$, $\mathbf{x}'$ stand for functions from $\mathbb{F}$ to $\mathbb{F}^w$ by setting

$$\begin{aligned} \mathbf{x}'(x_k) &= (x'_1, \ldots, x'_{k-1}, x_k, x'_{k+1}, \ldots x'_k) \\ \mathbf{x}''(x_k) &= (x''_1, \ldots, x''_{k-1}, x_k, x''_{k+1}, \ldots x''_k) \end{aligned}$$

for $x_k \in \mathbb{F}$. Then

$$\begin{aligned} P(x_k) &:= (\sigma^{-1} \circ \eta \circ \sigma)(\mathbf{x}'(x_k))[\ell] - (\sigma^{-1} \circ \eta \circ \sigma)(\mathbf{x}''(x_k))[\ell] \\ &= \begin{cases} x'_\ell - x''_\ell + \mathbf{b}_\ell[1](z(A\mathbf{x}'(x_k)) - z(A\mathbf{x}''(x_k))) & \text{if } \ell \neq k \\ \mathbf{b}_\ell[1](z(A\mathbf{x}'(x_k)) - z(A\mathbf{x}''(x_k))) & \text{if } \ell = k \end{cases} \end{aligned}$$

is a polynomial of degree (at most) d_w in x_k.

We assume next by contradiction that $P(x_k) = 0$ is the zero polynomial. We consider the cases $\ell = k$ and $\ell \neq k$ separately.

For $\ell \neq k$ we have

$$P(x_k) = x'_\ell - x''_\ell + \mathbf{b}_\ell[1] \left(\sum_{j=2}^{w} (\mathbf{a}_j^T \mathbf{x}'(x_k))^{d_j} - \sum_{j=2}^{w} (\mathbf{a}_j^T \mathbf{x}''(x_k))^{d_j} \right).$$

We recall that $\mathbf{b}_i[1] \neq 0$ for all i by our initial assumptions on A. Writing $\mathbf{a}_{j\backslash k}^T$ for the j-th row of A with the k-th entry removed we can write

$$P(x_k) = x'_\ell - x''_\ell + \mathbf{b}_\ell[1] \left(\sum_{j=2}^{w} (\mathbf{a}_{j\backslash k}^T \mathbf{x}' + \mathbf{a}_j[k]x_k)^{d_j} - \sum_{j=2}^{w} (\mathbf{a}_{j\backslash k}^T \mathbf{x}'' + \mathbf{a}_j[k]x_k)^{d_j} \right).$$

Since $P(x_k) = 0$ by assumption, the coefficient of $x_k^{d_w - 1}$ in $P(x_k)$ must be zero, i.e.,

$$d_w(\mathbf{a}_{w\backslash k}^T \mathbf{x}')\mathbf{a}_w[k]^{d_w - 1} = d_w(\mathbf{a}_{w\backslash k}^T \mathbf{x}'')\mathbf{a}_w[k]^{d_w - 1}.$$

Since the characteristic of $\mathbb{F}$ doesn't divide d_w, and since $\mathbf{a}_w[k] \neq 0$, this implies

$$\mathbf{a}_{w\backslash k}^T \mathbf{x}' = \mathbf{a}_{w\backslash k}^T \mathbf{x}''$$

whence

$$P(x_k) = x'_\ell - x''_\ell + \mathbf{b}_\ell[1] \left(\sum_{j=2}^{w-1} (\mathbf{a}_{j\backslash k}^T \mathbf{x}' + \mathbf{a}_j[k]x_k)^{d_j} - \sum_{j=2}^{w-1} (\mathbf{a}_{j\backslash k}^T \mathbf{x}'' + \mathbf{a}_j[k]x_k)^{d_j} \right).$$

Since the coefficient of $x_k^{d_{w-1}-1}$ in $P(x_k)$ must also be zero, we then find

$$d_{w-1}(\mathbf{a}_{w-1\backslash k}^T\mathbf{x}')\mathbf{a}_{w-1}[k]^{d_{w-1}-1} = d_{w-1}(\mathbf{a}_{w-1\backslash k}^T\mathbf{x}'')\mathbf{a}_{w-1}[k]^{d_{w-1}-1}$$

and (since $p \not| \, d_{w-1}$, $\mathbf{a}_{w-1}[k] \neq 0$)

$$\mathbf{a}_{w-1\backslash k}^T\mathbf{x}' = \mathbf{a}_{w-1\backslash k}^T\mathbf{x}''$$

whence

$$P(x_k) = x'_\ell - x''_\ell + \mathbf{b}_\ell[1]\left(\sum_{j=2}^{w-2}(\mathbf{a}_{j\backslash k}^T\mathbf{x}' + \mathbf{a}_j[k]x_k)^{d_j} - \sum_{j=2}^{w-2}(\mathbf{a}_{j\backslash k}^T\mathbf{x}'' + \mathbf{a}_j[k]x_k)^{d_j}\right).$$

Continuing like this, we eventually find that

$$P(x_k) = x'_\ell - x''_\ell + \mathbf{b}_1[\ell]\left((\mathbf{a}_{2\backslash k}^T\mathbf{x}' + \mathbf{a}_2[k]x_k)^{d_2} - (\mathbf{a}_{2\backslash k}^T\mathbf{x}'' + \mathbf{a}_2[k]x_k)^{d_2}\right)$$

and that

$$\mathbf{a}_{2\backslash k}^T\mathbf{x}' = \mathbf{a}_{2\backslash k}^T\mathbf{x}''$$

by considering the term $x_k^{d_2-1}$ in $P(x_k)$. (Here we use the fact that $d_2 > 1$.) Hence

$$\mathbf{a}_{j\backslash k}^T\mathbf{x}' = \mathbf{a}_{j\backslash k}^T\mathbf{x}'' \tag{44}$$

for all $2 \leq j \leq w$, i.e.,

$$A_{1k}\mathbf{x}' = A_{1k}\mathbf{x}'',$$

but this implies $\mathbf{x}' = \mathbf{x}''$, a contradiction, since $\det(A_{1k}) \neq 0$.

To establish a contradiction for the case $\ell = k$ one can reason analogously. In fact this case is simpler, since the only thing that changes is that $P(x_k)$ doesn't contain the term $x'_\ell - x''_\ell$. (As a technical comment, one can observe that for the case $\ell = k$ it suffices to have $d_2 \geq 1$ instead of $d_2 \geq 2$. For in the absence of the constant term $x'_\ell - x''_\ell$ one arrives at equation (44) even if $d_2 - 1 = 0$.)

Having obtained a contradiction in both cases (being $\ell = k$ and $\ell \neq k$), we can conclude that $P(x_k)$ is a nonzero polynomial. Since $P(x_k)$ has degree at most d_w, however, this implies that

$$(\sigma^{-1} \circ \eta \circ \sigma)(\mathbf{x}'(x_k)) = (\sigma^{-1} \circ \eta \circ \sigma)(\mathbf{x}''(x_k))$$

for at most d_w values of $x_k \in \mathbb{F}$. This precisely implies that $\mathsf{MaxColl}_{\pi[A,D]} \leq d_w$ since k, ℓ as well as $\mathbf{x}'$ and $\mathbf{x}''$ were arbitrary. □

Theorem 6. $\mathsf{MaxPreim}_{\pi[A,D]} \leq d_w$ *for* A, D *and* $\pi[A,D]$ *as described above.*

Proof. Fix values $k, \ell \in [w]$ as well as a vector

$$\mathbf{x}' = (x'_1, \ldots, x'_{k-1}, x'_{k+1}, \ldots, x'_w) \in \mathbb{F}^{w-1}.$$

We also set

$$\mathbf{x}'(x_k) = (x'_1, \ldots, x'_{k-1}, x_k, x'_{k+1}, \ldots, x'_w) \in \mathbb{F}^{w-1}$$

as in the proof of Theorem 5. Then

$$\pi[A,D](\mathbf{x}'(x_k))[\ell] = \begin{cases} x'_\ell + \mathbf{b}_\ell[1]z(A\mathbf{x}'(x_k)) & \text{if } \ell \neq k, \\ x_k + \mathbf{b}_k[1]z(A\mathbf{x}'(x_k)) & \text{if } \ell = k. \end{cases}$$

In either case (whether $\ell = k$ or $\ell \neq k$) $\pi[A,D](\mathbf{x}'(x_k))[\ell]$ is a nonzero polynomial in x_k of degree d_w. (The nonzeroness can be seen by focusing on the term of degree d_w, which has nonzero coefficient of $\mathbf{a}_w^T[k]^{d_w}$.) Hence the probability that $\pi[A,D](\mathbf{x}'(x_k))$ equals any particular value, over random choice of x_k, is at most d_w. □

Optimality. We can also prove that every permutation π has $\mathsf{MaxColl}(\pi) = \Omega(w)$. Specifically, we can prove the following theorem:

Theorem 7. *For all sufficiently large $|\mathbb{F}|$, any permutation $\pi : |\mathbb{F}|^w \to |\mathbb{F}|^w$ has $\mathsf{MaxColl}(\pi) \geq (w-1)/2$.*

For this universal lower bound, the algebraic structure of $\mathbb{F}$ obviously doesn't matter. Here $\mathbb{F}$ can be considered as a placeholder for an arbitrary finite set.

In fact we will prove something a bit stronger than Theorem 7, being namely that

$$\mathsf{MaxColl}_{j,h}(\pi) \geq (w-1)/2$$

for all $j, h \in [w]$ (and sufficiently large $|\mathbb{F}|$), where

$$\mathsf{MaxColl}_{j,h}(\pi) := \max_{\mathbf{x},\mathbf{x}'} \bigl|\{x \in \mathbb{F} : \pi^{\mathbf{x}}_{j,h}(x) = \pi^{\mathbf{x}'}_{j,h}(x)\}\bigr|$$

where $\pi^{\mathbf{x}}_{j,h}(x)$ is as defined in Section 2. We can obviously assume that $j = h = 1$.

Let $q = |\mathbb{F}|$. We will analyze the family of functions $\{\pi^{\mathbf{x}}_{1,1} : \mathbf{x} \in \mathbb{F}^w, \mathbf{x}[1] = 0\}$. This family is indexed by q^{w-1} different $\mathbf{x}$'s, which we will denote $\mathbf{x}_1, \ldots, \mathbf{x}_{q^{w-1}}$.

Each function $\pi^{\mathbf{x}_i}_{1,1} : \mathbb{F} \to \mathbb{F}$ can be encoded as a $q \times q$ 0-1 matrix β_i by setting

$$\beta_i[x,y] = \begin{cases} 1 & \text{if } \pi^{\mathbf{x}_i}_{1,1}(x) = y, \\ 0 & \text{otherwise.} \end{cases}$$

Then $\mathsf{MaxColl}_{1,1}(\pi)$ can be equivalently defined as the maximum dot product (over $\mathbb{R}$) between any two (distinct) β_i's. Thus, Theorem 7 will follow by the following proposition (which further relaxes[26] the problem by removing the requirement that the β_i's encode functions):

Proposition 1. *Let $B = \{\beta_1, \ldots, \beta_{q^{w-1}}\}$ be distinct 0-1 matrices of size $q \times q$. Then if q is sufficiently large compared to w there exists some $i \neq i'$ such that $\langle \beta_i, \beta_{i'} \rangle \geq (w-1)/2$ where $\langle x, y \rangle$ denotes the real-valued dot product of x and y.*

Proof. A family B that minimizes the maximum value of $\langle \beta_i, \beta_j \rangle$, $i \neq j$ can obviously be chosen such that deleting any entry of '1' from any vector β_i produces another vector in B (or else replace β_i with the new vector). We will thus assume that B has this form. Then

$$\max_{i \neq j} \langle \beta_i, \beta_j \rangle \geq \max_i \|\beta_i\|_1 - 1$$

where $\|\cdot\|_1$ denotes the 1-norm. (In fact this inequality holds with equality, but we only need the inequality.)

Let $d = \max_i \|\beta_i\|_1$. Each element in B can then be specified by a sequence of d choices, where each choice has arity $q^2 + 1$ (where to put the next '1', and if it should be a '1' at all), so

$$q^{w-1} = |B| \leq (1 + q^2)^d$$

which obviously implies that $d \geq (w-1)/2$ for sufficiently large q, for fixed w. □

[26] Well, not quite: Proposition 1 introduces the new requirement that β_i's be distinct, but this is obviously without loss of generality since otherwise $\mathsf{MaxColl}_{1,1}(\pi) = \mathsf{MaxColl}(\pi) = |\mathbb{F}|$.

```
Game $G_1$ $G_2$, $G_3$

random tapes: $p_{A_1}, \ldots, p_{L_w}$, $p_Z$

global static: XtraOuterRnd, XtraMiddleRnd, XtraUntglRnds

    if $Table(x) \neq \perp$ then abort
    if $Table^{-1}(y) \neq \perp$ then abort
    $Table(x) \leftarrow y$
    $Table^{-1}(y) \leftarrow x$

private procedure ReadTape($Table, x, p$)
    $y \leftarrow p(x)$
    SetTable($Table, x, y$)
    return $y$

public procedure Z($\mathbf{x}$)
    if $P_Z(\mathbf{x}) = \perp$ then
        $\mathbf{y} \leftarrow$ ReadTape($P_Z, \mathbf{x}, p_Z$)
        if $\exists j, Y_j^{-1}(\mathbf{y}[j]) \neq \perp$ then abort // $G_3$
    return $P_Z(\mathbf{x})$

public procedure Z$^{-1}$($\mathbf{y}$)
    if $P_Z^{-1}(\mathbf{y}) = \perp$ then
        $\mathbf{x} \leftarrow$ ReadTape($P_Z^{-1}, \mathbf{y}, p_J^{-1}$)
        if $\exists j, X_j(\mathbf{x}[j]) \neq \perp$ then abort // $G_3$
    return $P_Z^{-1}(\mathbf{y})$

private procedure CheckZ($\mathbf{x}, \mathbf{y}$)
    return $P_Z(\mathbf{x}) = \mathbf{y}$ // $G_2$, $G_3$
    return $Z(\mathbf{x}) = \mathbf{y}$

private procedure BlockDefined($T, \mathbf{z}, \sigma$)
    forall $k \in \{1, \ldots, w\}$ do
        if $(T_k^\sigma(\mathbf{z}[k]) = \perp \wedge$
            $\mathbf{z}[k] \notin$ ToBeAssigned$_{T_k}^\sigma$) return false
    return true
```

```
public procedure D($j, x$)
    if $D_j(x) = \perp$ then ReadTape($D_j, x, p_{D_j}$)
    return $D_j(x)$

public procedure D$^{-1}$($j, y$)
    if $D_j^{-1}(y) = \perp$ then ReadTape($D_j^{-1}, y, p_{D_j}^{-1}$)
    return $D_j^{-1}(y)$

public procedure E($j, x$)
    if $E_j(x) = \perp$ then ReadTape($E_j, x, p_{E_j}$)
    return $E_j(x)$

public procedure E$^{-1}$($j, y$)
    if $E_j^{-1}(y) = \perp$ then ReadTape($E_j^{-1}, y, p_{E_j}^{-1}$)
    return $E_j^{-1}(y)$

public procedure I($j, x$)
    if $I_j(x) = \perp$ then ReadTape($I_j, x, p_{I_j}$)
    return $I_j(x)$

public procedure I$^{-1}$($j, y$)
    if $I_j^{-1}(y) = \perp$ then ReadTape($I_j^{-1}, y, p_{I_j}^{-1}$)
    return $I_j^{-1}(y)$

public procedure J($j, x$)
    if $J_j(x) = \perp$ then ReadTape($J_j, x, p_{J_j}$)
    return $J_j(x)$

public procedure J$^{-1}$($j, y$)
    if $J_j^{-1}(y) = \perp$ then ReadTape($J_j^{-1}, y, p_{J_j}^{-1}$)
    return $J_j^{-1}(y)$

public procedure K($j, x$)
    if $K_j(x) = \perp$ then ReadTape($K_j, x, p_{K_j}$)
    return $K_j(x)$

public procedure K$^{-1}$($j, y$)
    if $K_j^{-1}(y) = \perp$ then ReadTape($K_j^{-1}, y, p_{K_j}^{-1}$)
    return $K_j^{-1}(y)$

public procedure L($j, x$)
    if $L_j(x) = \perp$ then ReadTape($L_j, x, p_{L_j}$)
    return $L_j(x)$

public procedure L$^{-1}$($j, y$)
    if $L_j^{-1}(y) = \perp$ then ReadTape($L_j^{-1}, y, p_{L_j}^{-1}$)
    return $L_j^{-1}(y)$
```

Fig. 2. Games G_1 through G_3, first set of procedures. Statements commented by // G_2 or // G_2, G_3 (in red) appear only in G_2 or in G_2 and G_3, respectively. The simulator is implemented by game G_1, excepted procedures Z and Z^{-1} (to which the simulator only has oracle access).

```
private procedure BackwardOutside($T, j, y$)
    $x \leftarrow$ ReadTape($T_j^{-1}, y, p_{T_j}^{-1}$)
    if ($\exists \mathbf{x}$ s.t. $\mathbf{x}[j] = x \wedge$
        $P_Z(\mathbf{x}) \neq \perp$) then abort // G$_3$

public procedure F$^{-1}(j, y)$
    if $F_j^{-1}(y) = \perp$ then
        BackwardOutside($F, j, y$)
    return $F_j^{-1}(y)$

public procedure F$(j, x)$
    if $F_j(x) = \perp$ then
        $y \leftarrow$ ReadTape($F_j, x, p_{F_j}$)
        if ($\exists \mathbf{y}$ s.t. $\mathbf{y}[j] = y \wedge$
            BlockDefined($F, \mathbf{y}, -$) $\wedge$
            BlockDefined($G, \nu(\mathbf{y}), +$)) then abort
    return $F_j(x)$

public procedure G$^{-1}(j, y)$
    if $G_j^{-1}(y) = \perp$ then
        if XtraOuterRnd then
            $x \leftarrow$ ReadTape($G_j^{-1}, y, p_{G_j}^{-1}$)
            if ($\exists \mathbf{x}$ s.t. $\mathbf{x}[j] = x \wedge$
                BlockDefined($G, \mathbf{x}, +$) $\wedge$
                BlockDefined($F, \nu^-(\mathbf{x}), -$)
                ) then abort
        else
            BackwardOutside($G, j, y$)
    return $G_j^{-1}(y)$

public procedure G$(j, x)$
    RequestG($j, x$)
    AdaptLeft()
    return $G_j(x)$

public procedure B$(j, x)$
    if $B_j(x) = \perp$ then
        $y \leftarrow$ ReadTape($B_j, x, p_{B_j}$)
        if ($\exists \mathbf{y}$ s.t. $\mathbf{y}[j] = y \wedge$
            BlockDefined($B, \mathbf{y}, -$) $\wedge$
            BlockDefined($C, \tau(\mathbf{y}), +$)) then abort
    return $B_j(x)$

public procedure B$^{-1}(j, y)$
    RequestU$^{-1}(j, y)$
    AdaptLeft()
    return $B_j^{-1}(y)$

public procedure A$^{-1}(j, y)$
    RequestU$^{-1}(j, y)$
    AdaptLeft()
    return $A_j^{-1}(y)$
```

```
private procedure ForwardOutside($T, j, x$)
    $y \leftarrow$ ReadTape($T_j, x, p_{T_j}$)
    if ($\exists \mathbf{y}$ s.t. $\mathbf{y}[j] = y \wedge$
        $P_Z^{-1}(\mathbf{y}) \neq \perp$) then abort // G$_3$

public procedure H$(j, x)$
    if $H(x) = \perp$ then
        ForwardOutside($H, j, x$)
    return $H_j(x)$

public procedure H$^{-1}(j, y)$
    RequestH$^{-1}(j, y)$
    AdaptRight()
    return $H_j^{-1}(y)$

public procedure C$^{-1}(j, y)$
    if $C_j^{-1}(y) = \perp$ then
        $x \leftarrow$ ReadTape($C_j^{-1}, y, p_{C_j}^{-1}$)
        if ($\exists \mathbf{x}$ s.t. $\mathbf{x}[j] = x \wedge$
            BlockDefined($C, \mathbf{x}, +$) $\wedge$
            BlockDefined($B, \tau^-(\mathbf{x}), -$)) then abort
    return $C_j^{-1}(y)$

public procedure C$(j, x)$
    RequestV($j, x$)
    AdaptRight()
    return $C_j(x)$

public procedure A$(j, x)$
    RequestV($j, x$)
    AdaptRight()
    return $A_j(x)$
```

Fig. 3. Games G$_1$ through G$_3$, second set of procedures.

```
private procedure RequestG$(j, x)$
    if $G_j(x) \neq \bot$ then return
    if $x \in \text{ToBeAssigned}_{G_j}$ then return
    $\text{ToBeAssigned}_{G_j} \leftarrow \text{ToBeAssigned}_{G_j} \cup \{x\}$
    $\text{NewYUs} \leftarrow \emptyset$  // local var
    forall $\mathbf{x}^G, \mathbf{y}^H$ s.t. $\mathbf{x}^G[j] = x$ do
        if $\exists \mathbf{y}^U$ s.t. $(\mathbf{x}^G, \mathbf{y}^U) \in \text{ToBeAdapted}_D$ continue
        if $(\text{BlockDefined}(G, \mathbf{x}^G, +) = \textbf{false})$ continue
        if $(\text{BlockDefined}(H, \mathbf{y}^H, -) = \textbf{false})$ continue
        if XtraOuterRnd then
            $\mathbf{y}^F \leftarrow \nu^-(\mathbf{x}^G)$
            if $(\text{BlockDefined}(F, \mathbf{y}^F, -) = \textbf{false})$ continue
            $\mathbf{x}^X \leftarrow \text{BlockF}^{-1}(\mathbf{y}^F)$
        else
            $\mathbf{x}^X \leftarrow \mathbf{x}^G$
        if $(\text{CheckZ}(\mathbf{x}^X, \mathbf{y}^H) = \textbf{false})$ continue
        if $(\text{++NumOuter} \geq q + 1)$ then abort
        $\mathbf{y}^U \leftarrow \text{RightToMiddle}(\mathbf{y}^H)$
        $\text{NewYUs} \leftarrow \text{NewYUs} \cup \mathbf{y}^U$
    forall $\mathbf{y}^U \in \text{NewYUs}$ do
        $\text{BlockRequestU}^{-1}(\mathbf{y}^U)$

private procedure RequestU$^{-1}(j, y)$
    if $U_j^{-1}(y) \neq \bot$ then return
    if $y \in \text{ToBeAssigned}^-_{U_j}$ then return
    $\text{ToBeAssigned}^-_{U_j} \leftarrow \text{ToBeAssigned}^-_{U_j} \cup \{y\}$
    forall $\mathbf{y}^U$ s.t. $\mathbf{y}^U[j] = y$ do
        if $(\text{BlockDefined}(U, \mathbf{y}^U, -) = \textbf{false})$ continue
        if XtraMiddleRnd then
            if $(\text{BlockDefined}(C, \tau(\mathbf{y}^U), +) = \textbf{false})$ continue
        $\mathbf{y}^H \leftarrow \text{MiddleToRight}(\mathbf{y}^U)$
        $\mathbf{x}^G \leftarrow \mathbf{x}^X \leftarrow \text{Z}^{-1}(\mathbf{y}^H)$
        if XtraOuterRnd then
            $\mathbf{x}^G \leftarrow \nu(\text{BlockF}(\mathbf{x}^X))$
        $\text{ToBeAdapted}_D \leftarrow \text{ToBeAdapted}_D \cup (\mathbf{x}^G, \mathbf{y}^U)$
    forall $(\mathbf{x}^G, \mathbf{y}^U) \in \text{ToBeAdapted}_D$ do
        $\text{BlockRequestG}(\mathbf{x}^G)$

private procedure RightToMiddle$(\mathbf{y}^H)$
    $\mathbf{x}^H \leftarrow \text{BlockH}^{-1}(\mathbf{y}^H)$
    if XtraUntglRnds then
        $\mathbf{x}^E \leftarrow \text{BlockE}^{-1}(\pi^-_L(\text{BlockL}^{-1}(\pi^-_H(\mathbf{x}^H))))$
        $\mathbf{y}^V \leftarrow \pi^-_C(\text{BlockK}^{-1}(\pi^{-1}_K(\mathbf{x}^E)))$
    else
        $\mathbf{y}^V \leftarrow \pi^-_C(\text{BlockE}^{-1}(\pi^-_H(\mathbf{x}^H)))$
    if XtraMiddleRnd then
        return $\tau^-(\text{BlockC}^{-1}(\mathbf{y}^V))$
    else
        return $\mathbf{y}^V$
```

```
private procedure RequestH$^{-1}(j, y)$
    if $H_j^{-1}(y) \neq \bot$ then return
    if $y \in \text{ToBeAssigned}^-_{H_j}$ then return
    $\text{ToBeAssigned}^-_{H_j} \leftarrow \text{ToBeAssigned}^-_{H_j} \cup \{y\}$
    $\text{NewXVs} \leftarrow \emptyset$  // local var
    forall $\mathbf{y}^H, \mathbf{x}^G$ s.t. $\mathbf{y}^H[j] = y$ do
        if $\exists \mathbf{x}^V$ s.t. $(\mathbf{x}^V, \mathbf{y}^H) \in \text{ToBeAdapted}_E$ continue
        if $(\text{BlockDefined}(H, \mathbf{y}^H, -) = \textbf{false})$ continue
        if $(\text{BlockDefined}(G, \mathbf{x}^G, +) = \textbf{false})$ continue
        if XtraOuterRnd then
            $\mathbf{y}^F \leftarrow \nu^-(\mathbf{x}^G)$
            if $(\text{BlockDefined}(F, \mathbf{y}^F, -) = \textbf{false})$ continue
            $\mathbf{x}^X \leftarrow \text{BlockF}^{-1}(\mathbf{y}^F)$
        else
            $\mathbf{x}^X \leftarrow \mathbf{x}^G$
        if $(\text{CheckZ}(\mathbf{x}^X, \mathbf{y}^H) = \textbf{false})$ continue
        if $(\text{++NumOuter} \geq q + 1)$ then abort
        $\mathbf{x}^V \leftarrow \text{LeftToMiddle}(\mathbf{x}^X)$
        $\text{NewXVs} \leftarrow \text{NewXVs} \cup \mathbf{x}^V$
    forall $\mathbf{x}^V \in \text{NewXVs}$ do
        $\text{BlockRequestV}(\mathbf{x}^V)$

private procedure RequestV$(j, x)$
    if $V_j(x) \neq \bot$ then return
    if $x \in \text{ToBeAssigned}_{V_j}$ then return
    $\text{ToBeAssigned}_{V_j} \leftarrow \text{ToBeAssigned}_{V_j} \cup \{x\}$
    forall $\mathbf{x}^V$ s.t. $\mathbf{x}^V[j] = x$ do
        if $(\text{BlockDefined}(V, \mathbf{y}^V, +) = \textbf{false})$ continue
        if XtraMiddleRnd then
            if $(\text{BlockDefined}(B, \tau^-(\mathbf{x}^V), -) = \textbf{false})$ continue
        $\mathbf{x}^X \leftarrow \text{MiddleToLeft}(\mathbf{x}^V)$
        $\mathbf{y}^H \leftarrow \text{Z}(\mathbf{x}^X)$
        $\text{ToBeAdapted}_E \leftarrow \text{ToBeAdapted}_E \cup (\mathbf{x}^V, \mathbf{y}^H)$
    forall $(\mathbf{x}^V, \mathbf{y}^H) \in \text{ToBeAdapted}_E$ do
        $\text{BlockRequestH}^{-1}(\mathbf{y}^H)$

private procedure LeftToMiddle$(\mathbf{x}^X)$
    if XtraOuterRnd then
        $\mathbf{y}^G \leftarrow \text{BlockG}(\nu(\text{BlockF}(\mathbf{x}^X)))$
    else
        $\mathbf{y}^G \leftarrow \text{BlockG}(\mathbf{x}^X)$
    if XtraUntglRnds then
        $\mathbf{y}^D \leftarrow \text{BlockD}(\pi_I(\text{BlockI}(\pi_G(\mathbf{y}^G))))$
        $\mathbf{x}^U \leftarrow \pi_B(\text{BlockJ}(\pi_J(\mathbf{y}^D)))$
    else
        $\mathbf{x}^U \leftarrow \pi_C(\text{BlockE}(\pi_G(\mathbf{y}^G)))$
    if XtraMiddleRnd then
        return $\tau(\text{BlockB}(\mathbf{x}^U))$
    else
        return $\mathbf{x}^U$
```

Fig. 4. Games G_1 through G_3, third set of procedures.

```
private procedure MiddleToRight($\mathbf{y}^U$)
    if XtraMiddleRnd then
        $\mathbf{y}^V \leftarrow \mathrm{BlockV}(\tau(\mathbf{y}^U))$
    else
        $\mathbf{y}^V \leftarrow \mathbf{y}^U$
    if XtraUntglRnds then
        $\mathbf{y}^E \leftarrow \mathrm{BlockE}(\pi_K(\mathrm{BlockK}(\pi_C(\mathbf{y}^V))))$
        $\mathbf{x}^H \leftarrow \pi_H(\mathrm{BlockL}(\pi_L(\mathbf{y}^E)))$
    else
        $\mathbf{x}^H \leftarrow \pi_H(\mathrm{BlockE}(\pi_C(\mathbf{y}^V))$
    return BlockH($\mathbf{x}^H$)

private procedure AdaptLeft()
    forall $j \in \{1, \ldots, w\}$ do
        forall $x \in \mathrm{ToBeAssigned}_{G_j}$ do
            ReadTape($G_j, x, p_{G_j}$)
        forall $y \in \mathrm{ToBeAssigned}^-_{U_j}$ do
            ReadTape($U_j^{-1}, y, p^-_{U_j}$)
        $\mathrm{ToBeAssigned}_{G_j} \leftarrow \emptyset$
        $\mathrm{ToBeAssigned}^-_{U_j} \leftarrow \emptyset$
    forall $(\mathbf{x}^G, \mathbf{y}^U) \in \mathrm{ToBeAdapted}_D$
        if XtraUntglRnds then
            $\mathbf{x}^D \leftarrow \pi_I(\mathrm{BlockI}(\pi_G(\mathrm{BlockG}(\mathbf{x}^G))))$
            $\mathbf{y}^D \leftarrow \pi^-_J(\mathrm{BlockJ}^{-1}(\pi^-_B(\mathrm{BlockU}^{-1}(\mathbf{y}^U))))$
        else
            $\mathbf{x}^D \leftarrow \pi_G(\mathrm{BlockG}(\mathbf{x}^G))$
            $\mathbf{y}^D \leftarrow \pi^-_B(\mathrm{BlockU}^{-1}(\mathbf{y}^U))$
        forall $j \in \{1, \ldots, w\}$ do
            SetTable($D_j, \mathbf{x}^D[j], \mathbf{y}^D[j]$)
    $\mathrm{ToBeAdapted}_D \leftarrow \emptyset$

private procedure BlockU$^{-1}$($\mathbf{y}$)
    forall $j \in \{1, \ldots, w\}$ do
        if $U_j^{-1}(\mathbf{y}[j]) = \bot$ abort
        $\mathbf{x}[j] \leftarrow U_j^{-1}(\mathbf{y}[j])$
    return $\mathbf{x}$

private procedure BlockG($\mathbf{x}$)
    forall $j \in \{1, \ldots, w\}$ do
        if $G_j(\mathbf{x}[j]) = \bot$ abort
        $\mathbf{y}[j] \leftarrow G_j(\mathbf{x}[j])$
    return $\mathbf{y}$

private procedure BlockRequestG($\mathbf{x}$)
    forall $j \in \{1, \ldots, w\}$ do
        RequestG($\mathbf{x}[j], j$)

private procedure BlockRequestU$^{-1}$($\mathbf{x}$)
    forall $j \in \{1, \ldots, w\}$ do
        RequestU$^{-1}$($\mathbf{x}[j], j$)
```

```
private procedure MiddleToLeft($\mathbf{x}^V$)
    if XtraMiddleRnd then
        $\mathbf{x}^U \leftarrow \mathrm{BlockU}^{-1}(\tau^-(\mathbf{x}^V))$
    else
        $\mathbf{x}^U \leftarrow \mathbf{x}^V$
    if XtraUntglRnds then
        $\mathbf{x}^D \leftarrow \mathrm{BlockD}^{-1}(\pi^-_J(\mathrm{BlockJ}^{-1}(\pi^-_B(\mathbf{x}^U))))$
        $\mathbf{y}^G \leftarrow \pi^-_G(\mathrm{BlockI}^{-1}(\pi^-_I(\mathbf{x}^D)))$
    else
        $\mathbf{y}^G \leftarrow \pi^-_G(\mathrm{BlockD}(\pi^-_B(\mathbf{x}^U))$
    if XtraOuterRnd then
        return $\mathrm{BlockF}^{-1}(\nu^-(\mathrm{BlockH}^{-1}(\mathbf{y}^G)))$
    else
        return $\mathrm{BlockH}^{-1}(\mathbf{y}^G)$

private procedure AdaptRight()
    forall $j \in \{1, \ldots, w\}$ do
        forall $y \in \mathrm{ToBeAssigned}^-_{H_j}$ do
            ReadTape($H_j^{-1}, y, p^-_{H_j}$)
        forall $x \in \mathrm{ToBeAssigned}_{V_j}$ do
            ReadTape($V_j, x, p_{V_j}$)
        $\mathrm{ToBeAssigned}^-_{H_j} \leftarrow \emptyset$
        $\mathrm{ToBeAssigned}_{V_j} \leftarrow \emptyset$
    forall $(\mathbf{x}^V, \mathbf{y}^H) \in \mathrm{ToBeAdapted}_E$
        if XtraUntglRnds then
            $\mathbf{y}^E \leftarrow \pi^-_L(\mathrm{BlockL}^{-1}(\pi^-_H(\mathrm{BlockH}^{-1}(\mathbf{y}^H))))$
            $\mathbf{x}^E \leftarrow \pi_K(\mathrm{BlockK}(\pi_C(\mathrm{BlockV}(\mathbf{x}^V))))$
        else
            $\mathbf{y}^E \leftarrow \pi^-_H(\mathrm{BlockH}^{-1}(\mathbf{y}^H))$
            $\mathbf{x}^E \leftarrow \pi_C(\mathrm{BlockV}(\mathbf{x}^V))$
        forall $j \in \{1, \ldots, w\}$ do
            SetTable($E_j, \mathbf{x}^E[j], \mathbf{y}^E[j]$)
    $\mathrm{ToBeAdapted}_E \leftarrow \emptyset$

private procedure BlockV($\mathbf{x}$)
    forall $j \in \{1, \ldots, w\}$ do
        if $V_j(\mathbf{x}[j]) = \bot$ abort
        $\mathbf{y}[j] \leftarrow V_j^{-1}(\mathbf{x}[j])$
    return $\mathbf{y}$

private procedure BlockH$^{-1}$($\mathbf{y}$)
    forall $j \in \{1, \ldots, w\}$ do
        if $H_j^{-1}(\mathbf{y}[j]) = \bot$ abort
        $\mathbf{x}[j] \leftarrow H_j^{-1}(\mathbf{y}[j])$
    return $\mathbf{x}$

private procedure BlockRequestH$^{-1}$($\mathbf{y}$)
    forall $j \in \{1, \ldots, w\}$ do
        RequestH$^{-1}$($\mathbf{x}[j], j$)

private procedure BlockRequestV($\mathbf{y}$)
    forall $j \in \{1, \ldots, w\}$ do
        RequestV($\mathbf{x}[j], j$)
```

Fig. 5. Games G_1 through G_3, fourth set of procedures.

```
private procedure BlockB($\mathbf{x}$)
    forall $j \in \{1, \ldots, w\}$ do
        $\mathbf{y}[j] \leftarrow \mathrm{B}(\mathbf{x}[j], j)$
    return $\mathbf{y}$

private procedure BlockG$^{-1}$($\mathbf{y}$)
    forall $j \in \{1, \ldots, w\}$ do
        $\mathbf{x}[j] \leftarrow \mathrm{G}^{-1}(\mathbf{y}[j], j)$
    return $\mathbf{x}$

private procedure BlockD($\mathbf{x}$)
    forall $j \in \{1, \ldots, w\}$ do
        $\mathbf{y}[j] \leftarrow \mathrm{D}(\mathbf{x}[j], j)$
    return $\mathbf{y}$

private procedure BlockD$^{-1}$($\mathbf{y}$)
    forall $j \in \{1, \ldots, w\}$ do
        $\mathbf{x}[j] \leftarrow \mathrm{D}^{-1}(\mathbf{y}[j], j)$
    return $\mathbf{x}$

private procedure BlockF($\mathbf{x}$)
    forall $j \in \{1, \ldots, w\}$ do
        $\mathbf{y}[j] \leftarrow \mathrm{F}(\mathbf{x}[j], j)$
    return $\mathbf{y}$

private procedure BlockF$^{-1}$($\mathbf{y}$)
    forall $j \in \{1, \ldots, w\}$ do
        $\mathbf{x}[j] \leftarrow \mathrm{F}^{-1}(\mathbf{y}[j], j)$
    return $\mathbf{x}$

private procedure BlockI($\mathbf{x}$)
    forall $j \in \{1, \ldots, w\}$ do
        $\mathbf{y}[j] \leftarrow \mathrm{I}(\mathbf{x}[j], j)$
    return $\mathbf{y}$

private procedure BlockI$^{-1}$($\mathbf{y}$)
    forall $j \in \{1, \ldots, w\}$ do
        $\mathbf{x}[j] \leftarrow \mathrm{I}^{-1}(\mathbf{y}[j], j)$
    return $\mathbf{x}$

private procedure BlockI($\mathbf{x}$)
    forall $j \in \{1, \ldots, w\}$ do
        $\mathbf{y}[j] \leftarrow \mathrm{I}(\mathbf{x}[j], j)$
    return $\mathbf{y}$

private procedure BlockI$^{-1}$($\mathbf{y}$)
    forall $j \in \{1, \ldots, w\}$ do
        $\mathbf{x}[j] \leftarrow \mathrm{I}^{-1}(\mathbf{y}[j], j)$
    return $\mathbf{x}$
```

```
private procedure BlockC$^{-1}$($\mathbf{y}$)
    forall $j \in \{1, \ldots, w\}$ do
        $\mathbf{x}[j] \leftarrow \mathrm{C}^{-1}(\mathbf{y}[j], j)$
    return $\mathbf{x}$

private procedure BlockH($\mathbf{x}$)
    forall $j \in \{1, \ldots, w\}$ do
        $\mathbf{y}[j] \leftarrow \mathrm{H}(\mathbf{x}[j], j)$
    return $\mathbf{y}$

private procedure BlockE($\mathbf{x}$)
    forall $j \in \{1, \ldots, w\}$ do
        $\mathbf{y}[j] \leftarrow \mathrm{E}(\mathbf{x}[j], j)$
    return $\mathbf{y}$

private procedure BlockE$^{-1}$($\mathbf{y}$)
    forall $j \in \{1, \ldots, w\}$ do
        $\mathbf{x}[j] \leftarrow \mathrm{E}^{-1}(\mathbf{y}[j], j)$
    return $\mathbf{x}$

private procedure BlockK($\mathbf{x}$)
    forall $j \in \{1, \ldots, w\}$ do
        $\mathbf{y}[j] \leftarrow \mathrm{K}(\mathbf{x}[j], j)$
    return $\mathbf{y}$

private procedure BlockK$^{-1}$($\mathbf{y}$)
    forall $j \in \{1, \ldots, w\}$ do
        $\mathbf{x}[j] \leftarrow \mathrm{K}^{-1}(\mathbf{y}[j], j)$
    return $\mathbf{x}$

private procedure BlockL($\mathbf{x}$)
    forall $j \in \{1, \ldots, w\}$ do
        $\mathbf{y}[j] \leftarrow \mathrm{L}(\mathbf{x}[j], j)$
    return $\mathbf{y}$

private procedure BlockL$^{-1}$($\mathbf{y}$)
    forall $j \in \{1, \ldots, w\}$ do
        $\mathbf{x}[j] \leftarrow \mathrm{L}^{-1}(\mathbf{y}[j], j)$
    return $\mathbf{x}$
```

Fig. 6. Games G_1, G_2, G_3, continuation and end.

```
Game G$_4$

random tapes: $q_{A_1}, \ldots, q_{L_w}$

global static: XtraOuterRnd, XtraMiddleRnd, XtraUntglRnds

private procedure Fwd($T, \mathbf{x}$)
    forall $j \in \{1, \ldots, w\}$ do
        $\mathbf{y}[j] \leftarrow q_{T_j}(\mathbf{x}[j])$
    return $\mathbf{y}$

private procedure EvaluateForward($\mathbf{x}^X$)
    if XtraOuterRnd then
        $\mathbf{y}^G \leftarrow \mathrm{Fwd}(G, \nu(\mathrm{Fwd}(F, \mathbf{x}^X)))$
    else
        $\mathbf{y}^G \leftarrow \mathrm{Fwd}(\mathbf{x}^X)$
    if XtraUntglRnds then
        $\mathbf{y}^D \leftarrow \mathrm{Fwd}(D, \pi_I(\mathrm{Fwd}(I, \pi_G(\mathbf{y}^G))))$
        $\mathbf{x}^U \leftarrow \pi_B(\mathrm{Fwd}(I, \pi_I(\mathbf{y}^D)))$
    else
        $\mathbf{x}^U \leftarrow \pi_C(\mathrm{Fwd}(E, \pi_G(\mathbf{y}^G)))$
    if XtraMiddleRnd then
        $\mathbf{y}^V \leftarrow \mathrm{Fwd}(C, \tau(\mathrm{Fwd}(B, \mathbf{x}^U)))$
    else
        $\mathbf{y}^V \leftarrow \mathrm{Fwd}(A, \mathbf{x}^U)$
    if XtraUntglRnds
        $\mathbf{y}^E \leftarrow \mathrm{Fwd}(E, \pi_K(\mathrm{Fwd}(K, \pi_C(\mathbf{y}^V))))$
        $\mathbf{x}^H \leftarrow \pi_H(\mathrm{Fwd}(L, \pi_L(\mathbf{y}^E)))$
    else
        $\mathbf{x}^H \leftarrow \pi_H(\mathrm{Fwd}(E, \pi_C(\mathbf{y}^V)))$
        return Fwd($H, \mathbf{x}^H$)

public procedure Z($\mathbf{x}$)
    if $P_Z(\mathbf{x}) = \perp$ then
        $\mathbf{y} \leftarrow$ EvaluateForward($\mathbf{x}$)
        SetTable($P_Z, \mathbf{x}, \mathbf{y}$)
        if $\exists j, H_j^{-1}(\mathbf{y}[j]) \neq \perp$ then abort
    return $P_Z(\mathbf{x})$
```

```
private procedure Bwd($T, \mathbf{y}$)
    forall $j \in \{1, \ldots, w\}$ do
        $\mathbf{x}[j] \leftarrow q^-_{T_j}(\mathbf{y}[j])$
    return $\mathbf{x}$

private procedure EvaluateBackward($\mathbf{y}^H$)
    $\mathbf{x}^H \leftarrow \mathrm{Bwd}(H, \mathbf{y}^H)$
    if XtraUntglRnds then
        $\mathbf{x}^E \leftarrow \mathrm{Bwd}(E, \pi^-_L(\mathrm{Bwd}(L, \pi^-_H(\mathbf{x}^H))))$
        $\mathbf{y}^V \leftarrow \pi^-_C(\mathrm{Bwd}(K, \pi^-_K(x^E)))$
    else
        $\mathbf{y}^V \leftarrow \pi^-_C(\mathrm{Bwd}(E, \pi^-_H(\mathbf{x}^H)))$
    if XtraMiddleRnd then
        $\mathbf{x}^U \leftarrow \mathrm{Bwd}(B, \tau^-(\mathrm{Bwd}(C, \mathbf{y}^V)))$
    else
        $\mathbf{x}^U \leftarrow \mathrm{Bwd}(A, \mathbf{y}^U)$
    if XtraUntglRnds
        $\mathbf{x}^D \leftarrow \mathrm{Bwd}(D, \pi^-_I(\mathrm{Bwd}(I, \pi^-_B(\mathbf{x}^U))))$
        $\mathbf{y}^G \leftarrow \pi^-_G(\mathrm{Bwd}(I, \pi^-_I(\mathbf{x}^D)))$
    else
        $\mathbf{y}^G \leftarrow \pi^-_G(\mathrm{Bwd}(E, \pi^-_B(\mathbf{x}^U))$
    if XtraOuterRnd then
        return Bwd($F, \nu^-(\mathrm{Bwd}(G, \mathbf{y}^G))$)
    else
        return Bwd($G, \mathbf{y}^G$)

public procedure Z$^{-1}$($\mathbf{y}$)
    if $P_Z^{-1}(\mathbf{y}) = \perp$ then
        $\mathbf{x} \leftarrow$ EvaluateBackward($\mathbf{y}$)
        SetTable($P_Z, \mathbf{x}, \mathbf{y}$)
        if $\exists j, X_j(\mathbf{x}[j]) \neq \perp$ then abort
    return $P_Z^{-1}(\mathbf{y})$
```

Fig. 7. Game G$_4$ with random tapes and public procedures Z, Z^{-1}. Other public procedures are implemented as in G$_3$, substituting G$_4$'s random tape q_{T_j} for G$_3$'s random tape p_{T_j}, $T \in \{A, B, \ldots, L, L\}$.

```
Game G_5

random tapes: $q_{A_1}, \ldots, q_{M_w}$

global static: XtraOuterRnd, XtraMiddleRnd, XtraUntglRnds

private procedure Fwd($T, \mathbf{x}$)
    forall $j \in \{1, \ldots, w\}$ do
        $\mathbf{y}[j] \leftarrow q_{T_j}(\mathbf{x}[j])$
    return y

private procedure EvaluateForward($\mathbf{x}^X$)
    if XtraOuterRnd then
        $\mathbf{y}^G \leftarrow \mathrm{Fwd}(G, \nu(\mathrm{Fwd}(F, \mathbf{x}^X)))$
    else
        $\mathbf{y}^G \leftarrow \mathrm{Fwd}(\mathbf{x}^X)$
    if XtraUntglRnds then
        $\mathbf{y}^D \leftarrow \mathrm{Fwd}(D, \pi_I(\mathrm{Fwd}(I, \pi_G(\mathbf{y}^G)))))$
        $\mathbf{x}^U \leftarrow \pi_B(\mathrm{Fwd}(I, \pi_I(\mathbf{y}^D)))$
    else
        $\mathbf{x}^U \leftarrow \pi_C(\mathrm{Fwd}(E, \pi_G(\mathbf{y}^G)))$
    if XtraMiddleRnd then
        $\mathbf{y}^V \leftarrow \mathrm{Fwd}(C, \tau(\mathrm{Fwd}(B, \mathbf{x}^U)))$
    else
        $\mathbf{y}^V \leftarrow \mathrm{Fwd}(A, \mathbf{x}^U)$
    if XtraUntglRnds
        $\mathbf{y}^E \leftarrow \mathrm{Fwd}(E, \pi_K(\mathrm{Fwd}(K, \pi_C(\mathbf{y}^V))))$
        $\mathbf{x}^H \leftarrow \pi_H(\mathrm{Fwd}(L, \pi_L(\mathbf{y}^E)))$
    else
        $\mathbf{x}^H \leftarrow \pi_H(\mathrm{Fwd}(E, \pi_C(\mathbf{y}^V))$
        return $\mathrm{Fwd}(H, \mathbf{x}^H)$

public procedure Z($\mathbf{x}$)
    return EvaluateForward($\mathbf{x}$)

public procedure T($x, j$)
    return $q_{T_j}(x)$
```

```
private procedure Bwd($T, \mathbf{y}$)
    forall $j \in \{1, \ldots, w\}$ do
        $\mathbf{x}[j] \leftarrow q^-_{T_j}(\mathbf{y}[j])$
    return x

private procedure EvaluateBackward($\mathbf{y}^H$)
    $\mathbf{x}^H \leftarrow \mathrm{Bwd}(H, \mathbf{y}^H)$
    if XtraUntglRnds then
        $\mathbf{x}^E \leftarrow \mathrm{Bwd}(E, \pi^-_L(\mathrm{Bwd}(L, \pi^-_H(\mathbf{x}^H)))))$
        $\mathbf{y}^V \leftarrow \pi^-_C(\mathrm{Bwd}(K, \pi^-_K(x^E)))$
    else
        $\mathbf{y}^V \leftarrow \pi^-_C(\mathrm{Bwd}(E, \pi^-_H(\mathbf{x}^H)))$
    if XtraMiddleRnd then
        $\mathbf{x}^U \leftarrow \mathrm{Bwd}(B, \tau^-(\mathrm{Bwd}(C, \mathbf{y}^V)))$
    else
        $\mathbf{x}^U \leftarrow \mathrm{Bwd}(A, \mathbf{y}^U)$
    if XtraUntglRnds
        $\mathbf{x}^D \leftarrow \mathrm{Bwd}(D, \pi^-_I(\mathrm{Bwd}(I, \pi^-_B(\mathbf{x}^U))))$
        $\mathbf{y}^G \leftarrow \pi^-_G(\mathrm{Bwd}(I, \pi^-_I(\mathbf{x}^D)))$
    else
        $\mathbf{y}^G \leftarrow \pi^-_G(\mathrm{Bwd}(E, \pi^-_B(\mathbf{x}^U))$
    if XtraOuterRnd then
        return $\mathrm{Bwd}(F, \nu^-(\mathrm{Bwd}(G, \mathbf{y}^G)))$
    else
        return $\mathrm{Bwd}(G, \mathbf{y}^G)$

public procedure $\mathrm{Z}^{-1}(\mathbf{y})$
    return EvaluateBackward($\mathbf{y}$)

public procedure $\mathrm{T}^{-1}(y, j)$
    return $q^-_{T_j}(y)$
```

Fig. 8. Game G$_5$. The procedures EvaluateForward and EvaluateBackward are as in game G$_4$. The procedures T and T^{-1} in G$_5$ are templates, to be instantiated with each $T \in \{A, B, \ldots, K, L\}$.