Cryptographic Constructions Supporting Implicit Data Integrity

Michael Kounavis, David Durham† and Sergej Deutsch†

† Intel Labs, Intel Corporation, 2111, NE 25th Avenue, Hillsboro, OR 97124
email: michael_kounavis@hotmail.com.
{david.durham, sergej.deutsh}@intel.com

Rev. 0.1 May 2018, Rev. 1.0 Feb. 2021

Abstract

We study a methodology for supporting data integrity called 'implicit integrity' and present cryptographic constructions supporting it. Implicit integrity allows for corruption detection without producing, storing or verifying mathematical summaries of the content such as MACs and ICVs, or any other type of message expansion. As with authenticated encryption, the main idea behind this methodology is that, whereas typical user data demonstrate patterns such as repeated bytes or words, decrypted data resulting from corrupted ciphertexts no longer demonstrate such patterns. Thus, by checking the entropy of some decrypted ciphertexts, corruption can be possibly detected.

The main contribution of this paper is a notion of security which is associated with implicit integrity, and which is different from the typical requirement that the output of cryptographic systems should be indistinguishable from the output of a random permutation. The notion of security we discuss reflects the fact that it should be computationally difficult for an adversary to corrupt some ciphertext so that the resulting plaintext demonstrates specific patterns. We introduce two kinds of adversaries. First, an input perturbing adversary performs content corruption attacks. Second an oracle replacing adversary performs content replay attacks. We discuss requirements for supporting implicit integrity in these two adversary models, and provide security bounds for a construction called IVP, a three-level confusion diffusion network which can support implicit integrity and is inexpensive to implement.

1 Introduction

1.1 The Concept of implicit data integrity

This paper addresses the problem of corruption detection without producing, storing or verifying mathematical summaries of the content. Such summaries, typically known as Message Authentication Codes (MACs) [6] [7] or Integrity Check Values (ICVs) are typically costly to maintain and use. The standard way of supporting data integrity is by using MACs produced by cryptographic hash functions such as SHA256 [4] or SHA3 [5], the use of which typically results in latency, storage and communication overheads. These overheads are due to the unavoidable message expansion associated with using the MACs. For example, if a system protects a cache line with a MAC [10], this MAC value needs to be read in every data read operation. This wastes memory access bandwidth resources as each data read operation needs to be realized as two memory read operations.

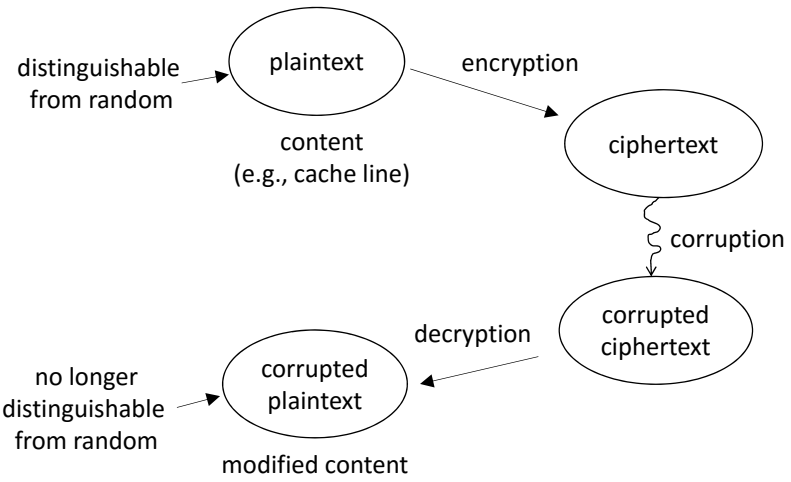


Figure 1: The concept of implicit integrity

Another area where our work applies is in network and communication systems which employ a number of different protocols at different layers of the network stack, ranging from the link layer to the application layer, with additional corruption detection mechanisms [6] [7] [31]. Many of these protocols employ corruption detection mechanisms to cater for any intentional or unintentional corruptions encountered during the transmission of data from point A to point B. The cost of providing such capabilities is the additional metadata per packet, which can be sigificant. This paper describes an alternative methodology that uses pattern techniques in order to support corruption detection for the large majority of user data without message expansion. The main idea is discussed below.

If some content exhibits patterns (i.e., has low entropy), then such content can be distinguished from random data. Let's consider that this content is encrypted, as shown in the figure, where the encryption algorithm is a good pseudo-random permutation and thus can successfully approximate a random oracle. The cipher text which is produced in this way is no longer distinguishable from random data, under certain reasonable assumptions about the adversary. Any corruption on the cipher text results in a new cipher text value, which is different from the original one. Furthermore, any decryption operation on this new cipher text value results in a corrupted plaintext value which is different from the original one as well. As decryption is the inverse operation of encryption, the decryption algorithm also approximates a random oracle under reasonable adversary models. Because of this reason, the corrupted plaintext value is also indistinguishable from random data with very high probability. From a system realization standpoint, the corrupted plaintext is indistinguishable from random data because block ciphers or cryptographic constructions typically perform strong mixing of their input bits. Due to an 'avalanche effect' associated with the decryption oracle, even a single bit change in the cipher text can affect all bits of the decrypted plaintext. For these reasons, checking the entropy of the result of a decryption operation can be a reliable test for detecting corruption for some data. We refer to such methodology as 'implicit data integrity' or just 'implicit integrity'.

One of the main challenges in building a system, the operation of which is based on the principle of implicit integrity is how to define 'high or 'low' entropy. It is not straightforward how to determine that some content's entropy is 'low enough' or 'high enough' so as to safely deduce that the original content has not been corrupted.

Another challenge has to do with the design of cryptographic constructions for supporting implicit integrity and the derivation of analytical proofs associated with security claims. A large body of work exists that focuses on bounds associated with distinguishing the output of cryptographic constructions from that of a random permutation or a random function. In our work we argue that such strong notion of security may not be required in the context of implicit integrity. The security objective which we discuss in this paper is that it should be computationally hard for adversaries to modify a set of given ciphertext messages so as to result in plaintexts of low entropy, while preserving confidentiality. This paper focuses on formalizing this security objective, exploring the requirements of cryptographic constructions supporting it, and providing an example of a simple, implementable three level confusion diffusion network that meets these requirements for a specific security level.

1.2 Paper Summary

A first topic covered by this technical report is the description of a class of constructions which we refer to as 'Random Oracles according to Observer functions' (RO^2). An observer function is a function that searches the output of cryptographic systems in order to detect unusual behavior, such as the presence of patterns. Patterns can be repeated nibbles, bytes, words or double words. If an observer function detects unusual behavior with the same or similar probability in a cryptographic system's output as in a random oracle's output then such cryptographic system belongs to the class of RO^2 constructions associated with the specific observer function.

A second topic covered by this paper is the description of a security model against data corruption and replay attacks, which is associated with implicit integrity and connected with the class of RO^2 constructions. Specifically, we show that if a construction is in the RO^2 class, then the construction always supports some form of implicit data integrity, and is secure in the proposed model. The proposed model comprises two kinds of adversaries. First, an input perturbing adversary is an algorithm which is given a set of ciphertext messages $q_0, \ldots, q_{m-1}$, the plaintexts of which exhibit patterns. The algorithm succeeds if it finds a ciphertext message y which is different from $q_0, \ldots, q_{m-1}$, the plaintext of which also exhibits patterns. Security in this adversary model indicates protection against data corruption attacks. Second, an oracle replacing adversary is an algorithm which, like the input perturbing adversary, is given a set of cipher messages $q_0, \ldots, q_{m-1}$ the plaintexts of which also exhibit patterns. The algorithm succeeds if it replaces a set of internal random permutations $\mathcal{R}_0, \mathcal{R}_1, \ldots, \mathcal{R}_{n-1}$ queried by the decryption system with a new set of random permutations $\mathcal{R}'_0, \mathcal{R}'_1, \ldots, \mathcal{R}'_{n-1}$, so that there exists a ciphertext message $y \in \{q_0, \ldots, q_{m-1}\}$ the plaintext of which continues to exhibit patterns even when the permutations $\mathcal{R}_0, \mathcal{R}_1, \ldots, \mathcal{R}_{n-1}$ are replaced by $\mathcal{R}'_0, \mathcal{R}'_1, \ldots, \mathcal{R}'_{n-1}$. Security in this adversary model indicates protection against content replay attacks.

A third topic of this paper is a cryptographic construction called IVP (stands for 'Implicit integrity Via Pre-processing'), which is in the proposed class RO^2. IVP is a three level confusion diffusion network that includes pseudo-random permutations of varying widths. Proofs for this construction derive from computing the probability that the system state is being altered by the flow of differentials inside the construction. Differentials are caused by adversarial perturbations. In our analysis we consider that the number of queries issued by adversaries is bounded in such a way, so that the internal pseudo-random permutations queried by the proposed construction are indistinguishable from truncated output random oracles. In this case, differential signals coming out of these components are indistinguishable from random, uniformly distributed and statistically independent signals.

1.3 Protecting data that do not exhibit patterns

One issue that needs addressing, when building a system that protects data using the principle of implicit integrity, is how to detect corruption if data do not exhibit patterns. As we elaborate below, patterns can be found in up to 91% of the data of client workloads and 84% of the data of server workloads. The numbers come from observations on 111 million representative client workload cache lines and 1.4 billion representative server workload cache lines. Whereas such data can be protected using implicit integrity, the remaining $9\% - 16\%$ of the data need protecting also.

Such design issue can be easily addressed. Implementations can protect the overwhelming majority of user data that exhibit patterns using implicit integrity and the remaining data using standard techniques. There is nothing in the implicit integrity methodology, discussed here, that prevents it from being used together with other independent integrity mechanisms such as MACs. Such solutions can co-exist with implicit integrity.

Cost reduction comes from the fact that only MACs computed from high entropy data need to be stored and accessed. If some decrypted content exhibits patterns, then there is some assurance that no corruption has occurred. If no patterns are exhibited, however, a search can be made for a MAC associated with the content. If no MAC is found, then the data is deemed corrupted. Otherwise, an integrity check is made using the returned MAC. Such implementation can use a content addressable memory unit or a hash table for accessing and managing MACs. We remind that MACs are significantly fewer than those required for protecting all the data, and this is the main advantage of this approach. Further investigations on hardware and operating system changes required in order to support implicit integrity are beyond the scope of this paper.

2 Related work

2.1 Robust Authenticated Encryption (RAE)

References [26, 27] discuss an integrity methodology called Robust Authenticated Encryption (RAE), which generally involves some type of message expansion in the form of some additional bits of length λ. Such redundancy bits are appended to an input message before encryption. At decryption time, RAE verifies that the redundancy bits have the value they are supposed to have. The authors of [26, 27] also discuss a subcase of RAE where $\lambda = 0$. In this subcase, integrity is provided by taking into account further redundancy which may exist in an input message, besides the additional bits of length λ. Such subcase of RAE is equivalent to our concept of implicit integrity. There are several differences between our work and references [26, 27], however.

The first difference is in the extent of the security analysis provided concerning integrity without message expansion. Message expansion (i.e., all cases where $\lambda > 0$) is an integral part of the security methodology of references [26, 27]. In our work it is not. For example, the game $\mathbf{RAE}_{\Pi,S}$ defined in reference [26] involves a simulator S which is invoked only when there is message expansion. Such simulator is part of this game and is used in the proofs that reduce the security of the paper's proposed construction AEZ to the indistinguishability of AEZ from a pseudo-random permutation.

A second difference between our work and references [26, 27] is in the formulation of the security objective that guides the design of the proposed cryptographic constructions. References [26, 27] base their notion of security on the indifferentiability or indistinguishability of their proposed constructions from a random permutation or a random function. Indifferen-

tiability and indistinguishability are mostly used in these references in the general sense (as in reference [28]). For example reference [26] proposes a wide block cipher design (AEZ) and proves that this design is indeed difficult to distinguish from a wide random permutation. In this paper we argue that, while such notion of security is useful, and the analysis of references [26, 27] insightful, implicit integrity may not need such strong notion of security.

There is a simple argument to see why this is the case. Let's consider some hypothetical implicit integrity system, the security of which is based on the presence of some specific patterns in input data. For example, patterns may be sequences of 10 repeated adjacent bytes, appearing with probability $2^{-65.1}$ among random 1024 bit inputs, while being frequent among specific user data. Let's also consider that the provided security level of 65 bits is adequate to the users of the system and that the users might not even mind of a small drop in the security offered (e.g., if 62 or 63 bits of security are offered). Finally, let's assume that the system uses a wide block cipher which is highly indistinguishable from a wide random permutation, where a distinguisher's advantage is bounded by 2^{-256} given some adversary query budget. In this hypothetical example, the distinguisher's advantage is much smaller than the probability of seeing patterns in corrupted input data. As a result, the security of the system is mostly determined by the probability of seeing patterns in corrupted input data, and not by the indistinguishability of the wide block cipher employed. Due to this fact, a wide block cipher like the one employed may be functional but possibly unnecessary. A valid design could have used an alternative wide construction, which may have been "more distinguishable" from a random permutation, but still able to support the necessary security level associated with 62-65 bit implicit integrity, and also provide confidentiality. The confidentiality provided might not have been the same as the one provided by a wide block cipher, but it may have been the same as the confidentiality provided by a mode of a block cipher.

One aspect of our methodology is that we decouple the derivation of any security statements concerning integrity from any statements concerning confidentiality, and deal only with integrity. We also address the question whether it is possible to study implicit integrity-based security without studying the indifferentiability of constructions from ideal primitives in the general sense. Our answer is affirmative. We use the concept of random oracles according to observer functions for this purpose. Random oracles according to observer functions can be used for formulating security objectives which are associated with specific types of pattern detectors and result in designs which are possibly simpler than alternative wide block cipher designs. Finally, one could argue in favor of a methodology like the one followed in this paper due to the fact that, for some constructions, it may be easier to study their behavior in the context of producing outputs with specific patterns, as opposed to studying their indifferentiability or indistinguishability in the general sense.

2.2 Correlation intractability

Besides the relationship between our work and references [26, 27], the concept of random oracles according to observer functions bears some similarity with the concept of correlation intractability discussed in reference [23]. The aim of reference [23] is different from ours. Reference [23] establishes that there exist signature and encryption schemes that are secure in the random oracle model, but for which any implementation of a random oracle results in insecure schemes. In the process, reference [23] introduces two concepts which are related to our definition of the class RO^2. First, a binary relation is introduced as "evasive" if, when given access to a random oracle $\mathcal{O}()$, it is infeasible to find a string x so that the pair $(x; \mathcal{O}(x))$ is in the relation. Second, a function ensemble F is called correlation intractable if for every evasive binary relation, given the description of a uniformly selected function f_s from F it is

infeasible to find an x such that $(x; f_s(x))$ is in the relation.

Notable differences between our paper and [23] are the following: First, the binary relation as defined in [23] involves a relationship between two strings. In our case, we employ an observer function which examines the output of the system and produces a Boolean value from this output, after running a polynomial time algorithm. The two definitions may be equivalent, however, the difference in the formalism reflects further contrasts. Second, in reference [23], much is said about intractability. There is a function defined, which characterizes probability values associated with the output of a system as negligible or non-negligible, and the whole function ensemble as tractable or intractable. This is a "black-and-white" binary characterization which is not directly relevant to our analysis. In the case of the RO^2 definition, we are more interested in the relative behavioral difference in the output of a system, when compared to a random oracle or a random function, in the context of producing outputs with specific patterns. For this purpose, we introduce the concepts of the pattern observation probability and the indistinguishability parameter ϵ. This parameter characterizes this relative behavioral difference between the real system we examine and a truncated output random oracle.

3 Preliminary concepts and definitions

3.1 Patterns and observer functions

We begin our analysis by formally defining the terms "pattern" and "observer function".

Definition 1: *Definition of a pattern.* A Boolean function $\mathcal{F}_{L,l,m,P} : \{0,1\}^L \rightarrow \{\mathsf{true}, \mathsf{false}\}$, associated with a string length value L, a segment length value $l \in [1, L]$, a number of segments $m \in [1, L/l]$, and a set of additional parameters P, is called a "pattern", if and only if it accepts as input a binary string x from the set $\{0,1\}^L$, outputs one of $\{\mathsf{true}, \mathsf{false}\}$, and the response depends only on the input string x and the values of L, l, m, P.

In the analysis that follows, some of the subscripts from L, l, m, P may be omitted when denoting patterns, if their values are implied. Furthermore, additional subscripts may be introduced in order to denote specific types of checks performed by patterns on input strings.

Definition 2: *Definition of the set of strings exhibiting a pattern.* A set of binary strings $\Pi(\mathcal{F}_{L,l,m,P}) \subseteq \{0,1\}^L$, associated with the pattern $\mathcal{F}_{L,l,m,P}$, and the string length value L, is called the "set of strings exhibiting the pattern $\mathcal{F}_{L,l,m,P}$", if and only if it contains all those binary strings $x \in \{0,1\}^L$, for which $\mathcal{F}_{L,l,m,P}(x) = \mathsf{true}$ and no other:

$$\Pi(\mathcal{F}_{L,l,m,P}) \;=\; \{\; x \in \{0,1\}^L \;:\; \mathcal{F}_{L,l,m,P}(x) = \mathsf{true} \;\} \tag{3.1}$$

Similarly, for every binary string $x \in \{0,1\}^L$, we will be using the expressions "x exhibits the pattern $\mathcal{F}_{L,l,m,P}$)", "x demonstrates the pattern $\mathcal{F}_{L,l,m,P}$)", or "x contains the pattern $\mathcal{F}_{L,l,m,P}$)", to refer to the relation $x \in \Pi(\mathcal{F}_{L,l,m,P})$.

Definition 3: *Definition of an observer function.* A function $\mathcal{F}_{\mathsf{obv},L} : \{0,1\}^L \rightarrow \{\mathsf{true}, \mathsf{false}\}$ associated with an input string length value L, is called an observer function if and only if it accepts as input a binary string from the set $\{0,1\}^L$ and outputs one of two values true or false.

We let $\Pi(\mathcal{F}_{\mathsf{obv},L})$ denote the set of all binary strings in $\{0,1\}^L$ for which $\mathcal{F}_{\mathsf{obv},L}(x) = \mathsf{true}$, and call such set the set of strings "satisfying the observer function $\mathcal{F}_{\mathsf{obv},L}(x)$". Similarly, for every

binary string $x \in \{0,1\}^L$, we will be using the expression "x satisfies the observer function $\mathcal{F}_{L,l,m,P}$", or "x demonstrates unusual behavior according to observer function $\mathcal{F}_{L,l,m,P}$", to refer to the relation $x \in \Pi(\mathcal{F}_{\mathsf{obv},L})$.

An observer function may or may not be a pattern. Though the analysis that follows applies to all observer functions, the most interesting applications of the theory are those for which the observer functions are patterns.

3.2 Random Oracles according to Observer functions (RO^2)

We continue our discussion presenting the concept random oracles according to observer functions.

Definition 4: *Random Oracle according to an Observer function* (RO^2). Let $\{y_0^{(0)}, y_1^{(0)}, \ldots, y_{m_0-1}^{(0)}\}$, $\{y_0^{(1)}, y_1^{(1)}, \ldots, y_{m_1-1}^{(1)}\}$, ... be sets of binary strings of length L, the cardinalities of which satisfy $m_i \leqslant \mathcal{B}$, $\forall i \geqslant 0$, for some value $\mathcal{B}$. Let also $\mathcal{F}_{\mathsf{obv},L}$ be an observer function associated with inputs of length L, and $\mathcal{O} \leftarrow 2^\infty$ a random oracle. A cryptographic system $\mathcal{S}$ with input output characteristics $\mathcal{S} : \{0,1\}^L \rightarrow \{0,1\}^L$ is called a "random oracle according to observer function $\mathcal{F}_{\mathsf{obv},L}$", associated with a lifetime value $\mathcal{B}$ and indistinguishability parameter ϵ if the following conditions are simultaneously true:

(i) $\forall\ y_i^{(k)}, y_j^{(k)} \in \{y_0^{(k)}, y_1^{(k)}, \ldots, y_{m_k-1}^{(k)}\}$, $i \neq j$, $k \geqslant 0 : y_i^{(k)} \neq y_j^{(k)}$;

(ii) if $y \in \Pi(\mathcal{F}_{\mathsf{obv},L})$, $y \in \{y_0^{(k)}, y_1^{(k)}, \ldots, y_{m_k-1}^{(k)}\}$, $k \geqslant 0$, then $\forall\ l \neq k$: $y \notin \{y_0^{(l)}, \ldots, y_{m_l-1}^{(l)}\}$;

(iii) $\forall\ y, y_0, \ldots, y_{q-1}$, $y \neq y_0, \ldots, y \neq y_{q-1}$, $y \in \{y_0^{(k)}, y_1^{(k)}, \ldots, y_{m_k-1}^{(k)}\}$, $y_i \in \{y_0^{(l_i)}, y_1^{(l_i)}, \ldots, y_{m_{l_i}-1}^{(l_i)}\}$, $k \geqslant 0$, $l_i \geqslant 0$, $0 \leqslant i < q$ and $0 \leqslant q \leqslant \mathcal{B}$, it holds that:

$$\text{Prob}[\, \mathcal{S}(y) \in \Pi(\mathcal{F}_{\mathsf{obv},L}) \mid y_0, \mathcal{S}(y_0), \ldots, y_{q-1}, \mathcal{S}(y_{q-1})] \leqslant P_{\mathcal{F}_{\mathsf{obv},L}} \cdot 2^\epsilon \tag{3.2}$$

where $P_{\mathcal{F}_{\mathsf{obv},L}} = \text{Prob}[\, \mathsf{trunc}_L(\mathcal{O}(y)) \in \Pi(\mathcal{F}_{\mathsf{obv},L}))\,]$ and function $\mathsf{trunc}_L()$ truncates its input returning the input's L most significant bits. The probability value $P_{\mathcal{F}_{\mathsf{obv},L}}$ is referred to as "observation probability" associated with $\mathcal{F}_{\mathsf{obv},L}$. If a cryptographic system $\mathcal{S} : \{0,1\}^L \rightarrow \{0,1\}^L$ is a random oracle according to observer function $\mathcal{F}_{\mathsf{obv},L}$, associated with lifetime $\mathcal{B}$ and indistinguishability parameter ϵ, we will be denoting this fact as $\mathcal{S} \in \mathsf{RO}^2(\mathcal{F}_{\mathsf{obv},L}, \mathcal{B}, \epsilon)$.

In Definition 4, we consider observer functions, which may be patterns, that search cryptographic system outputs in order to detect abnormal behavior, such as repetitions of values of different sizes. The concept is illustrated in Figure 2. Function $\mathcal{F}_{\mathsf{obv},L}$ observes unusual behavior in the values of the output of a random oracle with probability $P_{\mathcal{F}_{\mathsf{obv},L}}$. The same function $\mathcal{F}_{\mathsf{obv},L}$ observes the same unusual behavior in the values of the output of the real system $\mathcal{S}$ with probability $P^{\mathcal{S}}_{\mathcal{F}_{\mathsf{obv},L}} \leftarrow \text{Prob}[\, \mathcal{S}(y) \in \Pi(\mathcal{F}_{\mathsf{obv},L}) \mid y_0, \mathcal{S}(y_0), \ldots, y_{q-1}, \mathcal{S}(y_{q-1})\,]$. If $P^{\mathcal{S}}_{\mathcal{F}_{\mathsf{obv},L}} \leqslant P_{\mathcal{F}_{\mathsf{obv},L}} \cdot 2^\epsilon$ for a given maximum non-repeating input sequence of size $\mathcal{B}$, and if this relation holds even when the event $\mathcal{S}(y) \in \Pi(\mathcal{F}_{\mathsf{obv},L})$ is conditioned upon inputs $y_0, \ldots, y_{q-1}$ and their responses, then the system $\mathcal{S}$ is a random oracle according to the observer function $\mathcal{F}_{\mathsf{obv},L}$.

Cryptographic systems or constructions, which are random oracles according to observer functions, are not indistinguishable from ideal primitives. In fact, it may be possible to distinguish an RO^2 system from a random function or a random permutation. However, there is a notion of indistinguishability in Definition 4. An RO^2 system is indistinguishable from an

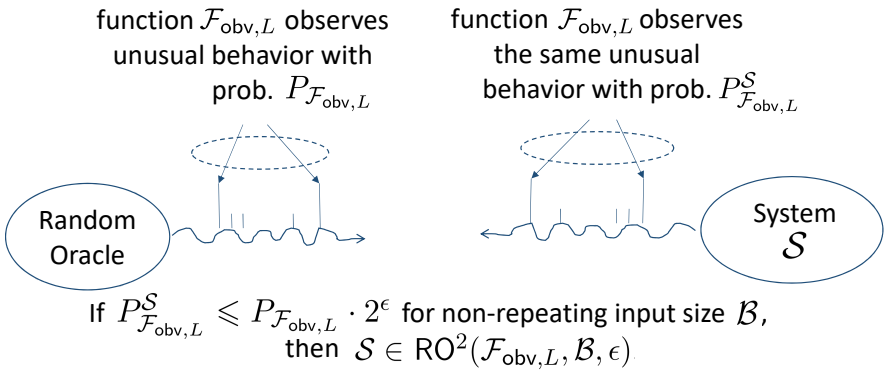


Figure 2: The Concept of a Random Oracle according to an Observer function (RO^2)

ideal primitive, only with respect to specific distinguishers. Such distinguishers are observer functions, and may be searching for patterns frequently encountered among user data. The notion of security we discuss in this paper is based on this property of RO^2 systems. Due to the indistinguishability of RO^2 systems according to Definition 4, it is difficult for an adversary to compute inputs, the outputs of which exhibit patterns, if the internal secrets of RO^2 systems are unknown.

Furthermore, systems which are RO^2 may be capable of supporting confidentiality, as modes of block ciphers do. Even though RO^2 systems are not indistinguishable from ideal primitives, as discussed, they can produce outputs consisting of indistinguishable blocks. Each block output of an RO^2 system may be indistinguishable from the output of a random permutation or a random function, even though the output consisting of the concatenated blocks may not be. This is the notion of confidentiality characterizing block cipher modes as well. The RO^2 systems, which produce such outputs, are of particular interest in this study, as they support both cryptographic integrity, by applying the implicit integrity methodology, and confidentiality, due to the indistinguishability of their block outputs. The IVP construction discussed is an example.

The query bound $\mathcal{B}$ denotes the lifetime of the construction. In order for a construction to be RO^2 it needs to satisfy the condition $P^{\mathcal{S}}_{\mathcal{F}_{\mathsf{obv},L}} \leqslant P_{\mathcal{F}_{\mathsf{obv},L}} \cdot 2^{\epsilon}$ for non-repeating input. So what does non-repeating input mean? From Definition 4, it follows that having non-repeating input means:

- within a lifetime of a construction $\{y_0, y_1, \ldots, y_{\mathcal{B}-1}\}$ input is not repeating, i.e., $y_i \neq y_j \; \forall i, j \in [0, \mathcal{B}-1]$; and
- inputs that result in unusual behavior in one lifetime $\{y_0, y_1, \ldots, y_{\mathcal{B}-1}\}$ of a construction are not repeated in any other lifetime $\{z_0, z_1, \ldots, z_{\mathcal{B}-1}\}$ of the same construction.

Whereas the inputs to constructions that are in the class RO^2 can be from any set of values, the constructions are most useful when the inputs considered are adversary queries, i.e., corrupted ciphertext values. In this case, the conditions of non-repeating input are not as restrictive as they sound. The intuition behind introducing these conditions, which are used in the derivation of our main results below, is that if an adversary repeats queries as part of the attack strategy, then the system provides the same output for these repeated queries. Specifically, we consider that there are no parameters, potentially randomizing the system

which are out of the adversary's control. Under this assumption it is clear that it is not beneficial for an adversary to repeat queries which are unsuccessful, as their result will be the same. On the other hand, if some queries are successful, then the adversary does not need to repeat these queries, as the adversary possesses the knowledge about the impact of such queries. Because of these reasons, it is not restrictive to introduce the non-repeating input requirement, as we consider adversaries that do not repeat their queries to the construction. We also note that we do not restrict every input to the construction. We just restrict only the inputs for which the condition $P^{\mathcal{S}}_{\mathcal{F}_{\mathsf{obv},L}} \leqslant P_{\mathcal{F}_{\mathsf{obv},L}} \cdot 2^{\epsilon}$ needs to be satisfied.

We conclude, noting that the indistinguishability of Definition 4 is expressed in a multiplicative form as opposed to the typical additive form. The reason why we use the multiplicative form in Definition 4, is because such form can better express the exponential drop in security which is due to the use of a non-ideal primitive, i.e., an RO^2 system. The security offered by an RO^2 system is expressed as the security stemming from the rarity of some specific patterns in a random oracle's output, minus an exponential drop ϵ associated with the use of the RO^2 system, which is non-ideal.

3.3 Constrained $\mathbf{RO}^2$ systems

Next, we discuss what a "constrained" RO^2 system is. Such system, shown in Figure 3, employs n internal invertible functions $\mathcal{R}_0, \mathcal{R}_1, \ldots, \mathcal{R}_{n-1}$ which are randomly chosen, uniformly distributed permutations from the set $\mathcal{P}$ of all permutations $\mathcal{R} : \{0,1\}^L \rightarrow \{0,1\}^N$. These primitives are referred to as "internal random permutations". The relationship between the constrained RO^2 system's input and output length L, and the input and output length of the internal permutations N is $L = n \cdot N$.

A constrained RO^2 system is invertible itself. One direction is denoted by $\mathsf{E}^{\mathcal{R}_0,\ldots,\mathcal{R}_{n-1}}()$ and referred to as "encryption", whereas the other direction is denoted by $\mathsf{D}^{\mathcal{R}_0,\ldots,\mathcal{R}_{n-1}}() = \mathsf{E}^{\mathcal{R}_0,\ldots,\mathcal{R}_{n-1}}()^{-1}$ and referred to as "decryption". A constrained RO^2 system accepts a binary input string $y \in \{0,1\}^L$ and employs an invertible polynomial time algorithm $\mathsf{Pre}()$ to perform preprocessing on the string y. The output of algorithm $\mathsf{Pre}()$ is a string of length $L = n \cdot N$, which contains the inputs to the internal random permutations $\mathcal{R}_0, \ldots, \mathcal{R}_{n-1}$. The internal random permutations $\mathcal{R}_0, \ldots, \mathcal{R}_{n-1}$ provide responses, which, after concatenation, form a response vector r. The response vector r is then passed by the RO^2 system to a postprocessing algorithm. Postprocessing is performed by a second invertible polynomial time algorithm $\mathsf{Post}()$.

In the remaining sections of the paper, we will be focusing only on constrained RO^2 systems, omitting the term "constrained", if implied. Furthermore, if $\mathsf{E}^{\mathcal{R}_0,\ldots,\mathcal{R}_{n-1}}()$ and $\mathsf{D}^{\mathcal{R}_0,\ldots,\mathcal{R}_{n-1}}()$ are RO^2 systems associated with an observer function $\mathcal{F}_{\mathsf{obv},L}$, a query bound $\mathcal{B}$ and an indistinguishability parameter ϵ, we will be denoting this fact as:

$$\mathsf{E}^{\mathcal{R}_0,\ldots,\mathcal{R}_{n-1}}(), \mathsf{D}^{\mathcal{R}_0,\ldots,\mathcal{R}_{n-1}}() \in \mathsf{RO}^2(\mathcal{F}_{\mathsf{obv},L}, \mathcal{B}, \epsilon) \tag{3.3}$$

Proposition 1: The set of constrained RO^2 systems is non-empty. Indeed, at least one construction has been proposed and proven to be in this class. This is IVP, which is discussed in this technical report.

We further note that if algorithm $\mathsf{Pre}()$ is replaced by the identity function and the data path of Figure 3 implements a decryption operation, then the ciphertext of such system is obtained directly from the output of random permutations $\mathcal{R}_0^{-1}(), \ldots, \mathcal{R}_{n-1}^{-1}()$. This means that the system does not compromise the confidentiality offered by $\mathcal{R}_0, \ldots, \mathcal{R}_{n-1}$ in any way, at the block granularity.

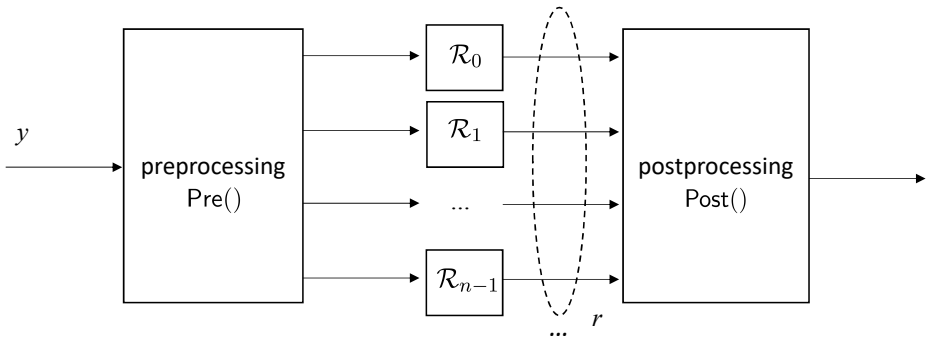


Figure 3: Internal structure of a constrained RO^2 construction

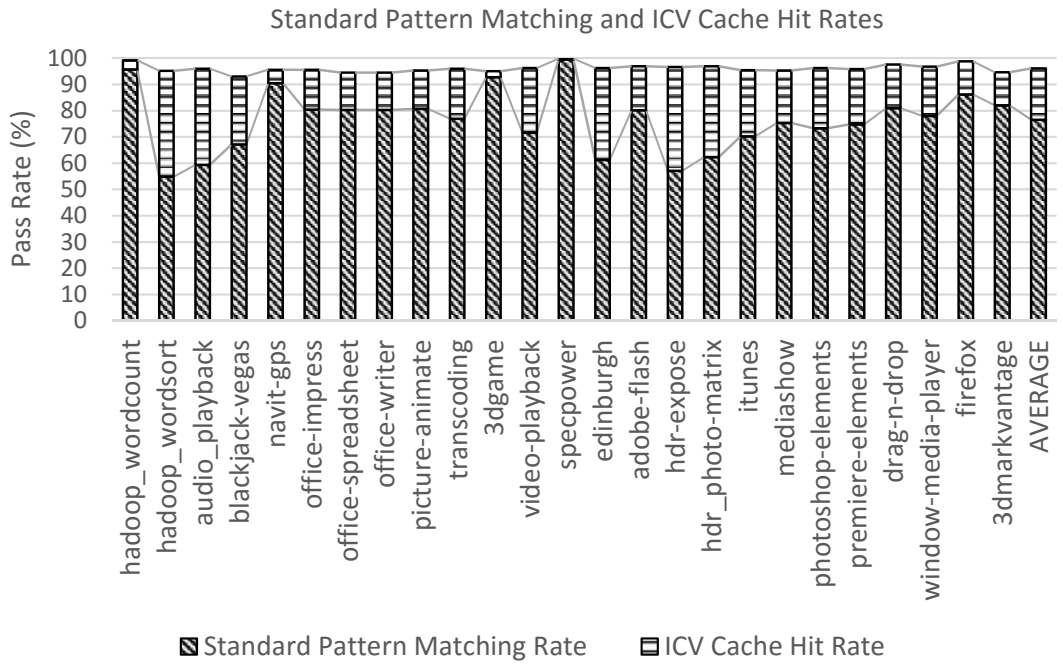
3.4 Pattern frequency observers

The pattern frequency observers, discussed in this technical report, output "true" if a number of values that are equal to each other, or exhibit certain properties, from a given input set, exceed a threshold. Value types can range and may include nibbles, bytes, words and double words. The reason why we study these functions in this paper, is because such functions have experimentally been proven successful in characterizing the overwhelming majority of typical uncompressed, unencrypted client and server data. We have observed that client and server data demonstrate repetitions of values of different types, which in truly random data (i.e, random oracle outputs) appear with very low probability. It is these observations that motivate the implicity integrity work.

The fact that uncompressed, unencrypted user data demonstrate patterns should not come as a surprise. User data often consist of code, data structures, media data, pointer tables, and other types of structured data that are characterized by significant redundancy. For example, there exists a simple pattern which is frequently encountered in client and server data. This is the appearance of 4 or more 16-bit words which are equal to each other in a collection of 32 words. In this pattern the input size is 512 bits (i.e., each data is a memory cache line). An observer function which detects the presence of such pattern in inputs of size $L = 512$ bits is denoted by $\mathcal{F}_{\mathsf{EQ},512,16,4}$. Our experimental observations come from over 111 million client cache lines and 1.47 billion server cache lines of typical workloads. According to these observations, the $\mathcal{F}_{\mathsf{EQ},512,16,4}$ pattern characterizes 82% of the client cache lines and 78% of the server cache lines.

Pass rate comparisons associated with different pattern detectors are shown in Figure 4. Pattern detection based on the observer function $\mathcal{F}_{\mathsf{EQ},512,16,4}$ is referred to as 'Standard Pattern Matching' (SPM). Pattern detection based on the equality of words as well as other types of data such as bytes, double words and nibbles is referred to as 'Extended Pattern Matching' (EPM). As is evident from the figure, there are many typical client workloads (e.g., Microsoft® Office, transcoding, video player), the pass rates of which are quite high, ranging between 75%-80%, when Standard Pattern Matching is employed. These pass rates are boosted to 98% when Extended Pattern Matching is employed. Overall, the average client pass rates associated with Standard Pattern Matching are 82%. The average pass rates associated with Extended Pattern Matching are 91%. For server data, the corresponding pass rates are 78% and 84% respectively.

The EPM scheme encompasses many more pattern detectors than SPM. Pattern frequency observers, associated with the EPM scheme detect entities among the input data which are not only equal to each other, but are also placed in continuous index positions. These observers are not necessarily the same as those detecting value equalities. One can associate these two



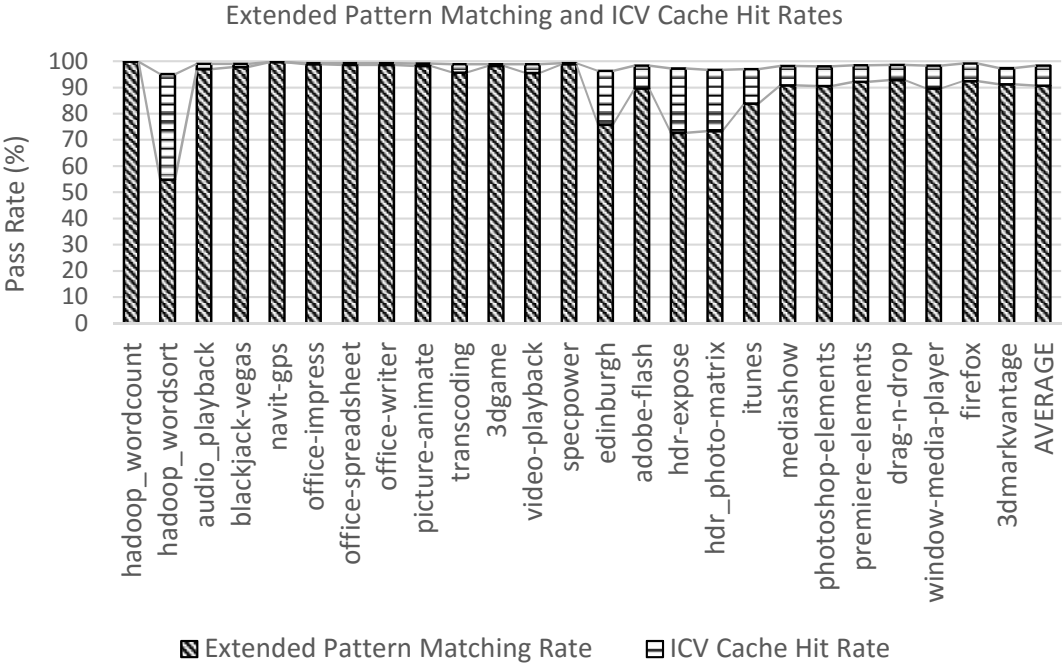


Figure 4: Pass rates associated with Standard and Extended Pattern Matching on client workload cache lines

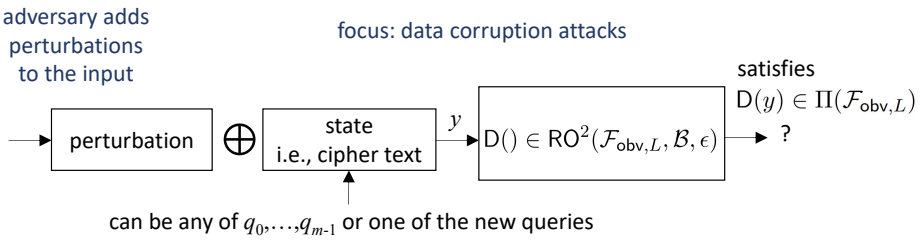


Figure 5: Input perturbing adversary

types of observer functions with different thresholds and, by doing so, build two different integrity schemes. Yet another type of observer function detects entities that take special values. Special values are values that are frequently encountered in regular user data but are infrequently encountered in random or corrupted plaintext data. For example, in memory cache lines obtained from client and server data workloads, a high percentage of bytes take the values of 0x00 or 0xFF.

A last type of observer functions used by EPM detects entities the value histogram of which demonstrates a sum of n highest entries (i.e., frequencies) being higher than a threshold. The intuition behind this type of pattern check is that there are several types of input messages, the content of which is not as random as that of encrypted data, but also does not demonstrate patterns at the byte or word granularity. One example of such content is media data, where nibble values may be replicated, but data do not demonstrate significant byte or word replications. Our experimental studies showed that there are millions of cache lines demonstrating a limited set of byte equalities but many nibble equalities. By checking whether the sum of the two highest nibble frequencies exceeds a threshold, a more flexible pattern detector can be built, which on the one hand encompasses significantly more regular user inputs, and on the other hand is associated with an event that is infrequent among random data.

Figure 4 also shows Integrity Check Value (ICV) cache hit rates for these algorithms. It is assumed that not all memory cache lines are protected visa implicit integrity, as discussed above, but some are protected using standard methods. For these, ICVs or MACs are cached. The hit rates of Figure 4 are obtained from a simulator implementing the caching of 32-bit ICVs using a 4KB cache unit for this purpose. When ICVs are cached the total pass rate observed by SPM and EPM are boosted significantly. For Standard Pattern Matching, the total pass rate, which includes a pattern check pass rate and an ICV cache hit rate, is 97%. For Extended pattern matching this rate is increased to 99%.

4 Adversary models and main security claims

4.1 The input perturbing adversary

The first type of adversary presented, describes adversaries which aim in corrupting encrypted data that are stored somewhere, or are in transit, in such a way so that the corruptions pass undetected. We refer to such type of adversary as "input perturbing" adversary.

To present the adversary, we first describe the system which is attacked. The system, which is attacked implements the procedures ReadImplicit() and WriteImplicit() below. Procedure ReadImplicit() accepts as input an address value A, a data array DataArray[], which may contain data stored in memory, in permanent storage, or transmitted as part of a communication session, an array of per address message authentication codes MACArray[], an observer function

$\mathcal{F}_{\mathsf{obv},L}()$, a decryption oracle $\mathsf{D}^{\mathcal{R}_0,\ldots,\mathcal{R}_{n-1}}()$, and a MAC verification function Verify(). The observer function and decryption oracle satisfy (3.3) for some $\mathcal{B}$, ϵ. In line 1, procedure ReadImplicit() reads data y from address A of the array DataArray[ ]. In line 2, it decrypts the data using the oracle $\mathsf{D}^{\mathcal{R}_0,\ldots,\mathcal{R}_{n-1}}()$. In line 3, it applies the observer function $\mathcal{F}_{\mathsf{obv},L}()$ on the decrypted data x to determine if data x satisfies the observer function. If the response is true, then it returns x and exits in line 4. In this case, procedure ReadImplicit() determines that the decrypted data x has not been corrupted. If the response from the observer function $\mathcal{F}_{\mathsf{obv},L}()$ is not true, a message authentication code for y is read in line 6 and verified in line 7. If the response from function Verify() is true, then procedure ReadImplicit() returns x and exits in line 8. Otherwise, it returns an empty string in line 10.

Procedure WriteImplicit() accepts as input an address value A, a data value x, the array DataArray[ ], the array of per address message authentication codes MACArray[ ], the observer function $\mathcal{F}_{\mathsf{obv},L}()$, an encryption oracle $\mathsf{E}^{\mathcal{R}_0,\ldots,\mathcal{R}_{n-1}}()$, and a MAC sign function Sign(). The observer function and encryption oracle satisfy (3.3) for the same $\mathcal{B}$, ϵ as those of procedure ReadImplicit(). Furthermore $\mathsf{E}^{\mathcal{R}_0,\ldots,\mathcal{R}_{n-1}}() = \mathsf{D}^{\mathcal{R}_0,\ldots,\mathcal{R}_{n-1}}()^{-1}$. In line 1, procedure WriteImplicit() encrypts data x using the oracle $\mathsf{E}^{\mathcal{R}_0,\ldots,\mathcal{R}_{n-1}}()$. In line 2, it writes the encrypted data y at address A of the array DataArray[ ]. In line 3, it applies the observer function $\mathcal{F}_{\mathsf{obv},L}()$ on the data x to determine if data x satisfies the observer function. If the response is true, then procedure WriteImplicit() exits in line 4. Otherwise, it computes a message authentication code z for the encrypted data y, which writes into MACArray[ ] in line 7, exiting in line 8.

ReadImplicit(A, DataArray[ ], MACArray[ ], $\mathcal{F}_{\mathsf{obv},L}()$, $\mathsf{D}^{\mathcal{R}_0,\ldots,\mathcal{R}_{n-1}}()$, Verify())

```
1. y ← DataArray[A]
2. x ← D^{R_0,...,R_{n-1}}(y)
3. if F_{obv,L}(x) = true
4.       return x
5. else
6.       z ← MacArray[A]
7.       if Verify(y, z) = true
8.             return x
9.       else
10.            return ⊥
```

WriteImplicit(A, x, DataArray[ ], MACArray[ ], $\mathcal{F}_{\mathsf{obv},L}()$, $\mathsf{E}^{\mathcal{R}_0,\ldots,\mathcal{R}_{n-1}}()$, Sign())

```
1. y ← E^{R_0,...,R_{n-1}}(x)
2. DataArray[A] ← y
3. if F_{obv,L}(x) = true
4.       return
5. else
6.       z ← Sign(y)
7.       MacArray[A] ← z
8.       return
```

The input perturbing adversary models any software process or physical intruder that aims in intentional corruptions of data. The underlying assumption of this adversary model is that the adversary can access data in their encrypted form. Specifically, the adversary can observe the contents of the array DataArray[ ] of the pseudocode, but cannot access the encryption key

which has produced the data. This adversary can corrupt ciphertext data in any possible way hoping that the corruptions will result in plaintexts with patterns, and thus pass undetected.

This adversary model is not unrealistic. In secure network connections or encrypted storage systems, many attacks originate from sources outside of these trusted domains (e.g., malware running in different processes or virtualized environments, untrusted hypervisors, man-in-the-middle attackers intercepting the packets of secure connections etc.) These attackers can possibly inspect a range of encrypted data, such as the whole encrypted memory of a computing system, but do not have access to the encryption keys required for obtaining the corresponding plaintexts. Thus, these attackers are unable to corrupt user data in a straightforward manner, such as directly changing plaintext bytes from 0x00 to 0xFF. Attackers can only do this through the modification of ciphertexts, where ciphertexts are produced using keys unknown to the attackers.

Definition 5: *Input perturbing adversary.* Let's consider a pair of encryption and decryption oracles $\mathsf{E}()$ and $\mathsf{D}()$ such that $\mathsf{E}() = \mathsf{D}^{-1}()$ and for which $\mathsf{E}(), \mathsf{D}() \in \mathsf{RO}^2(\mathcal{F}_{\mathsf{obv},L}, \mathcal{B}, \epsilon)$ for some $\mathcal{F}_{\mathsf{obv},L}, \mathcal{B}, \epsilon$. The oracle invocations to $\mathcal{R}_0, \ldots, \mathcal{R}_{n-1}$ are omitted from the notation for the sake of conciseness. An input perturbing adversary $\mathcal{A}^{\mathsf{D}()}_{\mathsf{INP}}(q_0, \ldots, q_{m-1}, Q_{\mathcal{B}'})$ is defined as a polynomial time algorithm for which the following hold:

- The adversary has oracle access to $\mathsf{D}()$.
- The adversary can observe m ciphertexts $q_0, \ldots, q_{m-1}$ stored in the array $\mathsf{DataArray}[\,]$, which, prior to his game have been submitted to $\mathsf{D}()$. These are referred to as "prior queries". The oracle responses $\mathsf{D}(q_0), \ldots, \mathsf{D}(q_{m-1})$ satisfy the observer function $\mathcal{F}_{\mathsf{obv},L}$, i.e., $\mathsf{D}(q_0), \ldots, \mathsf{D}(q_{m-1}) \in \Pi(\mathcal{F}_{\mathsf{obv},L})$. The adversary has knowledge of that, as well as knowledge of the oracle responses $\mathsf{D}(q_0), \ldots, \mathsf{D}(q_{m-1})$ themselves.
- The adversary can issue at most $\mathcal{B}'$ new, non-repeating queries from a set $Q_{\mathcal{B}'}$ to $\mathsf{D}()$ as part of his game, where $|Q_{\mathcal{B}'}| = \mathcal{B}'$, $m + \mathcal{B}' \leqslant \mathcal{B}$, and each new query does not need to be one of $q_0, \ldots, q_{m-1}$. For each of the new queries, the adversary observes the corresponding plaintext, as well as whether the plaintext satisfies $\mathcal{F}_{\mathsf{obv},L}$ or not.

The algorithm succeeds if it finds a new input data word y which is different from $q_0, \ldots, q_{m-1}$, as well as different from the new queries, the output of which satisfies the observer function $\mathcal{F}_{\mathsf{obv},L}$. The input perturbing adversary is shown in Figure 5. The advantage of the input perturbing adversary is defined as:

$$\begin{aligned}\mathbf{Adv}(\mathcal{A}^{\mathsf{D}()}_{\mathsf{INP}}(q_0, \ldots, q_{m-1}, Q_{\mathcal{B}'}), \mathcal{F}_{\mathsf{obv},L}) = \mathrm{Prob}[\, y \leftarrow \mathcal{A}^{\mathsf{D}()}_{\mathsf{INP}}(q_0, \ldots, q_{m-1}, Q_{\mathcal{B}'}); \\ y \notin \{q_0, \ldots, q_{m-1}\};\ y \notin Q_{\mathcal{B}'};\ \mathsf{D}(y) \in \Pi(\mathcal{F}_{\mathsf{obv},L})\,]\end{aligned} \tag{4.1}$$

In the paper, we argue that the game of adversary $\mathcal{A}^{\mathsf{D}()}_{\mathsf{INP}}()$ is difficult to win, if the oracle $\mathsf{D}()$ is in the class of RO^2 systems, associated with observer function $\mathcal{F}_{\mathsf{obv},L}$, lifetime $\mathcal{B}$, and indistinguishability parameter ϵ. This happens because such systems output values that satisfy their associated observer function with very low probability, which is quantifiable. Furthermore, such probability does not depend on the previous history of inputs, as long as the inputs which satisfy the observer function are not repeated. Moreover, as will be made clear, the prior queries $q_0, \ldots, q_{m-1}$, as well as new queries end up not helping adversary $\mathcal{A}^{\mathsf{D}()}_{\mathsf{INP}}()$ win his game at all.

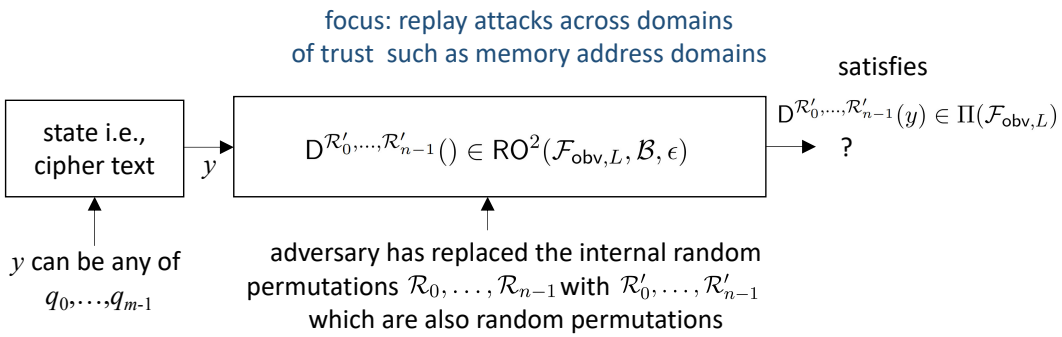


Figure 6: Oracle replacing adversary

Another observation is that the input perturbing adversary, as defined by Definition 5, diverges from the classical MAC adversary definition. The new adversary definition applies to the cases where data integrity is supported by applying the implicit integrity methodology. The input perturbing adversary of Definition 5 creates forgeries by crafting ciphertexts, the plaintexts of which exhibit certain patterns. In this way, forgeries created by the adversary of Definition 5 are successful, not because the adversary supplies a carefully crafted MAC, but because the adversary supplies a carefully created ciphertext, the plaintext of which exhibits patterns. The fact that the word y, returned by the input perturbing adversary, constitutes a forgery follows directly from the definition of procedures ReadImplicit() and WriteImplicit().

One special case of attacks performed by this adversary but not the only one studied in this paper, includes attacks where the query budget $\mathcal{B}'$ is tight. These are the cases of online attacks where a bounded number of failures from the adversary's side exposes the attack (e.g., $\mathcal{B}'$ cannot exceed 2^{32}). These attacks are further discussed in section 5.

4.2 Oracle replacing adversary

Another type of adversary, the "oracle replacing adversary", shown in Figure 6 is associated with replay attacks. Replay attacks may happen across key domains, such as network sessions that are encrypted with different keys, or encrypted memory domains. The oracle replacing adversary, as described in the section, performs replay attacks under the assumption that the internal random permutations of an RO^2 system are a unique property of the system and of a domain of trust.

Specifically, we consider that the array DataArray[ ] of the pseudocode of procedures ReadImplicit() and WriteImplicit() does not contain data encrypted in the same manner, but instead is segmented into different domains. The data of each domain is encrypted using oracles E() and D(), which are RO^2 systems. These oracles invoke a different set of internal random permutations when performing the encryptions and decryptions of each different domain. The internal random permutations of different domains are independently drawn and uniformly distributed in the set $\mathcal{P}$ of all permutations $\mathcal{R} : \{0,1\}^L \rightarrow \{0,1\}^N$.

Given the above, it is evident that when some valid encrypted data is replayed from one domain to another domain, then this is equivalent to replacing the internal random permutations of an RO^2 system and domain with a different set of internal random permutations, and then performing a decryption operation. In our analysis, the internal random permutations of domains are ideal primitives, which in practical realizations are implemented as ciphers using different keys and tweaks.

Definition 6: *Oracle replacing adversary.* Let's consider a pair of encryption and decryption oracles $\mathsf{E}()$ and $\mathsf{D}()$ such that $\mathsf{E}() = \mathsf{D}^{-1}()$, and for which $\mathsf{E}(), \mathsf{D}() \in \mathsf{RO}^2(\mathcal{F}_{\mathsf{obv},L}, \mathcal{B}, \epsilon)$ for some $\mathcal{F}_{\mathsf{obv},L}, \mathcal{B}, \epsilon$. For a particular domain, the oracles have access to internal random permutations $\mathcal{R}_0, \ldots, \mathcal{R}_{n-1}$. An oracle replacing adversary $\mathcal{A}^{\mathsf{D}()}_{\mathsf{REPL}}(q_0, \ldots, q_{m-1}, Q_{\mathcal{B}'}, \mathcal{R}_{\mathcal{D}})$ is defined as a polynomial time algorithm for which the following hold:

- The adversary has oracle access to $\mathsf{D}()$.
- The adversary can observe m ciphertexts $q_0, \ldots, q_{m-1}$ stored in the domain associated with permutations $\mathcal{R}_0, \ldots, \mathcal{R}_{n-1}$, and $m_{\mathcal{D}}$ ciphertexts of set $Q_{\mathcal{D}}$ stored in other domains. The observed ciphertexts are referred to as "prior queries". Prior queries are submitted before the beginning of the adversary's game. The oracle responses to prior queries, including $\mathsf{D}^{\mathcal{R}_0,\ldots,\mathcal{R}_{n-1}}(q_0), \ldots, \mathsf{D}^{\mathcal{R}_0,\ldots,\mathcal{R}_{n-1}}(q_{m-1})$, satisfy the observer function $\mathcal{F}_{\mathsf{obv},L}$. The adversary has knowledge of that, as well as the oracle responses themselves.
- The adversary is aware of the existence of a set $\mathcal{R}_{\mathcal{D}} = \{\{\mathcal{R}_0^{(0)}, \ldots, \mathcal{R}_{n-1}^{(0)}\}, \ldots, \{\mathcal{R}_0^{(n_{\mathcal{D}}-1)}, \ldots, R_{n-1}^{(n_{\mathcal{D}}-1)}\}\}$ of $n_{\mathcal{D}}$ sets of internal random permutations of domains other than the target domain, which have the same input and output length characteristics as $\mathcal{R}_0, \ldots, \mathcal{R}_{n-1}$ and are all different from $\mathcal{R}_0, \ldots, \mathcal{R}_{n-1}$.
- can issue at most $\mathcal{B}'$ new, non-repeating queries to the decryption oracles of any domain of array $\mathsf{DataArray}[\,]$ from a set $Q_{\mathcal{B}'}$, as part of his game. Prior queries are not repeated on the same domain. Furthermore, it should hold that $|Q_{\mathcal{B}'}| = \mathcal{B}'$, $m + \mathcal{B}' \leqslant \mathcal{B}$ and $m_{\mathcal{D}} + \mathcal{B}' \leqslant \mathcal{B}$. For each of the new queries, the adversary observes the corresponding plaintext, as well as whether the plaintext satisfies $\mathcal{F}_{\mathsf{obv},L}$ or not.

The algorithm succeeds if it finds a set of new internal random permutations $\{\mathcal{R}'_0, \ldots, \mathcal{R}'_{n-1}\} \in \mathcal{R}_{\mathcal{D}}$ and an input ciphertext $y \in \{q_0, \ldots, q_{m-1}\}$, the output of which satisfies the observer function $\mathcal{F}_{\mathsf{obv},L}$ when y is applied on an instance of $\mathsf{D}()$ that invokes the new internal random permutations $\mathcal{R}'_0, \ldots, \mathcal{R}'_{n-1}$, i.e., $\mathsf{D}^{\mathcal{R}'_0,\ldots,\mathcal{R}'_{n-1}}(y) \in \Pi(\mathcal{F}_{\mathsf{obv},L})$. Oracle $\mathsf{D}^{\mathcal{R}'_0,\ldots,\mathcal{R}'_{n-1}}()$ should not be queried with input y as part of the adversary's game. The oracle replacing adversary is shown in Figure 6. The advantage of the oracle replacing adversary is defined as:

$$
\begin{aligned}
&\mathbf{Adv}(\mathcal{A}^{\mathsf{D}}_{\mathsf{REPL}}(q_0, \ldots, q_{m-1}, Q_{\mathcal{D}}, Q_{\mathcal{B}'}, \mathcal{R}_{\mathcal{D}}), \mathcal{F}_{\mathsf{obv},L}) = \\
&\qquad \mathrm{Prob}[\, \{\{\mathcal{R}'_0, \ldots, \mathcal{R}'_{n-1}\},\ y\} \leftarrow \mathcal{A}^{\mathsf{D}}_{\mathsf{REPL}}(q_0, \ldots, q_{m-1}, Q_{\mathcal{D}}, Q_{\mathcal{B}'}, \mathcal{R}_{\mathcal{D}}); \\
&\qquad y \in \{q_0, \ldots, q_{m-1}\};\ \{\mathcal{R}'_0, \ldots, \mathcal{R}'_{n-1}\} \in \mathcal{R}_{\mathcal{D}}; \\
&\qquad \text{oracle } \mathsf{D}^{\mathcal{R}'_0,\ldots,\mathcal{R}'_{n-1}}() \text{ has not been queried with input } y; \\
&\qquad \mathsf{D}^{\mathcal{R}'_0,\ldots,\mathcal{R}'_{n-1}}(y) \in \Pi(\mathcal{F}_{\mathsf{obv},L})\,]
\end{aligned}
\tag{4.2}
$$

4.3 Security of RO^2 systems in the input perturbing and oracle replacing adversary models

The following two theorems connect the concept of a random oracle according to an observer function with security claims associated with the input perturbing and oracle replacing adversary models.

Theorem 1: *About the security of an* RO^2 *system in the input perturbing adversary model.* Given a pair of encryption and decryption oracles $\mathsf{E}()$ and $\mathsf{D}()$, $\mathsf{D}() = \mathsf{E}()^{-1}$, an observer

function $\mathcal{F}_{\mathsf{obv},L}$, a lifetime $\mathcal{B}$, a set of query bounds $\{m, \mathcal{B}'\}$, and an indistinguishability parameter ϵ such that: $\mathsf{E}(), \mathsf{D}() \in \mathsf{RO}^2(\mathcal{F}_{\mathsf{obv},L}, \mathcal{B}, \epsilon)$, then for any input perturbing adversary $\mathcal{A}^{\mathsf{D}()}_{\mathsf{INP}}(q_0, \ldots, q_{m-1}, Q_{\mathcal{B}'})$:

$$\mathbf{Adv}(\mathcal{A}^{\mathsf{D}()}_{\mathsf{INP}}(q_0, \ldots, q_{m-1}, Q_{\mathcal{B}'}), \mathcal{F}_{\mathsf{obv},L}) \leqslant P_{\mathcal{F}_{\mathsf{obv},L}} \cdot 2^{\epsilon} \tag{4.3}$$

where $P_{\mathcal{F}_{\mathsf{obv},L}}$ is the observation probability associated with the observer function $\mathcal{F}_{\mathsf{obv},L}$. The proof of Theorem 1 can be found in Appendix A. A next theorem is about the security of RO^2 systems in the oracle replacing adversary model.

Theorem 2: *About the security of an* RO^2 *system in the oracle replacing adversary model.* Given a pair of encryption and decryption oracles $\mathsf{E}()$ and $\mathsf{D}()$, $\mathsf{D}() = \mathsf{E}()^{-1}$, an observer function $\mathcal{F}_{\mathsf{obv},L}$, a lifetime $\mathcal{B}$, a set of query bounds $\{m, m_{\mathcal{D}}, \mathcal{B}'\}$, and an indistinguishability parameter ϵ such that: $\mathsf{E}(), \mathsf{D}() \in \mathsf{RO}^2(\mathcal{F}_{\mathsf{obv},L}, \mathcal{B}, \epsilon)$, then for any oracle replacing adversary $\mathcal{A}^{\mathsf{D}}_{\mathsf{REPL}}(q_0, \ldots, q_{m-1}, Q_{\mathcal{D}}, Q'_{\mathcal{B}}, \mathcal{R}_{\mathcal{D}})$:

$$\mathbf{Adv}(\mathcal{A}^{\mathsf{D}}_{\mathsf{REPL}}(q_0, \ldots, q_{m-1}, Q_{\mathcal{D}}, Q_{\mathcal{B}'}, \mathcal{R}_{\mathcal{D}}), \mathcal{F}_{\mathsf{obv},L}) \leqslant P_{\mathcal{F}_{\mathsf{obv},L}} \cdot 2^{\epsilon} \tag{4.4}$$

where again, $P_{\mathcal{F}_{\mathsf{obv},L}}$ is the observation probability associated with observer function $\mathcal{F}_{\mathsf{obv},L}$. The proof of Theorem 2 can be found in Appendix B.

Theorems 1 and 2 indicate that the probability $P_{\mathcal{F}_{\mathsf{obv},L}} \cdot 2^{\epsilon}$ of observing unusual behavior in the output of an RO^2 system, associated with observer function $\mathcal{F}_{\mathsf{obv},L}$, is also the advantage of input perturbing and oracle replacing adversaries attacking the RO^2 system. Such probability depends mainly on the probability $P_{\mathcal{F}_{\mathsf{obv},L}}$ of observing the unusual behavior associated with $\mathcal{F}_{\mathsf{obv},L}$ in the output of a random oracle. Therefore, to effectively design of a cryptographic system that applies the implicity integrity methodology, one needs to search for patterns which, on the one hand characterize the overwhelming majority of client and server data, and on the other hand minimize the advantage bound $P_{\mathcal{F}_{\mathsf{obv},L}} \cdot 2^{\epsilon}$.

5 Example of a cryptographic construction supporting implicit integrity

Implicit data integrity can be supported by block ciphers. In this section we analyze a cryptographic construction which is not a block cipher, but supports confidentiality as a mode of a block cipher does, as well as implicit integrity. The cryptographic construction we discuss, called IVP, is shown in Figures 7 and 8. Figure 7 provides an overview of the encrypt and decrypt paths of the construction, while Figure 8 provides a description of the stages involved in the decrypt path. The IVP construction is a three level confusion diffusion network. It first employs two rounds of substitution and permutation stages followed by a byte remapping stage. The byte remapping stage prepares the inputs to four parallel random permutations, which provide an encryption result. On the decrypt path this order is reversed. The construction is 512-bit wide and its internal stages are defined for specific input and state lengths, which are discussed below.

The purpose of the two rounds of substitution and permutation stages is to diffuse every bit of the input into sets of 128 bits. Specifically, each bit is fully diffused into one of four sets of 128 bits. The purpose of the subsequent byte remapping stage is to change the order of bytes so that every 128-bit input, which is passed into the subsequent random permutations, contains bytes that depend on all 512 bits of the input. The random permutations of the construction can be realized using any block cipher which is 128-bit wide. For example, they

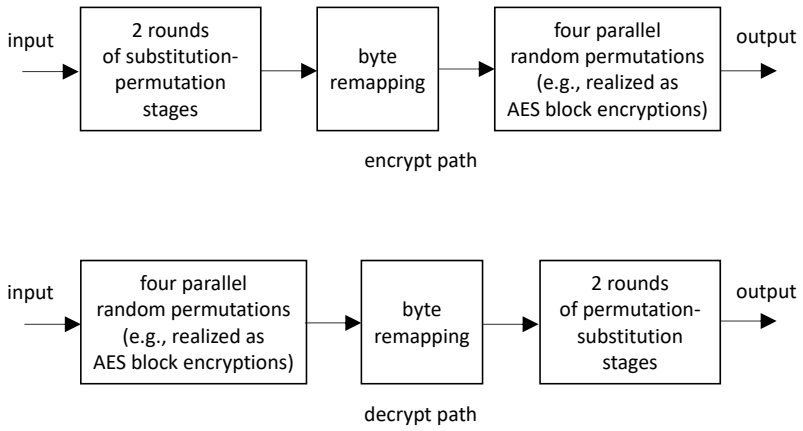


Figure 7: Overview of the encrypt and decrypt paths of the IVP construction

can be realized as four independent AES block encryption stages. The preceding substitution and permutation stages can also be realized using AES round building blocks for convenience, as discussed later in this section.

The IVP construction, as we prove later, is indeed RO^2 for the $\mathcal{F}_{\mathsf{EQ},512,16,4}$ observer function, the lifetime of $\mathcal{B} = 2^{32}$ queries and the indistinguishability parameter $\epsilon = 0.697$ bits. For the sake of clarity, we remind that we are interested in the RO^2 behavior of the decrypt path of the construction, where the inputs are provided by an entity who has no knowledge of any keys used by the construction. So, the lifetime $\mathcal{B} = 2^{32}$ is not the lifetime of a valid user of the construction but of an adversary (i.e., input perturbing or oracle replacing adversary) who attacks it.

5.1 Online attacks and their implications

IVP supports implicit integrity at a security level of 32 bits. Such security level is lower than the security levels supported by standard MAC algorithms (e.g., [4] [5]), which are typically at 256 or 512 bits. Still, 32-bit security is not insignificant in the context of online attacks. In online attacks, the detection of even a single corruption exposes the attack and the adversary. As the adversary can only corrupt data in their encrypted form, any subsequent read operation performed on corrupted data results in plaintext with high entropy with very high probability, thus exposing the attack. For this reason, even a lower level of security, such as at 32 bits, may be quite effective in protecting some computing systems. Defense against online attacks at 32 bits of security also means that the probability of the adversary succeeding one and only time is $2^{-32+\epsilon}$ for some ϵ. Moreover, a single corruption detection can cause re-encryption of all user data with new keys, thus mitigating the attack. Online attacks are discussed in RFC 4086 [34].

In the context of online attacks, a lower bound on the number of queries issued by an adversary can be set. Within such bound, not exceeding 2^{32} queries, the block cipher stages employed safely approximate truncated output random oracles. For example, if the output of an 128-bit random permutation is not truncated, then the advantage of distinguishing this random permutation from an 128-bit truncated output random oracle after 2^{32} queries are performed is still $\leqslant 2^{-64}$. Practically, this means that whereas bytes in the truncated output of a random oracle are random and uniformly distributed, bytes in the output of a random permutation take every value from 0 to 255 with probability that differs from 2^{-8} by no more than a value

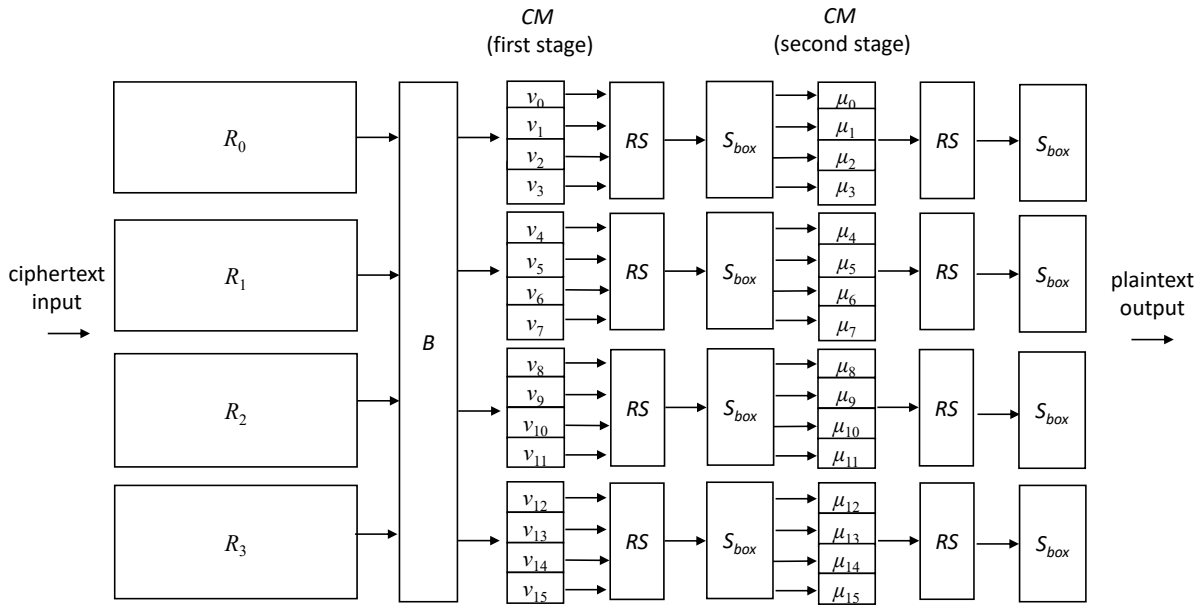


Figure 8: Description of the decrypt path of the IVP construction

$\Delta = 2^{-64}$. In the analysis that follows we will be using the term "almost" random, uniformly distributed and statistically independent to characterize bytes or words, when the statistical properties of bytes or words differ from the properties of random, uniformly distributed and statistically independent bytes or words by no more than $O(\Delta)$, where $\Delta = 2^{-64}$.

5.2 Construction stages

Figure 8 shows the decrypt path for the IVP construction. In this figure, the input is the cipher text and the output is the plaintext. In the figure, the four internal random permutations employed $R_0, \ldots, R_3$ are four symmetric encryption/decryption stages. Each stage applies on inputs of width W_1. In the specific design we propose $W_1 = 128$ bits and the four random permutations of the figure can be realized as AES encryption/decryption stages. Encryption/decryption is repeated four times, each for a separate 128-bit block of the input. If realized via an array of four block ciphers, encryption/decryption uses a key value K, which is a vector of four concatenated encryption keys, one for each block, and, optionally, a tweak vector T.

The stage B indicates an entity reordering operation. Entities are groups of W_2 bits. In this design $W_2 = 8$ bits and this stage is a byte remapping operation. Entity reordering takes place across the entire width of the construction which is $4 \cdot W_1$ bits. Entity reordering is an interleaving operation that ensures that outputs of the four internal random permutations of the construction are evenly distributed among the subsequent processing stages. Such interleaving operation is further discussed below. Two subsequent stages denoted by "CM" perform AES-like bit linear processing on their inputs, which we refer to as "column mixing". In one realization the CM stages may implement the inverse mix columns or the mix columns transformation of AES. In general, the requirement for each CM stage is to implement bit linear systems that connect m W_2-bit input entities to m W_2-bit output entities using an MDS matrix. The rank of each bit linear system for each output entity should be exactly W_2. It is easy to see that the AES mix columns and inverse mix columns transformations meet this

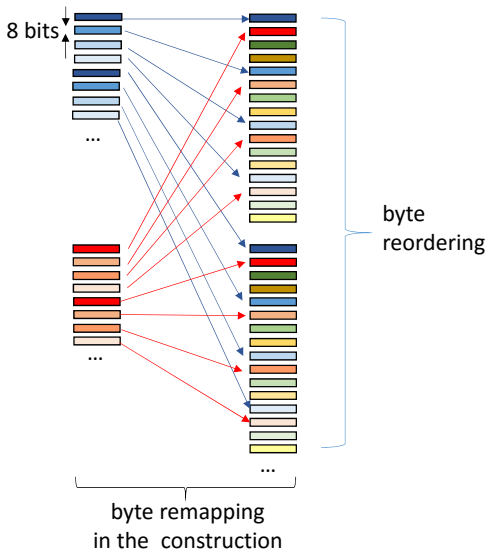


Figure 9: Byte remapping in the IVP construction

requirement for $m = 4$ and $W_2 = 8$ bits. For the proofs discussed below we require that $m = 4$. In the figure there are 16 first stage *CM* transformations denoted by $\nu_0, \ldots, \nu_{15}$ and 16 second stage *CM* transformations denoted by $\mu_0, \ldots, \mu_{15}$.

The subsequent "*RS*" stages indicate a "Row Shifting" operation which is an entity reordering operation similar to *B*. Row shifting occurs only inside W_1 bit blocks and not across such blocks as in the case of *B*. *RS* stages operate on entities of W_2 bits and perform cyclic rotation of such entities by increasing the number of rotate positions one at a time row-by-row. As in AES, it is assumed that W_2 bit entities are arranged in a matrix formation. In this formation rows are cyclically rotated either to the left or to the right by a number of positions which is increasing by one row-by-row. Row 1 for instance may be shifted by one entity position to the left. Row 2 by two entity positions to the left etc. In one realization *RS* stages implement the inverse shift rows or the shift rows transformation of AES.

Sbox stands for substitution box. Substitution occurs in the granularity of W_2 bits as in the case of the *CM* and *RS* stages. Each *Sbox* stage in the figure is an array of multiple substitution boxes of width W_2 bits. A substitution box is a randomly chosen 8-bit Pseudo-Random Permutation (PRP) which can be realized in many ways, such as by combining key additions with strong non-linear bit mixing operations. In one realization the key values used by these PRPs are set at the beginning of the operation of the IVP construction. This is equivalent to selecting a set of W_2-bit PRPs at random in the beginning of operation and using these PRPs as substitution boxes. It is under these considerations that we have proven specific security claims for the IVP construction, which we discuss in this section. Since we can select PRPs once at the beginning of operation of the construction, this means that we can have a single key set for the *Sbox* stages, which is independent of the keys used by the block ciphers which implement the random permutations $R_0, \ldots, R_3$.

The entity reordering operation *B* of the IVP construction is further illustrated in Figure 9 for the case where $W_2 = 8$ bits (byte remapping). Here, each byte is reordered to a new position so that all bytes coming from the same 128-bit block output are evenly distributed to all 128-bit block inputs of a next stage. For instance, regarding the outputs coming from a first random permutation R_0, a first byte is mapped to the position of index 0 in a next block.

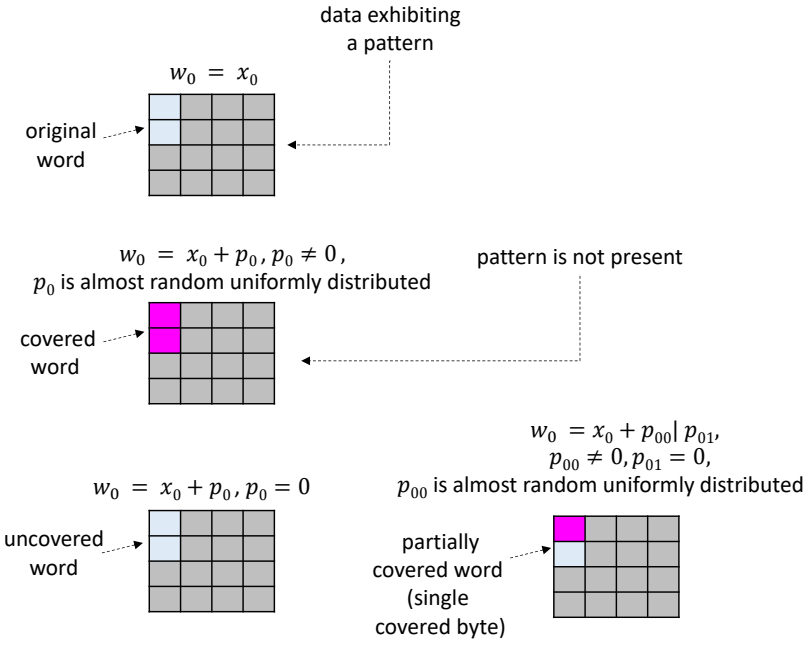


Figure 10: Covered, uncovered and partially covered words

A second byte is mapped to the position of index 4. A third byte to the position of index 8, and so on. Similarly, concerning the outputs coming from a second random permutation R_1, a first byte is mapped to the position of index 1. A second byte is mapped to the position of index 5. A third byte is mapped to the position of index 9, and so on.

To derive our byte remapping scheme we considered all possible ways to place 8 bits coming from the output of an internal random permutation into a 32-bit entity. This number which is equal to $\binom{32}{8} = 10,518,300$ is tractable and allows for the space to be searched even with exhaustive search. Each bit placement choice corresponds to a different bit linear system connecting the input bits of the CM transformation to the output bits. The bit placement choice determines which columns of the CM system matrix are selected in order to describe the input output relationship. As stated earlier, it is desirable for the rank of the system characterizing the derivation of each output byte to be exactly equal to $W_2 = 8$ bits. For the four output bytes we would like to have a cumulative rank equal to 32 bits. From the 10,518,300 choices 158,382 choices result in systems with cumulative rank 32, including the choice of Figure 9.

5.3 Covered, uncovered and partially covered words

To prove that the IVP construction is RO^2 we first demonstrate that the flow of differentials characterizing this construction is associated with values that are almost random, uniformly distributed and statistically independent, where the term almost random, uniformly distributed and statistically independent is defined in Section 5.1. For this purpose we introduce the concept of covered, uncovered and partially covered words and bytes shown in Figure 10. A word w_0 in the output is covered, $w_0 \in C(W)$, if the differential p_0 which is superimposed on its state x_0 as a result of some input perturbation is (i) non-zero, (ii) almost random, uniformly distributed, and (iii) almost statistically independent from other word differentials. A word w_0 is uncovered $w_0 \in U(W)$, if the differential p_0 superimposed on its state x_0 as a result of

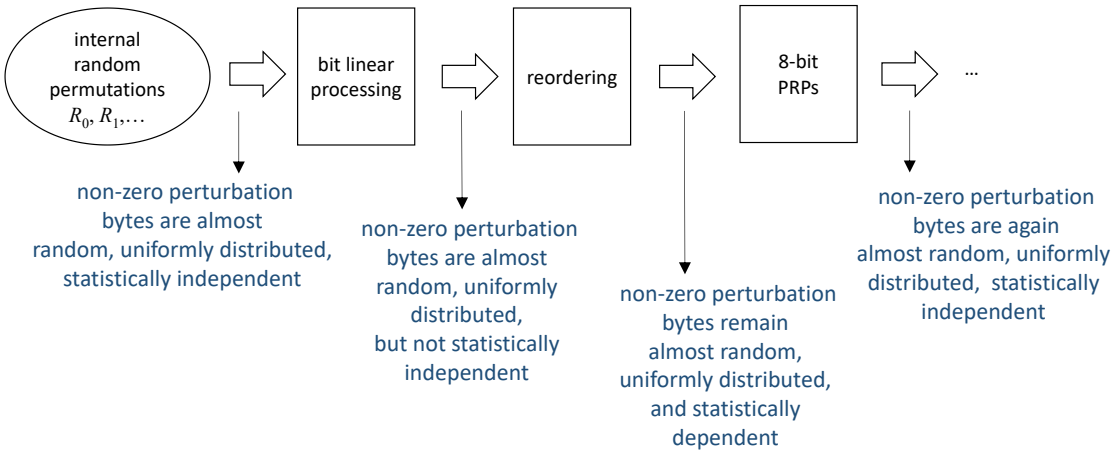


Figure 11: Statistical properties of differentials associated with internal random permutation outputs

some input perturbation is zero (e.g., the word may be exhibiting patterns).

Similarly, a byte b_0 in the output is covered, $b_0 \in C(B)$, if the differential p_0 which is superimposed on its state x_0 as a result of some input perturbation is (i) non-zero, (ii) almost random, uniformly distributed, and (iii) almsot statistically independent from other byte differentials. A byte b_0 is uncovered $b_0 \in U(B)$, if the differential p_0 superimposed on its state x_0 as a result of some input perturbation is zero. Finally, a word w_0 is partially covered $w_0 \in P(W)$ if the differential p_0 superimposed on its state x_0 causes one byte to be uncovered and the other byte to be covered.

It is not difficult to see that the covered and uncovered states are mutually exclusive for bytes and, consequently, each word can be in only one of the three states: covered, uncovered or partially covered. For the IVP construction, the mutual exclusivity of covered and uncovered states for bytes and can be established by observing that the internal random permutations are the sole source of non-zero byte differentials. Any subsequent processing involves only: (i) re-ordering operations at the byte granularity; (ii) full-rank bit-linear processing; and (iii) 8-bit PRPs. Based on this fact, byte differentials remain almost random, uniformly distributed and statistically independent as they flow through the IVP construction, as shown in Figure 11. First, at the output of the internal random permutation stages, non-zero byte differentials are almost random, uniformly distributed and statistically independent based on the indistinguishability of the internal random permutations from truncated output random oracles and the lifetime limitation $\mathcal{B} = 2^{32}$.

After the bit linear preprocessing stage, non-zero byte differentials are still almost random, uniformly distributed, but not necessarily statistically independent as each output byte is a linear combination of input bytes. After the byte remapping stage, non-zero byte differentials remain almost random, uniformly distributed, and possibly statistically dependent. It is in the last stage, and due to the fact that 8-bit PRPs are independently chosen at random, that non-zero byte differentials become again almost random, uniformly distributed and statistically independent. Furthermore, these properties of byte differentials do not change if additional diffusion stages or similar processing steps are added to the construction.

5.4 The state of column mixing transformations

The state $S(\mu_0)$ of a 4 byte column mixing transformation μ_0 can only be any of the following three of Figure 12: (i) zero state $S(\mu_0) = S_{zero}$. In this state all differential inputs $a_0, \ldots, a_3$ to the transformation μ_0 are equal to zero: $a_i = 0, \forall i \in [0, 3]$. Furthermore all output differentials are zero too; (ii) single byte stimulus state $S(\mu_0) = S_{stim}$. In this state only one of the input byte differentials is non-zero and all other byte differentials are zero: $\exists\ i \in [0, 3] : a_i \neq 0\ \wedge\ (\forall j, j \neq i, j \in [0, 3], a_j = 0)$. Since the column mixing transformation matrix is MDS, all output byte differentials are non-zero; and (iii) saturation state $S(\mu_0) = S_{sat}$ where more than one of the input byte differentials is non-zero: $\exists\, q_0, \ldots, q_{m-1} \in [0, 3] :\ 1 < m \leqslant 4\ \wedge\ (a_{q_i} \neq 0,\ \forall i \in [0, m-1])$. Since the transformation μ_0 is bit linear of full rank, the following four possible events may be true for a transformation in the saturation state: First, all output byte or byte differentials may be non-zero with probability $1 - 4 \cdot 2^{-8} + O(2^{-16}) + O(\Delta)$. Second, a single output byte or byte differential at a specific location may be zero with probability equal to $2^{-8} + O(\Delta)$, if inputs satisfy the statistical properties discussed above. Third, two output byte or byte differentials at specific locations may be zero with probability $2^{-16} + 2^{-8} \cdot O(\Delta) + O(\Delta^2)$, if, again, inputs satisfy the statistical properties discussed above. Fourth, three output byte or byte differentials at specific locations may be zero with probability $2^{-24} + 2^{-16} \cdot O(\Delta) + 2^{-8} \cdot O(\Delta^2) + O(\Delta^3)$, for the same inputs.

The effects different numbers of non-zero byte stimuli have on column mixing transformations are further illustrated in Figure 13. These observations are a direct consequence of the definition of the column mixing transformation being based on an MDS matrix. First, zero byte stimuli result in all output bytes being zero. Second, a single byte stimulus results in all output bytes being non-zero. Third, two byte stimuli result in at most one output byte being zero. Fourth, three byte stimuli result in at most two output bytes being zero. Finally, four byte stimuli result in at most three output bytes being zero.

5.5 Main result

We begin the discussion on the security of the IVP construction by establishing the fact that the observer function $\mathcal{F}_{\mathsf{EQ},512,16,4}$, which we choose to use is indeed associated with 32-bit security. This function observes whether there are 4 or more 16-bit words which are 16-bit aligned and equal to each other in a set of 512 bits.

Corollary 1: The $\mathcal{F}_{\mathsf{EQ},512,16,4}$ observer function is associated with observation probability $2^{-32.866}$.

To demonstrate that Corollary 1 holds, we need to show that the probability of finding at least 4 16-bit words that are equal to each other in a set of 32 words in random data is $2^{-32.866}$. This is an instance of the more generic problem of computing the birthday collision probability $P^{(B)}(m, n, |V|)$ for a number of people $m > 1$ having the same birthday from among the members of a set n, where the number of birthdays is $|V|$. In this case $|V| = 65536$. For this problem solutions exist [16, 18, 17, 19, 20] such as the the Suzuki et. al. bounds [19] and the approximation by Kounavis et. al. [20]. Using the Suzuki bounds, the probability of the observer function $\mathcal{F}_{\mathsf{EQ},512,16,4}$ returning true upon receiving some 512-bit input is computed and found to be $P^{(B)} < 2^{-32.8659}$. Using the Kounavis approximation, the same probability is found to be $P^{(B)} \approx 2^{-32.8662}$. Both numbers are consistent and establish the security of the $\mathcal{F}_{\mathsf{EQ},512,16,4}$ observer function.

Theorem 3: *On the security of the* IVP *construction.* The IVP construction is in the set $\mathsf{RO}^2(\mathcal{F}_{\mathsf{EQ},512,16,4}, 2^{32}, \epsilon)$, associated with the observer function $\mathcal{F}_{\mathsf{EQ},512,16,4}$, the life time 2^{32},

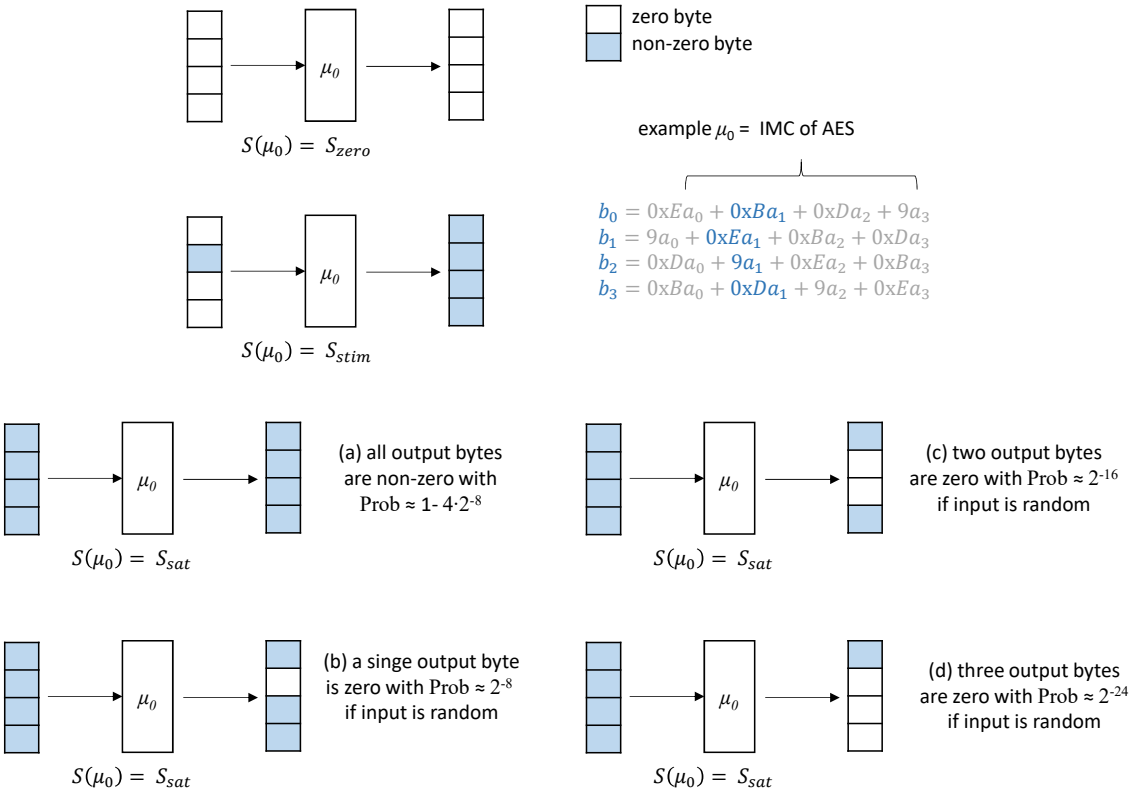


Figure 12: The states of a column mixing transformation

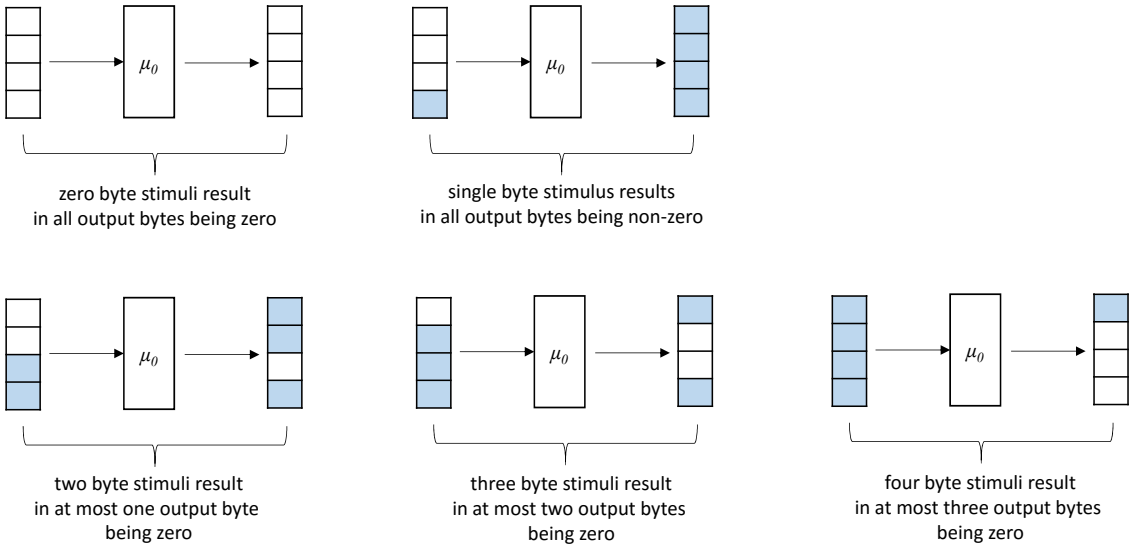


Figure 13: Behavior of a column mixing transformation as the number non-zero byte stimuli changes

and the indistinguishability parameter ϵ, which is equal to 0.697 bits:

$$\mathsf{IVP} \in \mathsf{RO}^2(\mathcal{F}_{\mathsf{EQ},512,16,4}, 2^{32}, 0.697) \tag{5.1}$$

Proof: Let's consider y to be the input to the IVP construction. We need to show that $\text{Prob}[\,\mathsf{IVP}(y) \in \Pi(\mathcal{F}_{\mathsf{EQ},512,16,4})\,] \leqslant P_{\mathcal{F}_{\mathsf{EQ},512,16,4}} \cdot 2^{\epsilon}$ where $\epsilon = 0.697$. Moreover, we need to show that the probability bound $P_{\mathcal{F}_{\mathsf{EQ},512,16,4}} \cdot 2^{\epsilon}$ remains the same even when the event $\mathsf{IVP}(y) \in \Pi(\mathcal{F}_{\mathsf{EQ},512,16,4})$ is conditioned upon inputs other than y and their responses. We consider y to be the sum of a state vector z and a perturbation vector p: $y = z + p$. Due to the fact that the input is not repeating, every perturbation vector is unique. This is true, even if a perturbation vector results from the concatenation of previously applied block perturbations. It is easy to see that the statistical properties of differentials discussed above hold for every such non-repeating perturbation vector.

In one case, the patterns of the $\mathcal{F}_{\mathsf{EQ},512,16,4}$ observer function are present in the output of the state vector z and remain uncovered. In this case, the probability $\text{Prob}[\mathsf{IVP}(y) \in \Pi(\mathcal{F}_{\mathsf{EQ},512,16,4})]$ is bounded by the probability that 4 words or more in the IVP construction output remain uncovered. In another case, patterns of the $\mathcal{F}_{\mathsf{EQ},512,16,4}$ appear when all words of the output of the IVP construction are covered. In this case, the probability $\text{Prob}[\mathsf{IVP}(y) \in \Pi(\mathcal{F}_{\mathsf{EQ},512,16,4})]$ is equal to $P_{\mathcal{F}_{\mathsf{EQ},512,16,4}}$ plus a negligible term associated with the advantage of distinguishing the internal random permutations of the IVP construction from truncated output random oracles. In between these two cases, a number of other subevents are possible, where patterns associated with the $\mathcal{F}_{\mathsf{EQ},512,16,4}$ observer function are formed from both uncovered words in the IVP construction output, the values of which depend on the state vector z, and covered words, the values of which depend on both the state z and the perturbation vector p.

Before analyzing each of these subevents, we compute the probability of having i-th word in the output of the IVP construction uncovered. We refer to such probability as $P_{UW}(i) = \text{Prob}[w_i \in U(W)]$. In what follows we demonstrate that $P_{UW}(i) = P_{UW} \leqslant 2^{-16} + O(\Delta')$ and is bounded independently of the word index i. The term $O(\Delta')$ can be considered negligible. To prove this bound for $P_{UW}(i)$ we state and prove a number of useful Lemmas.

Lemma 1: *On computing the probability of a single uncovered word if only one block perturbation value is non-zero.* If a block perturbation value p_0 (i.e., a perturbation on one of the four 128-bit blocks of the IVP construction input) is non-zero and all other block perturbations p_1, p_2, and p_3 are equal to zero then: (i) all four first stage *CM* transformations $\nu_0, \ldots, \nu_3$ of the IVP construction are in the single byte stimulus state; (ii) all four second stage *CM* transformations $\mu_0, \ldots, \mu_3$ of the IVP construction are in the saturation state; and (iii) the probability of a single uncovered word is equal to $2^{-16} + O(\Delta')$, where $\Delta' \leqslant 2^{-40}$:

$$\begin{gathered} \text{if } p_0 \neq 0 \text{ and } p_1 = p_2 = p_3 = 0, \text{ then } S(\nu_i) = S_{stim}\ \forall i \in [0,3], \\ S(\mu_i) = S_{sat}\ \forall i \in [0,3],\ P_{UW} = 2^{-16} + O(\Delta') \end{gathered} \tag{5.2}$$

Proof: The situation where only one block perturbation is non-zero is shown in Figure 14. If only a single block perturbation is non-zero, then each column mixing transformation from among $\nu_0, \ldots, \nu_3$ has exactly one non-zero input byte differential. Hence, each transformation from $\nu_0, \ldots, \nu_3$ is in the single byte stimulus state. All four byte differentials of each column, after each transformation completes, are in this case non-zero. Byte differentials remain non-zero after the subsequent row shifting and *Sbox* stages complete too. Hence, all four second stage column transformations $\mu_0, \ldots, \mu_3$ of the IVP construction are in the saturation state. A single unmodified (i.e., uncovered) word in the output of the construction results from two unaligned unmodified bytes in the preceding row shifting transformation. Since each of the

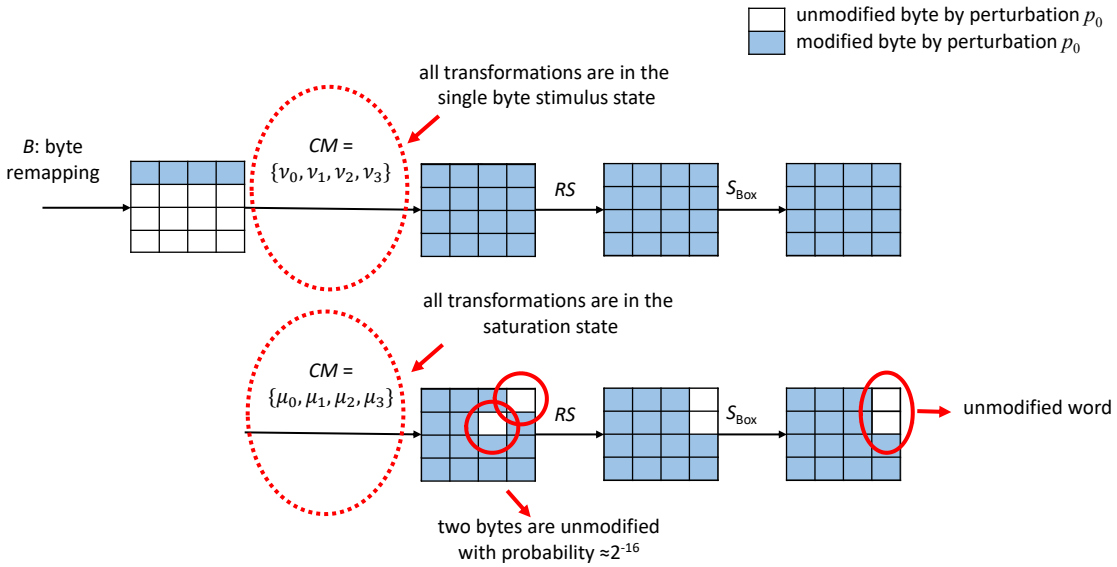


Figure 14: Corruptions in the IVP state when only one block perturbation is non-zero

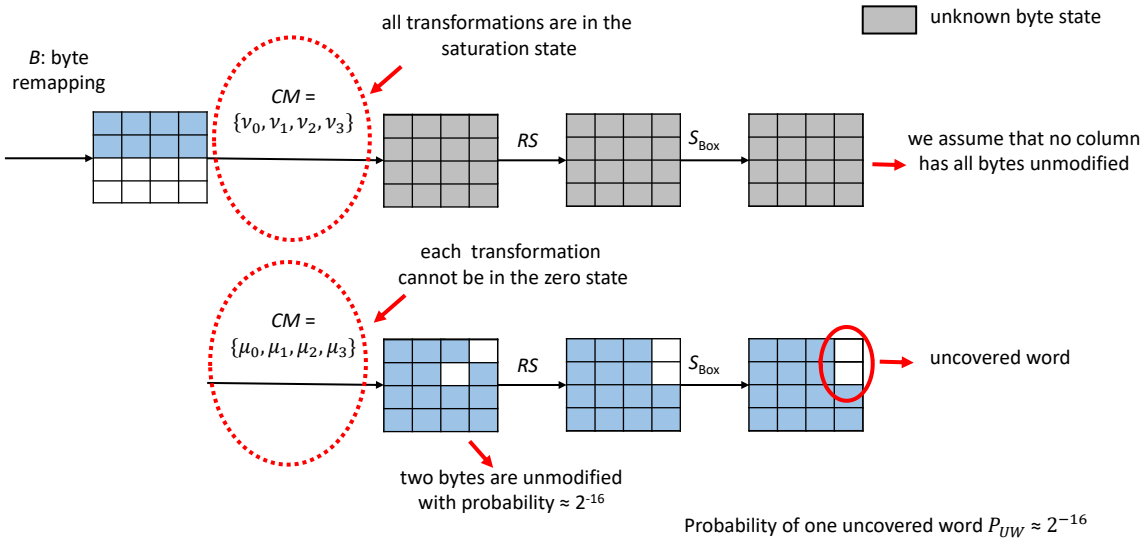


Figure 15: Corruptions in the IVP state when two block perturbations are non-zero (case I)

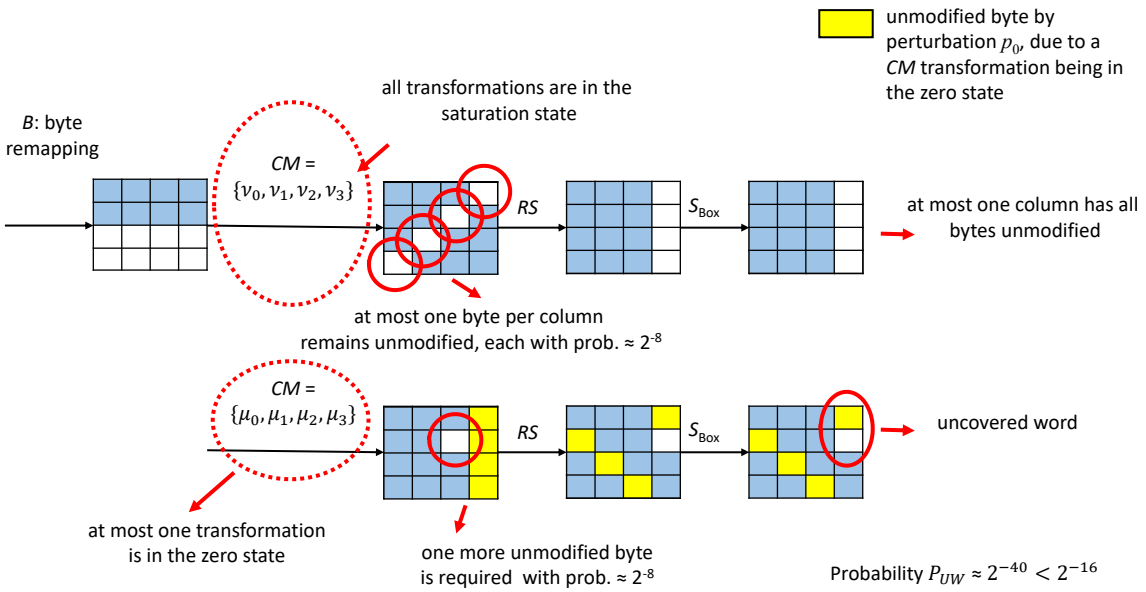


Figure 16: Corruptions in the IVP state when two block perturbations are non-zero (case II)

transformations $\mu_0, \ldots, \mu_3$ is in the saturation state each zero byte differential (i.e., unmodified byte) appears in the output of these transformations with probability $2^{-8} + O(\Delta)$. As a result the probability of a single uncovered word in the output of the IVP construction is equal to $2^{-16} + 2^{-8} \cdot O(\Delta) + O(\Delta^2)$.

We complete the proof setting $O(\Delta') \leftarrow 2^{-8} \cdot O(\Delta) + O(\Delta^2)$. This term compensates for the fact that the inputs are not exactly uniformly distributed, as they come from random permutations and not truncated output random oracles. Hence, Lemma 1 is proven. We note that the term $O(\Delta')$ is equal to $O(2^{-64})$, thus significantly smaller than the bound 2^{-40}, which appears in the proposition of Lemma 1 above. The reason why we state Lemma 1 this way is because we want to have the same negligible term bound appearing across all lemmas 1-4. We also note that the proof is the same if the index of the non-zero perturbation block is other than 0, and the row shifting transformation of the IVP construction shifts the rows of the state matrix in the opposite direction. Similar observations apply to the subsequent lemmas 2-4.

Lemma 2: *On computing the probability of a single uncovered word if exactly two block perturbation values are non-zero.* If two block perturbation values p_0 and p_1 are non-zero and the other perturbation values p_2 and p_3 are equal to zero then: (i) all four first stage *CM* transformations $\nu_0, \ldots, \nu_3$ of the IVP construction are in the saturation state; (ii) there can be at most a single second stage *CM* transformation in the zero state; and (iii) the probability of a single uncovered word is $\leqslant 2^{-16} + O(\Delta')$ for $\Delta' \leqslant 2^{-40}$.

$$\begin{gathered} \text{if } p_0 \neq 0, p_1 \neq 0 \text{ and } p_2 = p_3 = 0 \text{ then } S(\nu_i) = S_{sat}\ \forall i \in [0,3], \\ \text{there is at most one transformation } \mu_q:\ S(\mu_q) = S_{zero},\ q \in [0,3], \\ P_{UW} \leqslant 2^{-16} + O(\Delta') \end{gathered} \tag{5.3}$$

Proof: We consider two cases: In a first case (case I) we assume that no column has all bytes unmodified at the input to the second stage transformations $\mu_0, \ldots, \mu_3$. At another case (case II) we consider that this assumption does not hold. The situation of case I is shown in

Figure 15. In case I, if exactly two block perturbations are non-zero then each column mixing transformation from $\nu_0, \ldots, \nu_3$ has exactly two non-zero input byte differentials. Hence, each transformation from $\nu_0, \ldots, \nu_3$ is in the saturation state. Furthermore, in each column output of $\nu_0, \ldots, \nu_3$ there can be at most one zero byte differential. Due to the assumption associated with case I, each transformation from $\mu_0, \ldots, \mu_3$ is either in the single byte stimulus or in the saturation state. Furthermore, due to the fact that there is at most one byte per column in the output of $\nu_0, \ldots, \nu_3$ which is unmodified, the number of transformations from $\mu_0, \ldots, \mu_3$ which are in the single byte stimulus state cannot be more than 1. Indeed, if there were 2 or more transformations from $\mu_0, \ldots, \mu_3$ in the single byte stimulus state, then the input to $\mu_0, \ldots, \mu_3$ would have contained at least 6 unmodified bytes. Therefore, either 3 or 4 transformations from $\mu_0, \ldots, \mu_3$ are in the saturation state. From these transformations two unaligned zero byte differentials result in a single uncovered word with probability bounded by $2^{-16} + 2^{-8} \cdot O(\Delta) + O(\Delta^2)$. Next, we set the correcting term $2^{-8} \cdot O(\Delta) + O(\Delta^2)$ to $O(\Delta')$, as in Lemma 1. In this way we complete the proof of Lemma 2 for case I.

If case II holds, then four unmodified bytes at the output of $\nu_0, \ldots, \nu_3$ become aligned via the subsequent row shifting transformation, in order to form a zero differential input column to one of the transformations $\mu_0, \ldots, \mu_3$. This is shown in Figure 16. Since there can be at most four zero byte differentials at the output of $\nu_0, \ldots, \nu_3$ there can be no more than one unmodified input column to $\mu_0, \ldots, \mu_3$. Hence, exactly one transformation from $\mu_0, \ldots, \mu_3$ is in the zero state and three transformations are in the saturation state. Transformations $\nu_0, \ldots, \nu_3$ are all in the saturation state, as in case I. Because of this reason, each of the four zero byte differentials at the output of $\nu_0, \ldots, \nu_3$ appears with probability $2^{-8} + O(\Delta)$. Having one word uncovered at the output of the IVP construction requires at least one more byte differential to be zero at the output of $\mu_0, \ldots, \mu_3$. As a result, the probability of having one uncovered word in the output of the IVP construction is bounded by $2^{-40} + 2^{-32} \cdot O(\Delta) + \ldots + O(\Delta^5)$. This bound is equal to $O(\Delta')$. Lemma 2 follows directly by adding the bounds for case I and all instances of case II. We note that the proof is the same if the unmodified input column to $\mu_0, \ldots, \mu_3$ is other than the one shown in Figure 16.

Lemma 3: *On computing the probability of a single uncovered word if exactly three block perturbation values are non-zero.* If three block perturbation values p_0, p_1 and p_2 are non-zero and perturbation value p_3 is equal to zero then: (i) all four first stage *CM* transformations $\nu_0, \ldots, \nu_3$ of the IVP construction are in saturation state; (ii) there can be at most two second stage *CM* transformations in the zero state; and (iii) the probability of a single uncovered word is $\leqslant 2^{-16} + O(\Delta')$ for $\Delta' \leqslant 2^{-40}$.

$$\begin{gathered} \text{if } p_0 \neq 0, p_1 \neq 0, p_2 \neq 0 \text{ and } p_3 = 0 \text{ then } S(\nu_i) = S_{sat}\ \forall i \in [0,3], \\ \text{there are at most two transformations } \mu_q, \mu_r :\ S(\mu_q) = S(\mu_r) = S_{zero},\ q, r \in [0,3], \\ P_{UW} \leqslant 2^{-16} + O(\Delta') \end{gathered} \tag{5.4}$$

Proof: We consider three cases: In a first case (case I) we assume that no column has all bytes unmodified at the input to the second stage transformations $\mu_0, \ldots, \mu_3$. At another case (case II) we consider that the number columns that have all bytes unmodified at the input to the second stage transformations $\mu_0, \ldots, \mu_3$ is two or more. A third case is when the number columns that have all bytes unmodified at the input to the second stage transformations $\mu_0, \ldots, \mu_3$ is exactly one. This third case is similar to case II of Lemma 2, associated with the same bound $O(\Delta')$, and its proof is omitted.

The situation of case I is shown in Figure 17. In case I, if exactly three block perturbations

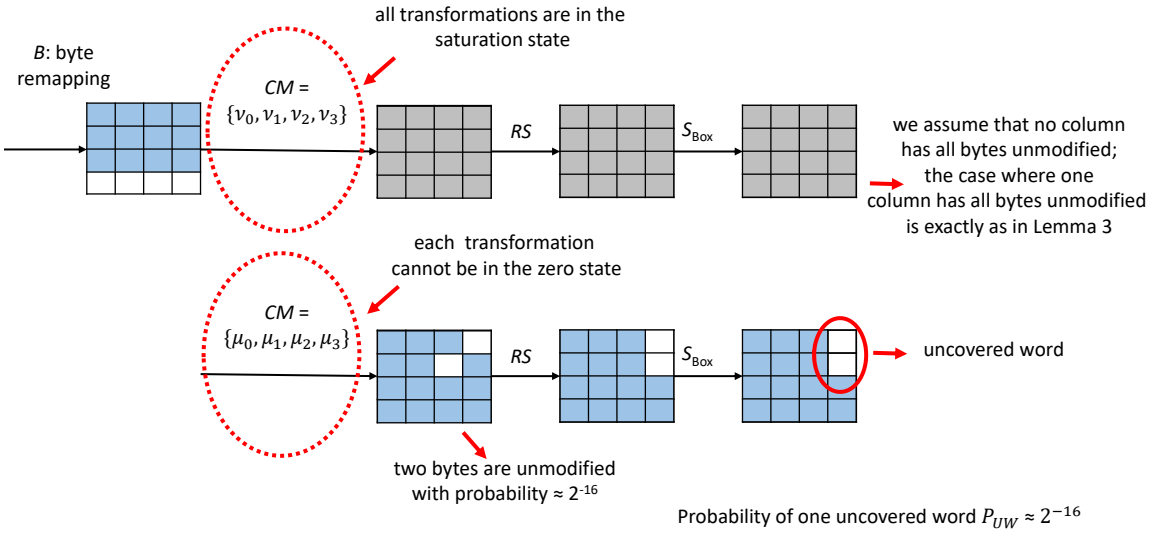


Figure 17: Corruptions in the IVP state when three block perturbations are non-zero (case I)

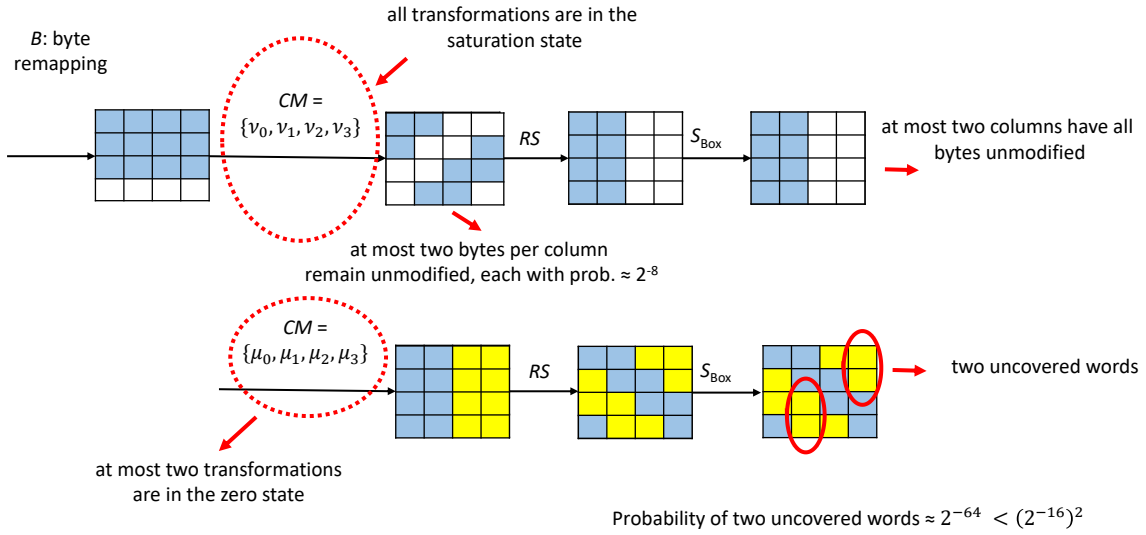


Figure 18: Corruptions in the IVP state when three block perturbations are non-zero (case II)

are non-zero then each column mixing transformation from $\nu_0, \ldots, \nu_3$ has exactly three non-zero input byte differentials. Hence, each transformation from $\nu_0, \ldots, \nu_3$ is in the saturation state. Furthermore, in each column output of $\nu_0, \ldots, \nu_3$ there can be at most two zero byte differentials. Due to the assumption associated with case I, each transformation from $\mu_0, \ldots, \mu_3$ is either in the single byte stimulus or in the saturation state. Furthermore, due to the fact that there are at most two byte differentials per column in the output of $\nu_0, \ldots, \nu_3$ which are zero, the number of transformations from $\mu_0, \ldots, \mu_3$ which are in the single byte stimulus state cannot be more than 2. If there were 3 or more transformations from $\mu_0, \ldots, \mu_3$ in the single byte stimulus state, then the input to $\mu_0, \ldots, \mu_3$ would have contained at least 9 unmodified bytes. Therefore 2, 3 or 4 transformations from $\mu_0, \ldots, \mu_3$ are in the saturation state. From these transformations, two unaligned byte differentials being equal to zero result in a single uncovered word with probability bounded by $2^{-16} + O(\Delta')$, where $O(\Delta')$ is as in Lemma 1. Hence Lemma 3 is proven for case I.

If case II holds, then eight unmodified bytes at the output of $\nu_0, \ldots, \nu_3$ become aligned via the subsequent row shifting transformation, in order to form two zero differential input columns to two of the transformations $\mu_0, \ldots, \mu_3$. This is shown in Figure 18. Since there can be at most eight zero byte differentials at the output of $\nu_0, \ldots, \nu_3$ there can be no more than two zero input columns to $\mu_0, \ldots, \mu_3$. Hence, exactly two transformations from $\mu_0, \ldots, \mu_3$ are in the zero state and two transformations are in the saturation state. Once again, transformations $\nu_0, \ldots, \nu_3$ are all in the saturation state, as in case I. Because of this reason, each of the eight zero byte differentials at the output of $\nu_0, \ldots, \nu_3$ appears with probability $2^{-8} + O(\Delta)$. In this case, at most two words are uncovered at the output of the IVP construction, as shown in the figure. These uncovered words result from having such unmodified byte differentials at the output of $\nu_0, \ldots, \nu_3$. As a result, the probability of having an uncovered word in the output of the IVP construction is bounded by $2^{-64} + 2^{-56} \cdot O(\Delta) + \ldots + O(\Delta^8) < O(\Delta')$. Lemma 3 is proven by adding the bounds for case I, and all instances of cases II and III. We note that the proof and bound, when case II holds, is the same for all possible pairs of column indexes, associated with the unmodified columns passed to $\mu_0, \ldots, \mu_3$.

Lemma 4: *On computing the probability of a single uncovered word if all four block perturbation values are non-zero.* If all four block perturbation values p_0, p_1, p_2 and p_3 are non-zero then: (i) all four first stage CM transformations $\nu_0, \ldots, \nu_3$ of the IVP construction are in saturation state; (ii) there can be at most three second stage CM transformations in the zero state; and (iii) the probability of a single uncovered word is $\leqslant 2^{-16} + O(\Delta')$ for $\Delta' \leqslant 2^{-40}$.

$$\begin{gathered} \text{if } p_i \neq 0,\ \forall i \in [0,3] \text{ then } S(\nu_i) = S_{sat}\ \forall i \in [0,3], \\ \text{there are at most three transformations } \mu_q, \mu_r, \mu_s : \\ S(\mu_q) = S(\mu_r) = S(\mu_s) = S_{zero},\ q,r,s \in [0,3],\ P_{UW} \leqslant 2^{-16} + O(\Delta') \end{gathered} \tag{5.5}$$

Proof: We consider three cases: In a first case (case I) we assume that no column has all bytes unmodified at the input to the second stage transformations $\mu_0, \ldots, \mu_3$. At another case (case II) we consider that the number columns that have all bytes unmodified at the input to the second stage transformations $\mu_0, \ldots, \mu_3$ is three or more. A third case is when the number columns that have all bytes unmodified at the input to the second stage transformations $\mu_0, \ldots, \mu_3$ is either one or two. This third case is similar to case II of Lemmas 2 and 7. For this case, the bound is $O(\Delta')$ and the proof is omitted.

The situation of case I is shown in Figure 19. In case I, if all four block perturbations are non-zero then each column mixing transformation from $\nu_0, \ldots, \nu_3$ has four non-zero input byte differentials. Hence, each transformation from $\nu_0, \ldots, \nu_3$ is in the saturation state in this

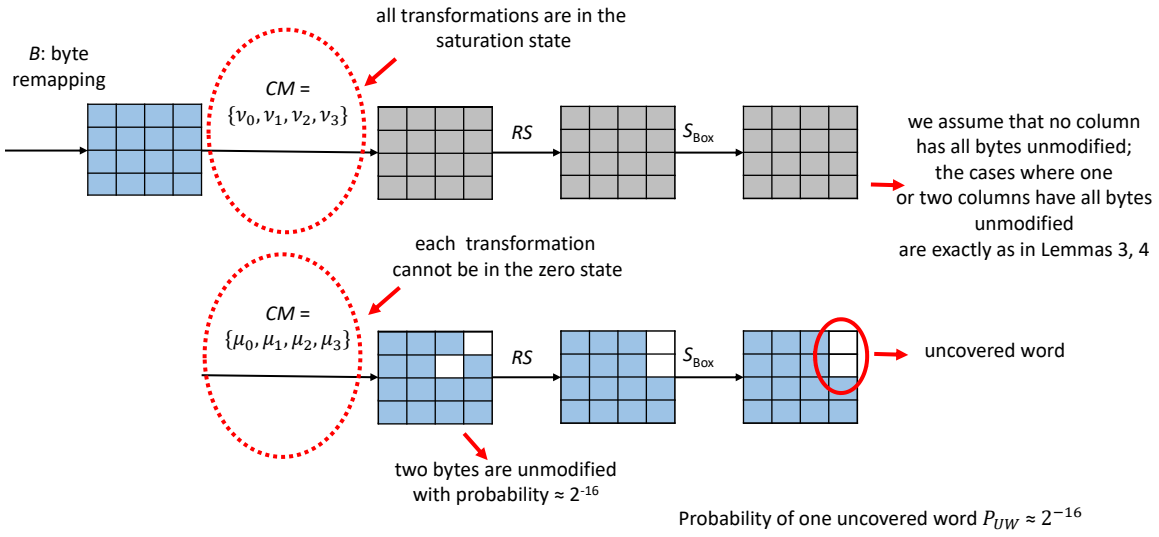


Figure 19: Corruptions in the IVP state when all block perturbations are non-zero (case I)

case too. Furthermore, in each column output of $\nu_0, \ldots, \nu_3$ there can be at most three zero byte differentials. Due to the assumption associated with case I, each transformation from $\mu_0, \ldots, \mu_3$ is either in the single byte stimulus or in the saturation state. Furthermore, due to the fact that there are up to three byte differentials per column in the output of $\nu_0, \ldots, \nu_3$ which are zero, the number of transformations from $\mu_0, \ldots, \mu_3$ which are in the single byte stimulus state can be any from 0 to 4. If two such transformations in the saturation state, then two unaligned byte differentials are equal to zero and result in a single uncovered word with probability bounded by $2^{-16} + O(\Delta')$, where $O(\Delta')$ is as in Lemma 1. Hence, Lemma 4 is proven for case I.

If case II holds, then twelve unmodified bytes at the output of $\nu_0, \ldots, \nu_3$ become aligned via the subsequent row shifting transformation, in order to form three zero differential input columns to three of the transformations $\mu_0, \ldots, \mu_3$. This is shown in Figure 20. Since there can be at most twelve zero byte differentials at the output of $\nu_0, \ldots, \nu_3$ there can be no more than three zero differential input columns to $\mu_0, \ldots, \mu_3$. Hence, exactly three transformations from $\mu_0, \ldots, \mu_3$ are in the zero state and one transformation is in the saturation state. Transformations $\nu_0, \ldots, \nu_3$ are all in the saturation state, as in case I. Because of this reason, each of the twelve zero byte differentials at the output of $\nu_0, \ldots, \nu_3$ appears with probability $2^{-8} + O(\Delta)$. This case always results in four words which are uncovered at the output of the IVP construction. These uncovered words result from having unmodified bytes at the output of $\nu_0, \ldots, \nu_3$ as shown in the figure. As a result, the probability of having an uncovered word in the output of the IVP construction is bounded by $2^{-96} + 2^{-88} \cdot O(\Delta) + \ldots + O(\Delta^{12}) < O(\Delta')$. Lemma 4 is proven by adding the bounds for case I, and all instances of cases II and III.

So far, Lemmas 1-4 have established the fact that the probability of having a single uncovered word appearing in the output of the IVP construction is bounded in a manner that is independent of the word index. Furthermore, the bound is equal to $2^{-16} + O(\Delta')$, where $O(\Delta')$ is a negligible term associated with the advantage of distinguishing the internal random permutations of the IVP construction from truncated output random oracles. A next corollary

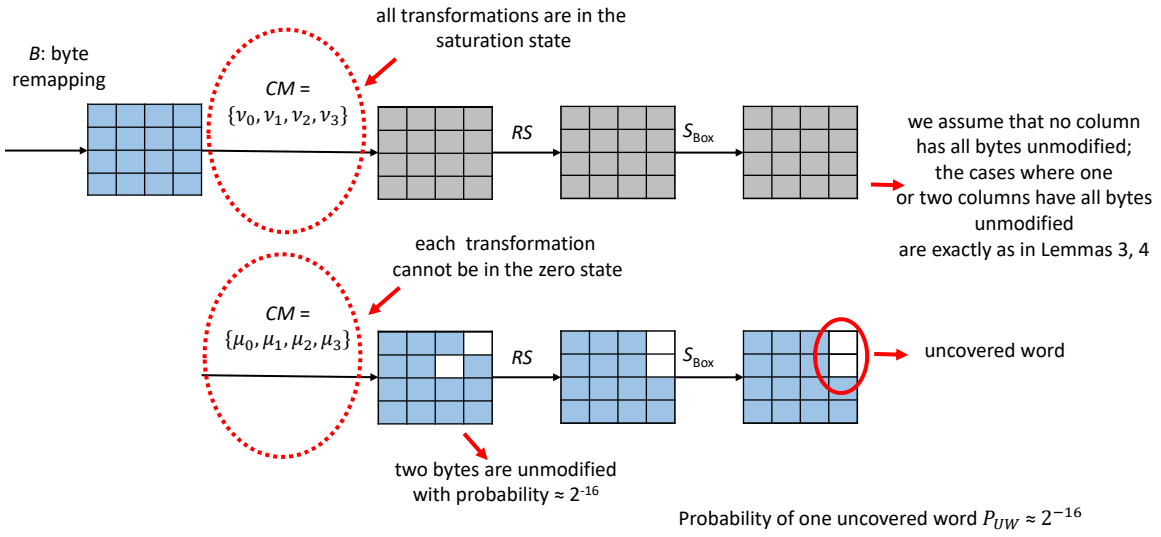


Figure 20: Corruptions in the IVP state when all block perturbations are non-zero (case II)

suggests a bound for the probability of having a single byte uncovered in the output of the IVP construction.

Corollary 2: A single uncovered byte appears at the output of the IVP construction with probability P_{UB} which is bounded as follows:

$$P_{UB} \leqslant 2^{-8} + O(\Delta) \tag{5.6}$$

where $\Delta < 2^{-64}$ is associated with the advantage of distinguishing the internal random permutations of the IVP construction from truncated output random oracles.

It is not difficult to see why Corollary 2 holds. In all situations covered by Lemma 1 and cases I and II of Lemmas 2, 3 and 4, either all of the column mixing transformations $\nu_0, \ldots, \nu_3$, or all of the column mixing transformations $\mu_0, \ldots, \mu_3$ are in the saturation state. Because of this fact, it is not possible for zero byte differentials to originate in any way other than from the bit linear mixing performed by these transformations. Hence, a byte is uncovered with probability $\leqslant 2^{-8} + O(\Delta)$.

Completing the proof of Theorem 3:

Having established bounds for the probability of seeing an uncovered word and an uncovered byte at the output of the IVP construction, we proceed with proving Theorem 3. The probability of seeing the patterns associated with the $\mathcal{F}_{\mathsf{EQ},512,16,4}$ observer function at the output of IVP can be expressed as a sum of probabilities associated with different subevents. In one subevent, as discussed earlier, the patterns are present in the output produced from the state vector z (i.e., the original plaintext) and remain visible at the IVP output after the addition of a perturbation vector p onto z. In another subevent, patterns are formed only from word differentials, and all IVP output bytes and words are covered. In other subevents patterns are formed from both covered and uncovered words or bytes. All these subevents are captured in Proposition 2 below:

Proposition 2: The probability of the event in which the output of the IVP construction exhibits the patterns associated with the $\mathcal{F}_{\mathsf{EQ},512,16,4}$ observer function is bounded as follows:

$$\text{Prob}[\mathsf{IVP}(y) \in \Pi(\mathcal{F}_{\mathsf{EQ},512,16,4})] \leqslant P_{\mathcal{F}_{\mathsf{EQ},512,16,4}} + \binom{32}{4} \cdot \mathcal{P}_{UW}^4 + T + O(\Delta'') \tag{5.7}$$

where $P_{\mathcal{F}_{\mathsf{EQ},512,16,4}}$ is the observation probability associated with $\mathcal{F}_{\mathsf{EQ},512,16,4}$, $\mathcal{P}_{UW} = 2^{-16}$ is the dominant term of the probability P_{UW} of a word in the output of IVP being uncovered, T is an additive term equal to $2^{-33.553}$ and $\Delta'' \leqslant 2^{-46.544}$.

The term $P_{\mathcal{F}_{\mathsf{EQ},512,16,4}}$ corresponds to the subevent, where the pattern is formed only from word differentials and all words in the IVP output are covered. The term $\binom{32}{4} \cdot \mathcal{P}_{UW}^4$ corresponds to the subevent where the pattern is present in the output of the state vector z and remains visible after the addition of the perturbation vector p. In this case, the pattern is formed from four words, all of which are uncovered. The additive term T corresponds to all other subevents where the pattern is formed from both covered and uncovered words or bytes. The proof of the correctness of Proposition 2 is provided in Appendix C.

Substituting $P_{\mathcal{F}_{\mathsf{EQ},512,16,4}}$ with $2^{-32.866}$ from Corollary 1, $\mathcal{P}_{UW}$ with the bound 2^{-16} from Lemmas 1-4, and T with $2^{-33.553}$ we obtain:

$$\begin{gathered}\text{Prob}[\mathsf{IVP}(y) \in \Pi(\mathcal{F}_{\mathsf{EQ},512,16,4})] \leqslant 2^{-32.169} + O(\Delta'') = \\ 2^{-32.866} \cdot 2^{0.697} + O(\Delta'') = P_{\mathcal{F}_{\mathsf{EQ},512,16,4}} \cdot 2^{\epsilon}\end{gathered} \tag{5.8}$$

where $\epsilon \leftarrow 0.697 + \log_2(\frac{2^{-32.169} + O(\Delta'')}{2^{-32.169}}) \approx 0.697$. To complete the proof we observe that the probability bounds of all subevents considered in the derivation of 5.8 are independent of input-output pairs, where inputs are other than y. Indeed all subevents considered involve bytes or words which are either covered or demonstrate patterns coming from the unperturbed state z. The probability bounds associated with covered words or bytes are independent of the values of words or bytes of different inputs or outputs. On the other hand, the bounds associated with subevents where patterns may exist in the unperturbed state z are derived in a way that is independent of the values of these patterns. Therefore the bound in inequality 5.8 holds even if the event $\mathsf{IVP}(y) \in \Pi(\mathcal{F}_{\mathsf{EQ},512,16,4})$ is conditioned upon input-output pairs, where inputs are other than y. This completes the proof of Theorem 3 and establishes the security of the IVP construction in the input perturbing and oracle replacing adversary models.

6 Discussion

There are several questions about implicit integrity that are open and possibly the subject of future work. We believe that constructions which generalize IVP and apply to larger inputs may be able to support higher security levels. Such constructions would potentially relax the assumption of adversaries performing on-line attacks which characterizes the IVP example discussed in this paper. Specifically, the width values used in the IVP example can be replaced by generic width parameters and the pattern of seeing for our more words equal to each other in a set of 32 can be replaced by a more generic requirement that larger quantities (e.g, 128 bits, 256 bits) should demonstrate entropy below a threshold. Such generalization is the subject of future work.

One may also ask why not simply compress the data and augment it by a MAC in the now free space. We believe there is a practical reason why implicit integrity is better than compression. Compressing/decompressing in combinatorial logic requires not only detecting

patterns, but also encoding the data in such a way so that some necessary space is freed for holding a MAC. For some patterns such as nibble-based patterns, this process can be quite costly, especially if implemented in combinatorial logic. Ongoing research of ours shows that the client cache lines that can be compressed at reasonable cost are significantly fewer than those protected via implicit integrity (78% as opposed to 91%). In contrast, the IVP construction requires only the detection of patterns, avoiding compressing or decompressing the data. Furthermore, it burdens the encrypt/decrypt data path with only two additional rounds of permutation-substitution steps. Detailed comparison between compression-based and implicit integrity is the subject of future work.

References

[1] D. Durham and M. Long, *Memory Integrity*, United States Patent, No. 9,213,653, Decednber 2013.

[2] D. Durham, S. Chhabra, S. Deutsch, M. Long and A. Trivedi, *Memory Integrity with Error Detection and Correction*, United States Patent, No.9,990,249, December 2015.

[3] D. Durham, S. Chhabra, M. Kounavis, S. Deutsch, K. Grewal, J. Cihula and S. Komijani, *Convolutional Memory Integrity*, United States Patent Application, No. 20170285976.

[4] *Secure Hash Standard*, Federal Information Processing Standards Publication FIPS PUB 180-4.

[5] *SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions*, Federal Information Processing Standards Publication FIPS PUB 202.

[6] *The Keyed-Hash Message Authentication Code (HMAC)*, Federal Information Processing Standards Publication FIPS PUB 198-1.

[7] *SHA-3 Derived Functions: cSHAKE, KMAC, TupleHash and ParallelHash*, NIST Special Publication 800-185.

[8] *Advanced Encryption Standard (AES)*, Federal Information Processing Standards Publication FIPS PUB 197.

[9] *Recommendation for Block Cipher Modes of Operation: The XTS-AES Mode for Confidentiality on Storage Devices*, NIST Special Publication 800-38E.

[10] F. McKeen, I. Alexandrovich, A. Berenzon, C. Rozas, H. Shafi, V. Shanbhogue and U. Savagaonkar, *Innovative instructions and software model for isolated execution*, Proceedings of the Workshop on Hardware and Architectural Support for Security and Privacy (HASP), 2013.

[11] A. J. Menezes and P. C. van Oorschot and S. A. Vanstone, *Handbook of Applied Cryptography*, CRC Press, 1996.

[12] M. Bellare and P. Rogaway, *Random Oracles are Practical: A Paradigm for Designing Efficient Protocols*, In Proceedings, ACM Conference on Computer and Communications Security, pp. 62-73, 1993.

[13] M. Luby and C. Rackoff, *How to Construct Pseudorandom Permutations and Pseudorandom Functions*, SIAM Journal of Computing, Vol. 17, No, 2, 1988.

[14] C. Hall, D. A. Wagner, J. Kelsey and B. Schneier, *Building PRFs from PRPs*, CRYPTO 1998: 370-389.

[15] S. Gilboa and S. Gueron, *Distinguishing a truncated random permutation from a random function*, IACR Cryptology ePrint Archive 2015: 773 (2015).

[16] M. S. Klamkin and D. J. Newman, *Extensions on the Birthday Surprise*, Journal of Combinatorial Theory, Vol. 3, pp. 279-282, 1967.

[17] A. DasGupta, *The matching, birthday and the strong birthday problem: a contemporary review*, Journal of Statistical Planning and Inference, Vol. 130, pp. 377-389, 2004.

[18] *Wolfram Mathworld: Birthday Problem*, website, available on-line at: `http://mathworld.wolfram.com/BirthdayProblem.html`

[19] K. Suzuki, D. Tonien, K. Kurosawa and K. Toyota, *Birthday Paradox for Multi-collisions*, International Conference on Information Security and Cryptology, pp. 29-40, 2006.

[20] M. Kounavis, S. Deutsch, D. Durham and S. Komijani, *Non-recursive computation of the probability of more than two people having the same birthday*, ISCC 2017: 1263-1270.

[21] B. Sun, M, Liu, J. Guo, L. Qu and V. Rijmen, *New insights on AES-Like SPN Ciphers*, CRYPTO 2016.

[22] A. Bogdano and V. Rijmen, *Linear hulls with correlation zero and linear cryptanalysis of block ciphers*, Design, Codes and Cryptography, 70(3), pp. 369-383, 2014.

[23] R. Canetti, O. Goldreich and S. Halevi, *The random oracle metholodology revisited*, 30th ACM Symposium of the Theory of Computing (STOC), pp. 209-218, 1998.

[24] S. Halevi and P. Rogaway, *A Parallelizable Enciphering Mode*, In: Okamoto T. (eds) Topics in Cryptology – CT-RSA 2004. CT-RSA 2004. Lecture Notes in Computer Science, vol 2964. Springer, Berlin, Heidelberg

[25] S. Halevi and P. Rogaway, *A Tweakable Enciphering Mode*, CRYPTO 2003.

[26] V. T. Hoang, T. Krovetz and P. Rogaway, *Robust Authenticated Encryption: AEZ and the Problem that it Solves*, EUROCRYPT 2015.

[27] C. Badertscher, C. Matt, U. Maurer, P. Rogaway and B. Tackmann, *Robust Authenticated Encryption and the Limits of Symmetric Cryptography*, 15th IMA International Conference on Cryptography and Coding, 2015.

[28] U. Maurer, R. Renner and C. Holenstein, *Indifferentability, Impossibility Results on Reductions, and Applications to the Random Oracle Methodology*, In Moni Naor, editor, First Theory of Cryptography Conference - TCC 2004, volume 2951 of LNCS, pages 21-39. Springer-Verlag, February 19–21 2004.

[29] Y. Dodis, T. Liu, M. Stam, J. Steinberger, *Indifferentiability of Confusion-Diffusion Networks*, hskip 1em plus 0.5em minus 0.4emePrint 2015/680.

[30] I. Dinur, O. Dunkelman, N. Keller, A. Shamir, *Memory-Efficient Algorithms for Finding Needles in Haystacks*, CRYPTO 2016.

[31] M. Dworkin, *Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC*, NIST Special Publication 800-38D.

[32] J. Salowey, A. Choudhury and D. McGrew, *AES Galois Counter Mode (GCM) Cipher Suites for TLS*, RFC 5288.

[33] N. Ferguson, S. Lucks, B. Schneier, D. Whiting, M. Bellare, T. Kohno, J. Calas and J. Walker, *The Skein Hash Function Family*, available online at `http://www.skein-hash.info/sites/default/files/skein1.1.pdf`

[34] *Randomness Requirements for Security*, RFC 4086, available online at `http://tools.ietf.org/html/rfc4086`. June 2006.

Appendices

A Proof of Theorem 1

We need to show that, for every input perturbing adversary $\mathcal{A}^{\mathsf{D}()}_{\mathsf{INP}} \leftarrow \mathcal{A}$, it holds that:

$$\text{Prob}[\, y \leftarrow \mathcal{A}(q_0, \ldots, q_{m-1}, Q_{\mathcal{B}'});\ y \notin \{q_0, \ldots, q_{m-1}\} \cup Q_{\mathcal{B}'};\ \mathsf{D}(y) \in \Pi(\mathcal{F}_{\mathsf{obv},L})\,] \leqslant P_{\mathcal{F}_{\mathsf{obv},L}} \cdot 2^{\epsilon} \tag{A.1}$$

We assume that an adversary $\mathcal{A}$ exists for which the relation A.1 does not hold. This adversary can succeed in repeatedly producing messages the outputs of which exhibit patterns with probability $> P_{\mathcal{F}_{\mathsf{obv},L}} \cdot 2^{\epsilon}$. We will show that if such adversary exists then it is not possible for $\mathsf{E}(), \mathsf{D}()$ to belong to the set $\mathsf{RO}^2(\mathcal{F}_{\mathsf{obv},L}, \mathcal{B}, \epsilon)$, which contradicts our assumption.

The adversary $\mathcal{A}$ is a polynomial time algorithm which performs at most $\mathcal{B}'$ queries to oracle $\mathsf{D}()$. At the end of its computations the algorithm returns one of the following three outputs: (i) a value y which, if passed to oracle $\mathsf{D}()$, produces an output which exhibits patterns. In this case, the algorithm is successful; (ii) a value y which, if passed to oracle $\mathsf{D}()$, produces an output which does not exhibit patterns. In this case, the algorithm is unsuccessful; (iii) an indication that the algorithm has halted before returning any value y. In this case, the algorithm is unsuccessful as well.

From algorithm $\mathcal{A}$, one can easily construct n algorithm $\mathcal{A}'$ which invokes $\mathcal{A}$ and behaves in the following way: If $\mathcal{A}$ does not halt before returning a y value, $\mathcal{A}'$ returns the same y value which $\mathcal{A}$ returns. If, on the other hand, $\mathcal{A}$ halts before returning a y value, then $\mathcal{A}'$ selects a value y_r at random and returns this value. Furthermore, algorithm $\mathcal{A}'$ maintains a table of previously returned y_r values so that every time algorithm $\mathcal{A}$ halts, the algorithm $\mathcal{A}'$ returns a different y_r value. From the definition of $\mathcal{A}'$, it is evident that algorithm $\mathcal{A}'$ also succeeds in repeatedly producing messages the outputs of which exhibit patterns with probability $> P_{\mathcal{F}_{\mathsf{obv},L}} \cdot 2^{\epsilon}$. Furthermore, algorithm $\mathcal{A}'$ never halts but always returns some y value.

The next step of the proof considers two cases for algorithm $\mathcal{A}'$ shown in Figures 21 and 22 respectively. In the first case, the algorithm $\mathcal{A}'$ as well as the underlying $\mathcal{A}$ distinguish between queries made to $\mathsf{D}()$ that assist in the computation of a returned y value and the proposal of a y value. Such distinction is supported by the control flow of algorithms $\mathcal{A}$, $\mathcal{A}'$. Because of

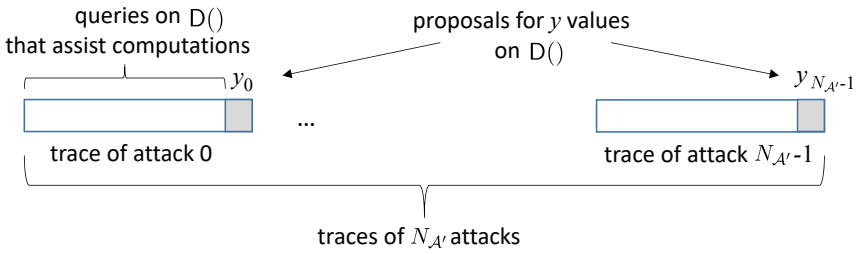


Figure 21: Hypothetical successful attacks of an input perturbing adversary $\mathcal{A}'$ (first case)

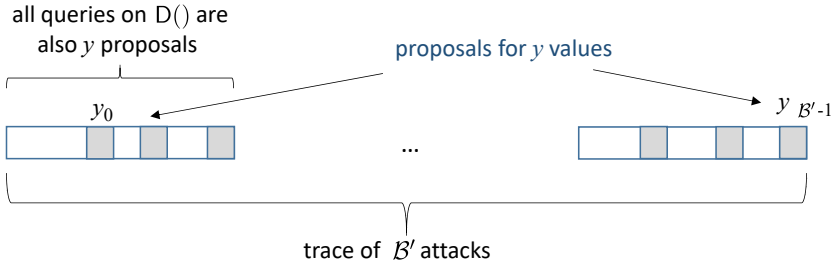


Figure 22: Hypothetical successful attacks of an input perturbing adversary $\mathcal{A}'$ (second case)

such distinction, the oracle $\mathsf{D}()$, which the adversary $\mathcal{A}'$ accesses, accepts both a final proposal for a y value, as well as queries that aid the computation of such proposal.

In the second case, there is no distinction between queries assisting the adversary computations and proposals of a y value. Nor such distinction is supported by the control flow of algorithms $\mathcal{A}$, $\mathcal{A}'$. In this case, every query is also a distinct attack, and furthermore, every query is computed based on the outcomes of the previous queries, which are also attacks.

In the first case, the adversary $\mathcal{A}'$ performs $N_{\mathcal{A}'} \leqslant \mathcal{B}$ attacks which are different from each other, each with a query budget $m+\mathcal{B}' \leqslant \mathcal{B}$. For these attacks the adversary issues all queries, but the final proposal of a computed y value, to oracle $\mathsf{D}()$ first. Then, the y values, computed from each attack, are gathered and passed to oracle $\mathsf{D}()$. The expected value of the ratio of the y values which result in patterns over all y values satisfies simultaneously $> P_{\mathcal{F}_{\mathsf{obv},L}} \cdot 2^{\epsilon}$ due to the assumption that the attacker exists and is successful and $\leqslant P_{\mathcal{F}_{\mathsf{obv},L}} \cdot 2^{\epsilon}$ due to the fact that the attacked oracle $\mathsf{D}()$ is $\mathsf{RO}^2(\mathcal{F}_{\mathsf{obv},L}, \mathcal{B}, \epsilon)$ accepting non-repeating input of maximum length $\mathcal{B}$, where each value of the input is conditioned upon at most $\mathcal{B}$ other input-output pairs, which is not possible. Such non-repeating input consists of the all the y proposals computed from all the distinct $N_{\mathcal{A}'}$ attacks, which adversary $\mathcal{A}'$ performs.

In the second case, the adversary performs $\mathcal{B}'$ attacks which are also different from each other, where $m + \mathcal{B}' \leqslant \mathcal{B}$. Furthermore each attack is also a query taken into account by a next attack. In these attacks there is no distinction between queries assisting the adversary computations and proposals for a y value. The expected value of the ratio of successful queries to the attacked oracle over all queries again satisfies simultaneously $> P_{\mathcal{F}_{\mathsf{obv},L}} \cdot 2^{\epsilon}$ and $\leqslant P_{\mathcal{F}_{\mathsf{obv},L}} \cdot 2^{\epsilon}$ which is not possible. This is because we assume that the adversary exists, and that the attacked oracle is $\mathsf{RO}^2(\mathcal{F}_{\mathsf{obv},L}, \mathcal{B}, \epsilon)$ accepting non-repeating input of maximum length $\mathcal{B}' \leqslant \mathcal{B}$. Hence Theorem 1 is proven.

The reason for distinguishing between cases 1 and 2 in this proof is because such distinction allows us to construct different sequences of non-repeating inputs in each case, where these sequences demonstrate a paradox. Such sequences need to include quite many successful queries,

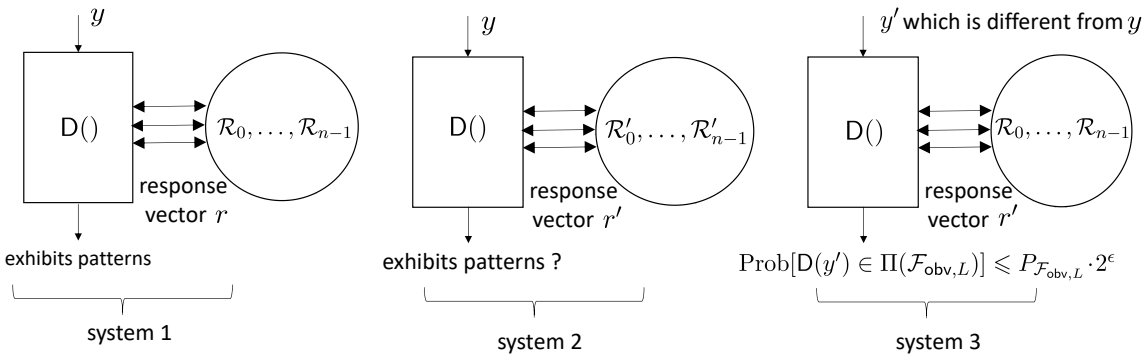


Figure 23: Replacing internal random permutations inside an RO^2 construction

due to the assumption that the adversary is successful, and at the same time quite too few due to the assumption that the construction used is $\mathsf{RO}^2(\mathcal{F}_{\mathsf{obv},L}, B, \epsilon)$.

B Proof of Theorem 2

We need to show that for every oracle replacing adversary $\mathcal{A}^{\mathsf{D}()}_{\mathsf{REPL}} \leftarrow \mathcal{A}$, it holds that:

$$\begin{aligned}
&\text{Prob}[\ \{\{\mathcal{R}'_0, \ldots, \mathcal{R}'_{n-1}\},\ y\} \ \leftarrow\ \mathcal{A}(q_0, \ldots, q_{m-1}, Q_{\mathcal{D}}, Q_{\mathcal{B}'}, \mathcal{R}_{\mathcal{D}});\\
&\qquad \text{oracle } \mathsf{D}^{\mathcal{R}'_0,\ldots,\mathcal{R}'_{n-1}}() \text{ has not been queried with input } y \text{ before};\\
&\qquad y \in \{q_0, \ldots, q_{m-1}\};\ \{\mathcal{R}'_0, \ldots, \mathcal{R}'_{n-1}\} \in \mathcal{R}_{\mathcal{D}};\ \ \mathsf{D}^{\mathcal{R}'_0,\ldots,\mathcal{R}'_{n-1}}(y) \in \Pi(\mathcal{F}_{\mathsf{obv},L})\ ]\\
&\qquad \leqslant\ \ P_{\mathcal{F}_{\mathsf{obv},L}} \cdot 2^{\epsilon}
\end{aligned} \tag{B.1}$$

We assume that an adversary exists for which the relation B.1 does not hold. This adversary repeatedly succeeds in producing random permutation replacements and inputs y the outputs of which exhibit patterns with probability greater than $P_{\mathcal{F}_{\mathsf{obv},L}} \cdot 2^{\epsilon}$. We show that if such adversary exists then it is not possible for $\mathsf{E}()$, $\mathsf{D}()$ to be $\mathsf{RO}^2(\mathcal{F}_{\mathsf{obv},L}, \mathcal{B}, \epsilon)$ which contradicts our assumption. The proof is similar as in Theorem 1. We first state and prove a lemma that bounds the probability of seeing patterns in the output of an RO^2 construction once we replace the internal random permutations.

Lemma B.1: Let's assume that we have a pair of encryption and decryption oracles $\mathsf{D}()$, $\mathsf{E}() \in \mathsf{RO}^2(\mathcal{F}_{\mathsf{obv},L}, \mathcal{B}, \epsilon)$ for some $\mathcal{F}_{\mathsf{obv},L}, \mathcal{B}, \epsilon$, $\mathsf{D}() = \mathsf{E}()^{-1}$, and some input y, such that $\mathsf{D}^{\mathcal{R}_0,\ldots,\mathcal{R}_{n-1}}(y) \in \Pi(\mathcal{F}_{\mathsf{obv},L})$. Then for any set of internal random permutation replacements $\mathcal{R}'_0, \ldots, \mathcal{R}'_{n-1}$, which are also random permutations, the probability $\text{Prob}[\mathsf{D}^{\mathcal{R}'_0,\ldots,\mathcal{R}'_{n-1}}(y) \in \Pi(\mathcal{F}_{\mathsf{obv},L})]$ of seeing patterns in the output of y is bounded by:

$$\text{Prob}[\mathsf{D}^{\mathcal{R}'_0,\ldots,\mathcal{R}'_{n-1}}(y) \in \Pi(\mathcal{F}_{\mathsf{obv},L}) \mid \mathsf{D}^{\mathcal{R}_0,\ldots,\mathcal{R}_{n-1}}(y) \in \Pi(\mathcal{F}_{\mathsf{obv},L})] \leqslant P_{\mathcal{F}_{\mathsf{obv},L}} \cdot 2^{\epsilon} \tag{B.2}$$

Proof: We consider a system 1 shown in Figure 23, where the decryption oracle $\mathsf{D}()$ accesses the original internal random permutations $\mathcal{R}_0, \ldots, \mathcal{R}_{n-1}$ and in this system one particular query to $\mathcal{R}_0, \ldots, \mathcal{R}_{n-1}$ returns a query response vector r. In another system, system 2, the

original internal random permutations $\mathcal{R}_0, \ldots, \mathcal{R}_{n-1}$ are replaced by $\mathcal{R}'_0, \ldots, \mathcal{R}'_{n-1}$, and the same query now returns a different response vector r'. In system 3 of the figure, the decryption oracle $\mathsf{D}()$ accesses the original internal random permutations $\mathcal{R}_0, \ldots, \mathcal{R}_{n-1}$, but in this system the corresponding query to $\mathcal{R}_0, \ldots, \mathcal{R}_{n-1}$ returns a response vector r', which is the same as the one returned by the permutations of system 2. As the internal random permutations $\mathcal{R}_0, \ldots, \mathcal{R}_{n-1}$ are bijective functions, they are invertible. By inverting the internal random permutations $\mathcal{R}_0, \ldots, \mathcal{R}_{n-1}$ on the response vector r', one computes an input y' which needs to be provided to system 3 in order for the internal random permutations of this system to return the same response vector r', which is returned in system 2. Due to $\mathcal{R}_0, \ldots, \mathcal{R}_{n-1}$ being bijective and the RO^2 construction constraints introduced in Figure 3, input y' must be different from y. Since $y' \neq y$, and system 3 is an RO^2 construction, then the output of system 3 exhibits patterns with probability:

$$\text{Prob}[\mathsf{D}^{\mathcal{R}_0,\ldots,\mathcal{R}_{n-1}}(y') \in \Pi(\mathcal{F}_{\mathsf{obv},L}) \mid \mathsf{D}^{\mathcal{R}_0,\ldots,\mathcal{R}_{n-1}}(y) \in \Pi(\mathcal{F}_{\mathsf{obv},L})] \leqslant P_{\mathcal{F}_{\mathsf{obv},L}} \cdot 2^{\epsilon} \tag{B.3}$$

The proof of Lemma B.1 completes by observing that systems 2 and 3 generate the same output. Hence:

$$\begin{aligned} &\text{Prob}[\mathsf{D}^{\mathcal{R}'_0,\ldots,\mathcal{R}'_{n-1}}(y) \in \Pi(\mathcal{F}_{\mathsf{obv},L}) \mid \mathsf{D}^{\mathcal{R}_0,\ldots,\mathcal{R}_{n-1}}(y) \in \Pi(\mathcal{F}_{\mathsf{obv},L})] = \\ &\text{Prob}[\mathsf{D}^{\mathcal{R}_0,\ldots,\mathcal{R}_{n-1}}(y') \in \Pi(\mathcal{F}_{\mathsf{obv},L}) \mid \mathsf{D}^{\mathcal{R}_0,\ldots,\mathcal{R}_{n-1}}(y) \in \Pi(\mathcal{F}_{\mathsf{obv},L})] \leqslant P_{\mathcal{F}_{\mathsf{obv},L}} \cdot 2^{\epsilon} \end{aligned} \tag{B.4}$$

and Lemma B.1 is proven.

We proceed with the proof of Theorem 2 by stating and proving one more Lemma:

Lemma B.2: Let's consider an ensemble of internal random permutation sets $\{\mathcal{R}_0^{(i)}, \ldots, \mathcal{R}_{n-1}^{(i)}\}$, $i \geqslant 0$. Let's also consider constructions $\mathsf{D}(), \mathsf{E}() \in \mathsf{RO}^2(\mathcal{F}_{\mathsf{obv},L}, \mathcal{B}, \epsilon)$ for some $\mathcal{F}_{\mathsf{obv},L}, \mathcal{B}, \epsilon$ and $\mathsf{D}() = \mathsf{E}()^{-1}$. We further consider a set of input indices, $J_0 = 0$, $J_1 > J_0$, $J_2 > J_1, \ldots$ for which $J_{i+1} - J_i \leqslant \mathcal{B}$ for all $i \geqslant 0$. Using the constructions $\mathsf{E}()$, $\mathsf{D}()$ and the indices $J_0, J_1, \ldots$, we define permutation swapping constructions $\mathsf{E}()'$, $\mathsf{D}()'$ as constructions that accept as input discrete sequences of values $y_0, y_1, \ldots$, have infinite lifetime as opposed to bounded by $\mathcal{B}$, and provide output which is obtained as follows:

$$\mathsf{E}'(y_j) = \mathsf{E}^{\mathcal{R}_0^{(i)},\ldots,\mathcal{R}_{n-1}^{(i)}}(y_j) \quad \text{and} \quad \mathsf{D}'(y_j) = \mathsf{D}^{\mathcal{R}_0^{(i)},\ldots,\mathcal{R}_{n-1}^{(i)}}(y_j), \quad \forall\ y_j :\ J_i \leqslant j < J_{i+1} \tag{B.5}$$

If $\mathsf{E}()'$, $\mathsf{D}()'$ are defined by equation B.5, then for any input sequence $\tilde{y}_0, \tilde{y}_1, \ldots$ to $\mathsf{D}()'$ which is not repeating inside the index bounds defined by $J_0, J_1, \ldots$ the following inequality is true:

$$\text{Prob}[\mathsf{D}'(\tilde{y}) \in \Pi(\mathcal{F}_{\mathsf{obv},L})] \leqslant P_{\mathcal{F}_{\mathsf{obv},L}} \cdot 2^{\epsilon} \tag{B.6}$$

where $\tilde{y}_{j'} \neq \tilde{y}_{j''}$ for all $J_i \leqslant j' < J_{i+1}$, $J_i \leqslant j'' < J_{i+1}$, $j' \neq j''$ and $i \geqslant 0$, and where the relation B.6 holds for every input $\tilde{y} \in \{\tilde{y}_0, \tilde{y}_1, \ldots\}$.

Proof: From the definition of equation B.5 it is evident that $\mathsf{E}()'$ and $\mathsf{D}()'$ are also encryption and decryption oracles and that $\mathsf{D}()' = \mathsf{E}()'^{-1}$. The inputs to decryption oracle $\mathsf{D}()'$ can either result in outputs with patterns or not. On the other hand, the inputs that result in patterns can be split into repeating inputs and non-repeating inputs, as inputs from the sequence $\tilde{y}_0, \tilde{y}_1, \ldots$

to $\mathsf{D}()'$ may be repeating across index bounds. It is sufficient to show that the property $\text{Prob}[\mathsf{D}'(\tilde{y}) \in \Pi(\mathcal{F}_{\mathsf{obv},L})] \leqslant P_{\mathcal{F}_{\mathsf{obv},L}} \cdot 2^{\epsilon}$ holds for the repeating inputs which produce at least one output with patterns. This is because the probability $\text{Prob}[\mathsf{D}'(\tilde{y}) \in \Pi(\mathcal{F}_{\mathsf{obv},L})]$ is trivially 0 if it is known that inputs are always unsuccessful. On the other hand, for the non-repeating inputs the property does hold, as the constructions defined by $\mathsf{E}()'$, $\mathsf{D}()'$ are RO^2 inside the index bounds.

It is easy to see that, for the remaining case of repeating inputs that produce at least one output with patterns, the property $\text{Prob}[\mathsf{D}'(\tilde{y}) \in \Pi(\mathcal{F}_{\mathsf{obv},L})] \leqslant P_{\mathcal{F}_{\mathsf{obv},L}} \cdot 2^{\epsilon}$ holds due to Lemma B.1. Indeed let's consider some input $\tilde{y}_j$ such that $J_i \leqslant j < J_{i+1}$ for some $i \geqslant 0$. Let's also consider that this input, if passed to the decryption oracle $\mathsf{D}()'$, produces output that exhibits patterns:

$$\mathsf{D}'(\tilde{y}_j) = \mathsf{D}^{\mathcal{R}_0^{(i)},\ldots,\mathcal{R}_{n-1}^{(i)}}(\tilde{y}_j) \in \Pi(\mathcal{F}_{\mathsf{obv},L}) \tag{B.7}$$

If this input $\tilde{y}_j$ appears in the input sequence again, outside the index bounds J_i and J_{i+1}, then the probability of seeing patterns at the output of this repeated instance of $\tilde{y}_j$ is bounded according to Lemma B.1. Specifically, if value $\tilde{y}_j$ appears again inside the index bounds $J_{i'}$ and $J_{i'+1}$ for some $i' \neq i$, then the output of the decryption oracle $\mathsf{D}()'$ for this repeated instance is equal to $\mathsf{D}^{\mathcal{R}_0^{(i')},\ldots,\mathcal{R}_{n-1}^{(i')}}(\tilde{y}_j)$. If we apply Lemma B.1 to the sets of internal random permutations $\{\mathcal{R}_0^{(i)},\ldots,\mathcal{R}_{n-1}^{(i)}\}$ and $\{\mathcal{R}_0^{(i')},\ldots,\mathcal{R}_{n-1}^{(i')}\}$ and to the input value $\tilde{y}_j$, we obtain inequality:

$$\text{Prob}[\mathsf{D}^{\mathcal{R}_0^{(i')},\ldots,\mathcal{R}_{n-1}^{(i')}}(\tilde{y}_j) \in \Pi(\mathcal{F}_{\mathsf{obv},L}) \mid \mathsf{D}^{\mathcal{R}_0^{(i)},\ldots,\mathcal{R}_{n-1}^{(i)}}(\tilde{y}_j) \in \Pi(\mathcal{F}_{\mathsf{obv},L})] \leqslant P_{\mathcal{F}_{\mathsf{obv},L}} \cdot 2^{\epsilon} \tag{B.8}$$

which completes the proof of Lemma B.2.

We also note that the inequality B.6 of Lemma B.2 holds even if the event $\mathsf{D}'(\tilde{y}) \in \Pi(\mathcal{F}_{\mathsf{obv},L})$ is conditioned upon inputs to $\mathsf{D}()'$ and their responses, where inputs are different from $\tilde{y}$ and there are at most $\mathcal{B}$ inputs associated with the same set of random permutations. This is due to two facts. First, that the construction $\mathsf{D}()'$ is RO^2 inside the index bounds. Second, that the probability of seeing patterns in the output of one lifetime of $\mathsf{D}()$ (i.e., inside one set of index bounds of $\mathsf{D}()'$) is conditionally independent of inputs and outputs that appear in other lifetimes of $\mathsf{D}()$, where different sets of internal random permutations may be queried. This is because the internal random permutations of different lifetimes are independently drawn.

Completing the proof of Theorem 2:

Any instance of $\mathsf{D}()$ that queries internal random permutations from one of the sets of $\mathcal{R}_{\mathcal{D}}$ is RO^2 by the definition of $\mathsf{D}()$ and due to the fact that the permutations contained in the sets of $\mathcal{R}_{\mathcal{D}}$ are random permutations. Similarly, any permutation swapping construction $\mathsf{D}()'$ which is produced from $\mathsf{D}()$ and has infinite lifetime, exhibits patterns in its output with probability which is bounded according to Lemma B.2, provided that the input is non-repeating inside index bounds.

Now, let's suppose we have an oracle replacing adversary $\mathcal{A}$, which is repeatedly successful with probability higher than the bound $P_{\mathcal{F}_{\mathsf{obv},L}} \cdot 2^{\epsilon}$ of relation B.1. In every attack, this adversary succeeds in computing a different y value and a different set of internal random permutations from $\mathcal{R}_{\mathcal{D}}$ all resulting in patterns with probability $> P_{\mathcal{F}_{\mathsf{obv},L}} \cdot 2^{\epsilon}$. Furthermore, as in the proof of Theorem 1, this adversary $\mathcal{A}$ can be turned into another adversary $\mathcal{A}'$ which always returns some output and is still successful in attacking $\mathsf{D}()$ with probability $> P_{\mathcal{F}_{\mathsf{obv},L}} \cdot 2^{\epsilon}$. We consider

that such attacks are repeated again and again. One can see that the attacks performed by adversary $\mathcal{A}'$ form a trace of queries to a permutation swapping construction $\mathsf{D}()'$ as defined in Lemma B.2. The expected value $\mathsf{E}()$ of the ratio of successful queries to $\mathsf{D}()'$ over all queries made needs to satisfy the inequality $E > P_{\mathcal{F}_{\mathsf{obv},L}} \cdot 2^{\epsilon}$, due to the assumption about the existence of the adversary $\mathcal{A}'$. On the other hand, the same expected value needs to satisfy the inequality $E \leqslant P_{\mathcal{F}_{\mathsf{obv},L}} \cdot 2^{\epsilon}$ due to the fact that Lemma B.2 holds, which is not possible. Hence, Theorem 2 is proven.

C Proof of the correctness of Proposition 2

For ease of notation, we refer to the event $\mathsf{IVP}(y) \in \Pi(\mathcal{F}_{\mathsf{EQ},512,16,4})$ as $E^{(\mathcal{F}_{\mathsf{EQ},512,16,4})}$. The probability of this event can be expressed as the sum of the probabilities of nine subevents, which are mutually exclusive:

$$
\begin{aligned}
\mathrm{Prob}[E^{(\mathcal{F}_{\mathsf{EQ},512,16,4})}] = \mathrm{Prob}[\ & [\text{seeing the } \mathcal{F}_{\mathsf{EQ},512,16,4} \text{ pattern in the } \mathsf{IVP} \text{ output}] \wedge \\
& [\text{no uncovered words}] \wedge [\text{no uncovered bytes}]\ ] + \\
\mathrm{Prob}[\ & [\text{seeing the } \mathcal{F}_{\mathsf{EQ},512,16,4} \text{ pattern in the } \mathsf{IVP} \text{ output}] \wedge \\
& [\text{having exactly 1 uncovered word in the } \mathsf{IVP} \text{ output}]\ ] + \\
\mathrm{Prob}[\ & [\text{seeing the } \mathcal{F}_{\mathsf{EQ},512,16,4} \text{ pattern in the } \mathsf{IVP} \text{ output}] \wedge \\
& [\text{having exactly 2 uncovered words in the } \mathsf{IVP} \text{ output}]\ ] + \\
\mathrm{Prob}[\ & [\text{seeing the } \mathcal{F}_{\mathsf{EQ},512,16,4} \text{ pattern in the } \mathsf{IVP} \text{ output}] \wedge \\
& [\text{having exactly 3 uncovered words in the } \mathsf{IVP} \text{ output}]\ ] + \\
\mathrm{Prob}[\ & [\text{seeing the } \mathcal{F}_{\mathsf{EQ},512,16,4} \text{ pattern in the } \mathsf{IVP} \text{ output}] \wedge \\
& [\text{having 4 or more uncovered words in the } \mathsf{IVP} \text{ output}]\ ] + \\
\mathrm{Prob}[\ & [\text{seeing the } \mathcal{F}_{\mathsf{EQ},512,16,4} \text{ pattern in the } \mathsf{IVP} \text{ output}] \wedge \\
& [\text{no uncovered words}] \wedge [\text{exactly 1 uncovered byte in } \mathsf{IVP} \text{ output}]\ ] + \\
\mathrm{Prob}[\ & [\text{seeing the } \mathcal{F}_{\mathsf{EQ},512,16,4} \text{ pattern in the } \mathsf{IVP} \text{ output}] \wedge \\
& [\text{no uncovered words}] \wedge [\text{exactly 2 uncovered bytes in } \mathsf{IVP} \text{ output}]\ ] + \\
\mathrm{Prob}[\ & [\text{seeing the } \mathcal{F}_{\mathsf{EQ},512,16,4} \text{ pattern in the } \mathsf{IVP} \text{ output}] \wedge \\
& [\text{no uncovered words}] \wedge [\text{exactly 3 uncovered bytes in } \mathsf{IVP} \text{ output}]\ ] + \\
\mathrm{Prob}[\ & [\text{seeing the } \mathcal{F}_{\mathsf{EQ},512,16,4} \text{ pattern in the } \mathsf{IVP} \text{ output}] \wedge \\
& [\text{no uncovered words}] \wedge [\text{4 or more uncovered bytes in } \mathsf{IVP} \text{ output}]\ ]
\end{aligned}
\tag{C.1}
$$

To facilitate computations, we also introduce the following notation to refer to events and subevents associated with $E^{(\mathcal{F}_{\mathsf{EQ},512,16,4})}$:

i $E^{(\mathcal{F}_{\mathsf{EQ},512,16,4})}(N)$: seeing the $\mathcal{F}_{\mathsf{EQ},512,16,4}$ pattern in N out of 16 words of the IVP output.

ii $E^{(\ \mathsf{ex\text{-}uw})}(n, N)$: having exactly n uncovered words among N in the output of the IVP construction.

iii $E^{(\ \mathsf{at\text{-}uw})}(n, N)$: having at least n uncovered words among N in the output of the IVP construction.

iv $E^{(\ \mathsf{at\text{-}eqw})}(n, N, p)$: having a set of at least n equal, covered or partially covered words among N covered or partially covered ones in the output of the IVP construction. If $p = 0$ then each of the words of the set is either covered of partially covered. If $p = 1$, then all words of the set, as well as all N words are covered.

v $E^{(\ \mathsf{at\text{-}eqw\text{-}v})}(n, v, N, p)$: having a set of at least n equal, covered or partially covered words among N covered or partially covered ones in the output of the IVP construction. The value of these words is v. If $p = 0$ then each of the words of the set is either covered of partially covered. If $p = 1$, then all words of the set, as well as all N words are covered.

vi $E^{(\ \mathsf{diff\text{-}uw})}(v)$: having an uncovered word in the output of the IVP construction with value different from v.

vii $E^{(\ \mathsf{eq\text{-}uw})}(v)$: having an uncovered word in the output of the IVP construction with value equal to v.

viii $E^{(\ \mathsf{ex\text{-}ub})}(n, N)$: having no uncovered words and exactly n uncovered bytes among N in the output of the IVP construction.

ix $E^{(\ \mathsf{at\text{-}ub})}(n, N)$: having no uncovered words and at least n uncovered bytes among N in the output of the IVP construction.

x $E^{(\ \mathsf{eq\text{-}ub})}(v)$: having an uncovered byte in the output of the IVP construction with value equal to v.

xi $E^{(\ \mathsf{ex\text{-}uw\text{-}pat})}(n_0, n_1, N)$: having exactly n_0+n_1 uncovered words among N in the output of the IVP construction, of which n_0 words are elements of a set exhibiting the $\mathcal{F}_{\mathsf{EQ},512,16,4}$ pattern and the remaining n_1 are not.

xii $E^{(\ \mathsf{ex\text{-}ub\text{-}pat})}(n_0, n_1, N)$: having exactly n_0+n_1 uncovered bytes among N in the output of the IVP construction, of which n_0 bytes are elements of a set exhibiting the $\mathcal{F}_{\mathsf{EQ},512,16,4}$ pattern and the remaining n_1 are not.

Using the notation above, we rewrite equation C.1 as follows:

$$\begin{aligned}
\mathrm{Prob}[E^{(\mathcal{F}_{\mathsf{EQ},512,16,4})}] = \mathrm{Prob}[E^{(\mathcal{F}_{\mathsf{EQ},512,16,4})}(32) \ \wedge\ E^{(\ \mathsf{ex\text{-}uw})}(0,32) \ \wedge\ E^{(\ \mathsf{ex\text{-}ub})}(0,64)\,] + \\
\mathrm{Prob}[E^{(\mathcal{F}_{\mathsf{EQ},512,16,4})}(32) \ \wedge\ E^{(\ \mathsf{ex\text{-}uw})}(1,32)\,] + \mathrm{Prob}[E^{(\mathcal{F}_{\mathsf{EQ},512,16,4})}(32) \ \wedge\ E^{(\ \mathsf{ex\text{-}uw})}(2,32)\,] + \\
\mathrm{Prob}[E^{(\mathcal{F}_{\mathsf{EQ},512,16,4})}(32) \ \wedge\ E^{(\ \mathsf{ex\text{-}uw})}(3,32)\,] + \mathrm{Prob}[E^{(\mathcal{F}_{\mathsf{EQ},512,16,4})}(32) \ \wedge\ E^{(\ \mathsf{at\text{-}uw})}(4,32)\,] + \\
\mathrm{Prob}[E^{(\mathcal{F}_{\mathsf{EQ},512,16,4})}(32) \ \wedge\ E^{(\ \mathsf{ex\text{-}uw})}(0,32) \ \wedge\ E^{(\ \mathsf{ex\text{-}ub})}(1,64)\,] + \\
\mathrm{Prob}[E^{(\mathcal{F}_{\mathsf{EQ},512,16,4})}(32) \ \wedge\ E^{(\ \mathsf{ex\text{-}uw})}(0,32) \ \wedge\ E^{(\ \mathsf{ex\text{-}ub})}(2,64)\,] + \\
\mathrm{Prob}[E^{(\mathcal{F}_{\mathsf{EQ},512,16,4})}(32) \ \wedge\ E^{(\ \mathsf{ex\text{-}uw})}(0,32) \ \wedge\ E^{(\ \mathsf{ex\text{-}ub})}(3,64)\,] + \\
\mathrm{Prob}[E^{(\mathcal{F}_{\mathsf{EQ},512,16,4})}(32) \ \wedge\ E^{(\ \mathsf{ex\text{-}uw})}(0,32) \ \wedge\ E^{(\ \mathsf{at\text{-}ub})}(4,64)\,]
\end{aligned} \tag{C.2}$$

We further define probabilities $P_0, P_1, \ldots, P_8$ as shown in equation C.3:

$$
\begin{aligned}
P_0 &= \text{Prob}[E^{(\mathcal{F}_{\mathsf{EQ},512,16,4})}(32) \ \wedge \ E^{(\ \mathsf{ex\text{-}uw})}(0,32) \ \wedge \ E^{(\ \mathsf{ex\text{-}ub})}(0,64)\,] + \\
P_1 &= \text{Prob}[E^{(\mathcal{F}_{\mathsf{EQ},512,16,4})}(32) \ \wedge \ E^{(\ \mathsf{ex\text{-}uw})}(1,32)\,] \\
P_2 &= \text{Prob}[E^{(\mathcal{F}_{\mathsf{EQ},512,16,4})}(32) \ \wedge \ E^{(\ \mathsf{ex\text{-}uw})}(2,32)\,] \\
P_3 &= \text{Prob}[E^{(\mathcal{F}_{\mathsf{EQ},512,16,4})}(32) \ \wedge \ E^{(\ \mathsf{ex\text{-}uw})}(3,32)\,] \\
P_4 &= \text{Prob}[E^{(\mathcal{F}_{\mathsf{EQ},512,16,4})}(32) \ \wedge \ E^{(\ \mathsf{at\text{-}uw})}(4,32)\,] \\
P_5 &= \text{Prob}[E^{(\mathcal{F}_{\mathsf{EQ},512,16,4})}(32) \ \wedge \ E^{(\ \mathsf{ex\text{-}uw})}(0,32) \ \wedge \ E^{(\ \mathsf{ex\text{-}ub})}(1,64)\,] \\
P_6 &= \text{Prob}[E^{(\mathcal{F}_{\mathsf{EQ},512,16,4})}(32) \ \wedge \ E^{(\ \mathsf{ex\text{-}uw})}(0,32) \ \wedge \ E^{(\ \mathsf{ex\text{-}ub})}(2,64)\,] \\
P_7 &= \text{Prob}[E^{(\mathcal{F}_{\mathsf{EQ},512,16,4})}(32) \ \wedge \ E^{(\ \mathsf{ex\text{-}uw})}(0,32) \ \wedge \ E^{(\ \mathsf{ex\text{-}ub})}(3,64)\,] \\
P_8 &= \text{Prob}[E^{(\mathcal{F}_{\mathsf{EQ},512,16,4})}(32) \ \wedge \ E^{(\ \mathsf{ex\text{-}uw})}(0,32) \ \wedge \ E^{(\ \mathsf{at\text{-}ub})}(4,64)\,]
\end{aligned}
\tag{C.3}
$$

so that:

$$
\text{Prob}[E^{(\mathcal{F}_{\mathsf{EQ},512,16,4})}] = \sum_{i=0}^{8} P_i \tag{C.4}
$$

Lemma C.1 The probability P_1 defined in equation C.3 is bounded by $2^{-39.959} + O(\Delta_1)$ where $\Delta_1 \leqslant 2^{-52.626}$.

$$
P_1 \leqslant 2^{-39.959} + O(\Delta_1), \quad \Delta_1 \leqslant 2^{-52.626} \tag{C.5}
$$

Proof: The probability P_1 can be expressed as the sum of the probabilities of two mutually exclusive subevents. The first subevent is the event in which the $\mathcal{F}_{\mathsf{EQ},512,16,4}$ pattern is visible in the output of the IVP construction, in which there is exactly one uncovered word in this output and in which the value of this uncovered word is not part of the visible pattern. The second subevent is the event in which the $\mathcal{F}_{\mathsf{EQ},512,16,4}$ pattern is also visible in the output of the IVP construction, in which there is exactly one uncovered word in the output, and in which the value of this uncovered word is now part of the visible pattern:

$$
\begin{aligned}
P_1 = \text{Prob}[E^{(\mathcal{F}_{\mathsf{EQ},512,16,4})}(32) \ \wedge \ E^{(\ \mathsf{ex\text{-}uw})}(1,32) \ \wedge E^{(\ \mathsf{ex\text{-}uw\text{-}pat})}(0,1,32)\,] + \\
\text{Prob}[E^{(\mathcal{F}_{\mathsf{EQ},512,16,4})}(32) \ \wedge \ E^{(\ \mathsf{ex\text{-}uw})}(1,32) \ \wedge E^{(\ \mathsf{ex\text{-}uw\text{-}pat})}(1,0,32)\,]
\end{aligned}
\tag{C.6}
$$

Equation C.6 can be rewritten as:

$$
\begin{aligned}
P_1 = \text{Prob}[E^{(\ \mathsf{ex\text{-}uw})}(1,32)] \cdot \\
\text{Prob}[E^{(\mathcal{F}_{\mathsf{EQ},512,16,4})}(32) \ \wedge \ E^{(\ \mathsf{ex\text{-}uw\text{-}pat})}(0,1,32) \mid E^{(\ \mathsf{ex\text{-}uw})}(1,32)] + \\
\text{Prob}[E^{(\ \mathsf{ex\text{-}uw})}(1,32)] \cdot \\
\text{Prob}[E^{(\mathcal{F}_{\mathsf{EQ},512,16,4})}(32) \ \wedge \ E^{(\ \mathsf{ex\text{-}uw\text{-}pat})}(1,0,32) \mid E^{(\ \mathsf{ex\text{-}uw})}(1,32)]
\end{aligned}
\tag{C.7}
$$

Next, we observe that, if it is known that there is one uncovered word in the output of the IVP construction, then the event of having a visible pattern in the output and that the uncovered word is not part of the pattern is the same as the event in which there are 4 or more

words equal to each other among 31 in the output of IVP and in which the uncovered word is not part of the pattern. Similarly, the event of having a visible pattern in the output and in which the uncovered word is part of the pattern is the same as the event in which there are 3 or more words equal to each other among 31 in the output of IVP and in which the uncovered word is part of the pattern.

$$\begin{aligned} P_1 = \mathrm{Prob}[E^{(\ \textsf{ex-uw})}(1,32)] \cdot \\ \mathrm{Prob}[E^{(\ \textsf{at-eqw})}(4,31,0) \ \wedge \ E^{(\ \textsf{ex-uw-pat})}(0,1,32) \mid E^{(\ \textsf{ex-uw})}(1,32)] + \\ \mathrm{Prob}[E^{(\ \textsf{ex-uw})}(1,32)] \cdot \\ \mathrm{Prob}[E^{(\ \textsf{at-eqw})}(3,31,0) \ \wedge \ E^{(\ \textsf{ex-uw-pat})}(1,0,32) \mid E^{(\ \textsf{ex-uw})}(1,32)] \end{aligned} \tag{C.8}$$

In the next step of the proof, we express the events $E^{(\ \textsf{at-eqw})}(4,31,0)$, $E^{(\ \textsf{at-eqw})}(3,31,0)$ as a union of mutually exclusive subevents $E^{(\ \textsf{at-eqw-v})}(4,v,31,0)$ and $E^{(\ \textsf{at-eqw-v})}(3,v,31,0)$, where, in each subevent, the elements of sets of at least 3 or 4 words are equal to a specific value v.

$$\begin{aligned} P_1 \leqslant \mathrm{Prob}[E^{(\ \textsf{ex-uw})}(1,32)] \cdot \\ \sum_v \mathrm{Prob}[E^{(\ \textsf{at-eqw-v})}(4,v,31,0) \ \wedge \ E^{(\ \textsf{ex-uw-pat})}(0,1,32) \mid E^{(\ \textsf{ex-uw})}(1,32)] + \\ \mathrm{Prob}[E^{(\ \textsf{ex-uw})}(1,32)] \cdot \\ \sum_v \mathrm{Prob}[E^{(\ \textsf{at-eqw-v})}(3,v,31,0) \ \wedge \ E^{(\ \textsf{ex-uw-pat})}(1,0,32) \mid E^{(\ \textsf{ex-uw})}(1,32)] \end{aligned} \tag{C.9}$$

Relation C.9 can be further written as:

$$\begin{aligned} P_1 \leqslant \mathrm{Prob}[E^{(\ \textsf{ex-uw})}(1,32)] \cdot \\ \sum_v \big(\mathrm{Prob}[E^{(\ \textsf{at-eqw-v})}(4,v,31,0) \ \wedge \ E^{(\ \textsf{diff-uw})}(v) \mid E^{(\ \textsf{ex-uw})}(1,32)] + \\ \mathrm{Prob}[E^{(\ \textsf{at-eqw-v})}(3,v,31,0) \ \wedge \ E^{(\ \textsf{eq-uw})}(v) \mid E^{(\ \textsf{ex-uw})}(1,32)]\big) \end{aligned} \tag{C.10}$$

Since the uncovered values of bytes are almost statistically independent from the values of covered words, each probability term, which is present in relation C.10, can be expressed as a product of two, as shown below. Moreover, the state of each word in the output of the IVP construction, as being covered or uncovered, is almost statistically independent from the state of other words. Because of this reason, the condition $E^{(\ \textsf{ex-uw})}(1,32)$ can be removed from the probabilities of events that describe equality of covered word values, and which are present in relation C.10. These changes to relation C.10 require the addition of the correcting term $O(\Delta)$, $\Delta \leqslant 2^{-64}$, which is the distinguishing advantage associated with the internal random permutations of IVP:

$$P_1 \leqslant \text{Prob}[E^{(\ \textsf{ex-uw})}(1,32)] \cdot \sum_v \big(\text{Prob}[E^{(\ \textsf{at-eqw-v})}(4,v,31,0)] \ \cdot\ \text{Prob}[E^{(\ \textsf{diff-uw})}(v) \mid E^{(\ \textsf{ex-uw})}(1,32)] + \text{Prob}[E^{(\ \textsf{at-eqw-v})}(3,v,31,0)] \ \cdot\ \text{Prob}[E^{(\ \textsf{eq-uw})}(v) \mid E^{(\ \textsf{ex-uw})}(1,32)]\big) \ +\ O(\Delta) \tag{C.11}$$

We conclude the proof observing that the probability terms $\text{Prob}[E^{(\ \textsf{at-eqw-v})}(4,v,31,0)]$ and $\text{Prob}[E^{(\ \textsf{at-eqw-v})}(3,v,31,0)]$ are bounded the same way for every value of v. Because of this reason, these probability terms can be taken out of the summation, where the term v can be replaced by some term v_0 that represents any value from 0 to 65536. We also observe that the probability $\text{Prob}[E^{(\ \textsf{diff-uw})}(v) \mid E^{(\ \textsf{ex-uw})}(1,32)]$ can be replaced by 1 in order to obtain an upper bound and that $\sum_v \text{Prob}[E^{(\ \textsf{eq-uw})}(v) \mid E^{(\ \textsf{ex-uw})}(1,32)] = 1$. As a result P_1 is bounded by:

$$P_1 \leqslant \text{Prob}[E^{(\ \textsf{ex-uw})}(1,32)] \cdot \big(65536 \cdot \text{Prob}[E^{(\ \textsf{at-eqw-v})}(4,v_0,31,0)] + \text{Prob}[E^{(\ \textsf{at-eqw-v})}(3,v_0,31,0)]\big) \ +\ O(\Delta) \tag{C.12}$$

The term $\text{Prob}[E^{(\ \textsf{ex-uw})}(1,32)]$ is the probability of having one uncovered word in the output of the IVP construction in any location, and is equal to $32 \cdot (2^{-16} + O(2^{-40}))$. This follows from Lemmas 1-4 and the fact that there are 32 word locations. On the other hand, the terms $\text{Prob}[E^{(\ \textsf{at-eqw-v})}(4,v_0,31,0)]$ and $\text{Prob}[E^{(\ \textsf{at-eqw-v})}(3,v_0,31,0)]$ are determined by the probability that a single word, either covered of partially covered, is equal to value v_0. From Lemmas 1-4 and Corollary 2, it follows that such probability is bounded by $(2^{-16} + O(2^{-64})) + (2^{-8} + O(2^{-64}))^2 = 2^{-15} + O(2^{-64})$. Based on this fact, the probability term $\text{Prob}[E^{(\ \textsf{at-eqw-v})}(4,v_0,31,0)]$ is bounded by $2^{-45.059} + O(2^{-57.626})$. The term $2^{-45.059}$ corresponds to the most dominant subevent where there is exactly one set with 4 words equal to v_0 that are covered or partially covered. The term $O(2^{-57.626})$ corresponds to the next most dominant subevent where there is exactly one set with 5 words equal to v_0 that are covered or partially covered. This term also includes the probabilities of other subevents where even more words are equal to v_0. The probability term $\text{Prob}[E^{(\ \textsf{at-eqw-v})}(3,v_0,31,0)]$ is similarly be bounded by $2^{-32.866} + O(2^{-45.059})$. From these bounds and relation C.12, it holds that:

$$P_1 \leqslant 2^{-39.959} + O(2^{-52.626}) \tag{C.13}$$

A number of subsequent lemmas are stated, the proof of which is similar to the proof of Lemma C.1.

Lemma C.2 The probability P_2 defined in equation C.3 is bounded by $2^{-44.281} + O(\Delta_2)$ where $\Delta_2 \leqslant 2^{-52.304}$.

$$P_2 \leqslant 2^{-44.281} + O(\Delta_2) \tag{C.14}$$

Lemma C.3 The probability P_3 defined in equation C.3 is bounded by $2^{-45.866} + O(\Delta_3)$ where $\Delta_3 \leqslant 2^{-55.474}$.

$$P_3 \leqslant 2^{-45.866} + O(\Delta_3) \tag{C.15}$$

A next lemma refers to the case where there are no uncovered words but only a single uncovered byte.

Lemma C.4 The probability P_5 defined in equation C.3 is bounded by $2^{-34.866} + O(\Delta_5)$ where $\Delta_5 \leqslant 2^{-48.626}$.

$$P_5 \leqslant 2^{-34.866} + O(\Delta_5) \tag{C.16}$$

Proof: The probability P_5 can be expressed as the sum of the probabilities of two mutually exclusive subevents. The first subevent is the event in which the $\mathcal{F}_{\mathsf{EQ},512,16,4}$ pattern is visible in the output of the IVP construction, in which there are no uncovered words but one uncovered byte in this output, and in which this uncovered byte is not part of the visible pattern. The second subevent is the event in which the $\mathcal{F}_{\mathsf{EQ},512,16,4}$ pattern is also visible in the output of the IVP construction, in which there are no uncovered words but one uncovered byte in the output, and in which the value of this uncovered byte is part of the visible pattern. According to such consideration, the probability P_5 can be written as:

$$\begin{aligned} P_5 = \mathrm{Prob}[&E^{(\ \mathsf{ex\text{-}uw})}(0,32) \ \wedge\ E^{(\ \mathsf{ex\text{-}ub})}(1,64)] \cdot \\ &\mathrm{Prob}[E^{(\mathcal{F}_{\mathsf{EQ},512,16,4})}(32) \ \wedge\ E^{(\ \mathsf{ex\text{-}ub\text{-}pat})}(0,1,64) \mid E^{(\ \mathsf{ex\text{-}uw})}(0,32) \ \wedge\ E^{(\ \mathsf{ex\text{-}ub})}(1,64)] + \\ &\mathrm{Prob}[E^{(\ \mathsf{ex\text{-}uw})}(0,32) \ \wedge\ E^{(\ \mathsf{ex\text{-}ub})}(1,64)] \cdot \\ &\mathrm{Prob}[E^{(\mathcal{F}_{\mathsf{EQ},512,16,4})}(32) \ \wedge\ E^{(\ \mathsf{ex\text{-}ub\text{-}pat})}(1,0,64) \mid E^{(\ \mathsf{ex\text{-}uw})}(0,32) \ \wedge\ E^{(\ \mathsf{ex\text{-}ub})}(1,64)] \end{aligned} \tag{C.17}$$

Next, as in the proof of Lemma C.1, we represent the probability of the event intersection $E^{(\ \mathsf{ex\text{-}uw})}(0,32) \ \wedge E^{(\ \mathsf{ex\text{-}ub})}(1,64)$ as a separate factor, while at the same time introducing conditional probabilities for the mutually exclusive subevents where the uncovered byte is, or is not part of the visible pattern.

$$\begin{aligned} P_5 = \mathrm{Prob}[&E^{(\ \mathsf{ex\text{-}uw})}(0,32) \ \wedge\ E^{(\ \mathsf{ex\text{-}ub})}(1,64)] \cdot \\ &\big(\mathrm{Prob}[E^{(\ \mathsf{at\text{-}eqw})}(4,31,1) \ \wedge\ E^{(\ \mathsf{ex\text{-}ub\text{-}pat})}(0,1,64) \mid E^{(\ \mathsf{ex\text{-}uw})}(0,32) \ \wedge\ E^{(\ \mathsf{ex\text{-}ub})}(1,64)] + \\ &\mathrm{Prob}[E^{(\ \mathsf{at\text{-}eqw})}(3,31,1) \ \wedge\ E^{(\ \mathsf{ex\text{-}ub\text{-}pat})}(1,0,64) \mid E^{(\ \mathsf{ex\text{-}uw})}(0,32) \ \wedge\ E^{(\ \mathsf{ex\text{-}ub})}(1,64)]\big) \end{aligned} \tag{C.18}$$

Next, express the events $E^{(\ \mathsf{at\text{-}eqw})}(4,31,1)$ and $E^{(\ \mathsf{at\text{-}eqw})}(3,31,1)$ as a union of mutually exclusive subevents $E^{(\ \mathsf{at\text{-}eqw\text{-}v})}(4,v|w,31,1)$ and $E^{(\ \mathsf{at\text{-}eqw\text{-}v})}(3,v|w,31,1)$, where, in each subevent, the elements of sets of at least 3 or 4 words are, not only equal to each other, but also equal to the word $v|w$, which results from the concatenation of two specific byte values v and w:

$$
\begin{aligned}
P_5 \leqslant\ & \mathrm{Prob}[E^{(\ \textsf{ex-uw})}(0,32)\ \wedge\ E^{(\ \textsf{ex-ub})}(1,64)]\cdot \\
& \big(\ \sum_{v|w} \mathrm{Prob}[E^{(\ \textsf{at-eqw-v})}(4,v|w,31,1)\ \wedge\ E^{(\ \textsf{ex-ub-pat})}(0,1,64) \\
& \mid\ E^{(\ \textsf{ex-uw})}(0,32)\ \wedge\ E^{(\ \textsf{ex-ub})}(1,64)] + \\
& \sum_{v|w} \mathrm{Prob}[E^{(\ \textsf{at-eqw-1})}(3,v|w,31,1)\ \wedge\ E^{(\ \textsf{ex-ub-pat})}(1,0,64) \\
& \mid\ E^{(\ \textsf{ex-uw})}(0,32)\ \wedge\ E^{(\ \textsf{ex-ub})}(1,64)]\ \big)
\end{aligned}
\tag{C.19}
$$

We proceed with the proof taking into account the almost statistical independence between the values of uncovered bytes, and the values of covered bytes or words. We also take into account the fact that the state of each byte or word in the output of the IVP construction, as being covered or uncovered, is almost statistically independent from the state and value of any other byte or word:

$$
\begin{aligned}
P_5 \leqslant\ & \mathrm{Prob}[E^{(\ \textsf{ex-uw})}(0,32)\ \wedge\ E^{(\ \textsf{ex-ub})}(1,64)]\cdot \\
& \sum_{v|w} \big(\ \mathrm{Prob}[E^{(\ \textsf{at-eqw-v})}(4,v|w,31,1)]\cdot \\
& \mathrm{Prob}[E^{(\ \textsf{ex-ub-pat})}(0,1,64)\mid\ E^{(\ \textsf{ex-uw})}(0,32)\ \wedge\ E^{(\ \textsf{ex-ub})}(1,64)] + \\
& \mathrm{Prob}[E^{(\ \textsf{at-eqw-v})}(3,v|w,31,1)]\cdot \\
& \mathrm{Prob}[E^{(\ \textsf{ex-ub-pat})}(1,0,64)\mid\ E^{(\ \textsf{ex-uw})}(0,32)\ \wedge\ E^{(\ \textsf{ex-ub})}(1,64)]\ \big) + O(\Delta)
\end{aligned}
\tag{C.20}
$$

The probability terms $\mathrm{Prob}[E^{(\ \textsf{at-eqw-v})}(4,v|w,31,1)]$ and $\mathrm{Prob}[E^{(\ \textsf{at-eqw-v})}(3,v|w,31,1)]$ are bounded independently of the exact values of v and w. These terms can be taken out of the summation, replacing v and w with v_0 and w_0 respectively in order to refer to any value of v, w. We also observe that the term $\mathrm{Prob}[E^{(\ \textsf{ex-ub-pat})}(0,1,64)\mid E^{(\ \textsf{ex-uw})}(0,32)\wedge E^{(\ \textsf{ex-ub})}(1,64)]$ can be bounded by 1. Moreover, it holds that $\sum_{v|w}\mathrm{Prob}[E^{(\ \textsf{ex-ub-pat})}(1,0,64)|E^{(\ \textsf{ex-uw})}(0,32)\wedge E^{(\ \textsf{ex-ub})}(1,64)] = 1$. Based on these observations, the relation C.20 can be simplified as follows:

$$
\begin{aligned}
P_5 \leqslant\ & \mathrm{Prob}[E^{(\ \textsf{ex-uw})}(0,32)\ \wedge\ E^{(\ \textsf{ex-ub})}(1,64)]\cdot \\
& \big(65536\cdot \mathrm{Prob}[E^{(\ \textsf{at-eqw-v})}(4,v_0|w_0,31,1)]\ +\ \mathrm{Prob}[E^{(\ \textsf{at-eqw-v})}(3,v_0|w_0,31,1)]\big)\ +\ O(\Delta)
\end{aligned}
\tag{C.21}
$$

The term $\mathrm{Prob}[E^{(\ \textsf{ex-uw})}(0,32)\wedge E^{(\ \textsf{ex-ub})}(1,64)]$ is the probability of having an uncovered byte in the output of the IVP construction in any location, and no uncovered words. This term is bounded by $64\cdot(2^{-8}+O(\Delta))$, according to Corollary 2 and the fact that there are 64 byte locations. The term $\mathrm{Prob}[E^{(\ \textsf{at-eqw-v})}(4,v_0|w_0,31,1)]$ is bounded by $2^{-35.059}+O(2^{-48.626})$. The term $2^{-35.059}$ corresponds to the most dominant subevent where there is exactly one set with 4 words equal to $v_0|w_0$ that are covered. The term $O(2^{-48.626})$ corresponds to the next most dominant subevent where there is exactly one set with 5 words equal to $v_0|w_0$ that are covered. This term also includes the probabilities of other subevents where even more covered

words are equal to $v_0|w_0$. The probability term $\text{Prob}[E^{(\ \mathsf{at\text{-}eqw\text{-}v})}(3, v_0|W_0, 31, 1)]$ is similarly be bounded by $2^{-37.866} + O(2^{-51.059})$.

From these bounds and relation C.21 we get:

$$P_5 \leqslant 2^{-34.866} + O(2^{-48.626}) \tag{C.22}$$

A number of subsequent lemmas are also stated concerning cases where bytes are uncovered, the proof of which is similar.

Lemma C.5 The probability P_6 defined in equation C.3 is bounded by $2^{-35.882} + O(\Delta_6)$ where $\Delta_6 \leqslant 2^{-49.059}$.

$$P_6 \leqslant 2^{-35.882} + O(\Delta_6) \tag{C.23}$$

Lemma C.6 The probability P_7 defined in equation C.3 is bounded by $2^{-35.624} + O(\Delta_7)$ where $\Delta_7 \leqslant 2^{-48.059}$.

$$P_7 \leqslant 2^{-35.624} + O(\Delta_7) \tag{C.24}$$

Lemma C.7 The probability P_8 defined in equation C.3 is bounded by $2^{-36.308} + \Delta_8$ where $\Delta_8 \leqslant 2^{-46.544}$.

$$P_8 \leqslant 2^{-36.308} + O(\Delta_8) \tag{C.25}$$

Lemma C.8 The probability P_0 defined in equation C.3 is bounded by $2^{-32.866} + O(\Delta_0)$ where $\Delta_0 \leqslant 2^{-64}$.

$$P_0 \leqslant 2^{-32.866} + O(\Delta_0) \tag{C.26}$$

Probability P_0 is the probability of seeing the $\mathcal{F}_{\mathsf{EQ},512,16,4}$ pattern in the output of the IVP construction when all words of the output are covered. Lemma C.8 follows directly from Corollary 1.

Lemma C.9 The probability P_4 defined in equation C.3 is bounded by $2^{-48.866} + O(\Delta_4)$ where $\Delta_4 \leqslant 2^{-94.866}$.

$$P_4 \leqslant 2^{-48.866} + O(\Delta_4) \tag{C.27}$$

Proof of Lemma C.9 Probability P_4 is the probability of seeing the $\mathcal{F}_{\mathsf{EQ},512,16,4}$ pattern in the output of the IVP construction when four or more words of the output are uncovered. This is bounded by the probability of having four or more words in the IVP output uncovered, which is further bounded by $\binom{32}{4} \cdot (2^{-16} + O(2^{-64}))^4$. This bound is equal to $2^{-48.866} + O(\Delta_4)$.

Completing the proof of the correctness of Proposition 2:

We let $\mathcal{P}_1$, $\mathcal{P}_2$, $\mathcal{P}_3$, $\mathcal{P}_5$, $\mathcal{P}_6$, $\mathcal{P}_7$ and $\mathcal{P}_8$ denote the non-negligible terms of the probabilities P_1, P_2, P_3, P_5, P_6, P_7, and P_8, respectively. Summing up these non-negligible terms, we compute a term T as follows:

$$T = \sum_{i \in \{1,2,3,5,6,7,8\}} \mathcal{P}_i = 2^{-33.553} \tag{C.28}$$

This is the additive term that appears in Proposition 2. On the other hand, the observation probability $P_{\mathcal{F}_{\mathsf{EQ},512,16,4}}$, which also appears in Proposition 2, is the non-negligible term of probability P_0. Furthermore, the remaining non-negligible term $\binom{32}{4} \cdot \mathcal{P}_{UW}^4$ in Proposition 2 is the non-negligible term of probability P_4. Finally, we set $O(\Delta'') \leftarrow O(\Delta_0) + O(\Delta_1) + \ldots + O(\Delta_8)$. This concludes the proof.