Crooked Indifferentiability Revisited

Rishiraj Bhattacharyya[1], Mridul Nandi[2], and Anik Raychaudhuri[2]

[1] NISER, HBNI, India, rishiraj.bhattacharyya@gmail.com
[2] Indian Statistical Institute, Kolkata, India,
mridul.nandi@gmail.com,anikrc1@gmail.com

Abstract. In CRYPTO 2018, Russell et al. introduced the notion of crooked indifferentiability to analyze the security of a hash function when the underlying primitive is subverted. They showed that the n-bit to n-bit function implemented using enveloped XOR construction (EXor) with $3n+1$ many n-bit functions and $3n^2$- bit random initial vectors (iv) can be proven secure asymptotically in the crooked indifferentiability setting.

- We identify several major issues and gaps in the proof by Russel et al. We show that their proof does not work when the adversary makes queries related to multiple messages or in the case of intricate function dependent subversion.
- We formalize new technique to prove crooked indifferentiability for multiple messages. Our technique can handle function dependent subversion. We apply our technique to provide a concrete proof for the EXor construction.
- We analyze crooked indifferentiability of the classical sponge construction. We show, using a simple proof idea, the sponge construction is a crooked-indifferentiable hash function using only n-bit random iv.

1 Introduction

Blackbox Reduction and Kleptographic attack. Many of the modern cryptographic constructions are analyzed in a modular and inherently black-box manner. The schemes or protocols are built on underlying primitives only exploiting the functionality of the primitives. While analyzing the security, one shows a reduction saying, a successful attack on the construction will lead to an attack against the underlying primitive. Unfortunately, this approach completely leaves out the implementation aspects. While the underlying primitive may be well studied, a malicious implementation may embed trapdoor or other sensitive information that can be used for the attack. Moreover, such implementation may well be indistinguishable from a faithful implementation [17]. These type attacks fall in the realm of *Kleptography*, introduced by Young and Young[17]. While the cryptographic community did not consider kleptography as a real threat, the scenario has changed in the past few years. Kleptographic attack has been a real possibility in the post-Snowden world. A line of work has appeared aiming to formalize and provide security against kleptographic attack [2,8,14,15].

Specifically, in [2], Bellare, Paterson, and Rogaway showed that it is possible to mount algorithm substitution attack against almost all known symmetric key encryption scheme to the extent that the attacker learns the secret key.

Random Oracle and Indifferentiability. The *Random Oracle* methodology is a very popular platform for proving the security of cryptographic constructions in the black-box fashion. In this model, all the parties, including the adversary, is given access to a common truly random function. One proves the security of a protocol assuming the existence of such a random function. During the implementation of the protocol, the random function is replaced by a hash function H. The *Indifferentiability* framework and the composition theorem [10] assert that if the hash function H is based on an ideal primitive f, and Indifferentiable from a random function, then the instantiated protocol is as secure as the protocol in the random oracle model (assuming the security of the ideal primitive f). Indifferentiability from Random Oracle has been one of the mainstream security criteria of cryptographic hash functions. Starting from the work of Coron et al. [7], a plethora of results [6,4,5,9,1,12,11,13] have been proven, showing indifferentiability of different constructions based on different ideal primitives.

Crooked Indifferentiability. In CRYPTO 2018, Russel et al.[16] introduced the notion of crooked indifferentiability as a security notion for hash functions in the kleptographic setting. In this setting, the distinguisher can *substitute* the ideal primitive f by a subverted implementation which "crooks" the function on ϵ fraction of the inputs, producing a crooked primitive $\tilde{f}$. In order to prove crooked-indifferentiability of a construction C^f, one constructs a simulator S such that $(C^{\tilde{f}}(.,R),f)$ and $(\mathcal{F},S^{\tilde{f}}(R))$ are indistinguishable.

Enveloped Xor Construction and its crooked-indifferentiability. We recall the Enveloped XOR construction. We fix two positive integers n and l. Let $\mathcal{D} := \{0,1,\ldots,l\}\times\{0,1\}^n$. Let H be the class of all functions $h : \mathcal{D} \to \{0,1\}^n$. For every $x \in \{0,1\}^n$ and an initial value $R := (r_1,\ldots,r_l) \in (\{0,1\}^n)^l$, we define

$$g_R(x) = \bigoplus_{i=1}^{l} h(i, x \oplus r_i) \quad \text{and} \quad \mathsf{EXor}(R,x) = h(0, g_R(x)).$$

In [16], Russel *et al* proved crooked-indifferentiability of the enveloped-xor construction. Their analysis is based on an interesting rejection sampling argument.

Even though, the notion of crooked indifferentiability is extremely important in the post-Snowden cryptographic arena, not much is known except the above results. Specifically, it is unknown whether the popular modes of operations used in cryptographic hash functions can be proven secure under this notion. The hash functions are often used to instantiate random oracles, and it is imperative that a strong hash function design can resist subversion. In addition, the efficiency of the enveloped xor Construction, proved secure in [16], is far from optimal. It uses $3n+1$ *independent* hash functions (one call to each) and $3n^2$ bits of randomness in order to construct an n-bit to n-bit subversion resilient function! When we instantiate that to construct practical hash functions, then this cost will amplify

and we end up with somewhat impractical construction. In fact, the problem of constructing an efficient crooked-indifferentiable hash function was left open as the central question by [16].

1.1 Our Contribution

In this paper, we introduce new techniques to prove crooked-indifferentiability of domain extension techniques and prove security of practical constructions.

Another Look at Russel *et al* proof. We start from the observation that the techniques of [16], while novel and interesting, are limited in their capability. The proof works perfectly fine when the adversary makes queries related to a single message. So, a natural question to ask is, *Do the techniques of [16] work in general with multiple messages or somewhat involved crooking algorithm.* We answer the question in negative. We identify several aspects of their technique that can not be reasoned for events related to multiple queries.

Crooked-indifferentiability for multiple messages and application to Enveloped XOR. We develop techniques to prove crooked indifferentiability against adversaries making queries related to multiple messages. Interestingly, our technique do not involve heavy machinery. Rather, we identify core domain points related to functions, and use simple tools like Markov inequality.
We apply our technique to prove security of Enveloped XOR construction.

Crooked-indifferentiability Sponge construction.
We prove crooked-indifferentiability of sponge construction with randomized initialization vector. We assume the underlying primitive to be a random function. The construction uses only n-bit randomness and even with the most conservative parameter choices, makes at most $2n$ many calls to the underlying primitive to construct an n-bit to n-bit function. The result is applicable to general hash function as well and is not limited to produce length preserving ones.

1.2 Overview of Our Technique.

Challenges for multiple messages. Why can't the technique of Russel *et al* [16] achieve security against multiple messages. Let us revisit the main idea of the proof. It is to argue that for every message there exists an index $i \in \{1, \cdots, l\}$ for which the evaluation of $h(i, m \oplus r_i)$ is unsubverted. Moreover, for no other index j, $\tilde{h}(j, m \oplus r_j)$, the subverted evaluation of j^{th} function queries $h(i, m \oplus r_i)$. In such a situation, if resampled, $\tilde{h}(i, m \oplus r_i)$, is uniform over $\{0,1\}^n$ and most importantly, independent from other query/responses . As there are at most ϵ fraction of crooked domain points of $h(0, \cdot)$, the probability that $\tilde{g}_R(m)$, the possibly subverted evaluation of $g_R(m)$, becomes one of the crooked point, is negligible.

Unfortunately, the situation changes for multiple messages. Consider an implementation, where for every i, x,$\tilde{h}(i, x)$ queries the values of $h(i, x)$ as well as $h(x \oplus 1^n)$. Now, for any message m, computation of $\tilde{g}_R(m \oplus 1^n)$, involves evaluation of both $h(i, m \oplus r_i)$ and $h(i, m \oplus 1^n \oplus r_i)$. Imagine, the adversary

making queries related to $m \oplus 1^n$ followed by m. In such a situation, there is no index i, that is not queried by other points. If we resample at some point, the $\tilde{g}_R(m \oplus 1^n)$ will change. As a result we get into a cycle. Hence, the argument hits a roadblock.

We stress that the computation of $\tilde{g}_R(m)$ is a joint distribution of the output of several $h(i, x)$ queries. Thus the probability distribution of $\tilde{g}_R(m) \in E$ and $\tilde{g}_R(m') \in E$ for any event E can be related.

We identify many more issues with the proof of [16]. They require more technical setup for explanation. A technical analysis is given in Section 3.

New Statistical Tool. We start with a counting approach. We say a point $\alpha \in \{0,1\}^n$ is bad, if the probability (over h) of $\tilde{h}(\alpha)$ being subverted is high. A point is good if it is not bad. A point is vulnerable if either it is bad or it is (with significant probability over h) queried by a bad point during the subverted evaluation. A non vulnerable point is dangerous if it is queried by too many good points. By averaging argument, we show that the size of the union of vulnerable or dangerous set is small. Hence, for overwhelming fraction of candidate h, R, for every message m, there will exist an index i such that $m \oplus r_i$ is neither vulnerable nor dangerous.

What do we gain by this? Now, we can say that even though $h(i, m \oplus r_i)$ was queried by other points, they are non-vulnerable. Hence, we can argue that, if we resample at $(i, m \oplus r_i)$ the values of those non-vulnerable points will not change. Hence, the issue with the previous approach gets taken care of.

Now, we could find a rejection resampling lemma on two or more points, and argue the uniformity of $\tilde{g}_R(m)$. However, we simplify things further. We observe that the with high probability over the output values of $h(i, m \oplus r_i)$, the transcript of the previous internal queries remain unchanged. Hence, we consider the conditional probabilities by conditioning on all possible transcripts, and take union bound to show near uniformity of $\tilde{g}_R(m)$. In Section 4 we state the things in detail.

Proof with Elementary Technique: Avoid Subversion via Randomness. In case of classical indifferentiability, the distinguisher aims to distinguish between (C^f, f) and $(\mathcal{F}, S^f)$. In crooked-indifferentiability, C has oracle access to the (subverted) implementation $\tilde{f}$. In order to gain any advantage from the subversion, the distinguisher must produce some message M such that at least one of the f queries made during the computation of $C^f(M)$ becomes subverted. In case of Enveloped XOR construction, the adversary can choose a message forcing this. All the heavy technical machineries developed and deployed in the last section, to handle such scenarios, showing indifferentiability even when some f queries are subverted. But what about a construction, where the adversary cannot force the subversion. Specifically, consider the sponge construction with random initialization vector. Our construction takes (padded) messages of length ℓr bit and produces output of hr many bits. By the definition of crooked-indifferentiability, probability that $\tilde{f}(R)$ is subverted, for a randomly chosen R, is ϵ. By union bound, the probability that for some $m_0 \in \{0,1\}^r$, $\tilde{f}\left(R \oplus (m \| 0^c)\right)$ is not equal to $f\left(R \oplus (m \| 0^c)\right)$ is at most $\epsilon 2^r$. Conditioned on the input being

"non-subverted", the output of $\tilde{f}$ is independently and uniformly distributed. Hence, we get a uniform random chaining value. Repeating the argument,and taking union bound, we get the following. For any message M, probability that one of the $h+\ell$ many queries made by C^f is subverted, is at most $(h+\ell)\epsilon 2^r$. Taking union bound over all the queries made by the distinguisher, the probability becomes bounded by $q(h+\ell)\epsilon 2^r$.

2 Notations and Preliminaries

NOTATIONS. For any tuples of pairs $\tau = ((x_1, y_1), \ldots, (x_s, y_s))$ we write $\mathcal{D}(\tau)$ (called domain of τ) to denote the set $\{x_i : 1 \leq i \leq s\}$. We write $x \xleftarrow{\$} S$ to denote the process of choosing x uniformly at random from a set S and independently from all other random variables defined so far.

CLASS OF FUNCTIONS. Let l be a positive integer. We use $(l]$ to denote the set of first l natural numbers. We also use $[l]$ is used to denote the set of whole numbers up to l, i.e. $\{0, 1, \ldots, l\}$.

The positive integer n is our security parameter. Let $\mathcal{D} := [l] \times \{0,1\}^n$. Let H and F denote the set of all n-bit valued functions from $\mathcal{D}$ and $\{0,1\}^n$ respectively. For any $z := ((a_1, b_1), \ldots, (a_{q_1}, b_{q_1}))$ with $b_1, \ldots, b_{q_1} \in \{0,1\}^n$ and distinct $a_1, \ldots, a_{q_1} \in \mathcal{D}$, we write $h \vdash z$ if $h(a_i) = b_i$ for all $i \in (q_1]$. We denote the set of all functions h such that $h \vdash z$ as $\mathsf{H}|_z$.

DISTINGUISHING ADVANTAGE. In this paper, we measure the efficiency of algorithms by the number of queries it make. An oracle algorithm $\mathcal{A}$ having access of one oracle is called q-query algorithm if it makes at most q queries to its oracle. Similarly, an oracle algorithm having access of two oracles is called (q_1, q_2)-query algorithm if it makes at most q_1 and q_2 queries to its first and second oracles respectively. We use the short-hand notation X^t to denote the tuple $(X_1, \ldots, X_t)$.

Definition 1 (Distinguishing Advantage). *Let $F^a := (F_1, F_2, \cdots, F_a)$ and $G^a := (G_1, G_2, \cdots, G_a)$ be tuple of some probabilistic oracle algorithms for some positive integer a. We define* advantage *of an adversary $\mathcal{A}$ at distinguishing F^a from G^a as*

$$\Delta_{\mathcal{A}}(F^a \ ; \ G^a) = \left|\Pr[\mathcal{A}^{F_1, F_2, \cdots, F_a} = 1] - \Pr[\mathcal{A}^{G_1, G_2, \cdots, G_a} = 1]\right|.$$

2.1 Classical Indifferentiability

IV-based oracle construction $C^{\mathcal{O}}(\cdot, \cdot)$ first fixes an initial value R (chosen randomly from an initial value space). Afterwards, on input M, it interacts with the oracle $\mathcal{O}$ and finally it returns an output, denoted as $C^{\mathcal{O}}(R, M)$. When the initial value space is singleton (i.e., degenerated), we simply call C an oracle construction. *An (initial value based) oracle construction C is called $\mathcal{F}$-compatible* if the domains and ranges of C and $\mathcal{F}$ (an ideal primitive) are same. Now we state the definition of indifferentiability of an oracle construction as stated in [7,10] in our terminologies.

Definition 2 (Indifferentiability). *Let $\mathcal{F}$ be an ideal primitive and C^P be a $\mathcal{F}$-compatible oracle construction. C is said to be* $((q_P, q_C, q_{\text{sim}}), \varepsilon)$ indifferentiable *from an ideal primitive $\mathcal{F}$ if there exists a q_{sim}-query algorithm $S^{\mathcal{F}}$ (called simulator) such that for any (q_P, q_C)-query algorithm $\mathcal{A}$, it holds that*

$$\Delta_{\mathcal{A}}\left((P, C^P(\cdot)) \; ; \; (S^{\mathcal{F}}(\cdot), \mathcal{F})\right) < \varepsilon.$$

In the above definition one may include the complexity (time, query etc.) of the adversary and simulator. However, for information theoretic security analysis, we may ignore the time complexity of the simulator as well as the adversary.[3] A popular indifferentiability treatment for hash function considers $\mathcal{F}$ to be a n-bit random oracle which returns independent and uniform n-bit strings for every distinct queries. However, the hash function C^P can be defined through different types of primitives P (a random oracle, or a random permutation π_n, chosen uniformly from the set of all permutations over $\{0,1\}^n$).

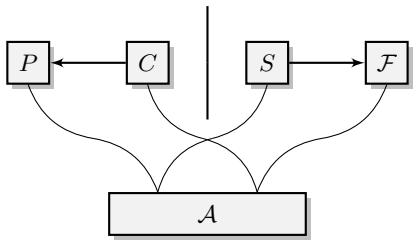

Fig. 1. The distinguishing game of $\mathcal{A}$ in the indifferentiability security game.

2.2 Crooked Indifferentiability

We recall the related terms and notations introduced in [16] in our terminologies.

Subverted Implementor. Let $\mathcal{A}_1^{\mathcal{O}}(r)$ be a q-query probabilistic algorithm (r denotes its random coin) and z (*advise string*) denote tuple of query-responses. Suppose $\mathcal{A}_1$ returns $H := H_{z,r}$ which is also a code of an oracle algorithm. Note that the oracle for H need not be same as that of $\mathcal{A}_1$. If the both oracles are same and chosen from a collection of functions H, we call $\mathcal{A}_1$ a (q, τ) *subverted implementor for* H. Note that a correct implementation should output H so that for all $h \in \mathsf{H}$ and for all $x \in \mathcal{D}(h)$, $\tilde{h}(x) := H^h_{z,r}(x) = h(x)$. However, we are

[3] One can easily extend the concrete setup to an asymptotic setup. Let $\langle \mathcal{F}_n, P_n \rangle_{n \in \mathbb{N}}$ be a sequence of primitives and $C(n)$ be a polynomial time $\mathcal{F}_n$-compatible oracle algorithm. $C^{P_n}(n)$ is said to be (computationally) indifferentiable from $\mathcal{F}_n$ if there exists a polynomial time simulator $S^{\mathcal{F}_n}$ such that for all polynomial time oracle algorithm $\mathcal{A}$, $\Delta_{\mathcal{A}}\left((P_n, C^{P_n}(n)) \; ; \; (S^{\mathcal{F}_n}, \mathcal{F}_n)\right) = \mathsf{negl}(n)$.

concerned on those adversarial implementation in which some incorrect outputs can be present.

Convention on Subverted Implementor. Without loss of generality, we will make the following reasonable assumptions on the subverted implementation H.

1. H makes exactly τ queries (as it can always make some additional queries).
2. Let $\tilde{x}_i := \mathcal{Q}_i^h(x)$ denote the ith query during the execution of $H_{z,r}^h(x)$. We assume that
   - $\tilde{x}_1 = x$ and
   - $\tilde{x}_i$'s are distinct and different from $\mathcal{D}(z)$ (i.e. not queried by subverted implementor $\mathcal{A}_1$).

Query Notations. Let $\mathcal{Q}^h(\alpha)$ denote the set of all queries during the computation of $H^h(\alpha)$. We use $\alpha \twoheadrightarrow_h \alpha'$ to denote that $\tilde{h}(\alpha)$ queries $h(\alpha')$. Similarly, $\alpha \not\twoheadrightarrow_h \alpha'$, denotes that $\tilde{h}(\alpha)$ does not query $h(\alpha')$. $\mathcal{Q}_j^h(\alpha)$ (alternatively $\tilde{\alpha}_j$ when the function is clear from the context) denotes the j^{th} query made by $\tilde{h}(\alpha)$.

Definition 3. *A (q, τ) subverted implementor $\mathcal{A}_1$ for H is called* ϵ-crooked *if for every $h \in \mathsf{H}$ and for all $0 \le i \le l$, $\Pr_\alpha(\tilde{h}(i, \alpha) \neq h(i, \alpha)) \le \epsilon$ where $\alpha \xleftarrow{\$} \{0,1\}^n$.*

Detection Algorithm. Given an implementation one may check the correctness of the algorithm by comparing the outputs of the implementation with a known correct algorithm. More precisely, we sample $\alpha_1, \ldots, \alpha_t \xleftarrow{\$} \{0,1\}^m$ and then for all $0 \le i \le l$, we check whether $\tilde{h}(i, \alpha) = h(i, \alpha)$ holds. If it does not hold, the implementation would not be used. It is easy to see that for ϵ-crooked implementation the subversion would not be detected with probability at least $(1-\epsilon)^t$. So for negligible ϵ this probability would be still close to one for all polynomial function t and so the implementation can be survived for further use.

Crooked Distinguisher. Now we define a two stage adversary. The first stage is a subverted implementor and the second stage is a distinguisher.

Definition 4 (crooked distinguisher). *We say that a pair $\mathcal{A} := (\mathcal{A}_1, \mathcal{A}_2)$ of probabilistic algorithms* $((q_1, \tau, \epsilon), q_2)$-crooked distinguisher *if*

(i) $\mathcal{A}_1(r)$ is a ϵ-crooked (q_1, τ) subverted implementor for H and

(ii) $\mathcal{A}_2(r, z, \cdot)$ is a q_2-query distinguisher where z is an advise string (transcript of $\mathcal{A}_1$ with its oracle).

Note that $\mathcal{A}_1$ and $\mathcal{A}_2$ can also be viewed as one stateful probabilistic algorithm $\mathcal{A}$ and the internal state and the advise string would be implicitly conveyed from one stage to the other.

Crooked Indifferentiability. Now we state H-crooked indifferentiable security definition (as introduced in [16]) in our notation and terminology.

Definition 5 (H-crooked indifferentiability [16]). *Let $\mathcal{F}$ be an ideal primitive and C be an initial value based $\mathcal{F}$-compatible oracle construction. The construction C is said to be*

$((q_1, \tau), (q_2, q_{\mathrm{sim}}), \epsilon, \delta)$-*H-crooked indifferentiable from* $\mathcal{F}$

if there is a q_{sim}*-query algorithm* S *(called simulator) such that for all* $((\epsilon, q_1, \tau), q_2)$*-crooked distinguisher* $(\mathcal{A}_1(r), \mathcal{A}_2(r, \cdot, \cdot))$*, we have*

$$\Delta_{\mathcal{A}_2(r,z,R)}\big((h, C^{\tilde{h}}(R, \cdot)) \; ; \; (S^{\mathcal{F}}(H_{z,r}, z, R), \mathcal{F})\big) \leq \delta \tag{1}$$

where z *is the advise string of* $\mathcal{A}_1^h$ *and* R *is the random initial value of the construction sampled after subverted implementation is set.*

Two-Stage Distinguishing Game. Now we explain the distinguishing game. In the first stage a crooked distinguisher determines a code of an oracle algorithm $H := H_{z,r}$ after interacting with a random oracle h. Then, a random initial value R of the hash construction C is sampled. In the real world, $\mathcal{A}_2$ interacts with the same h of the firsts stage and the hash function $C^{\tilde{h}}(R, \cdot)$ (based on the subverted implementation using the initial value R). In ideal world, there exists a simulator S which gets the advise string z, the initial value R and the code of the subverted implementation H as inputs,[4] and gets oracle access of a random oracle $\mathcal{F}$. Simulator is aimed to simulate h so that behavior of $(h, C^{\tilde{h}})$ is as close as $(S, \mathcal{F})$ to the distinguisher $\mathcal{A}_2$. If so, the hash function, even with the help of subverted implementation, behaves like a random oracle.

Convention on Crooked Distinguishers: Note that there is no loss to assume that both $\mathcal{A}_1$ and $\mathcal{A}_2$ are deterministic (so we skip the notation r) when we consider computational unbounded adversary[5]. We also assume that $\mathcal{A}_2$ makes all distinct queries and distinct from the queries made by $\mathcal{A}_1$. We sometimes skip the notation z as an input of $\mathcal{A}_2$ as it is conveyed implicitly.

2.3 Markov Inequality

Lemma 6. *Let* X *be a non-negative random variable and* $a > 0$ *be a real number. Then it holds that*

$$\Pr[X \geq a] \leq \frac{\mathsf{Ex}(X)}{a}$$

A simple application of Markov inequality (which is used repeatedly in this paper) is the following. Consider a joint distribution of random variables X and Y. Suppose E is an event for which $\Pr((X, Y) \in E) \leq \epsilon$. Let $f(x) := \Pr((X, Y) \in E | X = x)$ and $E_1 := \{x : f(x) \geq \delta\}$. It follows from the definition that $\mathsf{Ex}(f(X)) = \Pr(E)$. Now, we use Markov's inequality

$$\begin{aligned} \Pr(E_1) &= \Pr(f(X) \geq \delta) \\ &\leq \mathsf{Ex}(f(X))/\delta \\ &= \epsilon/\delta. \end{aligned}$$

Note that when X and Y are independent, $f(x) = \Pr((x, Y) \in E)$

[4] If an algorithm S take a code of an oracle algorithm H as an input, it can compute $H(x)$ by running the code of H and responding the oracle queries of H.

[5] $\mathcal{A}_1$ can fix the best random coin for which the distinguishing advantage of $\mathcal{A}_2$ is maximum.

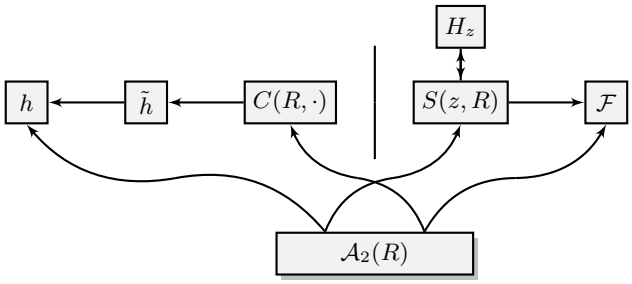

Fig. 2. The crooked indifferentiability notion. In the first phase of real world, $\mathcal{A}_1$ interacts with f and returns an oracle algorithm $\tilde{f}$ (which would be accessed by the construction C in the second phase). In the second phase the random initial value R will be sampled and given it to the construction C and also to $\mathcal{A}_2$. In ideal world, simulator $S^{\mathcal{F}}$ gets the advise string of the first phase, subverted implementation H_z and the initial value R.

3 Limitations of Techniques in [16]

In this section we revisit the crooked indifferentiability security analysis of EXor given in [16] and discuss the issues in it.

Enveloped XOR Construction or EXor: We fix two positive integers n and l. Let $\mathcal{D} := [l] \times \{0,1\}^n$. Let H be the class of all functions $h : \mathcal{D} \to \{0,1\}^n$. For every $x \in \{0,1\}^n$ and an initial value $R := (r_1, \ldots, r_l) \in (\{0,1\}^n)^l$, we define

$$g_R(x) = \bigoplus_{i=1}^{l} h(i, x \oplus r_i) \quad \text{and} \quad \mathsf{EXor}(R,x) = h(0, g_R(x)).$$

(**Theorem 3** from [16]). Let $l \geq 3n+1$ and $h \xleftarrow{\$} \mathsf{H}$. Then, the enveloped XOR construction is $((q_1,\tau),(q_2,q_{\text{sim}}),\epsilon,\epsilon')$-$\mathsf{H}$-crooked indifferentiable from n-bit random oracle $\mathcal{F}$ where $q_1, \tau, q_{\text{sim}}, q_2$ are polynomial function of n and ϵ, ϵ' are negligible function of n.

Convention on Second Stage Adversary. We assume the following for the second stage adversary $\mathcal{A}_2$.

1. (Batch query): It makes queries $h(i,x)$ for $i > 0$, it also makes queries $h(j, x \oplus r_i \oplus r_j)$ for all $j > 0$ simultaneously. In other words, it makes a batch query of the form $(h(j, m \oplus r_j))_{1 \leq j \leq l}$ for some $m \in \{0,1\}^n$. We simply say that $\mathcal{A}_2$ queries m to g_R and obtains responses $(h(j, m \oplus r_j))_{1 \leq j \leq l}$.
2. There is no loss to assume that it queries m to $\mathcal{F}$ before it queries to g_R oracle.

We fix any $z := ((a_1, b_1), \ldots, (a_{q_1}, b_{q_1}))$ with $b_1, \ldots, b_{q_1} \in \{0,1\}^n$ and distinct $a_1, \ldots, a_{q_1} \in \mathcal{D}$. Throughout this paper, we only consider function h from $\mathsf{H}|_z$ and hence notation for the advise string z would be mostly skipped.

3.1 Description of Simulator

We now describe the simulator defined in [16]. We would like to note that the we consider the same simulator in our proof and so we reuse the notations developed here in the subsequent sections. We define

$$\tilde{g}_R(m) := \bigoplus_{i=1}^{l} \tilde{h}(i, m \oplus \mathsf{r}_i) \text{ and } \widetilde{\mathsf{EXor}}(R, x) = \tilde{h}(0, \tilde{g}_R(x)).$$

In other words, if we express $\mathsf{EXor}(R, x)$ as $\mathsf{EXor}^h(R, x)$ then $\widetilde{\mathsf{EXor}}(R, x)$ represents $\mathsf{EXor}^{\tilde{h}}(R, x)$ which is one of the oracle in the real world for crooked indistinguishable game.

Simulator: We first quickly recall how the simulator $S^{\mathcal{F}}(H, z, R, \cdot)$ is defined. Note that main goal of the simulator is to simulate $h \xleftarrow{\$} \mathsf{H}$ as honestly[6] as possible such that $\widetilde{\mathsf{EXor}}(R, m) = \mathcal{F}(m)$ for all queried m. The simulator maintains a list L of pairs (α, β) to record $h(\alpha) = \beta$ for $\alpha \in \mathcal{D}$ and $\beta \in \{0,1\}^n$. It also maintains a sublist L^A of L consisting of all those pairs which are known to the distinguisher. Both lists are initialized to z (the advise string in the first stage which we fix to any tuple of q_1 pairs). Now we describe how the simulator responds.

1. (Type-1 Query): All queries of the form $h(0, w)$ returned honestly and updates the list L and L^A.
2. (Type-2 Query): For a query $g_R(m)$ (i.e. batch query) it computes $\tilde{h}(\alpha_j)$ for all j, one by one by executing the subverted implementation H, where $\alpha_j = (j, m \oplus R_j)$. During this execution, simulator responds honestly to all queries made by the subverted implementation and updates the L-list by incorporating all query responses of h. However, it updates L^A list only with $(\alpha_j, h(\alpha_j))$ for all j. Let $\tilde{g} := \bigoplus_j \tilde{h}(\alpha_j)$.

   Definition 7. *We say that the simulator* **aborts** *if* $(0, \tilde{g}) \in \mathcal{D}(L)$.

   If simulator does not abort, it makes a query $\mathcal{F}(m)$ and adds $((0, \tilde{g}), \mathcal{F}(m))$ into the both lists L and L^A.

Note that behavior of S is exactly same as random oracle possibly except those queries $h(0, w)$ made by $\mathcal{A}_2$ where $w = \tilde{g}_R(m)$ for some previous query m to g_R. In this case, it is answered as $\mathcal{F}(m)$.

Cautionary Note. Even though $\mathcal{F}$ is a random oracle, we cannot say that probability distribution of the response of $(0, \tilde{g})$ in the ideal world is uniform. Note that adversary can choose m after making several consultation with $\mathcal{F}$. In other words, m can be dependent on $\mathcal{F}$. For example, adversary can choose m for

[6] By honestly, we mean that if the responses is already in the list it returns that value, otherwise it samples a fresh random response and includes the input and output pairs in the lists.

which the last bit of $\mathcal{F}(m)$ is zero and so the response for the query $(0, w)$ always has zero as the last bit (which clearly diverts from the uniform distribution).

Transcript: Now we describe what is the transcript to the distinguisher and for the simulator in more details.

1. Let L^F denote the set of all pairs $(m', \tilde{z})$ of query response of $\mathcal{F}$ by $\mathcal{A}_2$.
2. Let L^g denote the set of all pairs (m, β^l) of query response of g_R oracle (batch query) made by $\mathcal{A}_2$ to the simulator where $\beta^l := (\beta_1, \ldots, \beta_l)$ and $\beta_j = h(j, m \oplus R_j)$ for all j. According to our convention all these m must be queried to $\mathcal{F}$ beforehand.
3. As we described we also have two lists, namely L and its sublist L^A, keeping the query responses of h oracle.

Now we define the transcript and partial transcript as the queries of the distinguishers going on. We note that q_1 is the number of queries in the first stage and $\mathcal{A}_2$ is a (q_F, q_2)-query algorithm. For any $1 \leq i \leq q = q_2 + q_F)$, we define partial transcripts of $\mathcal{A}$ and simulator as $\tau_i^A := (L_i^F, L_i^A)$ and $\tau_i^S := (L_i, L_i^g)$ respectively, where L_i^F, L_i^A, L_i, L_i^g denote the contents of the corresponding lists just before making ith query of the distinguisher. So when, $i = 1$, $L_1^A = L_1 = z$ and the rest are empty and when $i = q+1$, these are the final lists of transcripts. Let $\tau_i := (\tau_i^A, \tau_i^S)$ and $\tau := (\tau^A, \tau^S)$ denote the joint transcript on ith query or after completion respectively. As the adversary is deterministic, simulator is also deterministic for a given h and $\mathcal{F}$, and we have fixed z, a (partial) transcript is completely determined by the choice of R, h and $\mathcal{F}$ (in the ideal world). We write $(R, F, h) \vdash \tau_i^S$ if the transcript τ_i^S is obtained when the initial value is R, the random oracles are F and h. We similarly define $(R, F, h) \vdash \tau_i^A$ and $(R, F, h) \vdash \tau_i$.

Bad Event. Let bad_i denote the event that ith query is m to g_R oracle and either simulator aborts (as defined in Definition 7) or $(0, \tilde{g}_R(m))$ is a crooked point for h. Let $\mathsf{bad} = \vee_i \mathsf{bad}_i$.

Observation 8 *Given that* bad *does not hold, for all queries m we have*

$$\widetilde{\mathsf{EXor}}(R, m) = \mathcal{F}(m).$$

3.2 Revisiting the Proof of [16]

For every query number i, we define a set $E_i := \mathcal{D}(L_i) \cup \mathcal{C}^h$ where $\mathcal{C}^h$ is the set of all crooked elements for h. It is easy to see from the definition that bad_i holds if and only if $(0, \tilde{g}_R(m_i)) \in E_i$ where m_i denotes the ith query of $\mathcal{A}$ (made to g_R oracle of the simulator). So, the crooked indifferentiable advantage is bounded by $\sum_{i=1}^{q_2} \Pr(\tilde{g}_R(m_i) \in E_i)$. From the definition it is clear that $|\mathcal{D}(L_i)| \leq q_1 + i$. Moreover, $|\mathcal{C}^h|/2^n$ is negligible for every fixed function h as $\mathcal{A}$ can crook at most negligible fraction of inputs. Motivated from the above, the authors wanted to

show that the distribution of $\tilde{g}_R(m_i)$ is almost uniform and hence the following core result was proved.

(**Theorem 5** from [16]). Let $h \xleftarrow{\$} \mathsf{H}|_z$. With overwhelming probability (i.e., one minus a negligible amount) there exists a set $\mathcal{R}_z \subseteq (\{0,1\}^n)^l$ and for every i, a set of transcripts $\mathcal{T}_i^A$ (before ith query) such that for all $R \in \mathcal{R}_z$, $\tau_i := (L_i^F, L_i^A) \in \mathcal{T}_i^A$, and $m \notin \mathcal{D}(L_i^g)$,

$$\Pr_h((0, \tilde{g}_{\mathsf{R}}(m)) \in E_i \mid (R, \mathcal{F}, h) \vdash \tau_i) \leq \mathrm{poly}(n)\sqrt{|E_i|} + \mathsf{negl}(n).$$

Theorem 3 of [16] can be derived from the above theorem. Now we describe the main steps of proving the above theorem. In the first step authors assumes $i = 1$ and later with a very sketchy argument justifies how it works for general i. Note that τ_1 basically contains only the information of z. As we sample $h \xleftarrow{\$} \mathsf{H}|_z$, we can ignore the conditional event and we can simply focus to bound the probability of the event $(0, \tilde{g}_{\mathsf{R}}(m)) \in \mathcal{D}(z) \cup \mathcal{C}^h$ for any m. To prove that we show that $\tilde{g}_{\mathsf{R}}(m)$ behaves close to the uniform distribution over $\{0,1\}^n$ and so the above probability is negligible as $q_1/2^n$ and $|\mathcal{C}^h|/2^n$ is negligible. Before we do so, we develop some notations. Let us define

$$d(\alpha, h) = \begin{cases} 1, & \text{if } \tilde{h}(\alpha) \neq h(\alpha) \text{ or } \alpha \in \mathcal{D}(z) \\ 0, & \text{otherwise} \end{cases}$$

In other words, d sets value one for an element $z \notin \mathcal{D}$ if it is crooked.[7] For every $\alpha \in \mathcal{D}, \beta \in \{0,1\}^n$, we define a function $h_{\alpha \to \beta}$ which agrees with h on all points possibly except at α at which the function $h_{\alpha \to \beta}$ maps to β. Note that if $h(\alpha) = \beta$ then $h_{\alpha \to \beta} = h$. Let $D^1(\alpha, h) = \mathsf{Ex}_\beta(d(\alpha, h_{\alpha \to \beta}))$ where the expectation is computed under $\beta \xleftarrow{\$} \{0,1\}^n$. By using Markov inequality, authors are able to identify a set of overwhelming amount of pairs (R, h), called *unpredictable*, such that for any unpredictable (R, h) and for all m, there exists an index i such that

1. $D^1(\alpha_i, h)$ is negligible and
2. $\alpha_j \not\rightarrow_h \alpha_i$ for all $j \neq i$, where $\alpha_j = m \oplus R_i$.

Thus, if we resample $\beta = h(\alpha_i)$ then with overwhelming probability $\tilde{h}_{\alpha_i \to \beta}(\alpha) = h_{\alpha_i \to \beta}(\alpha)$ (i.e. not crooked and returned a random value) and all corresponding values for indices j different from i will remain same. So, $\tilde{g}_R(m) = \beta + A$ where A does not depend on choice of β. Thus, the modified distribution is close to uniform (as almost all values of β will be good). In particular the authors made the following claim:

Claim 9 *Under the modified distribution (i.e. after resampling),* $\Pr(\tilde{g}_R(m) \in E_1) \leq q_1/2^n + \epsilon + p_n$ *where* p_n *denotes the probability that a random pair* (R, h) *is not unpredictable.*

[7] Note that $\alpha \in \mathcal{D}(z)$ was not considered in [16]. However, it will be later clear that we need this simple modification.

As the choice of i depends on the function h and so rejection resampling lemma (introduced in the same paper) is used to bound the probability of the event under the original distribution (i.e. before resampling).

Lemma 10 (Rejection Resampling [16]). *Let $X := (X_1, \ldots, X_k)$ be a random variable uniform on $\Omega = \Omega_1 \times \Omega_2 \times \cdots \times \Omega_k$. Let $A : \Omega \to (k]$ and define $Z = (Z_1, \ldots, Z_k)$ where $Z_i = A_i$ except at $j = A(X^k)$ for which Z_j is sampled uniformly and independently of remaining random variables. Then for any event $S \subseteq \Omega$, it holds that*

$$|S|/|\Omega| \leq \sqrt{k \Pr(Z \in S)}$$

With this rejection resampling result and the above claim, we can conclude the following under original distribution: $\Pr(\tilde{g}_R(x) \in E_1) \leq \sqrt{l \cdot (q_1/2^n + \epsilon + p_n)}$.

3.3 Issues with the approach

Now we are ready to describe the issues and the limitations of the techniques in [16]. To prove the general case (i.e. for any query), authors provides a very sketchy argument. It seems that authors argued that with an overwhelming probability of realizable transcript $\mathcal{T}$ and for all $\tau \in \mathcal{T}$, $\Pr(\tilde{g}_R(m_i) \in E_i \mid \tau)$ is negligible.

The bad event E_i depends on the function h. Recall that the rejection resampling helps to bound

$$\Pr_{h*}(\tilde{g}_R(x) \in E_1) \leq \sqrt{l \cdot \Pr_{\text{resampled } h}(\tilde{g}_R(x) \in E_1)}.$$

Moreover, authors show that $\Pr_{\text{resampled } h}(\tilde{g}_R(x) \in E_1)$ is small as $\tilde{g}_R(x)$ is uniformly distributed under resampling distribution of h and size of E_1 is negligibly small. But, the crooked set of h may depend on the function family h and so we cannot bound the probability of the event $\tilde{g}_R(x) \in E_1$ as $|E_1|/2^n$ as E_1 is not independent of $\tilde{g}_R(x)$. In other words, the Claim 9 need not be true. This is one of the crucial observation which actually makes the crooked security analysis a bit complex.

Controlling query dependencies for the same index. Recall that the resampling is done on an index i, that is honest ($\tilde{h}(i, m \oplus r_i) = h(i, m \oplus r_i)$. Moreover, $h(i, m \oplus r_i)$ is not queried by any other $\tilde{h}(j, m \oplus r_j)$. The authors proved that with high probability over R, h, for every message x such an index i will exist. Unfortunately, this condition (no query to an index i) does not hold if the adversary can check consistency for multiple messages.

Consider a (q, τ) subverted implementation $\tilde{h}$ that is ϵ-crooked. We construct a $(q, \tau+1)$ subverted implementation $\hat{h}$ as follows. For every possible input (i, x), $\hat{h}$ simulates $\tilde{h}$. After the τ many queries made by $\tilde{h}$, $\hat{h}$ makes an additional query on $h(i, x \oplus 1^n)$, if not already made. Finally $\hat{h}$ outputs what $\tilde{h}$ outputs. Clearly, $\hat{h}$ is ϵ-crooked. The distinguisher makes queries, first for message $m \oplus 1^n$, and

then for m. In case of message m, all the $h(i, m \oplus 1^n)$ has been queried by the simulator when computing $\tilde{g}_R(m)$. Thus one can not find an index i such that $h(i, x \oplus 1^n)$ has never been queried.

The gap in the technique. The reason, these issues have not cropped up in the proof is the fact, that the authors *did not prove* the simulator to be consistent. In particular, they did not show that the bad events they considered, are complete. In order to prove indifferentiability, it is required that conditioned on not Bad, the real and the ideal games are indistinguishable. While there are several techniques (like H-coefficient technique, or game playing technique), there is no formal argument on why Theorem 4 proves crooked indifferentiability.

The number of queries to $\mathcal{F}$ is essential. We give an example why their proof is incomplete as it does not consider the $\mathcal{F}$ queries of the distinguisher. The bound is almost vanishing if $q_1 = 0$ and $q_2 = 2$ and has no crooked point. However, a distinguishers search for $m \neq m'$ such that $\mathcal{F}(m) = \mathcal{F}(m')$ and hence $g_R(m) = g_R(m')$ holds with probability about $1/2$. However, for the honest simulation of all h values, g value will collide with very low probability. Thus, the the number of queries to $\mathcal{F}$ oracle should be involved in the bound.

4 Basic Tools for Crooked Analysis

We follow the notations developed in the previous sections. For $1 \leq j \leq \tau$, let $D^j(\alpha, h) = \mathsf{Ex}_\beta(d(\alpha, h_{\tilde{\alpha}_j \to \beta}))$ (average number of crooked point after we resample the output of the jth query made by the subverted implementation).

Lemma 11. *Let $\alpha \xleftarrow{\$} \mathcal{D}$, $h \xleftarrow{\$} \mathsf{H}|_z$. For any ϵ-crooked implementation H_z, for all $1 \leq j \leq \ell$*

$$\mathsf{Ex}_{\alpha,h}\big(D^j(\alpha, h)\big) \leq \epsilon_1 := (\epsilon + q_1 2^{-n}) \tag{2}$$

Proof. For any ϵ-crooked implementation H and every $h \in \mathsf{H}|_z$, $0 \leq i \leq l$, we have $\Pr_x(\tilde{h}(i, x) \neq h(i, x)) \leq \epsilon$ where $x \xleftarrow{\$} \{0,1\}^n$. So, for $\alpha \xleftarrow{\$} \mathcal{D}$,

$$\begin{aligned}\mathsf{Ex}_\alpha(d(\alpha, h)) &= \Pr_\alpha(\tilde{h}(\alpha) \neq h(\alpha) \vee \alpha \in \mathcal{D}(z)) \\ &\leq \epsilon + q_1 2^{-n}.\end{aligned}$$

Now, we fix any $\alpha := (i, x)$ and $1 \leq j \leq \tau$. For any function $g \in \mathsf{H}|_z$, let $\mathcal{S}_{\alpha,g} := \{(f, \beta) \in \mathsf{H}_z \times \{0,1\}^n : \ f|_{\mathcal{Q}_j^f(\alpha) \to \beta} = g\}$.

Claim 12 *For a function g, we have $|\{(f, \beta) : \ f|_{\mathcal{Q}_j^f(\alpha) \to \beta} = g\}| = N$.*

Proof of Claim. Suppose $j = 1$ and so $\mathcal{Q}_j^f(\alpha) = \alpha$. Now, the function f agrees with g except that the output of f at α can be any n-bit string, which should be β. Now assume $j > 1$. So, $f(\mathcal{Q}_k^f(\alpha)) = g(\mathcal{Q}_k^f(\alpha))$ for all $k < j$. For $k = j$,

$f(\mathcal{Q}_j^f(\alpha)) = \beta$ (for any choice of $\beta \in \{0,1\}^n$). For all other inputs, f agrees with g. So the claim follows.

BACK TO PROVING LEMMA 11Now for each $j \in \{1, 2, \cdots, \ell\}$,

$$\begin{aligned}
\mathsf{Ex}_{\alpha,h} D^j(\alpha, h) &= \mathsf{Ex}_{\alpha,h}\mathsf{Ex}_{\beta}\big(d(\alpha, h|_{\tilde{\alpha}_j \to \beta})\big) \\
&= \sum_{\alpha,h,\beta} \Pr(h)\Pr(\alpha)\Pr(\beta) \cdot d(\alpha, h|_{\tilde{\alpha}_j \to \beta}) \\
&= 2^{-n} \sum_{(h,\beta)\in\mathcal{S}_{\alpha,g}} \sum_{\alpha,g} \Pr(g)\Pr(\alpha) \cdot d(\alpha, g) \\
&= \sum_{\alpha,g} \Pr(g)\Pr(\alpha) \cdot d(\alpha, g) \\
&= \mathsf{Ex}_{\alpha,g} d(\alpha, g) \\
&\leq \mathsf{Ex}_g(\epsilon + q_1 2^{-n}) \\
&\leq (\epsilon + q_1 2^{-n}) \qquad \square
\end{aligned}$$

Definition 13. *We call a pair (α', h) good if*

1. *For all $\alpha \twoheadrightarrow_h \alpha'$, we have $d(\alpha, h) = 0$ and $D^j(\alpha, h) \leq \epsilon_1^{1/2}$, where α' was the j^{th} query of $\tilde{h}(\alpha)$.*
2. *the number of α which queries α' is at most $1/\epsilon_1^{1/4}$.*

We call the pair (α', h) bad, if it is not good. Let G be the collection of all such good pairs. For any function h, let $G_h := \{\alpha' : (\alpha', h) \in G\}$ and $B_h := \{\alpha' : (\alpha', h) \notin G\}$.

In what follows we use $\alpha, \alpha' \xleftarrow{\$} \mathcal{D}$ and $h \xleftarrow{\$} \mathsf{H}|_z$ whenever these are used as random variables to compute probabilities. Otherwise, these are considered to be fixed elements from their respective domains.

Lemma 14. $\Pr_{\alpha',h}((\alpha', h)$ *is bad*$) \leq \epsilon_2 := 3\tau\epsilon_1^{1/4}$

Proof. A pair can be bad in two ways and we bound each case separately. We first bound that for a random α and h, either $d(\alpha, h) = 1$ or $D^j(\alpha, h) > \epsilon_1^{1/2}$. The probability of the first event is clearly bounded by ϵ and whereas the probability of the second event is bounded by $\epsilon_1^{1/4}$ (Applying Markov inequality and the Lemma 11). So a random α' is queried by some α satisfying the above event holds with probability at most $\tau(\epsilon + \epsilon_1^{1/4}) \leq 2\tau\epsilon^{1/4}$.

By using simple averaging argument, average number of α' for which the number of α that queries α' is at least $1/\epsilon^{1/4}$ is at most $\tau\epsilon_1^{1/4}$. By adding this two cases we complete our proof. $\square$

A function h is said to be good if $\Pr_{\alpha'}(\alpha' \in B_h) \leq \epsilon_2^{1/2}$, otherwise it is called bad. A simple application of Markov inequality proves that

$$\Pr_h(h \text{ is bad}) \leq \epsilon_2^{1/2} \tag{3}$$

Our next step is to construct a collection of good pairs (R, h). We construct a set $\mathcal{G}^*$ where a pair $(R, h) \in \mathcal{G}^*$ if

1. h is good.
2. for every m there exists i so that
   (a) $(\alpha_i := (i, m \oplus R_i), h)$ is good.
   (b) $\forall\ j < i$, $\alpha_j \not\twoheadrightarrow \alpha_i$.
   We call the smallest index i that satisfies the above two condition, "index of interest" for (m, R, h).

Lemma 15. *Let $\epsilon_2 \leq 1/16$ and $\ell \geq n$. It holds that*

$$\Pr_{R,h}((R, h) \notin \mathcal{G}^*) \leq p_1 := \epsilon_2^{1/2} + 2^{-n}$$

Proof. We know that $\Pr_h(h \text{ is not good}) \leq \epsilon_2^{1/2}$. Now, fix a good h. Then

$$\begin{aligned}\Pr_R((R, h) \notin \mathcal{G}^*) &\leq \sum_{m \in \{0,1\}^n} \prod_{i=1}^{l} \Pr_{R_i}(((i, m \oplus R_i), h) \text{ is not good}) \\ &\leq 2^n \times \epsilon_2^{l/2} \\ &\leq 1/2^n.\end{aligned}$$

Hence, we have

$$\begin{aligned}\Pr_{R,h}((R, h) \notin \mathcal{G}^*) &\leq \Pr_h(h \text{ is not good}) + \Pr_R((R, h) \notin \mathcal{G}^* | h \text{ is good}) \\ &\leq \epsilon_2^{1/2} + 1/2^n. \qquad \square\end{aligned}$$

Now we quickly summarize what we have proved so far. We have identified a set $\mathcal{G}^*$ from which the pair of initial value R and the random oracle h is sampled with overwhelming probability satisfying the following nice condition:

> For every fixed $(R, h) \in \mathcal{G}^*$ and for every m there exists the index of interest, i such that (α_i, h) is good where $\alpha_i = (i, m \oplus R_i)$.

Resampling. Our objective is to show that a transcript remains unchanged when $h(\alpha_i)$ is resampled. For that, our next step is to show the following. We can identify a set $\mathcal{S}$ (of size close to 2^n) such that for all $\beta \in \mathcal{S}$ and $h_\beta := h_{\alpha_i \to \beta}$, we have $\tilde{h}(x) = \tilde{h_\beta}(x)$ for all $x \neq \alpha_i$. We define $\mathcal{S} = \{\beta : d(\alpha, h_{\tilde{\alpha}_j \to \beta}) = 0 \forall\ j, \forall\ \alpha \in \mathcal{Q}^h_{\twoheadrightarrow \alpha_i}\}$ where $\mathcal{Q}^h_{\twoheadrightarrow \alpha_i} := \{\alpha : \alpha \twoheadrightarrow \alpha_i\}$.

Lower Bounding the size of S. As α_i is good, the size of the set $\mathcal{Q}^h_{\twoheadrightarrow \alpha_i}$ is at most $1/\epsilon^{1/4}$. For every such α, from the definition of good (α, h), we have $d(\alpha, h) = 0$ and $\tilde{D}(\alpha, h) \leq \epsilon_1^{1/2}$ (and so there are at most $2^n \times \epsilon_1^{1/2}$ many β for which there exists some j with $d(\alpha, h_{\tilde{\alpha}_j \to \beta}) = 1$). From the above discussion and using union bound for all α that queries α_i, the size of the set $\{0,1\}^n \setminus \mathcal{S}$ is at most $\frac{2^n \epsilon_1^{1/2}}{\epsilon_1^{1/4}} = 2^n \cdot \epsilon_1^{1/4}$.

Observation 16 $\tilde{h}(x) = \tilde{h_\beta}(x)\ \forall x \neq \alpha_i$.

Clearly, $\tilde{h_\beta}(x)$ can be different from $\tilde{h}(x)$, only if $x \twoheadrightarrow \alpha_i$. However, for all such α_i and for all $\beta \in \mathcal{S}$, we have shown that both $d(\alpha, h) = d(\alpha, h_\beta) = 0$. Hence, $\tilde{h_\beta}(\alpha) = h_\beta(\alpha) = h(\alpha) = \tilde{h}(x)$.

So from Eq.16, it is easy to see that (1) for all $m' \neq m$, $\tilde{g}_R^h(m') = \tilde{g}_R^{h_\beta}(m')$ and (2) $d((0,g),h) = d((0,g),h_\beta)$ (i.e. the crooked set of h_β for zero index is same for h for all such β).

Uniformity conditioned on transcript. The last step is to ensure that the point of resampling is independent of β. In other words, we need to show that the index of interest is also independent of β. Thus, for every message m, we can identify an index i such that $\tilde{g}_R^{h_\beta}(m) = \beta \oplus \tilde{g}_R^h(m) \oplus h(\alpha_i)$ holds for the fixed transcript.

Index of Interest is independent of β. Fix a good (R, h). Fix a message m. Let i be the index of interest. In this section, we show that i is also the index of interest for (R, h_β) and message m. Let $\alpha_i = (i, m \oplus R_i)$. We need to show three things.

1. $\alpha_i \notin B_{h_\beta}$.
2. for all $j < i$, $\alpha_j \not\twoheadrightarrow \alpha_i$ where $\alpha_j = (j, m \oplus R_j)$.
3. i is minimum index that satisfies the above two condition for (R, h_β) and m.

We start with the following observation.

Observation 17

$$Q^h_{\twoheadrightarrow \alpha_i} = Q^{h_\beta}_{\twoheadrightarrow \alpha_i}$$

Proof. Let $\alpha' \twoheadrightarrow_h \alpha_i$. Suppose α_i is the j^{th} query in the computation of $\tilde{h}(\alpha')$. Clearly, j^{th} query in the computation of $\tilde{h}(\alpha')$ depends on the query and responses of first $j-1$ queries made by $\tilde{h}(\alpha')$. Let $\tilde{\alpha}_1, \cdots, \tilde{\alpha}_{j-1}$ be those queries. As $\alpha_i \notin \{\tilde{\alpha}_1, \cdots, \tilde{\alpha}_{j-1}\}$, we get $h(\tilde{\alpha}_k) = h_\beta(\tilde{\alpha}_k)$. As all the previous query responses are same, $\alpha' \twoheadrightarrow_{h_\beta} \alpha_i$. □

We are now ready to prove the required three statements. Assume that i is the index of interest for (R, h) and m. by definition $\alpha_i \notin B_h$.

Claim 18 *$\alpha_i \notin B_{h_\beta}$ if and only if $\alpha_i \notin B_h$.*

Proof. As $\beta \in S$, $d(\alpha_i, h_\beta) = d(\alpha_i, h)$. By Observation 17,$Q^h_{\twoheadrightarrow \alpha_i} = Q^{h_\beta}_{\twoheadrightarrow \alpha_i}$. So, only things that require proofs are the following. First, for each k it should hold that, α_i is the k^{th} query of $\tilde{h}(\alpha)$ if and only if α_i is also the k^{th} query of $\tilde{h_\beta}(\alpha)$. Moreover, it needs to hold that $D^k(\alpha, h) = D^k(\alpha, h_\beta)$. The first statement follows from the following observation. As all the previous $k-1$ query responses are same in both cases, the k^{th} query made by the implementation on α will be

same. Hence α_i is the k^{th} query of $\tilde{h}(\alpha)$, if and only if α_i is also the k^{th} query of $\tilde{h_\beta}(\alpha)$. The second statement follows from the definition.

$$D^k(\alpha, h) = \mathsf{Ex}_{\beta'}(d(\alpha, h_{\alpha_i \to \beta'})) = D^k(\alpha, h_\beta)$$

□

The next claim proves the second statement

Claim 19 *For all $j < i$, $\alpha_j \not\twoheadrightarrow_{h_\beta} \alpha_i$ where $\alpha_j = (j, m \oplus R_j)$.*

Proof. Fix a $j < i$. We know $\alpha_j \not\twoheadrightarrow_h \alpha_i$. This implies $\alpha_j \notin Q^h_{\to \alpha_i}$. By Observation 17, $\alpha_j \notin Q^{h_\beta}_{\to \alpha_i}$. Hence, $\alpha_j \not\twoheadrightarrow_{h_\beta} \alpha_i$. □

The final step is to prove that i is the minimum such index even for (R, h_β) that satisfies first two points. For that we prove the following,

Claim 20 *If there exists and index $i' < i$ such that $\alpha_{i'} \notin B_{h_\beta}$ and $\forall\ j < i' \alpha_j \not\twoheadrightarrow_{h_\beta} \alpha_{i'}$, then it holds that $\alpha_{i'} \notin B_h$ and $\forall\ j < i', \alpha_j \not\twoheadrightarrow_h \alpha_{i'}$*

Proof. Using Claim 18, we get

$$\alpha_{i'} \notin B_{h_\beta} \implies \alpha_{i'} \notin B_h$$

For the second part, we show the contrapositive. Suppose for some $j < i'$, $\alpha_j \twoheadrightarrow_h \alpha_{i'}$. Hence $\alpha_j \in Q^h_{\to \alpha_{i'}}$. Moreover from assumption $\alpha_j \not\twoheadrightarrow_h \alpha_i$. All the queries made by $\tilde{h}(\alpha_j)$ and the responses remain same in $\tilde{h_\beta}(\alpha_j)$. This implies $\alpha_j \twoheadrightarrow_{h_\beta} \alpha_{i'}$. □

Finally, we are ready to state the main proposition

Proposition 21. *For any partial transcript for adversary $\tau_j := (L^F, L^A)$, let $(R, h, F) \vdash \tau_j$ such that $(R, h) \in \mathcal{G}^*$. For every $m \notin \mathcal{D}(L^g)$, there is a set $\mathcal{S}$ of size at least $2^n(1 - \epsilon_1^{1/2})$ such that for all $\beta \in \mathcal{S}$, $(R, h_\beta, F) \vdash \tau_j$ and $\tilde{g}_R^{h_\beta}(m) = \beta \oplus \tilde{g}_R^h(m) \oplus h(\alpha_i)$, where i is the index of interest of (m, R, h). Moreover, $C_0^{h_\beta} = C_0^h$ (the crooked set are same).*

Assuming $\epsilon_1^{1/2} \leq 1/2$, and for all $(R, h, F) \vdash \tau$ with $(R, h) \in \mathcal{G}^*$, we have

$$\Pr_\beta((0, \tilde{g}_R^{h_\beta}(m)) \in \mathcal{D}(L_0) \cup C^{h_\beta} \wedge (R, h_\beta, F) \vdash \tau) \leq 2\epsilon + 2(q_1 + i)/2^n$$

Proof. We let B_j denote the event of the choice of $h(\alpha_i)$ such that $(0, \tilde{g}_R^{h_\beta}(m)) \in \mathcal{D}(L_0) \cup C^{h_\beta} \wedge (R, h_\beta, F) \vdash \tau$ holds.

$$|B_j| \leq \epsilon 2^n + (q_1 + i).$$

Fix a function h. $E_j = \tau_j \wedge i$ is the index of interest for $(m, R, h) \wedge \{h_{\alpha_i \to \beta}\}_{\beta \in S}\}$. As there are $|S|$ many choices of $h(\alpha_i)$ in E_j, for all $y \in S$

$$\Pr[h(\alpha_i) = y | E_j] \leq \frac{1}{|S|}$$

Hence

$$\Pr[B_j | E_j] \leq \frac{|B_j|}{|S|} \leq \frac{2|B_j|}{2^n} \leq 2\epsilon + \frac{2(q_1 + i)}{2^n}$$

In the second inequality we use that $|S| \geq (1 - \epsilon_1^{1/2})2^n > 2^{n-1}$. Taking sum over all candidate h and thus all choice of i, we get the proposition. □

5 Crooked Indifferentiability Security Proof of EXor

Our simulator is same as before. However, we assume that distinguisher makes all $\mathcal{F}(m)$ queries to $g_R(m)$ in case it is not queried. However, it would be done after all queries are done. There is no loss to release all these m values after the original distinguisher finishes the queries. Now we recall the bad event (with this convention, the bad event now involves the $\mathcal{F}$ query transcript, which was not done before).

5.1 Game Transitions

Our crooked-indifferentiability proof relies on three intermediate game, denoted by G_0, G_1, and G_2. We start with the real game $G_0 := (h, C^{\tilde{h}})$. There are two public interfaces for the adversary to query. The first one is $\mathcal{O}_h$, which can be used to interact with the function H_l. The other one is $\mathcal{O}_C$, which can be used to compute $C^{\tilde{h}}$. For ease of explanation, we add two internal subroutines, one for computing $\tilde{g}_R$ and the other for evaluating $\tilde{h}(0, \cdot)$.

Game $\mathbf{G}_0$. In this game, we modify the $\mathcal{O}_h$ subroutine. For every (j, x) query we recover the message $m = x \oplus r_j$, and precompute the response of all the $(i, m \oplus r_i)$ queries. Further, we compute the value of $\tilde{z} = \tilde{h}(\tilde{g}_R(m))$. These precomputations do not change the output for any of the query. Hence

$$\Pr[\mathcal{A}^{h, C^{\tilde{h}}} = 1] = \Pr[\mathcal{A}^{\mathbf{G}_0} = 1]$$

Game $\mathbf{G}_1$. In this game, we introduce two lists L_f and L_c. The entries in both the lists are of the form $(m, x, z, \tilde{z})$. We also introduce two BAD events in the code of $\mathcal{O}_h$ as well as in the code of $\mathcal{O}_C$. Notice that, both the subroutine computes $\tilde{h}(0, \tilde{g}_R(m))$.

The first bad event (BAD1) happens if $h(0, \tilde{g}_R(m))$ has been set already. This can happen in two ways. The first one is during a previous $h(0, \tilde{g}_R(m'))$ computation for a different m'. In that case there is a collision in the output of $\tilde{g}_R$.

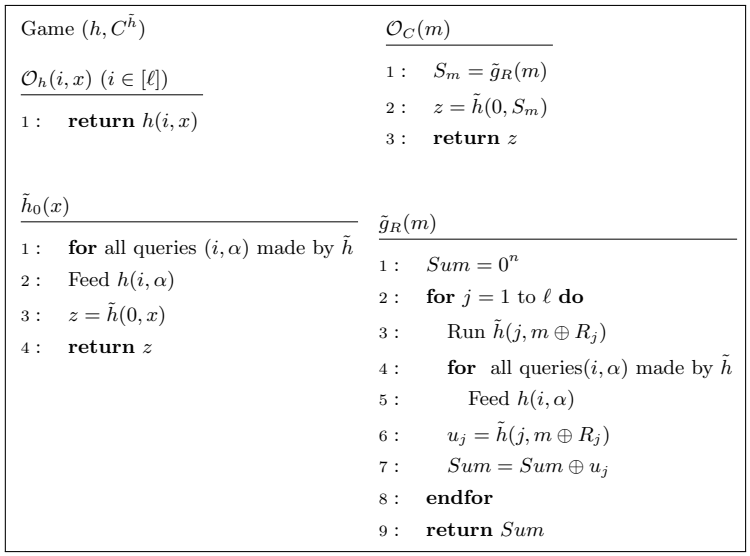

Fig. 3. Game Real

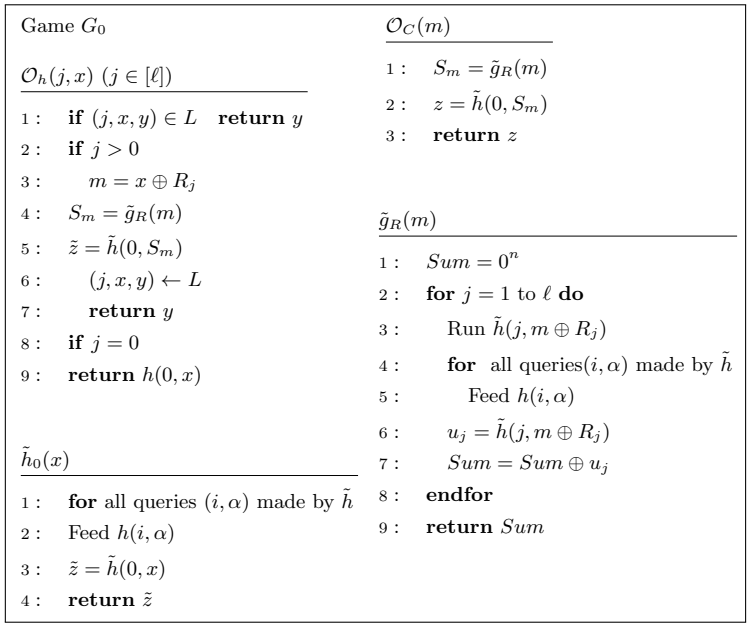

Fig. 4. Game G_0

The second way is via a $\mathcal{O}_h(0,x)$ query (by the distinguisher or the subverted implementations). When queried, such an x was not related to a message.
The second bad event happens (BAD2) if $\tilde{h}(0,\tilde{g}_R(m)) \neq h(0,\tilde{g}_R(m))$. In other

words, $\tilde{g}_R(m)$ is a subverted point for $\tilde{h}(0,.)$. The final change is in the introduction of the random oracle $\mathcal{F}$. Our intention in this game is to program the $h(0,x)$ as $\mathcal{F}(m)$ if $x = \tilde{g}_R(m)$. Hence after the compuation of $\tilde{g}_R(m)$, if we find $h(0,\tilde{g}_R(m))$ is not already set (BAD1 did not happen), we set $h(0,\tilde{g}_R(m)) = \mathcal{F}(m)$. As we are not changing the previously set values, the transcript is consistent. Thus, the distinguisher's view remains unchanged. Thus we get,

$$\Pr[\mathcal{A}^{\mathbf{G}_0} = 1] = \Pr[\mathcal{A}^{\mathbf{G}_1} = 1]$$

Before moving to the next game, we state the significance of BAD2. Looking ahead, in such a situation, the simulator will not be able to "program" the output of $\tilde{h}(0,\tilde{g}_R(m))$ as $\mathcal{F}(m)$, and thus loosing the consistency with the random oracle.

Game $\mathbf{G}_2$. In this game, we introduce the changes in the computation. First, we (re)program $h(0,\tilde{g}_R(m)) = \mathcal{F}(m)$ even if it was previously set. Moreover, we set $\tilde{z}$, the output of $O_c(m)$ query to always be same as $h(0,\tilde{g}_R(m))$. These modifications create changes in the output in two places. The first one is in the case of BAD1. The second is in the case of BAD2, $\tilde{g}_R(m)$ is a subverted point for $\tilde{h}(0,.)$. The rest of the game remains unchanged. As the two games are identical until one of the bad event happens, using the fundamental lemma of game playing proofs,

$$\left|\Pr[\mathcal{A}^{\mathbf{G}_1} = 1] = \Pr[\mathcal{A}^{\mathbf{G}_2} = 1]\right| \leq \Pr[\text{BAD1} \cup \text{BAD2}].$$

Game $(S^{\mathcal{F}}, \mathcal{F})$. It is also easy to see that the output distribution of the game $\mathbf{G}_2$ is identical to the game $(S^{\mathcal{F}}, \mathcal{F})$.

$$\Pr[\mathcal{A}^{\mathbf{G}_2} = 1] = \Pr[\mathcal{A}^{(S^{\mathcal{F}},\mathcal{F})} = 1]$$

Finally, collecting all the probabilities, we get,

$$\Delta_{\mathcal{A}_2(r,z,R)}\big((h, C^{\tilde{h}}(R,\cdot))\ ;\ (S^{\mathcal{F}}(H_{z,r}, z, R), \mathcal{F})\big) \leq \Pr[\text{BAD1} \cup \text{BAD2}].$$

5.2 Bounding Probability of Bad Events

We describe the bad events in terms of transcript notations as we did before. Following the transcript notation in Section 3, let bad_i denote the event that ith query is m to g_R oracle for which $(0,\tilde{g}_R(m)) \in \mathcal{D}(L_i)$ (same as BAD1) or $(0,\tilde{g}_R(m)) \in C^h$ (a crooked point for h and the vent is same as BAD2). Let $\mathsf{bad} = \vee_i \mathsf{bad}_i$. Thus, the distinguishing advantage is bounded by $\Pr(\vee_i \mathsf{bad}_i)$. We need to compute the probability given the randomness of R, h F such that $(R,h,F) \vdash \tau_i^A$ (transcript of the adversary). We first bound $\Pr(\mathsf{bad} \wedge (R,h) \in \mathcal{G}^*)$. By using Proposition 21, for every i, $\Pr(\mathsf{bad}_i | (R,h,F) \vdash \tau_i, (R,h) \in \mathcal{G}^*) \leq$

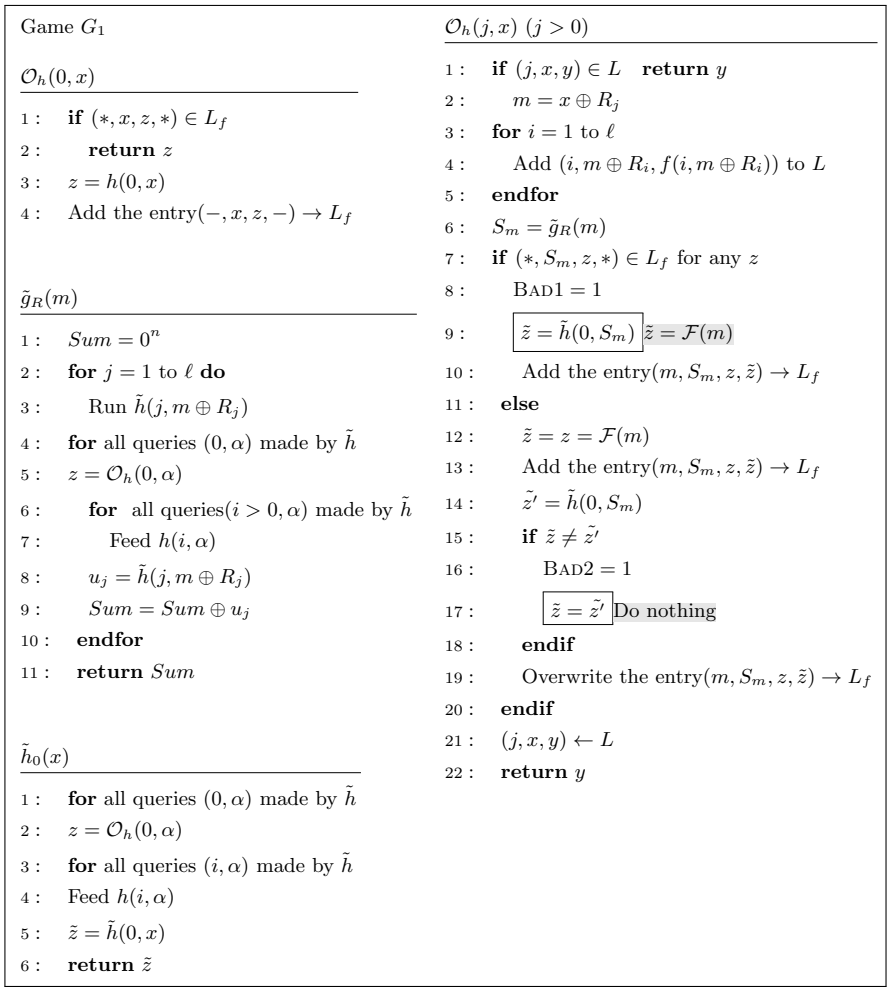

Fig. 5. Game G_1, G_2. The boxed entries are executed in G_1 whereas the highlighted entries are executed in G_2.

$2\epsilon + 2(q_1+i)/2^n$. Hence by summing over all i and using the bound of probability of not realizing $\mathcal{G}^*$, we get

$$\begin{aligned}\Pr(\mathsf{bad}) &\leq 2\epsilon q_2 + 2q_2(q_1+q_2)/2^n + p_1 \\ &= 2\epsilon q_2 + 2q_2(q_1+q_2)/2^n + \sqrt{3\tau\left(\epsilon + \frac{q_1}{2^n}\right)\frac{1}{4}} + \frac{1}{2^n}\end{aligned}$$

Note that q_2 denotes the total number of queries of $\mathcal{A}_2$ made to both the simulator and $\mathcal{F}$ (in our convention adversary makes all $\mathcal{F}$ queries to g_R of the simulator).

6 Crooked-Indifferentiability of Sponge Construction

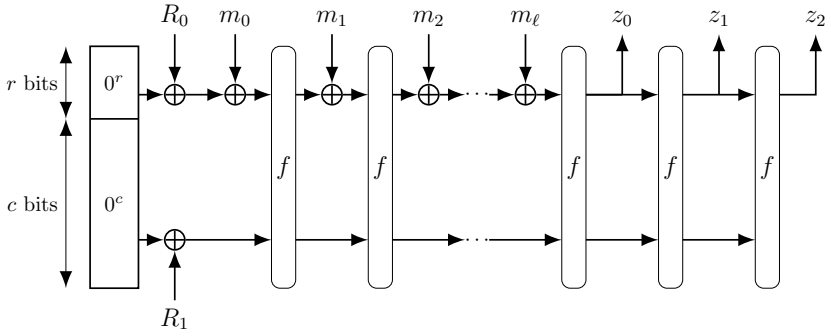

Fig. 6. Sponge Based Construction

6.1 Sponge Construction

We recall the sponge-construction [3]. Fix positive integers r, c, and let $n = r + c$. Let $f : \{0,1\}^n \to \{0,1\}^n$ be a function. The sponge construction C^f maps binary strings of length ℓ bits to binary strings of h bits using Figure 7.

Procedure Sponge

1 : $x = (x_a, x_c) = (0^r, 0^c)$ $\boxed{x = R}$
2 : **for** $i = 0$ to $\left\lceil \frac{\ell}{r} \right\rceil - 1$ **do**
3 : $(x_a, x_c) = f(x_a \oplus m_i, x_c)$
4 : **endfor**
5 : **for** $i = 0$ to $\left\lceil \frac{h}{r} \right\rceil - 1$ **do**
6 : Append x_a to output
7 : $(x_a, x_c) = f(x_a, x_c)$
8 : **endfor**

Fig. 7. The Sponge Construction C^f. Boxed Statement is executed in the subversion resistant implementation

6.2 Crooked Indifferentiable Sponge Construction

In order to handle subversion, we randomize the sponge construction by setting the IV to be equal to the random string R. In other words in Step 1, we set $x = R$. The rest of the construction is unchanged.

Theorem 22. *Let $f : \{0,1\}^n \to \{0,1\}^n$ be a random function and $C^f : \{0,1\}^\ell \to \{0,1\}^h$ be the sponge construction. Let r be the rate part and c be the capacity part of the chain. Then there exists a simulator S such that for all (κ, τ, ϵ) crooked distinguisher $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$*

$$\mathbf{Adv}^{\mathsf{crooked\text{-}indiff}}_{\mathcal{A},(C,f)} \leq \frac{q^2\tau^2 + q\kappa}{2^c} + 2^r \epsilon q$$

where q is the total number of queries made by $\mathcal{A}$.

The Simulator Our simulator emulates the simulator of Bertoni *et al* [4]. For completeness, we recall the simulator below. Specifically, we recall the following objects used in the proof.

The Simulator Graph The simulator maintains a graph G for recording the interactions for f. The vertex set of the graph is $V(G) \subseteq \{0,1\}^r \times \{0,1\}^c$. We represent a $v \in V(G)$ by an ordered pair (v_r, v_c) where $v_r \in \{0,1\}^r$ and $v_c \in \{0,1\}^c$. The (directed) edge set of the graph is represented by $E(G)$. The simulator also keeps a list $L \subseteq \{0,1\}^c$. L is used to ensure that the c-part of all the responses of the simulators are unique.

NewNode Algorithm. The algorithm Newnode samples a node randomly in the simulator graph.

Findpath Algorithm The Findpath algorithms finds a message $m \in \{0,1\}^{\leq \ell}$ such that evaluating C with the random string R as IV, message m and simulator's responses so far will generate x as a query to f. In other words, x will be a chaining value in the computation of $C^f_R(m)$.

Simulating f in Stage I We denote our simulator using $\hat{S}$. In the first stage, $\hat{S}$ simulates f honestly. $\hat{S}$ starts with a *local copy* of the simulator graph with all the nodes, but no edges ($E(G) = \emptyset$) and an empty list L. When queried with a new input x, the simulator generates $y_c \xleftarrow{\$} \{0,1\}^c \setminus L$ and $y_r \xleftarrow{\$} \{0,1\}^r$, creates node (y_r, y_c), and adds an edge $(x, (y_r, y_c))$ to $E(G)$. $\hat{S}$ updates L by including y_c to the list.

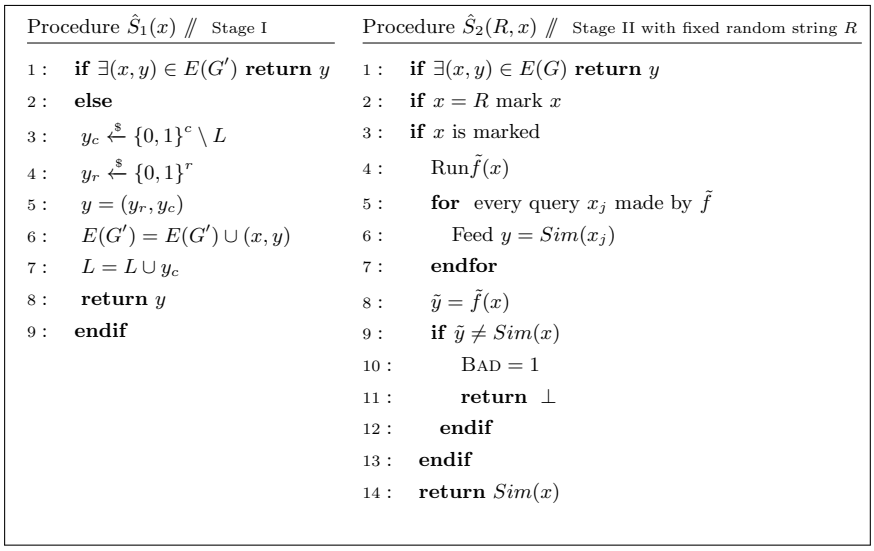

Fig. 8. Simulator for Sponge Construction. S_2 is initialized with z, R and $\tilde{f}$.

Simulating f in Stage II The simulator gets the implementation $\tilde{f}$, along with the advice string z. In addition, the simulator receives the random string $R \in \{0,1\}^n$. S initializes by *marking* the node R in the simulator graph. Following the simulator of [4], the idea of marking a node x is to declare that there is a path in the simulator graph from the root R to x.

Now, the simulator invokes $\tilde{f}$ on input x. For each query x_i made by $\tilde{f}$, $\hat{S}$ forwards the query to the simulator Sim as a query and upon receiving an answer, forwards it to the distinguisher. Finally when $\tilde{f}(x)$ returns a value, $\hat{S}$ checks whether $\tilde{f}(x) = Sim(x)$. If the check fails, the simulator raises the flag $Bad0$ and aborts. Otherwise, it returns $Sim(x)$.

Proving the Crooked Indifferentiability. The detail of the games and transitional probabilities are described in Section8.1. We present a proof sketch here. The crooked indifferentiability is proved via the following lemma.

Lemma 23. *If* Bad *does not happen then* $\epsilon \leq \frac{q^2\tau^2}{2^c}$. *Moreover,*

$$\Pr[\text{Bad}] \leq q\epsilon \cdot 2^r$$

Proof of Lemma 23. If Bad does not happen, then our simulator emulates the simulator of [4] perfectly. Note, the Newnode subroutine does not sample any x_c such that for some x_r, $(x_r, x_c) \in z$. Moreover, none of the marked nodes in the tree is subverted. Hence, in that case, the classical indifferentiability simulator perfectly simulates f maintaining consistency with the random oracle F. By

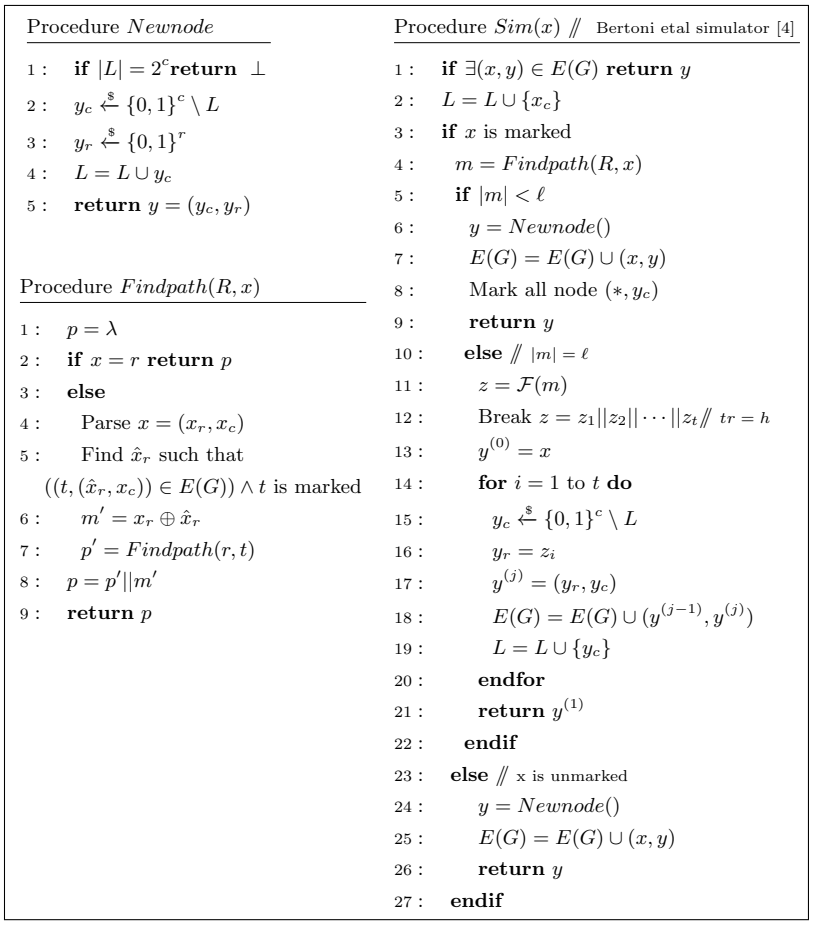

Fig. 9. Bertoni *et al* Simulator for Classical indifferentiability of Sponge Construction.[4]

the classical indifferentiability theorem of [4], $\epsilon \leq \frac{q^2\tau^2}{2^c}$. Here $\tau$is the number of queries made by the implementation.

Recall that the event BAD happens if for some marked node x in the graph, x is subverted; $\tilde{f}(x) \neq f(x)$. Let $\mathcal{A}$ makes q_1 queries in the first stage and q_2 queries to the second stage. We say a simulator query to be safe if it is unsubverted.Observe that for each safe query x, $f(x)$ is uniformly distributed. Let E_i denote the event that i^{th} query made by $\mathcal{A}$ to the simulator, the input x_i is marked and x_i is not safe. . $x_i = x_r||x_c$. If x_i is marked, then for some marked x_j, $j < i$, and for some $m_i \in \{0,1\}^r$, $x_i = m||0^c \oplus \tilde{f}(x_j)$. Conditioned on x_j is safe, $\tilde{f}(x_j) = f(x_j)$ is uniformly distributed. Hence, x_c is independently distributed. By definition of worst case subversion, and taking union bound over all possible x_r, the probability $\tilde{f}(x_i)$ is bounded by $\epsilon \cdot 2^r$.

$$\begin{aligned}\Pr[\textsc{Bad}] &\leq \sum_{i=1}^{q} \Pr[E_i | \wedge_{j=1}^{i-1} E_j] \\ &\leq \sum_{i=1}^{q} (\epsilon \cdot 2^r) \\ &= q\epsilon \cdot 2^r\end{aligned}$$

7 Conclusion

In this paper, we revisited the recently introduced crooked indifferentiability notion. We showed that the proof of crooked indifferentiability of enveloped XOR construction in [16] is incomplete. We developed new technique to prove crooked indifferentiability of the same construction. We also show that the sponge construction with randomized initial value is also crooked indifferentiable secure hash function.

References

1. Andreeva, E., Mennink, B., Preneel, B.: On the indifferentiability of the Grøstl hash function. In: Garay, J.A., Prisco, R.D. (eds.) SCN 10. LNCS, vol. 6280, pp. 88–105. Springer, Heidelberg (Sep 2010). https://doi.org/10.1007/978-3-642-15317-4_7
2. Bellare, M., Paterson, K.G., Rogaway, P.: Security of symmetric encryption against mass surveillance. In: Garay, J.A., Gennaro, R. (eds.) CRYPTO 2014, Part I. LNCS, vol. 8616, pp. 1–19. Springer, Heidelberg (Aug 2014). https://doi.org/10.1007/978-3-662-44371-2_1
3. Bertoni, G., Daemen, J., Peeters, M., Assche, G.: Sponge functions. ECRYPT Hash Workshop 2007 (01 2007)
4. Bertoni, G., Daemen, J., Peeters, M., Van Assche, G.: On the indifferentiability of the sponge construction. In: Smart, N.P. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 181–197. Springer, Heidelberg (Apr 2008). https://doi.org/10.1007/978-3-540-78967-3_11
5. Bhattacharyya, R., Mandal, A., Nandi, M.: Indifferentiability characterization of hash functions and optimal bounds of popular domain extensions. In: Roy, B.K., Sendrier, N. (eds.) INDOCRYPT 2009. LNCS, vol. 5922, pp. 199–218. Springer, Heidelberg (Dec 2009)
6. Chang, D., Nandi, M.: Improved indifferentiability security analysis of chopMD hash function. In: Nyberg, K. (ed.) FSE 2008. LNCS, vol. 5086, pp. 429–443. Springer, Heidelberg (Feb 2008). https://doi.org/10.1007/978-3-540-71039-4_27
7. Coron, J.S., Dodis, Y., Malinaud, C., Puniya, P.: Merkle-Damgård revisited: How to construct a hash function. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 430–448. Springer, Heidelberg (Aug 2005). https://doi.org/10.1007/11535218_26
8. Degabriele, J.P., Paterson, K.G., Schuldt, J.C.N., Woodage, J.: Backdoors in pseudorandom number generators: Possibility and impossibility results. In: Robshaw, M., Katz, J. (eds.) CRYPTO 2016, Part I. LNCS, vol. 9814, pp. 403–432. Springer, Heidelberg (Aug 2016). https://doi.org/10.1007/978-3-662-53018-4_15

9. Dodis, Y., Reyzin, L., Rivest, R.L., Shen, E.: Indifferentiability of permutation-based compression functions and tree-based modes of operation, with applications to MD6. In: Dunkelman, O. (ed.) FSE 2009. LNCS, vol. 5665, pp. 104–121. Springer, Heidelberg (Feb 2009). https://doi.org/10.1007/978-3-642-03317-9_7
10. Maurer, U.M., Renner, R., Holenstein, C.: Indifferentiability, impossibility results on reductions, and applications to the random oracle methodology. In: Naor, M. (ed.) TCC 2004. LNCS, vol. 2951, pp. 21–39. Springer, Heidelberg (Feb 2004). https://doi.org/10.1007/978-3-540-24638-1_2
11. Mennink, B.: Indifferentiability of double length compression functions. In: Stam, M. (ed.) 14th IMA International Conference on Cryptography and Coding. LNCS, vol. 8308, pp. 232–251. Springer, Heidelberg (Dec 2013). https://doi.org/10.1007/978-3-642-45239-0_14
12. Moody, D., Paul, S., Smith-Tone, D.: Improved indifferentiability security bound for the JH mode. Cryptology ePrint Archive, Report 2012/278 (2012), `http://eprint.iacr.org/2012/278`
13. Naito, Y.: Indifferentiability of double-block-length hash function without feed-forward operations. In: Pieprzyk, J., Suriadi, S. (eds.) ACISP 17, Part II. LNCS, vol. 10343, pp. 38–57. Springer, Heidelberg (Jul 2017)
14. Russell, A., Tang, Q., Yung, M., Zhou, H.S.: Cliptography: Clipping the power of kleptographic attacks. In: Cheon, J.H., Takagi, T. (eds.) ASIACRYPT 2016, Part II. LNCS, vol. 10032, pp. 34–64. Springer, Heidelberg (Dec 2016). https://doi.org/10.1007/978-3-662-53890-6_2
15. Russell, A., Tang, Q., Yung, M., Zhou, H.S.: Generic semantic security against a kleptographic adversary. In: Thuraisingham, B.M., Evans, D., Malkin, T., Xu, D. (eds.) ACM CCS 2017. pp. 907–922. ACM Press (Oct / Nov 2017). https://doi.org/10.1145/3133956.3133993
16. Russell, A., Tang, Q., Yung, M., Zhou, H.S.: Correcting subverted random oracles. In: Shacham, H., Boldyreva, A. (eds.) CRYPTO 2018, Part II. LNCS, vol. 10992, pp. 241–271. Springer, Heidelberg (Aug 2018). https://doi.org/10.1007/978-3-319-96881-0_9
17. Young, A., Yung, M.: The dark side of "black-box" cryptography, or: Should we trust capstone? In: Koblitz, N. (ed.) CRYPTO'96. LNCS, vol. 1109, pp. 89–103. Springer, Heidelberg (Aug 1996). https://doi.org/10.1007/3-540-68697-5_8

Appendix

8 Leftout proofs for sponge construction

8.1 Game Transitions

Our crooked-indifferentiability proof relies on four intermediate game, denoted by $\mathbf{G}_0, \mathbf{G}_1, \mathbf{G}_1$, and $\mathbf{G}_2$. We start with the real game $(h, C^{\tilde{h}})$. There are two public interfaces for the adversary to query. The first one is $\mathcal{O}_f$, which can be used to interact with the function f. The other one is $\mathcal{O}_C$, which can be used to compute $C^{\tilde{f}}$. In the intermediate games we shall use additional subroutine $\mathcal{G}$.

Game $\mathbf{G}_0$. In this game, we introduce the subroutine $\mathcal{G}$ which acts as a common interface to $f, \tilde{f}$. We modify the $\mathcal{O}_h$ and $\mathcal{O}_C$ subroutine. For queries to f, $\mathcal{G}$ is

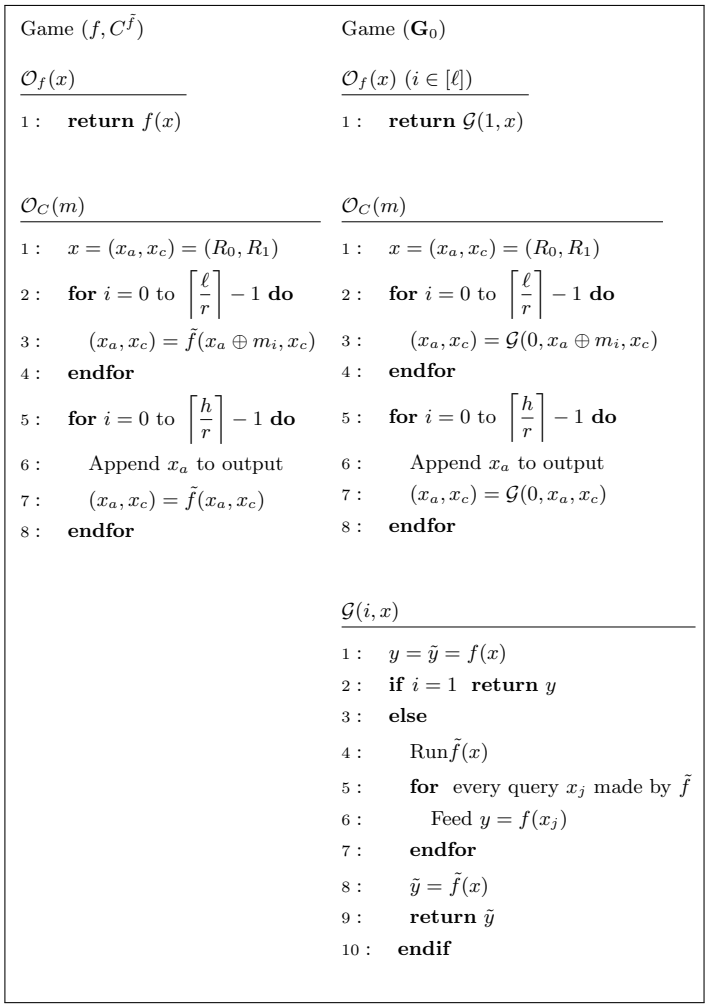

Fig. 10. Game Real and Game $\mathbf{G}_0$

called with parameter f whereas for $\tilde{f}$ the parameter value is set to be 0. These changes are ornamental and do not change the output for any of the query. Hence

$$\Pr[\mathcal{A}^{h, C^{\tilde{h}}} = 1] = \Pr[\mathcal{A}^{\mathbf{G}_0} = 1]$$

Game $\mathbf{G}_1$. In this game, we modify the subroutine $\mathcal{G}$. When queried with $(0, x)$ for intended value of $\tilde{f}(x)$, $\mathcal{G}$ checks whether $\tilde{f}(x) = f(x)$. If the equality does not hold $\mathcal{G}$ sets the BAD flag. However, it still returns $\tilde{f}(x)$. As the output of any query does not change,

$$\Pr[\mathcal{A}^{\mathbf{G}_0} = 1] = \Pr[\mathcal{A}^{\mathbf{G}_1} = 1]$$

Game $\mathbf{G}_1$. In this game, when BAD flag is set, $f(x)$ is returned. Everything else remain unchanged. Using the fundamental lemma of game playing proof,

$$\left|\Pr[\mathcal{A}^{\mathbf{G}_1} = 1] - \Pr[\mathcal{A}^{\mathbf{G}_2} = 1]\right| \leq \Pr[\text{BAD}]$$

Game $\mathbf{G}_2$. We note that in game $\mathbf{G}_1$, all the $\mathcal{G}(x)$ queries made by O_c is answered with $f(x)$. Hence, in Game $\mathbf{G}_2$, we give O_c direct access to f. Output distribution of the game remains exactly same after this change. Hence,

$$\Pr[\mathcal{A}^{\mathbf{G}_1} = 1] = \Pr[\mathcal{A}^{\mathbf{G}_2} = 1]$$

Game $\mathbf{G}_3$. We replace the oracles (f, C^f) by $(S, \mathcal{F}^S)$ where S is the simulator of Bertoni *et al* [4]. By the results in [4],

$$\left|\Pr[\mathcal{A}^{\mathbf{G}_2} = 1] - \Pr[\mathcal{A}^{\mathbf{G}_3} = 1]\right| \leq \frac{q^2\tau^2}{2^c}.$$

Note that the numerator in the right hand side is the square of total number of queries made to the simulator, which in Game $\mathbf{G}_3$, $q\tau$ as the implementation makes τ many queries for each invocation.

Finally, we observe that $\mathcal{G}$ works identically with the second stage simulator of Figure8. So As we initialize the simulator of [4] with L from the stage I, the two games are identical. Hence we get,

$$\Pr[\mathcal{A}^{\mathbf{G}_3} = 1] = \Pr[\mathcal{A}^{(S^{\mathcal{F}}, \mathcal{F})} = 1]$$

Collecting all the probabilities, we get

$$\begin{aligned}\Delta_{\mathcal{A}_2(r,z,R)}\big((f, C^{\tilde{f}}(R,\cdot))\ ;\ (S^{\mathcal{F}}(H_{z,r}, z, R), \mathcal{F})\big) &\leq \Pr[\text{BAD}] + \frac{q^2\tau^2}{2^c} \\ &\leq \frac{q^2\tau^2}{2^c} + q\epsilon 2^r.\end{aligned}$$

Game ($\mathbf{G}_1$)

$\mathcal{O}_f(x)$

1 : **return** $\mathcal{G}(1, x)$

$\mathcal{O}_C(m)$

1 : $x = (x_a, x_c) = (R_0, R_1)$
2 : **for** $i = 0$ to $\left\lceil \frac{\ell}{r} \right\rceil - 1$ **do**
3 : $(x_a, x_c) = \mathcal{G}(0, x_a \oplus m_i, x_c)$
4 : **endfor**
5 : **for** $i = 0$ to $\left\lceil \frac{h}{r} \right\rceil - 1$ **do**
6 : Append x_a to output
7 : $(x_a, x_c) = \mathcal{G}(0, x_a, x_c)$
8 : **endfor**

$\mathcal{G}(i, x)$

1 : $y = \tilde{y} = f(x)$
2 : **if** $i = 1$ **return** y
3 : Run$\tilde{f}(x)$
4 : **for** every query x_j made by $\tilde{f}$
5 : Feed $y = f(x_j)$
6 : **endfor**
7 : **if** $\tilde{f}(x) \neq f(x)$
8 : BAD $= 1$
9 : $\boxed{\tilde{y} = \tilde{f}(x)}$ $\tilde{y} = f(x)$
10 : **return** $\tilde{y}$
11 : **endif**

Game ($\mathbf{G}_2$)

$\mathcal{O}_f(x)$

1 : **return** $\mathcal{G}(1, x)$

$\mathcal{O}_C(m)$

1 : $x = (x_a, x_c) = (R_0, R_1)$
2 : **for** $i = 0$ to $\left\lceil \frac{\ell}{r} \right\rceil - 1$ **do**
3 : $(x_a, x_c) = f(, x_a \oplus m_i, x_c)$
4 : **endfor**
5 : **for** $i = 0$ to $\left\lceil \frac{h}{r} \right\rceil - 1$ **do**
6 : Append x_a to output
7 : $(x_a, x_c) = f(x_a, x_c)$
8 : **endfor**

$\mathcal{G}(i, x)$

1 : $y = \tilde{y} = f(x)$
2 : **if** $i = 1$ **return** y
3 : Run$\tilde{f}(x)$
4 : **for** every query x_j made by $\tilde{f}$
5 : Feed $y = f(x_j)$
6 : **endfor**
7 : **if** $\tilde{f}(x) \neq f(x)$
8 : BAD $= 1$
9 : $\tilde{y} = f(x)$
10 : **return** $\tilde{y}$
11 : **endif**

Fig. 11. Game $\mathbf{G}_1, \mathbf{G}_2$

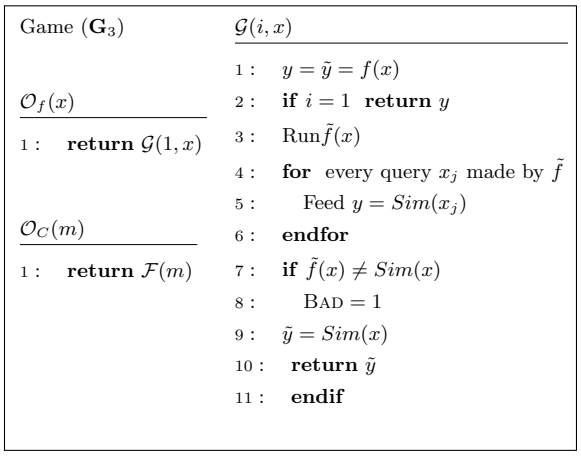

Fig. 12. Game $\mathbf{G}_3$