Crooked Indifferentiability of the Feistel Construction

Alexander Russell[1], Qiang Tang[2], and Jiadong Zhu[3]

[1] University of Connecticut, acr@cse.uconn.edu
[2] The University of Sydney, qiang.tang@sydney.edu.au
[3] State Key Lab of Processors, Institute of Computing Technology, Chinese Academy of Sciences, zhujiadong2016@163.com

Abstract. The Feistel construction is a fundamental technique for building pseudorandom permutations and block ciphers. This paper shows that a simple adaptation of the construction is resistant, even to algorithm substitution attacks—that is, adversarial subversion—of the component round functions. Specifically, we establish that a Feistel-based construction with more than $337n/\log(1/\epsilon)$ rounds can transform a subverted random function—which disagrees with the original one at a small fraction (denoted by ϵ) of inputs—into an object that is *crooked-indifferentiable* from a random permutation, even if the adversary is aware of all the randomness used in the transformation. Here, n denotes the length of both the input and output of the round functions that underlie the Feistel cipher. We also provide a lower bound showing that the construction cannot use fewer than $2n/\log(1/\epsilon)$ rounds to achieve crooked-indifferentiable security.

1 Introduction

Random oracles and random permutations are idealized models that have proven to be powerful tools for designing and reasoning about cryptographic schemes, particularly in the settings of symmetric key encryption (e.g., block ciphers) and hash designs. They consist of the following two steps: (i) design a scheme Π in which all parties (including the adversary) have oracle access to (a family of) truly random functions or random permutations (and the corresponding inversions), and establish the security of Π in this favorable setting; (ii) instantiate the oracle in Π with a suitable cipher (such as SHA-1 or AES) to obtain an instantiated scheme Π'. The random oracle (random permutation) heuristic states that if the original scheme Π is secure, then the instantiated scheme Π' is also secure. In this work we focus on the problem of correcting faulty—or adversarially corrupted—random oracles/random permutations so that they can be confidently applied for such cryptographic purposes.

One particular motivation for correcting random oracle/random permutation in a cryptographic context arises from works studying design and security in the subversion (i.e., *kleptographic*) setting. In this setting, various components of a cryptographic scheme may be subverted by an adversary, so long as the tampering cannot be detected

via blackbox testing. This is a challenging framework because many basic cryptographic techniques are not directly available: in particular, the random oracle/random permutation paradigm is directly undermined. In terms of the discussion above, the random oracle/random permutation—which is eventually to be replaced with a concrete cipher—is subject to adversarial subversion which complicates even the first step of the random oracle/random permutation methodology. To see a simple example, for AES, denoted as $(\mathsf{AES.K}, \mathsf{AES.E}, \mathsf{AES.D})$, whose software/hardware implementation (denoted as $\widetilde{\mathsf{AES.K}}, \widetilde{\mathsf{AES.E}}, \widetilde{\mathsf{AES.D}}$) might be subverted as follows: $\widetilde{\mathsf{AES.E}}(k, m^*) = k$, for a trigger message m^* randomly chosen by the adversary, while $\widetilde{\mathsf{AES.E}} = \mathsf{AES.E}$ otherwise, i.e., only when encrypting a special trigger message, the subverted encryption algorithm directly outputs the secret key. This is clearly undetectable via blackbox testing. Since the subverted implementation of AES now cannot be assumed to be an ideal cipher anymore, the security of constructions that previously relied on this assumption also becomes elusive.

Our goal is to provide a generic approach that can rigorously "protect" the usage of random oracle/random permutation from subversion and, essentially, establish a "crooked" random oracle/random permutation methodology. Specifically, given a function $\tilde{h}$ drawn from a distribution which *agrees in most places* with a uniform function (permutation), we would like to produce a corrected version which appears still as a random oracle/random permutation to adversaries with a polynomially bounded number of queries. This model is also analogous to the classical study of "program checking and self-correcting" [3,4,5]: the goal in this theory is to transform a program that is faulty at a small fraction of inputs (modeling an evasive adversary) to a program that is correct at all points with overwhelming probability. Our setting intuitively adapts this classical theory of self-correction to the study of "self-correcting a probability distribution." Notably, the functions to be corrected are structureless, instead of heavily structured.

The model of "crooked" indifferentiability. The first work in this line was [20], focusing on correcting subverted random oracles; in particular, they introduced a security model called *crooked-indifferentiability* to formally capture the problem as follows: First, a function $h : \{0,1\}^n \to \{0,1\}^n$ is drawn uniformly at random. Then, an adversary may *subvert* the function h, yielding a new function $\tilde{h}$. The subverted function $\tilde{h}(x)$ is described by an adversarially-chosen (polynomial-time) algorithm $\mathcal{A}$, with oracle access to h. This function may differ from the original function (so that $\tilde{h}(x) \neq h(x)$) at only a negligible fraction of inputs (to evade blackbox testing). To show that the resulting function (construction) is "as good as" a random oracle in the sense of indifferentiability [17,9], a *crooked-distinguisher* $\mathcal{D}$ was introduced; it first prepares the subverted implementation $\tilde{h}$ (after querying h first); then a fixed amount of (public) randomness R is drawn and published; the construction C may use only the subverted implementation $\tilde{h}$ and the randomness R. Now, following the indifferentiability framework, we will ask for a simulator $\mathcal{S}$ such that $(C^{\tilde{h}}(\cdot, R), h)$ and $(\mathcal{F}, \mathcal{S}^{\mathcal{A}}(R))$ are indistinguishable to any crooked-distinguisher $\mathcal{D}$ (even one who knows R).

1.1 Our Contribution

We investigate the above question in the more restrictive random permutation setting. We first adopt the security model of crooked-indifferentiability for random permutation[4]. (A formal definition appears in Section 2.)

A warm-up construction. To consider feasibility of correcting a subverted random permutation, and also as an example to explore the crooked-indifferentiability model, we start with a warm-up construction by composing the following two components.

Component I. The first component is built from a source random function that was proven to be *crooked*-indifferentiable from a random oracle [20].

The source function is expressed as a family of $\ell + 1$ independent random oracles:

$$h_0 : \{0,1\}^n \to \{0,1\}^{3n}, \text{ and } h_i : \{0,1\}^{3n} \to \{0,1\}^n \text{ for } i \in \{1,\ldots,\ell\}.$$

These can be realized as slices of a single random function $H : \{0,1\}^{n'} \to \{0,1\}^{n'}$, with $n' = 3n + \lceil \log \ell + 1 \rceil$ by an appropriate convention for embedding and extracting inputs and values. Given subverted implementations $\{\tilde{h}_i\}_{i=0,\ldots,\ell}$ (defined as above by the adversarially-defined algorithm $\mathcal{A}$), the corrected function is defined as:

$$C^{\tilde{h}}(x) \stackrel{\text{def}}{=} \tilde{h}_0 \left(\bigoplus_{i=1}^{\ell} \tilde{h}_i(x \oplus r_i) \right),$$

where $R = (r_1, \ldots, r_\ell)$ is sampled uniformly after $\tilde{h}$ is provided (and then made public).

Component II: the classical Feistel cipher. The second component is the classical Feistel cipher with ℓ rounds for $\ell = 14$. Coron et al. [10] proved it is indifferentiable from a random permutation. The classical ℓ-*round Feistel cipher* transforms a sequence of functions $F_1, \ldots, F_\ell : \{0,1\}^n \to \{0,1\}^n$ into a permutation on the set $\{0,1\}^{2n}$. The construction logically treats $2n$-bit strings as pairs (x, y), with $x, y \in \{0,1\}^n$, and is defined as the composition of a sequence of permutations defined by the F_i. Specifically, given an input (x_0, x_1), the construction defines

$$x_{i+1} := x_{i-1} \oplus F_i(x_i)$$

for each $i = 1, \ldots, \ell$, and results in the output string $(x_\ell, x_{\ell+1})$. It is easy to see that the resulting function is a permutation. In practical settings, the "round functions" (F_i) are often keyed functions (determined by secret keys of length $\text{poly}(n)$), in which case the construction results in a keyed permutation.

Composing the two components. We can compose the above two components by replacing the 14 round functions in component II with 14 independent copies of component I. The result construction, by the property of indifferentiability, is also crooked-indifferentiable

---

[4] In the crooked-indifferentiability model, it is noteworthy that for a specific construction C, the need to correct subverted random oracles and subverted random permutations can be simplified to addressing subverted random permutations alone. This is due to the fact that a subverted random permutation deviates negligibly from a subverted random function. Thus, the focus in the subsequent sections will be on correcting subverted random permutations exclusively.
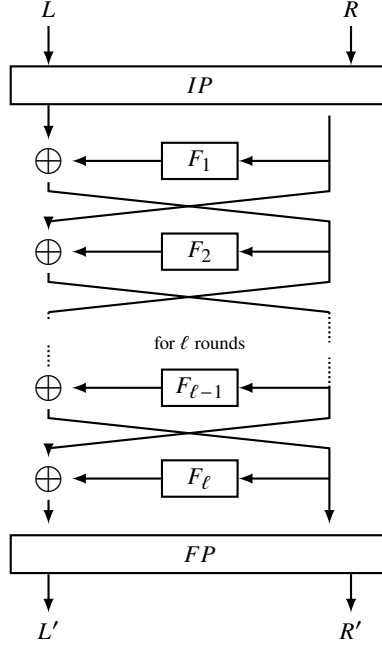
Fig. 1: The ℓ round classical Feistel construction.

from a random permutation as a corollary of the replacement theorem of crooked-indifferentiability (see Section A.3 in the Supplementary Materials).

Our direct and "optimal" construction. However, there are two drawbacks of this construction. First, the structure of the construction is complicated. Second, it makes at least linear number of invocations of the underlying subverted component (and also $O(n^2)$ random bits) to achieve security. Instead, we prove that a *direct* Feistel-based construction can also work and remove these drawbacks. In particular, our construction involving only *public* randomness can boost a "subverted" random permutation (or just a function) into a construction that is indifferentiable from a perfect random permutation. (Section 3, 4). Besides structure-wise simplicity (and the fact that it adopts the direct Feistel structure), our construction requires a smaller number of invocations of the underlying (subverted) random permutation, which is essentially optimal up to constant factors (at least for the Feistel structure, as we prove impossibility to have fewer rounds; there was also explicit attacks for the case of random oracle in [20], but the construction in [20] was not "tight" in this sense, with a factor of $O(\log(1/\epsilon))$).

Our subversion-resistant construction on strings of length $2n$ relies on the parameter ℓ and the Feistel construction applied to ℓ round functions that are determined by:

- ℓ functions $F_i : \{0, 1\}^n \to \{0, 1\}^n$ that are subject to subversion as described above,
- an additional family of ℓ public, uniform affine-linear functions determined by ℓ pairs $(a_i, b_i) \in \mathsf{GL}(\mathbb{F}_2, n) \times \mathbb{F}_2^n$.[5]

---

[5] For technical reasons, we need to encode the input of the round function with the pairwise independent function, please see the proof of Lemma 3 for detailed discussions.

The affine-linear functions are determined by independent and uniform selection of a_i from $\mathsf{GL}(n, \mathbb{F}_2)$ (to be concrete, the collection of invertible $n \times n$ matrices with elements in $\mathbb{F}_2$), and $b_i \in \mathbb{F}_2^n$. The i-th affine linear function, defined on an input $x \in \mathbb{F}_2^n$, is given by the rule $x \mapsto a_i \cdot x \oplus b$. The final construction is given by the Feistel construction applied to the round functions $x \mapsto \tilde{F}_i(a_i \cdot x \oplus b)$, where $\tilde{F}$ is the subverted version of the function F_i. To be concrete, with the data (F_i, a_i, b_i) (with $i = 1, \ldots, \ell$), the construction $C : \{0, 1\}^{2n} \to \{0, 1\}^{2n}$ is defined by the rule

$$C(x_0, x_1) := (x_\ell, x_{\ell+1}) \,, \text{ where}$$
$$x_{i+1} := x_{i-1} \oplus \tilde{F}_i(a_i \cdot x_i \oplus b_i) \,, \text{ for } i = 1, \ldots, \ell \,.$$

where n-bit strings x and b_i are viewed as length n column vectors, $a_i \cdot x_i$ is the multiplication between matrix a_i and column vector x_i, and $\tilde{F}_i(x)$ is the subverted function value at $(i.x)$ using the subversion algorithm $\mathcal{A}$.

New techniques for proving crooked-indifferentiability of Feistel structure. Besides that we aim to get a random permutation, which has stricter requirements, our security analysis needs substantially more sophisticated techniques than that in [20]. The security of the two-layer construction for random oracle in [20] relies on the fact that the XOR structure

$$\tilde{g}_R(x) \overset{\text{def}}{=} \bigoplus_{i=1}^{\ell} \tilde{h}_i(x \oplus r_i)$$

is unpredictable so that the simulator can always program h_0 (at $\tilde{g}_R(x)$). By contrast, our simulator cannot program at one fixed round function (because otherwise the distinguisher can always query this round function first). The simulator needs to flexibly choose where to program according the queries of the distinguisher.

We remark that some techniques in our proof are inspired by the elegant techniques of Coron et al. [10] for conventional indifferentiability; for example, we adopt the concept of "chain" to analyze the basic structure of Feistel construction. However, the subversion of the random function in our setting introduces multiple new challenges, because of, e.g., on-the-fly adaptive queries of the subverted $\tilde{F}$ when the simulator runs it.

To achieve "crooked" indifferentiability, our simulator needs to ensure consistency between two ways of generating output values: one is directly from the construction C; the other calls for an "explanation" of P—a truly random permutation—via reconstruction from related queries to F (in a way consistent with the subverted $\tilde{F}$). To ensure a correct simulation, the simulator must suitably answer related queries (defining one value of C). Essentially, the proof relies on the fact that for any Feistel "chain" $(x_0, \ldots, x_{\ell+1})$, the simulator can find two places (x_u, x_{u+1}) and program $F_u(a_u \cdot x_u \oplus b_u) := x_{u-1} \oplus x_{u+1}$, $F_{u+1}(a_{u+1} \cdot x_{u+1} \oplus b_{u+1}) := x_u \oplus x_{u+2}$ to make the Feistel chain consistent with $P(x_0, x_1) = (x_\ell, x_{\ell+1})$. There are two major challenges in the simulation: first, one of the two programmed terms $F_u(a_u \cdot x_u \oplus b_u)$ and $F_{u+1}(a_{u+1} \cdot x_{u+1} \oplus b_{u+1})$ may be already evaluated prior to programming by the simulator; second, the one of the two programmed terms may be dishonest (i.e., $\tilde{F} \neq F$) so that programming may not be possible.

In the security proof, to analyze the difference between the construction and the ideal object (random permutation), we need to carefully design several intermediate games for transition. Using the games, we reduce the gap between the construction and the

ideal object to the probability of two "bad events" that reflect the two challenges above. Finally, we prove the bad events are negligible by carefully analyzing the structure of our construction. We also need to give a more careful analysis of efficiency of the simulator as it has to internally generate many more terms because of the necessity of running $\tilde{F}$.

Related works. *Conventional indifferentiability of Feistel cipher.* The notion of indifferentiability was proposed in the elegant work of Maurer et al. [17]; this notably extends the classical concept of indistinguishability to circumstances where one or more of the relevant oracles are publicly available (such as a random oracle). It was later adapted by Coron et al. [9]; several other variants were proposed and studied in [15,16]. A line of notable work applied the framework to the ideal cipher problem: in particular the Feistel construction (with a small constant number of rounds) is indifferentiable from a random permutation, see [10,11,12]. Our work adopts the indifferentiability framework applied to the subverted case (that is, crooked-indifferentiability); the construction aims to sanitize a subverted random function to be indifferentiable from a clean random permutation.

Crooked-indifferentiability of random oracles. In [20], the authors proved that a simple two-layer construction using $O(n^2)$ public random bits is crooked-indifferentiable from a random oracle (following results [2,21] gave more rigorous analysis, and showed applications in subversion resistant digital signatures [7]). This work focuses on a strictly stronger goal: to obtain a random *permutation*, and with smaller number of rounds. This line of work was motivated to defend against kleptographic attacks, originally introduced by Young and Yung [22,23], with renewed recent interests (e.g., [1,13,19]).

Related work on non-uniformity and pre-processing. There are several recent approaches that study idealized objects in the auxiliary input model (or with pre-processing) [14,8]. As pointed out in [21], crooked-indifferentiability is strictly stronger than the pre-processing model: besides pre-processing queries, the adversary may embed (and keep) compressed state as backdoor; more importantly, our subverted implementation can further misbehave in ways that cannot be captured by any single-shot polynomial-query adversary because the subversion at each point is determined by a local adaptive computation.

2 The Model: Crooked Indifferentiability

The primitives that we focus on in this paper are random permutations. A random permutation is an ideal primitive which provides an independent random output for each new query so that the resulting function is a permutation. We next extend the model of *crooked indifferentiability* [20] for random oracles [6] to capture the setting of random permutations.

Crooked indifferentiability for random permutations. As mentioned in the introduction, we consider the problem of "repairing" a subverted random permutation (or function directly) in such a way that the corrected construction can be used as a drop-in replacement for an unsubverted random permutation. Same as [20], we model the act of *subversion*

---

[6] The concept of crooked indifferentiability for random oracles was initially an extension of classical indifferentiability. We put the definition and properties of classical indifferentiability in Section A.1 of the Supplementary Materials.

of h as the creation of an "implementation" $\tilde{h}$ of the new, subverted permutation (or function); in practice, this would be the source code of the subverted version of the function h. In our setting, we define $\mathcal{A}$ as a polynomial-time algorithm with oracle access to h; thus the subverted function is $x \mapsto \mathcal{A}^h(x)$. Specifically, in Fig. 2,

1. The deterministic construction will have oracle access to the random permutation only via the subverted implementation $\tilde{h}$ but not via the ideal primitive h. (Operationally, the construction has oracle access to the function $x \mapsto \mathcal{A}^h(x)$.) The construction depends on access to trusted, but public, randomness R.
2. The simulator is provided, as input, a description of the subversion algorithm $\mathcal{A}$ (a Turing machine) and the public randomness R; it has oracle access to the target ideal functionality ($\mathcal{F}$, here is a random permutation).

Point (2) is necessary, and desirable, as it is clearly impossible to achieve indifferentiability using a simulator that has no access to $\mathcal{A}$ (the distinguisher can simply query an input such that C will use a value that is modified by $\mathcal{A}$ while $\mathcal{S}$ has no way to reproduce this). As shown in [20], such an extended notion can also enjoy a replacement theorem (see Section A.3 in the Supplementary Materials.)
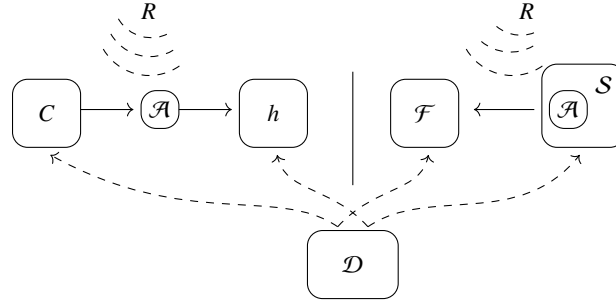

Fig. 2: The crooked indifferentiability notion: the distinguisher $\mathcal{D}$, in the first phase, manufactures and publishes a subverted implementation denoted as $\tilde{h}$, for ideal primitive h; then in the second phase, a random string R is published; after that, in the third phase, algorithm C, and simulator $\mathcal{S}$ are developed; the crooked-distinguisher $\mathcal{D}$, in the last phase, either interacting with algorithm C and ideal primitive h, or with ideal primitive $\mathcal{F}$ and simulator $\mathcal{S}$, return a decision bit. Here, algorithm C has oracle access to $\tilde{h}$, while simulator $\mathcal{S}$ has a description of $\mathcal{A}$ and oracle access to $\mathcal{F}$.

Definition 1 (Crooked indifferentiability[20]). *We define the notion of crooked indifferentiability by the following experiment.*

REAL EXECUTION

1. *Consider a distinguisher $\mathcal{D}$ and the following multi-phase real execution. Initially, the distinguisher $\mathcal{D}$ commences the first phase: with oracle access to ideal primitive h the distinguisher constructs and publishes a* subverted implementation *of h; this subversion is described as a deterministic polynomial time algorithm denoted $\mathcal{A}$. (Recall that the algorithm $\mathcal{A}$ implicitly defines a*

---

subverted version of h by providing h to $\mathcal{A}$ as an oracle—thus $\mathcal{A}^h(x)$ is the value taken by the subverted version of h at x.) Then, a uniformly random string R is sampled and published.

2. *Then the second phase begins involving a deterministic construction C: the construction C requires the random string R as input and has oracle access to $\tilde{h}$ (the crooked version of h); explicitly this is the oracle $x \mapsto \mathcal{A}^h(x)$.*

3. *Finally, the distinguisher $\mathcal{D}$, now with random string R as input and full oracle access to the pair (C, h), returns a decision bit b. Often, we call $\mathcal{D}$ the crooked-distinguisher.*

IDEAL EXECUTION

1. *Consider now the corresponding multi-phase ideal execution with the same crooked-distinguisher $\mathcal{D}$. The ideal execution introduces a simulator $\mathcal{S}$ responsible for simulating the behavior of h; $\mathcal{S}$ is provided full oracle access to the ideal object $\mathcal{F}$. Initially, $\mathcal{S}$ must answer any queries made to h by $\mathcal{D}$ in the first phase. Then $\mathcal{S}$ is given the random string R and the algorithm $\langle \mathcal{A} \rangle$ (generated by $\mathcal{D}$ at the end of the first phase) as input.*

2. *In the second phase, the crooked-distinguisher $\mathcal{D}$, now with random string R as input and oracle access to the alternative pair $(\mathcal{F}, \mathcal{S})$, returns a bit b.*

We say that construction C is $(n_{\text{source}}, n_{\text{target}}, q_{\mathcal{D}}, q_{\mathcal{A}}, \epsilon)$-crooked-indifferentiable from ideal primitive $\mathcal{F}$ if there is an efficient simulator $\mathcal{S}$ so that for any crooked-distinguisher $\mathcal{D}$ making no more than $q_{\mathcal{D}}(n)$ queries and producing a subversion $\mathcal{A}$ making no more than $q_{\mathcal{A}}(n)$ queries, the real execution and the ideal execution are indistinguishable. Specifically,

$$\left| \Pr_{u,R,h} \left[\tilde{h} \leftarrow \mathcal{D}^h(1^n) \;;\; \mathcal{D}^{C^{\tilde{h}}(R),h}(1^n, R) = 1 \right] - \right.$$

$$\left. \Pr_{u,R,\mathcal{F}} \left[\tilde{h} \leftarrow \mathcal{D}^h(1^n) \;;\; \mathcal{D}^{\mathcal{F}, \mathcal{S}^{\mathcal{F}}(R, \langle \tilde{h} \rangle)}(1^n, R) = 1 \right] \right| \leq \epsilon(n) \, .$$

Here R denotes a random string of length $r(n)$ and both $h : \{0, 1\}^{n_{\text{source}}} \to \{0, 1\}^{n_{\text{source}}}$ and $\mathcal{F} : \{0, 1\}^{n_{\text{target}}} \to \{0, 1\}^{n_{\text{target}}}$ denote random functions where $n_{\text{source}}(n)$ and $n_{\text{target}}(n)$ are polynomials in the security parameter n. We let u denote the random coins of $\mathcal{D}$. The simulator is efficient in the sense that it is polynomial in n and the running time of the supplied algorithm $\mathcal{A}$ (on inputs of length n_{source}). See Figure 2 for detailed illustration of the last phase in both real and ideal executions. (While it is not explicitly captured in the description above, the distinguisher $\mathcal{D}$ is permitted to carry state from the first phase to the second phase.) The notation $C^{\tilde{h}}(R)$ denotes oracle access to the function $x \mapsto \mathcal{A}^h(x)$.

Our main security proof will begin by demonstrating that in our particular setting, security in a simpler model suffices: this is the *abbreviated crooked indifferentiability* model, articulated in Section A.2 in the Supplementary Materials. We then show that—in

light of the special structure of our simulator—it can be effectively lifted to the full model above. Roughly speaking, the only difference between the full and abbreviated crooked-indifferentiability is that, in phase I of the abbreviated crooked indifferentiability model, the distinguisher can not query h(in the real execution) or $\mathcal{S}$(in the ideal execution) before it outputs the subversion algorithm.

3 Main Result and Technical Overview

3.1 The Construction and Main Result

For a security parameter n and a (polynomially related) parameter ℓ, the construction depends on public randomness $R = ((a_1, b_1), \dots, (a_\ell, b_\ell))$.

The source function is expressed as a family of ℓ independent random oracles:

$$F_i : \{0, 1\}^n \to \{0, 1\}^n, \qquad \text{for } i \in \{1, \dots, \ell\}.$$

These can be realized as slices of a single random function $F' : \{0, 1\}^{n'} \to \{0, 1\}^{n'}$, with $n' = n + \lceil \log \ell + 1 \rceil$ by an appropriate convention for embedding and extracting inputs and values. (Note that the F_i will not generally be permutations.) The family of ℓ public, uniform affine-linear functions are determined by $R = ((a_1, b_1), \dots, (a_\ell, b_\ell))$ where $(a_i, b_i) \in \mathsf{GL}(\mathbb{F}_2, n) \times \mathbb{F}_2^n$ for each $i = 1, \dots, \ell$. a_i and b_i are selected independently and uniformly from $\mathsf{GL}(n, \mathbb{F}_2)$ (to be concrete, the collection of invertible $n \times n$ matrices with elements in $\mathbb{F}_2$) and $\mathbb{F}_2^n$, respectively. The i-th affine linear function, defined on an input $x \in \mathbb{F}_2^n$, is given by the rule $x \mapsto a_i \cdot x \oplus b$. The final construction is given by the Feistel construction applied to the round functions $x \mapsto \tilde{F}_i(a_i \cdot x \oplus b)$, where $\tilde{F}$ is the subverted version of the function F_i. To be concrete, with the data (F_i, a_i, b_i) (with $i = 1, \dots, \ell$), the construction $C : \{0, 1\}^{2n} \to \{0, 1\}^{2n}$ is defined by the rule

$$C(x_0, x_1) := (x_\ell, x_{\ell+1}), \text{ where}$$
$$x_{i+1} := x_{i-1} \oplus \tilde{F}_i(a_i \cdot x_i \oplus b_i), \text{ for } i = 1, \dots, \ell,$$

where n-bit strings x and b_i are viewed as length n column vectors, $a_i \cdot x_i$ is the multiplication between matrix a_i and column vector x_i, and $\tilde{F}_i(x)$ is the subverted function value at (i, x) using the subversion algorithm $\mathcal{A}$. A visual illustration of the construction can be obtained by substituting the family of the round functions F_i in Fig. 1 with $\tilde{F}_i(a_i \cdot x \oplus b)$.

We wish to show that such a construction is indifferentiable from an actual random permutation (with the proper input/output length).

Theorem 1. *We treat a function $F' : \{0, 1\}^{n'} \to \{0, 1\}^{n'}$, with $n' = n + \lceil \log \ell + 1 \rceil$, as implicitly defining a family of random oracles*

$$F_i : \{0, 1\}^n \to \{0, 1\}^n, \qquad \text{for } i > 0,$$

by treating $\{0, 1\}^{n'} = \{0, \dots, L - 1\} \times \{0, 1\}^n$ and defining $F_i(\cdot) = F(i, \cdot)$, for $i = 0, \dots, \ell \leq L - 1$. (Output lengths are achieved by removing the appropriate number of

trailing symbols). Consider a (subversion) algorithm $\mathcal{A}$ so that it defines a subverted $\tilde{F}$. Assume that for every F (and every i),

$$\Pr_{x \in \{0,1\}^n}[\tilde{F}_i(x) \neq F_i(x)] \leq \epsilon(n) = \mathsf{negl}(n) . \tag{1}$$

For $\ell \geq 337n/\log(1/\epsilon)$, the above Feistel-based construction is $(n', 2n, q_\mathcal{D}, q_\mathcal{A}, \epsilon')$-indifferentiable from a random permutation $P : \{0,1\}^{2n} \to \{0,1\}^{2n}$, where $q_\mathcal{D}$ is the number of queries made by the distinguisher $\mathcal{D}$, $q_\mathcal{A}$ is the number of queries made by $\mathcal{A}$ as in Definition 1 and $\epsilon' = (22q_\mathcal{D}(q_\mathcal{A} + 1))^3/2^n$. Both $q_\mathcal{D}$ and $q_\mathcal{A}$ are polynomial functions of n, ensuring ϵ' is negligibly small.

Remark. For some technical reasons, we need the round number parameter ℓ to be at least $337n/\log(1/\epsilon)$. A more careful choice of parameters in the proof could potentially reduce the constant factor to below 200. When considering $\ell = 337n/\log(1/\epsilon)$, a particularly intriguing scenario arises when $\epsilon = 2^{-cn}$ for some constant $0 < c < 1$. In this case, ℓ becomes a constant value of $337/c$.

To somewhat simplify the notation, we define the function $CF_i : \{0,1\}^n \to \{0,1\}^n$ to be the unsubverted analog of the round function $CF_i(x) = F_i(a_i \cdot x \oplus b_i)$ and, similarly, define $C\tilde{F}_i(x) = \tilde{F}_i(a_i \cdot x \oplus b_i)$ to be actual round function. Since the function $x \to a_i \cdot x \oplus b_i$ is a permutation (note that a_i is an invertable linear function), reasoning about CF_i (and $C\tilde{F}_i$, respectively) is effectively equivalent to reasoning about F_i (and CF_i) for the bulk of the paper (i.e., we will treat the query and evaluation of $F_i(x)$ as the query and evaluation of $CF_i(x')$ such that $x = a_i \cdot x' \oplus b_i$). When evaluating $C\tilde{F}_i(x)$, the subversion algorithm queries CF at a set of points of polynomial size. We define the set of these points to be

$$Q_i(x) = \{(j, x') \mid \text{the evaluation of } C\tilde{F}_i(x) \text{ queries } CF_j(x')\} .$$

3.2 $2n/\log(1/\epsilon)$ rounds are not enough

We first show that the above construction is insecure with fewer than $2n/\log(1/\epsilon)$ rounds.

Lemma 1. *Let n be a positive integer and ϵ be a real number with $1/16 \geq \epsilon \geq 2^{-n}$. Let ℓ be a positive integer not greater than $2n/\log(1/\epsilon)$ and let $\lambda = \lfloor n/\ell + 1 \rfloor$. Consider selecting a uniform vector $B \in \mathbb{F}_2^{\lambda\ell/2}$ and a $\lambda\ell/2$ by n matrix $A = (C_1, ..., C_{\ell/2})^T$, where each C_i is a uniform full rank matrix in $\mathbb{F}_2^{n\times\lambda}$. Then, over the randomness of the choice of A and B,*

$$\Pr\left[\text{There does not exist a vector } X \in \mathbb{F}_2^n \text{ such that } A \cdot X = B.\right] = O(2^{-n/4}),$$

where $A \cdot X$ is the multiplication between a matrix and a column vector and B is viewed as a column vector.

Proof. Notice that it suffices to show that the matrix A chosen above has full rank with probability $1 - O(2^{-n/4})$. Rather than proving this, we establish a stronger statement by regarding A as a uniform matrix in $\mathbb{F}_2^{\lambda\ell/2\times n}$.

It is worth noting that $\lambda\ell/2 \leq (n+\ell)/2 < (n+n/2)/2 = 3n/4$. Thus, it is adequate to demonstrate that a uniform $3n/4$ by n binary matrix A' has full rank with probability $1 - O(2^{-n/4})$.

For any $i = 1, \ldots, \lambda\ell/2$, we denote by w_i the ith row vector of A' and W_i the set of first i row vectors of A'. For a set S of vectors, we use $\langle S \rangle$ to denote the vector space spanned by the elements of S.

Then, over the uniform choice of A', we have

$$\Pr[A' \text{ does not have full rank}]$$

$$\leq \sum_{i=1}^{3n/4} \Pr[w_i \in \langle W_{i-1} \rangle]$$

$$\leq \sum_{i=1}^{3n/4} |\langle W_{i-1} \rangle|/2^n$$

$$= \sum_{i=1}^{3n/4} 2^{i-1}/2^n = O(2^{-n/4}).$$

Theorem 2. *The construction is not crooked-indifferentiable from a random permutation if $\ell \leq 2n/\log(1/\epsilon)$.*

Proof. Let $\lambda = \lfloor n/\ell + 1 \rfloor$ (so $\lambda \geq n/\ell$). Consider the following subversion algorithm $\mathcal{A}$: for each F_i ($i = 1, \ldots, \ell$) and any n bit string x, define $\tilde{F}_i(x) := 0^n$ if the first λ bits of x are 0s. Otherwise, define $\tilde{F}_i(x) := F_i(x)$. (Notice that this subversion algorithm is legitimate since the dishonest fraction is $2^{-\lambda} \leq 2^{-n/\ell} \leq \epsilon$.)

Now we prove the distinguisher can launch the following attack with the subversion algorithm above. We will show that, with overwhelming probability over the choice of R, there is a pair of n-bit strings (x_0, x_1) such that for the Feistel chain $(x_1, x_2, \ldots, x_\ell)$ related to (x_0, x_1), $C\tilde{F}_i(x_i) = 0^n$ for all $i = 1, \ldots, \ell$. (We use the terminology "with overwhelming probability" in the paper to mean "with all but negligible probability.")

Notice that the fact that such a pair (x_0, x_1) exists is equivalent to the fact that there is a pair (x_0, x_1) such that the first λ bits of $a_{2i+1} \cdot x_1 \oplus b_{2i+1}$ and the first λ bits of $a_{2j} \cdot x_0 \oplus b_{2j}$ are 0s for all $0 < 2i+1, 2j \leq \ell$. And this is true with constant probability due to Lemma 1. (Also, the attack can be launched by a polynomial running time adversary since the linear equations in Lemma 1 can be solved efficiently.)

3.3 Technical Overviews and Notations

In this section we give a technical overview of proving Theorem 1.

Our strategy: Simulation via judicious preemptive chain completion. To convey the main idea, suppose that a distinguisher queries the *simulated* round functions in order to determine the value of the permutation P on input $(x_0, x_1) \in \{0, 1\}^{2n}$; in particular, the resulting output $(x_\ell, x_{\ell+1})$ is obtained by computing $x_{i+1} := x_{i-1} \oplus C\tilde{F}_i(x_i)$ for all $i = 1, \ldots, \ell$. Then, $(x_\ell, x_{\ell+1})$ must equal the output of P on input (x_0, x_1): otherwise the distinguisher can easily detect that it is not interacting with the real Feistel construction.

To ensure such consistency, the simulator must recognize that the queries $x_1, \ldots, x_\ell$ belong to an evaluation of C, and must set the values $CF_i(x_i)$ to enforce consistency with P. This mechanism is described informally below and in more detail in the next section.

The simulator maintains an internal table for each function CF_i that indicates a partial definition of this function: these tables typically expand during interaction with the distinguisher and satisfy the invariant that once a CF_i value is defined in the table for a particular element x of the domain, this is never removed or altered later in the computation. While the tables define the CF_i values used to respond to any query answered by the simulator, the table may record additional CF_i values not as yet queried by the distinguisher as a bookkeeping tool. Of course, distinguisher queries are always answered consistently with the values in the tables.

Subverted and unsubverted chains; honest chains. In the following, an index s, combined with a sequence of values $x_s, \ldots, x_{s+r}$ ($r \geq 1$, $1 \leq s < s + r \leq \ell$) such that $CF_i(x_i)$ is defined by the simulator for all $s \leq i \leq s + r$ and such that $x_{i+1} := x_{i-1} \oplus CF_i(x_i)$ for all $s + 1 \leq i \leq s + r - 1$, will be called an *unsubverted chain*(denoted by $(s, x_s, \ldots, x_{s+r})$). For each index i and an element $x \in S.CF_i$, we say $C\tilde{F}_i(x)$ is *defined* if its value can be determined by the subversion algorithm and the CF values that are already defined by the simulator. We assume without loss of generality that the subversion algorithm always evaluates $CF_i(x)$ when called upon to evaluate $C\tilde{F}_i(x)$ (for any i and x). Therefore, $CF_i(x)$ must be defined when $C\tilde{F}_i(x)$ is defined. An index s, combined with a sequence of values $x_s, \ldots, x_{s+r}$ ($r \geq 1$, $1 \leq s < s + r \leq \ell$) such that $C\tilde{F}_i(x_i)$ is defined by the simulator for all $s \leq i \leq s + r$, and such that $x_{i+1} := x_{i-1} \oplus C\tilde{F}_i(x_i)$ for all $s + 1 \leq i \leq s + r - 1$, will be called a *subverted chain*. The *length* $L(\cdot)$ of an unsubverted (or subverted) chain is defined to be the number of the elements in the chain. An unsubverted (or subverted) chain is called a *full chain* if it has length ℓ. Note, in general, that chains always have length of at least two (as $r \geq 1$).

For a chain $c = (s, x_s, \ldots, x_{s+r})$, we define $Q_c = \bigcup_{i=s}^{s+r} Q_i(x_i)$ if $C\tilde{F}_i(x_i)$ is defined for $i = s, \ldots, s + r$. For any $i \in \{1, \ldots, \ell\}$ and $x \in \{0, 1\}^n$, if $C\tilde{F}_i(x)$ is defined, we say (i, x) is *honest* when $CF_i(x) = C\tilde{F}_i(x)$; similarly, we say it is *dishonest* when $CF_i(x) \neq C\tilde{F}_i(x)$. We say a subverted chain is *honest* if all the elements on the chain are honest.

For a chain $c = (s, x_s, \ldots, x_{s+r})$ and a term (i, x), we say (i, x) is an element of c(or $(i, x) \in c$) if $s \leq i \leq s + r$ and $x_i = x$. For two chains $c_1 = (s_1, x_{s_1}, \ldots, x_{s_1+r_1})$ and $c_2 = (s_2, y_{s_2}, \ldots, y_{s_2+r_2})$, we say $c_1 \subset c_2$ if each element of c_1 is also an element of c_2. We say c_1 and c_2 are *disjoint* if there is no chain c for which $c \subset c_1$ and $c \subset c_2$ (i.e., the chains c_1 and c_2 do not share any pair of adjacent elements).

The definition of the simulator S. Our simulation strategy will consider a carefully chosen set of relevant unsubverted chains as "triggers" for completion: once a chain of this family is defined in the simulator's table, the simulator will preemptively "complete" the chain to ensure consistency of the resulting full chain with P. Upon a query for CF_i with input x_i (in fact, the query is a query for F_i with input x_i' such that $a_i \cdot x_i' \oplus b_i = x_i$), the simulator sets $CF_i(x_i)$ to a fresh random value and looks for new relevant partial chains involving x_i, adding them to a FIFO queue. (In general, many new chains may be added by this process.) The simulator then repeats the following, until the queue is empty: It removes the first unsubverted chain from the queue. If the chain satisfies a

certain property (will be described later), the simulator *completes* it to a full subverted chain $x_1, \ldots, x_\ell$ in such a way so as to guarantee that $P(x_0, x_1) = (x_\ell, x_{\ell+1})$, where $x_0 = x_2 \oplus C\tilde{F}_1(x_1)$ and $x_{\ell+1} = x_{\ell-1} \oplus C\tilde{F}_\ell(x_\ell)$. In particular, it sets each undefined CF in $Q_i(x_i)$ to a fresh uniform random string, with the exception of two consecutive values $CF_u(x_u)$ and $CF_{u+1}(x_{u+1})$ which are set adaptively to ensure consistency with P. We refer to this step as *adapting* or *programming* the values of $CF_u(x_u)$ and $CF_{u+1}(x_{u+1})$. Establishing that such adapting is always possible (for some carefully chosen u) will be a major challenge of our analysis below.

Technical Challenges. We now face two main challenges. Our choice of which chains are relevant and how they are completed will be crucial in order to solve them:

1. **Freshness and Honesty.** We need to show that the values of $CF_u(x_u)$ and $CF_{u+1}(x_{u+1})$ are always undefined when these values are selected for programming. Moreover, we hope the two terms (u, x_u) and $(u + 1, x_{u+1})$, which are adapted to ensure consistency are always honest; i.e., $CF_u(x_u) = C\tilde{F}_u(x_u)$ and $CF_{u+1}(x_{u+1}) = C\tilde{F}_{u+1}(x_{u+1})$.

2. **Efficiency.** We need to show that the simulation terminates with high probability when answering a query; i.e., the queue becomes empty after a small (polynomial) number of completions.

3. **Indistinguishability.** Finally, with the two demands above in hand, it is still necessary to show that the simulated world cannot be distinguished from the real world.

Addressing Challenge 1. To understand why proving freshness and honesty is hard, consider the following example. During the interaction with the distinguisher, suppose the simulator $\mathcal{S}$ sees an unsubverted chain $c = (s, x_s, \ldots, x_{s+r})$ that triggers completion. Let us call the current $\mathcal{S}.CF$ table T_{Initial}. For the full chain $c' = (1, x_1, \ldots, x_\ell)$ determined by c ($c \subset c'$), $\mathcal{S}$ hopes that it can find an index u so that (u, x_u) and $(u + 1, x_{u+1})$ are undefined before adaption. It is easy to find an index u so that these two terms are not in T_{Initial}. However, before $\mathcal{S}$ determines x_u and x_{u+1}, it needs to evaluate $C\tilde{F}_i(x_i)$ for $i \neq u, u + 1$. And there may exist an index i so that (u, x_u) or $(u + 1, x_{u+1})$ is in $Q_i(x_i)$, which breaks the freshness. It is also not obvious how to find u so that (u, x_u) and $(u + 1, x_{u+1})$ are honest since the distinguisher can subvert the round functions of any index. In our analysis, we will have to find $u, u + 1$ such that *both* freshness and honesty can be satisfied.

To prove honesty, we will show that for any term (i, x_i) in c', it is honest if i is much smaller than s or much greater than $s + r$ (i.e., the term is far away from the initial chain c that triggers completion). Therefore, there is a long subchain c'' of c' that is honest. The simulator will select the index u in this honest area. To prove freshness, we will show that, inside the long enough honest chain c'', for any term (j, x_j) in the "middle area" of c'' and any term $(i, x_i) \in c'$ with $i \neq j$, $CF_j(x_j)$ is not queried by $C\tilde{F}_i(x_i)$(i.e., $(j, x_j) \notin Q_i(x_i)$). To achieve freshness, the simulator only needs to pick u in the middle part of the honest area.

Addressing Challenge 2. To see why it is possible the queue may not become empty after a small number of completions, notice that the completion of a certain chain forces the evaluation of many terms that have not been queried by the distinguisher. These newly evaluated terms may generate another chain that triggers completion. The same efficiency problem also appears in the proof of classical indifferentiability of a

constant round Feistel construction. (See Coron et al. [10]) The efficiency problem in our case (the crooked-indifferentiability model) is more complex than that in [10] (the classical indifferentiability model) because when completing a chain in the crooked-indifferentiability model, the simulator needs to evaluate $C\tilde{F}$ instead of just CF values in the chain, which in general, generates many more terms than the classical model.

To prove efficiency, we will show that the recursion stops after at most $\text{poly}(q_{\mathcal{D}})$ steps, where $q_{\mathcal{D}}$ is the number of the queries made by the distinguisher $\mathcal{D}$. The proof relies on the observation that, for the chains that are completed, on average, all but a constant number of elements in each chain were once queried by $\mathcal{D}$. (Notice that not all the elements in these chains are evaluated because they are queried by $\mathcal{D}$. For example, some elements are evaluated when the simulator completes a chain.) Hence, the total number of the chains that are completed is in fact asymptotically equivalent to $q_{\mathcal{D}}/\ell$.

Addressing Challenge 3. It is still not easy to establish crooked-indifferentiability after we understand freshness, honesty, and efficiency. The reason is that the *CF* values that are maintained by $\mathcal{S}$ are not perfectly uniform conditioned on the distinguisher's query to the ideal object P, which is a crucial property in the proofs of efficiency, freshness and honesty.

To see why the *CF* values held by $\mathcal{S}$ are not perfectly uniform, imagine that the distinguisher queries $P(x_0, x_1)$ for some (x_0, x_1) and then makes several *CF* queries to trigger the completion of the chain corresponding to (x_0, x_1). The two adapted values $CF_u(x_u)$ and $CF_{u+1}(x_{u+1})$ are not uniform because they are, of course, adapted to maintain consistency.

To break down the proof, we introduce a sequence of game transitions involving 6 games, beginning with the simulator game (Game 1) and ending with the construction (Game 6). By mapping the randomness from one game to another, we prove that the gap between the 6 games is negligible if the gap between Games 5 and 6 is negligible. (In particular, we explain how to treat the games as coupled random variables that can be investigated with the same underlying randomness; this provides a convenient way to identify differences in the dynamics and conclusions of the games.) Then we turn our attention to Game 5, which maintains an explicit, additional table of uniform *CF* values. This table (in Game 5) provides a vantage point from which all future *CF* values are in fact uniform, and simplifies reasoning about many of the critical events of interest. Finally, we formally prove honesty and freshness in Game 5 to show the gap between Game 5 and 6 is negligible.

Technical differences between [10] and this paper. In [10] (the classical indifferentiability model), Coron et al. used a simulation strategy similar to ours—simulation via judicious preemptive chain completion—to demonstrate the classical indifferentiability of a constant round Feistel structure. Despite using similar simulation strategy, there are some significant technical differences between our security proof and the proof in [10].

1. **Freshness:** The proof of freshness is challenging in both [10] and our work, but for quite different reasons. The chains in [10] are very short (i.e., have only constant length), and when two of them are intersected, the terms of one chain can easily occupy the "adaptation space" for the other, which hinders freshness. In our case, however, we are not that worried about the intersection of chains since our construction has

many more than constant rounds. The difficulty of our freshness proof arises from the subversion algorithm: to prove freshness, we need to rule out the case that when completing a chain, the two adapted terms are queried by some previously evaluated $C\tilde{F}$.

2. **Honesty:** In the security proof of [10], the authors only need to show freshness and efficiency of the simulation since there is no subversion; they are not required to prove honesty.

3. **Efficiency:** The efficiency proof in [10] is quite straightforward. By contrast, in our case, it is not that obvious how to upper bound the number of the terms generated in the simulation. The difference is again due to the existence of the subversion algorithm. In our case, the chains that are completed are subverted chains, while the classical case has no subversion algorithm and therefore only completes "unsubverted" chains. The evaluation of a subverted chain generates many more terms than the evaluation of an unsubverted chain, which in general, may generate many more chains that trigger completion.

4 Security Proof

In the rest of the paper, we turn the explanation above into a real proof. We first introduce the detailed definition of the simulator.

4.1 The Detailed Definition of the Simulator

The simulator provides an interface $\mathcal{S}.CF(i,x)$ to query the simulated random function CF_i on input x. As mentioned above, for each i the simulator internally maintains a table whose entries are pairs (x,y) of n-bit strings; each such entry intuitively determines a simulated value of CF at a particular point: in particular, if the pair (x,y) appears then any query to $\mathcal{S}.CF(i,x)$ returns the value y. The simulator maintains the natural invariants described previously: responses provided to the distinguisher are always consistent with the table; furthermore, once an entry has been added to the table, it is never removed or changed. Note that in many cases the table will reflect function values that have not been queried by the distinguisher. We denote the ith table by $\mathcal{S}.CF_i$ and write $x \in \mathcal{S}.CF_i$ whenever x is a preimage in this table, often identifying $\mathcal{S}.CF_i$ with the set of preimages stored. When $x \in \mathcal{S}.CF_i$, $CF_i(x)$ denotes the corresponding image. $\mathcal{S}.CF$ is the collection of all these $\mathcal{S}.CF_i$ tables. We use the notation $(i,x) \in \mathcal{S}.CF$ when $x \in \mathcal{S}.CF_i$.

For each i, we additionally define a table $\mathcal{S}.C\tilde{F}_i$ induced implicitly by $\mathcal{S}.CF$. As with $\mathcal{S}.CF_i$, the table $\mathcal{S}.C\tilde{F}_i$ consists of pairs of inputs and outputs of $C\tilde{F}_i$. We write $x \in \mathcal{S}.C\tilde{F}_i$ when all queries generated by evaluation of $C\tilde{F}_i(x)$ are defined in $\mathcal{S}.CF$; naturally, the corresponding function value determines the pair (x,y) in the table. The collection of all of these $\mathcal{S}.C\tilde{F}_i$ is denoted by $\mathcal{S}.C\tilde{F}$. (Note that this table is not maintained explicitly by the simulator, but rather determined implicitly by $\mathcal{S}.CF$.)

Handling queries to $\mathcal{S}.CF$. On a query $\mathcal{S}.CF(i,x)$, the simulator first checks whether $x \in \mathcal{S}.CF_i$. If so, it answers with $CF_i(x)$. Otherwise the simulator picks a random value y and inserts (x,y) into $\mathcal{S}.CF_i$. (The process above is done by a procedure called

$\mathcal{S}.CF^{\text{Inner}}$ which takes input (i, x).) After this, the simulator takes further steps to ensure that its future answers are consistent with the permutation P. Only after this consistency maintenance step is the value y finally returned.

To ensure consistency, the simulator considers all newly generated unsubverted chains with length $\ell/20$ that terminate at the last-queried position; for a newly evaluated term $CF_s(x_s)$, these chains of interest either have the form $(s, x_s, ..., x_{s+\ell/20-1})$ or $(s - \ell/20 + 1, x_{s-\ell/20+1}, ..., x_s)$. Each such detected chain is enqueued by the simulator in a "completion queue," identifying the chain for future completion.

The simulator then repeats the following detection and completion step until the queue is emptied. (When the queue is finally empty, the simulator returns the answer y to the initial query.)

1. **Detection Step.** The first chain $c = (s, x_s, ..., x_{s+\ell/20-1})$ is removed from the queue. A procedure called $\mathcal{S}$.HonestyCheck is then run on the chain. The procedure $\mathcal{S}$.HonestyCheck evaluates $C\tilde{F}$ values of the elements of c and generates a four-tuple (s, x_s, x_{s+1}, u) for future completion if all the elements in c are honest. (In fact, not all chains removed from the queue are processed by $\mathcal{S}$.HonestyCheck. A chain removed from the queue is processed by $\mathcal{S}$.HonestyCheck only if it is disjoint with all the chains that are previously processed by $\mathcal{S}$.HonestyCheck and is disjoint with all the previously completed full subverted chains. Any chain that is not processed by $\mathcal{S}$.HonestyCheck is discarded. The procedure that decides whether a chain is going to be discarded or processed by $\mathcal{S}$.HonestyCheck is called $\mathcal{S}$.Check.) In the tuple (s, x_s, x_{s+1}, u), the value s ensures that later the simulator knows that the first value x_s corresponds to CF_s. The value u describes where to adapt (that is, program) the values of CF in order to ensure consistency with the given permutation: this will occur at positions u and $u + 1$. The convention for determining u is straightforward: If $s > 3\ell/4$ or $s + \ell/20 - 1 < \ell/4$, then there is "plenty of space around $\ell/2$," and $u = \ell/2$; otherwise, $u = \ell - 10$.

2. **Completion Step.** Finally, the simulator takes the four-tuple (s, x_s, x_{s+1}, u) and *completes* the subverted chain related to (s, x_s, x_{s+1}). Intuitively, this means that the chain is determined by iteratively determining neighbouring values of $C\tilde{F}(x)$ by evaluating the subversion algorithm and, when necessary, carrying out internal calls to $CF_i()$ in order to answer queries made by that algorithm to the F_i. This iterative process is continued, using P to "wrap around," until the only remaining undetermined values appear at positions u and $u + 1$; at this point, the values at u and $u + 1$ are programmed to ensure consistency. In more detail: Assuming that $u < s$, the completion process (conducted by a procedure called $\mathcal{S}$.Complete) proceeds as follows.

   – The initial chain consists of the two adjacent values x_s, x_{s+1}.
   – $C\tilde{F}_{s+1}(x_{s+1})$ is determined by simulating the subversion algorithm which generates oracle queries to CF to be answered using $\mathcal{S}.CF$. (Note that this process may enqueue new chains for completion.) The value $x_{s+2} = x_s \oplus C\tilde{F}_{s+1}(x_{s+1})$ is then determined, yielding the enlarged chain (x_s, x_{s+1}, x_{s+2}). This process is repeated until the chain is extended maximally "to the right" so that it has the form $(x_s, x_{s+1}, \ldots, x_\ell, x_{\ell+1})$.
   – P^{-1} is then applied to $x_\ell, x_{\ell+1}$ to yield x_0, x_1.

16

- Starting from the pair (x_0, x_1), this process is repeated, as above, to yield values for $x_2, \ldots, x_u$. Note that $x_u = x_{u-2} \oplus C\tilde{F}(x_{u-1})$ so that $C\tilde{F}(x_u)$ is never evaluated during this process (which is to say that the subversion algorithm is never simulated on x_u).
- Similarly, the original pair x_s, x_{s-1} is extended "to the left" to determine the values $x_{s-1}, \ldots, x_{u+1}$; as above, x_{u+1} is determined by $x_{u+3} \oplus C\tilde{F}(x_{u+2})$, so that $C\tilde{F}(x_{u+1})$ is never evaluated.
- Then, the simulator defines $CF_u(x_u)$ and $CF_{u+1}(x_{u+1})$ that is consistent with P, i.e., $CF_u(x_u) := x_{u-1} \oplus x_{u+1}$ and $CF_{u+1}(x_{u+1}) := x_u \oplus x_{u+2}$. The game aborts if either of these is defined from a previous action of $\mathcal{S}$. If the game does not abort, the simulator evaluates the subversion algorithm on both x_u and x_{u+1}. During this evaluation, the values $CF_u(x_u)$ and $CF_{u+1}(x_{u+1})$ are already determined; other queries are answered using $\mathcal{S}.CF$ as above. The game aborts if (u, x_u) or $(u+1, x_{u+1})$ is dishonest; otherwise, the chain is a valid subverted chain (and consistent with P).
- A set $\mathcal{S}.\mathsf{CompletedChains}$ is maintained to store the chains that are completed: for any (i, x_i, x_{i+1}) $(1 \le i \le \ell - 1)$, $\mathcal{S}$ updates

$$\mathcal{S}.\mathsf{CompletedChains} := \mathcal{S}.\mathsf{CompletedChains} \cup (i, x_i, x_{i+1}).$$

The alternative case, when $u > s + 1$, is treated analogously.

4.2 Plan of the Proof

To establish crooked indifferentiability, we need to prove that, from the perspective of $\mathcal{D}$, interacting with $(P, \mathcal{S}^P)$ (the ideal world) is indistinguishable from interacting with (C^F, F) (the real world).

Recall that we have three challenges in the security proof:

1. **Freshness and Honesty.** We need to show that the values of $CF_u(x_u)$ and $CF_{u+1}(x_{u+1})$ are always undefined when these values are selected for programming. Moreover, we hope the two terms (u, x_u) and $(u+1, x_{u+1})$, which are adapted to ensure consistency are always honest; i.e., $CF_u(x_u) = C\tilde{F}_u(x_u)$ and $CF_{u+1}(x_{u+1}) = C\tilde{F}_{u+1}(x_{u+1})$.
2. **Efficiency.** We need to show that the simulation terminates with high probability when answering a query; i.e., the queue becomes empty after a small (polynomial) number of completions.
3. **Indistinguishability.** Finally, with the two demands above in hand, it is still necessary to show that the simulated world cannot be distinguished from the real world.

Let us define the event that $\mathcal{S}$ aborts as Abort. According to the description of $\mathcal{S}$, Abort happens only when the distinguisher $\mathcal{D}$ finds a chain $(1, x_1, \ldots, x_\ell)$ such that the programmed term, (u, x_u) or $(u+1, x_{u+1})$, has been evaluated before it is programmed or is dishonest. It is easy to see that maintaining freshness and honesty is synonymous with preventing $\mathcal{S}$ from aborting. We will stick to the following plan of the proof to address these challenges.

1. In the first two steps of the proof, we begin by assuming that Challenge 2 has been adequately dealt with, allowing us to focus on resolving Challenges 3 and 1. First, in

17

Section 4.3, we aim to establish that resolving Challenge 1 enables us to address Challenge 3. Put differently, we will demonstrate that our construction is crooked indifferentiable if Abort happens negligibly.

2. Second, in Section 4.4, we address Challenge 1 under the assumption that Challenge 2 has been successfully addressed. In Theorem 4, we will establish that the likelihood of Abort occurring is negligible given that $\mathcal{S}$ is *efficient*, which means, with overwhelming probability, only a polynomial number of terms are evaluated by $\mathcal{S}$ (or P) when $\mathcal{D}$ interacts with $(P, \mathcal{S}^P)$.

3. Last, in Section 4.5, we will address Challenge 2 by showing the efficiency of $\mathcal{S}$ in Theorem 5.

A simplified proof. Unfortunately, due to space limitations, we can only provide a "simplified" proof (which is the plan above) in the main body. A complete and more rigorous proof is put in Section B of the Supplementary Materials. In the simplified proof, we omit less critical details while retaining a focus on the primary aspects relevant to the core argument. To assure readers that the essential concepts and outcomes from the complete proof are preserved in the simplified version, we will outline the structure of the complete proof and provide a concise explanation of the distinctions between the two versions.

Compare the complete and simplified proof. In the complete proof, we deal with Challenge 3 by developing a "game transition approach". We introduce four intermediate games to build the connection between the ideal world (the interaction between $\mathcal{D}$ and $(P, \mathcal{S}^P)$) and the real world (the interaction between $\mathcal{D}$ and (C^F, F)). Using the game transition, we clearly analyze the gaps between adjacent games. Summing up these gaps gives the gap between the ideal and real world, which is bounded by the probability of two bad events, $\mathsf{BadComplete}_5$ and $\mathsf{BadEval}_5$. The first and the major bad event $\mathsf{BadComplete}_5$ is same as the bad event Abort we defined above. (In fact, there is a little difference between $\mathsf{BadComplete}_5$ and Abort. $\mathsf{BadComplete}_5$ is defined in one of the four intermediate games, while Abort is defined in the interaction between $\mathcal{D}$ and $(P, \mathcal{S}^P)$. Otherwise, the two bad events are same and we can use the same proof to show their probabilities are negligible.) The second and the auxiliary bad event $\mathsf{BadEval}_5$ is derived from the game transition, which is used to make the proof rigorous. The missing part in the simplified proof are the four intermediate games in the game transitions and the proof that bounds the probability of $\mathsf{BadEval}_5$, which is an auxiliary event.

Although we omit the details of the four intermediate games in the simplified proof, we will provide a concise overview of the central ideas underpinning the game transition approach. This summary will explain why crooked indifferentiability can be reduced to the negligibility of Abort. (See Section 4.3) We also want to stress that the efficiency proof of the simulator in the simplified proof (Section 4.5) is same as that in the complete proof.

4.3 Relating Crooked Indifferentiability to the Bad Event

To understand how crooked indifferentiability is related to the probability of Abort, we consider a situation where we need to "complete" a chain in the ideal world $(P, \mathcal{S}^P)$ and in the real world (C^F, F).

Suppose in both worlds, we start with an initial table of CF values T_{initial}. Suppose there is an unsubverted chain $c = (s, x_s, ..., x_{s+\ell/20-1})$ in T_{initial} that has passed the test of $\mathcal{S}.\text{HonestyCheck}$, which means that all the elements of c are honest. (Without loss of generality, we assume $s + \ell/20 - 1 < \ell/4$. This means, when $\mathcal{S}$ completes c, it adapts the value of CF at $u = \ell/2$.) Now we want to see the gap between the two worlds when generating a subverted full chain c' that contain c.

In the ideal world $(P, \mathcal{S}^P)$, what $\mathcal{S}$ does is the following **Procedure 1**:

1. Generate $\tilde{CF}$ values before the adaption position $\ell/2$ by uniformly selecting CF values as needed: For $i = 2, \ldots, \ell/2$, generate (i, x_i) recursively by defining $x_i := x_{i-2} \oplus \tilde{CF}_{i-1}(x_{i-1})$ for $2 \le i \le \ell/2$ (each CF as needed is evaluated uniformly).
2. Generate $\tilde{CF}$ values after the adaption position $\ell/2 + 1$ by querying P and uniformly selecting CF values as needed: Query P at (x_0, x_1) and receive an (almost) uniform pair of n-bit strings $(x_\ell, x_{\ell+1})$. To generate (i, x_i) $(i = \ell/2, \ldots, \ell)$, recursively define $x_{i-2} := x_i \oplus \tilde{CF}_{i-1}(x_{i-1})$ for $\ell/2 + 3 \le i \le \ell + 1$ (each CF as needed is evaluated uniformly).
3. Adapt CF values at the adaption positions $\ell/2$ and $\ell/2 + 1$: Define $CF_u(x_{\ell/2}) := x_{\ell/2-1} \oplus x_{\ell/2+1}$ and $CF_{\ell/2+1}(x_{\ell/2+1}) := x_{\ell/2} \oplus x_{\ell/2+2}$. Evaluate $\tilde{CF}_{\ell/2}(x_{\ell/2})$ and $\tilde{CF}_{\ell/2+1}(x_{\ell/2+1})$ (each CF as needed is evaluated uniformly).
4. Abort if freshness or honesty is violated: The game aborts if there is an index j such that $\ell/4 \le j \le 3\ell/4$ and (j, x_j) is in T_{initial} or $\bigcup_{i=1}^{\ell} Q_i(x_i)/Q_j(x_j)$. The game also aborts if there is an index j such that $(\ell/2, x_{\ell/2})$ or $(\ell/2 + 1, x_{\ell/2+1})$ is dishonest.

In the real world (C^F, F), we extend c to a full chain by **Procedure 2**:

1. For $i = 2, \ldots, \ell + 1$, generate (i, x_i) recursively by defining $x_i := x_{i-2} \oplus \tilde{CF}_{i-1}(x_{i-1})$ for $2 \le i \le \ell + 1$ (each CF is evaluated uniformly).
2. Assign $P(x_0, x_1) = (x_\ell, x_{\ell+1})$.

To connect Procedure 1 to Procedure 2, we rewrite Procedure 1 as **Procedure 1'**:

1. For all $x \in \{0, 1\}^n$ and $\ell/2 + 2 \le i \le \ell$, evaluate $\tilde{CF}_i(x)$ (each CF is evaluated uniformly).
2. Same as step 2 of Procedure 1.
3. Same as step 3 of Procedure 1 except that no additional uniform CF values need to be selected because all needed CF values are already evaluated in Step 1.
4. Same as step 4 of Procedure 1.
5. Same as step 5 of Procedure 1.

Procedure 1 and Procedure 1' are equivalent in the sense that the resulting full chain c' in these two procedure are same.

Slightly changing Procedure 1' gives **Procedure 2'**:

1. Same as step 1 of Procedure 1'.
2. Same as step 2 of Procedure 1'.
3. Select $x_{\ell/2+1}$ and $x_{\ell/2+2}$ uniformly. To generate (i, x_i) (for $i = \ell/2 + 3, \ldots, \ell + 1$), define $x_i := x_{i-2} \oplus \tilde{CF}_{i-1}(x_{i-1})$ for $\ell/2+3 \le i \le \ell+1$. Assign $P(x_0, x_1) = (x_\ell, x_{\ell+1})$.
4. Same as step 4 of Procedure 1'.

Notice that step 3 of Procedure 2' is equivalent to that of Procedure 1' because the Feistel structure gives a permutation of $2n$-bit strings: selecting a uniform "input" string $(x_{\ell+1}, x_{\ell+2})$ is equivalent to selecting an uniform "output" string $(x_\ell, x_{\ell+1})$. (In fact, they are not perfectly equivalent since in Step 3 of Procedure 1', querying P at (x_0, x_1) does not give a perfectly uniform $(x_\ell, x_{\ell+1})$: P is a random permutation so $(x_\ell, x_{\ell+1})$ is chosen in a way to avoid collision. However, this only causes a negligible difference as we assume T_{Initial} contains a polynomial number of terms.)

Therefore, the only difference between Procedure 1' and Procedure 2' is that, in Step 5, Procedure 1' aborts when freshness or honesty is violated. And this is indicated by the occurrence of the bad event Abort. Moreover, observe that Procedure 2' is equivalent to Procedure 2 since in both procedures, all CF values are selected uniformly and P values are determined by CF. By combining these observations, it can be inferred that the gap between Procedure 1 and Procedure 2 is bounded by $\Pr[\text{Abort}]$.

4.4 Bounding the Bad Events

In this section, we assume $\mathcal{S}$ is efficient to prove $\Pr[\text{Abort}]$ is negligible. The efficiency of $\mathcal{S}$ will be proved in the next section. In the rest of the paper, all the definitions, lemmas and theorems are in the interaction game between $\mathcal{D}$ and $(P, \mathcal{S}^P)$ unless otherwise specified.

We recall the following example to explain our plan for bounding $\Pr[\text{Abort}]$. During the interaction with the distinguisher, suppose the simulator $\mathcal{S}$ sees an unsubverted chain $c = (s, x_s, ..., x_{s+r})$ that triggers completion. Let us call the current $\mathcal{S}.CF$ table T_{Initial}. For the full chain $c' = (1, x_1, \ldots, x_\ell)$ determined by c ($c \subset c'$), $\mathcal{S}$ hopes that it can find an index u so that (u, x_u) and $(u + 1, x_{u+1})$ are undefined before adaptation. It is easy to find an index u so that these two terms are not in T_{Initial}. However, before $\mathcal{S}$ determines x_u and x_{u+1}, it needs to evaluate $C\tilde{F}_i(x_i)$ for $i \neq u, u + 1$. And there may exist an index i so that (u, x_u) or $(u + 1, x_{u+1})$ is in $Q_i(x_i)$. It is also not obvious how to find u so that (u, x_u) and $(u + 1, x_{u+1})$ are honest since the distinguisher can subvert the round functions of any index.

We deal with the challenge in the following three steps.

Step 1: Analysis of unsubverted chains: We introduce the notion of monotone increasing (and decreasing) chains to analyze the property of unsubverted chains. We show that any unsubverted chain is a union of a decreasing chain and an increasing chain. We also show that (Theorem 3), inside a long monotone chain c^*, for any other term (j, x_j) in the "middle area" of c^* and any term $(i, x_i) \in c^*$ ($i \neq j$), (j, x_j) is honest and $CF_j(x_j)$ is not queried by $C\tilde{F}_i(x_i)$ (i.e., $(j, x_j) \notin Q_i(x_i)$).

Step 2: Analysis of subverted chains: We prove all the dishonest terms on a subverted chain are located on an interval shorter than $\ell/12$. As a result, the subverted chain c' can be viewed as an unsubverted chain except for a small dishonest area. That is to say, there always exists a long unsubverted chain $c'' \subset c'$.

Step 3: Bounding the bad event: By combining the two results above, we can deduce the existence of a long monotone chain $c^* \subset c'' \subset c'$. To conclude the proof regarding the negligibility of $\Pr[\text{Abort}]$, we demonstrate that the selection rule for the adaptation terms (u, x_u) and $(u + 1, x_{u+1})$ ensures that these two terms fall within the middle area of c^*, which implies honesty and freshness.

– Step 1: Analysis of unsubverted chains. To analyze the properties of unsubverted chains, we first introduce the notion of monotone chains.

The order function; monotone chains. To record the order in which $\mathcal{S}$ sets CF values, we define the following order function O from $\{1,\ldots,\ell\} \times \{0,1\}^n$ to positive integers (with an additional symbol $\perp$):

$$O(i,x) = \begin{cases} t & \text{if } CF_i(x) \text{ is the } t\text{-th evaluated } CF \text{ value by } \mathcal{S}, \\ \perp & \text{if } CF_i(x) \text{ is undefined in } \mathcal{S}.CF. \end{cases}$$

An unsubverted chain $(s, x_s, \ldots, x_{s+r})$ in $\mathcal{S}.CF$ is said to be *monotone increasing* (or *monotone decreasing*) if $O(i,x_i) < O(i+1,x_{i+1})$ for all $s \le i < s+r$ (or, likewise, $O(j,x_j) > O(j+1,x_{j+1})$ for all $s \le j < s+r$).

In the rest of the paper, w.l.o.g, we focus our analytic efforts on increasing chains; the results related to increasing chains can be easily transitioned into those related to decreasing chains. In the following lemma, we show that any unsubverted chain is a union of a decreasing chain and an increasing chain.

Lemma 2. *If $\mathcal{S}$ is efficient, then with overwhelming probability, any unsubverted chain $c = (s, x_s, \ldots, x_{s+r})$ in $\mathcal{S}.CF$ will satisfy one of the three conditions below:*

1. c is increasing,
2. c is decreasing,
3. There exists an index $s < v < s+r$ such that $(s, x_s, \ldots, x_v)$ is decreasing and $(v, x_v, \ldots, x_{s+r})$ is increasing.

Proof. It suffices to show that in $\mathcal{S}.CF$ there is no unsubverted length three chain $(s, x_s, x_{s+1}, x_{s+2})$ such that $CF_{s+1}(x_{s+1})$ is evaluated after both $CF_s(s_s)$ and $CF_{s+2}(x_{s+2})$ are evaluated. Suppose that throughout the interaction between $\mathcal{D}$ and $(P, \mathcal{S}^P)$, there are no more than P ($= \text{poly}(n)$) elements in $\mathcal{S}.CF$. Then,

$$\Pr\left[\begin{array}{l} \text{There is a length 3 chain } (s, x_s, x_{s+1}, x_{s+2}) \text{ such that} \\ O(s+1, x_{s+1}) > \max\{O(s+2, x_{s+2}), O(s, x_s).\} \end{array}\right]$$

$$= \sum_{i=2}^{P} \Pr\left[\begin{array}{l} \text{There is a length 3 chain } (s, x_s, x_{s+1}, x_{s+2}) \text{ such that} \\ O(s+1, x_{s+1}) = i > \max\{O(s+2, x_{s+2}), O(s, x_s)\}. \end{array}\right]$$

$$= \sum_{i=2}^{P} \sum_{\substack{j,k<i \\ j \ne k}} \Pr\left[\begin{array}{l} \text{There is a length 3 chain } (s, x_s, x_{s+1}, x_{s+2}) \text{ such} \\ \text{that } O(s+1, x_{s+1}) = i, O(s+2, x_{s+2}) = j \text{ and} \\ O(s, x_s) = k. \end{array}\right]$$

$$< \sum_{i=2}^{P} \sum_{\substack{j,k<i \\ j \ne k}} \frac{1}{2^n} < \frac{P^3}{2^n} = \mathsf{negl}(n),$$

where the first inequality is based on the fact that and $CF_{s+1}(x_{s+1})$ is selected uniformly and is independent of $CF_s(s_s)$ and $CF_{s+2}(x_{s+2})$. $\square$

Parameters in the Main Theorem. The security parameter ϵ' in Theorem 1 is determined by the last line of the inequality in the proof of Lemma 2, along with Theorem 5, which states that P is bounded by $22q_\mathcal{D}(q_\mathcal{A} + 1)$.

Next, we will use a sequence of lemmas to establish the following major theorem that describes the nice properties of increasing chains.

Theorem 3. *If S is efficient, then with overwhelming probability, any unsubverted increasing chain $c = (s, x_s, \ldots, x_{s+r})$ $(r > 8)$ in $S.CF$ will satisfy:*

1. *for any $0 < i < j$ and $8 < j \leq r$, $(s + j, x_{s+j}) \notin Q_{s+i}(x_{s+i})$;*
2. *for any $7 \leq i < j \leq r$, $(s + i, x_{s+i}) \notin Q_{s+j}(x_{s+j})$.*

Lemma 3. *With overwhelming probability, the following event does not happen: at some point of the game, there exist an unsubverted (or subverted) chain $c = (i, x_i, \ldots, x_j)$ and a length 10 unsubverted chain $c' = (s, y_s, \ldots, y_{s+9})$ in $S.CF$ such that*

- *for all $(j, x) \in c$, $C\tilde{F}_i(x)$ is defined;*
- *c and c' are disjoint;*
- *for each $s \leq k \leq s + 9$, $(k, y_k) \in Q_c$.*

Proof. Consider proving the following stronger statements: Imagine we fill the entire table $S.CF$ by uniformly selecting all the F values and (a_i, b_i) $(i = 1, \ldots, \ell)$. We will prove that with overwhelming probability over the choice of F values and (a_i, b_i), there are not two chains c and c' that satisfy the properties in the lemma.

Let (x_{i+1}, x_{i+2}), (y_s, y_{s+1}) be two pairs of n-bit strings and (i, j, s) be three positive indices. We denote by c the length $(j - i)$ chain starting with $(i + 1, x_{i+1}, x_{i+2})$ (without loss of generality, we assume c is a subverted chain for convenience in the rest of the proof) and denote by c' the length 10 unsubverted chain starting with (s, y_s, y_{s+1}). We denote by x_v $(v = i + 1, \ldots, j)$ the elements of c and denote by y_k $(k = s, \ldots, s + 9)$ the elements of c'. It is important to note that while x_{i+1}, x_{i+2}, y_s and y_{s+1} are specific n-bit strings, the values of x_v and y_k are currently undetermined. We use x_v and y_k purely to represent the elements of c and c' respectively. The actual values they will take on will be determined by choice of F values and (a_i, b_i). We define the event:

$$E_{i,j,s}(x_{i+1}, x_{i+2}, y_s, y_{s+1}) := \{c \text{ and } c' \text{ are disjoint, and for each } s \leq k \leq s+9, (k, y_k) \in Q_c\} .$$

For $s \leq t \leq s + 9$, we also define:

$$E_{i,j,s}^t(x_{i+1}, x_{i+2}, y_s, y_{s+1}) := \{c \text{ and } c' \text{ are disjoint, and for each } s \leq k \leq t, (k, y_k) \in Q_c\} .$$

To analyze the probability of $E_{i,j,s}(x_1, x_2, y_s, y_{s+1})$ over the choice of F and (a_i, b_i) $(i = 1, \ldots, \ell)$, we consider selecting uniformly the values of $F_i(x)$ for all $i = 1, \ldots, \ell$ and $x \in \{0, 1\}^n$ and selecting uniformly $a_v \cdot x_v \oplus b_v$ for $v = i, \ldots, j$. Since the function $x_v \rightarrow a_i \cdot x_v \oplus b_i$ is pairwise independent, the values of $a_k \cdot y_k \oplus b_k (k = s, \ldots, s + 9)$ are uniformly random. (For convenience, in the following, we will write $E_{i,j,s}$ for $E_{i,j,s}(x_1, x_2, y_s, y_{s+1})$ and $E_{i,j,s}^t$ for $E_{i,j,s}^t(x_1, x_2, y_s, y_{s+1})$.) Over the randomness of

$a_k \cdot y_k \oplus b_k$ $(k = s, \ldots, s+9)$, we have

$$\Pr[\, E_{i,j,s}\,]$$
$$= \Pr[E_{i,j,s} \mid E_{i,j,s}^{s+1}] \cdot \Pr[E_{i,j,s}^{s+1}]$$
$$< \Pr[\, E_{i,j,s} \mid E_{i,j,s}^{s+1}\,]$$
$$\quad \cdot (\Pr[\, CF_s(y_s) \in \cup_{v=i}^{j} Q_v(x_v) \mid y_s \neq x_s\,] + \Pr[\, CF_{s+1}(y_{s+1}) \in \cup_{v=i}^{j} Q_v(x_v) \mid y_{s+1} \neq x_{s+1}\,])$$
$$< \Pr[E_{i,j,s} \mid E_{i,j,s}^{s+3}] \cdot \Pr[E_{i,j,s}^{s+3} \mid E_{i,j,s}^{s+1}] \cdot 2 \cdot (\ell \cdot q_{\mathcal{A}}/2^n)$$
$$< \Pr[E_{i,j,s} \mid E_{i,j,s}^{s+5}] \cdot \Pr[E_{i,j,s}^{s+5} \mid E_{i,j,s}^{s+3}] \cdot (2\ell \cdot q_{\mathcal{A}}/2^n)^2$$
$$< \Pr[E_{i,j,s} \mid E_{i,j,s}^{s+7}] \cdot \Pr[E_{i,j,s}^{s+7} \mid E_{i,j,s}^{s+5}] \cdot (2\ell \cdot q_{\mathcal{A}}/2^n)^3$$
$$< \Pr[E_{i,j,s} \mid E_{i,j,s}^{s+7}] \cdot (2\ell \cdot q_{\mathcal{A}}/2^n)^4 < (2\ell \cdot q_{\mathcal{A}}/2^n)^5 \,.$$

The lemma is implied by taking the union bound over the choice of (x_1, x_2, y_s, y_{s+1}).

A similar proof can be used to prove the following lemma:

Lemma 4. *With overwhelming probability over the choice of all the F values and (a_i, b_i) $(i = 1, \ldots, \ell)$, there are not a term (i, x_i) and a length 8 unsubverted chain $c = (s, y_s, \ldots, y_{s+7})$ in $\mathcal{S}.CF$ such that $(k, y_k) \in Q_i(x_i)$ for all $k = s, s+2, s+4, s+6$.*

Lemma 5. *If $\mathcal{S}$ is efficient, then with overwhelming probability, for any unsubverted increasing chain $c = (s, x_s, \ldots, x_{s+r})$ in $\mathcal{S}.CF$, if $(s + 2t + 1, x_{s+2t+1}) \in Q_{s+2k}(x_{s+2k})$ (assuming $C\tilde{F}_{s+2k}(x_{s+2k})$ is defined) for some t, k with $0 < 2t + 1, 2k \leq r$, then $(s + 2i, x_{s+2i}) \in Q_{s+2k}(x_{s+2k})$ for all $0 < i \leq t$.*

Proof. We give a simple example to show the idea of the proof. Take $s = 1$, $r = 7$, $k = 2$ and $t = 3$ for example. We want to show that for any chain $c = (1, x_1, \ldots, x_8)$, if $(8, x_8) \in Q_5(x_5)$, then with overwhelming probability, $(1 + 2i, x_{1+2i}) \in Q_5(x_5)$ for $i = 1$.

Consider the following two ways of determining a length 8 unsubverted chain:

- **Procedure 1**:
  1. Pick an arbitrary moment in the interaction between $\mathcal{D}$ and $(P, \mathcal{S}^P)$ and abort the game. Denote the table $\mathcal{S}.CF$ at this moment by T_{initial}. Pick a length 2 increasing chain $(1, x_1, x_2)$ in T_{initial} such that it is not a subchain of a length 3 unsubverted chain.
  2. For $2 \leq i \leq 7$, select $CF_i(x_i)$ uniformly, set $x_{i+1} := CF_i(x_i) \oplus x_{i-1}$ and abort the procedure if $(i + 1, x_{i+1})$ is already in the table T_{initial}.
  3. Evaluate $C\tilde{F}_5(x_5)$.
- **Procedure 2**:
  1. Pick an arbitrary moment in the interaction between $\mathcal{D}$ and $(P, \mathcal{S}^P)$ and abort the game. Denote the table $\mathcal{S}.CF$ at this moment by T_{initial}. Pick a length 2 increasing chain $(1, x_1, x_2)$ in T_{initial} such that it is not a subchain of a length 3 unsubverted chain.
  2. Select $CF_2(x_2)$ uniformly and set $x_3 := a_2 \oplus x_1$.
  3. Select 4 uniform n-bit strings a_4, a_5, a_6 and a_7. Set $x_5 := a_4 \oplus x_3$, $x_7 := a_6 \oplus x_5$ and abort the procedure if either of them is in T_{initial}. Set $CF_5(x_5) := a_5$ and $CF_7(x_7) := a_7$.

4. Evaluate $C\tilde{F}_5(x_5)$.

5. Select $CF_3(x_3)$ uniformly (use the existing value if it has been evaluated), set $x_4 := CF_3(x_3) \oplus x_2$, $x_6 := a_5 \oplus x_4$, $x_8 := a_7 \oplus x_6$, and abort the procedure if any one of x_4, x_4 and x_8 is in T_{initial}.

A quick thought reveals that the above two procedures are equivalent in terms of the distribution of the chain and, furthermore, the probability they abort is negligible because of Lemma 2. We use the second procedure to analyze the distribution of the first one. In the second procedure, we can see that if $(3, x_3) \notin Q_5(x_5)$, then $CF_3(x_3)$ is still uniform conditioned on $Q_5(x_5)$, which implies that $x_8 = a_7 \oplus x_6 = a_7 \oplus a_5 \oplus x_4 = a_7 \oplus a_5 \oplus CF_3(x_3) \oplus x_2$ is uniform. Therefore, if $(3, x_3) \notin Q_5(x_5)$, $(8, x_8) \in Q_5(x_5)$ with negligible probability.

The full proof can be achieved by replacing the concrete numbers in the last example by more general parameters s, r, k and t and taking the union bound over the various values of these parameters.

Lemma 6. *If $\mathcal{S}$ is efficient, then with overwhelming probability, for any unsubverted increasing chain $c = (s, x_s, \dots, x_{s+r})$ in $\mathcal{S}.CF$ and any index i, j with $0 < i < j$ and $8 < j \leq r$, $(s + j, x_{s+j}) \notin Q_{s+i}(x_{s+i})$ (if $C\tilde{F}_{s+i}(x_{s+i})$ is defined).*

Proof. Without loss of generality, assume $i = 0$. Suppose $(s + j, x_{s+j}) \in Q_s(x_s)$. Notice that $(s + j - 1, x_{s+j-1}) \in Q_{s+i}(x_{s+i})$ with overwhelming probability because otherwise the randomness of $CF_{s+j-1}(x_{s+j-1})$ will cause the event $(s + j, x_{s+j}) \notin Q_s(x_s)$. Then,

- if j is odd, since $j > 8$ and $(s + j, x_{s+j}) \in Q_s(x_s)$, by Lemma 5, $(s + 2k, x_{s+2k}) \in Q_s(x_s)$ for $k = 1, 2, 3, 4$. This contradicts Lemma 4.
- if j is even, since $j > 8$ and $(s + j - 1, x_{s+j-1}) \in Q_s(x_s)$, by Lemma 5, $(s + 2j, x_{s+2j}) \in Q_s(x_s)$ for $j = 1, 2, 3, 4$, which contradicts with Lemma 4. □

Lemma 7. *If $\mathcal{S}$ is efficient, then with overwhelming probability, for any unsubverted increasing chain $c = (s, x_s, \dots, x_{s+r})$ in $\mathcal{S}.CF$, if $(s + 2t, x_{s+2t}) \in Q_{s+k}(x_{s+k})$ (assuming $C\tilde{F}_{s+k}(x_{s+k})$ is defined) for some t, k with $0 < 2t < k \leq r$, then $(s + 2i - 1, x_{s+2i-1}) \in Q_{s+k}(x_{s+k})$ for all $0 < i \leq t$.*

Proof. The proof of the lemma is similar to that of Lemma 5. Consider the example where $s = 1$, $r = 8$, $t = 2$ and $k = 8$. We want to show that for any chain $c = (1, x_1, \dots, x_9)$, if $(5, x_5) \in Q_9(x_9)$, then with overwhelming probability, $(2i, x_{2i}) \in Q_9(x_9)$ for $i = 1$.

Consider the following two ways of determining a length 9 unsubverted chain:

- **Procedure 1**:
  1. Pick an arbitrary moment in G_5 and abort the game. Denote the table $\mathcal{M}^3.CF$ at this moment by T_{initial}. Pick a length 2 increasing chain $(1, x_1, x_2)$ in T_{initial} such that it is not a subchain of a length 3 unsubverted chain.
  2. For $2 \leq i \leq 8$, select $CF_i(x_i)$ uniformly, set $x_{i+1} := CF_i(x_i) \oplus x_{i-1}$ and abort the procedure if $(i + 1, x_{i+1})$ is already in the table T_{initial}.
  3. Evaluate $C\tilde{F}_9(x_9)$.
- **Procedure 2**:

1. Pick an arbitrary moment in G_5 and abort the game. Denote the table $\mathcal{M}^3.CF$ at this moment by T_{initial}. Pick a length 2 increasing chain $(1, x_1, x_2)$ in T_{initial} such that it is not a subchain of a length 3 unsubverted chain.
2. Select 3 uniform n-bit strings a_3, a_4 and a_5. Set $x_4 := a_3 \oplus x_2$, $x_6 := a_5 \oplus x_4$ and aborts the procedure if either of them is in T_{initial}. Set $CF_4(x_4) := a_4$ and $CF_6(x_6) := a_6$.
3. Select x_7, x_8 and x_9 uniformly and aborts the procedure if any one of them is in T_{initial}. Set $CF_7(x_7) := x_6 \oplus x_8$ and $CF_8(x_8) := x_7 \oplus x_9$.
4. Evaluate $\tilde{CF}_9(x_9)$.
5. Select $CF_2(x_2)$ uniformly(use the existing value if it has been evaluated), set $x_3 := CF_2(x_2) \oplus x_1$, $x_5 := a_4 \oplus x_3$, $CF_6(x_6) := x_7 \oplus x_5$, and aborts the procedure if either x_3 or x_5 is in T_{initial}.

A quick thought reveals that the above two procedures are equivalent in terms of the distribution of the chain (and, furthermore, the probability they abort is negligible because of Lemma 2). We use the second procedure to analyze the distribution of the first one. In the second procedure, we can see that if $(2, x_2) \notin Q_9(x_9)$, then $CF_2(x_2)$ is still uniform conditioned on $Q_9(x_9)$, which implies that $x_5 = a_4 \oplus x_3 = a_4 \oplus a_2 \oplus x_1$ is uniform. Therefore, if $(2, x_2) \notin Q_9(x_9)$, $(5, x_5) \in Q_9(x_9)$ with negligible probability.

The formal proof can be achieved by replacing the concrete numbers in the last example by more general parameters s, r, t and k and taking the union bound over the various values of these parameters. □

Following directly from Lemma 4 and Lemma 7, we get:

Lemma 8. *If $\mathcal{S}$ is efficient, then with overwhelming probability, for any unsubverted increasing chain $c = (s, x_s, \ldots, x_{s+r})$ in $\mathcal{S}.CF$ and any index i, j with $7 < i < j \le r$, $(s + i, x_{s+i}) \notin Q_{s+j}(x_{s+j})$(if $\tilde{CF}_{s+j}(x_{s+j})$ is defined).*

Theorem 3 follows from the combination of Lemma 6 and Lemma 8.

- Step 2: Analysis of subverted chains. Now we turn our attention to subverted chains. We want to show that although, in general, there are some dishonest terms on a subverted chain, all of them gather in a small area.

Lemma 9. *If $\mathcal{S}$ is efficient, then with overwhelming probability, there does not exist an unsubverted increasing chain $c = (i, x_i, \ldots, x_{i+8})$ in $\mathcal{S}.CF$ such that $\tilde{CF}_{i+8}(x_{i+8})$ is defined in $\mathcal{S}.CF$ and $(i + 8, x_{i+8})$ is dishonest.*

Proof. We say the distinguisher $\mathcal{D}$ wins the interaction game with $(P, \mathcal{S}^P)$ if it is able to find an unsubverted increasing chain c in $\mathcal{S}.CF$ that satisfies the property in the lemma. By Lemma 6, the probability that there is a length-9 unsubverted increasing chain $c = (i, x_i, \ldots, x_{i+8})$ with $(i + 7, x_{i+7}) \in Q_{i+8}(x_{i+8})$ $(\tilde{CF}_{i+8}(x_{i+8}))$ is negligible. We denote this negligible probability by δ. To show the probability that $\mathcal{D}$ wins is negligible, consider the following experiment with a distinguisher $\mathcal{D}^*$:

Exp*
1. $\mathcal{D}^*$ takes an arbitrary moment of $\mathcal{S}$, stops the game and selects an arbitrary length-2 increasing chain (i, x_i, x_{i+1}) in $\mathcal{S}.CF$ such that $CF_{i+2}(x_{i+2})$ is not evaluated for $x_{i+1} := x_i \oplus CF_i(x_i)$.

---

2. Then, $\mathcal{D}^*$ extends (i, x_i, x_{i+1}) to $(i, x_i, \ldots, x_{i+7})$ by iteratively evaluating $CF_{j-1}(x_{j-1})$ (selected uniformly) and $x_j := x_{j-2} \oplus CF_{j-1}(x_{j-1})$ for $i + 3 \leq j \leq i + 7$. The experiment aborts if $CF_{i+7}(x_{i+7})$ is already evaluated.

3. For any term (j, y), if $CF_j(y)$ is still unevaluated and $(j, y) \neq (i + 7, x_{i+7})$, $\mathcal{D}^*$ selects $CF_j(y)$ uniformly.

4. Finally $\mathcal{D}^*$ selects $CF_{i+7}(x_{i+7})$ and check if $(i + 8, x_{i+8})$ is dishonest for $x_{i+8} := x_{i+6} \oplus CF_{i+7}(x_{i+7})$.

5. $\mathcal{D}^*$ wins **Exp*** if the experiment does not abort in Step 2 and $(i + 8, x_{i+8})$ is dishonest.

To prove $\mathcal{D}$ wins negligibly, it is sufficient to show the probability that the experiment aborts in Step 2 or $\mathcal{D}^*$ wins is negligible. We also stress that although **Exp*** is not the interaction between $\mathcal{D}$ and $(P, \mathcal{S}^P)$, the lemmas we proved in this section can still be applied because all the CF values here are also selected uniformly and independently.

$$\Pr_{\textbf{Exp*}} [\text{The experiment aborts in Step 2 or } \mathcal{D}^* \text{ wins.}]$$

$$\leq \Pr_{\textbf{Exp*}} [\text{The experiment aborts in Step 2.}] + \Pr_{\textbf{Exp*}} \begin{bmatrix} \mathcal{D}^* \text{ wins and there are at least} \\ \sqrt{\delta}2^n \ n\text{-bit strings } x \text{ such that} \\ (i + 7, x_{i+7}) \in Q_{i+8}(x). \end{bmatrix}$$

$$+ \Pr_{\textbf{Exp*}} \begin{bmatrix} \mathcal{D}^* \text{ wins and there are fewer than } \sqrt{\delta}2^n \ n\text{-bit} \\ \text{strings } x \text{ such that } (i + 7, x_{i+7}) \in Q_{i+8}(x). \end{bmatrix}$$

$$< \mathsf{negl}(n) + \Pr_{\textbf{Exp*}} \begin{bmatrix} \mathcal{D}^* \text{ wins and there are at least } \sqrt{\delta}2^n \ n\text{-bit strings } x \text{ such that} \\ (i + 7, x_{i+7}) \in Q_{i+8}(x). \end{bmatrix}$$

$$+ \Pr_{\textbf{Exp*}} \begin{bmatrix} \mathcal{D}^* \text{ wins.} & \text{There are fewer than } \sqrt{\delta}2^n \ n\text{-bit strings } x \text{ such that} \\ & (i + 7, x_{i+7}) \in Q_{i+8}(x). \end{bmatrix}$$

$$< \mathsf{negl}(n) + \sqrt{\delta} + \Pr_{\textbf{Exp*}} \begin{bmatrix} (i + 8, x_{i+8}) \text{ is dishon-} & \text{There are fewer than } \sqrt{\delta}2^n \ n\text{-bit} \\ \text{est and } (i + 7, x_{i+7}) \in & \text{strings } x \text{ such that } (i + 7, x_{i+7}) \in \\ Q_{i+8}(x_{i+8}). & Q_{i+8}(x). \end{bmatrix}$$

$$+ \Pr_{\textbf{Exp*}} \begin{bmatrix} (i + 8, x_{i+8}) \text{ is dishon-} & \text{There are fewer than } \sqrt{\delta}2^n \\ \text{est and } (i + 7, x_{i+7}) \notin & n\text{-bit strings } x \text{ such that} \\ Q_{i+8}(x_{i+8}). & (i + 7, x_{i+7}) \in Q_{i+8}(x). \end{bmatrix}$$

$$< \mathsf{negl}(n) + \sqrt{\delta} + \sqrt{\delta} + \epsilon = \mathsf{negl}(n). \quad \square$$

Definition 2 (Bad region). *For a subverted chain $c = (s, x_s, \ldots, x_{s+r})$ in $\mathcal{S}.CF$, we say a subchain $(i, x_i, \ldots, x_j)$ $(s \leq i < j \leq s + r)$ of c is a* bad region *of c if there is no sequence of 14 consecutive elements $(k, x_k, \ldots, x_{k+13})$ $(i \leq k \leq j - 13)$ that are honest.*

For a subverted chain $c = (s, x_s, \ldots, x_{s+r})$ in $\mathcal{S}.CF$, we say two bad regions of c, $(i, x_i, \ldots, x_j)$ and $(i', x_{i'}, \ldots, x_{j'})$ $(i < i', j < j')$ are separated if the subchain $(i, x_i, \ldots, x'_j)$ of c is not a bad region of c.

Lemma 10. *If $\ell > 337n/\log(1/\epsilon)$, then in the interaction game between $\mathcal{D}$ and $(P, \mathcal{S}^P)$, with overwhelming probability, there does not exist a subverted chain $(s, x_s, \ldots, x_{s+r})$ in $\mathcal{S}.CF$ such that it has a bad region with length greater than $\ell/12$.*

Proof. Consider proving the following stronger statement: with overwhelming probability over the uniform choice of (a_i, b_i) ($i = 1, \ldots, \ell$) and values of $F_i(x)$ for all $i = 1, \ldots, \ell$ and $x \in \{0, 1\}^n$, there is no bad region with length greater than $\ell/12$. Imagine we select $F_i(x)$ for all $i = \{1, \ldots, \ell\}$ and $x \in \{0, 1\}^n$ and leave a_i and b_i undetermined. Then, over the randomness of the choice of a_i and b_i, we have

$\Pr[\text{There is a subverted chain } c \text{ with a bad region longer than } \ell/12.]$

$$= \sum_{i=1}^{\ell} \Pr \begin{bmatrix} \text{There is a subverted chain } c \text{ with a bad region longer than } \ell/12 \text{ and the} \\ \text{bad region begins at index } i. \end{bmatrix}$$

$$= \sum_{i=1}^{\ell} \sum_{x, x' \in \{0,1\}^n} \Pr \begin{bmatrix} \text{There is a subverted chain } c \text{ with a bad region longer than } \ell/12. \\ \text{The bad region begins at index } i \text{ and its first two elements are} \\ (i, x) \text{ and } (i+1, x'). \end{bmatrix}$$

$$< \sum_{i=1}^{\ell} \sum_{x, x' \in \{0,1\}^n} \Pr \begin{bmatrix} \text{There is a subverted chain } c = (i, x, x', \ldots, x_r) \text{ such that its first} \\ \text{element has index } i \text{ and } r - i > \ell/12. \text{ Moreover, for any length} \\ 14 \text{ subchain of } c \text{ in the form of } (14k, x_{14k}, \ldots, x_{14k+13}), \text{ at least} \\ \text{one of } 14 \text{ elements is dishonest.} \end{bmatrix}$$

$$< \sum_{i=1}^{\ell} (2^n)^2 \cdot (14\epsilon)^{\ell/168-1} = \ell \cdot 2^{2n} \cdot (14\epsilon)^{\ell/168-1} = \mathsf{negl}(n).$$

Lemma 11. *If $\mathcal{S}$ is efficient, then with overwhelming probability, there is no subverted chain $c = (s, x_s, \ldots, x_{s+r})$ in $\mathcal{S}.CF$ that has two separated bad regions.*

Proof. The lemma is implied directly by Lemma 2 and Lemma 9.

– Step 3: Bounding the bad event. Now we put together the properties of unsubverted and subverted chains above to show the main theorem:

Theorem 4. *If $\mathcal{S}$ is efficient, the probability that Abort happens is negligible.*

Proof. Due to space limitations, we only give a high-level description of how integrating the properties of chains above gives the negligibility of Abort.

Imagine we start with an initial table of CF values T_{initial}. Suppose there is an unsubverted chain $c = (s, x_s, \ldots, x_{s+\ell/20-1})$ in T_{initial} that has passed the test of $\mathcal{S}.\text{HonestyCheck}$. This indicates that all the elements of c are honest. Without loss of generality, here we assume $s + \ell/20 - 1 < \ell/4$. Consequently, when $\mathcal{S}$ completes c, it adapts the value of CF at $u = \ell/2$.

Now we prove freshness and honesty when completing the chain c. We denote by $c' = (1, x_1, \ldots, x_\ell)$ the full chain that contains c.

  – Case 1: There exists a dishonest term (j, x_j) in $(s, x_s, \ldots, x_\ell)$. In this case, by Lemma 9 and 10, $j < (s + \ell/20 - 1) + 8 + \ell/12 < \ell/4 + 8 + \ell/12 = \ell/3 + 8$. This means the chain $c'' = (\ell/3 + 8, x_{\ell/3+8}, \ldots, x_\ell)$ is honest. Again by Lemma 9, the chain $c^* = (\ell/2 - 10, x_{\ell/2-10}, \ldots, x_\ell)$ is increasing. Therefore, the CF values in the adaption positions are honest. They also satisfy freshness since: 1. they are not in Q_{c^*} because of Theorem 3; 2. they are uniform and are therefore outside of T_{initial} and Q_{c'/c^*}.
  – Case 2: There does not exist a dishonest term (j, x_j) in $(s, x_s, \ldots, x_\ell)$. The proof in this case is simply a subset of that of Case 1.

27

4.5 Efficiency of $\mathcal{S}$

In this section, we are going to show that the number of the elements in $\mathcal{S}.CF$ is bounded by a polynomial function if the distinguisher $\mathcal{D}$ makes at most $q_{\mathcal{D}}$ ($q_{\mathcal{D}}$ is polynomial) queries to CF or the ideal object.

Lemma 12. *If $\mathcal{S}$ is efficient, then with overwhelming probability, there is not a chain $c = (1, w_1, \ldots, w_\ell)$ and three pairwise disjoint increasing chains $c_1 = (i, x_i, \ldots, x_{i+10})$, $c_2 = (j, y_j, \ldots, y_{j+10})$ and $c_3 = (k, z_k, \ldots, z_{k+10})$ in $\mathcal{S}.CF$, such that*

- *for all $(i, x) \in c$, $C\tilde{F}_i(x)$ is defined;*
- *c is disjoint with c_1, c_2, c_3;*
- *$(i + 10, x_{i+10}), (j + 10, y_{j+10}), (k + 10, z_{k+10}) \in Q_c$.*

Proof. According to Lemma 3, if $(i + 10, x_{i+10}) \in Q_c$, then there exists an index m ($i + 1 \le m \le i + 9$) such that in the length 3 monotone increasing chain (x_{i-1}, x_i, x_{i+1}), $(i, x_i) \notin Q_c$ but $(i + 1, x_{i+1}) \in Q_c$. Now we turn this observation into a proof.

Consider the following experiment in $\mathcal{S}$. Take an arbitrary pair of n-bit strings (w_1, w_2). $\mathcal{D}$ tries to find a subverted chain c starting with (w_1, w_2) (w.l.o.g., we only consider subverted chain for convenience) and a length 3 increasing chain (x_{i-1}, x_i, x_{i+1}) s.t., $(i, x_i) \notin Q_c$ and $(i + 1, x_{i+1}) \in Q_c$. We show the probability that $\mathcal{D}$ wins is negligible ($\ell q_{\mathcal{A}}/s^n$): Suppose, without loss of generality, $\mathcal{D}$ queries all the elements in Q_c at the beginning of $\mathcal{S}$. At some moment of the experiment, $\mathcal{D}$ will select a pair of terms $(i - 1, x_{i-1}, x_i)$ as the starting pair of target length 3 chain. It is easy to see that, since $(i, x_i) \notin Q_c$, $(i + 1, x_{i+1}) \in Q_c$ with probability not greater than $\text{poly}(n) \cdot \ell q_{\mathcal{A}}/2^n$, where $\text{poly}(n)$ denotes the upper bound of the number of the terms in $\mathcal{S}.CF$.

For any pair (w_1, w_2), we define the event:

$$
E(w_1, w_2) := \left\{ \begin{array}{l} \text{There are three monotone increasing unsub-} \\ \text{verted chains } c_1 = (i, x_i, \ldots, x_{i+10}), \ c_2 = \\ (j, y_j, \ldots, y_{j+10}) \text{ and } c_3 = (k, z_k, \ldots, z_{k+10}) \text{ in} \\ \mathcal{S}.CF, \text{ such that } c \text{ is disjoint with } c_1, c_2, c_3 \text{ and} \\ (i + 10, x_{i+10}), (j + 10, y_{j+10}), (k + 10, z_{k+10}) \quad \in \\ Q_c, \text{ where } c \text{ is the subverted starting with } (w_1, w_2) \end{array} \right\}
$$

Finally we have

$$
\sum_{(w_1, w_2) \in \{0,1\}^{2n}} \Pr[E(w_1, w_2)] < 2^{2n} \cdot (\text{poly}(n) \cdot \ell q_{\mathcal{A}}/2^n)^3 = \mathsf{negl}(n). \quad \square
$$

Lemma 13. *Suppose $\mathcal{S}$ is efficient. Let C_{11} be a set of length 11 increasing chains and c be a chain in $\mathcal{S}.CF$ such that c is disjoint with any element in C_{11}, and for all $(i, x) \in c$, $C\tilde{F}_i(x)$ is defined. Then, with overwhelming probability, there are at most 20 chains $c' = (i, x_i, \ldots, x_{i+10}) \in C_{11}$ such that $(i + 10, x_{i+10}) \in Q_c$.*

Proof. Suppose there are 21 chains in C_{11} that satisfy the property in the lemma. Notice that for each length 11 chain c', there are at most 9 other length 11 chains that are not disjoint with c'. Then, among the 21 chains satisfying the property in the lemma, we can find 3 pairwise disjoint chains. This contradicts Lemma 12.

Definition 3 (Order of a chain). *We define the order of an unsubverted chain* $c = (s, x_s, \ldots, x_{s+r})$ *in* $\mathcal{S}.CF$ *to be:*

$$O(c) := \min_{k=s,\ldots,s+r-1} \left\{ \max\{O(k, x_k), O(k+1, x_{k+1})\} \right\}$$

Intuitively speaking, the order of a chain describes the time when a chain is "determined."

Lemma 14. *Suppose* $\mathcal{S}$ *is efficient. Let* C_{Disj} *be a set of pairwise disjoint unsubverted chains with length greater than or equal to 4 in* $\mathcal{S}.CF$. *Define the set A to be the set of the elements of the chains in* C_{Disj}. *Then, with overwhelming probability,* $|A| \geq \sum_{c \in C_{Disj}} (L(c) - 3)$.

Proof. For any $c \in C_{\text{Disj}}$ and a term (i, x) in c, we say (i, x) is *original* in c if there does not exist a different element $c' \in C_{\text{Disj}}$ such that c and c' intersects at (i, x) and $O(c) \geq O(c')$. Notice that a term (i, x) can be original in at most one chain.

Now we are going to show that, with overwhelming probability, each element in C_{Disj} contains at most 3 non-original terms. Suppose there is a chain $c = (s, x_s, \ldots, x_{s+r})$ that has four non-original terms. Then there are two non-original elements, (i, x_i) and (j, x_j), such that $s \leq i < j - 2 \leq s + r - 2$. Because of Lemma 2, w.l.o.g, we assume

$$O(i, x_i) > O(i+1, x_{i+1}) > O(i+2, x_{i+2}).$$

Since (i, x_i) is non-original in c, there is a chain $c' \neq c$ such that $(i, x_i) \in c'$ and $O(c) \geq O(c')$. Since c and c' are disjoint, $(i+1, x_{i+1}) \notin c'$. Then, since $O(c) \geq O(c')$ and $O(i+1, x_{i+1}) > O(i+2, x_{i+2})$, we have $O(c) > O(c')$, which means $(i+1, x_{i+1})$ is not evaluated when c' has been determined. Finally, because $\mathcal{S}.CF(i+1, x_{i+1})$ is selected uniformly, $(i, x_i) \in c'$ with negligible probability. A contradiction.

Going back to the proof of the lemma, since each term is original in at most one chain and each chain in C_{Disj} has all but 3 original elements, $|A|$ is lower bounded by the sum of the original terms in the elements of C_{Disj}, which is not less than $\sum_{c \in C_{\text{Disj}}} (L(c) - 3)$.

Theorem 5. *For any positive integer* $k \leq q_{\mathcal{D}}$, *with overwhelming probability, at the end of the k-th round of* $\mathcal{S}$, *there are fewer than* $(22q_{\mathcal{A}} + 1)k$ *terms in* $\mathcal{S}.CF$.

Remark. In the proof of Theorem 5, we will make use of Lemma 13 and Lemma 14. However, these lemmas already take efficiency of $\mathcal{S}$ as their assumptions. To reassure the reader that there is not a circular argument here, we imagine that the k-th round of the game is forced to end when $\mathcal{S}.CF$ contains more than $(22q_{\mathcal{A}} + 1)k$ elements. In this way, we can also feel free to reason about $\mathcal{S}.CF[k]$.

Proof. In $\mathcal{S}.CF[k]$, for any unsubverted chain c, we call c a *generator* if c was processed by the procedure $\mathcal{S}.\text{HonestCheck}$. We denote by C_G the set of generators. We define a function g from $C_{\text{FComp}}[k]$ to C_G: for each $c_1 \in C_{\text{FComp}}[k]$ and $c \in C_G$, we say $g(c_1) = c$ if $c_1 \subset c$. Define $\quad G := \{(i, x) \mid \text{there is } c \in C_G \text{ such that } (i, x) \in c.\}$.

Since C_G is a set of pairwise disjoint chains, by Lemma 14,

$$|G| \geq \sum_{c \in C_G} (L(c) - 3) = (\ell/20 - 3) \cdot |C_G|. \tag{2}$$

To understand the structure of G, we define several subsets of G. We say a point $(i, x) \in G$ is a *tail point* if there is an increasing $c_2 = (s, x_s, \ldots, x_{s+10})$ in $\mathcal{S}.CF$ (w.l.o.g., we only consider the increasing case) and a chain $c \in C_G$ such that $(i, x) = (s + 10, x_{s+10})$ and $c_2 \subset c$. We say a point $(i, x) \in G$ is a *head point* if it is not a tail point. We denote the sets of the head points and tail points by G_{Head} and G_{Tail}, respectively. For any point $(i, x) \in G_{\text{Tail}}$ and any chain $c \in C_G$, we say c covers (i, x) $((i, x) \notin c)$ if $(i, x) \in Q_c$ or $(i, x) \in Q_{g^{-1}(c)}$ (if c has a preimage in function g). We define G_{Query} to be the set of the points in G_{Tail} that are not covered by any element in C_G. Notice that any element in G_{Query} was queried directly by the distinguisher $\mathcal{D}$. Our goal is to show $|G_{\text{Query}}|$ is big.

By Lemma 2, the number of the elements in G_{Head} is easily bounded by

$$|G_{\text{Head}}| \le 19 \cdot |C_G|. \tag{3}$$

By Lemma 13

$$|G_{\text{Tail}}/G_{\text{Query}}| \le 20 \cdot |C_G|. \tag{4}$$

Summarizing Equation 2, 3 and 4, we have

$$|G_{\text{Query}}| = |G| - |G_{\text{Head}}| - |G_{\text{Tail}}/G_{\text{Query}}| \ge (\ell/20 - 3)|C_G| - 19|C_G| - 20|C_{Fu}| = (\ell/20 - 42)|C_G|.$$

This implies that

$$|\mathcal{S}.CF[k]| \le \ell \cdot q_{\mathcal{A}} \cdot |C_G| + k \le \ell \cdot q_{\mathcal{A}} \cdot |G_{\text{Query}}|/(\ell/20 - 42) + k$$
$$\le \ell \cdot q_{\mathcal{A}} \cdot k/(\ell/20 - 42) + k \le \ell \cdot q_{\mathcal{A}} \cdot k/(\ell/22) + k = (22q_{\mathcal{A}} + 1)k.$$

We remark that all the statements in the proof are true with overwhelming probability.

References

1. M. Bellare, K. G. Paterson, and P. Rogaway. Security of symmetric encryption against mass surveillance. In J. A. Garay and R. Gennaro, editors, *CRYPTO 2014, Part I*, volume 8616 of *LNCS*, pages 1–19. Springer, Heidelberg, Aug. 2014.
2. R. Bhattacharyya, M. Nandi, and A. Raychaudhuri. Crooked indifferentiability of enveloped xor revisited. In *INDOCRYPT 2021*, pages 73–92, 2021.
3. M. Blum. Designing programs that check their work, November 1988. Technical Report TR-88-009, International Computer Science Institure. Available at http://www.icsi.berkeley.edu/pubs/techreports/tr-88-009.pdf.
4. M. Blum and S. Kannan. Designing programs that check their work. In *21st ACM STOC*, pages 86–97. ACM Press, May 1989.
5. M. Blum, M. Luby, and R. Rubinfeld. Self-testing/correcting with applications to numerical problems. In *22nd ACM STOC*, pages 73–83. ACM Press, May 1990.
6. R. Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *42nd FOCS*, pages 136–145. IEEE Computer Society Press, Oct. 2001.
7. S. S. M. Chow, A. Russell, Q. Tang, M. Yung, Y. Zhao, and H.-S. Zhou. Let a non-barking watchdog bite: Cliptographic signatures with an offline watchdog. In D. Lin and K. Sako, editors, *PKC 2019, Part I*, volume 11442 of *LNCS*, pages 221–251. Springer, Heidelberg, Apr. 2019.

8. S. Coretti, Y. Dodis, S. Guo, and J. P. Steinberger. Random oracles and non-uniformity. In J. B. Nielsen and V. Rijmen, editors, *EUROCRYPT 2018, Part I*, volume 10820 of *LNCS*, pages 227–258. Springer, Heidelberg, Apr. / May 2018.

9. J.-S. Coron, Y. Dodis, C. Malinaud, and P. Puniya. Merkle-Damgård revisited: How to construct a hash function. In V. Shoup, editor, *CRYPTO 2005*, volume 3621 of *LNCS*, pages 430–448. Springer, Heidelberg, Aug. 2005.

10. J.-S. Coron, T. Holenstein, R. Künzler, J. Patarin, Y. Seurin, and S. Tessaro. How to build an ideal cipher: The indifferentiability of the Feistel construction. *Journal of Cryptology*, 29(1):61–114, Jan. 2016.

11. D. Dachman-Soled, J. Katz, and A. Thiruvengadam. 10-round Feistel is indifferentiable from an ideal cipher. In M. Fischlin and J.-S. Coron, editors, *EUROCRYPT 2016, Part II*, volume 9666 of *LNCS*, pages 649–678. Springer, Heidelberg, May 2016.

12. Y. Dai and J. P. Steinberger. Indifferentiability of 8-round Feistel networks. In M. Robshaw and J. Katz, editors, *CRYPTO 2016, Part I*, volume 9814 of *LNCS*, pages 95–120. Springer, Heidelberg, Aug. 2016.

13. Y. Dodis, C. Ganesh, A. Golovnev, A. Juels, and T. Ristenpart. A formal treatment of backdoored pseudorandom generators. In E. Oswald and M. Fischlin, editors, *EUROCRYPT 2015, Part I*, volume 9056 of *LNCS*, pages 101–126. Springer, Heidelberg, Apr. 2015.

14. Y. Dodis, S. Guo, and J. Katz. Fixing cracks in the concrete: Random oracles with auxiliary input, revisited. In J.-S. Coron and J. B. Nielsen, editors, *EUROCRYPT 2017, Part II*, volume 10211 of *LNCS*, pages 473–495. Springer, Heidelberg, Apr. / May 2017.

15. Y. Dodis and P. Puniya. On the relation between the ideal cipher and the random oracle models. In S. Halevi and T. Rabin, editors, *TCC 2006*, volume 3876 of *LNCS*, pages 184–206. Springer, Heidelberg, Mar. 2006.

16. Y. Dodis and P. Puniya. Feistel networks made public, and applications. In M. Naor, editor, *EUROCRYPT 2007*, volume 4515 of *LNCS*, pages 534–554. Springer, Heidelberg, May 2007.

17. U. M. Maurer, R. Renner, and C. Holenstein. Indifferentiability, impossibility results on reductions, and applications to the random oracle methodology. In M. Naor, editor, *TCC 2004*, volume 2951 of *LNCS*, pages 21–39. Springer, Heidelberg, Feb. 2004.

18. T. Ristenpart, H. Shacham, and T. Shrimpton. Careful with composition: Limitations of the indifferentiability framework. In K. G. Paterson, editor, *EUROCRYPT 2011*, volume 6632 of *LNCS*, pages 487–506. Springer, Heidelberg, May 2011.

19. A. Russell, Q. Tang, M. Yung, and H.-S. Zhou. Cliptography: Clipping the power of kleptographic attacks. In J. H. Cheon and T. Takagi, editors, *ASIACRYPT 2016, Part II*, volume 10032 of *LNCS*, pages 34–64. Springer, Heidelberg, Dec. 2016.

20. A. Russell, Q. Tang, M. Yung, and H.-S. Zhou. Correcting subverted random oracles. In H. Shacham and A. Boldyreva, editors, *CRYPTO 2018, Part II*, volume 10992 of *LNCS*, pages 241–271. Springer, Heidelberg, Aug. 2018.

21. A. Russell, Q. Tang, M. Yung, H.-S. Zhou, and J. Zhu. Correcting subverted random oracles, 2021. https://eprint.iacr.org/2021/042.

22. A. Young and M. Yung. The dark side of "black-box" cryptography, or: Should we trust capstone? In N. Koblitz, editor, *CRYPTO'96*, volume 1109 of *LNCS*, pages 89–103. Springer, Heidelberg, Aug. 1996.

23. A. Young and M. Yung. Kleptography: Using cryptography against cryptography. In W. Fumy, editor, *EUROCRYPT'97*, volume 1233 of *LNCS*, pages 62–74. Springer, Heidelberg, May 1997.

Supplementary Materials

The supplementary material consists of four section. Section A describes the definition of classical indifferentiability and the replacement theorems for both classical and crooked indifferentiability. Section B is a complete proof for the main result in the paper. Section B.9 contains the pseudocode of the games we defined in Section B.

A Indifferentiability and Replacement Theorems

A.1 Preliminary: Indifferentiability

The notion of indifferentiability proposed in the elegant work of Maurer et al. [17] has proven to be a powerful tool for studying the security of hash function and many other primitives. The notion extends the classical concept of indistinguishability to the setting where one or more oracles involved in the construction are publicly available. The indifferentiability framework of [17] is built around random systems providing interfaces to other systems. Coron et al. [9] demonstrate a strengthened[7] indifferentiability framework built around Interactive Turing Machines (as in [6]). Our presentation borrows heavily from [9]. In the next subsection, we will introduce our new notion, *crooked indifferentiability*.

Defining indifferentiability. An *ideal primitive* is an algorithmic entity which receives inputs from one of the parties and returns its output immediately to the querying party. We now proceed to the definition of indifferentiability [17,9]:

Definition 4 (Indifferentiability [17,9]). *A Turing machine C with oracle access to an ideal primitive $\mathcal{G}$ is said to be $(t_{\mathcal{D}}, t_{\mathcal{S}}, q, \epsilon)$-indifferentiable from an ideal primitive $\mathcal{F}$, if there is a simulator $\mathcal{S}$, such that for any distinguisher $\mathcal{D}$, it holds that :*

$$\left| \Pr[\mathcal{D}^{C,\mathcal{G}}(1^n) = 1] - \Pr[\mathcal{D}^{\mathcal{F},\mathcal{S}}(1^n) = 1] \right| \leq \epsilon \,.$$

The simulator $\mathcal{S}$ has oracle access to $\mathcal{F}$ and runs in time at most $t_{\mathcal{S}}$. The distinguisher $\mathcal{D}$ runs in time at most $t_{\mathcal{D}}$ and makes at most q queries. Similarly, $C^{\mathcal{G}}$ is said to be (computationally) indifferentiable from $\mathcal{F}$ if ϵ is a negligible function of the security parameter n (for polynomially bounded $t_{\mathcal{D}}$ and $t_{\mathcal{S}}$). See Figure 3.

As illustrated in Figure 3, the role of the simulator is to simulate the ideal primitive $\mathcal{G}$ so that no distinguisher can tell whether it is interacting with C and $\mathcal{G}$, or with $\mathcal{F}$ and $\mathcal{S}$; in other words, the output of $\mathcal{S}$ should look "consistent" with what the distinguisher can obtain from $\mathcal{F}$. Note that the simulator does not observe the distinguisher's queries to $\mathcal{F}$; however, it can call $\mathcal{F}$ directly when needed for the simulation. Note that, in some sense, the simulator must "reverse engineer" the construction C so that the simulated oracle appropriately induces $\mathcal{F}$ and, of course, possesses the correct marginal distribution.

---

[7] Technically, the quantifiers in the security definitions in the original [17] and in the followup [9] are different; in the former, a simulator needs to be constructed for each adversary, while in the latter a simulator needs to be constructed for *all* adversaries.
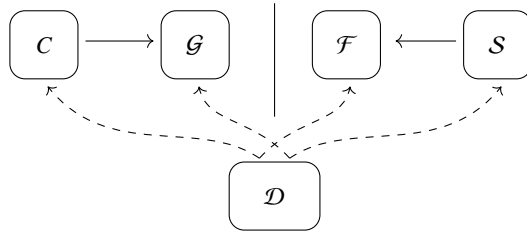
Fig. 3: The indifferentiability notion: the distinguisher $\mathcal{D}$ either interacts with algorithm C and ideal primitive $\mathcal{G}$, or with ideal primitive $\mathcal{F}$ and simulator $\mathcal{S}$. Algorithm C has oracle access to $\mathcal{G}$, while simulator $\mathcal{S}$ has oracle access to $\mathcal{F}$.

Replacement. It is shown in [17] that if $C^{\mathcal{G}}$ is indifferentiable from $\mathcal{F}$, then $C^{\mathcal{G}}$ can replace $\mathcal{F}$ in any cryptosystem, and the resulting cryptosystem is at least as secure in the $\mathcal{G}$ model as in the $\mathcal{F}$ model.

We use the definition of [17] to specify what it means for a cryptosystem to be at least as secure in the $\mathcal{G}$ model as in the $\mathcal{F}$ model. A cryptosystem is modeled as an Interactive Turing Machine with an interface to an adversary $\mathcal{A}$ and to a public oracle. The cryptosystem is run by an environment $\mathcal{E}$ which provides a binary output and also runs the adversary. In the $\mathcal{G}$ model, cryptosystem $\mathcal{P}$ has oracle access to C whereas attacker $\mathcal{A}$ has oracle access to $\mathcal{G}$. In the $\mathcal{F}$ model, both $\mathcal{P}$ and $\mathcal{A}$ have oracle access to $\mathcal{F}$. The definition is illustrated in Figure 4.
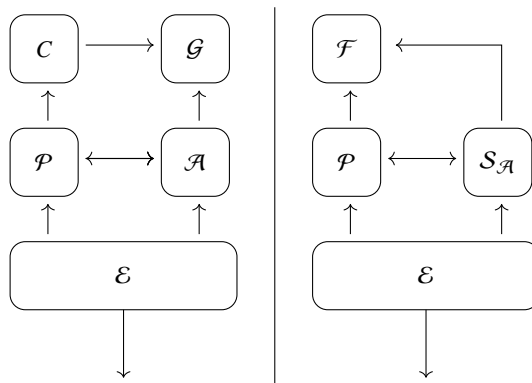
Fig. 4: The environment $\mathcal{E}$ interacts with cryptosystem $\mathcal{P}$ and attacker $\mathcal{A}$. In the $\mathcal{G}$ model (left), $\mathcal{P}$ has oracle access to C whereas $\mathcal{A}$ has oracle access to $\mathcal{G}$. In the $\mathcal{F}$ model, both $\mathcal{P}$ and $\mathcal{S}_{\mathcal{A}}$ have oracle access to $\mathcal{F}$.

Definition 5. *A cryptosystem is said to be at least as secure in the $\mathcal{G}$ model with algorithm C as in the $\mathcal{F}$ model, if for any environment $\mathcal{E}$ and any attacker $\mathcal{A}$ in the $\mathcal{G}$ model, there exists an attacker $\mathcal{S}_{\mathcal{A}}$ in the $\mathcal{F}$ model, such that:*

$$\Pr[\mathcal{E}(\mathcal{P}^C, \mathcal{A}^{\mathcal{G}}) = 1] - \Pr[\mathcal{E}(\mathcal{P}^{\mathcal{F}}, \mathcal{S}_{\mathcal{A}}^{\mathcal{F}}) = 1] \leq \epsilon.$$

where ϵ is a negligible function of the security parameter n. Similarly, a cryptosystem is said to be computationally at least as secure, etc., if $\mathcal{E}$, $\mathcal{A}$ and $\mathcal{S}_{\mathcal{A}}$ are polynomial-time in n.

We have the following security preserving (replacement) theorem, which says that when an ideal primitive is replaced by an indifferentiable one, the security of the "big" cryptosystem remains:

Theorem 6 ([17,9]). *Let $\mathcal{P}$ be a cryptosystem with oracle access to an ideal primitive $\mathcal{F}$. Let C be an algorithm such that $C^{\mathcal{G}}$ is indifferentiable from $\mathcal{F}$. Then cryptosystem $\mathcal{P}$ is at least as secure in the $\mathcal{G}$ model with algorithm C as in the $\mathcal{F}$ model.*

A.2 Abbreviated crooked indifferentiability

Definition 6 (Abbreviated crooked indifferentiability). *The abbreviated model calls for the distinguisher to provide the subversion algorithm $\tilde{h}$ at the outset (without the advantage of any preliminary queries to h). Thus, the abbreviated model consists only of the last phase of the full model. Formally, in the abbreviated model the distinguisher is provided as a pair $(\widehat{\mathcal{D}}, \tilde{h})$, the random string R is drawn (as in the full model), and insecurity is expressed as the difference between the behavior of $\widehat{\mathcal{D}}$ on the pair $(C^{\tilde{h}}(R), h)$ and the pair $(\mathcal{F}, \mathcal{S}^{\mathcal{F}}(R, \langle \tilde{h} \rangle))$. Specifically, the construction C is $(n_{\text{source}}, n_{\text{target}}, q_{\widehat{\mathcal{D}}}, q_{\tilde{h}}, r, \epsilon)$-Abbreviated-crooked-indifferentiable from ideal primitive $\mathcal{F}$ if there is an efficient simulator $\mathcal{S}$ so that for any crooked-distinguisher $\widehat{\mathcal{D}}$ making no more than $q_{\widehat{\mathcal{D}}}(n)$ queries and subversion algorithm $\tilde{h}$ making no more than $q_{\tilde{h}}(n)$ queries, the real execution and the ideal execution are indistinguishable:*

$$\left| \Pr_{u,R,h} \left[\widehat{\mathcal{D}}^{C^{\tilde{h}}(R),h}(1^n, R) = 1 \right] - \Pr_{u,R,\mathcal{F}} \left[\widehat{\mathcal{D}}^{\mathcal{F},\mathcal{S}^{\mathcal{F}}(R,\langle \tilde{h} \rangle)}(1^n, R) = 1 \right] \right| \leq \epsilon(n).$$

Here R denotes a random string of length $r(n)$ and both $h : \{0,1\}^{n_{\text{source}}} \to \{0,1\}^{n_{\text{source}}}$ and $\mathcal{F} : \{0,1\}^{n_{\text{target}}} \to \{0,1\}^{n_{\text{target}}}$ denote random functions where $n_{\text{source}}(n)$ and $n_{\text{target}}(n)$ are polynomials in the security parameter n. We let u denote the random coins of $\widehat{\mathcal{D}}$. The simulator is efficient in the sense that it is polynomial in n and the running time of the supplied algorithm $\tilde{h}$ (on inputs of length n_{source}).

Observe that while the abbreviated simulator is a fixed algorithm, its running time may depend on the running time of $\tilde{h}$—in particular, the definition permits $\mathcal{S}$ sufficient running time to simulate $\tilde{h}$ on a polynomial number of inputs.

Regarding the difference between the full and abbreviated crooked-indifferentiability, observe that the distinguisher can "compile into" the subversion algorithm $\tilde{h}$ any queries and pre-computation that might have been advantageous to carry out in phase

I; such queries and pre-computation can also be mimicked by the distinguisher itself. This technique can effectively simulate the two phase execution with a single phase. Nevertheless, the models do make slightly different demands on the simulator which must be prepared to answer some queries (in Phase I) prior to knowledge of R and $\tilde{h}$.

A.3 Replacement with Crooked indifferentiability [20]

Security preserving (replacement) has been shown in the indifferentiability framework [17]: if $C^{\mathcal{G}}$ is indifferentiable from $\mathcal{F}$, then $C^{\mathcal{G}}$ can replace $\mathcal{F}$ in any cryptosystem, and the resulting cryptosystem in the $\mathcal{G}$ model is at least as secure as that in the $\mathcal{F}$ model. We next show that the replacement property also holds in the crooked indifferentiability framework.

Recall that, in the "standard" indifferentiability framework [17,9], a cryptosystem can be modeled as an Interactive Turing Machine with an interface to an adversary $\mathcal{A}$ and to a public oracle. There the cryptosystem is run by a "standard" environment $\mathcal{E}$. In our "crooked" indifferentiability framework, a cryptosystem has the interface to an adversary $\mathcal{A}$ and to a public oracle. However, now the cryptosystem is run by a crooked-environment $\widehat{\mathcal{E}}$.

Consider an ideal primitive $\mathcal{G}$. Similar to the $\mathcal{G}$-crooked-distinguisher, we can define the $\mathcal{G}$-crooked-environment $\widehat{\mathcal{E}}$ as follows: Initially, the crooked-environment $\widehat{\mathcal{E}}$ manufactures and then publishes a subverted implementation of the ideal primitive $\mathcal{G}$, and denotes it $\tilde{\mathcal{G}}$. Then $\widehat{\mathcal{E}}$ runs the attacker $\mathcal{A}$, and the cryptosystem $\mathcal{P}$ is developed. In the $\mathcal{G}$ model, cryptosystem $\mathcal{P}$ has oracle access to C whereas attacker $\mathcal{A}$ has oracle access to $\mathcal{G}$; note that, C has oracle access to $\tilde{\mathcal{G}}$, not to directly $\mathcal{G}$. In the $\mathcal{F}$ model, both $\mathcal{P}$ and $\mathcal{A}$ have oracle access to $\mathcal{F}$. Finally, the crooked-environment $\widehat{\mathcal{E}}$ returns a binary decision output. The definition is illustrated in Figure 5.

Definition 7. *Consider ideal primitives $\mathcal{G}$ and $\mathcal{F}$. A cryptosystem $\mathcal{P}$ is said to be at least as secure in the $\mathcal{G}$-crooked model with algorithm C as in the $\mathcal{F}$ model, if for any $\mathcal{G}$-crooked-environment $\widehat{\mathcal{E}}$ and any attacker $\mathcal{A}$ in the $\mathcal{G}$-crooked model, there exists an attacher $\mathcal{S}_{\mathcal{A}}$ in the $\mathcal{F}$ model, such that:*

$$\Pr[\widehat{\mathcal{E}}(\mathcal{P}^{C^{\tilde{\mathcal{G}}}}, \mathcal{A}^{\mathcal{G}}) = 1] - \Pr[\widehat{\mathcal{E}}(\mathcal{P}^{\mathcal{F}}, \mathcal{S}_{\mathcal{A}}^{\mathcal{F}}) = 1] \le \epsilon.$$

where ϵ is a negligible function of the security parameter n.

The following theorem was given in [20] which shows that security is preserved when replacing an ideal primitive by a crooked-indifferentiable one:

Theorem 7 ([20]). *Consider an ideal primitive $\mathcal{G}$ and a $\mathcal{G}$-crooked-environment $\widehat{\mathcal{E}}$. Let $\mathcal{P}$ be a cryptosystem with oracle access to an ideal primitive $\mathcal{F}$. Let C be an algorithm such that $C^{\mathcal{G}}$ is $\mathcal{G}$-crooked-indifferentiable from $\mathcal{F}$. Then cryptosystem $\mathcal{P}$ is at least as secure in the $\mathcal{G}$-crooked model with algorithm C as in the $\mathcal{F}$ model.*

Proof. The proof is very similar to that in [17,9]. Let $\mathcal{P}$ be any cryptosystem, modeled as an Interactive Turing Machine. Let $\widehat{\mathcal{E}}$ be any crooked-environment, and $\mathcal{A}$ be any attacker in the $\mathcal{G}$-crooked model. In the $\mathcal{G}$-crooked model, $\mathcal{P}$ has oracle access to C (who
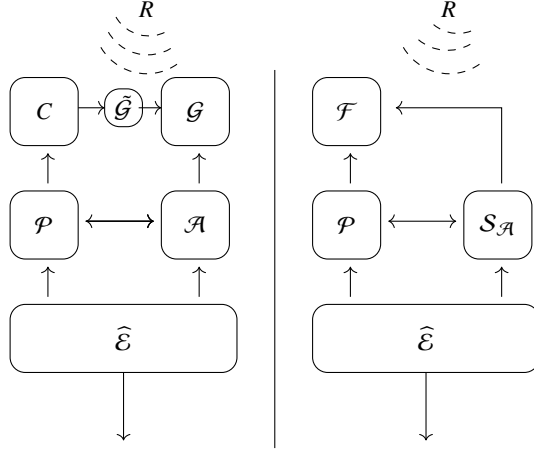
Fig. 5: The environment $\widehat{\mathcal{E}}$ interacts with cryptosystem $\mathcal{P}$ and attacker $\mathcal{A}$: In the $\mathcal{G}$ model (left), $\mathcal{P}$ has oracle access to C whereas $\mathcal{A}$ has oracle access to $\mathcal{G}$; the algorithm C has oracle access to the subverted $\tilde{\mathcal{G}}$. In the $\mathcal{F}$ model, both $\mathcal{P}$ and $\mathcal{S}_{\mathcal{A}}$ have oracle access to $\mathcal{F}$. In addition, in both $\mathcal{G}$ and $\mathcal{F}$ models, randomness R is publicly available to all entities.

has oracle access to $\tilde{\mathcal{G}}$, not to directly $\mathcal{G}$.) whereas $\mathcal{A}$ has oracle access to ideal primitive $\mathcal{G}$; moreover crooked-environment $\widehat{\mathcal{E}}$ interacts with both $\mathcal{P}$ and $\mathcal{A}$. This is illustrated in Figure 6 (left part).

Since C is crooked-indifferentiable from $\mathcal{F}$ (see Figure 2), one can replace $(C^{\tilde{\mathcal{G}}}, \mathcal{G})$ by $(\mathcal{F}, \mathcal{S})$ with only a negligible modification of the crooked-environment $\widehat{\mathcal{E}}$'s output distribution. As illustrated in Figure 6, by merging attacker $\mathcal{A}$ and simulator $\mathcal{S}$, one obtains an attacker $\mathcal{S}_{\mathcal{A}}$ in the $\mathcal{F}$ model, and the difference in $\widehat{\mathcal{E}}$'s output distribution is negligible. □

Similar proof can be used to show the following corollary.

Corollary 1 (Proof of the warm-up construction.). *Let $\mathcal{G}$, $\mathcal{F}_1$ and $\mathcal{F}_2$ be three ideal primitives. Suppose there are two algorithms C_1 and C_2 such that $C_1^{\mathcal{G}}$ is crooked-indifferentiable from $\mathcal{F}_1$ and $C_2^{\mathcal{F}_1}$ is indifferentiable from $\mathcal{F}_2$. Then $C^{\mathcal{G}}$ is crooked-indifferentiable from $\mathcal{F}_2$ for $C := C_1^{C_2}$.*

Restrictions (of using crooked indifferentiability). Ristenpart et al. [18] has demonstrated that the replacement/composition theorem (Theorem 6) in the original indifferentiability framework only holds in single-stage settings. We remark that, the same restriction also applies to our replacement/composition theorem (Theorem 7). We leave it as our future work to extend our crooked indifferentiability to the multi-stage settings where disjoint adversaries are split over several stages.
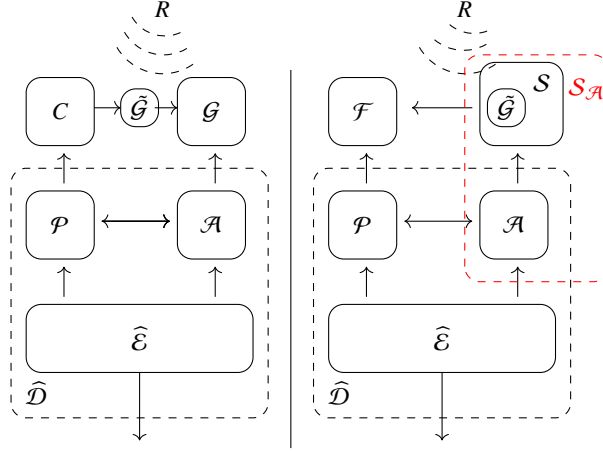
Fig. 6: Construction of attacker $\mathcal{S}_{\mathcal{A}}$ from attacker $\mathcal{A}$ and simulator $\mathcal{S}$

B A Complete Proof

The formal proof consists of four steps:

1. First, we define the simulator for the security proof in Section B.1 and introduce a sequence of intermediate games from Section B.2 to Section B.5 that build connections between the simulator's game and the construction. Using the game transition, we show that the security proof can be reduced to controlling the probability of two bad events (assuming the simulator is efficient). We address challenge 4 here.
2. Second, from Section B.6 to B.7, we formally prove the bad events are negligible.
3. Third, we prove the simulator is efficient in Section B.8.
4. Finally, we lift the security in the abbreviated model to the full model in Section B.9.

The security proof we have is applicable to any $\ell \geq 2000n/\log(1/\epsilon)$. However, for the sake of brevity, we will initially consider $\ell = 8n$ and then later replace it with the more precise value of $\ell = 2000n/\log(1/\epsilon)$ towards the conclusion of the proof.

B.1 The Detailed Definition of the Simulator

The simulator provides an interface $\mathcal{S}.CF(i, x)$ to query the simulated random function CF_i on input x. As mentioned above, for each i the simulator internally maintains a table whose entries are pairs (x, y) of n-bit strings; each such entry intuitively determines a simulated value of CF at a particular point: in particular, if the pair (x, y) appears then any query to $\mathcal{S}.CF(i, x)$ returns the value y. The simulator maintains the natural invariants described previously: responses provided to the distinguisher are always consistent with the table and, furthermore, once an entry has been added to the table, it is never removed or changed. Note that in many cases the table will reflect function values that have not been queried by the distinguisher. We denote the ith table by $\mathcal{S}.CF_i$ and write $x \in \mathcal{S}.CF_i$

37

whenever x is a preimage in this table, often identifying $\mathcal{S}.CF_i$ with the set of preimages stored. When $x \in \mathcal{S}.CF_i$, $CF_i(x)$ denotes the corresponding image. We also denote the collection of all these $\mathcal{S}.CF_i$ tables by $\mathcal{S}.CF$. We use the notation $(i, x) \in \mathcal{S}.CF$ when $x \in \mathcal{S}.CF_i$.

For each i, we additionally define a table $\mathcal{S}.\tilde{CF}_i$ induced implicitly by $\mathcal{S}.CF$. As with $\mathcal{S}.CF_i$, the table $\mathcal{S}.\tilde{CF}_i$ consists of pairs of inputs and outputs of $\tilde{CF}_i$. We write $x \in \mathcal{S}.\tilde{CF}_i$ when all queries generated by evaluation of $\tilde{CF}_i(x)$ are defined in $\mathcal{S}.CF$; naturally, the corresponding function value determines the pair (x, y) in the table. The collection of all of these $\mathcal{S}.\tilde{CF}_i$ is denoted by $\mathcal{S}.\tilde{CF}$. (Note that this table is not maintained explicitly by the simulator, but rather determined implicitly by $\mathcal{S}.CF$.)

Handling queries to $\mathcal{S}.CF$. On a query $\mathcal{S}.CF(i, x)$, the simulator first checks whether $x \in \mathcal{S}.CF_i$. If so, it answers with $CF_i(x)$. Otherwise the simulator picks a random value y and inserts (x, y) into $\mathcal{S}.CF_i$. The process above is done by a procedure called $\mathcal{S}.CF^{\text{Inner}}$ which takes input (i, x).) After this, the simulator takes further steps to ensure that its future answers are consistent with the permutation P. Only after this consistency maintenance step is the value y finally returned.

To ensure consistency, the simulator considers all newly generated unsubverted chains with length $n/10$ that terminate at the last-queried position; for a newly evaluated term $CF_s(x_s)$, these chains of interest either have the form $(s, x_s, ..., x_{s+n/10-1})$ or $(s - n/10 + 1, x_{s-n/10+1}, ..., x_s)$. Each such detected chain is enqueued by the simulator in a "completion queue," identifying the chain for future completion.

The simulator then repeats the following completion step until the queue is emptied. (When the queue is finally empty, the simulator returns the answer y to the initial query.)

1. **Detection Step.** The first chain $c = (s, x_s, ..., x_{s+n/10-1})$ is removed from the queue. A procedure called $\mathcal{S}.$HonestyCheck is then run on the chain. The procedure $\mathcal{S}.$HonestyCheck evaluates $\tilde{CF}$ values of the elements of c and generates a four-tuple (s, x_s, x_{s+1}, u) for future completion if all the elements in c are honest. (In fact, not all chains removed from the queue are processed by $\mathcal{S}.$HonestyCheck. A chain removed from the queue is processed by $\mathcal{S}.$HonestyCheck only if it is disjoint with all the chains that are previously processed by $\mathcal{S}.$HonestyCheck and is disjoint with all the previously completed full subverted chains. Any chain that is not processed by $\mathcal{S}.$HonestyCheck is discarded. The procedure that decides whether a chain is going to be discarded or processed by $\mathcal{S}.$HonestyCheck is called $\mathcal{S}.$Check.) In the tuple (s, x_s, x_{s+1}, u), the value s ensures that later the simulator knows that the first value x_s corresponds to CF_s. The value u describes where to adapt (that is, program) the values of CF_* in order to ensure consistency with the given permutation: this will occur at positions u and $u + 1$. The convention for determining u is straightforward: If $s > 5n$ or $s + n/10 - 1 < 3n$, then there is "plenty of space around $4n$," and $u = 4n$; otherwise, $u = 7n$.

2. **Completion Step.** Finally, the simulator takes the four-tuple (s, x_s, x_{s+1}, u) and *completes* the subverted chain related to (s, x_s, x_{s+1}). Intuitively, this means that the chain is determined by iteratively determining neighbouring values of $\tilde{CF}(x)$ by evaluating the subversion algorithm and, when necessary, carrying out internal calls to $CF_i()$ in order to answer queries made by that algorithm to the F_i. This

38

iterative process is continued, using P to "wrap around," until the only remaining undetermined values appear at positions u and $u + 1$; at this point, the values at u and $u + 1$ are programmed to ensure consistency. In more detail: Assuming that $u < s$, the completion process (conducted by a procedure called $\mathcal{S}.\text{Complete}$) proceeds as follows.

- The initial chain consists of the two adjacent values x_s, x_{s+1}.
- $C\tilde{F}_{s+1}(x_{s+1})$ is determined by simulating the subversion algorithm which generates oracle queries to CF to be answered using $\mathcal{S}.CF$. (Note that this process may enqueue new chains for completion.) The value $x_{s+2} = x_s \oplus C\tilde{F}_{s+1}(x_{s+1})$ is then determined, yielding the enlarged chain (x_s, x_{s+1}, x_{s+2}). This process is repeated until the chain is extended maximally "to the right" so that it has the form $(x_s, x_{s+1}, \ldots, x_{8n}, x_{8n+1})$.
- P^{-1} is then applied to x_{8n}, x_{8n+1} to yield x_0, x_1.
- Starting from the pair (x_0, x_1), this process is repeated, as above, to yield values for $x_2, \ldots, x_u$. Note that $x_u = x_{u-2} \oplus C\tilde{F}(x_{u-1})$ so that $C\tilde{F}(x_u)$ is never evaluated during this process (which is to say that the subversion algorithm is never simulated on x_u).
- Similarly, the original pair x_s, x_{s-1} is extended "to the left" to determine the values $x_{s-1}, \ldots, x_{u+1}$; as above, x_{u+1} is determined by $x_{u+3} \oplus C\tilde{F}(x_{u+2})$, so that $C\tilde{F}(x_{u+1})$ is never evaluated.
- Then, the simulator defines $CF_u(x_u)$ and $CF_{u+1}(x_{u+1})$ that is consistent with P, i.e., $CF_u(x_u) := x_{u-1} \oplus x_{u+1}$ and $CF_{u+1}(x_{u+1}) := x_u \oplus x_{u+2}$. The game aborts if either of these is defined from a previous action of $\mathcal{S}$. If the game does not abort, the simulator evaluates the subversion algorithm on both x_u and x_{u+1}. During this evaluation, the values $CF_u(x_u)$ and $CF_{u+1}(x_{u+1})$ are already determined; other queries are answered using $\mathcal{S}.CF$ as above. The game aborts if (u, x_u) and $(u + 1, x_{u+1})$ are dishonest; otherwise, the chain is a valid subverted chain (and consistent with P).
- A set $\mathcal{S}.\text{CompletedChains}$ is maintained to store the chains that are completed: for any (i, x_i, x_{i+1}) $(1 \le i \le 8n - 1)$, $\mathcal{S}$ updates

$$\mathcal{S}.\text{CompletedChains} := \mathcal{S}.\text{CompletedChains} \cup (i, x_i, x_{i+1}).$$

The alternative case, when $u > s + 1$, is treated analogously.

For a pseudocode description of the simulator, please see page .

Notice that we achieve freshness and honesty if the simulator above does not abort. However, even if we can prove the simulation of $\mathcal{S}$ is perfect (i.e., it does not abort) and $\mathcal{S}$ is efficient, it is still not a rigorous proof for the fact that the simulated world cannot be distinguished from the real world. We provide a rigorous proof below by using a game transition approach.

B.2 Game Transition Approach: Some preparations

Our overall purpose is to show that for any deterministic distinguisher $\mathcal{D}$ that make at most $q_\mathcal{D}$ queries (where $q_\mathcal{D}$ is some polynomial function in n), the probability that $\mathcal{D}$

outputs 1 when interacting with $(P, \mathcal{S}^P)$ differs negligibly from the probability it outputs 1 when interacting with (C^F, F). Here C is the construction in Section 3, and F is a collection of $8n$ uniform functions. We also wish to establish that, with overwhelming probability, only a polynomial number of terms are evaluated by $\mathcal{S}$ (or P) when $\mathcal{D}$ interacts with $(P, \mathcal{S}^P)$.

We denote the game where the distinguisher $\mathcal{D}$ interacts with $(P, \mathcal{S}^P)$ by G_1 (the ideal world), and the game where $\mathcal{D}$ interacts with (C^F, F) by G_6 (the real world). We will introduce four intermediate games, G_2, G_3, G_4 and G_5, in the following narrative to study the relationship between G_1 and G_6. When we use the term "G_i" ($i = 1, ..., 6$), we always have a fixed deterministic distinguisher (denoted by $\mathcal{D}$) in mind, without mentioning it explicitly. Also, whenever we prove a statement about "G_i," this is understood to mean that the statement holds every fixed distinguisher that issues at most $q_{\mathcal{D}}$ queries in the setting defined by G_i (including the queries to CF and to the ideal object).

In the description of G_1, we defined various concepts (e.g., unsubverted/subverted chains, honest, $Q_i(x_i)$, etc.) that are used to describe the behavior of the simulator $\mathcal{S}$ (or its table $\mathcal{S}.CF$). In the rest of the paper, we likewise will apply these concepts to describe the simulators in other games. To avoid confusion, we will specify the simulator (or its counterpart) we are working with when using these concepts.

In the following, we say a game is *efficient* if, with overwhelming probability, only a polynomial size of terms are evaluated by the simulator or the ideal object when the game ends. To prove G_1 is efficient and is indistinguishable from G_6, our plan is to show the following statements:

1. G_1 vs G_2: assuming G_2 is efficient, G_1 is efficient and is indistinguishable from G_2.
2. G_2 vs G_3: assuming G_3 is efficient, G_2 is efficient and the total variation distance between the transcript of G_2 and G_3 is 0.
3. G_3 vs G_4: assuming G_4 is efficient, the total variation distance between the transcript of the distinguisher in G_3 and G_4 is bounded by the probability of the bad events in G_4, and G_3 is efficient if the probability of the bad events in G_4 is negligible.
4. G_4 vs G_5: assuming G_5 is efficient, G_4 is efficient and the total variation distance between the transcript of G_4 and G_5 is 0.
5. G_5 vs G_6: the total variation distance between the transcript of G_5 and G_6 is bounded by the probability of the bad events in G_5.
6. Bounding bad events: assuming G_5 is efficient, the probability of the bad events in G_4 and G_5 is negligible.
7. Efficiency: G_5 is efficient.
8. Full model: The crooked-indifferentiability in the abbreviated model can be lifted to the full model.

It is easy to see how these results together imply the efficiency of simulator $\mathcal{S}$ in G_1 and the crooked-indifferentiability of our construction (G_1 is indistinguishable from G_6). Statements 1 to 5 will be formally defined and proved as we introduce new games. Statement 6, 7 and 8 will be proved in B.7, B.8 and B.9, respectfully.

B.3 The Second Game

Description of the Second Game To obtain G_2 from G_1, we replace the random permutation P by a "two-sided random function" RF. RF is a system that provides

answers to "RF queries" and "RF^{-1} queries." Ideally, the system would maintain a consistent, partially defined bijection that is extended independently and uniformly whenever a "fresh value" is requested. These constraints are, of course, impossible to guarantee perfectly beyond the first query, so the system adopts a particular convention for subsequent answers that yields a probability distribution that is nearly indistinguishable from a random permutation so long as the number of queries is polynomial. Specifically, the system maintains an initially empty table with entries of the form $(\downarrow, \alpha, \beta)$, intuitively indicating that $RF(\alpha) = \beta$, or the form $(\uparrow, \alpha, \beta)$, intuitively indicating that $RF^{-1}(\beta) = \alpha$. Under expected circumstances, these tables will provide the guarantees mentioned above, defining a consistent, partially defined permutation. A query of the form $RF(\alpha)$ is answered as follows: (i.) if there is a tuple $(\downarrow, \alpha, \beta)$ in the table, β is returned, (ii.) otherwise, a uniformly random value β is drawn and both $(\downarrow, \alpha, \beta)$ and $(\uparrow, \alpha, \beta)$ are added to the table and, (ii.') in the event that there is a tuple $(\uparrow, \alpha', \beta)$ in the table for some $\alpha \neq \alpha'$, this is removed. Queries to RF^{-1} are handled similarly. Of course, it is natural to expect and easy to prove that such collisions occur only with negligible probability. In the absence of such collisions RF behaves like a random permutation.

See page 85 for the pseudocode description of G_2.

The Gap Between the First Game and the Second Game To understand the gap between G_1 and G_2, we notice that the only difference between the two games is the ideal object. The following lemma shows that the two ideal objects, P and RF are indistinguishable for a distinguisher that issues polynomial queries.

Lemma 15 (P **and** RF **are indistinguishable**). *A distinguisher $\mathcal{D}'$ that issues q' queries to either P or RF has advantage at most $(q')^2/2^{2n}$ to distinguish the two objects.*

Proof. We denote the transcripts of $\mathcal{D}'$ interacting with P and RF by T_P and T_{RF} respectively. We say T_P (T_{RF}) is equal to t for a certain bitstring t if it reflects the transcript correctly. We say a query $A(\alpha)$ ($A = P, P^{-1}, RF, RF^{-1}$) is *redundant* if $\mathcal{D}'$ has queried $A(\alpha)$ or $A^{-1}(\beta)$ ($= \alpha$) before. In the proof, without loss of generality, we assume $\mathcal{D}'$ does not make any redundant queries.

We bound the advantage of the distinguisher by bounding the total variation distance between the two scenarios. We say a bitstring t is *good* if when $T_{RF} = t$, there is no overwrite in the execution of RF. t is called *bad* if it is not good. We notice that if t is good it behaves like a transcript of a permutation. Then, for any good t,

$$\Pr[T_P = t] = \prod_{i=1}^{q'} \frac{1}{2^{2n} - i + 1} > \frac{1}{2^{2nq'}} = \Pr[T_{RF} = t].$$

Let α_P and α_{RF} denote the distribution of the T_P and T_{RF}. The above observation shows that

$$\|\alpha_P - \alpha_{RF}\|_{tv} \leq \Pr[T_{RF} = t \text{ for a bad } t].$$

Now we proceed to bound the probability that T_{RF} is equal to a bad string. By definition, T_{RF} is bad if and only if there is an overwrite in the execution of RF. Since RF are queried for q' times, the probability that there is an overwrite is less than $(q')^2/2^{2n}$. $\qquad\square$

41

Lemma 16 (The Gap between G_1 and G_2). *If G_2 is efficient, then G_1 is efficient and the probability of $\mathcal{D}$ outputting 1 in G_2 differs negligibly from that in G_1.*

Proof. The lemma is implied directly by Lemma 15. □

B.4 The Third and the Fourth Game

We will put the descriptions of G_3 and G_4 together since they have similar structures. For convenience, we will introduce G_4 first and then make some changes to G_4 to arrive at G_3.

Description of the Fourth Game In G_4, we adopt a new simulator O^2 with two inbound parties $\mathcal{S}^2$ and $\mathcal{M}^2$. (In the rest of the paper, we will use different superscripts for the simulators and parties in different games.) $\mathcal{S}^2$ is a direct analogue of $\mathcal{S}$ in G_2. $\mathcal{M}^2$ is a party that knows the distinguisher's query to the ideal object RF. In other words, the simulator O^2 in G_4 is a "public" simulator, which means it answers the CF queries from $\mathcal{D}$ and at the same time, knows all queries made by $\mathcal{D}$ to RF.

O^2 maintains two tables of CF values (denoted by $\mathcal{S}^2.CF$ and $\mathcal{M}^2.CF$), which have the same format as $\mathcal{S}.CF$ in G_2. The CF values in the table $\mathcal{S}^2.CF$ are evaluated in a way such that it is a subset of $\mathcal{M}^2.CF$ unless the game aborts: roughly speaking, when O^2 evaluates a term $CF_i(x)$ in $\mathcal{S}^2.CF$, it inserts the same value to the table $\mathcal{M}^2.CF$; when O^2 needs to evaluate a previously undefined term $CF_i(x)$ in $\mathcal{S}^2.CF$ and $(i, x) \in \mathcal{M}^2.CF$, it typically copies $CF_i(x)$ in $\mathcal{M}^2.CF$ to $\mathcal{S}^2.CF$.

Like $\mathcal{S}$ in G_2, both $\mathcal{S}^2$ and $\mathcal{M}^2$ have two important procedures: the "evaluation procedure" $\mathcal{S}^2.CF^{\text{Inner}}$ ($\mathcal{M}^2.CF^{\text{Inner}}$, respectively) takes care of query to the table $\mathcal{S}^2.CF$ ($\mathcal{M}^2.CF$, respectively), and the "completion procedure" $\mathcal{S}^2.\text{Complete}$ ($\mathcal{M}^2.\text{Complete}$, respectively) is used to *complete* chains in $\mathcal{S}^2.CF$ ($\mathcal{M}^2.CF$, respectively).

The two procedures of $\mathcal{M}^2$, $\mathcal{M}^2.CF^{\text{Inner}}$ and $\mathcal{M}^2.\text{Complete}$, only insert values into the table $\mathcal{M}^2.CF$, while the two procedures of $\mathcal{S}^2$ can insert values to both $\mathcal{S}^2.CF$ and $\mathcal{M}^2.CF$. For convenience, in the following description of $\mathcal{M}^2$'s procedures, we will just say a certain term $CF(i, x)$ is evaluated instead of saying it is evaluated in the table $\mathcal{M}^2.CF$. However, in the description of the procedures of $\mathcal{S}^2$, we will mention clearly whether CF is inserted into $\mathcal{S}^2.CF$ or $\mathcal{M}^2.CF$.

Handling $\mathcal{D}$'s query to the ideal object. When the distinguisher makes a query to RF on a $2n$-bit string (x_0, x_1), O^2 does nothing if $\mathcal{S}^2$ or $\mathcal{M}^2$ has queried $RF(x_0, x_1)$ or $RF^{-1}(x_{8n}, x_{8n+1})$ (for some (x_{8n}, x_{8n+1}) with $RF^{-1}(x_{8n}, x_{8n+1}) = (x_0, x_1)$) before. Otherwise, $\mathcal{M}^2$ proceeds to *complete* a chain in $\mathcal{M}^2.CF$. In more detail: Assuming that the distinguisher queries $RF(x_0, x_1)$, the "completion procedure" $\mathcal{M}^2.\text{Complete}$ proceeds as follows.

  – $C\tilde{F}_1(x_1)$ is determined by simulating the subversion algorithm. This will generate oracle queries to $CF_*()$ which are answered using $\mathcal{M}^2.CF$. (The way a query to $\mathcal{M}^2.CF$ is answered will be explained later.) The value $x_2 = x_0 \oplus C\tilde{F}_1(x_1)$ is then determined, yielding the chain $(1, x_1, x_2)$. This process is repeated until the chain is extended "to the right" to x_{4n} so that it has the form $(x_1, x_2, \dots, x_{4n-1}, x_{4n})$. Note

that $x_{4n} = x_{4n-2} \oplus C\tilde{F}(x_{4n-1})$ so that $C\tilde{F}(x_{4n})$ is never evaluated during this process (which is to say that the subversion algorithm is never simulated on x_u).

– RF is then applied to x_0, x_1 to yield x_{8n}, x_{8n+1}.
– Similarly, the pair (x_{8n}, x_{8n+1}) is extended "to the left" to determine the values for $x_{4n+1}, \ldots, x_{8n-1}$; as above, x_{4n+1} is determined by $x_{4n+3} \oplus C\tilde{F}(x_{4n+2})$, so that $C\tilde{F}(x_{4n+1})$ is never evaluated.
– Then, $\mathcal{M}^2$ defines $CF_{4n}(x_{4n})$ and $CF_{4n+1}(x_{4n+1})$ in such a way that consistency with P is ensured, i.e., $CF_{4n}(x_{4n}) := x_{4n-1} \oplus x_{4n+1}$ and $CF_{4n+1}(x_{4n+1}) := x_{4n} \oplus x_{4n+2}$. The game aborts if either of these is defined from a previous action of the simulator. If the game does not abort, $\mathcal{M}^2$ evaluates the subversion algorithm on both x_{4n} and x_{4n+1}. During this evaluation, the values $CF_{4n}(x_{4n})$ and $CF_{4n+1}(x_{4n+1})$ are already determined; other queries are answered using $\mathcal{M}^2.CF$ as above. The game aborts if for any i with $3n \le i \le 5n$, $(i, x_i) \in \bigcup_{j=1}^{8n} Q_j(x_j)/Q_i(x_i)$ or (i, x_i) is dishonest; otherwise, the chain is a valid subverted chain (and consistent with RF).
– A set $\mathcal{M}^2.\text{CompletedChains}$ is maintained to store the chains that are completed: for any (i, x_i, x_{i+1}) $(1 \le i \le 8n - 1)$, $\mathcal{M}^2$ updates $\mathcal{M}^2.\text{CompletedChains} := \mathcal{M}^2.\text{CompletedChains} \cup (i, x_i, x_{i+1})$.
– A set $\mathcal{M}^2.\text{MiddlePoints}$ is maintained to store the points with index between $3n$ and $5n$: for any (i, x_i) with $3n \le i \le 5n$, $\mathcal{M}^2$ updates $\mathcal{M}^2.\text{MiddlePoints} := \mathcal{M}^2.\text{MiddlePoints} \cup (i, x_i)$.
– A set $\mathcal{M}^2.\text{AdaptedPoints}$ is maintained to store the points with index $4n$ and $4n+1$: for (i, x_i) with $i = 4n$ or $4n+1$, $\mathcal{M}^2$ updates $\mathcal{M}^2.\text{AdaptedPoints} := \mathcal{M}^2.\text{AdaptedPoints} \cup (i, x_i)$.

O^2 reacts to the distinguisher's query to $RF^{-1}(\cdot)$ similarly.

Handling queries to $\mathcal{M}^2.CF$. On a query $\mathcal{M}^2.CF(i, x)$, $\mathcal{M}^2$ implements the evaluation procedure $\mathcal{M}^2.CF^{\text{Inner}}$:

– First, $\mathcal{M}^2$ checks whether $x \in \mathcal{S}^2.CF_i$. If so, it answers with $\mathcal{S}^2.CF_i(x)$.
– If $x \notin \mathcal{S}^2.CF_i$, $\mathcal{M}^2$ checks whether $x \in \mathcal{M}^2.CF_i$ and $(i, x) \notin \mathcal{M}^2.\text{MiddlePoints}$. If so, it answers with $\mathcal{M}^2.CF_i(x)$.
– If $x \notin \mathcal{S}^2.CF_i$, $x \in \mathcal{M}^2.CF_i$ and $(i, x) \in \mathcal{M}^2.\text{MiddlePoints}$, the game aborts.
– If $x \notin \mathcal{S}^2.CF_i$ and $x \notin \mathcal{M}^2.CF_i$, $\mathcal{M}^2$ picks a random value y and inserts (x, y) into $\mathcal{M}^2.CF_i$.

Notice that in the first case above, we do not need to check whether $x \in \mathcal{M}^2.CF_i$ because $\mathcal{S}^2.CF$ is a subset of $\mathcal{M}^2.CF$ (which will be clear later).

Handling $\mathcal{D}$'s query to CF In G_1 and G_2, we view the distinguisher's queries to CF as queries to the simulator's table $\mathcal{S}.CF$. Similarly, in G_4, the distinguisher's query to CF is viewed as a query to the table $\mathcal{S}^2.CF$. See below for a description of how O^2 handles queries to $\mathcal{S}^2.CF$.

Handling queries to $\mathcal{S}^2.CF$. On a query $\mathcal{S}^2.CF(i, x)$, $\mathcal{S}^2$ implements the procedure $\mathcal{S}^2.CF^{\text{Inner}}$:

– First, $\mathcal{S}^2$ checks whether $x \in \mathcal{S}^2.CF_i$. If so, it answers with $\mathcal{S}^2.CF_i(x)$.

43

- If $x \notin S^2.CF_i$, S^2 checks whether $x \in M^2.CF_i$ and $(i, x) \notin M^2.\text{MiddlePoints}$. If so, it assigns and answers with $S^2.CF_i(x) := M^2.CF_i(x)$. (This is the reason that $S^2.CF$ is always a subset of $M^2.CF$.)
- If $x \notin S^2.CF_i$, $x \in M^2.CF_i$ and $(i, x) \in M^2.\text{MiddlePoints}$, the game aborts.
- If $x \notin S^2.CF_i$ and $x \notin M^2.CF_i$, S^2 picks a random value y and inserts (x, y) into $S^2.CF_i$ and $M^2.CF_i$.
- After this, S^2 takes further steps to ensure that its future answers are consistent with the permutation RF. Only after this consistency maintenance step is the value y finally returned.

Completing chains in $S^2.CF$. For any input (s, x_s, x_{s+1}, u), the procedure $S^2.\text{Complete}$ proceeds as follows.

- if $(s, x_s, x_{s+1}) \notin M^2.\text{CompletedChains}$
  - $S^2.\text{Complete}$ first generates the two sequences $(x_0, \ldots, x_u)$ and $(x_{u+1}, \ldots, x_{8n+1})$ in the way as $S.\text{Complete}$ does in G_2.
  - Then, S^2 defines $CF_u(x_u) := x_{u-1} \oplus x_{u+1}$ and $CF_{u+1}(x_{u+1}) := x_u \oplus x_{u+2}$ in both $S^2.CF$ or $M^2.CF$. The game aborts if either of these is defined previously in $S^2.CF$ or $M^2.CF$. If the game does not abort, S^2 evaluates the subversion algorithm on both x_u and x_{u+1}. During this evaluation, the values $CF_u(x_u)$ and $CF_{u+1}(x_{u+1})$ are already determined; other queries are answered using $S^2.CF$ as above. The game aborts if for $i = u$ or $u + 1$, $(i, x_i) \in \bigcup_{j=1}^{8n} Q_j(x_j)/Q_i(x_i)$ or (i, x_i) is dishonest.
- if $(s, x_s, x_{s+1}) \in M^2.\text{CompletedChains}$ and $u = 7n$, the game aborts.
- if $(s, x_s, x_{s+1}) \in M^2.\text{CompletedChains}$ and $u = 4n$, S^2 copies Q_c from $M^2.CF$ to $S^2.CF$, where c is the full subverted chain containing (s, x_s, x_{s+1}) in $M^2.CF$. The game aborts if for either $i = u$ or $u + 1$, (i, x_i) was in $S^2.CF$ before this execution of $S^2.\text{Complete}$, $(i, x_i) \in \bigcup_{j=1}^{8n} Q_j(x_j)/Q_i(x_i)$, or (i, x_i) is dishonest.
- A set $S^2.\text{CompletedChains}$ is maintained to store the chains that are completed: for any (i, x_i, x_{i+1}) $(1 \le i \le 8n - 1)$, the simulator updates $S^2.\text{CompletedChains} := S^2.\text{CompletedChains} \cup (i, x_i, x_{i+1})$.

See page 88 for the pseudocode description of G_4.

Description of the Third Game G_3 is a small pivot from G_4 that does not abort in some cases where G_4 aborts. For clarity, we denote by O^1 the simulator in G_3 and by S^1 and M^1 the two parties of O^1. The parties of O^1 maintain similar tables and sets to their counterparts in G_4. (The formal notation for these tables and sets is determined by changing the identifier in the notations from S^2 to S^1 and from M^2 to M^1.)

The procedures in G_3 are similar to their counterparts in G_4 except that they abort in fewer cases. To aid the reader, we color the differences between G_4 and G_3 in red below.

The completion procedure $M^1.\text{Complete}$. $M^1.\text{Complete}$ is same as its counterpart in G_4 except that it never aborts:

44

– When $\mathcal{M}^1$.Complete programs $CF_{4n}(x_{4n})$ and $CF_{4n+1}(x_{4n+1})$, it does not abort even if either $CF_{4n}(x_{4n})$ or $CF_{4n+1}(x_{4n+1})$ is previously defined. It just assigns $CF_{4n}(x_{4n}) := x_{4n-1} \oplus x_{4n+1}$ and $CF_{4n+1}(x_{4n+1}) := x_{4n} \oplus x_{4n+2}$ (this may overwrite the value).
– Also, the game does not abort even if for any i with $3n \leq i \leq 5n$, $(i, x_i) \in \bigcup_{j=1}^{8n} Q_j(x_j)/Q_i(x_i)$ or (i, x_i) is dishonest.

The evaluation procedure $\mathcal{M}^1.CF^{Inner}$. On a query $\mathcal{M}^1.CF(i, x)$:

– First, $\mathcal{M}^1$ checks whether $x \in \mathcal{S}^1.CF_i$. If so, it answers with $\mathcal{S}^1.CF_i(x)$.
– If $x \notin \mathcal{S}^1.CF_i$, $\mathcal{M}^1$ checks whether $x \in \mathcal{M}^1.CF_i$ and $(i, x) \notin \mathcal{M}^1$.MiddlePoints. If so, it answers with $\mathcal{M}^1.CF_i(x)$.
– If $x \notin \mathcal{S}^1.CF_i$, $x \in \mathcal{M}^1.CF_i$ and $(i, x) \in \mathcal{M}^1$.MiddlePoints, it answers with $\mathcal{M}^1.CF_i(x)$.
– If $x \notin \mathcal{S}^1.CF_i$ and $x \notin \mathcal{M}^1.CF_i$, $\mathcal{M}^1$ picks a random value y and inserts (x, y) into $\mathcal{M}^1.CF_i$.

The evaluation procedure $\mathcal{S}^1.CF^{Inner}$. On a query $\mathcal{S}^1.CF(i, x)$:

– First, $\mathcal{S}^1$ checks whether $x \in \mathcal{S}^1.CF_i$. If so, it answers with $\mathcal{S}^1.CF_i(x)$.
– If $x \notin \mathcal{S}^1.CF_i$, $\mathcal{S}^1$ checks whether $x \in \mathcal{M}^1.CF_i$ and $(i, x) \notin \mathcal{M}^1$.AdaptedPoints. If so, it assigns and answers with $\mathcal{S}^1.CF_i(x) := \mathcal{M}^1.CF_i(x)$.
– If $x \notin \mathcal{S}^1.CF_i$, $x \in \mathcal{M}^1.CF_i$ and $(i, x) \in \mathcal{M}^1$.AdaptedPoints, $\mathcal{S}^1$ picks a random value y and inserts (x, y) into $\mathcal{S}^1.CF_i$.
– If $x \notin \mathcal{S}^1.CF_i$ and $x \notin \mathcal{M}^1.CF_i$, $\mathcal{S}^1$ picks a random value y and inserts (x, y) into $\mathcal{S}^1.CF_i$ and $\mathcal{M}^1.CF_i$.
– After this, $\mathcal{S}^1$ takes further steps to ensure that its future answers are consistent with the permutation RF. Only after this consistency maintenance step is the value y finally returned.

The completion procedure $\mathcal{S}^1.Complete$. For any input (s, x_s, x_{s+1}, u), the procedure $\mathcal{S}^1$.Complete proceeds as follows.

– if $(s, x_s, x_{s+1}) \notin \mathcal{M}^1$.CompletedChains
  • $\mathcal{S}^1$.Complete first generates the two sequences $(x_0, \ldots, x_u)$ and $(x_{u+1}, \ldots, x_{8n+1})$ in the same way as $\mathcal{S}$.Complete does in G_2.
  • Then, $\mathcal{S}^1$ defines $CF_u(x_u) := x_{u-1} \oplus x_{u+1}$ and $CF_{u+1}(x_{u+1}) := x_u \oplus x_{u+2}$ in both $\mathcal{S}^1.CF$ or $\mathcal{M}^1.CF$. (This may overwrite the values in $\mathcal{M}^1.CF$.) The game aborts if either of these was defined previously in $\mathcal{S}^1.CF$. If the game does not abort, $\mathcal{S}^1$ evaluates the subversion algorithm on both x_u and x_{u+1}. During this evaluation, the values $CF_u(x_u)$ and $CF_{u+1}(x_{u+1})$ are already determined; other queries are answered using $\mathcal{S}^1.CF$ as above. The game aborts if for $i = u$ or $u + 1$, $(i, x_i) \in \bigcup_{j=1}^{8n} Q_j(x_j)/Q_i(x_i)$ or (i, x_i) is dishonest.
– if $(s, x_s, x_{s+1}) \in \mathcal{M}^1$.CompletedChains, the procedure proceeds in the same way as the last case.
– A set $\mathcal{S}^1$.CompletedChains is maintained to store the chains that are completed: for any (i, x_i, x_{i+1}) $(1 \leq i \leq 8n - 1)$, the simulator updates $\mathcal{S}^1$.CompletedChains := $\mathcal{S}^1$.CompletedChains $\cup (i, x_i, x_{i+1})$.

See page 85 for the pseudocode description of G_3.

45

The Gap Between the Second and the Third Game We define the *status* of a game G_i ($i = 1, \ldots, 5$) at some point (if the game does not abort at this moment) to be the collection of the transcript of the distinguisher, the tables of the simulator and the table of the ideal object at this moment. (To make the definition work for G_1, we imagine that a table of the ideal object P is maintained in G_1 to store the evaluated terms of P.) We say the status of the game is equal to "Abort" at some moment if the game already aborts at this moment. Note that in G_i ($i = 1, \ldots, 6$) we always assume a fixed distinguisher $\mathcal{D}$ that issues at most $q_{\mathcal{D}}$ queries.

We say the status of G_i ($i = 1, \ldots, 5$) at some moment t_1 is same as the status of G_j ($j = 1, \ldots, 5$, $j \neq i$) at some moment t_2 if G_4 at t_1 and G_5 at t_2 have the same transcripts of the distinguisher, the same RF tables, and the same simulator tables. (When we compare the simulator tables between different games, we only compare a table with its counterpart in another game. For example, when we compare the status of G_2 and G_3, we only compare $\mathcal{S}.CF$ to $\mathcal{S}^1.CF$ and ignore $\mathcal{M}^1.CF$.)

Lemma 17 (The gap between G_2 and G_3). *If G_3 is efficient, then G_2 is efficient and the probability of $\mathcal{D}$ outputting 1 in G_3 equals that in G_2.*

Proof. To understand the gap between the transcripts of $\mathcal{D}$ in G_2 and G_3, it is sufficient to compare the tables $\mathcal{S}.CF$ and $\mathcal{S}^1.CF$ since $\mathcal{D}$ queries $\mathcal{S}$ ($\mathcal{S}^1$) in G_2 (G_3) for CF values.

First, we compare the evaluation procedures $\mathcal{S}.CF^{\text{Inner}}$ and $\mathcal{S}^1.CF^{\text{Inner}}$ between the two games. The evaluation procedure $\mathcal{S}.CF^{\text{Inner}}$ in G_2 always selects CF uniformly. A quick check reveals that $\mathcal{S}^1.CF^{\text{Inner}}$ also evaluates CF uniformly because all the values it takes from table $\mathcal{M}^1.CF$ were selected by $\mathcal{M}^1$ uniformly. Then, the lemma follows by the fact that the procedure $\mathcal{S}.\text{Complete}$ has the same abort condition with $\mathcal{S}^1.\text{Complete}$ with respect to the status of table $\mathcal{S}.CF$ and $\mathcal{S}^1.CF$. □

The Gap Between the Third and the Fourth Game To analyze the relationship between G_3 and G_4, we define the following two bad events in G_4.

The first bad event happens when G_4 aborts during the execution of the completion procedure $\mathcal{S}^2.\text{Complete}$ or $\mathcal{M}^2.\text{Complete}$, which means $\mathcal{S}^2$ or $\mathcal{M}^2$ fails to complete a chain. Remember that the simulators fail to complete a chain when some terms on the subverted chain they are evaluating have been evaluated before or are dishonest.

The second bad event happens when G_4 aborts during the execution of the evaluation procedure $\mathcal{S}^2.C\tilde{F}^{\text{Inner}}$ or $\mathcal{S}^2.C\tilde{F}^{\text{Inner}}$, which means when $\mathcal{S}^2$ or $\mathcal{M}^2$ wants to set a uniform CF value to a certain term (i, x), $CF(i, x)$ is a middle point of $\mathcal{M}^2$, and it has not been put in the table $\mathcal{S}^2.CF$.

The formal names and definitions of the bad events are:

$$\text{BadComplete}_4 = \begin{cases} G_4 \text{ aborts during the execution of the procedure} \\ \mathcal{S}^2.\text{Complete or } \mathcal{M}^2.\text{Complete} \end{cases},$$

$$\text{BadEval}_4 = \begin{cases} G_4 \text{ aborts during the execution of the procedure } \mathcal{S}^2.C\tilde{F}^{\text{Inner}} \\ \text{or } \mathcal{M}^2.C\tilde{F}^{\text{Inner}} \end{cases}.$$

Remark. Notice that the completion procedure $\mathcal{S}^2$.Complete ($\mathcal{M}^2$.Complete) sometimes calls the evaluation procedure $\mathcal{S}^2.\widetilde{CF}^{\mathrm{Inner}}$ ($\mathcal{M}^2.\widetilde{CF}^{\mathrm{Inner}}$), so the completion procedure may abort because its Inner evaluation procedure aborts. We stress that we categorize this event as $\mathsf{BadEval}_4$.

It is clear that a bad event ($\mathsf{BadComplete}_4$ or $\mathsf{BadEval}_4$) happens if and only if the game G_4 aborts.

Lemma 18 (The gap between G_3 and G_4). *The probability of $\mathcal{D}$ outputting 1 in G_4 differs by at most $\Pr[\mathsf{BadComplete}_4] + \Pr[\mathsf{BadEval}_4]$ from that in G_3. Moreover, if G_4 is efficient and $\Pr[\mathsf{BadComplete}_4] + \Pr[\mathsf{BadEval}_4]$ is negligible, G_3 is efficient.*

Proof. From the descriptions of G_3 and G_4, we can see that there are two differences between the two games:

– In G_3, when $\mathcal{S}^1$ completes a chain that has been completed by $\mathcal{M}^1$ before and $u = 4n$, it completes the chain as usual (as if it is not completed by $\mathcal{M}^1$). However, in the same situation, $\mathcal{S}^2$ will just copy Q_c to $\mathcal{S}^2.CF$, where $c = (1, x_1, \ldots, x_{8n})$ is the full subverted chain completed by $\mathcal{M}^2$.
– G_3 does not abort in several cases where G_4 aborts.

We want to show gap between the two games caused the above two differences are bounded by the probability that G_4 aborts.

– The first difference causes zero gap between the two games unless G_4 aborts. Imagine that G_3 and G_4 have the same status (assuming the status of G_4 is not "Abort") at some point and $\mathcal{S}^1$.Complete ($\mathcal{S}^2$.Complete) is going to complete a chain that has been completed by $\mathcal{M}^1$ ($\mathcal{M}^2$). In this case, $\mathcal{S}^2$.Complete copies Q_c to $\mathcal{S}^2.CF$. We want to show that what $\mathcal{S}^1$.Complete does is equivalent to copying Q_c to $\mathcal{S}^1.CF$. For any $(i, x) \in Q_c$ ($(i, x) \neq (4n, x_{4n})$ or $(4n+1, x_{4n+1})$), if $\mathcal{S}^1$ assigns a different $CF_i(x)$ in $\mathcal{S}^1.CF$ than in $\mathcal{M}^1.CF$, then by definition of $\mathcal{S}^1.CF^{\mathrm{Inner}}$, (i, x) is an adapted point that is not in $\mathcal{S}^1.CF$, which violates the assumption that G_4 does not abort. For $(i, x) = (4n, x_{4n})$ or $(4n + 1, x_{4n+1})$, the values of $CF_i(x)$ are equal in $\mathcal{S}^1.CF$ and $\mathcal{M}^1.CF$ by programming rules.
– The second difference also causes zero gap between the two games unless G_4 aborts, which is obviously true.

As a result, the total variation distance between the transcripts of the distinguisher in G_3 and G_4 is smaller than $\Pr[\mathsf{BadComplete}_4] + \Pr[\mathsf{BadEval}_4]$, the probability that G_4 aborts.

To show the second claim, we observe that if G_4 is efficient and aborts with negligible probability, G_3 will be efficient since the total variation distance between G_3 and G_4 is negligible. □

B.5 The Fifth Game

To obtain G_5 from G_4, we update the simulator O^2 to O^3 which has two parties $\mathcal{M}^3$ and $\mathcal{S}^3$.

O^3 has a significantly different structure than the simulators in previous games. In the previous games, when the simulators complete a chain, they query RF first and then adapt two particular terms ((u, x_u) and $(x_{u+1}, u+1)$) on the chain to ensure consistency. In contrast, in G_5 we imagine the ideal object RF as simulated by $\mathcal{M}^3$ and $\mathcal{S}^3$. During the execution of the completion procedure, $\mathcal{M}^3$ and $\mathcal{S}^3$ set all the CF values of the target chain uniformly and then program the RF value to ensure consistency.

The evaluation procedures. The evaluation procedures $\mathcal{M}^3.CF^{\text{Inner}}$ and $\mathcal{S}^3.CF^{\text{Inner}}$ in G_5 are the same as their counterparts in G_4.

Handling $\mathcal{D}$'s query to the ideal object. When the distinguisher makes a query to RF on a $2n$-bit string (x_0, x_1), if $\mathcal{S}^3$ or $\mathcal{M}^3$ has evaluated $RF(x_0, x_1)$ or $RF^{-1}(x_{8n}, x_{8n+1})$ (for some (x_{8n}, x_{8n+1}) with $RF^{-1}(x_{8n}, x_{8n+1}) = (x_0, x_1)$) before, O^3 answers correspondingly. Otherwise, $\mathcal{M}^3$ proceeds to *complete* a chain in $\mathcal{M}^3.CF$. In more detail: Assuming that the distinguisher queries $RF(x_0, x_1)$, the "completion procedure" $\mathcal{M}^3$.Complete proceeds as follows. (Similar to the situation in $\mathcal{M}^2$.Complete, all the CF evaluations made by $\mathcal{M}^3$.Complete are in the table $\mathcal{M}^3.CF$.)

- $\tilde{CF}_1(x_1)$ is determined by simulating the subversion algorithm. This will generate oracle queries to $CF_*()$ which are answered using $\mathcal{M}^3.CF$. The value $x_2 = x_0 \oplus \tilde{CF}_1(x_1)$ is then determined, yielding the chain $(1, x_1, x_2)$. This process is repeated until the chain is extended maximally "to the right" so that it has the form $(x_1, x_2, \ldots, x_{8n-1}, x_{8n})$. Set $x_{8n+1} := x_{8n-1} \oplus \tilde{CF}_{8n}(x_{8n})$.
- Set $RF(x_0, x_1) := (x_{8n}, x_{8n+1})$ and return (x_{8n}, x_{8n+1}) to $\mathcal{D}$.
- The game aborts if for any $i = u$ or $u+1$, (i, x_i) was in $\mathcal{M}^3.CF$ before this execution of $\mathcal{M}^3$.Complete, $(i, x_i) \in \bigcup_{j=1}^{8n} Q_j(x_j)/Q_i(x_i)$, or (i, x_i) is dishonest.
- A set $\mathcal{M}^3$.CompletedChains is maintained to store the chains that are completed: for any (i, x_i, x_{i+1}) ($1 \le i \le 8n-1$), $\mathcal{M}^3$ updates $\mathcal{M}^3$.CompletedChains := $\mathcal{M}^3$.CompletedChains $\cup (i, x_i, x_{i+1})$.
- A set $\mathcal{M}^3$.MiddlePoints is maintained to store the points with index between $3n$ and $5n$: for any (i, x_i) with $3n \le i \le 5n$, $\mathcal{M}^3$ updates $\mathcal{M}^3$.MiddlePoints := $\mathcal{M}^3$.MiddlePoints $\cup (i, x_i)$.
- A set $\mathcal{M}^3$.AdaptedPoints is maintained to store the points with index $4n$ and $4n+1$: for (i, x_i) with $i = 4n$ or $4n+1$, $\mathcal{M}^3$ updates $\mathcal{M}^3$.AdaptedPoints := $\mathcal{M}^3$.AdaptedPoints $\cup (i, x_i)$.

The completion procedure $\mathcal{S}^3$.Complete. For input (s, x_s, x_{s+1}, u), the completion procedure $\mathcal{S}^3$.Complete (assuming $(s, x_s, x_{s+1}) \notin \mathcal{M}^3$.CompletedChains) proceeds as follows.

- The initial chain consists of the two adjacent values x_s, x_{s+1}.
- $\tilde{CF}_{s+1}(x_{s+1})$ is determined by simulating the subversion algorithm. This will generate oracle queries to $CF_*()$ which are answered using $\mathcal{S}^3.CF$. (Note that this process may enqueue new chains for completion.) The value $x_{s+2} = x_s \oplus \tilde{CF}_{s+1}(x_{s+1})$ is then determined, yielding the enlarged chain (x_s, x_{s+1}, x_{s+2}). This process is repeated until the chain is extended maximally "to the right" so that it has the form $(x_s, x_{s+1}, \ldots, x_{8n}, x_{8n+1})$.

- Similarly, the original pair x_s, x_{s-1} is extended "to the left" to determine the values $x_{s-1}, ..., x_0$.
- Set $RF(x_0, x_1) = (x_{8n}, x_{8n+1})$.
- The game aborts if for either $i = u$ or $u + 1$, (i, x_i) was in $\mathcal{S}^3.CF$ or $\mathcal{M}^3.CF$ before this execution of $\mathcal{S}^3$.Complete, $(i, x_i) \in \bigcup_{j=1}^{8n} Q_j(x_j)/Q_i(x_i)$, or (i, x_i) is dishonest.
- A set $\mathcal{S}^3$.CompletedChains is maintained to store the chains that are completed: for any (i, x_i, x_{i+1}) ($1 \leq i \leq 8n - 1$), the simulator updates $\mathcal{S}^3$.CompletedChains := $\mathcal{S}$.CompletedChains $\cup (i, x_i, x_{i+1})$.

The case when $(s, x_s, x_{s+1}) \in \mathcal{M}^3$.CompletedChains is taken care of in the same way as in G_4.

See page 90 for the pseudocode description of G_5.

The Gap Between the Fourth and the Fifth Game We wish to show that the distinguisher can not distinguish between G_4 and G_5. To prove this claim, we note that these two games behave differently only when they complete chains: when completing a chain, G_4 queries the ideal object RF and adapts the CF values of (x_u, x_{u+1}) to ensure consistency; in contrast, G_5 assigns CF of x_u and x_{u+1} uniformly, and then programs RF. In the following lemma, we are going to show that these two different conventions of completing a chain will yield the same distribution of the status of the game.

Lemma 19 (The gap between G_4 and G_5). *If G_5 is efficient, then G_4 is efficient and the probability of $\mathcal{D}$ outputting 1 in G_5 equals that in G_4.*

Proof. It is sufficient to show that, for any $0 < k \leq q_{\mathcal{D}}$, the total variation distance between the status of G_4 and G_5 is 0 at the end of k-th round of interaction between the distinguisher and the simulators.

Consider a proof by induction. Denote by α_4^s and α_5^s the distribution of the status of G_4 and G_5 at the end of the s-th interaction. For any $0 < k \leq q_{\mathcal{D}}$, assume $\alpha_4^s = \alpha_5^s$ when $s = k - 1$. We will show

$$\|\alpha_4^k - \alpha_5^k\|_{\text{tv}} = 0.$$

- If G_4 (G_5) already aborts before the end of the $k - 1$-th round of the game, then G_4 (G_5) also aborts at the end of the k-th round and $\|\alpha_4^t - \alpha_5^t\|_{\text{tv}} = 0$.
- If G_4 (G_5) does not abort before the end of the $k-1$-th round, and the k-th query made by $\mathcal{D}$ does not activate the procedures $\mathcal{S}^2$.Complete or $\mathcal{M}^2$.Complete ($\mathcal{S}^3$.Complete or $\mathcal{M}^3$.Complete), it is obvious that $\|\alpha_4^k - \alpha_5^k\|_{\text{tv}} = 0$.
- If G_4 (G_5) does not abort before the end of the $k - 1$-th round, and the k-th query made by $\mathcal{D}$ activates the procedures $\mathcal{S}^2$.Complete or $\mathcal{M}^2$.Complete ($\mathcal{S}^3$.Complete or $\mathcal{M}^3$.Complete), the situation is more complicated. We will deal with this case in the rest of the proof.

To show $\|\alpha_4^k - \alpha_5^k\|_{\text{tv}} = 0$ in the last case, it suffices to prove that if the "initial statuses" before the execution of the completion procedures are identical in G_4 and G_5, the "resulting statuses" after the execution are identically distributed. Formally speaking, assume that at some moment of G_4 and G_5, the two games have the same status and begin

49

to execute procedure $\mathcal{M}^2$.Complete and $\mathcal{M}^3$.Complete. (Without loss of generality, we assume this execution is activated by a query from $\mathcal{D}$ to $RF(x_0, x_1)$ for some bitstring pair (x_0, x_1).) We will show the distribution of α_4 and α_5 at the end of the execution are identical. The proof of the two games executing $\mathcal{S}^2$.Complete and $\mathcal{S}^3$.Complete are omitted because it is the same as the proof for $\mathcal{M}^2$.Complete and $\mathcal{M}^3$.Complete.

To prove the above claim, we introduce the following transition of the four different completion procedures. $\mathcal{M}^2$.Complete and $\mathcal{M}^3$.Complete are the first and the last procedure. Two middle procedures are used to build their connections. We assume the four procedures share a common initial table of the simulator, T_{initial}. (T_{initial} is $\mathcal{M}^2.CF$ in G_4 and $\mathcal{M}^3.CF$ in G_5.) Our goal is to prove that the distributions of the resulting table, T_{final}, are identical among the four procedures. Roughly speaking, the job of a completion procedure is to generate a sequence of terms (i, x_i) (for $i = 1, \ldots, 8n$) that are supposed to form a full subverted chain, a pair of two bitstrings $((x_0, x_1), (x_{8n}, x_{8n+1}))$ such that $RF(x_0, x_1) = (x_{8n}, x_{8n+1})$, and checks whether the game needs to abort in the process. Our proof will focus mainly on the distribution of these variables and when the game aborts.

- **Procedure 1**: Procedure 1 is $\mathcal{M}^2$.Complete.
    1. For $i = 2, \ldots, u$, generate (i, x_i) recursively by defining $x_i := x_{i-2} \oplus C\tilde{F}_{i-1}(x_{i-1})$ for $2 \le i \le u$ (each CF is evaluated uniformly).
    2. Query RF at (x_0, x_1) and receive a uniform pair of n-bit strings (x_{8n}, x_{8n+1}). To generate (i, x_i) $(i = u, \ldots, 8n)$, define $x_{i-2} := x_i \oplus C\tilde{F}_{i-1}(x_{i-1})$ for $u + 3 \le i \le 8n + 1$ (each CF is evaluated uniformly).
    3. Define $CF_u(x_u) := x_{u-1} \oplus x_{u+1}$ and $CF_{u+1}(x_{u+1}) := x_u \oplus x_{u+2}$. The game aborts if there is an index j such that $3n \le j \le 5n$ and (j, x_j) is in T_{initial} or $\bigcup_{i=1}^{8n} Q_i(x_i)/Q_j(x_j)$.
    4. Evaluate $C\tilde{F}_u(x_u)$ and $C\tilde{F}_{u+1}(x_{u+1})$; the game aborts if there is an index j such that $3n \le j \le 5n$ and (j, x_j) is dishonest.
- **Procedure 2**:
    1. Same as step 1 of Procedure 1.
    2. Select x_{u+1} and x_{u+2} uniformly. To generate (i, x_i) $(i = u + 3, \ldots, 8n + 1)$, define $(i, x_i) := x_{i-2} \oplus C\tilde{F}_{i-1}(x_{i-1})$ for $u + 3 \le i \le 8n + 1$ (each CF is evaluated uniformly). Assign $RF(x_0, x_1) = (x_{8n}, x_{8n+1})$.
    3. Same as step 3 of Procedure 1.
    4. Same as step 4 of Procedure 1.
- **Procedure 3**: Procedure 3 is a small pivot from Procedure 2.
    1. Same as step 1 of Procedure 2.
    2. Select $CF_u(x_u)$ uniformly and define $x_{u+1} := x_{u-1} \oplus CF_u(x_u)$. The game aborts if $CF_u(x_u)$ is previously assigned. Select $CF_{u+1}(x_{u+1})$ uniformly and define $x_{u+2} := x_u \oplus CF_{u+1}(x_{u+1})$. The game aborts if $CF_{u+1}(x_{u+1})$ is previously assigned. To generate the sequence (i, x_i) (for $i = u + 3, \ldots, 8n + 1$), define $x_i := x_{i-2} \oplus C\tilde{F}_{i-1}(x_{i-1})$ for $u+3 \le i \le 8n+1$ (each CF is evaluated uniformly). Assign $RF(x_0, x_1) = (x_{8n}, x_{8n+1})$.
    3. The game aborts if there is an index j such that $3n \le j \le 5n$ and (j, x_j) is in T_{initial} or $\bigcup_{i=1}^{8n} Q_i(x_i)/Q_j(x_j)$.
    4. Same as step 2 of Procedure 2.

– **Procedure 4**: Procedure 4 is $\mathcal{M}^3$.Complete.
  1. For $i = 2, \ldots, 8n + 1$, generate (i, x_i) recursively by defining $x_i := x_{i-2} \oplus C\tilde{F}_{i-1}(x_{i-1})$ for $2 \le i \le 8n + 1$ (each CF is evaluated uniformly). The game aborts if there is an index j such that $3n \le j \le 5n$ and (j, x_j) is in T_{initial} or $\bigcup_{i=1}^{8n} Q_i(x_i)/Q_j(x_j)$.
  2. The game aborts if there is an index j such that $3n \le j \le 5n$ and (j, x_j) is dishonest.
  3. Assign $RF(x_0, x_1) = (x_{8n}, x_{8n+1})$.

For $i = 1, 2, 3, 4$, We denote the distribution of T_{final} in Procedure i by α_{P_i}. We will prove $\|\alpha_{P_1} - \alpha_{P_4}\|_{\text{tv}} = 0$ by showing that $\|\alpha_{P_j} - \alpha_{P_{j+1}}\|_{\text{tv}} = 0$ for $j = 1, 2, 3$.

To see why $\|\alpha_{P_1} - \alpha_{P_2}\|_{\text{tv}} = 0$, we rewrite Procedure 1 and 2 as:

– **Procedure 1'**: Procedure 1' is a rewrite of Procedure 1.
  1. For all $x \in \{0, 1\}^n$ and $u + 2 \le i \le 8n$, evaluate $C\tilde{F}_i(x)$ (each CF is evaluated uniformly).
  2. Same as step 1 of Procedure 1.
  3. Select an uniform pair of n-bit strings (x_{8n}, x_{8n+1}) and assign $RF(x_0, x_1) = (x_{8n}, x_{8n+1})$. To generate (i, x_i) (for $i = u + 1, \ldots, 8n$), define $x_{i-2} := x_i \oplus C\tilde{F}_{i-1}(x_{i-1})$ for $u + 3 \le i \le 8n + 1$.
  4. Define $CF_u(x_u) := x_{u-1} \oplus x_{u+1}$ and $CF_{u+1}(x_{u+1}) := x_u \oplus x_{u+2}$. For all $x \in \{0, 1\}^n$ and $1 \le i \le 8n$, remove $CF_i(x)$ from the table if $CF_i(x)$ is not in T_{initial} and is not in $Q_j(x_j)$ for any $1 \le j \le 8n$ ($j \ne u, u + 1$). The game aborts if there is an index j such that $3n \le j \le 5n$ and (j, x_j) is in T_{initial} or $\bigcup_{i=1}^{8n} Q_i(x_i)/Q_j(x_j)$.
  5. Same as step 4 of Procedure 1.
– **Procedure 2'**: Procedure 2' is a rewrite of Procedure 2.
  1. Same as step 1 of Procedure 1'.
  2. Same as step 2 of Procedure 1'.
  3. Select x_{u+1} and x_{u+2} uniformly. To generate (i, x_i) (for $i = u+3, \ldots, 8n+1$), define $x_i := x_{i-2} \oplus C\tilde{F}_{i-1}(x_{i-1})$ for $u+3 \le i \le 8n+1$. Assign $RF(x_0, x_1) = (x_{8n}, x_{8n+1})$.
  4. Same as step 4 of Procedure 1'.
  5. Same as step 5 of Procedure 1'.

Notice that the step 3 of Procedure 1' is equivalent to that of Procedure 1' because the Feistel structure gives a permutation of $2n$-bit strings: selecting an uniform "input" string (x_{u+1}, x_{u+2}) is equivalent to selecting an uniform "output" string (x_{8n}, x_{8n+1}). Therefore, $\|\alpha_{P_1} - \alpha_{P_2}\|_{\text{tv}} = \|\alpha_{P_{1'}} - \alpha_{P_{2'}}\|_{\text{tv}} = 0$, where $\alpha_{P_{1'}}$ and $\alpha_{P_{2'}}$ are the distributions of T_{final} in Procedure 1' and 2'.

The fact that $\|\alpha_{P_2} - \alpha_{P_3}\|_{\text{tv}} = 0$ and $\|\alpha_{P_3} - \alpha_{P_4}\|_{\text{tv}} = 0$ are clear from the definitions of Procedures 2, 3, and 4. $\qquad\square$

Similar to G_4, we define the following two bad events in G_5:

$$\mathsf{BadComplete}_5 = \begin{cases} G_5 \text{ aborts during the execution of the procedure} \\ \mathcal{S}^3.\text{Complete or } \mathcal{M}^3.\text{Complete} \end{cases},$$

$$\mathsf{BadEval}_5 = \begin{cases} G_5 \text{ aborts during the execution of the procedure } \mathcal{S}^3.C\tilde{F}^{\text{Inner}} \\ \text{or } \mathcal{S}^3.C\tilde{F}^{\text{Inner}} \end{cases}.$$

Remark. Same as the bad events in G_4, if $\mathcal{S}^3$.Complete or $\mathcal{M}^3$.Complete aborts because its inner procedure $\mathcal{S}^3.C\tilde{F}^{\text{Inner}}$ or $\mathcal{S}^3.C\tilde{F}^{\text{Inner}}$ aborts, we say $\mathsf{BadEval}_5$ happens rather than $\mathsf{BadComplete}_5$ happens.

It is clear that a bad event ($\mathsf{BadComplete}_5$ or $\mathsf{BadEval}_5$) happens if and only if the game G_5 aborts.

The proof of Lemma 19 also implies:

Lemma 20. $\Pr[\mathsf{BadComplete}_4] = \Pr[\mathsf{BadComplete}_5], \text{ and } \Pr[\mathsf{BadEval}_4] = \Pr[\mathsf{BadEval}_5].$

Remark. Recall that in Section 3, we mentioned the following statement: Assuming G_5 is efficient, the probability that G_4 or G_5 aborts is negligible. Due to Lemma 20, the statement can be reduced to: assuming G_5 is efficient, the probability that G_5 aborts is negligible.

Lemma 21. *In G_5, $\mathcal{S}^3.CF$ is a subset of $\mathcal{M}^3.CF$ unless the game aborts.*

Proof. Clear from the definition. $\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\square$

Because of Lemma 21, we can treat unsubverted (subverted) chains in $\mathcal{S}^3.CF$ as unsubverted (subverted) chains in $\mathcal{M}^3.CF$.

The Gap Between the Fifth and the Sixth Game

Lemma 22 (The gap between G_5 and G_6). *The probability of $\mathcal{D}$ outputting 1 in G_5 differs that in G_6 by at most $\Pr[\mathsf{BadComplete}_5] + \Pr[\mathsf{BadEval}_5]$.*

Proof. The proof of the lemma is clear from the definition of G_5. $\qquad\qquad\qquad\square$

B.6 Preparations for Security and Efficiency Proof

In this section, we will make some technical preparations for the proof of the two remaining statements:

– Security: Assuming G_5 is efficient, the probability that G_5 aborts is negligible.
– Efficiency: G_5 is efficient. $\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\square$

We will focus on G_5 and its tables (especially $\mathcal{M}^3.CF$) in this section since both statements above are about G_5. The key property of G_5 we use to understand its distribution is that all the CF values in $\mathcal{M}^3.CF$ are selected uniformly and independently. Formally speaking, to understand the property of G_5 and its table $\mathcal{M}^3.CF$, we will consider the following simple probability model:

– Consider uniformly selecting a table CF_{Full} that contains CF values for all (i, x).
– In G_5, when the simulator proceeds to assign a new value to the table $\mathcal{M}^3.CF$, the simulator takes the value from CF_{Full} instead of selecting it uniformly as usual.

It is easy to see that the above model does not change the distribution of G_5 at all. Under this model, $\mathcal{M}.CF$ is an uniform table in the sense that, at any moment of G_5, conditioned on the current exposed (evaluated) terms in $\mathcal{M}.CF$, any unexposed term, if it is ever evaluated, will be evaluated uniformly.

We will work on two main results in this section. First, we introduce the notion of monotone increasing (and decreasing) chains, prove that each unsubverted chain can be viewed as a union of a decreasing chain and an increasing chain, and then show several nice properties of increasing (or decreasing) chains. Second, we prove all the dishonest terms on a subverted chain are located on an interval shorter than $n/6$.

The order function; monotone chains. To record the order in which $\mathcal{M}^3$ sets CF values, we define the following order function $O_{\mathcal{M}^3}$ from $\{1, \ldots, 8n\} \times \{0,1\}^n$ to positive integers (with an additional symbol $\perp$):

$$O_{\mathcal{M}^3}(i, x) = \begin{cases} t & \text{if } CF_i(x) \text{ is the } t\text{-th evaluated } CF \text{ value by } \mathcal{M}^3, \\ \perp & \text{if } CF_i(x) \text{ is undefined in } \mathcal{M}^3.CF. \end{cases}$$

An unsubverted chain $(s, x_s, \ldots, x_{s+r})$ in $\mathcal{M}^3.CF$ is said to be *monotone increasing* (or *monotone decreasing*) if $O_{\mathcal{M}^3}(i, x_i) < O_{\mathcal{M}^3}(i+1, x_{i+1})$ for all $s \le i < s+r$ (or, likewise, $O_{\mathcal{M}^3}(j, x_j) > O_{\mathcal{M}^3}(j+1, x_{j+1})$ for all $s \le j < s+r$).

In the rest of the paper, without loss of generality, we focus our analytic efforts on increasing chains; the results related to increasing chains can be easily transitioned into those related to decreasing chains.

We first show that any unsubverted chain can be viewed as a union of a deceasing chain and an increasing chain.

Lemma 23. *If G_5 is efficient, then with overwhelming probability, any unsubverted chain $c = (s, x_s, \ldots, x_{s+r})$ in $\mathcal{M}^3.CF$ will satisfy one of the three conditions below:*

1. *c is increasing,*
2. *c is decreasing,*
3. *There exists an index $s < v < s+r$ such that $(s, x_s, \ldots, x_v)$ is decreasing and $(v, x_v, \ldots, x_{s+r})$ is increasing.*

Proof. It suffices to show that in $\mathcal{M}^3.CF$ there is no unsubverted length three chain $(s, x_s, x_{s+1}, x_{s+2})$ such that $CF_{s+1}(x_{s+1})$ is evaluated after both $CF_s(s_s)$ and $CF_{s+2}(x_{s+2})$ are evaluated. Suppose that throughout G_5, there are no more than $P (= \text{poly}(n))$ elements

in $\mathcal{M}^3.CF$. Then in G_5,

$$\Pr \begin{bmatrix} \text{There is a length 3 chain } (s, x_s, x_{s+1}, x_{s+2}) \text{ such that } O_{\mathcal{M}^3}(s+1, x_{s+1}) > \\ \max\{O_{\mathcal{M}^3}(s+2, x_{s+2}), O_{\mathcal{M}^3}(s, x_s).\} \end{bmatrix}$$

$$= \sum_{i=2}^{P} \Pr \begin{bmatrix} \text{There is a length 3 chain } (s, x_s, x_{s+1}, x_{s+2}) \text{ such that } O_{\mathcal{M}^3}(s+1, x_{s+1}) = \\ i > \max\{O_{\mathcal{M}^3}(s+2, x_{s+2}), O_{\mathcal{M}^3}(s, x_s)\}. \end{bmatrix}$$

$$= \sum_{i=2}^{P} \sum_{\substack{j,k<i \\ j \neq k}} \Pr \begin{bmatrix} \text{There is a length 3 chain } (s, x_s, x_{s+1}, x_{s+2}) \text{ such that } O_{\mathcal{M}^3}(s+1, x_{s+1}) = i, \\ O_{\mathcal{M}^3}(s+2, x_{s+2}) = j \text{ and } O_{\mathcal{M}^3}(s, x_s) = k. \end{bmatrix}$$

$$< \sum_{i=2}^{P} \sum_{\substack{j,k<i \\ j \neq k}} \frac{1}{2^n}$$

$$< \frac{P^3}{2^n}$$

$$= \mathsf{negl}(n),$$

where the first inequality is based on the fact that and $CF_{s+1}(x_{s+1})$ is selected uniformly and is independent of $CF_s(s_s)$ and $CF_{s+2}(x_{s+2})$. $\qquad\square$

Advantages of $\mathcal{M}^3$ over $\mathcal{S}^3$. Note that Lemma 23 does not work for the simulator $\mathcal{S}^3$ because not all terms in $\mathcal{S}^3.CF$ are selected independently. Classifying the unsubverted chains in $\mathcal{M}^3.CF$ as increasing and decreasing is extremely useful to analyze the property of the chains in G_5.

Next, we will use a sequence of lemmas to establish the following major theorem that describes the nice properties of increasing chains.

Theorem 8. *If G_5 is efficient, then with overwhelming probability, any unsubverted increasing chain $c = (s, x_s, \ldots, x_{s+r})$ $(r > 8)$ in $\mathcal{M}^3.CF$ will satisfy:*

1. *for any $0 < i < j$ and $8 < j \leq r$, $(s+j, x_{s+j}) \notin Q_{s+i}(x_{s+i})$;*
2. *for any $7 \leq i < j \leq r$, $(s+i, x_{s+i}) \notin Q_{s+j}(x_{s+j})$;*
3. *for any $7 < i \leq r$, $(s+i, x_{s+i})$ is honest if $C\tilde{F}_{s+i}(x_{s+i})$ is defined.*

Lemma 24. *In G_5, with overwhelming probability, there is not an unsubverted (or subverted) chain $c = (i, x_i, \ldots, x_j)$ and a length 10 unsubverted chain $c' = (s, y_s, \ldots, y_{s+9})$ in $\mathcal{M}^3.CF$ such that*

- *for all $(j, x) \in c$, $C\tilde{F}_j(x)$ is defined;*
- *c and c' are disjoint;*
- *for each $s \leq k \leq s+9$, $(k, y_k) \in Q_c$.*

Proof. Consider proving the following stronger statement: Imagine we fill the entire table $\mathcal{M}^3.CF$ by uniformly selecting all the F values and (a_i, b_i) $(i = 1, \ldots, 8n)$. We will prove that with overwhelming probability over the choice of F values and (a_i, b_i), there are not two chains c and c' that satisfy the properties in the lemma.

Let (x_{i+1}, x_{i+2}), (y_s, y_{s+1}) be two pairs of n-bit strings and (i, j, s) be three positive indices. We denote by c the length $(j - i)$ chain starting with $(i + 1, x_{i+1}, x_{i+2})$ (without loss of generality, we assume c is a subverted chain for convenience in the rest of the proof) and denote by c' the length 10 unsubverted chain starting with (s, y_s, y_{s+1}). We denote by x_v $(v = i + 1, \ldots, j)$ the elements of c and denote by y_k $(k = s, \ldots, s + 9)$ the elements of c'. It is important to note that while x_{i+1}, x_{i+2}, y_s and y_{s+1} are specific n-bit strings, the values of x_v and y_k are currently undetermined. We use x_v and y_k purely to represent the elements of c and c' respectively. The actual values they will take on will be determined by choice of F values and (a_i, b_i). We define the event:

$$E_{i,j,s}(x_{i+1}, x_{i+2}, y_s, y_{s+1}) := \{c \text{ and } c' \text{ are disjoint, and for each } s \le k \le s+9, (k, y_k) \in Q_c\} \ .$$

For $s \le t \le s + 9$, we also define:

$$E_{i,j,s}^t(x_{i+1}, x_{i+2}, y_s, y_{s+1}) := \{c \text{ and } c' \text{ are disjoint, and for each } s \le k \le t, (k, y_k) \in Q_c\} \ .$$

To analyze the probability of $E_{i,j,s}(x_1, x_2, y_s, y_{s+1})$ over the choice of F and (a_i, b_i) $(i = 1, \ldots, 8n)$, we consider selecting uniformly the values of $F_i(x)$ for all $i = 1, \ldots, 8n$ and $x \in \{0, 1\}^n$ and selecting uniformly $a_v \cdot x_v \oplus b_v$ for $v = i, \ldots, j$. Since the function $x_v \to a_i \cdot x_v \oplus b_i$ is pairwise independent, the values of $a_k \cdot y_k \oplus b_k (k = s, \ldots, s + 9)$ are uniformly random. (For convenience, in the following, we will write $E_{i,j,s}$ for $E_{i,j,s}(x_1, x_2, y_s, y_{s+1})$ and $E_{i,j,s}^t$ for $E_{i,j,s}^t(x_1, x_2, y_s, y_{s+1})$.) Over the randomness of $a_k \cdot y_k \oplus b_k$ $(k = s, \ldots, s + 9)$, we have

$$\Pr[E_{i,j,s}]$$
$$= \Pr[E_{i,j,s} \mid E_{i,j,s}^{s+1}] \cdot \Pr[E_{i,j,s}^{s+1}]$$
$$< \Pr[E_{i,j,s} \mid E_{i,j,s}^{s+1}]$$
$$\quad \cdot (\Pr[CF_s(y_s) \in \cup_{v=i}^j Q_v(x_v) \mid y_s \neq x_s] + \Pr[CF_{s+1}(y_{s+1}) \in \cup_{v=i}^j Q_v(x_v) \mid y_{s+1} \neq x_{s+1}])$$
$$< \Pr[E_{i,j,s} \mid E_{i,j,s}^{s+3}] \cdot \Pr[E_{i,j,s}^{s+3} \mid E_{i,j,s}^{s+1}] \cdot 2 \cdot (8n \cdot q_{\mathcal{A}}/2^n)$$
$$< \Pr[E_{i,j,s} \mid E_{i,j,s}^{s+5}] \cdot \Pr[E_{i,j,s}^{s+5} \mid E_{i,j,s}^{s+3}] \cdot (16n \cdot q_{\mathcal{A}}/2^n)^2$$
$$< \Pr[E_{i,j,s} \mid E_{i,j,s}^{s+7}] \cdot \Pr[E_{i,j,s}^{s+7} \mid E_{i,j,s}^{s+5}] \cdot (16n \cdot q_{\mathcal{A}}/2^n)^3$$
$$< \Pr[E_{i,j,s} \mid E_{i,j,s}^{s+7}] \cdot (16n \cdot q_{\mathcal{A}}/2^n)^4$$
$$< (16n \cdot q_{\mathcal{A}}/2^n)^5 \ .$$

The lemma is implied by taking the union bound over the choice of (x_1, x_2, y_s, y_{s+1}). $\quad \square$

A similar proof can be used to prove the following lemma:

Lemma 25. *With overwhelming probability over the choice of all the F values and (a_i, b_i) $(i = 1, \ldots, 8n)$, there are not a term (i, x_i) and a length 8 unsubverted chain $c = (s, y_s, \ldots, y_{s+7})$ in $\mathcal{M}^3.CF$ such that $(k, y_k) \in Q_i(x_i)$ for all $k = s, s+2, s+4, s+6$.*

Lemma 26. *If G_5 is efficient, then with overwhelming probability, for any unsubverted increasing chain $c = (s, x_s, \ldots, x_{s+r})$ in $\mathcal{M}^3.CF$, if $(s + 2t + 1, x_{s+2t+1}) \in Q_{s+2k}(x_{s+2k})$ (assuming $C\tilde{F}_{s+2k}(x_{s+2k})$ is defined) for some t, k with $0 < 2t + 1, 2k \le r$, then $(s + 2i, x_{s+2i}) \in Q_{s+2k}(x_{s+2k})$ for all $0 < i \le t$.*

Proof. We give a simple example to show the idea of the proof. Take $s = 1$, $r = 7$, $k = 2$ and $t = 3$ for example. We want to show that for any chain $c = (1, x_1, \ldots, x_8)$, if $(8, x_8) \in Q_5(x_5)$, then with overwhelming probability, $(1 + 2i, x_{1+2i}) \in Q_5(x_5)$ for $i = 1$.

Consider the following two ways of determining a length 8 unsubverted chain:

- **Procedure 1**:
    1. Pick an arbitrary moment in G_5 and abort the game. Denote the table $\mathcal{M}^3.CF$ at this moment by T_{initial}. Pick a length 2 increasing chain $(1, x_1, x_2)$ in T_{initial} such that it is not a subchain of a length 3 unsubverted chain.
    2. For $2 \le i \le 7$, select $CF_i(x_i)$ uniformly, set $x_{i+1} := CF_i(x_i) \oplus x_{i-1}$ and abort the procedure if $(i + 1, x_{i+1})$ is already in the table T_{initial}.
    3. Evaluate $C\tilde{F}_5(x_5)$.
- **Procedure 2**:
    1. Pick an arbitrary moment in G_5 and abort the game. Denote the table $\mathcal{M}^3.CF$ at this moment by T_{initial}. Pick a length 2 increasing chain $(1, x_1, x_2)$ in T_{initial} such that it is not a subchain of a length 3 unsubverted chain.
    2. Select $CF_2(x_2)$ uniformly and set $x_3 := a_2 \oplus x_1$.
    3. Select 4 uniform n-bit strings a_4, a_5, a_6 and a_7. Set $x_5 := a_4 \oplus x_3$, $x_7 := a_6 \oplus x_5$ and abort the procedure if either of them is in T_{initial}. Set $CF_5(x_5) := a_5$ and $CF_7(x_7) := a_7$.
    4. Evaluate $C\tilde{F}_5(x_5)$.
    5. Select $CF_3(x_3)$ uniformly (use the existing value if it has been evaluated), set $x_4 := CF_3(x_3) \oplus x_2$, $x_6 := a_5 \oplus x_4$, $x_8 := a_7 \oplus x_6$, and abort the procedure if any one of x_4, x_4 and x_8 is in T_{initial}.

A quick thought reveals that the above two procedures are equivalent in terms of the distribution of the chain and, furthermore, the probability they abort is negligible because of Lemma 23. We use the second procedure to analyze the distribution of the first one. In the second procedure, we can see that if $(3, x_3) \notin Q_5(x_5)$, then $CF_3(x_3)$ is still uniform conditioned on $Q_5(x_5)$, which implies that $x_8 = a_7 \oplus x_6 = a_7 \oplus a_5 \oplus x_4 = a_7 \oplus a_5 \oplus CF_3(x_3) \oplus x_2$ is uniform. Therefore, if $(3, x_3) \notin Q_5(x_5)$, $(8, x_8) \in Q_5(x_5)$ with negligible probability.

The full proof can be achieved by replacing the concrete numbers in the last example by more general parameters s, r, k and t and taking the union bound over the various values of these parameters. $\square$

Lemma 27. *If G_5 is efficient, then with overwhelming probability, for any unsubverted increasing chain $c = (s, x_s, \ldots, x_{s+r})$ in $\mathcal{M}^3.CF$ and any index i, j with $0 < i < j$ and $8 < j \le r$, $(s + j, x_{s+j}) \notin Q_{s+i}(x_{s+i})$ (if $C\tilde{F}_{s+i}(x_{s+i})$ is defined).*

Proof. Without loss of generality, assume $i = 0$. Suppose $(s + j, x_{s+j}) \in Q_s(x_s)$. Notice that $(s + j - 1, x_{s+j-1}) \in Q_{s+i}(x_{s+i})$ with overwhelming probability because otherwise the randomness of $CF_{s+j-1}(x_{s+j-1})$ will cause the event $(s + j, x_{s+j}) \notin Q_s(x_s)$. Then,

- if j is odd, since $j > 8$ and $(s + j, x_{s+j}) \in Q_s(x_s)$, by Lemma 26, $(s + 2k, x_{s+2k}) \in Q_s(x_s)$ for $k = 1, 2, 3, 4$. This contradicts Lemma 25.
- if j is even, since $j > 8$ and $(s + j - 1, x_{s+j-1}) \in Q_s(x_s)$, by Lemma 26, $(s + 2j, x_{s+2j}) \in Q_s(x_s)$ for $j = 1, 2, 3, 4$, which contradicts with Lemma 25. $\square$

Lemma 28. *If G_5 is efficient, then with overwhelming probability, for any unsubverted increasing chain $c = (s, x_s, \ldots, x_{s+r})$ in $\mathcal{M}^3.CF$, if $(s + 2t, x_{s+2t}) \in Q_{s+k}(x_{s+k})$ (assuming $C\tilde{F}_{s+k}(x_{s+k})$ is defined) for some t, k with $0 < 2t < k \leq r$, then $(s + 2i - 1, x_{s+2i-1}) \in Q_{s+k}(x_{s+k})$ for all $0 < i \leq t$.*

Proof. The proof of the lemma is similar to that of Lemma 26. Consider the example where $s = 1, r = 8, t = 2$ and $k = 8$. We want to show that for any chain $c = (1, x_1, \ldots, x_9)$, if $(5, x_5) \in Q_9(x_9)$, then with overwhelming probability, $(2i, x_{2i}) \in Q_9(x_9)$ for $i = 1$.

Consider the following two ways of determining a length 9 unsubverted chain:

- **Procedure 1**:
    1. Pick an arbitrary moment in G_5 and abort the game. Denote the table $\mathcal{M}^3.CF$ at this moment by T_{initial}. Pick a length 2 increasing chain $(1, x_1, x_2)$ in T_{initial} such that it is not a subchain of a length 3 unsubverted chain.
    2. For $2 \leq i \leq 8$, select $CF_i(x_i)$ uniformly, set $x_{i+1} := CF_i(x_i) \oplus x_{i-1}$ and abort the procedure if $(i + 1, x_{i+1})$ is already in the table T_{initial}.
    3. Evaluate $C\tilde{F}_9(x_9)$.
- **Procedure 2**:
    1. Pick an arbitrary moment in G_5 and abort the game. Denote the table $\mathcal{M}^3.CF$ at this moment by T_{initial}. Pick a length 2 increasing chain $(1, x_1, x_2)$ in T_{initial} such that it is not a subchain of a length 3 unsubverted chain.
    2. Select 3 uniform n-bit strings a_3, a_4 and a_5. Set $x_4 := a_3 \oplus x_2$, $x_6 := a_5 \oplus x_4$ and aborts the procedure if either of them is in T_{initial}. Set $CF_4(x_4) := a_4$ and $CF_6(x_6) := a_6$.
    3. Select x_7, x_8 and x_9 uniformly and abort the procedure if any one of them is in T_{initial}. Set $CF_7(x_7) := x_6 \oplus x_8$ and $CF_8(x_8) := x_7 \oplus x_9$.
    4. Evaluate $C\tilde{F}_9(x_9)$.
    5. Select $CF_2(x_2)$ uniformly(use the existing value if it has been evaluated), set $x_3 := CF_2(x_2) \oplus x_1$, $x_5 := a_4 \oplus x_3$, $CF_6(x_6) := x_7 \oplus x_5$, and aborts the procedure if either x_3 or x_5 is in T_{initial}.

A quick thought reveals that the above two procedures are equivalent in terms of the distribution of the chain (and, furthermore, the probability they abort is negligible because of Lemma 23). We use the second procedure to analyze the distribution of the first one. In the second procedure, we can see that if $(2, x_2) \notin Q_9(x_9)$, then $CF_2(x_2)$ is still uniform conditioned on $Q_9(x_9)$, which implies that $x_5 = a_4 \oplus x_3 = a_4 \oplus a_2 \oplus x_1$ is uniform. Therefore, if $(2, x_2) \notin Q_9(x_9)$, $(5, x_5) \in Q_9(x_9)$ with negligible probability.

The formal proof can be achieved by replacing the concrete numbers in the last example by more general parameters s, r, t and k and taking the union bound over the various values of these parameters. □

Lemma 29. *If G_5 is efficient, then with overwhelming probability, for any unsubverted increasing chain $c = (s, x_s, \ldots, x_{s+r})$ in $\mathcal{M}^3.CF$ and any index i, j with $7 < i < j \leq r$, $(s + i, x_{s+i}) \notin Q_{s+j}(x_{s+j})$(if $C\tilde{F}_{s+j}(x_{s+j})$ is defined).*

Proof. The lemma is derived directly from Lemma 25 and Lemma 28. □

Lemma 30. *If G_5 is efficient, then with overwhelming probability, there does not exist an unsubverted increasing chain $c = (i, x_i, \ldots, x_{i+8})$ in $\mathcal{M}^3.CF$ such that $C\tilde{F}_{i+8}(x_{i+8})$ is defined in $\mathcal{M}^3.CF$ and $(i+8, x_{i+8})$ is dishonest.*

Proof. We say the distinguisher $\mathcal{D}$ wins G_5 if it is able to find an unsubverted increasing chain c in $\mathcal{M}^3.CF$ that satisfies the property in the lemma. By Lemma 27, the probability that there is a length-9 unsubverted increasing chain $c = (i, x_i, \ldots, x_{i+8})$ with $(i+7, x_{i+7}) \in Q_{i+8}(x_{i+8})$ ($C\tilde{F}_{i+8}(x_{i+8})$) is negligible. We denote this negligible probability by δ.

To show the probability that $\mathcal{D}$ wins is negligible, consider the following experiment with a distinguisher $\mathcal{D}^*$:

Exp*

1. $\mathcal{D}^*$ takes an arbitrary moment of G_5, stops the game and selects an arbitrary length-2 increasing chain (i, x_i, x_{i+1}) in $\mathcal{M}^3.CF$ such that $CF_{i+2}(x_{i+2})$ is not evaluated for $x_{i+1} := x_i \oplus CF_i(x_i)$.
2. Then, $\mathcal{D}^*$ extends (i, x_i, x_{i+1}) to $(i, x_i, \ldots, x_{i+7})$ by iteratively evaluating $CF_{j-1}(x_{j-1})$ (selected uniformly) and $x_j := x_{j-2} \oplus CF_{j-1}(x_{j-1})$ for $i+3 \leq j \leq i+7$. The experiment aborts if $CF_{i+7}(x_{i+7})$ is already evaluated.
3. For any term (j, y), if $CF_j(y)$ is still unevaluated and $(j, y) \neq (i+7, x_{i+7})$, $\mathcal{D}^*$ selects $CF_j(y)$ uniformly.
4. Finally $\mathcal{D}^*$ selects $CF_{i+7}(x_{i+7})$ and check if $(i+8, x_{i+8})$ is dishonest for $x_{i+8} := x_{i+6} \oplus CF_{i+7}(x_{i+7})$.
5. $\mathcal{D}^*$ wins **Exp*** if the experiment does not abort in Step 2 and $(i+8, x_{i+8})$ is dishonest.

It is easy to see that to prove $\mathcal{D}$ wins G_5 negligibly, it is sufficient to show the probability that the experiment aborts in Step 2 or $\mathcal{D}^*$ wins is negligible. We also stress that although **Exp*** is not G_5, the lemmas we proved in this section can still be applied because all the CF values here are also selected uniformly and independently.

$$\Pr_{\textbf{Exp*}}[\text{The experiment aborts in Step 2 or } \mathcal{D}^* \text{ wins.}]$$

$$\leq \Pr_{\textbf{Exp*}}[\text{The experiment aborts in Step 2.}] + \Pr_{\textbf{Exp*}}\left[\begin{array}{l}\mathcal{D}^* \text{ wins and there are at least } \sqrt{\delta}2^n \text{ } n\text{-bit strings } x \\ \text{such that } (i+7, x_{i+7}) \in Q_{i+8}(x).\end{array}\right]$$

$$+ \Pr_{\textbf{Exp*}}\left[\begin{array}{l}\mathcal{D}^* \text{ wins and there are fewer than } \sqrt{\delta}2^n \text{ } n\text{-bit strings} \\ x \text{ such that } (i+7, x_{i+7}) \in Q_{i+8}(x).\end{array}\right]$$

$$< \mathsf{negl}(n) + \Pr_{\textbf{Exp*}}\left[\begin{array}{l}\mathcal{D}^* \text{ wins and there are at least } \sqrt{\delta}2^n \text{ } n\text{-bit strings } x \text{ such that } (i+7, x_{i+7}) \in \\ Q_{i+8}(x).\end{array}\right]$$

$$+ \Pr_{\textbf{Exp*}}\left[\mathcal{D}^* \text{ wins.} \; \middle| \; \begin{array}{l}\text{There are fewer than } \sqrt{\delta}2^n \text{ } n\text{-bit strings } x \text{ such that} \\ (i+7, x_{i+7}) \in Q_{i+8}(x).\end{array}\right]$$

$$< \mathsf{negl}(n) + \sqrt{\delta} + \Pr_{\textbf{Exp*}}\left[\begin{array}{l}(i+8, x_{i+8}) \text{ is dishonest and} \\ (i+7, x_{i+7}) \in Q_{i+8}(x_{i+8}).\end{array} \middle| \begin{array}{l}\text{There are fewer than } \sqrt{\delta}2^n \text{ } n\text{-bit strings } x \\ \text{such that } (i+7, x_{i+7}) \in Q_{i+8}(x).\end{array}\right]$$

$$+ \Pr_{\textbf{Exp*}}\left[\begin{array}{l}(i+8, x_{i+8}) \text{ is dishonest and} \\ (i+7, x_{i+7}) \notin Q_{i+8}(x_{i+8}).\end{array} \middle| \begin{array}{l}\text{There are fewer than } \sqrt{\delta}2^n \text{ } n\text{-bit strings } x \\ \text{such that } (i+7, x_{i+7}) \in Q_{i+8}(x).\end{array}\right]$$

$$< \mathsf{negl}(n) + \sqrt{\delta} + \sqrt{\delta} + \epsilon$$

$$= \mathsf{negl}(n). \quad \square$$

Proof (Proof of Theorem 8). The theorem is the combination of Lemma 27, Lemma 29 and Lemma 30. $\qquad\square$

Definition 8 (Quasi-honest). *In G_5, for any $i = 9, \ldots, 8n$ and $x \in \{0, 1\}^n$, we say (i, x) is* quasi-honest *if there is an increasing chain $(s, x_s, \ldots, x_{s+r})$ $(r \geq 8)$ in $\mathcal{M}^3.CF$ such that $(s + r, x_{s+r}) = (i, x)$.*

Now we turn our attention to the dishonest terms on a subverted chain. We want to show that although, in general, there are some dishonest terms on a subverted chain, all of them gather in a small area.

Definition 9 (Bad region). *For a subverted chain $c = (s, x_s, \ldots, x_{s+r})$ in $\mathcal{M}^3.CF$, we say a subchain $(i, x_i, \ldots, x_j)$ $(s \leq i < j \leq s + r)$ of c is a* bad region *of c if there is no sequence of 14 consecutive elements $(k, x_k, \ldots, x_{k+13})$ $(i \leq k \leq j - 13)$ that are honest.*

For a subverted chain $c = (s, x_s, \ldots, x_{s+r})$ in $\mathcal{M}^3.CF$, we say two bad regions of c, $(i, x_i, \ldots, x_j)$ and $(i', x_{i'}, \ldots, x_{j'})$ $(i < i', j < j')$ are separated *if the subchain $(i, x_i, \ldots, x'_j)$ of c is not a bad region of c.*

Lemma 31. *In G_5, with overwhelming probability, there does not exist a subverted chain $(s, x_s, \ldots, x_{s+r})$ in $\mathcal{M}^3.CF$ such that it has a bad region with length greater than $n/6$.*

Proof. Consider proving the following stronger statement: with overwhelming probability over the uniform choice of (a_i, b_i) $(i = 1, \ldots, 8n)$ and values of $F_i(x)$ for all $i = 1, \ldots, 8n$ and $x \in \{0, 1\}^n$, there is no bad region with length greater than $n/6$. Imagine we select

$F_i(x)$ for all $i = \{1, \ldots, 8n\}$ and $x \in \{0, 1\}^n$ and leave a_i and b_i undetermined. Then, over the randomness of the choice of a_i and b_i, we have

$\Pr[\text{There is a subverted chain } c \text{ with a bad region longer than } n/6.]$

$$= \sum_{i=1}^{8n} \Pr \left[\begin{array}{l} \text{There is a subverted chain } c \text{ with a bad region longer than } n/6 \text{ and the} \\ \text{bad region begins at index } i. \end{array} \right]$$

$$= \sum_{i=1}^{8n} \sum_{x,x' \in \{0,1\}^n} \Pr \left[\begin{array}{l} \text{There is a subverted chain } c \text{ with a bad region longer than } n/6. \text{ The bad region begins} \\ \text{at index } i \text{ and its first two elements are } (i, x) \text{ and } (i + 1, x'). \end{array} \right]$$

$$< \sum_{i=1}^{8n} \sum_{x,x' \in \{0,1\}^n} \Pr \left[\begin{array}{l} \text{There is a subverted chain } c = (i, x, x', \ldots, x_r) \text{ such that its first element has index } i \text{ and} \\ r - i > n/6. \text{ Moreover, for any length 14 subchain of } c \text{ in the form of } (14k, x_{14k}, \ldots, x_{14k+13}), \\ \text{at least one of 14 elements is dishonest.} \end{array} \right]$$

$$< \sum_{i=1}^{8n} (2^n)^2 \cdot (14\epsilon)^{n/84-1}$$

$$= 8n \cdot 2^{2n} \cdot (14\epsilon)^{n/84-1}$$

$$= \mathsf{negl}(n). \quad \square$$

Lemma 32. *If G_5 is efficient, then with overwhelming probability, there is no subverted chain $c = (s, x_s, \ldots, x_{s+r})$ in $\mathcal{M}^3.CF$ that has two separated bad regions.*

Proof. The lemma is implied directly by Lemma 23 and Lemma 30. $\square$

B.7 Bounding Bad Events

Now we proceed to show the security: assuming G_5 is efficient, the probability that G_5 aborts is negligible.

There are two bad events that can cause G_5 to abort: $\mathsf{BadComplete}_5$ and $\mathsf{BadEval}_5$. For some technical reasons, we divide $\mathsf{BadComplete}_5$ into three smaller bad events:

– $\mathcal{M}.\mathsf{BadComplete}_5$: $\mathcal{M}.\mathsf{BadComplete}_5$ happens when G_5 aborts during the execution of $\mathcal{M}^3.\mathsf{Complete}$.
– $\mathcal{S}.\mathsf{NewBadComplete}_5$: $\mathcal{S}.\mathsf{NewBadComplete}_5$ happens when G_5 aborts during the execution of $\mathcal{S}^3.\mathsf{Complete}$ on a chain (i, x_i, x_{i+1}, u) and

$$(i, x_i, x_{i+1}) \notin \mathcal{M}^3.\mathsf{CompletedChains},$$

which means the chain (i, x_i, x_{i+1}) has not been completed by $\mathcal{M}^3$.
– $\mathcal{S}.\mathsf{ExistingBadComplete}_5$: $\mathcal{S}.\mathsf{ExistingBadComplete}_5$ happens when G_5 aborts during the execution of $\mathcal{S}^3.\mathsf{Complete}$ on a chain (i, x_i, x_{i+1}, u) and

$$(i, x_i, x_{i+1}) \in \mathcal{M}^3.\mathsf{CompletedChains},$$

which means $\mathcal{S}^3$ aborts when it is completing an existing chain in $\mathcal{M}^3.CF$. By definition of $\mathcal{S}^3.\mathsf{Complete}$, this bad event can happen only if $u = 7n$.

For any bad event A (e.g., $\mathcal{M}.\mathsf{BadComplete}_5$, $\mathsf{BadEval}_5$, etc.) and a positive integer $k \le q_{\mathcal{D}}$, we denote by $A[k]$ the event that bad event A causes G_5 to abort before the end of the k-th round of the interaction between $\mathcal{D}$ and the simulators. We also denote the table $\mathcal{M}^3.CF$ ($\mathcal{S}^3.CF$) at the end of the k-th interaction of G_5 by $\mathcal{M}^3.CF[k]$ ($\mathcal{S}^3.CF[k]$).

Definition 10 (Completed chains). *For any subverted chain $c = (i, x_i, \ldots, x_j)$ in $\mathcal{M}^3.CF$, we say it is $\mathcal{M}.Completed$ if $(i, x_i, x_{i+1}) \in \mathcal{M}^3.CompletedChains$. We denote the set of full $\mathcal{M}.Completed$ chains by $C_{\mathcal{M}.FComp}$.*

The concepts of $\mathcal{S}.Completed$ chains, $C_{\mathcal{S}.FComp}$ are defined similarly.

For any set E we mentioned above (e.g., $\mathcal{M}.CompletedChains$, $\mathcal{S}.CompletedChains$, $C_{\mathcal{M}.FComp}$, etc.) and a positive integer $k \le q_{\mathcal{D}}$, we denote by $E[k]$ the set E at the end of k-th round of the game.

To understand the probability of the bad events, we introduce the following property of the status of the game:

Definition 11 (Good Status). *For any $0 < k \le q_{\mathcal{D}}$, we say G_5 has a* good *status at round k (denoted by $\mathsf{GoodStatus}[k]$) if at the end of the k-th round of the game:*

– *The game does not abort with any bad events.*
– *For any increasing chain $(i, x_i, x_{i+1}, x_{i+2})$ in $\mathcal{M}^3.CF$, if $(i + 1, x_{i+1}) \notin \mathcal{S}^3.CF$, then*
  1. *$(i + 2, x_{i+2}) \notin \mathcal{S}^3.CF$, and*
  2. *for any $c \in C_{\mathcal{M}.FComp}[k]$ such that $(i + 2, x_{i+2}) \in Q_c$, $(i + 1, x_{i+1}) \in Q_c$.*
– *For any decreasing chain (x_i, x_{i+1}, x_{i+2}) in $\mathcal{M}^3.CF$, if $(i + 1, x_{i+1}) \notin \mathcal{S}^3.CF$, then*
  1. *$(i, x_i) \notin \mathcal{S}^3.CF$, and*
  2. *for any $c \in C_{\mathcal{M}.FComp}[k]$ such that $(i, x_i) \in Q_c$, $(i + 1, x_{i+1}) \in Q_c$.*

We will use an induction proof to show the following three theorems:

Theorem 9 ($\mathsf{BadComplete}_5$ is negligible.). *If G_5 is efficient and for some positive integer $T < q_{\mathcal{D}}$,*

$$\Pr[\mathsf{GoodStatus}[k] \text{ does not happen}] = \mathsf{negl}(n),$$

then

$$\Pr[\mathsf{BadComplete}_5[k + 1] \text{ happens}] = \mathsf{negl}(n).$$

Theorem 10 ($\mathsf{BadEval}_5$ is negligible.). *If G_5 is efficient and for some positive integer $k < q_{\mathcal{D}}$,*

$$\Pr[\mathsf{GoodStatus}[k] \text{ does not happen}] = \mathsf{negl}(n),$$

then

$$\Pr[\mathsf{BadEval}_5[k + 1] \text{ happens}] = \mathsf{negl}(n).$$

Theorem 11. *If G_5 is efficient and for some positive integer $k < q_{\mathcal{D}}$,*

$$\Pr[\mathsf{GoodStatus}[k] \text{ does not happen}] = \mathsf{negl}(n),$$

then

$$\Pr[\mathsf{GoodStatus}[k + 1] \text{ does not happen}] = \mathsf{negl}(n).$$
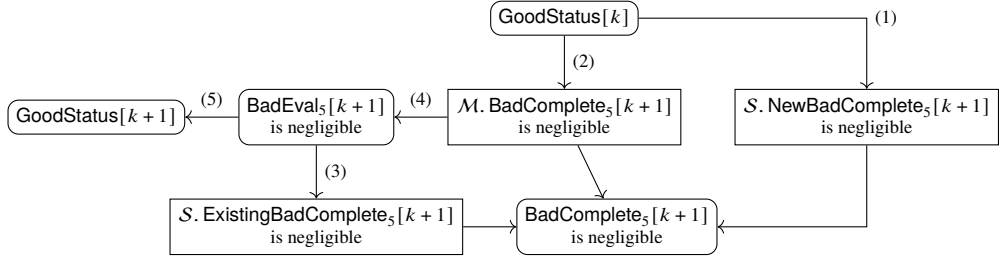
Fig. 7: A diagram showing the flow of the proof.

The structure of the proof is:

Each arrow in the diagram above corresponds to a lemma we use to show the three main theorems. In section B.7, we will prove Theorem 9 by showing lemma (1), (2), (3) are true and assuming lemma (4) is true. In section B.7 we will prove Theorem 10 by proving lemma (4). In section B.7, we will proving Theorem 11 by lemma (5).

Putting the three theorems together, we have:

Theorem 12. *If G_5 is efficient, the probability that it aborts because of bad events is negligible.*

Proof. The theorem is derived from Theorem 9, Theorem 10, Theorem 11 and the fact that $\mathsf{GoodStatus}[k]$ is true for $k = 1$. □

$\mathsf{BadComplete}_5[k+1]$ is negligible We first prove $\mathcal{S}.\,\mathsf{NewBadComplete}_5[k+1]$ is negligible when $\mathsf{GoodStatus}[k]$ does not happen. Consider the following experiment **Exp-LongGeneration**$[k]$ with a distinguisher $\mathcal{D}_{\mathrm{Long}}$:

EXP-LONGGENERATION$[k]$

For a positive integer $k < q_{\mathcal{D}}$, if G_5 does not abort by the end of the k-th round of the game, $\mathcal{D}_{\mathrm{Long}}$ takes the table $\mathcal{M}^3.CF[k]$ and renames it by T_1. As the experiment goes, $\mathcal{D}_{\mathrm{Long}}$ will add new CF values to T_1. All the newly added CF values will be evaluated uniformly.

1. Define C_{long} to be the set of unsubverted chains in T_1 with length $n/10 - 22$. Define the set $S := C_{\mathrm{long}} \times \{1, 2\}$.

---

2. $\mathcal{D}_{\text{Long}}$ uniformly selects three distinct elements in $s_1, s_2, s_3 \in S$. Run the following For-loop:

> **For** $i = 1, 2, 3$:
>
> **If** s_i is in the form of $(c, 1)$ for some $c = (r, x_r, \ldots, x_{r+n/10-23})$ **then**
>
>> Select an integer m uniformly from $(0, 1, 2, \ldots, 22)$.
>>
>> Extend the chain c to the right by m terms and to the left by $22 - m$ terms.
>>
>> Call the extended chain c_i. Evaluate $C\tilde{F}_j(x_j)$ for each $(j, x_j) \in c_i$.
>
> **If** s_i is in the form of $(c, 2)$ for some $c = (r, x_r, \ldots, x_{r+n/10-23})$ **then**
>
>> Evaluate $C\tilde{F}_j(x_j)$ for $j = r, \ldots, r + n/10 - 23$.
>>
>> **If** $C\tilde{F}_i(x_i) \neq CF_i(x_i)$ for some $r \leq i \leq r + n/10 - 23$ **then**
>>
>>> The experiment aborts.
>>
>> **Else**
>>
>>> Evaluate the full subverted chain that contains c as a subchain and name the full chain by c_i.

The new table after the For-loop is called T_2.

$\mathcal{D}_{\text{Long}}$ *wins* if there is an increasing chain $c' = (i, x_i, x_{i+1}, x_{i+2})$ in T_2 such that

- none of the elements of c' is in T_1;
- $(i + 2, x_{i+2}) \in Q_{c_3}$;
- $(i + 1, x_{i+1}) \notin Q_{c_3}$.

Lemma 33. *If G_5 is efficient,*

$$\Pr[\mathcal{D}_{Long} \text{ wins } \textbf{\textit{Exp-LongGeneration}}[k]] = \mathsf{negl}(n).$$

Proof. We first notice that, although **Exp-LongGeneration**$[k]$ is not G_5, the lemmas about the properties of monotone chains can still be applied since, in **Exp-LongGeneration**$[k]$, all the CF values are selected uniformly and independently.

Suppose **Exp-LongGeneration**$[k]$ fails with non-negligible probability. Consider the following realization of **Exp-LongGeneration**$[k]$: with non-negligible probability in **Exp-LongGeneration**$[k]$, we can select T_1, s_1, s_2 and s_3 so that the probability in the lemma is non-negligible; conditioned on T_1, s_1, s_2 and s_3, with non-negligible probability over the choice of T_2, we can find an increasing chain $c' = (i, x_i, x_{i+1}, x_{i+2})$ in T_2 that makes **Exp-LongGeneration**$[k]$ fail.

Consider the following realization of **Exp-LongGeneration**$[k]$ which have the same non-negligible probability with the above one. First we select the same T_1. Then we select the same three elements of S in the order s_3, s_1, s_2. Finally the same T_2 is selected. Now we want to find a contradiction by showing that (i, x_i, x_{i+1}) is increasing and $(i+1, x_{i+1}, x_{i+2})$ is decreasing, which violates Lemma 23. Notice that in the last realization of **Exp-LongGeneration**$[k]$, $c' = (i, x_i, x_{i+1}, x_{i+2})$ is increasing, $(i + 2, x_{i+2}) \in Q_{c_3}$ and $(i + 1, x_{i+1}) \notin Q_{c_3}$. It is clear that $CF_{i+1}(x_{i+1})$ is evaluated after $CF_{i+2}(x_{i+2})$. We want to show $CF_{i+1}(x_{i+1})$ is also evaluated after $CF_i(x_i)$.

- Case 1: if $(i, x_i) \in Q_{c_3}$, $CF_{i+1}(x_{i+1})$ is evaluated after $CF_i(x_i)$ since $(i+1, x_{i+1}) \notin Q_{c_3}$.
- Case 2: if $(i, x_i) \notin Q_{c_3}$, $CF_{i+1}(x_{i+1})$ is evaluated after $CF_i(x_i)$ since c' is increasing.

□

Lemma 34. *If G_5 is efficient and does not abort by the end of k-th round of the game for some positive integer $k < q_{\mathcal{D}}$, then with overwhelming probability, for any unsubverted chain $c = (s, x_s, \ldots, x_{s+n/10-1})$ that is processed by the procedure $\mathcal{S}^3.HonestyCheck$ in the $(k+1)$-th round of G_5, there is a length $(n/10 - 22)$ subchain $c' = (i, x_i, \ldots, x_{i+n/10-23})$ of c such that each element of c' is in $\mathcal{M}^3.CF[k]$.*

Proof. Suppose that all the chains that are processed by $\mathcal{S}^3.HonestyCheck$ before $c = (s, x_s, \ldots, x_{s+n/10-1})$ have the properties in the lemma (i.e., has a length $(n/10 - 22)$ subchain that is included in $\mathcal{M}^3.CF[k]$). We want to show that either c will not be processed by $\mathcal{S}^3.HonestyCheck$ or c has a length $(n/10 - 22)$ subchain in $\mathcal{M}^3.CF[k]$.

If c does not have a length $(n/10 - 22)$ subchain in $\mathcal{M}^3.CF[k]$, then, without loss of generality, we assume the chain $(s + n/10 - 11, x_{s+n/10-11}, \ldots, x_{s+n/10-1})$ is increasing and none of the elements in the chain is in $\mathcal{M}^3.CF[k]$.

Denote by t the moment when $\mathcal{S}^3$ is about to run the procedure $\mathcal{S}^3.Check$ on c. We will focus on the status of the game at moment t. Define S_1 to be the set of the chains that have been processed at $\mathcal{S}^3.HonestyCheck$ at t in the $k + 1$-th round of G_5. Define $S_2 := C_{\mathcal{S}.FComp} / C_{\mathcal{S}.FComp}[k]$, where $C_{\mathcal{S}.FComp}$ is the set of full $\mathcal{S}.Completed$ chains at t. Since $\mathcal{S}^3.HonestyCheck$ and $\mathcal{S}^3.Complete$ are the only two procedures that add new CF values, there exists an element $c^* \in S_1 \cup S_2$ such that $(s + n/10 - 1, x_{s+n/10-1}) \in Q_{c^*}$. By Lemma 33, $(i, x_i) \in Q_{c^*}$ for $i = s + n/10 - 10, \ldots, s + n/10 - 1$. Then, because of Lemma 24, c^* and $(s + n/10 - 10, x_{s+n/10-10}, \ldots, x_{s+n/10-1})$ are not disjoint. Finally, by definition of $\mathcal{S}^3.Check$, c will not be processed by $\mathcal{S}^3.HonestyCheck$. □

Next we prove a lemma that describes the property of $\mathcal{M}^3.CF[k]$.

Lemma 35. *If G_5 is efficient and $\mathsf{GoodStatus}[k]$ happens for some positive integer $k < q_{\mathcal{D}}$, then for any length $3n/10 + 30$ increasing chain $c = (s, x_s, \ldots, x_{s+3n/10+29})$ in $\mathcal{M}^3.CF[k]$, $(s + 3n/10 + 28, x_{s+3n/10+28}, x_{s+3n/10+29})$ is in $\mathcal{M}^3.CompletedChains[k]$ or $\mathcal{S}^3.CompletedChains[k]$.*

Proof. We analyze the following two cases:

- **Case 1:** $(s + 3n/10 + 10, x_{s+3n/10+10}) \in \mathcal{S}^3.CF[k]$.
  Because of $\mathsf{GoodStatus}[k]$, the chain $(s, x_s, \ldots, x_{s+3n/10+10})$ is in $\mathcal{S}^3.CF[k]$. Then suppose (j, x_j) $(s + n/10 + 11 \leq j \leq s + 3n/10 + 10)$ is the last element of $(s + n/10 + 11, x_{s+n/10+11}, \ldots, x_{s+3n/10+10})$ that is evaluated in $\mathcal{S}^3.CF[k]$.
  1. if $j \leq s + 2n/10 + 11$, then $c_1 = (j, x_j, \ldots, x_{j+n/10-1})$ is a length $n/10$ subchain of $(s + n/10 + 11, x_{s+n/10+11}, \ldots, x_{s+3n/10+10})$, which means c_1 was checked by the procedure $\mathcal{S}^3.Check$ at some moment before the end of round k of the game. Then, since any unsubverted chain that is not disjoint with c_1 consists of quasi-honest points, $(j, x_j, x_{j+1}) \in \mathcal{S}^3.CompletedChains[k]$, therefore

  $$(s + 3n/10 + 28, x_{s+3n/10+28}, x_{s+3n/10+29}) \in \mathcal{S}^3.CompletedChains[k] \,.$$

2. if $j \geq s + 2n/10 + 12$, similar analysis can be used to show that $(j-1, x_{j-1}, x_j) \in \mathcal{S}^3.\text{CompletedChains}[k]$ and

$$(s + 3n/10 + 28, x_{s+3n/10+28}, x_{s+3n/10+29}) \in \mathcal{S}^3.\text{CompletedChains}[k] \,.$$

– **Case 2:** $(s + 3n/10 + 10, x_{s+3n/10+10}) \notin \mathcal{S}^3.CF[k]$.
Because of $\text{GoodStatus}[k]$, none of the elements of the chain

$$(s + 3n/10 + 10, x_{s+3n/10+10}, \ldots, x_{s+3n/10+29})$$

is in $\mathcal{S}^3.CF[k]$. Suppose $(s+3n/10+29, x_{s+3n/10+29}) \in Q_{c'}$ for some $c' \in C_{\mathcal{M}.\text{FComp}}$. By $\text{GoodStatus}[k]$, all elements in the chain

$$c_2 = (s + 3n/10 + 19, x_{s+3n/10+19}, \ldots, x_{s+3n/10+29})$$

are in $Q_{c'}$. By Lemma 24, c_2 and c' are not disjoint. Since all the elements of c_2 are quasi-honest, $c_2 \subset c'$ and therefore,

$$(s + 3n/10 + 28, x_{s+3n/10+28}, x_{s+3n/10+29}) \in \mathcal{M}^3.\text{CompletedChains}[k] \,. \qquad \square$$

Lemma 36. *If G_5 is efficient and for some positive integer $k < q_{\mathcal{D}}$,*

$$\Pr[\text{GoodStatus}[k] \text{ does not happen}] = \text{negl}(n),$$

then

$$\Pr[\mathcal{S}.\text{NewBadComplete}_5[k + 1]] = \text{negl}(n).$$

Proof. Suppose at some moment during the $(k + 1)$-th interaction of G_5, the simulator $\mathcal{S}^3$ is starting to execute procedure $\mathcal{S}^3.\text{Complete}$ on a chain (s, x_s, x_{s+1}, u) such that $(s, x_s, x_{s+1}) \notin \mathcal{M}^3.\text{CompletedChains}$. We will show that, with overwhelming probability, $\mathcal{S}.\text{NewBadComplete}_5$ does not happen in this execution.

Denote by T_{initial} the table $\mathcal{M}^3.CF$ right before the execution of $\mathcal{S}^3.\text{Complete}$ on (s, x_s, x_{s+1}, u). Assume, without loss of generality, that $s < u$. (Then, by our convention of the simulator, $u - s > n$.) Imagine we run $\mathcal{S}^3.\text{Complete}$ on (s, x_s, x_{s+1}, u) and stop when the procedure just finished evaluating $\tilde{CF}_{s+9n/10+8}(x_{s+9n/10+8})$. We call the table $\mathcal{M}^3.CF$ at this moment T_{final}.

Consider the subverted chain $c = (s, x_s, \ldots, x_{s+9n/10+8})$ and its subchain $c' = (s + 9n/10, x_{s+9n/10}, \ldots, x_{s+9n/10+8})$. To prove $\mathcal{S}.\text{NewBadComplete}_5[k + 1]$ does not happen during the completion of (s, x_s, x_{s+1}, u), it is sufficient to show that c' has the following properties:

– c' is honest, and therefore can be viewed as an unsubverted chain;
– c' is increasing;
– None of the elements in c' is in T_{initial}.

Before we prove these properties, it is helpful to see how these properties imply that the bad event $\mathcal{S}.\text{NewBadComplete}_5$ does not happen.

- First, because of Lemma 23 and Theorem 8, the future subverted chain $(s + 9n/10, x_{s+9n/10}, \ldots, x_{8n})$ is honest and increasing (as an unsubverted chain). By Theorem 8 again, for $m = u, u+1$, (m, x_m) will not be queried by any $C\tilde{F}_i(x_i)$ with $s + 9n/10 + 8 < i \leq 8n$ and $i \neq m$.
- Second, imagine evaluating the subverted chain $c'' = (1, x_1, \ldots, x_{s+9n/10+8})$ (by uniformly assigning new CF values in need). We will show $(m, x_m) \notin Q_{c''}$ for $m = u, u+1$. Suppose $(m, x_m) \in Q_{c''}$. Since c' is increasing, $(s + 9n/10 + 9, x_{s+9n/10+9}, \ldots, x_m)$ is also increasing by Lemma 23. This implies all the elements of $(s + 9n/10 + 9, x_{s+9n/10+9}, \ldots, x_m)$ are in $Q_{c''}$, which is contradictory to Lemma 24. (Although the order we generate CF values here is not same as that in G_5, we still select CF values uniformly, which allows us to use the lemmas we showed about increasing chains.)
- Finally, since $(s + 9n/10, x_{s+9n/10}) \notin T_{\text{initial}}$, $(m-1, x_{m-1}) \notin T_{\text{initial}}$ for $m = u, u+1$. By the randomness of $CF_{m-1}(x_{m-1})$, (m, x_m) is not in T_{initial}.

We prove the above properties of c' in three steps.

- **Step 1:** We first show the chain $(s + n/2, x_{s+n/2}, \ldots, x_{s+9n/10+8})$ is honest and increasing as an unsubverted chain.
  - **Case 1:** if $(s, x_s, \ldots, x_{s+3n/10+39})$ is not honest, then by Lemma 31 and Lemma 32, the chain

    $$(s + 3n/10 + n/6 + 39, x_{s+3n/10+n/6+39}, \ldots, x_{s+9n/10+8})$$

    is honest. By Theorem 8 and Lemma 23, the chain

    $$(s + 3n/10 + n/6 + 47, x_{s+3n/10+n/6+47}, \ldots, x_{s+9n/10+8})$$

    is increasing.
  - **Case 2:** if $(s, x_s, \ldots, x_{s+3n/10+39})$ is honest, then by Lemma 23 and Lemma 35,

    $$(s + 3n/10 + 30, x_{s+3n/10+30}, \ldots, x_{s+3n/10+39})$$

    is increasing. By Theorem 8, the chain

    $$(s + 3n/10 + 39, x_{s+3n/10+30}, \ldots, x_{s+9n/10+8})$$

    is increasing and honest.
- **Step 2:** Next we show none of the elements in

  $$(s + 4n/5 + 30, x_{s+4n/5+30}, \ldots, x_{s+9n/10+8})$$

  is in $\mathcal{M}^3.CF[k]$.
  It suffices to show that $(s + 4n/5 + 30, x_{s+4n/5+30}) \notin \mathcal{M}^3.CF[k]$. Suppose $(s + 4n/5 + 30, x_{s+4n/5+30}) \in \mathcal{M}^3.CF[k]$, then by Lemma 35, $(s+4n/5+30, x_{s+4n/5+29}, x_{s+4n/5+30})$ is in $\mathcal{M}^3.CompletedChains[k]$ or $\mathcal{S}^3.CompletedChains[k]$. This contradicts the fact that $\mathcal{S}^3.Complete$ only processes uncompleted chains and our assumption that $(s, x_s, x_{s+1}) \notin \mathcal{M}^3.CompletedChains$.

- **Step 3:** Lastly, we show that no elements in $(s+4n/5+40, x_{s+4n/5+40}, \ldots, x_{s+9n/10+8})$ are in T_{initial}.

  It suffices to show that $(s + 4n/5 + 40, x_{s+4n/5+40}) \notin T_{\text{initial}}$. Suppose $(s + 4n/5 + 40, x_{s+4n/5+40}) \in T_{\text{initial}}$. Since $(s + 4n/5 + 30, x_{s+4n/5+30}) \notin \mathcal{M}^3.CF[k]$, the chain

$$(s + 4n/5 + 30, x_{s+4n/5+30}, \ldots, x_{s+4n/5+40})$$

  is in T_{initial}, which, by the proof of Lemma 34, means that

$$(s + 4n/5 + 30, x_{s+4n/5+30}, \ldots, x_{s+4n/5+40})$$

  is already a $\mathcal{S}$.Completed chain. This contradicts the fact that $\mathcal{S}^3$.Complete only processes uncompleted chains. $\qquad\square$

Lemma 37. *If G_5 is efficient and for some positive integer $k < q_{\mathcal{D}}$,*

$$\Pr[\mathsf{GoodStatus}[k] \ \textit{does not happen}] = \mathsf{negl}(n),$$

then

$$\Pr[\mathcal{M}.\,\mathsf{BadComplete}_5[k + 1]] = \mathsf{negl}(n).$$

Proof. The proof is similar to (and simpler than) that of Lemma 36. $\qquad\square$

Lemma 38. *If G_5 is efficient and for some positive integer $k < q_{\mathcal{D}}$,*

$$\Pr[\mathsf{BadEval}_5[k + 1]] = \mathsf{negl}(n),$$

then

$$\Pr[\mathcal{S}.\,\mathsf{ExistingBadComplete}_5[k + 1]] = \mathsf{negl}(n).$$

Proof. Suppose at some moment during the $(k + 1)$-th interaction of G_5, the simulator $\mathcal{S}^3$ is starting to execute procedure $\mathcal{S}^3$.Complete on a chain (s, x_s, x_{s+1}, u) such that $(s, x_s, x_{s+1}) \in \mathcal{M}^3$.CompletedChains.

Because of $\mathcal{S}^3$'s convention of choosing u and the fact that $\mathsf{BadEval}_5[k+1]$ is negligible, u is equal to $4n$ in this case, which avoids the bad event $\mathcal{S}.\,\mathsf{ExistingBadComplete}_5[k+1]$. $\qquad\square$

Proof (Proof of Theorem 9). The theorem is proved by combining Lemma 36, Lemma 37, Theorem 10 and Lemma 38. $\qquad\square$

$\mathsf{BadEval}_5[k + 1]$ is negligible We assume $\mathsf{GoodStatus}[k]$ throughout this section to prove $\mathsf{BadEval}_5[k + 1]$ is negligible. Consider the following experiment **Exp-Eval**$[k + 1]$ where an adaptive distinguisher $\mathcal{D}_1$ interacts with $(\mathcal{S}^3, \mathcal{M}^3)$ and tries to trigger $\mathsf{BadEval}_5[k + 1]$:

Exp-Eval$[k + 1]$

1. $\mathcal{D}_1$ makes a queries for some $0 \le a < k$ and outputs a.
2. At the $a + 1$-th round of the game, $\mathcal{D}_1$ queries $RF(x_0, x_1)$ for some pair of strings (x_0, x_1). (By definition, $\mathcal{M}^3$ will complete a subverted chain determined by (x_0, x_1). We call this full subverted chain c.)
3. $\mathcal{D}_1$ makes another b queries for $b = k - a$.

We say $\mathcal{D}_1$ wins **Exp-Eval**$[k+1]$ if c is not a $\mathcal{S}$.Completed chain by the end of the k-th round of the experiment and **Exp-Eval**$[k+1]$ aborts at the $(k+1)$-th round with the bad event $\mathsf{BadEval}_5[k+1]$: $\mathcal{S}^3.CF^{\mathrm{Inner}}$(or $\mathcal{M}^3.CF^{\mathrm{Inner}}$) calls a term (i,x) such that $3n \le i \le 5n$ and $(i,x) \in c$.

Here, without loss of generality, we do not consider the case where $\mathcal{D}_1$ makes a $RF^{-1}(\cdot)$ query in round $a+1$.

It is easy to see that Theorem 10 is equivalent to:

Theorem 13 ($\mathsf{BadEval}_5$ **is negligible.**). *If G_5 is efficient and for some positive integer $k < q_{\mathcal{D}}$,*

$$\Pr[\mathsf{GoodStatus}[k] \text{ does not happen}] = \mathsf{negl}(n),$$

for any distinguisher $\mathcal{D}$, then, for any distinguisher $\mathcal{D}_1$,

$$\Pr[\mathcal{D}_1 \text{ wins } \textbf{\textit{Exp-Eval}}[k+1]] = \mathsf{negl}(n).$$

We take two steps to prove the theorem. First, notice that the only information $\mathcal{D}_1$ has to predict $(i,x) \in c$ is the answer to the query $RF(x_0, x_1)$. We turn this observation into a lemma that says $\mathcal{D}_1$ can not win **Exp-Eval**$[k+1]$ without receiving the answer to the query $RF(x_0, x_1)$. Second, we prove that whether receiving the answer or not does not affect too much the probability that $\mathcal{D}_1$ wins.

Consider the following modified version of **Exp-Eval**$[k+1]$ where the distinguisher $\mathcal{D}^2$ does not receive the answer to $RF(x_0, x_1)$.

Exp-EvalNoAnswer$[k+1]$

1. $\mathcal{D}_2$ makes a queries for some $0 \le a < k$ and outputs a.
2. At the $a+1$-st round of the game, $\mathcal{D}_2$ queries $RF(x_0, x_1)$ for some pair of strings (x_0, x_1), but does not receive the answer. (By definition, $\mathcal{M}^3$ will complete a subverted chain determined by (x_0, x_1). We call this full subverted chain c.)
3. $\mathcal{D}_2$ makes another b queries for $b = k - a$.

The winning condition of **Exp-EvalNoAnswer**$[k+1]$ is same as that of **Exp-Eval**$[k+1]$.

To prove $\mathcal{D}_2$ wins **Exp-EvalNoAnswer**$[k+1]$ with negligible probability, we consider the following distinguisher $\mathcal{D}_3$, which uses $\mathcal{D}_2$ in **Exp-EvalNoAnswer**$[k+1]$ to trigger the bad event $\mathcal{M}.\mathsf{BadComplete}_5[k+1]$.

Exp-BadMComplete$[k+1]$

1. $\mathcal{D}_3$ uses $\mathcal{D}_2$ to make a queries for some $0 \le a < k$ and outputs a. For each round of the game, $\mathcal{D}_3$ receives the query from and gives the answer to $\mathcal{D}_2$.
2. $\mathcal{D}_3$ remembers the $a+1$-th query $RF(x_0, x_1)$ of $\mathcal{D}_2$ but does not make the query (x_0, x_1) to $RF(\cdot)$.
3. $\mathcal{D}_3$ uses $\mathcal{D}_2$ to make another b queries for $b = k - a$. Again, for each round of the game, $\mathcal{D}_3$ receives the query from and gives the answer to $\mathcal{D}_2$.

---

4. $\mathcal{D}_3$ makes its last query $RF(x_0, x_1)$.

We say $\mathcal{D}_3$ wins **Exp-BadMComplete**$[k + 1]$ if the subverted chain corresponding to (x_0, x_1) is not in $\mathcal{S}$.Completed at the end of the k-th round of the experiment and **Exp-BadMComplete**$[k + 1]$ aborts at the $(k + 1)$-th round with the following event: when $\mathcal{M}^3$.Complete is completing the subverted chain c determined by (x_0, x_1), there is a term $(i, x) \in c$ such that $3n \leq i \leq 5n$ and (i, x) is queried in the k-th round of the game.

Lemma 39. *If for some positive integer $k < q_{\mathcal{D}}$,*

$$\Pr[\mathcal{M}.\, \mathsf{BadComplete}_5[k + 1] \ happens] = \mathsf{negl}(n),$$

for any distinguisher $\mathcal{D}$, then, for any distinguisher $\mathcal{D}_2$,

$$\Pr[\mathcal{D}_2 \ wins \ \textbf{\textit{Exp-EvalNoAnswer}}[k + 1]] = \mathsf{negl}(n).$$

Proof. Conditioned on the same randomness of CF in **Exp-EvalNoAnswer**$[k + 1]$ and **Exp-BadMComplete**$[k + 1]$, we can see that if $\mathcal{D}_2$ wins **Exp-EvalNoAnswer**$[k + 1]$, $\mathcal{D}_3$ will win **Exp-BadMComplete**$[k + 1]$, which by assumption, is negligible. □

Now we proceed to show if $\mathcal{D}_2$ wins **Exp-EvalNoAnswer**$[k + 1]$ with negligible probability, $\mathcal{D}_1$ wins **Exp-Eval**$[k + 1]$ with negligible probability, too.

Choose an arbitrary integer m with $4n \leq m < 5n$. Consider the following variation of **Exp-Eval**$[k + 1]$ for $\mathcal{D}_1'$.

Exp-EvalRight$[k + 1, m]$

Select uniformly a full table of CF values CF_{full} and a $2n$-bit string α. In step 1 and 3 of the experiment, all the CF values queried by the simulators are taken from CF_{full} instead of being selected uniformly as usual. In step 2, CF values queried by the simulators are generated with a special convention explained below.

1. Same as step 1 of **Exp-Eval**$[k + 1]$.
2. Same as step 2 of **Exp-Eval**$[k + 1]$ and the answer to the query $RF(x_0, x_1)$ is α.
3. Same as step 3 of **Exp-Eval**$[k + 1]$.

In step 2, $\mathcal{M}^3$ uses the following convention to fill in its table:

1. Set $(x_{8n}, x_{8n+1}) := \alpha$.
2. For $i = 2, \ldots, m$, define $x_i := x_{i-2} \oplus C\tilde{F}_{i-1}(x_{i-1})$. For $i = m + 3, \ldots, 8n + 1$, define $x_{i-2} := x_i \oplus C\tilde{F}_{i-1}(x_{i-1})$. All the CF values are taken from CF_{full}.
3. Set $CF_m(x_m) := x_{m-1} \oplus x_{m+1}$ and $CF_{u+1}(x_{u+1}) := x_u \oplus x_{u+2}$. The game aborts if there is an index j such that $3n \leq j \leq 5n$ and (j, x_j) is in $\mathcal{M}^3.CF[a]$ or $\bigcup_{i=1}^{8n} Q_i(x_i)/Q_j(x_j)$.
4. Evaluate $C\tilde{F}_m(x_m)$ and $C\tilde{F}_{m+1}(x_{m+1})$ and the game aborts if there is an index j such that $3n \leq j \leq 5n$ and (j, x_j) is dishonest.

The winning condition of the experiment is same as that of **Exp-Eval**$[k + 1]$ except that in this experiment, $\mathcal{D}_1'$ wins if the index i of the term (i, x) causing the bad event has the range $3n \leq i \leq m$. For simplicity, we say $W_{\text{EvalRight}}(CF_{\text{full}}, \alpha, m) = 1$ if $\mathcal{D}_1'$ wins with the choice of $(CF_{\text{full}}, \alpha)$ in the experiment.

From the proof of Lemma 19, we can see the distribution of $\mathcal{M}^3.CF$ in **Exp-EvalRight**$[k + 1, m]$ is same as that of **Exp-Eval**$[k + 1]$. The only difference between the two experiments is that the new version has a more strict winning condition than the original one.

To prove the probability that $\mathcal{D}_1$ wins **Exp-Eval**$[k + 1]$ is negligible, we first show, for any distinguisher $\mathcal{D}_1'$, the probability that $\mathcal{D}_1'$ wins **Exp-EvalRight**$[k + 1, m]$ is negligible. Consider the following rewrite of **Exp-EvalNoAnswer**$[k + 1]$, where the randomness of CF is set like **Exp-EvalRight**$[k + 1, m]$ and $\mathcal{D}_2$ uses $\mathcal{D}_1'$ as an oracle to play the game.

EXP-EVALNOANSWER$[k + 1]$

Select uniformly a full table of CF values CF_{full} and a pair of $2n$-bit strings α and β. The randomness of CF and RF are set the same way as they are set in **Exp-EvalRight**$[k + 1, m]$.

1. $\mathcal{D}_2$ uses $\mathcal{D}_1'$ to make a queries for some $0 \leq a < k$ and outputs a. For each round of the game, $\mathcal{D}_2$ receives the query from and gives the answer to $\mathcal{D}_1'$.
2. At the $a + 1$-st round of the game, $\mathcal{D}_2$ uses $\mathcal{D}_1'$ to query $RF(x_0, x_1)$ for some pair of strings (x_0, x_1). $\alpha := RF(x_0, x_1)$ is evaluated by $\mathcal{M}^3$ but not returned to $\mathcal{D}_2$. (By definition, $\mathcal{M}^3$ will complete a subverted chain determined by (x_0, x_1). We call this full subverted chain c.) $\mathcal{D}_2$ selects β uniformly and gives it to $\mathcal{D}_1'$.
3. $\mathcal{D}_2$ uses $\mathcal{D}_1'$ to make b queries ($b = k - a$). For each round of the game, $\mathcal{D}_2$ receives the query from and gives the answer to $\mathcal{D}_1'$.

For simplicity, we say $W_{\text{EvalNA}}(CF_{\text{full}}, \alpha, \beta) = 1$ if $\mathcal{D}_2$ wins for the choice of $(CF_{\text{full}}, \alpha, \beta)$.

Lemma 40. *If for some positive integer $k < q_{\mathcal{D}}$,*

$$\Pr[\mathcal{D}_2 \text{ wins } \textbf{Exp-EvalNoAnswer}[k + 1]] = \mathsf{negl}(n)$$

over the randomness of $(CF_{\text{full}}, \alpha, \beta)$, then, for any integer $4n \leq m < 5n$,

$$\Pr[\mathcal{D}_1' \text{ wins } \textbf{Exp-EvalRight}[k + 1, m]] = \mathsf{negl}(n)$$

over the randomness of $(CF_{\text{full}}, \alpha)$.

Proof. Suppose that the first probability is negligible and the second is not. Then, with non-negligible probability, uniformly selecting a table CF_{full} and two $2n$-bit strings α, β yields:

- $W_{\text{EvalRight}}(CF_{\text{full}}, \beta) = 1$;

- $W_{\text{EvalNA}}(CF_{\text{full}}, \alpha, \beta) \neq 1$.

For convenience, we call the two experiments above E_1 and E_2. Suppose the full subverted chain generated in the $(a+1)$-th round of E_1 is $c_1 = (1, x_1, \ldots, x_m, y_{m+1}, \ldots, y_{8n})$, and the chain in E_2 is $c_2 = (1, x_1, \ldots, x_m, z_{m+1}, \ldots, z_{8n})$. Notice that the first m terms of c_1 and c_2 are same because the two experiments share the same table CF_{full}.

We will find a contradiction by showing that the queries to CF (made by the distinguisher or the simulators) are identical in E_1 and E_2 after the $(a+1)$-th round of the games. It is sufficient to show that E_1 or E_2 does not query (m, x_m), $(m+1, y_{m+1})$ and $(m+1, z_{m+1})$, the only three terms that have different CF values in E_1, E_2 and CF_{full}. Since $W_{\text{EvalNA}}(CF_{\text{full}}, \alpha, \beta) \neq 1$, E_2 does not query (m, x_m) or $(m+1, z_{m+1})$. Since $W_{\text{EvalRight}}(CF_{\text{full}}, \beta) = 1$, E_1 does not query $(m+1, y_{m+1})$.

Since $W_{\text{EvalRight}}(CF_{\text{full}}, \beta) = 1$ and E_1, E_2 make same queries to CF after the $(a+1)$-th round of the games, $W_{\text{EvalNA}}(CF_{\text{full}}, \alpha, \beta) = 1$. A contradiction. $\qquad\square$

A similar proof can be used to show that, for any $3n < m \leq 4n$, no distinguisher can win the following game with non-negligible probability,

---

EXP-EVALLEFT$[k+1, m]$

1. $\mathcal{D}_1'$ makes a queries for some $0 \leq a < k$ and outputs a.
2. At the $a+1$-th round of the game, $\mathcal{D}_1'$ queries $RF(x_0, x_1)$ for some pair of strings (x_0, x_1). (By definition, $\mathcal{M}^3$ will complete a subverted chain determined by (x_0, x_1). We call this full subverted chain c.)
3. $\mathcal{D}_1'$ makes another b queries for $b = k - a$.

We say $\mathcal{D}_1'$ wins **Exp-Eval**$[k+1]$ if c is not a $\mathcal{S}$.Completed chain by the end of the k-th round of the experiment and **Exp-Eval**$[k+1]$ aborts at the $(k+1)$-th round with the bad event $\mathsf{BadEval}_5[k+1]$: $\mathcal{S}^3.CF^{\text{Inner}}$(or $\mathcal{M}^3.CF^{\text{Inner}}$) calls a term (i, x) such that $m \leq i \leq 5n$ and $(i, x) \in c$.

Summarizing the results above, we have:

Lemma 41. *If for some positive integer $k < q_{\mathcal{D}}$,*

$$\Pr[\mathcal{D}_2 \text{ wins } \textbf{Exp-EvalNoAnswer}[k+1]] = \mathsf{negl}(n),$$

then, for any distinguisher $\mathcal{D}_1$,

$$\Pr[\mathcal{D}_1 \text{ wins } \textbf{Exp-Eval}[k+1]] = \mathsf{negl}(n).$$

Putting Lemma 39 and Lemma 41 together, we have:

Lemma 42. *If for some positive integer $k < q_{\mathcal{D}}$,*

$$\Pr[\mathcal{M}.\mathsf{BadComplete}_5[k+1] \text{ happens}] = \mathsf{negl}(n),$$

for any distinguisher $\mathcal{D}$, then, for any distinguisher $\mathcal{D}_1$,

$$\Pr[\mathcal{D}_1 \text{ wins } \textbf{Exp-Eval}[k+1]] = \mathsf{negl}(n).$$

Proof (Proof of Theorem 13). The theorem is implied by combining Lemma 37 and Lemma 42. $\qquad\square$

GoodStatus[k] is overwhelming In this section we prove that, assuming GoodStatus[k], GoodStatus[$k+1$] happens with overwhelming probability. We introduce the following experiment between $\mathcal{D}_4$ and $\mathcal{S}^3, \mathcal{M}^3$ to formulate the bad event:

EXP-STATUS[$k+1$]

1. $\mathcal{D}_4$ makes a queries for some $0 \le a < k$ and outputs a.
2. At the $a+1$-th round of the game, $\mathcal{D}_4$ queries $RF(x_0, x_1)$ for some pair of strings (x_0, x_1). (By definition, $\mathcal{M}^3$ will complete a subverted chain determined by (x_0, x_1). We call this full subverted chain c.)
3. $\mathcal{D}_4$ makes another b queries for $b = k - a$.

   We say $\mathcal{D}_4$ wins **Exp-Status**[$k+1$] if there is an increasing chain $(i, x_i, x_{i+1}, x_{i+2})$ in $\mathcal{M}^3.CF[k+1]$ such that $(i+1, x_{i+1}) \in Q_c$, $(i+1, x_{i+1}) \notin \mathcal{S}^3.CF[k+1]$, and:

   – $(i+2, x_{i+2}) \in \mathcal{S}^3.CF[k+1]$, or
   – there is a chain $c' \in C_{\mathcal{M}.\text{FComp}}[k+1]$ such that $(i+2, x_{i+2}) \in Q_{c'}$ and $(i+1, x_{i+1}) \notin Q_{c'}$.

   Here, without loss of generality, we ignore the case when $(i, x_i, x_{i+1}, x_{i+2})$ is decreasing.

Notice that to prove Theorem 11, it is sufficient to show the probability that $\mathcal{D}_4$ wins **Exp-Status**[$k+1$] is negligible. We will restrict our attention to the experiment in the rest of the section.

Definition 12 (Covering Index). *For any positive integer $k < q_{\mathcal{D}}$, any index i with $1 \le i \le 8n$, and any $x \in \{0,1\}^n$, we say the covering index of (i, x) at round k, $CI_k(i, x)$, is equal to t ($t \ge 1$) if $(i, x) \in \mathcal{M}^3.CF[k]$, $(i, x) \notin \mathcal{S}^3.CF[k]$, and there are exactly t elements $c_1, \ldots, c_t \in C_{\mathcal{M}.\text{FComp}}[k]$ such that $(i, x) \in Q_{c_j}$ for $j = 1, \ldots, t$. Otherwise, we say $CI_k(i, x) = 0$.*

Suppose, with non-negligible probability, $\mathcal{D}_4$ can win **Exp-Status**[$k+1$] with an increasing chain $(i, x_i, x_{i+1}, x_{i+2})$ that has the property $CI_{k+1}(i+1, x_{i+1}) \ge 2$. Then, without loss of generality, we can assume $CI_a(i+1, x_{i+1}) \ge 1$, which means $(i+1, x_{i+1})$ has been evaluated before the $(a+1)$-st query from $\mathcal{D}^4$. Using this observation, we rewrite **Exp-Status**[$k+1$] as:

EXP-STATUS[$k+1$]

1. $\mathcal{D}_4$ makes a queries for some $0 \le a < k$ and outputs a.
2. At the $a+1$-th round of the game, $\mathcal{D}_4$ queries $RF(x_0, x_1)$ for some pair of strings (x_0, x_1). (By definition, $\mathcal{M}^3$ will complete a subverted chain determined by (x_0, x_1). We call this full subverted chain c.)
3. $\mathcal{D}_4$ makes another b queries for $b = k - a$.

   We say $\mathcal{D}_4$ wins **Exp-Status**[$k+1$] if there is an increasing chain $(i, x_i, x_{i+1}, x_{i+2})$ in $\mathcal{M}^3.CF[k+1]$ such that $(i+1, x_{i+1}) \in Q_c$, $(i+1, x_{i+1}) \notin \mathcal{S}^3.CF[k+1]$, and

---

1. $CI_{k+1}(i+1, x_{i+1}) = 1$, or $CI_{k+1}(i+1, x_{i+1}) \geq 2$ and $CI_a(i+1, x_{i+1}) \geq 1$;
2. $(i+2, x_{i+2}) \in \mathcal{S}^3.CF[k+1]$, or there is a chain $c' \in C_{\mathcal{M}.\text{FComp}}[k+1]$ such that $(i+2, x_{i+2}) \in Q_{c'}$ and $(i+1, x_{i+1}) \notin Q_{c'}$.

Here, without loss of generality, we ignore the case when $(i, x_i, x_{i+1}, x_{i+2})$ is decreasing.

As in the last section, we take two steps to prove $\mathcal{D}_4$ wins **Exp-Status**$[k+1]$ negligibly. First, notice that the only information that helps $\mathcal{D}_4$ to generate a chain $(i, x_i, x_{i+1}, x_{i+2})$ that breaks the good status is the answer to the query in the $(a+1)$-th round of the game. We turn this observation into a lemma that says, without receiving the answer in the $(a+1)$-th round, the distinguisher can not win the experiment. Second, we prove whether receiving the answer or not does not affect too much the probability that $\mathcal{D}_4$ wins.

Consider the following modified version of **Exp-Status**$[k+1]$ where the distinguisher $\mathcal{D}_5$ does not receive the answer in round $(a+1)$.

EXP-STATUSNOANSWER$[k+1]$

1. $\mathcal{D}_5$ makes a queries for some $0 \leq a < k$ and outputs a.
2. At the $a+1$-st round of the game, $\mathcal{D}_5$ queries $RF(x_0, x_1)$ for some pair of strings (x_0, x_1), but does not receive the answer. (By definition, $\mathcal{M}^3$ will complete a subverted chain determined by (x_0, x_1). We call this full subverted chain c.)
3. $\mathcal{D}_5$ makes another b queries for $b = k - a$.

The winning condition of **Exp-StatusNoAnswer**$[k+1]$ is same as that of **Exp-Status**$[k+1]$.

Lemma 43. *If G_5 is efficient and for some positive integer $k < q_{\mathcal{D}}$,*

$$\Pr[\mathsf{GoodStatus}[k] \text{ does not happen}] = \mathsf{negl}(n),$$

for any distinguisher $\mathcal{D}$, then, for any distinguisher $\mathcal{D}_1$,

$$\Pr[\mathcal{D}_5 \text{ wins } \textbf{\textit{Exp-StatusNoAnswer}}[k+1]] = \mathsf{negl}(n).$$

Proof. Suppose $\mathcal{D}_5$ wins **Exp-StatusNoAnswer**$[k+1]$ with some realization of the randomness of CF: at the end of the experiment, there is an increasing chain $(i, x_i, x_{i+1}, x_{i+2})$ in $\mathcal{M}^3.CF[k+1]$ such that $(i+1, x_{i+1}) \in Q_c$, $(i+1, x_{i+1}) \notin \mathcal{S}^3.CF[k+1]$, and $(i+2, x_{i+2}) \in \mathcal{S}^3.CF[k+1]$. (Without loss of generality, we ignore the case where $(i+2, x_{i+2}) \notin \mathcal{S}^3.CF[k+1]$.)

Consider the following experiment:

EXP-MONOTONE$[k+1]$

1. $\mathcal{D}_6$ uses $\mathcal{D}_5$ to make a queries for some $0 \leq a < k$ and outputs a. For each round of the game, $\mathcal{D}_6$ receives the query from and gives the answer to $\mathcal{D}_5$.

73

2. $\mathcal{D}_6$ remembers the $a + 1$-th query $RF(x_0, x_1)$ of $\mathcal{D}_5$ but does not make the query (x_0, x_1) to $RF(\cdot)$.
3. $\mathcal{D}_6$ uses $\mathcal{D}_5$ to make another b queries for $b = k - a$. Again, for each round of the game, $\mathcal{D}_6$ receives the query from and gives the answer to $\mathcal{D}_5$.
4. $\mathcal{D}_6$ makes its last query $RF(x_0, x_1)$.

Imagine **Exp-Monotone**$[k+1]$ has the same randomness of CF with **Exp-StatusNoAnswer**$[k+1]$. Then, $(i, x_i, x_{i+1}, x_{i+2})$ is also a chain of $\mathcal{M}^3.CF[k+1]$ in **Exp-Monotone**$[k+1]$. And,

- if $CI_{k+1}(i+1, x_{i+1}) = 1$, then $CF_{i+1}(x_{i+1})$ is evaluated by $\mathcal{M}^3$ after $CF_i(x_1)$ and $CF_{i+2}(x_{i+2})$, which violates Lemma 23;
- if $CI_{k+1}(i+1, x_{i+1}) \geq 2$, $CI_a(i+1, x_{i+1}) \geq 1$, then GoodStatus$[k]$ does not happen.

The analysis above shows that, conditioned on the fact that GoodStatus$[k]$ happens negligibly, $\mathcal{D}_5$ wins negligibly. $\square$

Now we proceed to show if $\mathcal{D}_5$ wins **Exp-StatusNoAnswer**$[k + 1]$ with negligible probability, $\mathcal{D}_4$ wins **Exp-Status**$[k + 1]$ with negligible probability, too.

Choose an arbitrary integer m with $4n \leq m < 5n$. Consider the following variation of **Exp-Status**$[k + 1]$ for $\mathcal{D}_4'$.

Exp-StatusRight$[k + 1, m]$

Select uniformly a full table of CF values CF_{full} and a $2n$-bit string α. In step 1 and 3 of the experiment, all the CF values queried by the simulators are taken from CF_{full} instead of being selected uniformly as usual. In step 2, CF values queried by the simulators are generated with a special convention explained below.

1. Same as step 1 of **Exp-Status**$[k + 1]$.
2. Same as step 2 of **Exp-Status**$[k + 1]$ and the answer to the query $RF(x_0, x_1)$ is α.
3. Same as step 3 of **Exp-Status**$[k + 1]$.

In step 2, $\mathcal{M}^3$ uses the following convention to fill in its table:

1. Set $(x_{8n}, x_{8n+1}) := \alpha$.
2. For $i = (2, \ldots, m)$, define $x_i := x_{i-2} \oplus C\tilde{F}_{i-1}(x_{i-1})$. For $m + 3 \leq i \leq 8n + 1$ define $x_{i-2} := x_i \oplus C\tilde{F}_{i-1}(x_{i-1})$. All the CF values are taken from CF_{full}.
3. Set $CF_m(x_m) := x_{m-1} \oplus x_{m+1}$ and $CF_{u+1}(x_{u+1}) := x_u \oplus x_{u+2}$. The game aborts if there is an index j such that $3n \leq j \leq 5n$ and (j, x_j) is in $\mathcal{M}^3.CF[a]$ or $\bigcup_{i=1}^{8n} Q_i(x_i)/Q_j(x_j)$.
4. Evaluate $C\tilde{F}_m(x_m)$ and $C\tilde{F}_{m+1}(x_{m+1})$ and the game aborts if there is an index j such that $3n \leq j \leq 5n$ and (j, x_j) is dishonest.

The winning condition of the experiment is same as that of **Exp-Status**$[k + 1]$ except that in this experiment, $\mathcal{D}_4'$ wins if the chain $(i, x_i, x_{i+1}, x_{i+2})$ causing the bad event has the index i with $1 \leq i + 1 \leq m$. For simplicity, we say

---

74

$W_{\text{StatusRight}}(CF_{\text{full}}, \alpha, m) = 1$ if $\mathcal{D}_4'$ wins for the choice of $(CF_{\text{full}}, \alpha)$ in the experiment.

From the proof of Lemma 19, we can see the distribution of CF in **Exp-EvalRight**$[k+1, m]$ is same as that of **Exp-Eval**$[k+1]$. The only difference between the two experiments is that the new version has a more strict winning condition than the original one.

To prove the probability that $\mathcal{D}_1$ wins **Exp-Status**$[k+1]$ is negligible, we first show, for any distinguisher $\mathcal{D}_4'$, the probability that $\mathcal{D}_4'$ wins **Exp-StatusRight**$[k+1, m]$ is negligible. Consider the following rewrite of **Exp-StatusNoAnswer**$[k+1]$, where the randomness of CF is set like **Exp-StatusRight**$[k+1, m]$ and $\mathcal{D}_5$ uses $\mathcal{D}_4'$ as an oracle to play the game.

Exp-StatusNoAnswer$[k+1]$

Select uniformly a full table of CF values CF_{full} and a pair of $2n$-bit strings α and β. The randomness of CF and RF are set the same way as they are set in **Exp-StatusRight**$[k+1, m]$.

1. $\mathcal{D}_5$ uses $\mathcal{D}_4'$ to make a queries for some $0 \le a < k$ and outputs a. For each round of the game, $\mathcal{D}_5$ receives the query from and gives the answer to $\mathcal{D}_4'$.
2. At the $a+1$-st round of the game, $\mathcal{D}_5$ uses $\mathcal{D}_4'$ to query $RF(x_0, x_1)$ for some pair of strings (x_0, x_1). $\alpha := RF(x_0, x_1)$ is evaluated by $\mathcal{M}^3$ but not returned to $\mathcal{D}_5$. (By definition, $\mathcal{M}^3$ will complete a subverted chain determined by (x_0, x_1). We call this full subverted chain c.) $\mathcal{D}_5$ selects β uniformly and gives it to $\mathcal{D}_4'$.
3. $\mathcal{D}_5$ uses $\mathcal{D}_4'$ to make b queries ($b = k - a$). For each round of the game, $\mathcal{D}_5$ receives the query from and gives the answer to $\mathcal{D}_4'$.

For simplicity, we say $W_{\text{StatusNA}}(CF_{\text{full}}, \alpha, \beta) = 1$ if $\mathcal{D}_5$ wins for the choice of $(CF_{\text{full}}, \alpha, \beta)$.

Lemma 44. *If for some positive integer $k < q_{\mathcal{D}}$,*

$$\Pr[\mathcal{D}_5 \text{ wins } \textbf{Exp-StatusNoAnswer}[k+1]] = \mathsf{negl}(n)$$

over the randomness of $(CF_{\text{full}}, \alpha, \beta)$, and

$$\Pr[\mathsf{BadEval}_5[k+1]] = \mathsf{negl}(n),$$

for any distinguisher $\mathcal{D}$, then, for any integer $4n \le m < 5n$,

$$\Pr[\mathcal{D}_4' \text{ wins } \textbf{Exp-StatusRight}[k+1, m]] = \mathsf{negl}(n)$$

over the randomness of $(CF_{\text{full}}, \alpha)$.

Proof. Suppose that the first probability is negligible and the third is not. Then, with non-negligible probability, uniformly selecting a table CF_{full} and two $2n$-bit strings α, β yields:

- $W_{\text{StatusRight}}(CF_{\text{full}}, \beta) = 1$ and $\mathsf{BadEval}_5[k+1]$ does not happen in **Exp-StatusRight**$[k+1, m]$ with the parameters $(CF_{\text{full}}, \beta)$;

- $W_{\text{StatusNA}}(CF_{\text{full}}, \alpha, \beta) \neq 1$ and $\mathsf{BadEval}_5[k+1]$ does not happen in **Exp-StatusNoAnswer**$[k+1]$ with the parameters $(CF_{\text{full}}, \alpha, \beta)$.

For convenience, we call the two experiments above E_1 and E_2. Suppose the full subverted chain generated in $(a+1)$-st round of E_1 is $c_1 = (1, x_1, \ldots, x_m, y_{m+1}, \ldots, y_{8n})$, and the chain in E_2 is $c_2 = (1, x_1, \ldots, x_m, z_{m+1}, \ldots, z_{8n})$. Notice that the first m terms of c_1 and c_2 are same because the three experiments share the same table CF_{full}.

We will find a contradiction by showing that the queries to CF (made by the distinguisher or the simulators) are identical in E_1 and E_2 after the $(a+1)$-st round of the games. It is sufficient to show E_1 or E_2 does not query (m, x_m), $(m+1, y_{m+1})$ and $(m+1, z_{m+1})$, the only three terms that have different CF values in $E_1(E_2)$ and CF_{full}. This is directly implied by the fact that $\mathsf{BadEval}_5[k+1]$ does not happen in E_1 and E_2.

Since $W_{\text{StatusRight}}(CF_{\text{full}}, \beta) = 1$ and E_1, E_2 make same queries to CF after the $(a+1)$-st round of the games, $W_{\text{StatusNA}}(CF_{\text{full}}, \alpha, \beta) = 1$. A contradiction. $\qquad\square$

A similar proof can be used to show that, for any $3n < m \leq 4n$, no distinguisher can win the following game with non-negligible probability,

---

EXP-STATUSLEFT$[k+1, m]$

1. $\mathcal{D}_4'$ makes a queries for some $0 \leq a < k$ and outputs a.
2. At the $a+1$-st round of the game, $\mathcal{D}_4'$ queries $RF(x_0, x_1)$ for some pair of strings (x_0, x_1). (By definition, $\mathcal{M}^3$ will complete a subverted chain determined by (x_0, x_1). We call this full subverted chain c.)
3. $\mathcal{D}_4'$ makes another b queries for $b = k - a$.

We say $\mathcal{D}_4'$ wins **Exp-Status**$[k+1]$ if c is not a $\mathcal{S}$.Completed chain by the end of the k-th round of the experiment and **Exp-Status**$[k+1]$ aborts at the $(k+1)$-th round with the bad event $\mathsf{GoodStatus}_5[k+1]$: $\mathcal{S}^3.CF^{\text{Inner}}$(or $\mathcal{M}^3.CF^{\text{Inner}}$) calls a term (i, x) such that $m \leq i \leq 8n$ and $(i, x) \in c$.

Summarize the results above and we have:

Lemma 45. *If for some positive integer $k < q_{\mathcal{D}}$,*

$$\Pr[\mathcal{D}_5 \text{ wins } \textbf{Exp-StatusNoAnswer}[k+1]] = \mathsf{negl}(n)$$

over the randomness of $(CF_{\text{full}}, \alpha, \beta)$, and

$$\Pr[\mathsf{BadEval}_5[k+1]] = \mathsf{negl}(n),$$

for any distinguisher $\mathcal{D}$, then, for any distinguisher $\mathcal{D}_4$,

$$\Pr[\mathcal{D}_4 \text{ wins } \textbf{Exp-Status}[k+1]] = \mathsf{negl}(n).$$

Proof (Proof of Theorem 11). The theorem is implied by combining Lemma 43, Theorem 10 and Lemma 45. $\qquad\square$

B.8 Efficiency of G_5

In this section, we are going to show that the number of the elements in $S^3.CF$ and $M^3.CF$ are bounded by a polynomial function if the distinguisher $\mathcal{D}$ makes at most $q_{\mathcal{D}}$ ($q_{\mathcal{D}}$ is polynomial) queries to CF or RF (and RF^{-1}).

Lemma 46. *If G_5 is efficient, then with overwhelming probability, there are not a chain $c = (1, w_1, \ldots, w_{8n})$ and three pairwise disjoint increasing chains $c_1 = (i, x_i, \ldots, x_{i+10})$, $c_2 = (j, y_j, \ldots, y_{j+10})$ and $c_3 = (k, z_k, \ldots, z_{k+10})$ in $M^3.CF$, such that*

- *for all $(i, x) \in c$, $C\tilde{F}_i(x)$ is defined;*
- *c is disjoint with c_1, c_2, c_3;*
- *$(i + 10, x_{i+10}), (j + 10, y_{j+10}), (k + 10, z_{k+10}) \in Q_c$.*

Proof. According to Lemma 24, if $(i + 10, x_{i+10}) \in Q_c$, then there exists an index m ($i + 1 \leq m \leq i + 9$) such that in the length 3 monotone increasing chain (x_{i-1}, x_i, x_{i+1}), $(i, x_i) \notin Q_c$ but $(i + 1, x_{i+1}) \in Q_c$. Now we turn this observation into a proof.

Consider the following experiment in G_5. Take an arbitrary pair of n-bit strings (w_1, w_2). $\mathcal{D}$ tries to find a subverted chain c starting with (w_1, w_2) (w.l.o.g., we only consider subverted chain for convenience) and a length 3 increasing chain (x_{i-1}, x_i, x_{i+1}) such that $(i, x_i) \notin Q_c$ and $(i + 1, x_{i+1}) \in Q_c$. A quick thought reveals that the probability that $\mathcal{D}$ wins is negligible ($8nq_{\mathcal{A}}/s^n$): Suppose, without loss of generality, $\mathcal{D}$ queries all the elements in Q_c at the beginning of G_5. At some moment of the experiment, $\mathcal{D}$ will select a pair of terms $(i - 1, x_{i-1}, x_i)$ as the starting pair of target length 3 chain. It is easy to see that, since $(i, x_i) \notin Q_c$, $(i + 1, x_{i+1}) \in Q_c$ with probability not greater than $\text{poly}(n) \cdot 8nq_{\mathcal{A}}/2^n$, where $\text{poly}(n)$ denotes the upper bound of the number of the terms in $M^3.CF$.

For any pair (w_1, w_2), we define the event:

$$
E(w_1, w_2) := \left\{
\begin{array}{l}
\text{There are three monotone increasing unsubverted chains } c_1 = (i, x_i, \ldots, x_{i+10}), c_2 = \\
(j, y_j, \ldots, y_{j+10}) \text{ and } c_3 = (k, z_k, \ldots, z_{k+10}) \text{ in } M^3.CF, \text{ such that } c \text{ is disjoint with} \\
c_1, c_2, c_3 \text{ and } (i + 10, x_{i+10}), (j + 10, y_{j+10}), (k + 10, z_{k+10}) \in Q_c, \text{ where } c \text{ is the subverted} \\
\text{starting with } (w_1, w_2)
\end{array}
\right\}
$$

Finally we have

$$
\sum_{(w_1, w_2) \in \{0,1\}^{2n}} \Pr[E(w_1, w_2)] < 2^{2n} \cdot (\text{poly}(n) \cdot 8nq_{\mathcal{A}}/2^n)^3 = \mathsf{negl}(n). \quad \square
$$

Lemma 47. *Suppose G_5 is efficient. Let C_{11} be a set of length 11 increasing chains and c be a chain in $M^3.CF$ such that c is disjoint with any element in C_{11}, and for all $(i, x) \in c$, $C\tilde{F}_i(x)$ is defined. Then, with overwhelming probability, there are at most 20 chains $c' = (i, x_i, \ldots, x_{i+10}) \in C_{11}$ such that $(i + 10, x_{i+10}) \in Q_c$.*

Proof. Suppose there are 21 chains in C_{11} that satisfy the property in the lemma. Notice that for each length 11 chain c', there are at most 9 other length 11 chains that are not disjoint with c'. Then, among the 21 chains satisfying the property in the lemma, we can find 3 pairwise disjoint chains. This contradicts Lemma 46. $\square$

Definition 13 (Order of a chain). *We define the order of an unsubverted chain $c = (s, x_s, \ldots, x_{s+r})$ in $\mathcal{M}^3.CF$ to be:*

$$O_{\mathcal{M}^3}(c) := \min_{k=s,\ldots,s+r-1} \left\{ \max\{O_{\mathcal{M}^3}(k, x_k), O_{\mathcal{M}^3}(k+1, x_{k+1})\} \right\}$$

Intuitively speaking, the order of a chain describes the time when a chain is "determined."

Lemma 48. *Suppose G_5 is efficient. Let C_{Disj} be a set of pairwise disjoint unsubverted chains with length greater than or equal to 4 in $\mathcal{M}^3.CF$. Define the set A to be the set of the elements of the chains in C_{Disj}. Then, with overwhelming probability, $|A| \geq \sum_{c \in C_{Disj}}(L(c) - 3)$.*

Proof. For any $c \in C_{Disj}$ and a term (i, x) in c, we say (i, x) is *original* in c if there does not exist a different element $c' \in C_{Disj}$ such that c and c' intersects at (i, x) and $O_{\mathcal{M}^3}(c) \geq O_{\mathcal{M}^3}(c')$. Notice that a term (i, x) can be original in at most one chain.

Now we are going to show that, with overwhelming probability, each element in C_{Disj} contains at most 3 non-original terms. Suppose there is a chain $c = (s, x_s, \ldots, x_{s+r})$ that has four non-original terms. Then there are two non-original elements, (i, x_i) and (j, x_j), such that $s \leq i < j - 2 \leq s + r - 2$. Because of Lemma 23, without loss of generality, we assume

$$O_{\mathcal{M}^3}(i, x_i) > O_{\mathcal{M}^3}(i+1, x_{i+1}) > O_{\mathcal{M}^3}(i+2, x_{i+2}).$$

Since (i, x_i) is non-original in c, there is a chain $c' \neq c$ such that $(i, x_i) \in c'$ and $O_{\mathcal{M}^3}(c) \geq O_{\mathcal{M}^3}(c')$. Since c and c' are disjoint, $(i+1, x_{i+1}) \notin c'$. Then, since $O_{\mathcal{M}^3}(c) \geq O_{\mathcal{M}^3}(c')$ and $O_{\mathcal{M}^3}(i+1, x_{i+1}) > O_{\mathcal{M}^3}(i+2, x_{i+2})$, we have $O_{\mathcal{M}^3}(c) > O_{\mathcal{M}^3}(c')$, which means $(i+1, x_{i+1})$ is not evaluated when c' has been determined. Finally, because $\mathcal{M}^3.CF(i+1, x_{i+1})$ is selected uniformly, $(i, x_i) \in c'$ with negligible probability. A contradiction.

Going back to the proof of the lemma, since each term is original in at most one chain and each chain in C_{Disj} has all but 3 original elements, $|A|$ is lower bounded by the sum of the original terms in the elements of C_{Disj}, which is not less than $\sum_{c \in C_{Disj}}(L(c) - 3)$. $\qquad\square$

Theorem 14. *[Efficiency of G_5] For any positive integer $k \leq q_{\mathcal{D}}$, with overwhelming probability, at the end of the k-th round of G_5, there are fewer than $(88q_{\mathcal{A}} + 1)k$ terms in $\mathcal{S}^3.CF$ and fewer than $8nq_{\mathcal{A}}k$ terms in $\mathcal{M}^3.CF$.*

Remark. In the proof of Theorem 14, we will make use of Lemma 47 and Lemma 48. However, these lemmas already take efficiency of G_5 as their assumptions. To reassure the reader that there is not a circular argument here, we imagine that the k-th round of the game is forced to end when $\mathcal{S}^3.CF$ contains more than $(88q_{\mathcal{A}} + 1)k$ elements or $\mathcal{M}^3.CF$ contains more than $8nq_{\mathcal{A}}k$ elements. In this way, we can also feel free to reason about the tables of the simulators at the end of k-th round of the game (e.g., $\mathcal{S}^3.CF[k]$, $C_{\mathcal{S}.FComp}[k]$).

Proof. In $\mathcal{S}^3.CF[k]$, for any unsubverted chain c, we call c a *generator* if c was processed by the procedure $\mathcal{S}^3.HonestCheck$. We denote by C_G the set of generators. We define a function g from $C_{\mathcal{S}.FComp}[k]$ to C_G: for each $c_1 \in C_{\mathcal{S}.FComp}[k]$ and $c \in C_G$, we say $g(c_1) = c$ if $c_1 \subset c$.

Define
$$G := \{(i,x) \mid \text{there is } c \in C_G \text{ such that } (i,x) \in c.\} .$$

Since C_G is a set of pairwise disjoint chains, by Lemma 48,

$$|G| \geq \sum_{c \in C_G} (L(c) - 3) = (n/10 - 3) \cdot |C_G|. \tag{5}$$

To understand the structure of G, we define several subsets of G. We say a point $(i,x) \in G$ is a *tail point* if there is an increasing $c_2 = (s, x_s, \ldots, x_{s+10})$ in $\mathcal{M}^3.CF$ (w.l.o.g., we only consider the increasing case) and a chain $c \in C_G$ such that $(i,x) = (s+10, x_{s+10})$ and $c_2 \subset c$. We say a point $(i,x) \in G$ is a *head point* if it is not a tail point. We denote the sets of the head points and tail points by G_{Head} and G_{Tail}, respectively. For any point $(i,x) \in G_{\text{Tail}}$ and any chain $c \in C_G$, we say c covers (i,x) $((i,x) \notin c)$ if $(i,x) \in Q_c$ or $(i,x) \in Q_{f^{-1}(c)}$ (if c has a preimage in function f). We define G_{Query} to be the set of the points in G_{Tail} that are not covered by any element in C_G. Notice that any element in G_{Query} was queried directly by the distinguisher $\mathcal{D}$. Our goal is to show the size of the set G_{Query} is big.

By Lemma 23, the number of the elements in G_{Head} is easily bounded by

$$|G_{\text{Head}}| \leq 19 \cdot |C_G|. \tag{6}$$

By Lemma 47

$$|G_{\text{Tail}}/G_{\text{Query}}| \leq 20 \cdot |C_G|. \tag{7}$$

Summarizing Equation 5, 6 and 7, we have

$$
\begin{aligned}
&|G_{\text{Query}}| \\
&= |G| - |G_{\text{Head}}| - |G_{\text{Tail}}/G_{\text{Query}}| \\
&\geq (n/10 - 3)|C_G| - 19|C_G| - 20|C_{Fu}| \\
&= (n/10 - 42)|C_G| .
\end{aligned}
$$

This implies that

$$
\begin{aligned}
&|\mathcal{S}^3.CF[k]| \\
&\leq 8n \cdot q_{\mathcal{A}} \cdot |C_G| + k \\
&\leq 8n \cdot q_{\mathcal{A}} \cdot |G_{\text{Query}}|/(n/10 - 42) + k \\
&\leq 8n \cdot q_{\mathcal{A}} \cdot k/(n/10 - 42) + k \\
&\leq 8n \cdot q_{\mathcal{A}} \cdot k/(n/11) + k \\
&= (88q_{\mathcal{A}} + 1)k.
\end{aligned}
$$

Suppose $\mathcal{D}$ makes t $(0 \leq t \leq k)$ queries to the ideal object and $k - t$ queries to CF, then

$$
\begin{aligned}
&|\mathcal{M}^3.CF[k]| \\
&\leq 8n \cdot q_{\mathcal{A}} \cdot (k-t)/(n/10 - 42) + (k-t) + 8n \cdot q_{\mathcal{A}} \cdot t \\
&< (88q_{\mathcal{A}} + 1)(k-t) + 8nq_{\mathcal{A}}t \\
&\leq 8nq_{\mathcal{A}}k
\end{aligned}
$$

when n is large.

We remark that all the statements in the proof are true with overwhelming probability, we omit "with overwhelming probability" for simplicity. □

B.9   Crooked indifferentiability in the full model

Now we show the simulator $\mathcal{S}$ achieving abbreviated crooked indifferentiability can be lifted to a simulator that achieves full indifferentiability (Definition 6).

Theorem 15. *If the construction in Section 1.1 is $(n', n, q_{\mathcal{D}}, q_{\mathcal{A}}, r, \epsilon')$-Abbreviated-H-crooked-indifferentiable from a random oracle F, it is $(n', n, q_{\mathcal{D}}, q_{\mathcal{A}}, r, \epsilon' + 8n \cdot q_{\mathcal{D}}^{n/10} \cdot 2^{(2n - n^2/10)})$-H-crooked-indifferentiable from F.*

Proof. Consider the following simulator $\mathcal{S}_F$ built on $\mathcal{S}$:

1. In the first phase, $\mathcal{S}_F$ answers $f_i(x)$ $(1 \leq i \leq 8n)$ queries uniformly.
2. The second phase, after which $\mathcal{S}_F$ receives R, is divided into two sub-phases.
   – First, $\mathcal{S}_F$ simulates $\mathcal{S}$ in G_1. It then plays the role of the distinguisher, and asks $\mathcal{S}$ all the questions that were actually asked by the the distinguisher in the first phase. $\mathcal{S}_F$ *aborts* the game if, in this sub-phase, there are $n/10$ (simulated) queries such that they form a length $n/10$ unsubverted chain.
   – Second, $\mathcal{S}_F$ simulates $\mathcal{S}$ and answers the second-phase questions from the distinguisher.

For an arbitrary full model distinguisher $\mathcal{D}_F$, we construct the an abbreviated model distinguisher $\mathcal{D}$ as follows. The proof will show that, with high probability, the execution that takes place between $\mathcal{D}$ and $\mathcal{S}$ can be "lifted" to an associated execution between $\mathcal{D}_F$ and $\mathcal{S}_F$.

1. Prior to the game, $\mathcal{D}$ must publish a subversion algorithm $\mathcal{A}$. This program is constructed as follows. To decide how to subvert a certain term $f_i(x)$, $\mathcal{A}$ first simulates the first phase of $\mathcal{D}_F$; all queries made by this simulation are asked as regular queries by $\mathcal{A}$ and, at the conclusion, this first phase of $\mathcal{D}_F$ produces, as output, a subversion algorithm $\mathcal{A}_F$. $\mathcal{A}$ then simulates the algorithm $\mathcal{A}_F$ on the term $h_i(x)$.
2. In the game, $\mathcal{D}$ simulates the queries of $\mathcal{D}_F$ in $\mathcal{D}_F$'s first phase. After that, $\mathcal{D}$ continues to simulate $\mathcal{D}_F$ in the second phase. (Note that at the point in $\mathcal{D}_F$'s game where it produces the subversion algorithm $\mathcal{A}$, this is simply ignored by $\mathcal{D}$.)

Now we are ready to prove $\mathcal{S}_F$ is secure against the arbitrarily chosen distinguisher $\mathcal{D}_F$. We organize the proof around four different transcripts:

γ_{FC}	transcript of C interacting with $\mathcal{D}_F$
γ_C	transcript of C interacting with $\mathcal{D}$
γ_{FS}	transcript of $\mathcal{S}_F$ interacting with $\mathcal{D}_F$
γ_S	transcript of $\mathcal{S}$ interacting with $\mathcal{D}$

$$\mathcal{S} \qquad\qquad C \qquad\qquad \mathcal{S}_F$$
$$\overset{\gamma_S}{\nwarrow} \quad \overset{\gamma_C}{\nearrow} \qquad \overset{\gamma_{FC}}{\nwarrow} \quad \overset{\gamma_{FS}}{\nearrow}$$
$$\mathcal{D} \qquad\qquad\qquad \mathcal{D}_F$$

80
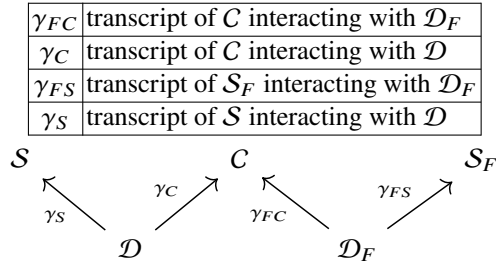
(Here C denotes the construction, as usual.) Since

$$\|\gamma_{FC} - \gamma_{FS}\|_{\text{tv}} \leq \|\gamma_{FC} - \gamma_C\|_{\text{tv}} + \|\gamma_C - \gamma_S\|_{\text{tv}} + \|\gamma_S - \gamma_{FS}\|_{\text{tv}},$$

it is sufficient to prove the three terms in the right-hand side of the inequality are all negligible.

- $\|\gamma_{FC} - \gamma_C\|_{\text{tv}} = 0$. This is obvious by observing that $\gamma_{FC} = \gamma_C$ when the underlying values of H are the same.
- $\|\gamma_C - \gamma_S\|_{\text{tv}} = \epsilon'$. This is true because $\mathcal{S}$ achieves abbreviated crooked indifferentiability.
- $\|\gamma_S - \gamma_{FS}\|_{\text{tv}} = 8n \cdot q_{\mathcal{D}}^{n/10} \cdot 2^{(2n - n^2/10)}$. To prove this statement, we suppose both the full model game and the abbreviated model game select all randomness *a priori* (as in the descriptions above). For the game between $\mathcal{S}_F$ and $\mathcal{D}_F$(the full game) or the game between $\mathcal{S}$ and $\mathcal{D}$(the abbreviated game), suppose we select a table T_F of $F(i, x)$ values for all $1 \leq i \leq 8n$ and $x \in \{0, 1\}^n$. When the simulator(in the full or abbreviated game) needs to assign a certain term uniformly, it takes the value from the table T_F. Suppose the full and the abbreviated game share the same table T_F and R. Notice that the two games have same transcripts unless, $\mathcal{S}_F$ aborts the game in the first sub-phase of the second phase. We denote this bad event by **LongChain**, which by the following lemma 49, is negligible. $\square$

N.b. While the description of the simulator above calls for all randomness to be generated in advance, it is easy to see that the simulator can in fact be carried out lazily with tables.

Lemma 49. *For any distinguisher $\mathcal{D}_F$, the probability that **LongChain** happens is less than $8n \cdot q_{\mathcal{D}}^{n/10} \cdot 2^{(2n - n^2/10)}$.*

Proof. For any pair of n-bit strings (x, y) and i with $1 \leq i \leq 8n$, over the randomness of R, the probability that **LongChain** happens with a length $n/10$ unsubverted chain starting with (i, x, y) is bounded by $(q_{\mathcal{D}} \cdot 2^{-n})^{n/10}$, which is equal to $q_{\mathcal{D}}^{n/10} \cdot 2^{-n^2/10}$. The lemma follows by taking the union bound over the choice of (x, y) and i. $\square$

To see why we need $\ell \geq 2000n/\log(1/\epsilon)$, notice that in Lemma 31, when we set $\ell = 2000n/\log(1/\epsilon)$, the upper bound of the length of a bad region is $1/48 \cdot 2000n/\log(1/\epsilon)$. Therefore, the probability in the proof will be $8n \cdot 2^{2n} \cdot (14\epsilon)^{(1/(48 \cdot 14)) \cdot 2000n/\log(1/\epsilon) - 1} = \text{negl}(n)$.

Pseudocodes of Intermediate Games

Game1. Game 1 is the ideal world interaction.

System $\mathcal{S}$:
Variable:
Queue: $Q_{\mathcal{S}}$
Tables: $\mathcal{S}.CF_1, ..., \mathcal{S}.CF_{8n}$
Order function: $O_{\mathcal{S}}$
Set $\mathcal{S}.\text{CompletedChains} := \emptyset$
Set $\mathcal{S}.\text{HonestyCheckedChains} := \emptyset$
Sets $Q_i(z) := \emptyset$ for all $i \in \{1, \ldots, 8n\}$ and $x \in \{0,1\}^n$
Sets $Q_i := \emptyset$ for all $i \in \{1, \ldots, 8n\}$
Hashtable $P \subset \{\uparrow, \downarrow\} \times \{0,1\}^{2n} \times \{0,1\}^{2n}$

1 **public procedure** $CF(i, x)$**:**
2      $\mathcal{S}.CF^{\text{Inner}}(i, x)$
3   **while** $\neg Q_{\mathcal{S}}.Empty()$ **do**
4          $(s, x_s, ..., x_{s+n/10-1}) := Q_{\mathcal{S}}.\text{Dequeue}()$
5   **if** $\mathcal{S}.Check(s, x_s, ..., x_{s+n/10-1}) = (s, x_s, x_{s+1}, u)$ **then**
6              $\mathcal{S}.\text{Complete}(s, x_s, x_{s+1}, u)$
7   Return $\mathcal{S}.CF_i(x)$

8 **public procedure** $P(x_0, x_1)$**:**
9   **while** $(\downarrow, x_0, x_1) \notin P$ **do**
10          $x_{8n} \leftarrow_R \{0,1\}^n$
11          $x_{8n+1} \leftarrow_R \{0,1\}^n$
12   **if** $(\uparrow, x_{8n}, x_{8n+1}) \notin P$ **then**
13              $P(\downarrow, x_0, x_1) := (x_{8n}, x_{8n+1})$
14              $P(\uparrow, x_{8n}, x_{8n+1}) := (x_0, x_1)$
15   Return $P(\downarrow, x_0, x_1)$

16 **public procedure** $P^{-1}(x_{8n}, x_{8n+1})$**:**
17   **while** $(\uparrow, x_{8n}, x_{8n+1}) \notin P$ **do**
18          $x_0 \leftarrow_R \{0,1\}^n$
19          $x_1 \leftarrow_R \{0,1\}^n$
20   **if** $(\downarrow, x_0, x_1) \notin P$ **then**
21              $P(\downarrow, x_0, x_1) := (x_{8n}, x_{8n+1})$
22              $P(\uparrow, x_{8n}, x_{8n+1}) := (x_0, x_1)$
23   Return $P(\uparrow, x_{8n}, x_{8n+1})$

24 **private procedure** $\mathcal{S}.CF^{Inner}(i, x)$**:**
25   **if** $x \notin \mathcal{S}.CF_i$ **then**
26          $\mathcal{S}.CF_i(x) \leftarrow_R \{0,1\}^n$
27          $\mathcal{S}.\text{EnqueueNewChain}(i, x)$
28   Return $\mathcal{S}.CF_i(x)$

29 **private procedure** $\mathcal{S}.\tilde{CF}^{Inner}(i, x)$**:**
30   **while** $\mathcal{A}(i, x)$ *queries* $CF_j(y)$ **do**
31          $Q_i(x) := Q_i(x) \cup \{(j, y)\}$
32          $\mathcal{S}.CF^{\text{Inner}}(j, y)$
     /* Simulates the subversion algorithm $\mathcal{A}$ on input (i, x) */
33   Return $\mathcal{A}(i, x)$

34 **private procedure** $\mathcal{S}.EnqueueNewChain(i, x)$:
35   **forall** $(x_{i-n/10+1}, ..., x) \in \mathcal{S}.CF_{i-n/10+1} \times \cdots \times \mathcal{S}.CF_i$ **do**
36 $Q_{\mathcal{S}}.\text{Enqueue}(i - n/10 + 1, x_{i-n/10+1}, ..., x)$
37   **forall** $(x, x_{i+1}..., x_{i+n/10-1}) \in \mathcal{S}.CF_i \times \cdots \times \mathcal{S}.CF_{i+n/10-1}$ **do**
38 $Q_{\mathcal{S}}.\text{Enqueue}(i, x, x_{i+1}..., x_{i+n/10-1})$

39 **private procedure** $\mathcal{S}.Check(s, x_s, ..., x_{s+n/10-1})$:
40   **if** $(i, x_i, x_{i+1}) \notin \mathcal{S}.Completedchains \cup \mathcal{S}.HonestyCheckedChains$ for all
       $s \leq i \leq s + n/10 - 2$ **then**
41 $\mathcal{S}.\text{HonestyCheck}(s, x_s, ..., x_{s+n/10-1})$

42 **private procedure** $\mathcal{S}.HonestyCheck(s, x_s, ..., x_{s+n/10-1})$:
43   k:=1
44   i:=s
45   **while** $s \leq i \leq s + n/10 - 2$ **do**
46 $\mathcal{S}.\text{HonestyCheckedChains} := \mathcal{S}.\text{HonestyCheckedChains} \cup (i, x_i, x_{i+1})$
47   **if** $\mathcal{S}.C\tilde{F}^{Inner}(i, x_i) = \mathcal{S}.CF_i(x_i)$ and $\mathcal{S}.C\tilde{F}^{Inner}(i + 1, x_{i+1}) = \mathcal{S}.CF_{i+1}(x_{i+1})$
        then
48   k:=k+1
49   i:=i+1
50   **if** $k=n/10$ **then**
51   **if** $i + n/10 - 1 < 3n \ or \ i > 5n$ **then**
52   Return $(s, x_s, x_{s+1}, 4n)$
53   **else**
54   Return $(s, x_s, x_{s+1}, 7n)$

55 **private procedure** $\mathcal{S}.Complete \ (s, x_s, x_{s+1}, u)$:
56 $(x_{u-2}, x_{u-1}) := \mathcal{S}.\text{EvaluateForward}(s, x_s, x_{s+1}, u)$
57 $(x_{u+2}, x_{u+3}) := \mathcal{S}.\text{EvaluateBackward}(s, x_s, x_{s+1}, u)$
58 $\mathcal{S}.\text{Adapt}(x_{u-2}, x_{u-1}, x_{u+2}, x_{u+3}, u)$

59 **private procedure** $\mathcal{S}.EvaluateForward(s, x_s, x_{s+1}, u)$:
60 $\mathcal{S}.\text{CompletedChains} := \mathcal{S}.\text{CompletedChains} \cup \{(s, x_s, x_{s+1})\}$
61   **while** $s \neq u - 1$ **do**
62   **if** $s = 8n$ **then**
63 $(x_0, x_1) := P^{-1}(x_{8n}, x_{8n+1})$
64 $s := 0$
65   **else**
66 $\mathcal{S}.\text{CompletedChains} := \mathcal{S}.\text{CompletedChains} \cup \{(s + 1, x_{s+1}, x_{s+2})\}$
67 $x_{s+2} := x_s \oplus \mathcal{S}.C\tilde{F}^{\text{Inner}}(s + 1, x_{s+1})$
68 $Q_{s+1} := Q_{s+1}(x_{s+1})$
69 $s := s + 1$
70   Return (x_{s-1}, x_s)

71 **private procedure** $\mathcal{S}.EvaluateBackward(s, x_s, x_{s+1}, u)$:
72   **while** $s \neq u + 1$ **do**
73   **if** $s = 0$ **then**
74 $(x_{8n}, x_{8n+1}) := P(x_0, x_1)$
75 $s := 8n$
76   **else**
77 $\mathcal{S}.\text{CompletedChains} := \mathcal{S}.\text{CompletedChains} \cup \{(s - 1, x_{s-1}, x_s)\}$
78 $x_{s-1} := x_{s+1} \oplus \mathcal{S}.C\tilde{F}^{\text{Inner}}(s, x_s)$
79 $Q_s := Q_s(x_s)$
80 $s := s - 1$
81   Return (x_{s+1}, x_{s+2})

82 **private procedure** $\mathcal{S}.Adapt(x_{u-2}, x_{u-1}, x_{u+2}, x_{u+3}, u)$:

83     $x_u := x_{u-2} \oplus \mathcal{S}.CF^{\text{Inner}}(u-1, x_{u-1})$

84     $x_{u+1} := x_{u+3} \oplus \mathcal{S}.CF^{\text{Inner}}(u+2, x_{u+2})$

85   **if** $x_u \notin \mathcal{S}.CF_u$ **and** $x_{u+1} \notin \mathcal{S}.CF_{u+1}$ **then**

86         $\mathcal{S}.CF_u(x_u) \leftarrow x_{u-1} \oplus x_{u+1}$

87         $\mathcal{S}.CF_{u+1}(x_{u+1}) \leftarrow x_u \oplus x_{u+2}$

88   **else**

89   The game aborts.

90   **if** $\mathcal{S}.C\tilde{F}^{\text{Inner}}(u, x_u) = \mathcal{S}.CF_u(x_u)$ *and* $\mathcal{S}.C\tilde{F}^{\text{Inner}}(u+1, x_{u+1}) = \mathcal{S}.CF_{u+1}(x_{u+1})$
      then

91         $\mathcal{S}.\text{CompletedChains} := \mathcal{S}.\text{CompletedChains} \cup \{(u, x_u, x_{u+1})\}$

92   **else**

93   The game aborts.

94     $Q_u := Q_u(x_u)$ and $Q_{u+1} := Q_{u+1}(x_{u+1})$

95   **if** $(u, x_u) \in \cup_{j=1}^{8n} Q_j/Q_u$ *or* $(u+1, x_{u+1}) \in \cup_{j=1}^{8n} Q_j/Q_{u+1}$ **then**

96   The game aborts.

Game2. The Game 2 is same as Game 1 except that the random permutation is replaced by the following two-sided random function RF.

    System RF:
    Variable:
    Hashtable $RF \subset \{\uparrow, \downarrow\} \times \{0, 1\}^{2n} \times \{0, 1\}^{2n}$

1 **public procedure** $RF(x_0, x_1)$:

2   **if** $(\downarrow, x_0, x_1) \notin RF$ **then**

3         $x_{8n} \leftarrow_R \{0, 1\}^n$

4         $x_{8n+1} \leftarrow_R \{0, 1\}^n$

5         $RF(\downarrow, x_0, x_1) := (x_{8n}, x_{8n+1})$

6         $RF(\uparrow, x_{8n}, x_{8n+1}) := (x_0, x_1)$ `/* May over write an entry */`

7   Return $RF(\downarrow, x_0, x_1)$

8 **public procedure** $RF^{-1}(x_{8n}, x_{8n+1})$:

9   **if** $(\uparrow, x_{8n}, x_{8n+1}) \notin RF$ **then**

10         $x_0 \leftarrow_R \{0, 1\}^n$

11         $x_1 \leftarrow_R \{0, 1\}^n$

12         $RF(\downarrow, x_0, x_1) := (x_{8n}, x_{8n+1})$ `/* May over write an entry */`

13         $RF(\uparrow, x_{8n}, x_{8n+1}) := (x_0, x_1)$

14   Return $RF(\uparrow, x_{8n}, x_{8n+1})$

Game3. The Game 3 has two systems, $\mathcal{S}^1$ and $\mathcal{M}^1$. We will describe all the public procedures, all the private procedures of $\mathcal{M}^1$, and the private procedures of $\mathcal{S}^1$ that are different from their counterparts of $\mathcal{S}$.

System $\mathcal{M}^1, \mathcal{S}^1, RF$**:**
Variable:
Queue: $Q_{\mathcal{S}^1}$
Tables of $\mathcal{M}^1$: $\mathcal{M}^1.CF_1, ..., \mathcal{M}^1.CF_{8n}$
Tables of $\mathcal{S}^1$: $\mathcal{S}^1.CF_1, ..., \mathcal{S}^1.CF_{8n}$
Order function: $O_{\mathcal{M}^1}, O_{\mathcal{S}^1}$
Set $\mathcal{M}^1.\text{CompletedChains}, \mathcal{S}^1.\text{CompletedChains} := \emptyset$
Set $\mathcal{S}^1.\text{HonestyCheckedChains} := \emptyset$
Set $\mathcal{M}^1.\text{MiddlePoints} := \emptyset$
Set $\mathcal{M}^1.\text{AdaptedPoints} := \emptyset$
Sets $Q_i(z) := \emptyset$ for all $i \in \{1, \ldots, 8n\}$ and $x \in \{0, 1\}^n$
Sets $Q_i := \emptyset$ for all $i \in \{1, \ldots, 8n\}$
Strings $y_i := \{0, 1\}^n$ for all $i \in \{1, \ldots, 8n\}$
Hashtable $RF \subset \{\uparrow, \downarrow\} \times \{0, 1\}^{2n} \times \{0, 1\}^{2n}$

1 **public procedure** $CF(i, x)$**:**
2 $\quad$ $\mathcal{S}^1.CF^{\text{Inner}}(i, x)$
3 $\quad$ **while** $\neg Q_{\mathcal{S}^1}.Empty()$ **do**
4 $\quad\quad$ $(s, x_s, ..., x_{s+n/10-1}) := Q_{\mathcal{S}^1}.\text{Dequeue}()$
5 $\quad\quad$ **if** $\mathcal{S}^1.Check(s, x_s, ..., x_{s+n/10-1}) = (s, x_s, x_{s+1}, u)$ **then**
6 $\quad\quad\quad$ $\mathcal{S}^1.\text{Complete}(s, x_s, x_{s+1}, u)$
7 $\quad$ Return $\mathcal{S}^1.CF_i(x)$

8 **public procedure** $RF(x_0, x_1)$**:**
9 $\quad$ **if** $(\downarrow, x_0, x_1) \notin RF$ **then**
10 $\quad\quad$ $x_{8n} \leftarrow_R \{0, 1\}^n$
11 $\quad\quad$ $x_{8n+1} \leftarrow_R \{0, 1\}^n$
12 $\quad\quad$ $RF(\downarrow, x_0, x_1) := (x_{8n}, x_{8n+1})$
13 $\quad\quad$ $RF(\uparrow, x_{8n}, x_{8n+1}) := (x_0, x_1)$/* May over write an entry */
14 $\quad\quad$ $\mathcal{M}^1.\text{Complete}(0, x_0, x_1, 4n)$
15 $\quad$ Return $RF(\downarrow, x_0, x_1)$

16 **public procedure** $RF^{-1}(x_{8n}, x_{8n+1})$**:**
17 $\quad$ **if** $(\uparrow, x_{8n}, x_{8n+1}) \notin RF$ **then**
18 $\quad\quad$ $x_0 \leftarrow_R \{0, 1\}^n$
19 $\quad\quad$ $x_1 \leftarrow_R \{0, 1\}^n$
20 $\quad\quad$ $RF(\downarrow, x_0, x_1) := (x_{8n}, x_{8n+1})$ /* May over write an entry */
21 $\quad\quad$ $RF(\uparrow, x_{8n}, x_{8n+1}) := (x_0, x_1)$
22 $\quad\quad$ $\mathcal{M}^1.\text{Complete}(8n, x_{8n}, x_{8n+1}, 4n)$
23 $\quad$ Return $RF(\uparrow, x_{8n}, x_{8n+1})$

24 **private procedure** $\mathcal{M}^1.\text{Complete}(s, x_s, x_{s+1}, u)$**:**
25 $\quad$ $(x_{u-2}, x_{u-1}) := \mathcal{M}^1.\text{EvaluateForward}(s, x_s, x_{s+1}, u)$
26 $\quad$ $(x_{u+2}, x_{u+3}) := \mathcal{M}^1.\text{EvaluateBackward}(s, x_s, x_{s+1}, u)$
27 $\quad$ $\mathcal{M}^1.\text{Adapt}(x_{u-2}, x_{u-1}, x_{u+2}, x_{u+3}, u)$

28   **private procedure** $\mathcal{M}^1.EvaluateForward(s, x_s, x_{s+1}, u)$:

29        $\mathcal{M}^1.\text{CompletedChains} := \mathcal{M}^1.\text{CompletedChains} \cup \{(s, x_s, x_{s+1})\}$

30   **while** $s \neq u - 1$ **do**

31   **if** $s = 8n$ **then**

32              $(x_0, x_1) := RF(\uparrow, x_{8n}, x_{8n+1})$

33              $s := 0$

34   **else**

35              $\mathcal{M}^1.\text{CompletedChains} := \mathcal{M}^1.\text{CompletedChains} \cup \{(s+1, x_{s+1}, x_{s+2})\}$

36              $x_{s+2} := x_s \oplus \mathcal{M}^1.C\tilde{F}^{\text{Inner}}(s+1, x_{s+1})$

37   **if** $3n \leq s+1 \leq 5n$ **then**

38                 $M^1.\text{MiddlePoints} = M^1.\text{MiddlePoints} \cup (s+1, x_{s+1})$

39              $s := s+1$

40   Return (x_{s-1}, x_s)

41   **private procedure** $\mathcal{M}^1.EvaluateBackward(s, x_s, x_{s+1}, u)$:

42   **while** $s \neq u + 1$ **do**

43   **if** $s = 0$ **then**

44              $(x_{8n}, x_{8n+1}) := RF(\downarrow, x_0, x_1)$

45              $s := 8n$

46   **else**

47              $\mathcal{M}^1.\text{CompletedChains} := \mathcal{M}^1.\text{CompletedChains} \cup \{(s-1, x_{s-1}, x_s)\}$

48              $x_{s-1} := x_{s+1} \oplus \mathcal{M}^1.C\tilde{F}^{\text{Inner}}(s, x_s)$

49   **if** $3n \leq s \leq 5n$ **then**

50                 $\mathcal{M}^1.\text{MiddlePoints} = \mathcal{M}^1.\text{MiddlePoints} \cup (s, x_s)$

51              $s := s-1$

52   Return (x_{s+1}, x_{s+2})

53   **private procedure** $\mathcal{M}^1.Adapt(x_{u-2}, x_{u-1}, x_{u+2}, x_{u+3}, u)$:

54        $x_u := x_{u-2} \oplus \mathcal{M}^1.CF^{\text{Inner}}(u-1, x_{u-1})$

55        $x_{u+1} := x_{u+3} \oplus \mathcal{M}^1.CF^{\text{Inner}}(u+2, x_{u+2})$

56   **if** $x_u \notin \mathcal{M}^1.CF_u$ **then**

57           $\mathcal{M}^1.CF_u(x_u) \leftarrow x_{u-1} \oplus x_{u+1}$

58           $\mathcal{M}^1.\text{MiddlePoints} = \mathcal{M}^1.\text{MiddlePoints} \cup (u, x_u)$

59           $\mathcal{M}^1.\text{AdaptedPoints} = \mathcal{M}^1.\text{AdaptedPoints} \cup (u, x_u)$

60   **if** $x_{u+1} \notin \mathcal{M}^1.CF_{u+1}$ **then**

61           $\mathcal{M}^1.CF_{u+1}(x_{u+1}) \leftarrow x_u \oplus x_{u+2}$

62           $\mathcal{M}^1.\text{MiddlePoints} = \mathcal{M}^1.\text{MiddlePoints} \cup (u+1, x_{u+1})$

63           $\mathcal{M}^1.\text{AdaptedPoints} = \mathcal{M}^1.\text{AdaptedPoints} \cup (u+1, x_{u+1})$

64   **if** $\mathcal{M}^1.C\tilde{F}^{\text{Inner}}(u, x_u) = \mathcal{M}^1.CF_u(x_u)$ *and*

          $\mathcal{M}^1.C\tilde{F}^{\text{Inner}}(u+1, x_{u+1}) = \mathcal{M}^1.CF_{u+1}(x_{u+1})$ **then**

65           $\mathcal{M}^1.\text{CompletedChains} := \mathcal{M}^1.\text{CompletedChains} \cup \{(u, x_u, x_{u+1})\}$

66   **private procedure** $\mathcal{M}^1.CF^{\text{Inner}}(i, x)$:

67   **if** $x \notin \mathcal{M}^1.CF_i$ **then**

68           $\mathcal{M}^1.CF_i(x) \leftarrow_R \{0,1\}^n$

69   Return $\mathcal{M}^1.CF_i(x)$

116 **private procedure** $\mathcal{S}^1.EvaluateBackward(s, x_s, x_{s+1}, u)$:
117   **while** $s \neq u + 1$ **do**
118   **if** $s = 0$ **then**
119   **if** $(\downarrow, x_0, x_1) \notin RF$ **then**
120                 $x_{8n} \leftarrow_R \{0, 1\}^n$
121                 $x_{8n+1} \leftarrow_R \{0, 1\}^n$
122                 $RF(\downarrow, x_0, x_1) := (x_{8n}, x_{8n+1})$
123                 $RF(\uparrow, x_{8n}, x_{8n+1}) := (x_0, x_1)$`/* May over write an entry */`
124   **else**
125                 $(x_{8n}, x_{8n+1}) := RF(\downarrow, x_0, x_1)$
126             $s := 8n$
127   **else**
128             $\mathcal{S}^1.\text{CompletedChains} := \mathcal{S}^1.\text{CompletedChains} \cup \{(s-1, x_{s-1}, x_s)\}$
129             $x_{s-1} := x_{s+1} \oplus \mathcal{S}^1.C\tilde{F}^{\text{Inner}}(s, x_s)$
130             $Q_s := Q_s(x_s)$
131             $s := s - 1$
132   Return (x_{s+1}, x_{s+2})

133 **private procedure** $\mathcal{S}^1.Adapt(x_{u-2}, x_{u-1}, x_{u+2}, x_{u+3}, u)$:
134     $x_u := x_{u-2} \oplus \mathcal{S}^1.CF^{\text{Inner}}(u-1, x_{u-1})$
135     $x_{u+1} := x_{u+3} \oplus \mathcal{S}^1.CF^{\text{Inner}}(u+2, x_{u+2})$
136   **if** $x_u \notin \mathcal{S}^1.CF_u$ and $x_{u+1} \notin \mathcal{S}^1.CF_{u+1}$ **then**
137         $\mathcal{S}^1.CF_u(x_u) \leftarrow x_{u-1} \oplus x_{u+1}$
138         $\mathcal{S}^1.CF_{u+1}(x_{u+1}) \leftarrow x_u \oplus x_{u+2}$
139         $\mathcal{M}^1.CF_u(x_u) \leftarrow \mathcal{S}^1.CF_u(x_u)$`/* May over write an entry */`
140         $\mathcal{M}^1.CF_{u+1}(x_{u+1}) \leftarrow \mathcal{S}^1.CF_{u+1}(x_{u+1})$`/* May over write an entry */`
141   **else**
142   The game aborts.
143   **if** $\mathcal{S}^1.C\tilde{F}^{\text{Inner}}(u, x_u) = \mathcal{S}^1.CF_u(x_u)$ and
            $\mathcal{S}^1.C\tilde{F}^{\text{Inner}}(u+1, x_{u+1}) = \mathcal{S}^1.CF_{u+1}(x_{u+1})$ **then**
144         $\mathcal{S}^1.\text{CompletedChains} := \mathcal{S}^1.\text{CompletedChains} \cup \{(u, x_u, x_{u+1})\}$
145   **else**
146   The game aborts.
147     $Q_u := Q_u(x_u)$ and $Q_{u+1} := Q_{u+1}(x_{u+1})$
148   **if** $(u, x_u) \in \cup_{j=1}^{8n} Q_j/Q_u$ or $(u+1, x_{u+1}) \in \cup_{j=1}^{8n} Q_j/Q_{u+1}$ **then**
149   The game aborts.

Game4. To obtain G_4, we just need to add some abortion conditions to several procedures of G_3. Below we only show the procedures of $\mathcal{S}^2$ and $\mathcal{M}^2$ that are different from their counterparts of $\mathcal{S}^1$ and $\mathcal{M}^1$. We use red color to stress the extra abortion conditions.

```
72  private procedure M¹.CF̃ˡⁿⁿᵉʳ(i, x):
73      while 𝒜(i, x) queries CFⱼ(y) do
74          Qᵢ(x) := Qᵢ(x) ∪ {(j, y)}
75          M¹.CFⁱⁿⁿᵉʳ(j, y) /* Simulates the subversion algorithm 𝒜 on
                  input (i, x) */
76      Return 𝒜(i, x)

77  private procedure 𝒮¹.CFⁱⁿⁿᵉʳ(i, x):
78      if x ∉ 𝒮¹.CFᵢ then
79          if x ∉ M¹.CFᵢ then
80              𝒮¹.CFᵢ(x) ←ᵣ {0, 1}ⁿ
81              M¹.CFᵢ(x) ← 𝒮¹.CFᵢ(x)
82          else if x ∈ M¹.CFᵢ and (i, x) ∉ M¹.AdaptedPoints then
83              𝒮¹.CFᵢ(x) ← M¹.CFᵢ(x)
84          else
85              𝒮¹.CFᵢ(x) ←ᵣ {0, 1}ⁿ
86          𝒮¹.EnqueueNewChain(i, x)
87      Return 𝒮¹.CFᵢ(x)

88  private procedure 𝒮¹.EvaluateForward(s, xₛ, xₛ₊₁, u):
89      𝒮¹.CompletedChains := 𝒮¹.CompletedChains ∪ {(s, xₛ, xₛ₊₁)}
90      while s ≠ u − 1 do
91          if s = 8n then
92              if (↑, x₈ₙ, x₈ₙ₊₁) ∉ RF then
93                  x₀ ←ᵣ {0, 1}ⁿ
94                  x₁ ←ᵣ {0, 1}ⁿ
95                  RF(↓, x₀, x₁) := (x₈ₙ, x₈ₙ₊₁) /* May over write an entry */
96                  RF(↑, x₈ₙ, x₈ₙ₊₁) := (x₀, x₁)
97              else
98                  (x₀, x₁) := RF(↑, x₈ₙ, x₈ₙ₊₁)
99              s := 0
100         else
101             𝒮¹.CompletedChains := 𝒮¹.CompletedChains ∪ {(s + 1, xₛ₊₁, xₛ₊₂)}
102             xₛ₊₂ := xₛ ⊕ 𝒮¹.CF̃ⁱⁿⁿᵉʳ(s + 1, xₛ₊₁)
103             Qₛ₊₁ := Qₛ₊₁(xₛ₊₁)
104             s := s + 1
105     Return (xₛ₋₁, xₛ)
```

1 **private procedure** $\mathcal{M}^2.CF^{Inner}(i,x)$:

2   **if** $x \notin \mathcal{M}^2.CF_i$ **then**

3         $\mathcal{M}^2.CF_i(x) \leftarrow_R \{0,1\}^n$

4   **else if** $(i,x) \in \mathcal{M}^2.MiddlePoints$ and $(i,x) \notin \mathcal{S}^2.CF_i$ **then**

5   The game aborts.

6   Return $\mathcal{M}^1.CF_i(x)$

7 **private procedure** $\mathcal{M}^2.EvaluateForward(s, x_s, x_{s+1}, u)$:

8     $\mathcal{M}^2.\text{CompletedChains} := \mathcal{M}^2.\text{CompletedChains} \cup \{(s, x_s, x_{s+1})\}$

9   **while** $s \neq u - 1$ **do**

10   **if** $s = 8n$ **then**

11             $(x_0, x_1) := RF(\uparrow, x_{8n}, x_{8n+1})$

12             $s := 0$

13   **else**

14             $\mathcal{M}^2.\text{CompletedChains} :=$
            $\mathcal{M}^2.\text{CompletedChains} \cup \{(s+1, x_{s+1}, x_{s+2})\}$

15   **if** $3n \leq s+1 \leq 5n$ **then**

16                 $\mathcal{M}^2.\text{MiddlePoints} = \mathcal{M}^2.\text{MiddlePoints} \cup (s+1, x_{s+1})$

17   **if** $3n \leq s+1 \leq 5n$ and $x_{s+1} \in \mathcal{M}^2.CF$ **then**

18   The game aborts.

19             $x_{s+2} := x_s \oplus \mathcal{M}^2.\tilde{CF}^{Inner}(s+1, x_{s+1})$

20             $Q_{s+1} := Q_{s+1}(x_{s+1})$

21             $y_{s+1} := x_{s+1}$

22   **if** $3n \leq s+1 \leq 5n$ and $\mathcal{M}^2.CF_{s+1}(x_{s+1}) \neq x_{s+2} \oplus x_s$ **then**

23   The game aborts.

24             $s := s + 1$

25   Return (x_{s-1}, x_s)

Game5. G_5 is different from G_4 in the following procedures:

24 **private procedure** $\mathcal{M}^2.EvaluateBackward(s, x_s, x_{s+1}, u)$:
25   **while** $s \neq u + 1$ **do**
26   **if** $s = 0$ **then**
27              $(x_{8n}, x_{8n+1}) := RF(\downarrow, x_0, x_1)$
28              $s := 8n$
29   **else**
30              $\mathcal{M}^2.\text{CompletedChains} := \mathcal{M}^2.\text{CompletedChains} \cup \{(s-1, x_{s-1}, x_s)\}$
31   **if** $3n \leq s \leq 5n$ **then**
32                  $\mathcal{M}^2.\text{MiddlePoints} = \mathcal{M}^2.\text{MiddlePoints} \cup (s, x_s)$
33   **if** $3n \leq s \leq 5n$ and $x_s \in \mathcal{M}^2.CF$ **then**
34   The game aborts.
35              $x_{s-1} := x_{s+1} \oplus \mathcal{M}^2.C\bar{F}^{\text{Inner}}(s, x_s)$
36              $Q_s := Q_s(x_s)$
37              $y_s := x_s$
38   **if** $3n \leq s \leq 5n$ and $\mathcal{M}^2.CF_s(x_s) \neq x_{s+1} \oplus x_{s-1}$ **then**
39   The game aborts.
40              $s := s - 1$
41   Return (x_{s+1}, x_{s+2})

42 **private procedure** $\mathcal{M}^2.Adapt(x_{u-2}, x_{u-1}, x_{u+2}, x_{u+3}, u)$:
43      $x_u := x_{u-2} \oplus \mathcal{M}^2.CF^{\text{Inner}}(u-1, x_{u-1})$
44      $x_{u+1} := x_{u+3} \oplus \mathcal{M}^2.CF^{\text{Inner}}(u+2, x_{u+2})$
45   **if** $x_u \notin \mathcal{M}^2.CF_u$ and $x_{u+1} \notin \mathcal{M}^2.CF_{u+1}$ **then**
46          $\mathcal{M}^2.CF_u(x_u) \leftarrow x_{u-1} \oplus x_{u+1}$
47          $\mathcal{M}^2.\text{MiddlePoints} = \mathcal{M}^2.\text{MiddlePoints} \cup (u, x_u)$
48          $\mathcal{M}^2.\text{AdaptedPoints} = \mathcal{M}^2.\text{AdaptedPoints} \cup (u, x_u)$
49          $\mathcal{M}^2.CF_{u+1}(x_{u+1}) \leftarrow x_u \oplus x_{u+2}$
50          $\mathcal{M}^2.\text{MiddlePoints} = \mathcal{M}^2.\text{MiddlePoints} \cup (u+1, x_{u+1})$
51          $\mathcal{M}^2.\text{AdaptedPoints} = \mathcal{M}^2.\text{AdaptedPoints} \cup (u+1, x_{u+1})$
52   **else**
53   The game aborts.
54   **if** $\mathcal{M}^2.C\bar{F}^{Inner}(u, x_u) = \mathcal{M}^2.CF_u(x_u)$ and
          $\mathcal{M}^2.C\bar{F}^{Inner}(u+1, x_{u+1}) = \mathcal{M}^2.CF_{u+1}(x_{u+1})$ **then**
55          $\mathcal{M}^2.\text{CompletedChains} := \mathcal{M}^2.\text{CompletedChains} \cup \{(u, x_u, x_{u+1})\}$
56   **else**
57   The game aborts.
58      $Q_u := Q_u(x_u)$ and $Q_{u+1} := Q_{u+1}(x_{u+1})$
59      $y_u := x_u$ and $y_{u+1} := x_{u+1}$
60   **while** $3n \leq i \leq 5n$ **do**
61   **if** $(i, y_i) \in \cup_{j=1}^{8n} Q_j / Q_i$ **then**
62   The game aborts.

91

65 **private procedure** $\mathcal{S}^2.CF^{Inner}(i, x)$:
66   **if** $x \notin \mathcal{S}^2.CF_i$ **then**
67   **if** $x \notin \mathcal{M}^2.CF_i$ **then**
68             $\mathcal{S}^2.CF_i(x) \leftarrow_R \{0, 1\}^n$
69             $\mathcal{M}^2.CF_i(x) \leftarrow \mathcal{S}^2.CF_i(x)$
70   **else if** $x \in \mathcal{M}^2.CF_i$ *and* $(i, x) \notin \mathcal{M}^2.MiddlePoints$ **then**
71             $\mathcal{S}^2.CF_i(x) \leftarrow \mathcal{M}^2.CF_i(x)$
72   **else**
73   The game aborts.
74         $\mathcal{S}^2.\text{EnqueueNewChain}(i, x)$
75   Return $\mathcal{S}^2.CF_i(x)$

76 **private procedure** $\mathcal{S}^2.Complete\ (s, x_s, x_{s+1}, u)$:
77   **if** $(s, x_s, x_{s+1}) \in \mathcal{M}^2.CompletedChains$ *and* $u = 7n$ **then**
78   The game aborts.
79   **else if** $(s, x_s, x_{s+1}) \in \mathcal{M}^2.CompletedChains$ *and* $u = 4n$ **then**
80   Copy the full subverted chain containing (s, x_s, x_{s+1}) in $\mathcal{M}^2.CF$ to $\mathcal{S}^2.CF$
81   **else**
82         $(x_{u-2}, x_{u-1}) := \mathcal{S}^2.\text{EvaluateForward}(s, x_s, x_{s+1}, u)$
83         $(x_{u+2}, x_{u+3}) := \mathcal{S}^2.\text{EvaluateBackward}(s, x_s, x_{s+1}, u)$
84         $\mathcal{S}^2.\text{Adapt}(x_{u-2}, x_{u-1}, x_{u+2}, x_{u+3}, u)$

85 **private procedure** $\mathcal{S}^2.Adapt(x_{u-2}, x_{u-1}, x_{u+2}, x_{u+3}, u)$:
86     $x_u := x_{u-2} \oplus \mathcal{S}^2.CF^{\text{Inner}}(u - 1, x_{u-1})$
87     $x_{u+1} := x_{u+3} \oplus \mathcal{S}^2.CF^{\text{Inner}}(u + 2, x_{u+2})$
88   **if** $x_u \notin \mathcal{S}^2.CF_u, x_{u+1} \notin \mathcal{S}^2.CF_{u+1}$ **then**
89         $\mathcal{S}^2.CF_u(x_u) \leftarrow x_{u-1} \oplus x_{u+1}$
90         $\mathcal{S}^2.CF_{u+1}(x_{u+1}) \leftarrow x_u \oplus x_{u+2}$
91   **if** $x_u \notin \mathcal{M}^2.CF_u$ *and* $x_{u+1} \notin \mathcal{M}^2.CF_{u+1}$ **then**
92             $\mathcal{M}^2.CF_u(x_u) \leftarrow \mathcal{S}^2.CF_u(x_u)$
93             $\mathcal{M}^2.CF_u(x_{u+1}) \leftarrow \mathcal{S}^2.CF_{u+1}(x_{u+1})$
94   **else**
95   The game aborts.
96   **else**
97   The game aborts.
98   **if** $\mathcal{S}^2.C\tilde{F}^{Inner}(u, x_u) = \mathcal{S}^2.CF_u(x_u)$ *and*
        $\mathcal{S}^2.C\tilde{F}^{Inner}(u + 1, x_{u+1}) = \mathcal{S}^2.CF_{u+1}(x_{u+1})$ **then**
99         $\mathcal{S}^2.CompletedChains := \mathcal{S}^2.CompletedChains \cup \{(u, x_u, x_{u+1})\}$
100   **else**
101   The game aborts.
102     $Q_u := Q_u(x_u)$ and $Q_{u+1} := Q_{u+1}(x_{u+1})$
103   **if** $(u, x_u) \in \cup_{j=1}^{8n} Q_j/Q_u$ *or* $(u + 1, x_{u+1}) \in \cup_{j=1}^{8n} Q_j/Q_{u+1}$ **then**
104   The game aborts.

92

1 **public procedure** $RF(x_0, x_1)$:
2   **if** $(\downarrow, x_0, x_1) \notin RF$ **then**
3          $\mathcal{M}^3.\text{Complete } (0, x_0, x_1, 4n)$
4   Return $RF(\downarrow, x_0, x_1)$

5 **public procedure** $RF^{-1}(x_{8n}, x_{8n+1})$:
6   **if** $(\uparrow, x_{8n}, x_{8n+1}) \notin RF$ **then**
7          $\mathcal{M}^3.\text{Complete } (8n, x_{8n}, x_{8n+1}, 4n)$
8   Return $RF(\uparrow, x_{8n}, x_{8n+1})$

9 **private procedure** $\mathcal{M}^3.Complete \ (s, x_s, x_{s+1}, u)$:
10   **if** $s = 0$ **then**
11   **while** $s \neq 8n$ **do**
12              $x_{s+2} := x_s \oplus \mathcal{M}^3.C\tilde{F}^{\text{Inner}}(s + 1, x_{s+1})$
13   **if** $3n \leq s + 1 \leq 5n$ and $\mathcal{M}^3.CF_{s+1}(x_{s+1}) \neq x_s \oplus x_{s+2}$ **then**
14   The game aborts.
15   **if** $3n \leq s + 1 \leq 5n$ **then**
16                  $\mathcal{M}^3.\text{MiddlePoints} = \mathcal{M}^3.\text{MiddlePoints} \cup (s + 1, x_{s+1})$
17   **if** $s + 1 = u, u + 1$ **then**
18                  $\mathcal{M}^3.\text{AdaptedPoints} = \mathcal{M}^3.\text{AdaptedPoints} \cup (s + 1, x_{s+1})$
19              $\mathcal{M}^3.\text{CompletedChains} :=$
                 $\mathcal{M}^3.\text{CompletedChains} \cup \{(s + 1, x_{s+1}, x_{s+2})\}$
20              $s := s + 1$
21   **else**
22   **while** $s \neq 0$ **do**
23              $x_{s-1} := x_{s+1} \oplus \mathcal{M}^3.C\tilde{F}^{\text{Inner}}(s, x_s)$
24   **if** $3n \leq s \leq 5n$ and $\mathcal{M}^3.CF_s(x_s) \neq x_{s+1} \oplus x_{s-1}$ **then**
25   The game aborts.
26   **if** $3n \leq s \leq 5n$ **then**
27                  $\mathcal{M}^3.\text{MiddlePoints} = \mathcal{M}^3.\text{MiddlePoints} \cup (s, x_s)$
28   **if** $s = u, u + 1$ **then**
29                  $\mathcal{M}^3.\text{AdaptedPoints} = \mathcal{M}^3.\text{AdaptedPoints} \cup (s, x_s)$
30              $\mathcal{M}^3.\text{CompletedChains} := \mathcal{M}^3.\text{CompletedChains} \cup \{(s-1, x_{s-1}, x_s)\}$
31              $s := s - 1$
32   **while** $3n \leq i \leq 5n$ **do**
33   **if** $(i, x_i) \in \cup_{j=1}^{8n} Q_j / Q_i$ **then**
34   The game aborts.
35      $RF(\downarrow, x_0, x_1) := (x_{8n}, x_{8n+1})$
36      $RF(\uparrow, x_{8n}, x_{8n+1}) := (x_0, x_1)$

```
32  private procedure $\mathcal{S}^3.Complete\ (s, x_s, x_{s+1}, u)$:
33      if $(s, x_s, x_{s+1}) \in \mathcal{M}^3.CompletedChains$ and $u = 7n$ then
34          The game aborts.
35      else if $(s, x_s, x_{s+1}) \in \mathcal{M}^3.CompletedChains$ and $u = 4n$ then
36          Copy the full subverted chain containing $(s, x_s, x_{s+1})$ in $\mathcal{M}^3.CF$ to $\mathcal{S}^3.CF$
37      else
38          $i := s$
39          while $s \leq i \leq 8n$ do
40              if $i + 1 = u$ or $i + 1 = u + 1$ then
41                  if $x_{i+1}$ is in $\mathcal{S}^3.CF_{i+1}$ or $\mathcal{M}^3.CF_{i+1}$ then
42                      The game aborts.
43              $x_{i+2} := x_i \oplus \mathcal{S}^3.C\tilde{F}^{\text{Inner}}(i + 1, x_{i+1})$
44              if $i + 1 = u$ or $i + 1 = u + 1$ then
45                  if $\mathcal{S}^3.CF^{\text{Inner}}(i + 1, x_{i+1}) \neq \mathcal{S}^3.C\tilde{F}^{\text{Inner}}(i + 1, x_{i+1})$ then
46                      The game aborts.
47              $\mathcal{S}^3.CompletedChains := \mathcal{S}^3.CompletedChains \cup \{(i + 1, x_{i+1}, x_{i+2})\}$
48              $i := i + 1$
49          $j := s$
50          while $0 \leq j \leq s$ do
51              if $j = u$ or $j = u + 1$ then
52                  if $x_j$ is in $\mathcal{S}^3.CF_j$ or $\mathcal{M}^3.CF_j$ then
53                      The game abort.
54              $x_{j-1} := x_{j+1} \oplus \mathcal{S}^3.C\tilde{F}^{\text{Inner}}(j, x_j)$
55              if $j = u$ or $j = u + 1$ then
56                  if $\mathcal{S}^3.CF^{\text{Inner}}(j, x_j) \neq \mathcal{S}^3.C\tilde{F}^{\text{Inner}}(j, x_j)$ then
57                      The game aborts.
58              $\mathcal{S}^3.CompletedChains := \mathcal{S}^3.CompletedChains \cup \{(j - 1, x_{j-1}, x_j)\}$
59              $j := j - 1$
60          if $(u, x_u) \in \cup_{j=1}^{8n} Q_j / Q_u$ or $(u + 1, x_{u+1}) \in \cup_{j=1}^{8n} Q_j / Q_{u+1}$ then
61              The game aborts.
62      $RF(\downarrow, x_0, x_1) := (x_{8n}, x_{8n+1})$
63      $RF(\uparrow, x_{8n}, x_{8n+1}) := (x_0, x_1)$
```