Provable Security of Symmetric-key Cryptographic Schemes

Ashwin Jha

under the supervision of

Mridul Nandi

Indian Statistical Institute, Kolkata
203 B. T. Road, Kolkata 700108
India

October, 2019

A thesis submitted in partial fulfillment of the thesis requirements
for the degree of
Doctor of Philosophy
in
Computer Science
by
Ashwin Jha
to the
Applied Statistics Unit
Indian Statistical Institute, Kolkata
on
October 10, 2019

Examination Committee:

Dr. Mridul Nandi, Supervisor
Indian Statistical Institute, Kolkata

Prof. Utpal Garain, Convener, Ph.D. & D.Sc. Committee
Indian Statistical Institute, Kolkata

Prof. Abhijit Das, External Examiner
Indian Institute of Technology, Kharagpur

Dedicated to my mother.

Abstract

In this thesis, we provide quantitative and/or qualitative improvements in the provable security of several symmetric-key schemes, encompassing major information security goals, viz. data authentication, encryption, and authenticated encryption.

AUTHENTICATION AND INTEGRITY: Among authentication schemes, we analyze the CBC-MAC family and counter-based MACs (XMACC, XMACR, PCS, LightMAC etc.), referred as the XMAC family. First, we revisit the security proofs for CBC-MAC and EMAC, and identify a critical flaw in the state-of-the-art results. We revise the security proofs and obtain significantly better bounds in case of EMAC, ECBC and FCBC. Second, we study the security of CBC-MAC family, when the underlying primitive is pseudorandom function (PRF), and derive tight security bounds for EMAC, ECBC, FCBC, XCBC and TMAC. Third, we study the counter-based input encoding used in XMAC family. We present a generalized view on counter-based encoding and propose some efficient alternatives to the classical fixed length counter. Further, based on the generalized view, we identify some necessary and sufficient conditions, which result in simplified security arguments. As a side-result we also prove second preimage security for HAIFA-based hash function (using Davies-Meyer compression function) in the ideal cipher model.

ENCRYPTION: Among encryption schemes, we study the problem of constructing beyond-the-birthday bound (BBB) secure online encryption schemes using tweakable block ciphers (TBCs). First, we construct a birthday bound distinguisher for POEx, which invalidates the BBB security claims of POEx. Second, we propose a BBB secure online cipher, called XTC, and prove that it is optimally secure. As a by-product we suggest a generic distinguisher for a class of TBC-based online ciphers that encompasses both POEx and XTC. On a related topic, we study the problem of constructing BBB secure TBCs from block ciphers. Specifically, we derive a tight security bound for cascaded LRW2 under the assumption that the underlying hash functions are 3-wise almost XOR universal.

AUTHENTICATED ENCRYPTION: Finally, among authenticated encryption schemes, we study a generalization of the OCB family, called GOCB, with an aim to achieve efficient random read access. We introduce a relaxed notion of universal hash functions, called locally imperfect XOR universal (LIXU), and prove that GOCB is secure under this relaxed notion of universality. Further, we instantiate GOCB with AES round function based LIXU hash functions. These instantiations achieve significantly better random read access than OCB3.

Acknowledgements

This thesis is not only the culmination of my personal endeavor, but guidance, inspiration and assistance of many people. At this time of acknowledgement, I would like to convey my heartfelt gratitude to all of them.

First and foremost, I must acknowledge my great debt to my mentor and supervisor Mridul-da for the nurturing and mentoring he has provided me over the years. My association with Mridul-da goes back to 2014, when I first approached him to supervise my master's dissertation. Since then, he has been a constant source of encouragement, ideas and guidance. His highly intuitive yet mathematically rigorous approach to technical issues have played an important role in shaping my research outlook. I am grateful to him for collaborating with me on all the problems considered in this thesis. Mridul-da has been more than just a Ph.D. supervisor. Often, I sought his counsel on non-academic matters and he readily obliged me with his advice and support. More than anything else, I am thankful to him for his patience and trust in me—I know I have not been the most tractable of students.

During my Ph.D. tenure, I have had the pleasure to meet and interact with some of the brightest cryptographers. I am especially grateful to my co-authors, Avik-da, Bishwajit, Nilanjan-da, Avijit-da, Shay, Eik, Cuauhtemoc, Avradip-da, Minematsu-san, Sweta, Snehal, Mridul-da and Sasaki-san. I would also like to thank Bart, Iwata-san, Krzysztof, Minematsu-san and Yasuda-san for the stimulating discussions on symmetric-key provable security. Particularly, I want to thank Yasuda-san, whose insights and creative thinking have left a deep impact on me. I am indebted to Sasaki-san and Avradip-da for hosting me as a research intern at NTT Laboratories, Japan and Fujitsu Laboratories of America, USA, respectively.

My inclination towards theoretical computer science in general and cryptography in particular is largely due to two courses—Elements of Algebraic Structures by Palash-da and Probability and Stochastic Processes by Arijit-da—that I took in the first year of my master's degree at Indian Statistical Institute (ISI). The mathematical rigor and thoroughness of Palash-da's course coupled with his anecdotes related to the Enigma machine and mathematicians like Niels Henrik Abel and Évariste Galois fascinated me. Arijit-da's course was full of real-life applications of probability theory that helped me in developing a keen sense of appreciation for the subject. Later, courses such as, Cryptology by Mridul-da, Information and Coding Theory by Kishan-da, Computational Finance by Diganta-da, Quantum Computing by Guru-da, and Compiler Design by Mandar-da and Utpal-da, were equally enjoyable and inspirational.

I am grateful to ISI for providing all the facilities, especially the computing facility at CSSC, and support required for my research. A word of appreciation goes to the RFAC, Applied Statistics Division for timely evaluation and renewal of my research fellowship. Their critique and advice were essential for me to proceed in the right direction. A warm thanks to Utpal-da, the convener of Ph.D. & D.Sc. committee in computer science, for his continuous help and support.

During my long association with ISI, I have been fortunate to have some wonderful colleagues and friends. I have spent considerable amount of time with Amit, Aniruddha, Avik-da, Binanda-da, Butu-da, Diptendu, Durgesh, Jyotirmoy, Nilanjan-da, Nishant, Srimanta-da and Subhadip. I cherish our debates, discussions and banter over countless cups of *chai* (tea). I am glad to have seniors like Indranil-da, Kaushik-da, Sanjay-da, Shashank-da and Sumit-da. I had great time with colleagues and juniors like Avijit-da, Ritam-da, Sebati, Laltu, Prabal, Mostafizar, Suprita, Arghya, Anik, Bishwajit and Snehal. I sincerely apologize to all other people whom I did not include here, but made my stay in ISI wonderful.

I would like to thank Amit (Gupta), Amrit, Awanish, Gourav, Gurpreet, Kanj, and Sonu for their everlasting friendship and support.

Last, but not the least, I would like to extend my heartiest gratitude towards my family for their constant support and encouragement. I would like to acknowledge the motivation provided by my mother Bandna Jha, in pursuing my interests. She supported me at every point of my life and gave me strength and motivation when I needed it the most. My brother Raju (Himanshu Jha), has always been a source of confidence for me. Bhaiji (Santosh Kumar Jha) and Lallu mama (Rakesh Kumar) were my first teachers who instilled scientific curiosity in me. Baba (Shambhu Nath Pathak) and Nani have always showered their unconditional love and affection on me.

Ashwin Jha
Kolkata, October 2019

Update: I would like to thank the two anonymous reviewers for their appreciations, suggestions and detailed comments that have greatly helped in preparing the final version of my thesis. I would also like to thank Prof. Abhijit Das for agreeing to be the external examiner at my thesis defence seminar.

Ashwin Jha
New Delhi, June 2020

Contents

List of Figures

List of Tables

"It is not knowledge, but the act of learning, not possession but the act of getting there, which grants the greatest enjoyment."

– Carl Friedrich Gauss

Chapter 1

Introduction

This thesis is an exploration in the field of cryptography. The word cryptography comes from two greek words, *kryptós* meaning "secret" and *graphein* meaning "to write". In its most simplest form, it is the study and practice of techniques for concealing meaningful information by camouflaging it with some secret randomness. Since the dawn of civilization, humans have felt a need for secrecy in communication, be it military, diplomatic or otherwise. Indeed, the earliest known encrypted text occurred some 4000 years ago in the hieroglyphic inscriptions of ancient Egyptian civilization. In recent past, the two world wars followed by the subsequent cold war, and the Internet revolution have acted as catalysts in the rapid development of this field. In [113], Kahn gives an extensive account on the history and development of cryptography.

In circa 1948–49, Claude Shannon wrote two landmark papers. The first paper [179] laid the foundations of information theory, while the second paper, titled *Communication Theory of Secrecy Systems* [180], is the first concrete mathematical treatment of the field of cryptography. Up until mid-1970s, cryptography was quite one dimensional, concerned with just confidentiality or privacy in communication. In 1976, a seminal paper titled *New Directions in Cryptography* [60] by Diffie and Hellman identified the requirement of integrity, authenticity and non-repudiation in communication. Modern cryptography has incorporated all these elements along with the traditional need for confidentiality. Apart from identifying new security goals, Diffie and Hellman also invented a new paradigm in cryptography, the so-called *public-key cryptography* [60].

Modern cryptographic schemes are classified into two major classes: *symmetric-key* (or secret-key) and *asymmetric-key* (or public-key). In a symmetric-key scheme, a common (secret) key is shared between the two (or more) parties for secure communication. In an asymmetric-key scheme, each party holds a different (set of) key(s), but the keys are related in some precise mathematical sense. In general, symmetric-key schemes are

computationally much faster than asymmetric-key schemes, but they have an obvious prerequisite – *both parties should possess a common secret key*. This is not easy in a purely symmetric-key setting, because it requires face-to-face interaction or a trusted courier or a secure channel. In contrast, this is much easier in asymmetric-key setting, where a secure key exchange protocol can establish a common secret key. In fact, most of the modern communication protocols employ a hybrid of asymmetric and symmetric schemes, where the computationally intensive asymmetric-key scheme is used to fulfill the initial prerequisites, and the actual communication takes place using symmetric-key schemes.

In this thesis, we focus on purely symmetric-key setting, i.e., we assume that the key distribution prerequisite is satisfied.

1.1 Symmetric-key Security Goals and Primitives

Symmetric-key cryptography forms the back bone of contemporary information security needs. These schemes are employed in defense communications, banking operations, and, probably the most important of them all, network protocols such as SSH [93], TLS [129], WPA2 [181] etc. Any symmetric-key scheme is expected to guarantee either confidentiality or authenticity (and integrity), or both. We briefly discuss these goals, and the symmetric-key primitives that achieve these goals. Let us fix two parties Alice and Bob communicating over an insecure channel using a common secret key K.

1.1.1 Data Authenticity

Data authenticity demands that when Alice sends some information to Bob, then Bob should be able to verify that Alice was the sender. In symmetric-key settings, a related but different goal of data integrity is subsumed within data authenticity.

Message authentication codes or MACs are symmetric-key primitives, which ensure data integrity and authenticity. Informally, a MAC scheme $\mathfrak{M}$ is a tuple of two algorithms, the tag-generation algorithm $\mathfrak{M}^+$ and the tag-verification algorithm $\mathfrak{M}^-$. The working principle of a basic MAC scheme $\mathfrak{M}$ instantiated with secret key K is quite simple. Whenever Alice wants to send a message M to Bob, she sends (M, T), where the *tag* T is the output of tag-generation algorithm, i.e., $T = \mathfrak{M}^+(K, M)$. When Bob receives a message-tag pair (M', T'), he runs $\mathfrak{M}^-(K, M', T')$, which internally checks the equality $T' =_? \mathfrak{M}^+(K, M')$. See section 2.4 of chapter 2 for a formal definition of MAC.

There are several ways to construct MAC schemes. In this thesis, we will mainly study MAC schemes based on block ciphers. CBC-MAC family [15, 21, 34, 65, 97, 121, 189] and PMAC family [35, 171, 190, 191] are two most popular families of MACs. The CBC-MAC family is sequential whereas PMAC is parallelizable, though it requires more memory. Both CBC-MAC and PMAC are length dependent, i.e., their security reduces as the message length increases [19, 74]. Counter-based MACs, such as XMACC and XMACR [16], PCS [23], LightMAC [126], and LightMAC+[143], avoid such security degradation by using counter-based encodings.

1.1.2 Data Confidentiality or Privacy

Data confidentiality demands that when Alice sends some information to Bob, then Bob and only Bob should be able to read that information, and others should not get any knowledge about the information.

In symmetric-key setting, *encryption schemes* such as CTR and OFB [153] strive for confidentiality of data. An encryption scheme $\mathfrak{E}$ is a tuple of two algorithms, the encryption algorithm $\mathfrak{E}^+$, and the decryption algorithm $\mathfrak{E}^-$. The working principle of a basic encryption scheme $\mathfrak{E}$ instantiated with secret key K is as follows. Whenever Alice wants to send a message M to Bob, she sends C, where the *ciphertext* C is the output of encryption algorithm, i.e., $C = \mathfrak{E}^+(K, M)$. When Bob receives the ciphertext C, he runs $\mathfrak{E}^-(K, C)$, to get back the decrypted text M. Obviously, $\mathfrak{E}^+$ should be a bijective function and $\mathfrak{E}^-$ should be its inverse, otherwise Bob cannot decrypt with certainty. Further, if $\mathfrak{E}$ is secure, then no one else can decrypt the ciphertext without the knowledge of key K. Some popular examples of encryption schemes include, CTR [153], CBC encryption [153], OCB encryption [119, 171, 174], CMC [88], EME [89], HCTR [184], and HEH [176].

1.1.2.1 Online Encryption Schemes

In low-memory devices and continuous data streaming platforms with high throughput demands, it is often desirable to produce encrypted data in online fashion, i.e., the encryption of the current data block should only depend on the previous data blocks and the current data block. For instance, [71, 72] noticed that several network APIs are, in practice, stream-oriented, for example `OpenSSL EVP DecryptUpdate` interface. In such scenarios, the online property is desirable. Paraphrasing the informal definition by Rogaway and Zhang [173], an encryption scheme is said to satisfy the online property if (1) it can be realized by an algorithm that, for any input, read its input blocks one at a time in order and computes the corresponding output blocks one at a time in

order, and (2) it uses only a constant-bound amount of memory and/or latency. The introductory definition by Bellare et al. [18] satisfied (1), which was later strengthened by Boldyreva and Taesombut [39] to satisfy (2). See section 2.5 of chapter 2 for a formal definition of online encryption schemes.

Apart from their practical importance, online ciphers are also used to construct other symmetric-key primitives, most prominently they are used in authenticated encryption schemes to dilute the effect of nonce misuses [1, 3, 40, 56]. Recently, Andreeva et al. [7] used online ciphers to build deterministic encryption schemes. In the same paper they also showed that online ciphers are equivalent to tweakable block ciphers (block ciphers with an extra public input called *tweak*) with variable length tweaks.

Some popular online encryption modes include, HCBC1 and HCBC2 [18], HPCBC [39], MHCBC and MCBC [146], TC1, TC2 and TC3 [173], OleF [29], and POEx [71, 72]. In this thesis we will study online encryption schemes and their relationship with tweakable block ciphers.

1.1.3 Data Authenticity and Confidentiality

Suppose now that Alice wants to send some confidential information to Bob, and Bob wants to verify that Alice was the sender. This requires both authenticity and confidentiality.

Authenticated encryption (AE) or authenticated cipher schemes are symmetric-key primitives which achieve both confidentiality and authenticity. An authenticated encryption scheme $\mathfrak{A}$ is a tuple of two algorithms, the encryption algorithm $\mathfrak{A}^+$ and the decryption algorithm $\mathfrak{A}^-$. The working principle of any AE scheme $\mathfrak{A}$ instantiated with secret key K is similar to a MAC scheme. Whenever Alice wants to send a message M to Bob, she sends (C, T), where the *ciphertext-tag* pair (C, T) is the output of encryption algorithm, i.e., $(C, T) = \mathfrak{A}^+(K, M)$. When Bob receives a ciphertext-tag pair (C', T'), he runs $\mathfrak{A}^-(K, C', T')$, which (in most cases) internally decrypts the ciphertext to some plaintext M' and then checks the equality $\mathfrak{A}^+(K, M') =? (C', T')$. See chapter 2 section 2.6 for formal definition of AE.

Usually, AE schemes also take an auxiliary input called the *associated data* (AD) or header that may contain some information about the sender. In these cases, the modified AE scheme is called AE with associated data functionality or AEAD. A secure AEAD scheme should ensure authenticity of both AD and message, and privacy of only message (as AD is usually some publicly known information). OCB family [119, 171, 174], GCM [130], CCM [154], COLM [5] etc. are some popular AEAD schemes.

1.2 Provable Security in Symmetric-key Settings

In [13], Bellare notes that the idea of provable security was introduced by Goldwasser and Micali [79] in context of asymmetric encryption, but it soon spread to other tasks, particularly pseudorandomness [37, 76].

The most fundamental step in proving the security of any scheme is to establish the security goal (privacy or authenticity or both) and a formal adversarial model—a security game that defines the meaning of security with regard to the security goal. Given the goal and the model, a (symmetric-key) cryptographic scheme is deemed to be provably secure, if one can formally argue that it is secure given the hardness of some underlying computational problem. For example AES-PMAC [35] is a secure MAC (see chapter 2) if AES [152] is a secure pseudorandom permutation or PRP[1] (see chapter 2). Here MAC gives a precise definition of the security goal (unforgeability of tag values) and adversarial model (access to tag-generation and tag-verification algorithms as black boxes), and PRP assumption on AES is the hardness assumption.

1.2.1 Information-Theoretic Security

In general, a symmetric-key scheme is made up of two components:

- A *concrete primitive*, such as the block cipher AES [152], that operates on short and fixed length inputs; and
- A suitable *mode of operation*, such as GCM [130], that is used to extend the domain of applicability from short and fixed lengths to long and variable lengths.

For example, AES-GCM is a symmetric-key scheme that combines the AES block cipher (the concrete primitive) and GCM authenticated encryption mode (the mode of operation) to achieve both authenticity and confidentiality.

The usual method for proving the computational[2] security of a symmetric-key scheme involves two steps:

1. Replacing the underlying concrete primitive with a suitable ideal object. For example, AES is replaced with a uniform random permutation Π; This step uses the computational indistinguishability[3] of the underlying primitive with respect to

[1] Computationally indistinguishable from a uniform random permutation.
[2] Any algorithm must halt in polynomial (of input length) time.
[3] For example, a PRP is computationally indistinguishable from a uniform random permutation.

the ideal object. This is, in general, heuristic, and mostly depends upon the confidence on a particular primitive. For example, AES is considered to be a good PRP, as it is well-analyzed over a longer period of time.

2. Proving the security of the mode of operation using this ideal primitive. For example, Π-GCM (GCM instantiated with Π) is shown to be a secure AE. This second step, in most of the cases, proves information-theoretic (the adversary is allowed unbounded computational time) indistinguishability of the mode of operation. In other cases, the original security game can be reduced to some variant indistinguishability game.

In this thesis, we will be mainly concerned with the second step. The security is, in general, parameterized in terms of adversarial resources. For example, throughout this thesis we use q, ℓ, σ, and t to denote the number of queries, length of the longest query, the total length of all queries, and the computational time, respectively.

1.2.2 Directions in Provable Security

A provable security result has two main components: i) the *underlying hardness assumption/security precondition*; (ii) the *security bound under this assumption*. We use the following terminologies to refer to these facets:

- *Qualitative*: what is the underlying security precondition? This could be some number-theoretic problem like "factoring problem", or the pseudorandomness of some family of function like "AES is PRP", or the universal property of some hash function, etc.
- *Quantitative*: how much security (data and time complexity)? The quantitative value of some security bound depends on the relationship between the security parameters (i.e. q, ℓ, σ) and the input/output size of the primitive, for example the block size n and key size κ of the block cipher. This is represented by the notion of *advantage* of an adversary against the scheme (see chapter 2).

Naturally, any result on the provable security of some scheme means improvement in either its qualitative aspect, i.e. a relaxation in the security notions of underlying primitives, or its quantitative aspect, i.e. an improved security bound, or both. Consequently, all the results in this thesis improve over the existing results in quantitative and/or qualitative aspects.

1.3 Thesis Outline and Our Contributions

In this thesis, we aim to provide qualitative and/or quantitative improvements in the security of several symmetric-key schemes encompassing authentication, encryption and authenticated encryption. We start off with establishing the relevant notations and conventions in chapter 2. In the same chapter we also give a formal description of symmetric-key primitives and their security. Our contributions are given in chapter 3 onwards. Broadly our contributions can be grouped into three parts: a) analysis of message authentication codes (in chapters 3-5); b) analysis of online and tweakable ciphers (in chapters 6 and 7); and c) analysis of authenticated ciphers (in chapter 8). In each of the subsections 1.3.1-1.3.3, we first give a detailed summary of related works, followed by our contributions. Finally, subsection 1.3.4 lists down all the research papers on which this thesis is based.

1.3.1 Analysis of Message Authentication Codes

We analyze two family of MAC schemes, namely, the CBC-MAC family comprising of CBC-MAC [65], EMAC [15, 21], ECBC, FCBC and XCBC [34], TMAC [121] and OMAC [97], and the XMAC family comprising of XMACC and XMACR [16], PCS [23], and LightMAC [126].

The security of MAC constructions has seen constant research interest. Among block cipher based constructions, CBC-MAC [15], PMAC [35] and their variants [34, 97, 126] are the most popular.

ANALYSIS OF CBC-MAC: The first concrete result on CBC-MAC was given by Bellare et al. [15]. They showed a bound of $2q^2\ell^2/2^n$ for fixed length queries, which was further improved to $q^2\ell^2/2^n$ by Maurer [128]. Later, Bernstein [27] simplified the proof for fixed-length CBC-MAC. Petrank and Rackoff [162] extended the proof in [15] to prefix-free[4] queries and a similar extension on Bernstein's proof was done by Rackoff and Gorbunov [80]. Both bounds are about $O(q^2\ell^2/2^n)$. The most recent bound on CBC-MAC is due to Bellare et al. [19] who improved (in terms of ℓ) the bound to $12q^2\ell/2^n + 64q^2\ell^4/2^{2n}$. Another way of improvement is to show a bound of the form $O(\sigma q/2^n)$ (see [149] for more details).

ANALYSIS OF EMAC: In [15], Bellare et al. also suggested some variants of CBC-MAC to handle variable length messages. In particular, they mentioned a construction, where the output of CBC-MAC is further encrypted by an independently keyed block

[4]None of the message is a prefix of another message.

cipher. This construction known as EMAC was first developed during the RACE project [21]. Petrank and Rackoff [162] proved that DMAC (same as EMAC) has security bound $2.5q^2\ell^2/2^n$. Bellare et al. [19] improved the bound to $O(q^2d'(\ell)/2^n)$, where $d'(\ell) = \ell^{o(1)}$ denotes the maximum number-of-divisors function. The latest bound on EMAC is due to Pietrzak [163] that claims $O(q^2/2^n)$ bound for $\ell < 2^{n/8}$.

ANALYSIS OF OTHER MEMBERS OF THE CBC-MAC FAMILY: Both CBC-MAC and EMAC require input lengths in multiple of the underlying block size n. Black and Rogaway [34] introduced three refinements to EMAC, viz., ECBC, FCBC and XCBC to allow use of variable block length strings. They showed that ECBC and FCBC have security bound $2.5\sigma^2/2^n$ and the bound on XCBC is $3.75\sigma^2/2^n$. Jaulmes et al. [102] gave a randomized version of EMAC which they called RMAC and proved that the construction resists birthday attacks. However the proof seems to be incorrect (as suggested in [19]). Other excellent variants of CBC-MAC are TMAC [121], OMAC [97] and GCBC [147]. A variant of OMAC, namely OMAC1 is equivalent to CMAC which became an NIST recommendation [155] in 2005. Another design approach is PMAC [35] by Black and Rogaway that processes the input message in parallel. In [35], Black and Rogaway proved $O(\sigma^2/2^n)$ bound for PMAC. In [98, 140, 148, 151] the improved bounds for XCBC, TMAC, PMAC and OMAC are shown in the form of $O(q^2\ell/2^n)$, $O(\sigma^2/2^n)$ and $O(\sigma q/2^n)$. Apart from these specific constructions, Jutla [112] suggested a general class of DAG-based constructions.

ANALYSIS OF PRF-BASED CBC-MAC FAMILY: There has been little study about the security of CBC-MAC family instantiated by pseudorandom functions or PRFs[5] (see chapter 2). In fact, we came across only three prior works. First work is a study of iterated MAC constructions by Preneel and van Oorschot [167], which gives tight bounds for compressing PRFs. But, the result is not applicable to CBC-MAC family which requires just length-preserving primitives. Second one is the diploma thesis of Berke [22], where he showed that CBC-MAC restricted to prefix-free messages can be attacked with advantage $\Omega(q^2\ell^2/2^n)$. This is in fact a non trivial result which shows the gap between PRP vs PRF instantiation, because for PRP instantiation of CBC-MAC the advantage is bounded by $O(q^2\ell/2^n)$, whereas for PRF instantiation of CBC-MAC the attack shows that the PRF bound in [15] is tight. The third work is Guo et al's [86] collision distinguisher for iterated random function with advantage $\Omega(q^2\ell/2^n)$. As it turns out, this attack can be easily translated to attack against PRF-based EMAC, ECBC, FCBC, XCBC and TMAC. However, the bound holds when $q^2\ell^2 \leq 2^n$. Subject to this condition, $\Omega(q^2\ell/2^n)$ can be some constant only when $q = \Omega(2^{n/2})$. This lower bound on q is actually no better than the trivial birthday attack with advantage $\Omega(q^2/2^n)$. All of the above mentioned works concentrated on the lower bounds on the PRF advantage. In context

[5] Computationally indistinguishable from a uniform random function.

of upper bounds, we note that any PRP-based security bound can be transformed to PRF-based security bound with a loss of $O(\sigma^2/2^n)$ security using the PRP-PRF switching lemma [14, 47, 183]. However this generic transformation is known to be rather loose, and may not necessarily capture the exact security of PRF-based instantiations.

ANALYSIS OF XMAC FAMILY: All the above schemes have length dependent security bounds, i.e., the security decreases as the length increases. In [16], Bellare et al. proposed two counter-based MACs called XMACC and XMACR. Bellare et al. showed that XMACC and XMACR achieve length independent security bounds of $O(q/2^n)$ and $O(q^2/2^n)$ for large range of message length (up to two power the counter size). Bernstein utilized a similar counter-based encoding to obtain $O(q^2/2^n)$ bound for protected counter sum or PCS [23] assuming that the underlying primitive is PRF. Luykx et al. [126] gave the block cipher version of PCS, called LightMAC, and showed the security bound of $O(q^2/2^n)$. We call the group of all these counter-based schemes, the XMAC family. Although, the XMAC family achieves ℓ-free security bound, it loses on efficiency, since the encoded message is generally much longer than the original message. This is due to the encoding of the counter bits. In general, reducing the counter size as much as possible is desirable for efficiency, but it is not known how one can reduce the counter size, and whether the modified encoding will be secure or not.

1.3.1.1 Improved Security of CBC-MAC Family

In chapter 3, we identify and fix a critical flaw in the state-of-the-art results [19, 163] on the PRF security of CBC-MAC and EMAC. This results in a small change in the existing security bound for CBC-MAC, whereas the existing security bounds for EMAC degrades significantly. Consequently, we provide tight security bound for EMAC while $\ell < 2^{n/4}$. In concrete terms, for CBC-MAC we prove $O(\sigma q/2^n)$ security while $\ell < 2^{n/3}$, and for EMAC we prove $O(q^2/2^n)$ security while $\ell < 2^{n/4}$. Our bound for EMAC also implies tight security for ECBC and FCBC while $\ell < 2^{n/4}$. Table 1.3.1 summarizes the state-of-the-art results and our improvements on CBC-MAC family.

PUBLICATION HISTORY: Chapter 3 is based on our paper [104], accepted in *Journal of Mathematical Cryptology*.

1.3.1.2 Exact Security of PRF-based CBC-MAC Family

In chapter 4, we study the security of CBC-MAC family when the underlying primitive is a length-preserving PRF. We obtain a tight security bound for EMAC, ECBC, FCBC, XCBC, and TMAC. Further, our lower bound attack also applies to OMAC. In concrete

Table 1.3.1: Summary of security (PRF advantage) bounds for the CBC-MAC family. Here q, ℓ, and σ denote the number of queries, maximum permissible message length, and sum of message lengths of all q queries, respectively.

Scheme	State-of-the-art		Chapter 3	
	Bound	Restriction	Bound	Restriction
CBC-MAC [65]	$O\left(q^2\ell/2^n\right)$ [19][1]	$\ell = o\left(2^{n/3}\right)$	$O\left(\sigma q/2^n\right)$	$\ell = o\left(2^{n/3}\right)$
EMAC [15, 21]	$O\left(q^2/2^n\right)$ [163][1] $O\left(q^2 d'(\ell)/2^n\right)$ [19][1]	$\ell = o\left(2^{n/8}\right)$ $\ell = o\left(2^{n/3}\right)$	$O\left(q^2/2^n\right)$	$\ell = o\left(2^{n/4}\right)$
ECBC [34]	$O\left(q^2 d'(\ell)/2^n\right)$ [19][1]	$\ell = o\left(2^{n/3}\right)$	$O\left(q^2/2^n\right)$	$\ell = o\left(2^{n/4}\right)$
FCBC [34]	$O\left(q^2 d'(\ell)/2^n\right)$ [19][1,2]	$\ell = o\left(2^{n/3}\right)$	$O\left(q^2/2^n\right)$	$\ell = o\left(2^{n/4}\right)$
XCBC [34]	$O\left(q^2\ell/2^n\right)$ [140][3] $O\left(\sigma^2/2^n\right)$ [98][3]	$\ell = o\left(2^{n/3}\right)$ -	-	-
TMAC [121]	$O\left(q^2\ell/2^n\right)$ [140][3] $O\left(\sigma^2/2^n\right)$ [98][3]	$\ell = o\left(2^{n/3}\right)$ -	-	-
OMAC [97]	$O\left(\sigma q/2^n\right)$ [148]	$\ell = o\left(2^{n/3}\right)$	-	-

[1] Proof is flawed. See Chapter 3 for details.
[2] Although, FCBC is not mentioned in [19], the EMAC bound is applicable to FCBC as well.
[3] σ^2 and $q^2\ell$ are incomparable, as the exact relation depends on the query length distribution.

terms, for EMAC, ECBC, FCBC, XCBC, and TMAC we prove $O(\sigma q/2^n)$ security while $\ell < 2^{n/3}$, and for EMAC, ECBC, FCBC, XCBC, TMAC, and OMAC we present an adversary that breaks the security with $\Omega(q^2\ell/2^n)$ advantage. Table 1.3.2 summarizes the state-of-the-art and our results on PRF-based CBC-MAC family.

PUBLICATION HISTORY: Chapter 4 is based on our paper [109], accepted in *IACR Transactions on Symmetric Cryptology*.

1.3.1.3 Formalizing Counter-based Encoding

In chapter 5, we study the counter-based encoding used in the XMAC family (XMACC, XMACR [16], PCS [23], LightMAC [126]). We present a unified notion for counters, called the *counter function family*. Based on this generalization we identify some necessary and sufficient conditions on counters which give (possibly) simple proof of security for various counter-based cryptographic schemes, including the XMAC family. We observe that these conditions are trivially true for the classical counters. We also identify and study two variants of the classical counter which satisfy the security conditions.

Table 1.3.2: Summary of security (PRF advantage) bounds for PRF-based CBC-MAC family. Here q, ℓ, and σ denote the number of queries, maximum permissible message length, and sum of message lengths of all q queries, respectively.

Scheme	State-of-the-art		Chapter 4	
	Upper Bound	Lower Bound	Upper Bound	Lower Bound
CBC-MAC	$O\left(q^2\ell^2/2^n\right)$ [15]	$\Omega\left(q^2\ell^2/2^n\right)$ [22]	-	-
EMAC	$O\left(\sigma^2/2^n\right)$[1]	$\Omega\left(q^2\ell/2^n\right)$ [86][2]	$O\left(\sigma q/2^n\right)$	$\Omega\left(q^2\ell/2^n\right)$[3]
ECBC	$O\left(\sigma^2/2^n\right)$[1]	$\Omega\left(q^2\ell/2^n\right)$ [86][2]	$O\left(\sigma q/2^n\right)$	$\Omega\left(q^2\ell/2^n\right)$[3]
FCBC	$O\left(\sigma^2/2^n\right)$[1]	$\Omega\left(q^2\ell/2^n\right)$ [86][2]	$O\left(\sigma q/2^n\right)$	$\Omega\left(q^2\ell/2^n\right)$[3]
XCBC	$O\left(\sigma^2/2^n\right)$[1]	$\Omega\left(q^2\ell/2^n\right)$ [86][2]	$O\left(\sigma q/2^n\right)$	$\Omega\left(q^2\ell/2^n\right)$[3]
TMAC	$O\left(\sigma^2/2^n\right)$[1]	$\Omega\left(q^2\ell/2^n\right)$ [86][2]	$O\left(\sigma q/2^n\right)$	$\Omega\left(q^2\ell/2^n\right)$[3]
OMAC	$O\left(\sigma^2/2^n\right)$[1]	-	-	$\Omega\left(q^2\ell/2^n\right)$[3]

[1] Using PRP to PRF switching.

[2] The bound achieves constant probability for $q = \Omega\left(2^{n/2}\right)$, and hence no better than folklore attack.

[3] The bound holds while $\ell < 2^{n/3}$ for sufficiently large n. In fact, we can choose any $q, \ell \in \Theta(2^{n/3})$.

The first variant has message length dependent counter size, whereas the second variant, based on prefix codes, has message length independent counter size. Furthermore, these variants provide better performance for shorter messages. For instance, when the message size is 2^{19} bits, AES-LightMAC with 64-bit (classical) counter takes 1.51 cycles per byte (CPB), whereas it takes 0.81 CPB and 0.89 CPB for the first and second variant, respectively. We benchmark the software performance of these variants against the classical counter by implementing them in MACs and hash function.

PUBLICATION HISTORY: Chapter 5 is based on our paper [63], accepted in *IEEE Transactions on Computers*.

1.3.2 Analysis of Online and Tweakable Ciphers

We study a generic method of constructing beyond-the-birthday bound (BBB) secure online encryption schemes using tweakable block ciphers, and methods of constructing a tweakable block cipher using block ciphers.

BBB SECURE ONLINE CIPHERS: In [131], Mennink showed that it is (almost) impossible to get BBB secure tweakable block ciphers using block ciphers. A recent work of Andreeva et al. [7] shows that online ciphers are equivalent to arbitrary tweak length

(ATL) tweakable block ciphers (TBCs). By combining these two results one can easily see that building a BBB secure online cipher using block ciphers is also difficult. On the other hand, the proof in [7] is constructive, i.e., it might be possible to get highly secure online ciphers using ATL TBCs (such as XTX [139]). POEx by Forler et al. [71] is loosely based on this approach, and has been shown to have $O(\sigma^2/2^{n+\tau})$ security bound, where τ is the tweak size.

SECURITY OF BLOCK CIPHER-BASED TBCS: In light of [131], it would be interesting to analyze the security of existing TBCs based on block ciphers. Cascaded LRW2 or CLRW2, by Landecker et al. [123], is a popular block cipher to TBC converter. Landecker et al. [123], showed that CLRW2 has roughly $q^3/2^{2n}$ security. Recently, Mennink showed an attack on CLRW2 with advantage $q^4/2^{3n}$. In the same paper, he also showed that CLRW2 based on 4-wise independent AXU (see chapter 2) has $\gamma q/2^n$ security, where γ denotes the maximum number of queries with same tweak. However, Mennink's proof depends on the validity of a restricted version of the generalized Mirror Theory [160, 161], which has several unverified gaps [54, 58].

1.3.2.1 Online Ciphers using Tweakable Block Ciphers

In chapter 6, we study the consequences of instantiating the ATL TBC to online cipher converter of [7] with XTX by Minematsu and Iwata [139]. We first observe that XTX-based instantiations has security in the order of $\sigma^2\epsilon$, where ϵ is the pAXU bound (see chapter 2) of the underlying hash function. We show that there are genuine practical issues which render it almost impossible to get full security using this approach. We then observe that POEx by Forler et al. [71, 72] is actually an implicit example of this approach. We show a flaw in the analysis of POEx which results in a birthday bound attack and invalidates the BBB security claim. We take a slightly different approach then the one just mentioned and propose XTC which achieves security of $O(\max\{n\sigma/2^n, \sigma^2/2^{(n+\tau)}\})$ where τ is the tweak size. While doing so we present an impossibility result for $\tau > n$ which can be of independent interest.

PUBLICATION HISTORY: Chapter 6 is based on our paper [105], accepted in *Cryptography and Communications*.

1.3.2.2 Cascaded LRW2

In chapter 7, we revisit the recent result on Cascaded LRW2 or CLRW2 by Mennink [132]. In that paper, Mennink discussed some non-trivial bottlenecks in proving tight security bound, i.e. security up to $2^{3n/4}$ queries. Subsequently, he proved security up

to $2^{3n/4}$ queries for a variant of CLRW2 using 4-wise independent AXU assumption and the restriction that each tweak value occurs at most $2^{n/4}$ times. Moreover, his proof relies on a version of mirror theory which is yet to be publicly verified. We resolve the bottlenecks in Mennink's approach and prove that CLRW2 is a secure tweakable block cipher up to roughly $2^{3n/4}$ queries, with only one assumption that the underlying hash function family is 3-wise independent AXU. To do so, we develop two new tools: first, we give a probabilistic result that provides improved bound on the joint probability of some special collision events; second, we present a variant of Patarin's mirror theory in tweakable permutation settings with a self-contained and concrete proof. Both these results are of generic nature, and can be of independent interests.

PUBLICATION HISTORY: An abridged version of chapter 7 was under review in *Journal of Cryptology* at the time of submission of this thesis. Subsequently, the results of this work have been improved significantly from what has been described in chapter 7. Particularly, we have been able to drop the 3-wise independent AXU assumption and show tight security for the original CLRW2. This improved and updated result has been accepted in *Journal of Cryptology* [107].

1.3.3 Analysis of Authenticated Encryptions

We explore the scope of random read access and out-of-order decryption in OCB [119, 171, 174].

RANDOM READ ACCESS AND OUT-OF-ORDER DECRYPTION: *A block cipher based encryption mode of operation is said to have random read access feature, if it allows efficient decryption of any arbitrary encrypted block of message.* This property could be beneficial in secure storage of read only data on cloud servers. The encrypted storage helps in avoiding unauthorized access, and the authentication tag helps in maintaining the integrity of the stored data. For example, say some enterprise application stores day to day system log files on some cloud server. Here the day timestamp may act as the nonce value. Now suppose on a later day the application faces some issue which warrants some portion of a particular day's log file. In this case the system administrator should be able to repeatedly read arbitrary blocks from the decrypted log, with as little overhead as possible, by simply mentioning the relevant day timestamp and the offset in the log file.

Yet another use-case for random read access is in disk encryption where access to an arbitrary block of some disk sector is highly beneficial. A popular and standardized disk encryption scheme, called XTS by Rogaway [171], is quite similar to OCB encryption.

A related feature is *out-of-sequence decryption where the mode of operation allows for the decryption of the encrypted block data stream in an arbitrary order*. In other words, the original ordering of the ciphertext blocks is not necessary for the decryption phase. This is quite useful in secure data transmission over connection-less network protocols such as User Datagram Protocol (UDP) [165], which allows for out-of-order delivery of data packets. If the underlying encryption scheme offers efficient out-of-sequence decryption then the data packets can be decrypted in on-the-fly manner; otherwise the received packets have to be reordered first which adds to the network protocol overhead. Yet another application area is real time data streaming protocols such as Secure Real-time Transport Protocol (SRTP) [10, 111, 124] and SRTP Control Protocol (SRTCP) [178], where on-the-fly decryption of received data packets is necessary to maintain the continuity of audio or visual data. In general, a scheme with random read access feature also allows out-of-sequence decryption.

1.3.3.1 Random Read Access in OCB

In chapter 8, we first observe that the current versions of OCB are inefficient in random read access owing to the ineptness of the underlying mask (or offsets) generating function (MGF). We propose new candidates for MGF based on AES round function, which are efficient in direct computation and provide comparable performance in the usual setting. Our schemes are not the obvious choices for MGF in conventional sense as they do not have optimal almost XOR universal (AXU) bound. In existing OCB designs the MGFs are required to have 2^{-n}, i.e. optimal, AXU bound in order to upper bound the distinguishing advantage to $O(\sigma^2/2^n)$. We find this specific requirement too restrictive. We abstract the OCB design, termed as GOCB, to look into the universal notion required from the underlying MGF. We propose a relaxed notion of AXU, called locally imperfect XOR universal (LIXU) hash, which can be of independent interest. Using LIXU as the underlying MGF we recover reasonable security bounds for our schemes.

PUBLICATION HISTORY: Chapter 8 is based on our paper [110], accepted in *IEEE Transactions on Information Theory*.

1.3.4 List of Research Papers

This thesis is based on following research papers:

1. Ashwin Jha and Mridul Nandi: *Revisiting Structure Graphs: Applications to CBC-MAC and EMAC*. In J. Mathematical Cryptology, 10(3–4):157–180, 2016.

2. Ashwin Jha, Avradip Mandal and Mridul Nandi: *On the Exact Security of Message Authentication Using Pseudorandom Functions*. In IACR Trans. Symmetric Cryptology, 2017(1):427–448, 2017.

3. Avijit Dutta, Ashwin Jha and Mridul Nandi: *A New Look at Counters: Don't Run Like Marathon in a Hundred Meter Race*. In IEEE Trans. Computers, 66(11):1851–1864, 2017.

4. Ashwin Jha and Mridul Nandi: *On Rate-1 and Beyond-the-Birthday Bound Secure Online Ciphers using Tweakable Block Ciphers*. In Cryptography and Communications, 10(5):731–753, 2018.

5. Ashwin Jha and Mridul Nandi: *Tight Security of Cascaded LRW2*. In J. Cryptology, 2020.

6. Ashwin Jha, Cuauhtemoc Mancillas-López, Mridul Nandi and Sourav Sen Gupta: *On Random Read Access in OCB*. In IEEE Trans. Information Theory, 65(12):8325–8344, 2019.

Note. The ordering of the authors' name, in the publication, follows the **Hardy-Littlewood principle** i.e., it is determined by the alphabetical ordering of the last names of the authors.

Chapter 2

Preliminaries

2.1 Notational Setup

We write the set of all positive integers as $\mathbb{N}$ and $\mathbb{N}_0 := \mathbb{N} \cup \{0\}$. For $n \geq r \in \mathbb{N}_0$, we define the falling factorial $(n)_r := n!/(n-r)! = n(n-1)\cdots(n-r+1)$. For $a < b \in \mathbb{N}_0$, $[a \dots b]$ denotes the set $\{a, a+1, \dots, b\}$. For $a = 0$ and $a = 1$, we simply write it as $(b]$ and $[b]$, respectively. We sometimes use the short hand notation $\exists^*$ to represent the phrase "there exists distinct". For any finite set $\mathcal{X}$, $\mathsf{X} \leftarrow\!\!\$\, \mathcal{X}$ represents the uniform at random sampling of X from $\mathcal{X}$.

STRINGS AND TUPLES: For $n \in \mathbb{N}$, $\{0,1\}^n$ denotes the set of all n-bit strings, and $\{0,1\}^+ := \cup_{i=1}^{\infty}\{0,1\}^i$. We write ε to denote the empty string and $\{0,1\}^* := \{0,1\}^+ \cup \{\varepsilon\}$. For $x \in \{0,1\}^*$, $|x|$ denotes the *bit-length* (or simply length) of x, and $|\varepsilon| = 0$ by convention. For $x, y \in \{0,1\}^*$, $z = x\|y$ denotes the concatenation of x and y with $|z| = |x| + |y|$. Thus, by definition $x\|\varepsilon = \varepsilon\|x = x$. For $z = x\|y \in \{0,1\}^*$, x and y are called the *prefix* and *suffix*, respectively, of z. For any $r, s \in \mathbb{N}$, $\langle s \rangle_r$ denotes the r-bit unsigned representation of integer s. For any $x \in \{0,1\}^*$ and $r < |x|$, $\mathsf{msb}_r(x)$ and $\mathsf{lsb}_r(x)$ denote the substrings $x_1\| \cdots \|x_r$ and $x_{|x|-r+1}\| \cdots \|x_{|x|}$, respectively.

For a finite set $\mathcal{X} \subset \{0,1\}^*$ and $q \in \mathbb{N}$, $x^q \in \mathcal{X}^q$ denotes the q-tuple (or sequence) $(x_i)_{i\in[q]} := (x_1, x_2, \dots, x_q)$, where $x_i \in \mathcal{X}$ for all $i \in [q]$. By an abuse of notation, we also use x^q to denote the multiset $\{x_i : i \in [q]\}$ and write $\mu(x^q, x')$ to denote the multiplicity of $x' \in x^q$. We sometimes write $\widehat{x}^q$ to denote the set $\{x_i : i \in [q]\}$. We write $(\mathcal{X})_q \subset \mathcal{X}^q$ as the set of all q-tuples with distinct elements from $\mathcal{X}$, i.e., for $x^q \in (\mathcal{X})_q$, $x^q = \widehat{x}^q$ and $|(\mathcal{X})_q| = (|\mathcal{X}|)_q$. For any tuple $x^q \in \mathcal{X}^q$, and for any function $f : \mathcal{X} \to \mathcal{Y}$, $f(x^q)$ denotes the q-tuple $(f(x_1), \dots, f(x_q)) \in \mathcal{Y}^q$. For a set $\mathcal{I} \subseteq [q]$ and a q-tuple x^q, $x^{\mathcal{I}}$ denotes the

tuple $(x_i)_{i\in\mathcal{I}}$. By extending the notation, we write $x^{\mathcal{I}} = (x_\alpha)_{\alpha\in\mathcal{I}}$ for any abstract index set $\mathcal{I}$. Thus, x^q is just a shorthand notation for $x^{[q]}$.

For $n, r, s \in \mathbb{N}$, let $x^r \in (\{0,1\}^n)^r$, and $y^s \in (\{0,1\}^n)^s$. We define the longest common prefix of x^r and y^s, denoted $\mathsf{lcp}_n(x^r, y^s)$, as

$$\mathsf{lcp}_n(x^r, y^s) := \begin{cases} 0 & \text{if } x_1 \neq y_1, \\ i & \text{if } (x_1, \ldots, x_i) = (y_1, \ldots, y_i) \text{ and } x_{i+1} \neq y_{i+1}, \end{cases}$$

and the longest common suffix of x^r and y^s, denoted $\mathsf{lcs}_n(x^r, y^s)$, as

$$\mathsf{lcs}_n(x^r, y^s) := \begin{cases} 0 & \text{if } x_r \neq y_s, \\ i & \text{if } (x_{r-i+1}, \ldots, x_r) = (y_{s-i+1}, \ldots, y_s) \text{ and } x_{r-i} \neq y_{s-i}, \end{cases}$$

It is easy to see that $\mathsf{lcp}_n(x^r, y^s), \mathsf{lcs}_n(x^r, y^s) \leq \min\{r, s\}$. The subscript n is dropped whenever it is clear from the context.

COMPATIBILITY OF TUPLES: For a pair of tuples $x^q, y^q \in \mathcal{X}^q$, (x^q, y^q) denotes the 2-ary q-tuple $((x_1, y_1), \ldots, (x_q, y_q))$. An n-ary q-tuple is defined analogously.

1. A 2-ary q-tuple (x^q, y^q) is called *function compatible*, denoted $x^q \leftrightarrow y^q$, if $x_i = x_j \implies y_i = y_j$.
2. A 2-ary q-tuple (x^q, y^q) is called *permutation compatible*, denoted $x^q \leftrightsquigarrow y^q$, if $x_i = x_j \iff y_i = y_j$.
3. A 3-ary tuple (t^q, x^q, y^q) is called *tweakable permutation compatible*, denoted by $(t^q, x^q) \leftrightsquigarrow (t^q, y^q)$, if $(t_i, x_i) = (t_j, x_j) \iff (t_i, y_i) = (t_j, y_j)$.

2.2 Security Game, Adversary and Advantage

In (symmetric-key) cryptology the security of a given scheme is argued via various security games. A *security game* is nothing but an interactive game involving two parties: an *adversary* $\mathscr{A}$ that tries to break some scheme with respect to some security notion, and an oracle $\mathcal{O}$ that either represents the ideal object or the real scheme.

ADVERSARY AND ORACLE: An adversary $\mathscr{A}$ is an interactive Turing machine (or an algorithm)[1] that interacts with a given set of oracles that appear as black boxes to $\mathscr{A}$. An oracle $\mathcal{O}$ is nothing but an interface to a set of functions. For an oracle $\mathcal{O}$, $\mathscr{A}^{\mathcal{O}}$ denotes

[1] We deliberately withheld the adjective "efficient", as we will allow computationally unbounded adversaries in some cases.

the output of $\mathscr{A}$ after its interaction with $\mathcal{O}$. Based on $\mathscr{A}$'s response, the security game results in either success or failure.

DISTINGUISHING GAME: Looking ahead momentarily, in the context of this thesis, we mostly consider security game as a standard distinguishing game with an optional set of additional restrictions, chosen to reflect the desired security goal(s). For instance, some security games give bidirectional access to the underlying function of an oracle $\mathcal{O}$ and its inverse. This is denoted by $\mathcal{O}^{\pm}$. In a *distinguishing game*, the adversary or *distinguisher* $\mathscr{A}$ tries to distinguish between two sets of oracle(s), say $\mathcal{O}_0$ and $\mathcal{O}_1$. It interacts with the oracle(s) at hand (either $\mathcal{O}_0$ or $\mathcal{O}_1$) and then outputs a single bit response. We say that $\mathscr{A}$ wins the distinguishing game if $\mathscr{A}^{\mathcal{O}_b} = b$ for $b \in \{0, 1\}$. Formally, the distinguishing advantage of $\mathscr{A}$ is defined as

$$\mathbf{Adv}_{\mathcal{O}_1;\mathcal{O}_0}(\mathscr{A}) := \left|\Pr\left[\mathscr{A}^{\mathcal{O}_1} = 1\right] - \Pr\left[\mathscr{A}^{\mathcal{O}_0} = 1\right]\right| \tag{2.1}$$

CONVENTIONS: $\mathcal{O}_0$ conventionally represents an ideal primitive, while $\mathcal{O}_1$ represents either an actual construction or a mode of operation built of some other ideal primitives. Typically, the goal of the function represented by $\mathcal{O}_1$ is to emulate the ideal primitive represented by $\mathcal{O}_0$. We use the standard terms *ideal oracle* and *real oracle* for $\mathcal{O}_0$ and $\mathcal{O}_1$, respectively. When we talk of distinguishing advantage with a specific distinguishing game G in mind, we include g in the superscript, i.e., $\mathbf{Adv}^{\mathsf{g}}_{\mathcal{O}_1;\mathcal{O}_0}(\mathscr{A})$. Usually, the ideal oracle $\mathcal{O}_0$ is completely identified by G, and hence dropped from the subscript.

In this thesis, we assume that the adversary is non-trivial, i.e. it never makes a duplicate query, and it never makes a query for which the response is already known due to some previous query. Typically, any adversary's resources will be measured by a subset of the following parameters:

- q, denotes the upper bound on the number of queries made by the adversary.
- ℓ, denotes the upper bound on the length of any query.
- σ, denotes the upper bound on the aggregate total length of all queries.
- t, denotes the upper bound on the computation time allowed to the adversary.

The actual parameters would depend upon the security game. We write $\mathbb{A}(\mathsf{res})$ to denote the set of all adversaries whose resources are bounded by the tuple of parameters res.

2.2.1 The Expectation Method and Coefficient H-Technique

Let $\mathscr{A}$ be a computationally unbounded and deterministic[2] distinguisher. We denote the query-response tuple of $\mathscr{A}$'s interaction with its oracle by a transcript ω. This may also include any additional information the oracle chooses to reveal to the distinguisher at the end of the query-response phase of the game. We denote by Θ_1 (res. Θ_0) the random transcript variable when $\mathscr{A}$ interacts with $\mathcal{O}_1$ (res. $\mathcal{O}_0$). The probability of realizing a given transcript ω in the distinguishing game with an oracle $\mathcal{O}$ is known as the *interpolation probability* of ω with respect to $\mathcal{O}$. Note that, for a transcript to be realized, two things need to happen:

- The distinguisher needs to make the queries listed in the transcript;
- The oracle needs to make the corresponding responses.

Of these, the former is deterministic; the latter, probabilistic. Thus, when we talk of interpolation probability, we are only concerned with the oracle responses, with the assumption that the distinguisher's queries are consistent with the transcript. For any other distinguisher, the interpolation probability is trivially 0. A transcript ω is said to be *attainable* if $\Pr[\Theta_0 = \omega] > 0$. The expectation method (stated below) is quite useful in obtaining upper bounds on the distinguishing advantage in many cases [85, 90, 91].

Lemma 2.2.1 (Expectation Method [90])**.** *Let Ω be the set of all realizable transcripts. For some $\epsilon_{\mathsf{bad}} > 0$ and a non-negative function $\epsilon_{\mathsf{ratio}} : \Omega \to [0, \infty)$, suppose there is a set $\Omega_{\mathsf{bad}} \subseteq \Omega$ satisfying the following:*

- $\Pr[\Theta_0 \in \Omega_{\mathsf{bad}}] \leq \epsilon_{\mathsf{bad}}$;
- *For any $\omega \notin \Omega_{\mathsf{bad}}$,* $\dfrac{\Pr[\Theta_1 = \omega]}{\Pr[\Theta_0 = \omega]} \geq 1 - \epsilon_{\mathsf{ratio}}(\omega)$.

Then for any deterministic distinguisher $\mathscr{A}$ that distinguishes $\mathcal{O}_1$ from $\mathcal{O}_0$, we have the following bound on its distinguishing advantage:

$$\mathbf{Adv}_{\mathcal{O}_1;\mathcal{O}_0}(\mathscr{A}) \leq \epsilon_{\mathsf{bad}} + \mathsf{Ex}[\epsilon_{\mathsf{ratio}}(\Theta_0)],$$

where $\mathsf{Ex}[\mathsf{X}]$ *denotes the expectation of random variable* X.

Proof. A proof of the technique is given in [90], and we reproduce it here for the sake of completeness. As $\mathscr{A}$ is deterministic, its advantage is at most the statistical distance

[2] One can always replace a probabilistic adversary with a deterministic one (with at least the same advantage) given unbounded computation time.

between Θ_0 and Θ_1 (see [48, 90, 106, 144, 159]). Thus, we have,

$$\begin{aligned}
\mathbf{Adv}_{\mathcal{O}_1;\mathcal{O}_0}(\mathscr{A}) &= \left|\Pr\left[\mathscr{A}^{\mathcal{O}_1}=1\right]-\Pr\left[\mathscr{A}^{\mathcal{O}_0}=1\right]\right| \\
&\leq \sum_{\omega\in\Omega}\max\left\{0,\Pr\left[\Theta_0=\omega\right]-\Pr\left[\Theta_1=\omega\right]\right\} \\
&\leq \sum_{\omega\in\Omega_{\mathsf{bad}}}\Pr\left[\Theta_0=\omega\right]+\sum_{\omega\in\Omega\setminus\Omega_{\mathsf{bad}}}\Pr\left[\Theta_0=\omega\right]\cdot\max\left\{0,1-\frac{\Pr\left[\Theta_1=\omega\right]}{\Pr\left[\Theta_0=\omega\right]}\right\} \\
&\leq \epsilon_{\mathsf{bad}}+\mathsf{Ex}\left[\epsilon_{\mathsf{ratio}}(\Theta_0)\right].
\end{aligned}$$

□

The coefficient-H technique due to Patarin [158, 159] is a simple corollary of this result where $\epsilon_{\mathsf{ratio}}$ is a constant function.

Corollary 2.2.2 (Coefficient-H Technique [158, 159])**.** *Let Ω be the set of all realizable transcripts. For some $\epsilon_{\mathsf{bad}},\epsilon_{\mathsf{ratio}}>0$, suppose there is a set $\Omega_{\mathsf{bad}}\subseteq\Omega$ satisfying the following:*

- $\Pr\left[\Theta_0\in\Omega_{\mathsf{bad}}\right]\leq\epsilon_{\mathsf{bad}}$;
- *For any $\omega\notin\Omega_{\mathsf{bad}}$,* $\dfrac{\Pr\left[\Theta_1=\omega\right]}{\Pr\left[\Theta_0=\omega\right]}\geq 1-\epsilon_{\mathsf{ratio}}$.

Then for any deterministic distinguisher $\mathscr{A}$ that distinguishes $\mathcal{O}_1$ from $\mathcal{O}_0$, we have the following bound on its distinguishing advantage:

$$\mathbf{Adv}_{\mathcal{O}_1;\mathcal{O}_0}(\mathscr{A})\leq\epsilon_{\mathsf{bad}}+\epsilon_{\mathsf{ratio}}.$$

In context of the coefficient-H technique, we note that in an independent work [23] Bernstein rediscovered a similar result, called the *interpolation theorem*. This was later strengthened by Nandi [144] as the *strong interpolation theorem*.

2.3 (Tweakable) Block Cipher and (XOR) Universal Hash

For $\mathcal{X},\mathcal{Y}\subset\{0,1\}^*$, we write $\mathsf{Perm}(\mathcal{X})$ to denote the set of all permutations over $\mathcal{X}$, and $\mathsf{Func}(\mathcal{X},\mathcal{Y})$ to denote the set of all functions from $\mathcal{X}$ to $\mathcal{Y}$. For $\mathcal{K},\mathcal{X}\subset\{0,1\}^*$, $\mathsf{BPerm}(\mathcal{K},\mathcal{X})$ denotes the set of all families of permutations $\pi_k:=\pi(k,\cdot)\in\mathsf{Perm}(\mathcal{X})$, indexed by $k\in\mathcal{K}$. We sometime extend this notation for some $\mathcal{T}\subset\{0,1\}^*$, whereby $\mathsf{BPerm}(\mathcal{K},\mathcal{T},\mathcal{X})$ denotes the set of all families of permutations $\pi_{(k,t)}$, indexed by $(k,t)\in\mathcal{K}\times\mathcal{T}$. For $\mathcal{K},\mathcal{X},\mathcal{Y}\subset\{0,1\}^*$, $\mathsf{BFunc}(\mathcal{K},\mathcal{X},\mathcal{Y})$ denotes the set of all families of functions $\gamma_k:=\gamma(k,\cdot)\in\mathsf{Func}(\mathcal{X},\mathcal{Y})$, indexed by $k\in\mathcal{K}$. We will drop the parametrization of these sets (of functions/permutations), whenever the parameters are clear from the context.

2.3.1 (Tweakable) Block Cipher

A *block cipher* E-n/κ, with key size κ and block size n is a family of permutations $\mathsf{E} \in \mathsf{BPerm}(\{0,1\}^\kappa, \{0,1\}^n)$. For $k \in \{0,1\}^\kappa$, we denote $\mathsf{E}_k(\cdot) := \mathsf{E}(k,\cdot)$, and $\mathsf{E}_k^{-1}(\cdot) := \mathsf{E}^{-1}(k,\cdot)$. A *tweakable block cipher* $\widetilde{\mathsf{E}}$-$n/\tau/\kappa$, with key size κ, tweak size τ and block size n is a family of permutations $\widetilde{\mathsf{E}} \in \mathsf{BPerm}(\{0,1\}^\kappa, \{0,1\}^\tau, \{0,1\}^n)$. For $k \in \{0,1\}^\kappa$ and $t \in \{0,1\}^\tau$, we write $\widetilde{\mathsf{E}}_k^t(\cdot) = \widetilde{\mathsf{E}}_k(t,\cdot) := \widetilde{\mathsf{E}}(k,t,\cdot)$, and $\widetilde{\mathsf{E}}_k^{-t}(\cdot) = \widetilde{\mathsf{E}}_k^{-1}(t,\cdot) := \widetilde{\mathsf{E}}^{-1}(k,t,\cdot)$.

Symmetric-key literature is filled with a plethora of (tweakable) block cipher candidates. Some popular block ciphers include, AES-$128/128$ [52, 152], PRESENT-$64/128$ [38], Simon-$128/192$ and Speck-$96/144$ [11], GIFT-$64/128$ [8] etc. Deoxys-BC-$128/128/128$ and Kiasu-BC-$128/64/128$ [103], Skinny-$128/128/128$ and Mantis-$64/64/128$ [12] etc., are some examples of tweakable block ciphers.

Throughout the thesis, we fix $\kappa, \tau, n \in \mathbb{N}$ as the key size, tweak size and block size, respectively, of the given (tweakable) block cipher. We also write $\mathbb{B} = \{0,1\}^n$ and $\mathbb{T} = \{0,1\}^\tau$. Accordingly, the elements of $\mathbb{B}$, $\mathbb{T}$, $\{0,1\}^\kappa$ are referred as *blocks*, *tweaks*, and *keys*, respectively. For some $k \geq 0$, a tuple $x^k \in \mathbb{B}^k$ is called *block tuple (or block sequence)*. Similarly, $x^k \in \mathbb{T}^k$ is called tweak sequence. Sometimes, we also write $|x|_n = \lceil |x|/n \rceil$ to denote the *block-length* of x.

Let $\eta : \mathbb{N} \to \mathbb{N}$ be a function such that for all $r \in \mathbb{N}$, $r \overset{\eta}{\mapsto} s$, where $s \gg 2^r$. Throughout, we use $\eta(n)$ to denote a sufficiently large input length. Accordingly, we almost always write η to mean $\eta(n)$. For any finite set $\mathcal{S} \subset \{0,1\}^*$, we write $\mathcal{S}^{\eta^*}$ to denote the set $\cup_{i=0}^{\eta} \mathcal{S}^i$. The set $\mathcal{S}^{\eta^+}$ is defined analogously.

We define the padding function $\mathsf{pad} : \{0,1\}^{\eta^*} \to (\mathbb{B}^n)^{\eta^+}$ by the following mapping

$$\forall M \in \{0,1\}^{\eta^*},\ \overline{M} = \mathsf{pad}(M) := \begin{cases} M \| 10^{n|M|_n - |M| - 1} & \text{if } n \nmid |M|, \\ M & \text{otherwise.} \end{cases}$$

2.3.1.1 (Tweakable) Strong Pseudorandom Permutation

In the Strong Pseudorandom Permutation or SPRP security game, the distinguisher $\mathscr{A}$ tries to distinguish a block cipher E-n/κ instantiated with a key $\mathsf{K} \xleftarrow{\$} \{0,1\}^\kappa$ from a uniform random permutation $\Pi \xleftarrow{\$} \mathsf{Perm}(\mathbb{B})$. The *SPRP advantage* of $\mathscr{A}$ against E is defined as

$$\mathbf{Adv}_{\mathsf{E}}^{\mathsf{sprp}}(\mathscr{A}) := \mathbf{Adv}_{\mathsf{E}^{\pm};\Pi^{\pm}}(\mathscr{A}) \tag{2.2}$$

For $\epsilon \in [0,1]$, a block cipher family E is called (q,t,ϵ)-SPRP, if we have

$$\mathbf{Adv}^{\mathsf{sprp}}_{\mathsf{E}}(q,t) := \max_{\mathscr{A} \in \mathbb{A}(q,t)} \mathbf{Adv}^{\mathsf{sprp}}_{\mathsf{E}}(\mathscr{A}) \leq \epsilon.$$

Note that, the distinguisher is given bidirectional access to the oracles. When the distinguisher is restricted to forward-only queries, the security game is called PRP. $\mathbf{Adv}^{\mathsf{prp}}_{\mathsf{E}}(\mathscr{A})$ and $\mathbf{Adv}^{\mathsf{prp}}_{\mathsf{E}}(q,t,\epsilon)$ are defined analogously as above with appropriate restrictions on oracle access.

In the Tweakable SPRP or TSPRP security game, the distinguisher $\mathscr{A}$ tries to distinguish a tweakable block cipher $\widetilde{\mathsf{E}}\text{-}n/\tau/\kappa$ instantiated with a key $\mathsf{K} \leftarrow\!\!\$\, \{0,1\}^{\kappa}$ from a uniform tweakable random permutation $\widetilde{\Pi} \leftarrow\!\!\$\, \mathsf{BPerm}(\mathbb{T},\mathbb{B})$. The *TSPRP advantage* of distinguisher $\mathscr{A}$ against $\widetilde{\mathsf{E}}$ is defined as

$$\mathbf{Adv}^{\mathsf{tsprp}}_{\widetilde{\mathsf{E}}}(\mathscr{A}) := \mathbf{Adv}_{\widetilde{\mathsf{E}}^{\pm},\widetilde{\Pi}^{\pm}}(\mathscr{A}). \tag{2.3}$$

For $\epsilon \in [0,1]$, a tweakable block cipher family $\widetilde{\mathsf{E}}$ is called (q,t,ϵ)-TSPRP, if we have

$$\mathbf{Adv}^{\mathsf{tsprp}}_{\widetilde{\mathsf{E}}}(q,t) := \max_{\mathscr{A} \in \mathbb{A}(q,t)} \mathbf{Adv}^{\mathsf{tsprp}}_{\widetilde{\mathsf{E}}}(\mathscr{A}) \leq \epsilon,$$

where q and t are defined analogous to SPRP game. When the distinguisher is restricted to forward-only queries, the security game is called TPRP. $\mathbf{Adv}^{\mathsf{tprp}}_{\widetilde{\mathsf{E}}}(\mathscr{A})$ and $\mathbf{Adv}^{\mathsf{tprp}}_{\widetilde{\mathsf{E}}}(q,t,\epsilon)$ are defined analogously as above with appropriate restrictions on oracle access.

Throughout the thesis, we use Π and $\widetilde{\Pi}$ to denote uniform random permutation and uniform tweakable random permutation, respectively, over suitable sets which will be clear from the context.

2.3.2 (XOR) Universal Hash Functions

For some $k \in \mathbb{N}$, $\mathcal{K}, \mathcal{X} \subset \{0,1\}^{\eta(k)^*}$, and $\mathcal{Y} = \{0,1\}^k$, a $(\mathcal{K},\mathcal{X},\mathcal{Y})$-hash function H is a family of functions $H \in \mathsf{BFunc}(\mathcal{K},\mathcal{X},\mathcal{Y})$ defined over its *domain or message space* $\mathcal{X}$, *digest or hash space* $\mathcal{Y}$ and indexed by the *key space* $\mathcal{K}$.

There are several popular notions of universality for hash functions in literature [43, 57, 117, 132, 139, 171, 176, 185]. We discuss some of them which will be used later on in this thesis.

For the sake of simplicity, we let $\mathcal{H}$ denote the multiset $\{H_i : i \in \mathcal{K}\}$, and $\mathsf{H} = H_{\mathsf{K}}$ where $\mathsf{K} \leftarrow\!\!\$\, \mathcal{K}$. Thus, $\mathsf{H} \leftarrow\!\!\$\, \mathcal{H}$.

Definition 2.3.1 (Almost XOR universal hash function [117, 170]). A $(\mathcal{K}, \mathcal{X}, \mathcal{Y})$-hash function H is called *ϵ-almost XOR universal* or AXU for some $\epsilon \in [0, 1]$, if for any $x^2 \in (\mathcal{X})_2$ and any $\delta \in \mathcal{Y}$, the *differential probability* is at most ϵ. In other words, we have

$$\Pr_{\mathsf{H} \xleftarrow{\$} \mathcal{H}} [\mathsf{H}(x_1) \oplus \mathsf{H}(x_2) = \delta] \leq \epsilon. \tag{2.4}$$

When $\delta = 0$, the 0-differential event is equivalent to a collision in hash value. This special case is handled by universal hash functions.

Definition 2.3.2 (Almost universal hash function [43]). A $(\mathcal{K}, \mathcal{X}, \mathcal{Y})$-hash family H is called *ϵ-almost universal* or AU for some $\epsilon \in [0, 1]$, if for any $x^2 \in (\mathcal{X})_2$, the *collision probability* is at most ϵ. In other words, we have

$$\Pr_{\mathsf{H} \xleftarrow{\$} \mathcal{H}} [\mathsf{H}(x_1) = \mathsf{H}(x_2)] \leq \epsilon. \tag{2.5}$$

The notion of AXU hash functions was first introduced by Krawczyk [117] and later by Rogaway [170], as a generalization of the original universal hash definition by Carter and Wegman [43, 185]. Multi-linear hash [87, 177], PDP hash [36, 87, 118, 188], CBC [62] and PHash [35] are some popular examples of A(X)U hash functions.

In [132], Mennink gave a slightly generalized notion of AXU, called ℓ-wise independent AXU or AXU$_\ell$, for $\ell \geq 2$.

Definition 2.3.3 (ℓ-wise Independent AXU hash function [132]). A $(\mathcal{K}, \mathcal{X}, \mathcal{Y})$-hash family H is called ϵ-AXU$_\ell$ for some $\epsilon \in [0, 1]$, if for any $j \in [\ell] \setminus \{1\}$, any $x^j \in (\mathcal{X})_j$, and any $\delta^{j-1} \in \mathcal{Y}^{j-1}$ we have

$$\Pr_{\mathsf{H} \xleftarrow{\$} \mathcal{H}} [\mathsf{H}(x_1) \oplus \mathsf{H}(x_2) = \delta_1, \ldots, \mathsf{H}(x_1) \oplus \mathsf{H}(x_j) = \delta_{j-1}] \leq \epsilon^{j-1}. \tag{2.6}$$

In [132], Mennink suggested some ways to construct AXU$_\ell$ hash function family over a small domain using finite field arithmetic. Note that, ℓ-wise independent hash functions [185] are also AXU$_\ell$.

Minematsu and Iwata [139] introduced the notion of ϵ-partial-almost-XOR-universal or pAXU hash functions to capture simultaneous collision and differential events on two disjoint substrings of any hash value.

Definition 2.3.4 (ϵ-partial-almost-XOR-universal hash function [139]). A $(\mathcal{K}, \mathcal{X}, \mathcal{Y}_1 \times \mathcal{Y}_2)$-hash function H is called *ϵ-partial-almost-XOR-universal* (or pAXU) if for all $x^2 \in (\mathcal{X})_2$ and $\delta \in \mathcal{Y}_2$, we have

$$\Pr_{\mathsf{H} \xleftarrow{\$} \mathcal{H}} \left[\mathsf{H}(x) \oplus \mathsf{H}(x') = (0, \delta)\right] \leq \epsilon. \tag{2.7}$$

2.3.2.1 Combiners of Hash Functions

We consider two types of combiners of A(X)U hash functions.

- *Concatenated Hash* — Let H_i be a $(\mathcal{K}_i, \mathcal{X}, \mathcal{Y}_i)$ hash family for $i \in [2]$. The concatenated hash $H_1 \| H_2$ is a $(\mathcal{K}_1 \times \mathcal{K}_2, \mathcal{X}, \mathcal{Y}_1 \times \mathcal{Y}_2)$ hash family defined by the following mapping:

$$(K_1, K_2, x) \mapsto H_1(K_1, x) \| H_2(K_2, x).$$

- *Sum Hash* — Let H_i be a $(\mathcal{K}_i, \mathcal{X}_i, \mathcal{Y})$ hash family for $i \in [2]$. The sum hash $H_1 \oplus H_2$ is a $(\mathcal{K}_1 \times \mathcal{K}_2, \mathcal{X}_1 \times \mathcal{X}_2, \mathcal{Y})$ hash family defined by the following mapping:

$$((K_1, K_2), (x_1, x_2)) \mapsto H_1(K_1, x_1) \oplus H_2(K_2, x_2).$$

Some easily verifiable properties of (p)A(X)U hash functions are accumulated in proposition 2.3.5.

Proposition 2.3.5. *Let H_i be a $(\mathcal{K}_i, \mathcal{X}_i, \mathcal{Y}_i)$ hash family for $i \in [2]$, and H_1 and H_2 are keyed independently. Then, we have*

1. *If H_1 is an ϵ-AXU hash, then it is an ϵ-AU hash.*
2. *For $\mathcal{X}_1 = \mathcal{X}_2$, consider the concatenated hash $H_1 \| H_2$.*
   (a) *If H_i is ϵ_i-AXU hash for $i \in [2]$, then $H_1 \| H_2$ is an $\epsilon_1\epsilon_2$-AXU hash.*
   (b) *If H_1 is ϵ_1-AU hash and H_2 is ϵ_2-AXU hash, then $H_1 \| H_2$ is an $\epsilon_1\epsilon_2$-pAXU hash.*
3. *For $\mathcal{Y}_1 = \mathcal{Y}_2$, consider the sum hash $H_1 \oplus H_2$. If H_i is ϵ_i-AXU hash for $i \in [2]$, then $H_1 \oplus H_2$ is $\max\{\epsilon_1, \epsilon_2\}$-AXU hash.*
4. *For $k \leq m \in \mathbb{N}$ and $\mathcal{Y}_1 = \{0,1\}^m$, if H_1 is an ϵ_1-AXU, then the truncated hash $\mathsf{msb}_k \circ H_1$ is an $\epsilon_1 2^{m-k}$-AU hash function.*

2.4 Message Authentication Codes

A *message authentication code* or MAC is a tuple of algorithms $\mathfrak{M} = (\mathfrak{M}^+, \mathfrak{M}^-)$, defined over the key space $\mathcal{K}$, nonce space $\mathcal{N}$, message space $\mathcal{M}$, and tag space $\mathcal{T}$, where:

$$\mathfrak{M}^+ : \mathcal{K} \times \mathcal{N} \times \mathcal{M} \to \mathcal{T} \qquad \mathfrak{M}^- : \mathcal{K} \times \mathcal{N} \times \mathcal{M} \times \mathcal{T} \to \{\top, \bot\}.$$

Here $\mathfrak{M}^+$ and $\mathfrak{M}^-$ are called the tag-generation and tag-verification algorithms, respectively, and $\top$ and $\perp$ denote the authentication success and failure symbols, respectively. Further, it is required that $\mathfrak{M}^-(K, N, \mathfrak{M}^+(K, N, M)) = \top$ for any $(K, N, M) \in \mathcal{K} \times \mathcal{N} \times \mathcal{M}$. For all key $K \in \mathcal{K}$, we write $\mathfrak{M}_K^+(\cdot)$ and $\mathfrak{M}_K^-(\cdot)$ to denote $\mathfrak{M}^+(K, \cdot)$ and $\mathfrak{M}^-(K, \cdot)$, respectively. In this thesis, we have $\mathcal{K}, \mathcal{T} \subseteq \mathbb{B}^{\eta^+}$, $\mathcal{N} \in \mathbb{B}^{\eta^*}$ and $\mathcal{M} \in \{0,1\}^{\eta^*}$.

MACs can be classified into two classes based on the cardinality of $\mathcal{N}$:

1. **Deterministic MACs:** When the nonce space is empty, i.e., $\mathcal{N} = \emptyset$, the MAC scheme is classified as a deterministic MAC. Some popular deterministic MACs include, CBC-MAC [65], EMAC [15, 21], ECBC, FCBC and XCBC [34], TMAC [121], OMAC [97, 155], GCBC1 and GCBC2 [147], PCS [23], PMAC [35], LightMAC [126], SUM-ECBC [189], PMAC+[190], LightMAC+[143] etc.

2. **Non-deterministic MACs:** When the nonce space is non-empty, i.e., $\mathcal{N} \neq \emptyset$, the MAC scheme is classified as a non-deterministic MAC. Now, depending upon the nature of the nonce value in tag-generation algorithm, we may have two sub-classifications:

   (a) **Stateful MACs:** In stateful MACs, each invocation of the tag-generation algorithm requires a distinct nonce value, i.e., each nonce value is used only once in tag-generation calls. In other words, the tag-generation algorithm is stateful. The uniqueness of nonce value is, in general, necessary for security, though there are some schemes which allow some amount of nonce repetitions. Some popular nonce-based MACs include, Wegman-Carter MACs [185], XMACC [16], WMAC [33], EWCDM [49], DWCDM [58], nEHtM [64] etc.

   (b) **Probabilistic MACs:** In probabilistic MACs, the nonce value is treated as random *salt* in the tag-generation algorithm, i.e., the nonce value is sampled uniformly at random in each tag-generation call. In other words, the tag-generation algorithm is probabilistic. Some popular probabilistic MACs include, XMACR [16], MACRX [17], RMAC [102], FRMAC [101], EHtM [136], RWMAC [136] etc.

In this thesis, we will mainly focus on deterministic MACs.

2.4.1 MAC Security Definition

In MAC security game, a non-trivial adversary $\mathscr{A}$ has oracle access to $\mathfrak{M}_{\mathsf{K}}^+$ and $\mathfrak{M}_{\mathsf{K}}^-$ for $\mathsf{K} \xleftarrow{\$} \mathcal{K}$. It can make "MAC" queries to $\mathfrak{M}_{\mathsf{K}}^+$ and "verification" queries to $\mathfrak{M}_{\mathsf{K}}^-$ in arbitrary order. We say that $\mathscr{A}$ forges if any of its verification queries to $\mathfrak{M}_{\mathsf{K}}^-$ returns $\top$.

Suppose $\mathscr{A}$ halts with response 1 if it was successful in forgery, and 0 otherwise. The *MAC advantage* of $\mathscr{A}$ against $\mathfrak{M}$ is defined as

$$\mathbf{Adv}^{\mathsf{mac}}_{\mathfrak{M}}(\mathscr{A}) := \Pr_{\mathsf{K} \leftarrow\$ \mathcal{K}}\left[\mathscr{A}^{(\mathfrak{M}^{+},\mathfrak{M}^{-})} = 1\right]. \tag{2.8}$$

Note that, the non-triviality of $\mathscr{A}$ means that $\mathscr{A}$ is not allowed to ask a verification query (N, M, T) if it has previously made a MAC query (N, M). Further, for non-deterministic MACs the adversary abides by the following restrictions:

- For stateful MACs, the adversary cannot repeat the nonce value in any MAC query, although it is allowed to repeat nonces in verification queries.
- For probabilistic MACs, the salt value is chosen uniformly at random in all MAC queries. In verification queries the adversary is allowed to pick the salt value freely.

In MAC game, we use the following notations to parametrize adversary's resources: q_m and q_v denote the number of queries to $\mathfrak{M}^{+}_{\mathsf{K}}$ and $\mathfrak{M}^{-}_{\mathsf{K}}$, respectively; ℓ_m and ℓ_v denote the maximum permissible message length in tag-generation and verification queries, respectively; σ_m and σ_v denote the maximum permissible aggregate total length across all tag-generation and verification queries, respectively. We also write $q = q_m + q_v$, $\ell = \ell_m + \ell_v$, and $\sigma = \sigma_m + \sigma_v$ to denote the combined query resources. For $\epsilon \in [0, 1]$, $\mathfrak{M}$ is called a $(q, \ell, \sigma, t, \epsilon)$-MAC if and only if,

$$\mathbf{Adv}^{\mathsf{mac}}_{\mathfrak{M}}(q, \ell, \sigma, t) := \max_{\mathscr{A} \in \mathbb{A}(q,\ell,\sigma,t)} \mathbf{Adv}^{\mathsf{mac}}_{\mathfrak{M}}(\mathscr{A}) \leq \epsilon.$$

2.4.2 Some MAC Design Paradigms

We explain two popular MAC design paradigms which will be frequently used in this thesis. First, we discuss an important symmetric-key primitive called pseudorandom function or PRF [76] which are quite useful in constructing MAC schemes.

2.4.2.1 Pseudorandom Function

A keyed function F is a family of functions $\mathsf{F} \in \mathsf{BFunc}(\mathcal{K}, \mathcal{X}, \mathcal{Y})$. In the Pseudorandom Function or PRF security game, the distinguisher $\mathscr{A}$ tries to distinguish a keyed function F instantiated with a key $\mathsf{K} \leftarrow\$ \mathcal{K}$ from a uniform random function $\Gamma \leftarrow\$ \mathsf{Func}(\mathcal{X}, \mathcal{Y})$. The *PRF advantage* of $\mathscr{A}$ against F is defined as

$$\mathbf{Adv}^{\mathsf{prf}}_{\mathsf{F}}(\mathscr{A}) := \mathbf{Adv}_{\mathsf{F}_{\mathsf{K}};\Gamma}(\mathscr{A}). \tag{2.9}$$

For $\epsilon \in [0,1]$, F is called a (q,t,ϵ)-PRF if we have

$$\mathbf{Adv}^{\mathsf{prf}}_{\mathsf{F}}(q,t) := \max_{\mathscr{A} \in \mathbb{A}(q,t)} \mathbf{Adv}^{\mathsf{prf}}_{\mathsf{F}}(\mathscr{A}) \leq \epsilon.$$

Throughout the thesis, we use Γ to denote a uniform random function over suitable sets which will be clear from the context.

The following proposition is a well-known result [15, 76, 77] which shows that the PRF security of the tag-generation algorithm of any deterministic MAC scheme is sufficient for MAC security of that scheme. In other words, any PRF construction can be viewed as a secure deterministic MAC scheme. Indeed, most of the provable security results on deterministic MAC schemes show PRF security, which implies MAC security.

Proposition 2.4.1 (PRF $\implies$ MAC)**.** *For any deterministic MAC scheme $\mathfrak{M}$, if $\mathfrak{M}^+$ is a $(q_m, \ell_m, \sigma_m, t, \epsilon)$-PRF then $\mathfrak{M}$ is $(q, \ell, \sigma, t, \epsilon + q_v/|\mathcal{T}|)$-MAC.*

We skip the proof of this proposition as it is rather straightforward and available in [15].

2.4.2.2 Hash-then-PRF

For finite sets $\mathcal{K}_1, \mathcal{K}_2, \mathcal{M}, \mathcal{X}, \mathcal{Y} \subset \{0,1\}^{\eta^*}$, let $H \in \mathsf{BFunc}(\mathcal{K}_1, \mathcal{M}, \mathcal{X})$ be an ϵ-AU hash function, and $\mathsf{F} \in \mathsf{BFunc}(\mathcal{K}_2, \mathcal{X}, \mathcal{Y})$ be a keyed function. The composition function $\mathsf{F} \circ H$ is known as *Hash-then-PRF* construction. It is probably the easiest and most popular way of constructing a PRF over variable-length inputs. Many PRF constructions, including, EMAC [15, 21], ECBC and FCBC [34], PMAC [35], and LightMAC [126], follow this paradigm. The following result due to Shoup [183] quantifies the PRF security of $\mathsf{F} \circ H$.

Proposition 2.4.2 (Hash-then-PRF [183])**.** *Let H and F be defined as above. Then, we have*

$$\mathbf{Adv}^{\mathsf{prf}}_{\mathsf{F}\circ H}(q,\ell,\sigma,t) \leq \mathbf{Adv}^{\mathsf{prf}}_{\mathsf{F}}(q,t') + \binom{q}{2}\epsilon, \tag{2.10}$$

where $t' = t + qO(T_H)$ and T_H denotes the time complexity of computing H.

A proof of this proposition is available in many places, including [183] and [106]. In real-life applications for $\mathcal{Y} = \mathcal{X} = \mathbb{B}$, we often use a PRP E (some popular block cipher E-n/κ) in place of F. Accordingly, we call the composition "Hash-then-PRP". The following equation on the PRF security of Hash-then-PRP is a direct consequence of Proposition 2.4.2 and the PRP-PRF switching lemma [14, 47, 183].

Proposition 2.4.3 (Hash-then-PRP)**.** *Let H and E be defined as above. Then, we have*

$$\mathbf{Adv}^{\mathsf{prf}}_{\mathsf{E}\circ H}(q,\ell,\sigma,t) \leq \mathbf{Adv}^{\mathsf{prp}}_{\mathsf{E}}(q,t') + \binom{q}{2}\epsilon + \binom{q}{2}\frac{1}{2^n}, \tag{2.11}$$

where $t' = t + qO(T_H)$ and T_H denotes the time complexity of computing H.

Since PRF implies deterministic MAC, Hash-then-PRF and Hash-then-PRP are natural paradigms for constructing deterministic MACs.

2.4.2.3 Hash-then-Mask

For finite sets $\mathcal{K}_1, \mathcal{K}_2, \mathcal{M}, \mathcal{N} \subset \{0,1\}^{\eta^*}$, let $H \in \mathsf{BFunc}(\mathcal{K}_1, \mathcal{M}, \mathbb{B})$ be an ϵ-AXU hash function, and $\mathsf{F} \in \mathsf{BFunc}(\mathcal{K}_2, \mathcal{N}, \mathbb{B})$ be a keyed function. The function $\mathsf{F} \oplus H$, defined by the mapping

$$(K, L, N, M) \mapsto \mathsf{F}_K(N) \oplus H_L(M)$$

for all $(K, L, N, M) \in \mathcal{K}_1 \times \mathcal{K}_2 \times \mathcal{N} \times \mathcal{M}$, is known as the *Hash-then-Mask* construction. Here $\mathcal{N}$ and $\mathcal{M}$ denote the nonce/salt and message space, respectively. Many non-deterministic MACs, including Wegman-Carter MAC [185], XMACC and XMACR [16], follow this paradigm. Hash-then-Mask was proposed by Wegman and Carter [185], whence it is often referred as the Wegman-Carter authenticator. The following result due to Wegman and Carter [185] quantifies the MAC security of $\mathsf{F} \oplus H$.

Proposition 2.4.4 (Wegman-Carter [185])**.** *Let H and F be defined as above. Then,*

1. *for stateful or nonce-based MACs, we have*

$$\mathbf{Adv}^{\mathsf{mac}}_{\mathsf{F}\oplus H}(q,\ell,\sigma,t) \leq \mathbf{Adv}^{\mathsf{prf}}_{\mathsf{F}}(q,t') + q_v\epsilon, \tag{2.12}$$

2. *for probabilistic MACs, we have*

$$\mathbf{Adv}^{\mathsf{mac}}_{\mathsf{F}\oplus H}(q,\ell,\sigma,t) \leq \mathbf{Adv}^{\mathsf{prf}}_{\mathsf{F}}(q,t') + \binom{q_m}{2}\frac{1}{2^n} + q_v\epsilon, \tag{2.13}$$

where $t' = t + qO(T_H)$ and T_H denotes the time complexity of computing H.

A proof of this proposition is available in multiple places, including [185], [25] and [49]. Shoup [182] gave a variant of Hash-then-Mask, in which the mask is computed using a PRP $\mathsf{E} \in \mathsf{BPerm}(\mathcal{K}_2, \mathcal{N})$ for $\mathcal{N} = \mathbb{B}$, instead of a PRF. We call the resultant scheme "Wegman-Carter-Shoup" authenticator. The poly1305 MAC by Bernstein [24] is directly based on Wegman-Carter-Shoup. The following proposition, due to Bernstein [25], quantifies the MAC security of $\mathsf{E} \oplus H$.

Proposition 2.4.5 (Wegman-Carter-Shoup)**.** *Let H and* E *be defined as above. Then,*

1. *for stateful or nonce-based MACs, we have*

$$\mathbf{Adv}^{\mathsf{mac}}_{\mathsf{E}\oplus H}(q,\ell,\sigma,t) \leq \mathbf{Adv}^{\mathsf{prp}}_{\mathsf{E}}(q,t') + q_v\epsilon\left(1-\frac{q_m}{2^n}\right)^{-\frac{q_m+1}{2}}, \tag{2.14}$$

2. *for probabilistic MACs, we have*

$$\mathbf{Adv}^{\mathsf{mac}}_{\mathsf{E}\oplus H}(q,\ell,\sigma,t) \leq \mathbf{Adv}^{\mathsf{prp}}_{\mathsf{E}}(q,t') + q_v\epsilon\left(1-\frac{q_m}{2^n}\right)^{-\frac{q_m+1}{2}} + \binom{q_m}{2}\frac{1}{2^n}, \tag{2.15}$$

where $t' = t + qO(T_H)$ and T_H denotes the time complexity of computing H.

We remark that Bernstein [25] only proved the first part of Proposition 2.4.5. The second part can be easily derived from the first one by including an additional term to account for the probability of salt collision among any pair of distinct tag-generation queries.

2.5 Online Encryption Schemes

An online encryption scheme or *online cipher*, is a tuple of algorithms $\mathfrak{O} = (\mathfrak{O}^+, \mathfrak{O}^-)$, defined over the key space $\mathcal{K}$ and message space $\mathbb{B}^{\eta^*}$, where:

$$\mathfrak{O}^+ : \mathcal{K} \times \mathbb{B}^{\eta^*} \to \mathbb{B}^{\eta^*} \qquad \mathfrak{O}^- : \mathcal{K} \times \mathbb{B}^{\eta^*} \to \mathbb{B}^{\eta^*},$$

and $\mathfrak{O}^+_K(\cdot) = \mathfrak{O}^+(K,\cdot)$ is a length-preserving permutation over $\mathbb{B}^{\eta^*}$ satisfying the *online property*: $x \in \mathbb{B}^{\eta^*}$ is a prefix of $y \in \mathbb{B}^{\eta^*}$ if and only if $\mathfrak{O}^+_K(x)$ is a prefix of $\mathfrak{O}^+_K(y)$. $\mathfrak{O}^-_K$ is defined as the inverse of $\mathfrak{O}^+_K$, i.e., for all $x \in \mathbb{B}^{\eta^*}$, $\mathfrak{O}^-_K(\mathfrak{O}^+_K(x)) = x$.

In their foundational paper Bellare et al. [18] proposed CBC like constructions with online property, viz. HCBC1 and HCBC2, both of which employed one call to block cipher and one call to an almost XOR universal (AXU) hash function. HPCBC, proposed by Boldyreva and Taesombut [39], was a variant of HCBC2 that prepends the encryption of a random IV in order to fit a stronger notion. Nandi [145, 146] proposed simpler proofs for HCBC1 and HCBC2, and gave two improved schemes called MHCBC and MCBC. Among these MCBC has the feature of replacing the call to a hash function by a second call to the block cipher.

In [173] Rogaway and Zhang proposed three schemes, namely TC1, TC2, and TC3, based on tweakable block ciphers. These schemes exploited the tweak input of TBCs to eliminate the additional calls to hash function/block cipher. In an independent work

Fleischmann et al. [70] presented a scheme called MCOE-G which is similar to TC3, albeit with a more practical handling of arbitrary length inputs.

For a variant of online property called diblock-online, Bhaumik and Nandi [29] gave an inverse-free construction called OleF, that achieved *diblock-online SPRP* security. Apart from these, several online ciphers were also proposed within full fledged authenticated encryption schemes [1, 3, 4, 40, 56].

Recently, Forler et al. [71] proposed an online cipher POEx, based on a tweakable block cipher and pAXU hash function, that was claimed to have beyond-the-birthday bound[3] (BBB) security. Later, this claim was refuted (see chapter 6). Consequently, in a journal version of the same work Forler et al. proposed a modified definition [72] for POEx that provides BBB security under a strong assumption on the underlying hash function.

2.5.1 OSPRP Security Definition

We let $\mathsf{Operm}(\mathbb{B}^{\eta^*})$ to be the set of all online permutations over $\mathbb{B}^{\eta^*}$, and $\vec{\Pi} \leftarrow_{\$} \mathsf{Operm}(\mathbb{B}^{\eta^*})$. In the Online Strong Pseudorandom Permutation or OSPRP security game, the distinguisher $\mathscr{A}$ tries to distinguish an online cipher $\mathfrak{O}$ instantiated with a key $\mathsf{K} \leftarrow_{\$} \mathcal{K}$ from an online uniform random permutation $\vec{\Pi}$. The *OSPRP advantage* of $\mathscr{A}$ against $\mathfrak{O}$ is defined as

$$\mathbf{Adv}^{\mathsf{osprp}}_{\mathfrak{O}}(\mathscr{A}) := \mathbf{Adv}_{\mathfrak{O}^{\pm};\vec{\Pi}^{\pm}}(\mathscr{A}). \tag{2.16}$$

For $\epsilon \in [0,1]$, $\mathfrak{O}$ is called a $(q,\ell,\sigma,t,\epsilon)$-OSPRP if we have

$$\mathbf{Adv}^{\mathsf{osprp}}_{\mathfrak{O}}(q,\ell,\sigma,t) := \max_{\mathscr{A}\in\mathbb{A}(q,\ell,\sigma,t)} \mathbf{Adv}^{\mathsf{osprp}}_{\mathfrak{O}}(\mathscr{A}) \leq \epsilon.$$

Throughout the thesis, we use $\vec{\Pi}$ to denote a uniform online random permutation over some suitable set(s) which will be clear from the context.

2.6 Authenticated Encryption with Associated Data

An *authenticated encryption scheme with associated data* functionality, or AEAD in short, is a tuple of algorithms $\mathfrak{A} = (\mathfrak{A}^+, \mathfrak{A}^-)$, defined over the *key space* $\mathcal{K}$, *nonce space* $\mathcal{N}$, *associated data space* $\mathcal{A}$, *message space* $\mathcal{M}$, *ciphertext space* $\mathcal{C}$, and *tag space* $\mathcal{T}$, where:

$$\mathfrak{A}^+ : \mathcal{K}\times\mathcal{N}\times\mathcal{A}\times\mathcal{M} \to \mathcal{C}\times\mathcal{T} \quad \text{and} \quad \mathfrak{A}^- : \mathcal{K}\times\mathcal{N}\times\mathcal{A}\times\mathcal{C}\times\mathcal{T} \to \mathcal{M}\cup\{\perp\}.$$

[3] Secure up to 2^{xn} queries for some $x \in (0.5, 1]$.

Here, $\mathfrak{A}^+$ and $\mathfrak{A}^-$ are called the encryption and decryption algorithms, respectively, of $\mathfrak{A}$, and $\perp$ denotes the error symbol used to indicate authentication failure. Further, it is required that $\mathfrak{A}^-(K, N, A, \mathfrak{A}^+(K, N, A, M)) = M$ for any $(K, N, A, M) \in \mathcal{K} \times \mathcal{N} \times \mathcal{A} \times \mathcal{M}$. For all key $K \in \mathcal{K}$, we write $\mathfrak{A}_K^+(\cdot)$ and $\mathfrak{A}_K^-(\cdot)$ to denote $\mathfrak{A}^+(K, \cdot)$ and $\mathfrak{A}^-(K, \cdot)$, respectively.

In this thesis, we have $\mathcal{K}, \mathcal{T} \subseteq \mathbb{B}^{\eta^+}$, $\mathcal{N}, \mathcal{A}, \mathcal{M} \in \{0,1\}^{\eta^*}$ and $\mathcal{C} = \mathcal{M}$, so we use $\mathcal{M}$ instead of $\mathcal{C}$ wherever necessary. In addition, in most popular authenticated encryption schemes (including the OCB family), the map $\mathsf{proj}_{\mathcal{M}} \circ \mathsf{Enc}(K, N, A, \cdot)$ for fixed K, N, and A is a length-preserving permutation, where $\mathsf{proj}_{\mathcal{M}} : \mathcal{M} \times \mathcal{T} \to \mathcal{M}$ is the projection on $\mathcal{M}$, and $\mathcal{T} = \{0,1\}^{kn}$ for some fixed $k \in \mathbb{N}$.

AEAD schemes can be classified based on the cardinality of $\mathcal{N}$, much in the same vein as MAC schemes. There are some good examples of deterministic AEAD ($\mathcal{N} = \emptyset$) schemes based on the SIV paradigm by Rogaway and Shrimpton [172]. GCM-SIV [84], GCM-SIV2 and its generalized variant GCM-SIVr [99], and the lightweight scheme SUNDAE [9] are some of the notable SIV-based deterministic AEAD schemes. A major issue with SIV-based constructions is efficiency. These schemes are inherently two-pass, i.e., the message input is read twice, which makes them slow (and somewhat less popular) as compared to single-pass nonce-based ($\mathcal{N} \neq \emptyset$) AEAD schemes. Some of the popular nonce-based AEAD schemes include CCM [186], GCM [130], OCB3 [119], COLM [5], Ascon [61], Beetle [44] etc.

In this thesis, we will concentrate on nonce-based AEAD. Particularly, we focus on the provable security of OCB family [119, 171, 174, 174].

2.6.1 NAEAD Security Definition

In the nonce-respecting AEAD or NAEAD security game, the adversary $\mathscr{A}$ tries to distinguish between $(\mathfrak{A}_{\mathsf{K}}^+, \mathfrak{A}_{\mathsf{K}}^-)$ and $(\Gamma, \perp)$, where

$$\Gamma \xleftarrow{\$} \{f \in \mathsf{Func}[\mathcal{N}\times\mathcal{A}\times\mathcal{M}, \mathcal{M}\times\mathcal{T}] \,:\, \forall (N, A, M) \in \mathcal{N}\times\mathcal{A}\times\mathcal{M},\ |f(N, A, M)| = |M| + kn\},$$

and $\perp : \mathcal{N} \times \mathcal{A} \times \mathcal{M} \times \mathcal{T} \to \{\perp\}$, respectively. The distinguishing adversary operates under the restriction — no two encryption queries can have the same nonce. The *NAEAD advantage* of any adversary $\mathscr{A}$ against $\mathfrak{A}$ is defined as

$$\mathbf{Adv}_{\mathfrak{A}}^{\mathsf{naead}}(\mathscr{A}) := \mathbf{Adv}_{\mathfrak{A}^{\pm};(\Gamma,\perp)}(\mathscr{A}). \tag{2.17}$$

In NAEAD game, we use the following notations to parametrize adversary's resources: q_e and q_d denote the number of queries to $\mathfrak{A}_{\mathsf{K}}^+$ and $\mathfrak{A}_{\mathsf{K}}^-$, respectively; ℓ_e and ℓ_d denote

the maximum permissible input (associated data and message/ciphertext) lengths in encryption and decryption queries, respectively; σ_e and σ_d denote the sum of input lengths across all encryption and decryption queries, respectively. We also write $q = q_e + q_d$, $\ell = \ell_e + \ell_d$ and $\sigma = \sigma_e + \sigma_d$ to denote the combined query resources. For $\epsilon \in [0, 1]$, $\mathfrak{A}$ is called a $(q, \ell, \sigma, t, \epsilon)$-NAEAD if and only if,

$$\mathbf{Adv}^{\mathsf{naead}}_{\mathfrak{A}}(q, \ell, \sigma, t) := \max_{\mathscr{A} \in \mathbb{A}(q,\ell,\sigma,t)} \mathbf{Adv}^{\mathsf{naead}}_{\mathfrak{A}}(\mathscr{A}) \leq \epsilon.$$

Note that, security under this formulation covers the two standard security goals of authenticated encryption:

1. *Privacy:* Security against an adversary who tries to distinguish the AE or AEAD construction from an ideal random function Γ using encryption queries, and

2. *Integrity:* Security against an adversary who tries to make a successful forging attempt on the scheme using both encryption and decryption queries.

Part I

Analysis of Message Authentication Codes

Chapter 3

Pseudorandom Permutation based CBC-MAC Family

This chapter studies the security of one of the most popular family of message authentication codes, the CBC-MAC family. CBC-MAC is a block cipher based MAC construction which is based on the CBC mode of operation [153] invented by Ehrsam et al. [65]. The CBC-MAC was an international standard [95], which was proven to be secure for fixed length messages in [15, 27], and for prefix-free[1] messages in [80, 162]. The fixed length constraint is not desired in practice. One way to circumvent this is to use the length of message as the first block in CBC-MAC computation. But this requires prior knowledge of the message length. A more reasonable and popular approach is to encrypt the CBC-MAC output with an independent keyed permutation. This later approach called EMAC was first proposed during the RACE project [21]. Petrank and Rackoff [162] proved that EMAC is secure for all messages in its message space.

Although the EMAC construction is tolerant to variable length messages it has a domain limited to $\mathbb{B}^{\eta^+}$. Later works by Black and Rogaway [34], and Iwata et al. [97, 121] propose variants of CBC-MAC that theoretically[2] work for any messages in $\{0,1\}^{\eta^*}$. We refer the readers to section 1.3.1 of chapter 1 for a brief overview on the historical progress in the design and analysis of CBC-MAC and its family. In this chapter, we will mainly concentrate on CBC-MAC and EMAC. However, we present some interesting implications for ECBC and FCBC towards the end of the chapter.

[1] No two messages are prefix of each other.
[2] There could be some limitations on the length of messages due to security bounds.

3.1 PRF Analysis of CBC-MAC and EMAC

Recall that proposition 2.4.1 shows that PRF security of deterministic MACs is sufficient for MAC security. So, we concentrate on the PRF security of CBC-MAC and EMAC.

3.1.1 Permutation-based CBC-MAC and EMAC

In this chapter, we always consider the set $\mathsf{Perm}(\mathbb{B})$, and hence skip the parametrization in Perm. The CBC (cipher block chaining) function (see Figure 3.1.1) with a permutation $\pi \in \mathsf{Perm}$, viewed as a key of the construction, takes as input a message $M \in \mathbb{B}^m$ with m blocks and outputs $\mathsf{CBC}_\pi(M) := \mathfrak{v}^\pi_m$. This is inductively computed as follows: $\mathfrak{v}^\pi_0(M) = 0^n$ and for all $i \in [m]$, we have

$$\begin{aligned} \mathfrak{u}^\pi_i(M) &:= \mathfrak{v}^\pi_{i-1}(M) \oplus M_i, \\ \mathfrak{v}^\pi_i(M) &:= \pi(\mathfrak{u}^\pi_i(M)). \end{aligned} \tag{3.1}$$

We call $\mathfrak{u}^\pi(M) := (\mathfrak{u}^\pi_i(M))_{i\in[m]}$ and $\mathfrak{v}^\pi(M) := (\mathfrak{v}^\pi_i(M))_{i\in(m]}$, the *intermediate input* and *intermediate output* vectors, respectively, associated to π and M. Note that, the intermediate input vector $\mathfrak{u}^\pi(M)$ is uniquely determined by $\mathfrak{v}^\pi(M)$ (and does not depend on the permutation π). We can write down this association generically as a function $\mathsf{out2in}_M : \mathbb{B}^{m+1} \to \mathbb{B}^m$ mapping any block vector y to another block vector x where $x_i = y_{i-1} \oplus M_i$ for all $i \in [m]$. So for all permutations $\pi \in \mathsf{Perm}$, we have $\mathsf{out2in}_M(\mathfrak{v}^\pi(M)) = \mathfrak{u}^\pi(M)$.

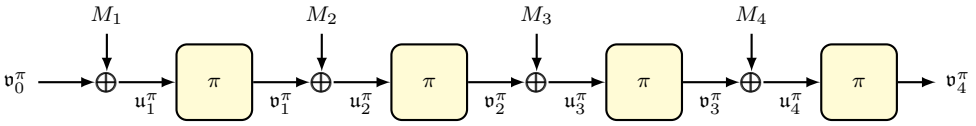

Figure 3.1.1: Evaluation of CBC function over a 4-block message M. Note that, we skipped M from the notations for intermediate input and output vectors for economical reasons.

Given the definition of CBC_π, one can easily define CBC-MAC and EMAC as follows:

1. CBC-MAC, based on a block cipher E instantiated with a key $K \in \mathcal{K}$, is defined as, $\mathsf{CBC\text{-}MAC}_{\mathsf{E}_K}(M) := \mathsf{CBC}_{\mathsf{E}_K}(M)$ for all $M \in \mathbb{B}^{\eta^*}$.
2. EMAC, based on a block cipher E instantiated twice with keys $K_1, K_2 \in \mathcal{K}$, is defined as, $\mathsf{EMAC}_{\mathsf{E}_{K_1},\mathsf{E}_{K_2}}(M) := \mathsf{E}_{K_2}(\mathsf{CBC}_{\mathsf{E}_{K_1}}(M))$ for all $M \in \mathbb{B}^{\eta^*}$.

The first step in the analysis reduces the PRF analysis of $\mathsf{CBC\text{-}MAC}_\mathsf{E}$ and $\mathsf{EMAC}_{\mathsf{E},\mathsf{E}}$ to the PRF analysis of $\mathsf{CBC\text{-}MAC}_\Pi$ and $\mathsf{EMAC}_{\Pi_1,\Pi_2}$, respectively, where Π, Π_1 and Π_2 are

uniform random permutations over $\mathbb{B}$ and Π_1 is statistically independent of Π_2. This step uses a standard hybrid argument that incurs additional cost of at most $\mathbf{Adv}^{\mathsf{prp}}_{\mathsf{E}}(\sigma, t)$ and $2\mathbf{Adv}^{\mathsf{prp}}_{\mathsf{E}}(\sigma, t)$ for CBC-MAC and EMAC, respectively. Formally, we have

$$\mathbf{Adv}^{\mathsf{prf}}_{\mathsf{CBC\text{-}MAC}_{\mathsf{E}}}(q, \ell, \sigma, t) \leq \mathbf{Adv}^{\mathsf{prp}}_{\mathsf{E}}(\sigma, t) + \mathbf{Adv}^{\mathsf{prf}}_{\mathsf{CBC\text{-}MAC}_{\Pi}}(q, \ell, \sigma) \tag{3.2}$$

$$\mathbf{Adv}^{\mathsf{prf}}_{\mathsf{EMAC}_{\mathsf{E},\mathsf{E}}}(q, \ell, \sigma, t) \leq 2\mathbf{Adv}^{\mathsf{prp}}_{\mathsf{E}}(\sigma, t) + \mathbf{Adv}^{\mathsf{prf}}_{\mathsf{EMAC}_{\Pi_1,\Pi_2}}(q, \ell, \sigma) \tag{3.3}$$

3.1.2 PRF Analysis of $\mathsf{EMAC}_{\Pi_1,\Pi_2}$

Let M and M' be two distinct messages having m and m' many blocks, respectively, and $\pi \in \mathsf{Perm}$. Let $\mathtt{Ocoll}^{\pi}(M, M')$ denote the event $\mathsf{CBC}_{\pi}(M) = \mathfrak{v}^{\pi}_{m}(M) = \mathfrak{v}^{\pi}_{m'}(M') = \mathsf{CBC}_{\pi}(M')$. We call $\mathtt{Ocoll}^{\pi}(M, M')$ the *output collision event* for a pair of messages M and M'. One can similarly define the *input collision event* $\mathtt{Icoll}^{\pi}(M, M')$ as the event $\mathfrak{u}^{\pi}_{m}(M) = \mathfrak{u}^{\pi}_{m'}(M')$. It is clear to see that $\mathtt{Icoll}^{\pi}(M, M')$ is equivalent to $\mathtt{Ocoll}^{\pi}(M, M')$, as $\mathfrak{u}^{\pi}_{m}(M) = \mathfrak{u}^{\pi}_{m'}(M')$ if and only if $\mathfrak{v}^{\pi}_{m}(M) = \mathfrak{v}^{\pi}_{m'}(M')$. So henceforth we mainly focus on the event $\mathtt{Ocoll}^{\pi}(M, M')$, which is also referred as collision event.

By extending the notation, we similarly define the collision event for a tuple of $q \geq 2$ distinct messages $M^q = (M_1, \ldots, M_q)$, as

$$\mathtt{Ocoll}^{\pi}(M^q) = \bigcup_{i \neq j \in [q]} \mathtt{Ocoll}^{\pi}(M_i, M_j). \tag{3.4}$$

We define *collision probability* as $\mathsf{outCP}(M^q) = \Pr\left[\mathtt{Ocoll}^{\Pi_1}(M^q)\right]$.

Let $\mathsf{outCP}^{\mathsf{atk}}_{q,\ell,\sigma} = \max_{M^q} \mathsf{outCP}(M^q)$ where the maximum is taken over all q-tuples of distinct messages M^q having at most ℓ blocks each, and the total length over all q queries is at most σ. Further, the message tuple satisfies the input constraint atk, which could be one of the following:

1. eq, messages must have equal length.
2. pf, no message is a prefix to others.
3. any, messages can be chosen arbitrarily within the specified length.

If no constraint is mentioned, assume any for collision events. Looking ahead momentarily, we will use pf for full collision event (defined in the next subsection), and hence in case of full collision we can always assume pf constraint.

Following [19, 34], we view EMAC as an instance of Hash-then-PRP paradigm [183] (see chapter 2 section 2.4.2.2). This enables us to reduce the problem of bounding the

PRF advantage of EMAC to bounding the collision probability. The following Eq. (3.5) is a direct application of proposition 2.4.3.

$$\mathbf{Adv}^{\mathsf{prf}}_{\mathsf{EMAC}_{\Pi_1,\Pi_2}}(q,\ell,\sigma) \leq \mathsf{outCP}^{\mathsf{any}}_{q,\ell,\sigma} + \frac{q(q-1)}{2^{n+1}}. \tag{3.5}$$

Note that, $\mathsf{outCP}^{\mathsf{any}}_{q,\ell,\sigma} \leq \binom{q}{2}\mathsf{outCP}^{\mathsf{any}}_{2,\ell}$ as the collision for q messages is the union of collision events for each of the $\binom{q}{2}$ pairs of messages. Bellare et al. [19] proved that

$$\mathsf{outCP}^{\mathsf{any}}_{2,\ell} \leq \frac{2d'(\ell)}{2^n} + \frac{64\ell^4}{2^{2n}}. \tag{3.6}$$

where $d'(\ell) = \max_{\ell' \leq \ell} d(\ell')$ and $d(\ell')$ is the the number of divisors of ℓ'. In [187], Wigert showed that $d'(\ell) = \ell^{1/\Theta(\ln\ln\ell)} = \ell^{o(1)}$. Using this bound of collision probability for a pair of messages, we see that the PRF advantage of EMAC is about $\mathrm{O}(d'(\ell)q^2/2^n)$ for $\ell < 2^{n/4}$. Later Pietrzak [163] provided an improved analysis of EMAC and proved that the PRF advantage of EMAC is about $\mathrm{O}(q^2/2^n)$ for $\ell < \min\{q^{1/2}, 2^{n/8}\}$. We revisit this improved analysis later in section 3.5.

We remark that for equal length messages Dodis et al. claim that $\mathsf{outCP}^{\mathsf{eq}}_{2,\ell} = 2^{-n} + (d(\ell))^2 \cdot \ell \cdot 2^{-2n} + \ell^6 \cdot 2^{-3n}$ (see [62, Lemma 3]), which is tight (if true).

3.1.3 PRF Analysis of CBC-MAC$_\Pi$

Let M and M' be two distinct messages having m and m' many blocks, respectively, and $\pi \in \mathsf{Perm}$. Let $\mathtt{Fcoll}^\pi(M, M')$ denote the event that

$$\mathfrak{u}^\pi_{m'}(M') \in \mathfrak{u}^\pi(M) \cup \{\mathfrak{u}^\pi_j(M') : j \in [m'-1]\}.$$

We call this the *full collision event* for a pair of messages M and M'. In other words, if the event $\mathtt{Fcoll}^\pi(M, M') \cup \mathtt{Fcoll}^\pi(M', M)$ does not hold then $\mathfrak{u}^\pi_m(M)$ and $\mathfrak{u}^\pi_{m'}(M')$ are "fresh" (did not appear before). Intuitively, for a uniform random permutation Π if this event does not hold then the CBC output should behave "almost" randomly. This intuition is one of the crucial steps in the PRF analysis of CBC-MAC.

Remark 3.1.1. We remark that in the original paper [19], the full collision event is defined through the intermediate outputs instead of inputs. Since we only consider permutation based CBC-MAC, equalities among inputs imply identical equalities among outputs.

By extending the notation, we define the full collision event for a tuple of $q \geq 2$ distinct messages $M^q = (M_1, \ldots, M_q)$ as

$$\texttt{Fcoll}^{\pi}(M^q) = \bigcup_{i \neq j \in [q]} \texttt{Fcoll}^{\pi}(M_i, M_j).$$

We define *full collision probability* as $\mathsf{fullCP}(M^q) = \Pr\left[\texttt{Fcoll}^{\Pi}(M^q)\right]$.

Analogous to $\mathsf{outCP}^{\mathsf{atk}}_{q,\ell,\sigma}$, we have $\mathsf{fullCP}^{\mathsf{atk}}_{q,\ell,\sigma} = \max_{M^q} \mathsf{fullCP}(M^q)$. Note that, $\mathsf{fullCP}^{\mathsf{any}}_{q,\ell,\sigma} = 1$ as one can always choose two messages M and M' such that M is a prefix of M', whence the full collision holds trivially. So it only makes sense to study fullCP for input constraints, eq and pf. This is fine as CBC-MAC is trivially insecure when the adversary can make prefix queries (using the classical length extension attack). In [19], Bellare et al. proved the following result

$$\mathbf{Adv}^{\mathsf{prf}}_{\mathsf{CBC\text{-}MAC}_{\Pi}}(q, \ell, \sigma) \leq q^2(\mathsf{fullCP}^{\mathsf{pf}}_{2,\ell} + 4\ell/2^n). \tag{3.7}$$

We state and prove another relation between PRF advantage of CBC-MAC and full collision probability among q prefix-free messages. The above Eq. (3.7) would be a straightforward application of the following result.

Proposition 3.1.2. *For prefix-free queries, we have*

$$\mathbf{Adv}^{\mathsf{prf}}_{\mathsf{CBC\text{-}MAC}_{\Pi}}(q, \ell, \sigma) \leq \mathsf{fullCP}^{\mathsf{pf}}_{q,\ell,\sigma} + \frac{2\sigma q}{2^n} + \frac{q^2}{2^{n+1}}.$$

Proof. We use the coefficient-H technique to prove this proposition. Let Ω denote the set of all transcripts realizable via a uniform random function Γ. A typical transcript $\omega \in \Omega$ is of the form (M^q, T^q), where $M^q \in (\mathbb{B}^{\eta^+})_q$ is prefix-free and $T^q \in \mathbb{B}^q$.

We say that a transcript $\omega \in \Omega$ is bad, denoted $\omega \in \Omega_{\mathsf{bad}}$, if $T^q \neq \widehat{T}^q$, i.e., there exist $i \neq j \in [q]$ such that $T_i = T_j$. We have

$$\Pr\left[\Theta_0 \in \Omega_{\mathsf{bad}}\right] \leq \binom{q}{2} \frac{1}{2^n},$$

as there are at most $\binom{q}{2}$ pairs (i, j) and for each such pair the equality $\mathsf{T}_i = \mathsf{T}_j$ holds with at most 2^{-n} probability.

Now, fix a transcript $\omega = (M^q, T^q) \in \Omega \setminus \Omega_{\mathsf{bad}}$. Then, we must have $T^q \in (\mathbb{B})_q$. For $i \in [q]$, let m_i denote the length of M_i. Then $m_i \leq \ell$ and $\sum_{j \in [q]} m_j \leq \sigma$. In the ideal world, we have

$$\Pr\left[\Theta_0 = \omega\right] = \Pr\left[\Gamma(M^q) = T^q\right] = \frac{1}{2^{nq}}. \tag{3.8}$$

In the real world, a permutation $\pi \in \mathsf{Perm}$ is called bad if and only if

1. $\texttt{Fcoll}^{\pi}(M^q)$ holds, or
2. $\mathfrak{v}_i^{\pi}(M_r) = T_{r'}$, for some $r, r' \in [q], i \in [m_r]$.

Let $\mathsf{badPerm} = \{\pi \in \mathsf{Perm} : \pi \text{ is bad}\}$. All other permutations $\pi \notin \mathsf{badPerm}$ are called good. We define an equivalence relation $\sim$ on Perm as $\pi \sim \pi'$ if $\mathfrak{u}^{\pi}(M_i) = \mathfrak{u}^{\pi'}(M_i)$ for all $i \in [q]$. It is clearly an equivalence relation and a good permutation can only be related with another good permutation and good permutations are unrelated with bad permutations. An equivalence class consisting of good permutations is referred as good equivalence class. Let $\mathcal{C}$ be a good equivalence class. Let s denote the cardinality of $\cup_{r\in[q]}\widehat{\mathfrak{u}}^{\pi}(M_r)$, where $\pi \in \mathcal{C}$. Recall that $\widehat{\mathfrak{u}}^{\pi}(M_r)$ denotes the set of distinct elements in $\mathfrak{u}^{\pi}(M_r)$. Note that, s is the same for all $\pi \in \mathcal{C}$. Then, $|\mathcal{C}| = (2^n - s)!$, as the outputs of exactly s inputs of $\pi \in \mathcal{C}$ are determined due to CBC computation on M^q. Since the T_i's are not intermediate outputs, we have

$$|\{\pi \in \mathcal{C} : \mathsf{CBC}_{\pi}(M^q) = T^q\}| = (2^n - s - q)!,$$

since q additional restrictions on input-output are being added. So for any class of good permutations $\mathcal{C}$,

$$\Pr\left[\mathsf{CBC}_{\Pi}(M^q) = T^q \mid \Pi \in \mathcal{C}\right] = \frac{(2^n - s - q)!}{(2^n - s)!} \geq 2^{-nq}. \tag{3.9}$$

Thus,

$$\begin{aligned}
\Pr\left[\Theta_1 = \omega\right] &= \Pr\left[\mathsf{CBC}_{\Pi}(M^q) = T^q\right] \\
&\overset{1}{\geq} \sum_{\mathcal{C} \text{ is good}} \Pr\left[\Pi \in \mathcal{C}\right] \times \Pr\left[\mathsf{CBC}_{\Pi}(M^q) = T^q \mid \Pi \in \mathcal{C}\right] \\
&\overset{2}{\geq} \Pr\left[\Pi \in \mathsf{Perm} \setminus \mathsf{badPerm}\right] \times 2^{-nq} \\
&\overset{3}{\geq} \Big(1 - \Pr\left[\Pi \in \mathsf{badPerm}\right]\Big) \times \Pr\left[\Theta_0 = \omega\right] \\
&\overset{4}{\geq} \left(1 - \mathsf{fullCP}^{\mathsf{pf}}_{q,\ell,\sigma} - \frac{\sigma q}{2^{n-1}}\right) \times \Pr\left[\Theta_0 = \omega\right].
\end{aligned}$$

where 1 to 2 follows from Eq. (3.9); 2 to 3 follows from Eq. (3.8); and 3 to 4 follows from the two conditions for $\pi \in \mathsf{badPerm}$. The first condition leads to the term $\mathsf{fullCP}^{\Pi}_{q,\ell,\sigma}$. The second condition says that we sample at most $\sigma - q$ outputs of a random permutation and one of them belongs to the set $\widehat{T}^q$. This can happen with probability at most $\sigma q/(2^n - \sigma + q)$ which is further less than $\sigma q/2^{n-1}$ provided $\sigma < 2^{n-1}$. If $\sigma \geq 2^{n-1}$, then the above bound holds trivially.

The result follows from coefficient-H technique (Corollary 2.2.2). □

The bound in Proposition 3.1.2 is potentially a better bound than the original in Eq. (3.7) as it uses the total number of blocks σ, which could be much less than ℓq in some scenarios.[3]

Note that, $\mathsf{fullCP}^{\mathsf{pf}}_{q,\ell} \leq q(q-1)\mathsf{fullCP}^{\mathsf{pf}}_{2,\ell}$ by considering all ordered pairs (M_i, M_j). This proves the original claim from [19] as stated in Eq. (3.7). In [19], it is proved that

$$\mathsf{fullCP}^{\mathsf{pf}}_{2,\ell} \leq \frac{8\ell}{2^n} + \frac{64\ell^4}{2^{2n}}. \tag{3.10}$$

In section 3.2, we show a critical flaw in [19, Lemma 10]. This invalidates the proofs of Eq. (3.6) and Eq. (3.10) given in [19]. Further, it also invalidates the security analysis of EMAC in [163]. Consequently, we revise the bounds for $\mathsf{outCP}^{\mathsf{any}}_{q,\ell,\sigma}$ and $\mathsf{fullCP}^{\mathsf{pf}}_{q,\ell,\sigma}$ in section 3.4. Moreover, our revised bound of $\mathsf{fullCP}^{\mathsf{pf}}_{q,\ell,\sigma}$ would be in the order $\sigma q/2^n$ instead of $q^2\ell/2^n$ (whenever $\ell \leq 2^{n/3}$). We also revise the analysis of [163] and obtain a degraded bound in section 3.5. In the same section, we provide a simplified security analysis for EMAC that achieves significantly improved bounds for message lengths less than $2^{n/4}$.

3.2 Revisiting Structure Graph

In the previous section, we have seen how the PRF advantage of CBC-MAC or EMAC is essentially reduced to the bound on probability of some collision events on internal inputs or outputs of the underlying permutation. Thus, it would be useful to have an object which deals with the intermediate inputs and outputs. Bellare et al. [19] introduced one such object, called structure graph, and used it to bound the (full) collision probabilities in [19]. In this section we revisit the structure graph formalism and show that one of the main claims, namely [19, Lemma 10] about structure graphs is false.

Let us fix a tuple of messages M^q for the rest of this chapter, where $M_i \in \mathbb{B}^{m_i}$ and $m_i \leq \ell$ for all $i \in [q]$, and $\sum_{i\in[q]} m_i = m \leq \sigma$. In addition, we may put additional constraint on these messages, like prefix-free property.

3.2.1 Intermediate Inputs and Outputs

INDEX SET: We first collect all intermediate inputs and outputs which are obtained through the computation of $\mathsf{CBC}_\pi(M_r)$ for all r. These intermediate values will be

[3]Consider a scenario where most of the messages are of few kilobytes, but the maximum permissible message length is in Gigabytes.

defined as a sequence over a two-dimensional index set. Each index is a pair where the first element of the pair corresponds to the message number and the second element is the block number of that message. More formally, we define the *index set*

$$\mathcal{I} = \{(r, i) : r \in [q], i \in [m_r]\},$$

and the *dictionary order* $\prec$ on it as follows: $(r, i) \prec (r, i)$, and $(r, i) \prec (r', i')$ if $r < r'$ or $r = r'$ and $i < i'$. Let x be a sequence over this index set. For any $r \in [q]$, the subsequence $(x_{(r,1)}, \ldots, x_{(r,m_r)})$ is denoted by x_{r*}. Sometimes we also consider the index set $\mathcal{I}_0 = \mathcal{I} \cup \{(r, 0) : r \in [q]\}$, and the natural extension of the order $\prec$ on $\mathcal{I}_0$.

SEQUENCES FOR INTERMEDIATE INPUTS AND OUTPUTS: We denote the *sequences of intermediate outputs* and *inputs* over the index set $\mathcal{I}_0$ and $\mathcal{I}$ as $\mathfrak{v}^\pi(M^q)$ and $\mathfrak{u}^\pi(M^q)$, respectively, where $\forall r \in [q]$, $\mathfrak{v}^\pi_{r*}(M^q) := \mathfrak{v}^\pi(M_r)$, and $\mathfrak{u}^\pi_{r*}(M^q) := \mathfrak{u}^\pi(M_r)$.

For a single message M, we have previously seen that the intermediate input sequence is uniquely determined by the intermediate output sequence, and we denoted the association by a function $\mathsf{out2in}_M$. The same is true for q messages and we extend this definition as follows: Given any block sequence y over the index set $\mathcal{I}_0$, we define $\mathsf{out2in}(y)$ as a block sequence x over the index set $\mathcal{I}$, where $x_{r*} = \mathsf{out2in}_{M_r}(y_{r*})$ for all $r \in [q]$. Thus, for any π, we have $\mathsf{out2in}_{M^q}(\mathfrak{v}^\pi(M^q)) = \mathfrak{u}^\pi(M^q)$. Onwards, we selectively skip the parametrization of $\mathfrak{v}$ and $\mathfrak{u}$ whenever the parameters are clear from the context.

3.2.2 Structure Graphs and Block-Vertex Structure Graphs

A block-vertex structure graph is a graph theoretic representation of intermediate output sequence $\mathfrak{v}^\pi$. The *block-vertex structure graph* $\mathsf{BStruct}^\pi = (\mathcal{V}^\pi, \mathcal{E}^\pi)$ for a permutation π is defined by the labeled vertex set $\mathcal{V}^\pi := \widehat{\mathfrak{v}^\pi}$, and the set of labeled edges

$$\mathcal{E}^\pi := \bigcup_{r=1}^{q} \{(\mathfrak{v}^\pi_{(r,i-1)}, \mathfrak{v}^\pi_{(r,i)}, M_{(r,i)}) : i \in [m_r]\}.$$

Clearly, $\mathsf{BStruct}^\pi$ is a union of M_r-walks (see section A.1.2 of appendix A) for all $r \in [q]$, and vertex $0^n \in \mathcal{V}$ has positive out-degree. Note that,

$$u \xrightarrow{A} v \;\Rightarrow\; \pi(u \oplus A) = v. \tag{3.11}$$

So, for every $v \in \mathcal{V}$, *all outward edges (similarly for inward edges) have distinct edge labels.* Using this property, it is easy to see that *the walks are unique* and we denote them by W_{M_i} or simply W_i whenever the message tuple is understood. See Figure 3.2.1 for a possible block structure graph corresponding to a single message. It is clear to see that

block structure graph is just another view in which the input and output vectors are stored in a directed graph. While storing the intermediate sequences as a set of labeled

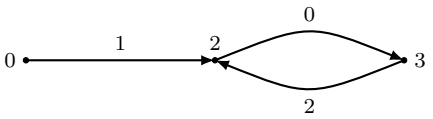

Figure 3.2.1: Let $M_1 = (1, 0, 2, 0, 2)$ and $\pi(1) = 2$, $\pi(2) = 3$. For any such π, we have $\mathfrak{v}^\pi = (0, 2, 3, 2, 3, 2)$ and $\mathfrak{u}^\pi = (1, 2, 1, 2, 1)$. However, the graph consists of just three vertices $\mathcal{V}^\pi = \{0, 2, 3\}$ and edge set $\mathcal{E}^\pi = \{(0, 2, 1), (2, 3, 0), (3, 2, 2)\}$. We see that the intermediate input and output sequences actually can be reconstructed from this labeled structure graph. The walk corresponding to the message M_1 will uniquely identify the output vector as $\mathfrak{v}^\pi = (0, 2, 3, 2, 3, 2)$ and the input vector $\mathfrak{u}^\pi = (1, 2, 1, 2, 1)$ can be constructed using the relation between input, output and message.

edges, we may loose the order as well as the repetition of the elements. Interestingly, we see that we can uniquely reconstruct the intermediate sequences from such an edge-labeled graph by using uniqueness of M_i-walk. More precisely, $\mathfrak{v}^\pi_{(r,i)} = W_{(r,i)}$, where $W_{(r,i)}$ denotes the i-th vertex on the walk W_r.

Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be a labeled directed graph and $f : \mathcal{V} \to \mathcal{V}'$ be an injective function. Let $\widetilde{\mathcal{V}} = \{v' \in \mathcal{V}' : \exists v \in \mathcal{V},\ f(v) = v'\}$. Then, f is a bijection from $\mathcal{V}$ to $\widetilde{\mathcal{V}}$. One can define a labeled directed graph $\widetilde{\mathcal{G}} = (\widetilde{\mathcal{V}}, \widetilde{\mathcal{E}})$ isomorphic to $\mathcal{G}$ for which f is an isomorphism. More precisely, $(u, v, a) \in \mathcal{E}$ if and only if $(f(u), f(v), a) \in \widetilde{\mathcal{E}}$. We call the graph $\widetilde{\mathcal{G}}$ so obtained, the f-transformed $\mathcal{G}$, denoted $\widetilde{\mathcal{G}} = \alpha(\mathcal{G})$.

For a block-vertex structure graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, we define the *minimum index mapping* $\alpha : \mathcal{V} \to \mathcal{I}_0$ as $\alpha_v = \alpha(v) \mapsto (r, i)$ for all $v \in \mathcal{V}$, where (r, i) is the minimum index such that $W_{(r,i)} = v$. Clearly, α is an injective mapping, with some range set $\widetilde{\mathcal{V}}$.

Definition 3.2.1 (Structure Graph)**.** The structure graph $\mathsf{Struct}^\pi = (\widetilde{\mathcal{V}}^\pi, \widetilde{\mathcal{E}}^\pi)$ associated to π is the α-transformed $\mathsf{BStruct}^\pi$, i.e. $\mathsf{Struct}^\pi = \alpha(\mathsf{BStruct}^\pi)$.

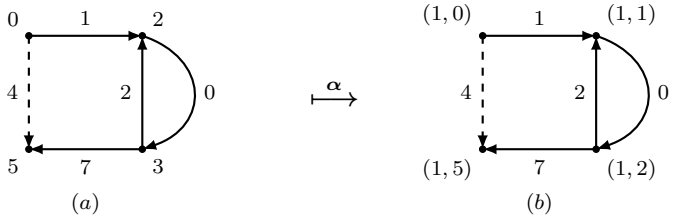

Figure 3.2.2: Structure graph corresponding to a block-vertex structure graph.

Example 3.1. *Let* $M_1 = (1, 0, 2, 0, 7)$ *and* $M_2 = (4)$ *be two messages and* $\pi(1) = 2$; $\pi(2) = 3$; $\pi(4) = 5$ *for some* $\pi \in \mathsf{Perm}$. *Then, we have* $\mathfrak{v}^\pi(M_1) = (0, 2, 3, 2, 3, 5)$ *and* $\mathfrak{v}^\pi(M_2) = (0, 5)$. *The corresponding block-vertex structure graph* $\mathsf{BStruct}^\pi$, *shown in Figure 3.2.2(a), has vertex set* $\mathcal{V}^\pi = \{0, 2, 3, 5\}$ *and edge set*

$$\mathcal{E}^\pi = \{(0, 2, 1), (0, 5, 4), (2, 3, 0), (3, 2, 2), (3, 5, 7)\}.$$

The corresponding structure graph Struct^{π}*, illustrated in Figure 3.2.2(b), has vertex set* $\widetilde{\mathcal{V}}^{\pi} = \{(1,0),(1,1),(1,2),(1,5)\}$ *and edges set*

$$\widetilde{\mathcal{E}}^{\pi} = \{((1,0),(1,1),1),((1,1),(1,2),0),((1,2),(1,1),2),((1,2),(1,5),7),((1,0),(1,5),4)\}.$$

Let $\widetilde{W_r}$ denote the M_r-walk in a structure graph $\widetilde{\mathcal{G}}$ (corresponding to some block-vertex structure graph $\mathcal{G}$). It is easy to see that a structure graph is again a *union of* M_r*-walks* $\widetilde{W_r}$ starting from $(1,0)$. Note that, as per the convention used here and in the preceding discussion $\widetilde{W_r} = \alpha(W_r)$, where W_r is the corresponding M_r-walk in $\mathcal{G}$. A structure graph is called a *zero-output graph* if $(1,0)$ has positive in-degree, otherwise we call it *non-zero output graph*. To express it mathematically, we define a binary function Iszero such that for each zero-output graph $\widetilde{\mathcal{G}}$, $\mathsf{Iszero}(\widetilde{\mathcal{G}}) = 1$, otherwise it maps to 0.

To reconstruct a block-vertex structure graph realizing $\widetilde{\mathcal{G}}$ we have to find labels from $\mathbb{B}$ for all the vertices in a "consistent manner" and we call such a labeling *valid*. The mapping must be injective as distinct vertices in a block vertex structure graph map to distinct vertices in the corresponding structure graph.

Definition 3.2.2 (valid block labeling)**.** An injective function $\beta : \widetilde{\mathcal{V}} \to \mathbb{B}$ is called a *valid block labeling* for a structure graph $\widetilde{\mathcal{G}} = (\widetilde{\mathcal{V}}, \widetilde{\mathcal{E}})$ if the graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is a block-vertex structure graph where

1. $\mathcal{V} := \beta(\widetilde{\mathcal{V}}) = \{\beta_v := \beta(v) : v \in \widetilde{\mathcal{V}}\}$, and
2. $\mathcal{E}$ is the edge set after relabeling v by β_v, i.e. $\mathcal{E} = \{(\beta_u, \beta_v, x) : (u, v, x) \in \mathcal{E}\}$.

NECESSARY CONDITIONS FOR A VALID BLOCK LABELING: Not all injective functions from $\widetilde{\mathcal{V}}$ to $\mathbb{B}$ are valid block labeling. An obvious class of counter-example functions are those which map $(1,0) \in \widetilde{\mathcal{V}}$ to a non-zero value in $\mathbb{B}$. We now enumerate other necessary conditions for valid block labeling. First of all, by definition, β_i should all be distinct as the valid block label is injective. In addition to this, whenever $e_1 := (u, w, a), e_2 := (v, w, b) \in \widetilde{\mathcal{E}}$ we must have $\beta_u \oplus a = \beta_v \oplus b$ as these are equal to $\pi^{-1}(z)$ for some $\pi \in \mathsf{Perm}$.

An *input-collision or simply a collision* in a structure graph $\widetilde{\mathcal{G}}$ is defined by such a triple $\delta = (u, v; w)$. The set $\{u, v\}$ is called the *source of the collision* whereas w is called the *head of the collision*. We also say the edges e_1 and e_2 are colliding edges. Thus, an input-collision $\delta = (u, v; w)$ induces a linear restriction $L_\delta : \beta_u \oplus \beta_v = c_\delta$ where $c_\delta = a \oplus b \in \mathbb{B}$. A valid block label must satisfy such linear restrictions for all collisions δ. Let $\Delta(\widetilde{\mathcal{G}})$ denote the set of all collisions in $\widetilde{\mathcal{G}}$. Let $\mathsf{rank}(\widetilde{\mathcal{G}})$ denote the rank of the system of linear equations $\{L_\delta : \delta \in \Delta(\widetilde{\mathcal{G}})\}$.

Now, we define an important parameter, called *accident*, useful in characterizing structure graphs. It is defined depending on whether the graph is zero-output graph or not.

Definition 3.2.3 (Accident of a structure graph)**.** The (number of) accident of a structure graph $\widetilde{\mathcal{G}}$ is defined as, $\mathsf{Acc}(\widetilde{\mathcal{G}}) := \mathsf{rank}(\widetilde{\mathcal{G}}) + \mathsf{Iszero}(\widetilde{\mathcal{G}})$.

Thus, the accident of a non-zero structure graph $\widetilde{\mathcal{G}}$ is simply $\mathsf{rank}(\widetilde{\mathcal{G}})$, whereas the accident of a zero-output graph is $\mathsf{rank}(\widetilde{\mathcal{G}}) + 1$.

Lemma 3.2.4. *If there is a vertex v with in-degree d then $\mathsf{Acc}(\widetilde{\mathcal{G}}) \geq d - 1$. Moreover, if the graph is a zero-output graph then $\mathsf{Acc}(\widetilde{\mathcal{G}}) \geq d$.*

Proof. Let $u_1, \ldots, u_d$ be all the predecessors of v. Let us define an input-collision $\delta_{i,j} := (u_i, u_j; v)$ (ignoring the edge labels). It is now easy to see that $L_{\delta_{i,j}} = L_{\delta_{1,i}} \oplus L_{\delta_{1,j}}$ for all $i, j \in [d] \setminus \{1\}$. Moreover, the $L_{\delta_{1,i}}$'s are linearly independent. This, immediately proves the first part. The second part is just an extension of the first part using the definition of accident. □

Remark 3.2.5. Another simple but useful observation is as follows: if a structure graph $\widetilde{\mathcal{G}}$ has at least two collisions with distinct sources, then $\mathsf{rank}(\widetilde{\mathcal{G}}) \geq 2$.

Let $\widetilde{\mathcal{G}} = (\widetilde{\mathcal{V}}, \widetilde{\mathcal{E}})$ be a structure graph with rank r and $|\widetilde{\mathcal{V}}| = s + 1$. Then fixing some $s - r$ choices of β_i values will uniquely determine the rest[4], and so the number of valid block labeling is at most $(2^n)_{s-r}$. Any valid choice of β induces a block-vertex structure graph $\mathcal{S} = (\mathcal{V}, \mathcal{E})$ such that $\widetilde{\mathcal{G}} = \widetilde{S} = \alpha(\mathcal{S})$. Note that, $s + \mathsf{Iszero}(\mathcal{S})$ is the number of vertices $v \in \mathcal{V}$ with positive in-degree. So exactly $(2^n - s - \mathsf{Iszero}(\mathcal{S}))!$ permutations can result in the block-vertex structure graph $\mathcal{S}$. Therefore,

$$\Pr\left[\mathsf{BStruct}^{\Pi} = \mathcal{S}\right] = \frac{(2^n - (s + \mathsf{Iszero}(\mathcal{S})))!}{2^n!} = \frac{1}{(2^n)_{s+\mathsf{Iszero}(\mathcal{S})}},$$

whence, we get

$$\begin{aligned}\Pr\left[\mathsf{Struct}^{\Pi} = \widetilde{\mathcal{G}}\right] &= \sum_{\mathcal{S}:\widetilde{\mathcal{S}}=\widetilde{G}} \Pr\left[\mathsf{BStruct}^{\Pi} = \mathcal{S}\right] \\ &\overset{1}{=} \sum_{\mathcal{S}:\widetilde{\mathcal{S}}=\widetilde{G}} \frac{1}{(2^n)_{s+\mathsf{Iszero}(\mathcal{S})}} \\ &\overset{2}{\leq} \frac{(2^n)_{s-r}}{(2^n)_{s+a-r}}. \end{aligned} \tag{3.12}$$

[4] After assigning values for any $s - r$ variables, the system of linear equations will have a unique solution.

In 1 the sum is taken over all block-vertex structure graphs $\mathcal{S}$ such that the induced structure graph $\widetilde{\mathcal{S}} = \widetilde{\mathcal{G}}$. From 1 to 2, there are at most $(2^n)_{s-r}$ choices for $\mathcal{S}$, and for each such choice $\mathsf{lszero}(\mathcal{S})$ is equal to $a - r$ (i.e. lszero is invariant for fixed $\widetilde{\mathcal{G}}$), where a denotes the accident of $\widetilde{\mathcal{G}}$.

The following result bounds the probability of realizing a structure graph with accident a for the tuple of messages M^q. The proof is just an extension of Eq. (3.12) using the condition that $s < m$.

Lemma 3.2.6. *For any structure graph $\widetilde{\mathcal{G}}$ with accident a, we have*

$$\Pr_{\Pi}\left[\mathsf{Struct}^{\Pi} = \widetilde{\mathcal{G}}\right] \leq \frac{1}{(2^n - m)^a}.$$

Now we state another important result which bounds the number of structure graphs with accident a. We skip the proof of this result as it is readily available in [19, 163].

Lemma 3.2.7. *The number of structures graphs associated to $M^q = (M_1, \ldots, M_q)$ with accident a is at most $\binom{m}{2}^a$. In particular, there exists exactly one structure graph with accident 0.*

Corollary 3.2.8. *For $a \in \mathbb{N}$ and $m < 2^{n-1}$, we have*

$$\Pr_{\Pi}\left[\mathsf{Acc}(\mathsf{Struct}^{\Pi}) \geq a\right] \leq \left(\frac{m^2}{2^n}\right)^a.$$

Proof. The result follows from straightforward algebraic simplification after applying Lemma 3.2.6 and Lemma 3.2.7. □

3.2.3 True Collision and an Observation on [19, Lemma 10]

Let $\widetilde{\mathcal{G}}$ be a structure graph and $\widetilde{W}_i$ the M_i-walk. Suppose we reconstruct the graph $\widetilde{\mathcal{G}}$ again by traversing all the walks $\widetilde{W}_i$ for $i \in [q]$ in ascending order. While we walk along $\widetilde{W}_i$ for all i we count how many times we reach an existing vertex, which increases its current in-degree. The total count is defined to be the number of *true collisions* of the graph.

Mathematically, one can define it as follows: For a vertex $v \in \widetilde{\mathcal{V}} \setminus \{(1,0)\}$, we define the number of true collisions at v by $\mathsf{TC}(v) := \mathbf{deg_{in}}(v) - 1$ and $\mathsf{TC}(1,0) = |\mathbf{deg_{in}}(1,0)|$, where $\mathbf{deg_{in}}(u)$ denotes the in-degree of u (for details see section A.1.1 of appendix A). The (number of) true collisions of $\widetilde{\mathcal{G}}$ is the sum $\mathsf{TC}(\widetilde{\mathcal{G}}) := \sum_{v \in \widetilde{\mathcal{V}}} \mathsf{TC}(v)$. By lemma 3.2.4, we know that $\mathsf{Acc}(\widetilde{\mathcal{G}}) \geq \mathsf{TC}(v)$ for all $v \in \widetilde{\mathcal{V}}$. From the definition of the accident it is also obvious that $\mathsf{Acc}(\widetilde{\mathcal{G}}) \leq \mathsf{TC}(\widetilde{\mathcal{G}})$.

LEMMA 10 OF [19]: Like [19, 163], we will predominantly work with accident 1 structure graphs. To identify all structure graphs with accident 1 it would be good if we can have some relationship between true collisions and accidents. Lemma 10 of [19] was meant for this. It says that when $q = 2$, $\mathsf{Acc}(\widetilde{\mathcal{G}}) = 1 \Rightarrow \mathsf{TC}(\widetilde{\mathcal{G}}) = 1$. We show that this lemma is not true, via two counter-examples described below.

1. For the first example, illustrated in Figure 3.2.3(a), let

$$M_1 = (M_{(1,1)}, M_{(1,2)}, M_{(1,3)}, M_{(1,2)}, M_{(1,4)}) \text{ and } M_2 = (M_{(2,1)})$$

be two messages such that $M_{(2,1)} := M_{(1,1)} \oplus M_{(1,3)} \oplus M_{(1,4)}$. Here, we have two input-collisions $\delta_1 := ((1,0),(1,2);(1,1))$ and $\delta_2 := ((1,0),(1,2);(1,5))$. The two linear equations L_{δ_1} and L_{δ_2} corresponding to the two input-collisions are same as $\beta_{(1,0)} \oplus \beta_{(1,2)} = M_{(1,1)} \oplus M_{(1,3)}$ and so the rank (which is also the accident in this case) is 1. However, true collision is two (at $(1,1)$ and $(1,5)$). This contradicts [19, Lemma 10].

2. For the second example, illustrated in Figure 3.2.3(b), let

$$M_1 = (M_{(1,1)}, M_{(1,2)}, M_{(1,3)}) \text{ and } M_2 = (M_{(2,1)})$$

be two messages such that $M_{(2,1)} := M_{(1,1)} \oplus M_{(1,2)} \oplus M_{(1,3)}$. Here, we have two input-collisions $\delta_1 := ((1,0),(1,1);(1,1))$ and $\delta_2 := ((1,0),(1,1);(1,3))$. The two linear equations L_{δ_1} and L_{δ_2} corresponding to the two input-collisions are same as $\beta_{(1,0)} \oplus \beta_{(1,1)} = M_{(1,1)} \oplus M_{(1,2)}$ and so the rank and accident is 1. Again, this contradicts [19, Lemma 10].

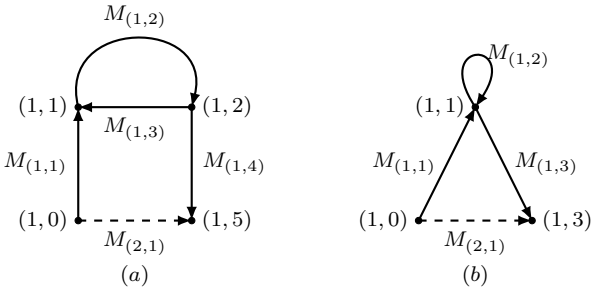

Figure 3.2.3: Two examples of structure graphs with accident 1 and true collision 2.

This lemma was a crucial step in characterizing all accident 1 structure graphs in [19, 163]. As this is shown to be wrong, it is important to revisit and rectify the results in [19, 163] as much as possible.

3.3 Characterization of Accident-1 Structure Graphs

In this section, we characterize all structure graphs with accident 0 or 1. To do so we characterize single message structure graphs which is much easier. Later in the section, we characterize all structure graphs for a pair of messages satisfying some event. From here onwards, we will not deal with block-vertex structure graph. So, for simplicity we use $\mathcal{G}$ (instead of $\widetilde{\mathcal{G}}$) to represent a structure graph and W_r (instead of $\widetilde{W_r}$) to represent the M_r-walk in the structure graph.

Let $\mathsf{Struct}(M^q)$ denote the set of all structure graphs for the tuple of messages M^q. Let $\mathsf{Struct}_a(M^q) = \{\mathcal{G} \in \mathsf{Struct}(M^q) : \mathsf{Acc}(\mathcal{G}) = a\}$, the set of all structure graphs associated to M^q with accident a. As before, we will drop the parameter M^q, whenever it is clear from the context. We are interested in $\mathsf{Struct}_0(M^q)$ and $\mathsf{Struct}_1(M^q)$, the sets of all structure graphs with accident 0 and 1, respectively. Lemma 3.2.7 says that the number of graphs with accident 1 is at most $\binom{m}{2}$, where $m = \sum_i m_i$ and $M_i \in \mathbb{B}^{m_i}$. The number of structure graphs with accident 0 is at most one. In the following we actually identify this structure graph.

FREE GRAPH: As there is no accident every non-zero vertex has in-degree 1 and $(1,0)$ has in-degree 0 (i.e., non-zero output graph). Being a structure graph, $\mathcal{G}$ is union of M_i-walks W_i. An M_i-walk starting from $(1,0)$ with no vertex having in-degree 2 must be a path. So $\mathcal{G}$ is a union of M_i-paths W_i. Now, for any $i \neq j$, let $p = \mathsf{lcp}(M_i, M_j)$. Then, $W_i^{[p]} = W_j^{[p]}$, where $W_i^{[p]}$ denotes the sub-walk of W_i consisting of the first p vertices, and $W_{(i,p+1)} \neq W_{(j,p+1)}$ (if these are defined). It is also easy to see that $W_i^{[p]}$, $W_i^{[p+1\ldots m_i]}$, and $W_j^{[p+1\ldots m_j]}$ are disjoint paths. Thus, any two paths W_i and W_j are identical up to the length of the largest common prefix of M_i and M_j and afterwards they remain disjoint. We call this unique graph the *free graph* of M^q. A free graph for three messages is illustrated in Figure 3.3.1.

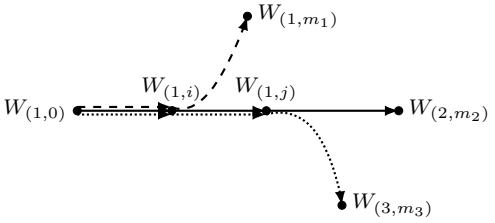

Figure 3.3.1: A free graph of three messages. The dashed, solid and dotted lines denote the three walks corresponding to the three messages.

3.3.1 Accident-one Structure Graphs for a Single Message

Now, we consider accident 1 structure graphs for a single message $M \in \mathbb{B}^m$. Note that, any such structure graph must be a walk W of length m.

In this subsection, we exclusively work with single message, hence the index set $\mathcal{I} = \{(1, i) : i \in (m]\}$ is uniquely identified by the set $(m]$. So, we skip the first coordinate in this subsection. We say a node W_i *is fresh* in the walk if $W_i \neq W_j$ for all $j \neq i$.

3.3.1.1 Case A: Zero-output Graphs

As 0 (this denotes $(1, 0)$) has positive in-degree there cannot be any more collision pairs otherwise the accident would be at least two. Let c be the minimum positive integer such that $W_c = 0$, so we have a cycle $(W_0, W_1, \ldots, W_c)$. Let A be its label. Suppose $M = A^i \| B$, where i is the maximum positive integer for which we can write M in this form. So A is not a prefix of B. Let $s = \mathsf{lcp}(A, B)$. Thus, $W_{ic+j} = W_j$ for all $j \in (s]$.

1. If B is a prefix of A then the structure graph is a cycle of size c ending at W_s. It is illustrated in Figure 3.3.2(a) (the $*$ is empty).
2. If B is not a prefix of A then $W_{ic+s} = W_s$ and $W_{ic+s+1} \neq W_{s+1}$. Further, $W_{ic+s+1} \neq W_j$ for all $j \in [c]$ since otherwise we get a collision. In fact, it can be shown that all subsequent nodes are fresh. Suppose not, then let $j > ic + s + 1$ be the first such integer for which $W_j = W_k$ for some $k < j$, hence we obtain a collision. So, the structure graph is an edge disjoint union of a cycle of size c and a path starting from s, as illustrated in Figure 3.3.2(a) (the $*$ is non-empty). The length of the cycle is c, whereas the length of the path is $m - ic - s$. We also call this graph ρ' graph. The tail (path from 0 to the cycle) of the ρ' walk is empty.

3.3.1.2 Case B: Non-zero Output Graphs

As 0 has in-degree 0, there is a collision $\delta = (u_0, v_0; w)$. In fact, all other collisions must have same source as that of δ (otherwise the number of accidents increases).

Let (i_0, j_0) be the smallest positive distinct integers such that $W_{i_0} = W_{j_0}$.[5] As 0 has in-degree 0 so $1 \leq i_0 < j_0$, $W_{i_0-1} = u_0$, and $W_{j_0-1} = v_0$. Let A and B be the labels of $W^{(i_0]}$, and $W^{[i_0 \ldots j_0]}$, respectively, and $j_0 - i_0 = c$. Then, $A \| B$ is a prefix of M. Let t be the largest positive integer such that $M = A \| B^t \| C$. So, B is not a prefix of C. If C is a

[5] i_0 and j_0 can be fixed one by one. First fix i_0 to be the smallest positive integer such that $W_{i_0} = W_j$, for some $j \in [i_0 + 1 \ldots m]$. Now, fix the smallest positive integer j_0 such that $W_{j_0} = W_{i_0}$.

prefix of B then we can have structure graphs as illustrated in Figure 3.3.2(d) or 3.3.2(f) (the end point lies inside the cycle). Suppose C is not a prefix and let $s = \mathsf{lcp}(B; C)$.

Claim 3.3.1. *The sub-walk left after traversing W till $A\|B^t\|C^{[s]}$ is a path and disjoint from the rest (illustrated in Figure 3.3.2(c)).*

Proof. Suppose $\exists v \neq W_{i_0+tc+s} \in W^{[i_0+tc+s\ldots m]} \cap W^{[i_0+tc+s]}$. Let $r = i_0 + tc + s$. We distinguish the following two cases:

Case B.1 — $W_{r+1} \in W^{[r]}$: If $s \neq c-1$, then we have a new collision $\delta' = (W_{i-1}, W_r; W_i)$ independent of δ which increases the accident to 2. If $s = c - 1$, then $i \neq i_0 + kc$ for all $k \in (t]$. This can be argued as $\beta_{i_0-1} \oplus A_{i_0} = \beta_{j_0-1} \oplus B_c = \beta_{j_0-1} \oplus B_{s+1} \neq \beta_{j_0-1} \oplus C_{s+1}$ (as $B_{s+1} \neq C_{s+1}$). Now, the only way to make δ' dependent on δ is to have $W_{i-1} = W_{i_0-1}$. This implies a collision at W_j where $j \in [i_0 - 1]$, as the walk must come back to W_{i_0-1} at the $(i-1)$-th step. This again gives a new accident as the source of collision is different.

Case B.2 — $W_{r+1} \notin W^{[r]}$ and $W_j = W_i$ for some $i \in [r]$ and $j \in [r+2 \ldots m]$: This gives a new collision $\delta' = (W_{j-1}, W_{i-1}; W_i)$ which is independent of δ, whence it increases the accident.

Hence, both case B.1 and B.2 lead to contradiction. Thus, we have $W^{[r+1\ldots m]} \cap W^{(r]} = \emptyset$. Suppose $W^{[r\ldots m]}$ is not a path. Therefore, $\exists i, j \in [r \ldots m]$ such that $(W_i, W_j; W_{i+1})$ is a collision. Clearly, this collision is independent from δ (as $W^{[r+1\ldots m]} \cap W^{(r]} = \emptyset$), and hence gives a new accident.

Observe that $s = c - 1$ is a special case. In addition to this condition, suppose we have an edge $e := (W_{i_0-1}, W_{r+1})$. This creates a collision $\delta' = (W_{i_0-1}, W_r; W_{r+1})$ which is dependent on δ. Obviously, the edge e cannot occur in graph associated to a single message, as that will imply $\mathbf{deg_{in}}(W_j) \geq 2$ for some $j \in [0..i_0 - 1]$ which gives a new accident. But for two messages this is realizable, as exploited in the counter-examples given in Figure 3.2.3. We illustrate the structure graphs corresponding to this special case in Figure 3.3.2(b) and 3.3.2(e). □

We summarize the above discussion in the following lemma.

Lemma 3.3.2. *For $m \geq 1, M \in \mathbb{B}^m$, the graphs in figure 3.3.2 exhaust all possible forms for any $\mathcal{G} \in \mathsf{Struct}_1(M)$.*

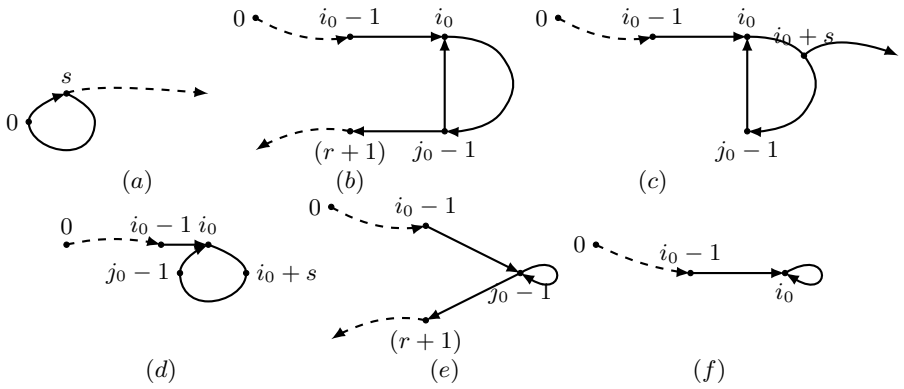

Figure 3.3.2: Characterizing all accident-one structure graphs realizable by a single message. The dashed lines in these illustrations represent optional subwalks. The vertex W_i is represented by i, for notational simplicity.

3.4 Revisiting $\mathsf{outCP}^{\mathsf{any}}_{q,\ell,\sigma}$ and $\mathsf{fullCP}^{\mathsf{pf}}_{q,\ell,\sigma}$ Bounds

In this section, we revise the upper bounds on outCP and fullCP, and consequently the PRF advantages of CBC-MAC and EMAC as given in [19]. We start off with a discussion that establishes the role of structure graphs in the PRF security analysis of CBC-MAC and EMAC. We have already seen that bounding PRF advantages of CBC-MAC and EMAC is reduced to bounding full collision probability $\mathsf{fullCP}^{\mathsf{pf}}_{q,\ell,\sigma}$ and collision probability $\mathsf{outCP}^{\mathsf{any}}_{q,\ell,\sigma}$, respectively. So, it would be sufficient to bound these probabilities. For this we first prove a general claim stated in Proposition 3.4.1.

STRUCTURE GRAPH EVENTS: Let $\mathtt{E}$ be an event defined on the intermediate output sequence $\mathfrak{v}^{\pi}(M^q)$ for some permutation π. We say that the event $\mathtt{E}$ is *defined by a structure graph* if there is an event $\mathtt{E}'$ defined on the structure graph Struct^{π} such that $\mathtt{E}$ holds if and only if $\mathtt{E}'$ holds. We call such an event a *structure graph event*. Moreover, we say that $\mathtt{E}$ is *non-free* if it does not hold for free graphs (the structure graph with accident 0). The collision event for distinct messages as well as the full collision event for prefix-free messages are examples of non-free structure graph events. We write $\mathsf{Struct}_a[\mathtt{E}]$ to denote the set of all structure graphs with a accidents and satisfying a non-free event $\mathtt{E}$.

Proposition 3.4.1. *Let $\mathtt{E}$ be a non-free structure graph event for the message tuple M^q. Then,*

$$\Pr_{\Pi}[\mathtt{E}] \leq \frac{|\mathsf{Struct}_1[\mathtt{E}]|}{2^n - m} + \frac{m^4}{2^{2n}}.$$

Proof. For any structure graph event $\mathtt{E}$,

$$\Pr_{\Pi}[\mathtt{E}] = \sum_{a \geq 0} \Pr\left[\mathsf{Struct}^{\Pi} \in \mathsf{Struct}_a[\mathtt{E}]\right]$$

$$= \sum_{\mathcal{G} \in \mathsf{Struct}_1[\mathbf{E}]} \Pr\left[\mathsf{Struct}^{\Pi} = \mathcal{G}\right] + \sum_{a \geq 2} \Pr\left[\mathsf{Struct}^{\Pi} \in \mathsf{Struct}_a\right]$$
$$\leq \frac{|\mathsf{Struct}_1[\mathbf{E}]|}{2^n - m} + \frac{m^4}{2^{2n}},$$

where the last inequality follows from Lemma 3.2.6 and Corollary 3.2.8. □

3.4.1 Revision of $\mathsf{outCP}^{\mathsf{any}}_{q,\ell,\sigma}$ Bound

Suppose $M_1 \in \mathbb{B}^{m_1}$, $M_2 \in \mathbb{B}^{m_2}$, and $0 \leq m_1 \leq m_2$. Also, we assume that $M_{(1,m_1)} \neq M_{(2,m_2)}$, since otherwise we can remove the largest common suffix which does not change the collision probability. Note that, the first message M_1 can be empty and in this case the collision event means $\mathfrak{v}^{\Pi}_{m_2}(M_2) = 0^n$. This is a structure graph event because $(1, 0)$ is a vertex of the structure graph. Due to Proposition 3.4.1, we only need to bound the number of structure graphs with accident 1 satisfying the `Ocoll` event for the pair of messages. More precisely, we have to bound the size of the set $\mathsf{Struct}_1(M_1, M_2)[\texttt{Ocoll}]$.

3.4.1.1 Case A: M_1 is Empty

In this case we have

$$\mathsf{Struct}_1(M_1, M_2)[\texttt{Ocoll}] = \mathsf{Struct}_1(M_2)[W_{(2,m_2)} = (1, 0)].$$

We make the following claim, which is essentially [19, Lemma 14] in our notations.

Claim 3.4.2. $|\mathsf{Struct}_1(M_2)[W_{(2,m_2)} = (1, 0)]| \leq d(m_2)$, *where* d *denotes the number of divisors function.*

Proof. Let i be the smallest positive integer such that $W_{(2,i)} = 0$. Let A be the label of the cycle $W_2^{(i]}$. If $M_2 = A^k$ for some $k \in \mathbb{N}$, then $\mathsf{Struct}_1(M_2)[W_{(2,i)} = (1, 0)]$ contains exactly one structure graph. Note that, k must divide m_2 and hence the number of possible choices of k is at most $d(m_2)$, the number of divisors of m_2. Suppose $M_2 = A^k \| B$ for some non-empty B where k is the largest such integer. If B is a prefix of A then $W_{(2,m_2)} = (1, 0)$ only if $B = A$ which contradicts the maximality of k. So now assume that $\mathsf{lcp}(A, B) = s$ where s is strictly less than the block length of B. Then, we must have $A_{s+1} \neq B_{s+1}$, whence $W_{(2,ki+s+1)} \neq W_{2,s+1}$. But, $W_{2,m_2} = (1, 0)$, whence the walk must collide with the cycle $W_2^{(i]}$ at some point beyond $W_{2,s+1}$. However, as we have a zero-output structure graph with accident 1, we can not have any more collisions. Thus we can have at most $d(m_2)$ accident 1 graphs satisfying the event $W_{(2,m_2)} = (1, 0)$. □

3.4.1.2 Case B: M_1 is Non-empty

In this case, we must have a collision $(u := W_{(1,m_1-1)}, v := W_{(2,m_2-1)}; z := W_{(2,m_2)})$, as the labels of the last edges for walks W_1 and W_2 are different. Any other collision must have the same source set $\{u, v\}$ (otherwise, the number of accidents increases). Moreover, $(1, 0)$ can not have positive in-degree. Now, we consider different sub-cases:

Case B.1 — Both W_1 and W_2 are paths: The union of $W_1^{(m_1-1]}$ and $W_2^{(m_2-1]}$ is a free graph (as $W_{(1,m_1-1)}$ and $W_{(2,m_2-1)}$ can not appear before in the graph and so no collision among the path can occur). This gives only one choice of the graph as shown in the Figure 3.4.1(a). So the number of choices is bounded by at most 1. This is also proved as a part of [19, Lemma 15].

Case B.2 — W_2 is not a path: We have already characterized all possibilities of W_2 (in Figure 3.3.2). There exist some integers t and c such that $W_2^{(t]}$ is a path with $W_{(2,t-1)} = u$ and $W_{(2,t)} = w$, $W_2^{[t\ldots t+c]}$ is a cycle of length c such that $W_{(2,t+c-1)} = v$. Recall that, $W_{(1,m_1-1)} = u$ and $W_{(2,m_2-1)} = v$.

Claim 3.4.3. $W_1^{(m_1-1]} = W_2^{(t-1]}$ *and so* $m_1 = t$.

Proof. Let s be the length of the largest common prefix of $M_1^{[t-1]}$ and $M_2^{[t-1]}$. If $s < t-1$ then in the walk W_1 there is no way to reach u without coming back to the walk $W_2^{(t-1]}$. Coming back is not possible as it leads to a collision with a different generator set. Similarly, we can disprove that $s = t-1$ and $m_1 > t$. Thus, we have $m_1 = t$ and $W_1^{(m_1-1]} = W_2^{(t-1]}$. □

Now, based on $p = \mathsf{lcp}(M_1, M_2)$, we may have two sub-cases.

> Case B.2.(a) — $W_{(1,p)} = w$: In this case M_1 is a prefix of M_2, i.e. $t = p$. The structure graph corresponding to this is illustrated in Figure 3.4.1(b). The number of such structure graphs is again at most $d(m_2 - m_1)$ (similar to the previous case where M_1 is the empty message). This is also shown in [19, Lemma 13].
>
> Case B.2.(b) — $W_1[p] \neq w$: This is the missing case in [19]. In this case, $W_{1,p}$ should be a fresh vertex otherwise we get a collision with different source set. The structure graph corresponding to this case is shown in Figure 3.4.1(c). Let $M_1 = A\|a$ where $A = M_1^{[t-1]}$ and $a = M_{(1,t)}$. Note that, $t - 1 = p$ is the length of the largest common prefix of M_1 and M_2. Then,
>
> $$M_2 = A\|b\|(B\|e)^{c-1}\|B\|f, \text{ where } f = M_{(2,m_2)}, b = M_{(2,t)}, e = a \oplus b \oplus f.$$

The choice of B is variable. But, it must satisfy the above restrictions for some $c > 1$. In fact, B is uniquely determined by c (as it occurs exactly c many times, and e and f are fixed). Now, c must divide $m_2 - m_1$ and hence the number of choices of c is at most $d(m_2 - m_1) - 1$.

This completes the characterization of all accident-1 structure graphs satisfying `Ocoll` event and we get explicit upper bounds on the number of such graphs for all cases. Note that, cases B.2.(a) and B.2.(b) cannot hold simultaneously. But, Cases B.2.(b) and B.1 can hold simultaneously, which makes the total count of these two cases at most $d(m_2 - m_1)$. Since the ordering of messages does not matter in `Ocoll` we are done.

Lemma 3.4.4. *For $m_1 \leq m_2 \in \mathbb{N}$, $M_1 \in \mathbb{B}^{m_1}$, $M_2 \in \mathbb{B}^{m_2}$, and $M_1 \neq M_2$, we have*

1. *If $\mathsf{lcp}(M_1, M_2) = m_1$, then any $\mathcal{G} \in \mathsf{Struct}_1(M_1, M_2)[\texttt{Ocoll}]$ is of the form illustrated in Figure 3.4.1(b) and the number of such graphs is at most $d'(m_2)$ (taking the maximum over all possible $d(\cdot)$ values).*
2. *If $\mathsf{lcp}(M_1, M_2) = m_1 - 1$, then any $\mathcal{G} \in \mathsf{Struct}_1(M_1, M_2)[\texttt{Ocoll}]$ is of the form illustrated in Figure 3.4.1(c) and the number of such graphs is at most $d'(m_2)$ (taking the maximum over all possible $d(\cdot)$ values).*
3. *In all other cases, any $\mathcal{G} \in \mathsf{Struct}_1(M_1, M_2)[\texttt{Ocoll}]$ is of the form illustrated in Figure 3.4.1(a) and the number of such graphs is at most 1.*

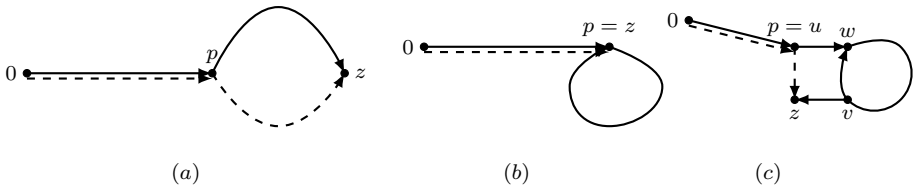

Figure 3.4.1: Characterizing all accident-one structure graphs realizable by two messages which satisfy the `Ocoll` event. Dashed lines represent W_1 and solid lines represent W_2, and $*$ denote some arbitrary string.

Lemma 3.4.4 and Proposition 3.4.1 straightaway bound $\mathsf{outCP}(M_i, M_j) \leq d'(\ell)/2^n + (m_i + m_j)^4/2^{2n}$ for any $i < j \in [q]$. Further, there exist a message tuple M^q such that $\mathsf{outCP}^{\mathsf{any}}_{q,\ell,\sigma} \leq \sum_{i<j\in[q]} \mathsf{outCP}(M_i, M_j)$, which gives

Corollary 3.4.5. $\mathsf{outCP}^{\mathsf{any}}_{q,\ell,\sigma} \leq \dfrac{q^2 d'(\ell)}{2^n} + \dfrac{8\sigma q \ell^3}{2^{2n}}$.

Corollary 3.4.5 in combination with Eq. (3.3) and (3.5) gives the revised PRF security advantage for EMAC,

$$\mathbf{Adv}^{\mathsf{prf}}_{\mathsf{EMAC}_{\mathsf{E},\mathsf{E}}}(q, \ell, \sigma, t) \leq 2\mathbf{Adv}^{\mathsf{prp}}_{\mathsf{E}}(\sigma, t) + \frac{q^2 d'(\ell)}{2^n} + \frac{8\sigma q \ell^3}{2^{2n}}. \tag{3.13}$$

3.4.2 Revision of $\mathsf{fullCP}^{\mathsf{pf}}_{q,\ell,\sigma}$ Bound

Since Fcoll is a non-free structure graph event, we have, using Proposition 3.4.1,

$$\mathsf{fullCP}(M_1, M_2) \leq \frac{|\mathsf{Struct}_1(M_1, M_2)[\texttt{Fcoll}]|}{2^n - m_1 - m_2} + \frac{(m_1 + m_2)^4}{2^{2n}}, \tag{3.14}$$

where M_1 and M_2 satisfy the pf (prefix-free) constraint. Thus, it would be again sufficient to bound the number of structure graphs for two messages with accident 1 which satisfy full collision event. Bellare et al. [19] proved that $|\mathsf{Struct}_1[\texttt{Fcoll}]| \leq 4\max\{m_1, m_2\}$. While bounding the $|\mathsf{Struct}_1[\texttt{Fcoll}]|$, they proved a strong result [19, Lemma 19] that will be useful in our analysis also. We reproduce it here in our notations.

Lemma 3.4.6. *For $b \in \{1, 2\}$ and any $i \in (m_b]$, we have*

$$|\mathsf{Struct}_1(M_1, M_2)[W_{(b,i)} \in W_b^{(i-1]\cup[i+1\ldots m_b]}]| \leq m_b.$$

We skip the proof of lemma 3.4.6 as it is available in [19]. We revise the upper bound on $|\mathsf{Struct}_1(M_1, M_2)[\texttt{Fcoll}]|$ to $3(m_1 + m_2)$, and consequently the new bound on fullCP is as follows

Lemma 3.4.7. $\mathsf{fullCP}(M_1, M_2) \leq \frac{3(m_1+m_2)}{2^n - m_1 - m_2} + \frac{(m_1+m_2)^4}{2^{2n}}$.

Proof. We need to bound the number of structure graphs for a pair of prefix-free messages $M_1 \in \mathbb{B}^{m_1}$ and $M_2 \in \mathbb{B}^{m_2}$ which satisfy the Fcoll event and have at most accident 1. Note that, the event implies that the structure graphs must have at least accident 1 as the messages are prefix-free. The event Fcoll can be written as $W_{(2,m_2)} \in W_2^{(m_2-1]} \cup W_1^{[m_1]}$.

3.4.2.1 Case A: $W_{(2,m_2)} \in W_2^{(m_2-1]}$

This case can be bounded to at most m_2, by direct application of Lemma 3.4.6.

3.4.2.2 Case B: $W_{(2,m_2)} \in W_1^{[m_1]}$

Suppose $\texttt{Fcoll}(M_1, M_2)$ happens due to $W_{(2,m_2)} = W_{(1,r)}$ for an arbitrary $r \in [m_1 - 1]$. Then $\texttt{Fcoll}(M_1, M_2)$ is equivalent to $\texttt{Ocoll}(M_1^{[r]}, M_2)$. Let $s := \mathsf{lcs}(M_1^{[r]}; M_2)$. Then $M_{(1,r-s)} \neq M_{(2,m_2-s)}$. Let $N_1 = M_1^{[r-s]}$ and $N_2 = M_2^{[m_2-s]}$. From lemma 3.4.4, we know

that $\widetilde{\mathcal{G}} \in \mathsf{Struct}_1(N_1, N_2)[\texttt{Ocoll}]$ must be one of (a), (b) or (c) in Figure 3.4.1. Keep in mind that $\widetilde{\mathcal{G}}$ is a subgraph of some $\mathcal{G} \in \mathsf{Struct}_1(M_1, M_2)[\texttt{Fcoll}]$.

Case B.1 — $\widetilde{\mathcal{G}}$ is of the form Figure 3.4.1(a): In this case $\widetilde{W}_1$ and $\widetilde{W}_2$ are paths. For a fixed r the only possible collision is at $(\widetilde{W}_{(1,r-s-1)}, \widetilde{W}_{(2,m_2-s-1)}; \widetilde{W}_{(1,r-s)})$, and hence the number of such graphs is at most 1. There are at most m_1 possible values for r. So, the number of choices for $\mathcal{G} \in \mathsf{Struct}_1(M_1, M_2)[\texttt{Fcoll}]$ is at most m_1.

Case B.2 — $\widetilde{\mathcal{G}}$ is not of the form Figure 3.4.1(a): In this case, at least one of $\widetilde{W}_1$ or $\widetilde{W}_2$ is not a path. Without loss of generality, assume $\widetilde{W}_1$ is not a path. Let $\widetilde{p} = \mathsf{lcp}(N_1, N_2)$. We know that N_1 and N_2 are prefixes of M_1 and M_2, respectively. Thus $M_1^{[\widetilde{p}]} = M_2^{[\widetilde{p}]}$. Now, we must have a collision $(u, v; w)$ in $\widetilde{W}_1$. From Lemma 3.4.4, we know that the graph can be either (b) or (c) in Figure 3.4.1 depending on whether $w = \widetilde{W}_{(1,\widetilde{p})}$ or $w = \widetilde{W}_{(1,\widetilde{p}+1)}$. Next, we make two claims which will enable us to bound these two cases. The proofs for these two claims are given later in the section.

Claim 3.4.8. *If $\widetilde{\mathcal{G}}$ is of the form Figure 3.4.1(b), then $W_{(1,\mathsf{lcp}(M_1,M_2))}$ is not fresh in W_1.*

Claim 3.4.9. *If $\widetilde{\mathcal{G}}$ is of the form Figure 3.4.1(c), then $W_{(1,\mathsf{lcp}(M_1,M_2)+1)}$ is not fresh in W_1.*

Recall that in a walk W a vertex W_i is not fresh if $\exists j \neq i$ such that $W_j = W_i$. By Claim 3.4.8, we know that $W_{(1,\mathsf{lcp}(M_1,M_2))}$ is not fresh when $\widetilde{\mathcal{G}}$ is as in Figure 3.4.1(b). Similarly, by Claim 3.4.9, we know that $W_{(1,\mathsf{lcp}(M_1,M_2)+1)}$ is not fresh when $\widetilde{\mathcal{G}}$ is as in Figure 3.4.1(c). So using Lemma 3.4.6, we bound the number of such graphs $\mathcal{G}$ to at most $2m_1$ (at most m_1 graphs for each of the two cases) when $\widetilde{W}_1$ is not a path. Similarly we have at most $2m_2$ choices when $\widetilde{W}_2$ is not a path. Therefore the total number of choices in case B.2 is at most $2(m_1 + m_2)$. Combining cases A, B.1 and B.2 we have at most $3(m_1 + m_2)$ choices. The result follows. □

3.4.2.3 Proof of Claim 3.4.8

If $\widetilde{\mathcal{G}}$ is like Figure 3.4.1(b), we must have $w = \widetilde{W}_{(1,\widetilde{p})}$. Let p be the smallest integer such that $\widetilde{W}_{(1,p)} = \widetilde{W}_{(1,\widetilde{p})}$. Let P and Q be the labels of $\widetilde{W}_1^{[\widetilde{p}]}$ and $\widetilde{W}_1^{[\widetilde{p}\ldots p]}$, respectively, and $c = p - \widetilde{p}$. Then, $N_1 = P\|Q^i$ and $N_2 = P$ for some $i > 0$. As N_1 and N_2 are formed by removing the largest common suffix from of $M_1^{[r]}$ and M_2, respectively, therefore $M_1^{[r]} = P\|Q^{i'}\|R$ and $M_2 = P\|Q^j\|R$, where $i' \geq i$ and $j \geq 0$ are the largest such integers. Since $M_1^{[r]}$ and M_2 are prefix-free, we must have $i' > j$. Now, $M_1 = M_1^{[r]}\|S = P\|Q^{i'}\|R\|S$, where $|S| \geq 0$. From now on, we will work on the walk W_1 of graph $\mathcal{G}$ (instead of $\widetilde{W}_1$ which is a subwalk of W_1) corresponding to M_1. If R is a prefix of Q then M_2 must be a prefix of M_1, which contradicts the prefix-free constraint. So R is not a prefix of Q. If Q is a prefix of R then it contradicts the maximality of i' and j. So Q is not a

prefix of R. Let $s = \mathsf{lcp}(Q, R)$. Then, $Q_{s+1} \neq R_{s+1}$. Thus, $M_1^{[\widetilde{p}+jc+s]} = M_2^{[\widetilde{p}+jc+s]}$ and $M_{(1,\widetilde{p}+jc+s+1)} \neq M_{(2,\widetilde{p}+jc+s+1)}$. So, $\mathsf{lcp}(M_1, M_2) = \widetilde{p} + jc + s$. Further, since $j < i'$, we must have $W_{(1,\widetilde{p}+i'c+s)} = W_{(1,\widetilde{p}+jc+s)}$. Thus, $W_{(1,\mathsf{lcp}(M_1,M_2))}$ is not fresh. Since, we started off with an arbitrary r, we conclude that $W_{(1,\mathsf{lcp}(M_1,M_2))}$ is not fresh whenever $\widetilde{\mathcal{G}}$ is of the form Figure 3.4.1(b). □

3.4.2.4 Proof for Claim 3.4.9

If $\widetilde{\mathcal{G}}$ is like Figure 3.4.1(c), we must have $w = \widetilde{W}_1[\widetilde{p}+1]$. As noted earlier, in the revision of the outCP bound, this case was missing in the proof in [19]. Using a similar line of argument as in the proof of Claim 3.4.8, we can conclude that irrespective of the value of r, the cycle goes through $W_{(1,\mathsf{lcp}(M_1,M_2)+1)}$ twice. Thus, $W_{(1,\mathsf{lcp}(M_1,M_2)+1)}$ is not fresh. □

Note that, our approach in Case B.2 above is a bit subtle. We used lemma 3.4.4 to identify a fundamental property (cycle goes through either $\mathsf{lcp}(M_1, M_2)$ or $\mathsf{lcp}(M_1, M_2) + 1$ twice), and then exploited this property to bound the counting. A straightforward approach of summing the number of graphs in Figure 3.4.1(b) and 3.4.1(c) over all values of r will give a worse bound of $m_b d'(m_b)$ for $b \in \{1, 2\}$. To get a tighter bound of m_b we needed this subtlety. Now, we extend the bound for $\mathsf{fullCP}^{\mathsf{pf}}(M_1, M_2)$ to $\mathsf{fullCP}^{\mathsf{pf}}_{q,\ell,\sigma}$, in order to get the revised PRF bound for CBC-MAC.

$$\begin{aligned}
\mathsf{fullCP}^{\mathsf{pf}}_{q,\ell,\sigma} &\leq \sum_{i\neq j\in[q]} \mathsf{fullCP}^{\mathsf{pf}}(M_i; M_j) \\
&\leq \sum_{i\neq j\in[q]} \left(\frac{3(m_i + m_j)}{2^n - m_1 - m_2} + \frac{(m_i + m_j)^4}{2^{2n}} \right) \\
&\leq \sum_{i\neq j\in[q]} \left(\frac{6(m_i + m_j)}{2^n} + \frac{(m_i + m_j)^4}{2^{2n}} \right) \\
&\leq \frac{6\sigma q}{2^n} + \frac{8\sigma q \ell^3}{2^{2n}} \qquad (3.15)
\end{aligned}$$

Here we have computed the bound in terms of q, ℓ and σ. Another approach (as used in [19]) is to bound the value using q and ℓ only, in which case the bound will be

$$\mathsf{fullCP}^{\mathsf{pf}}_{q,\ell,\sigma} \leq \frac{12q^2\ell}{2^n} + \frac{16q^2\ell^4}{2^{2n}}$$

Using Proposition 3.1.2, and Eq. 3.2 and 3.15, we get the following result.

Theorem 3.4.10. *For prefix-free messages, we have*

$$\mathbf{Adv}^{\mathsf{prf}}_{\mathsf{CBC\text{-}MAC_E}}(q,\ell,\sigma,t) \leq \mathbf{Adv}^{\mathsf{prp}}_{\mathsf{E}}(\sigma,t) + \frac{8\sigma q}{2^n} + \frac{8\sigma q\ell^3}{2^{2n}} + \frac{q^2}{2^{n+1}}. \tag{3.16}$$

This gives a bound of $O(\frac{\sigma q}{2^n})$ for $\ell < 2^{n/3}$. As noted earlier, this is a better bound whenever the average message length is much smaller than the length of the longest message.

3.5 Revised Security Analysis of EMAC

In this section, we revisit the improved PRF analysis of EMAC due to Pietrzak [163]. We first identify and rectify the flaw in the proof and obtain a degraded security bounds. Later, we obtain a better bound (in terms of ℓ) using a much simpler proof.

3.5.1 Flaw and Revision of PRF Advantage of EMAC [163]

For brevity, in this section we will use asymptotic bounds and avoid exact constant factors. The proposed bound for $\mathsf{EMAC}_{\Pi_1,\Pi_2}$ as stated in [163] is

$$\mathbf{Adv}^{\mathsf{prf}}_{\mathsf{EMAC}_{\Pi_1,\Pi_2}}(q,\ell,\sigma) = O\left(\frac{q^2}{2^n}\left(1+\frac{\ell^8}{2^n}\right)\right),$$

provided $\ell^2 \leq q$. Thus, the bound becomes $O(q^2/2^n)$ when $\ell \leq \min\{q^{1/2}, 2^{n/8}\}$.

PIETRZAK'S APPROACH: To show the above result, we need to bound the collision probability $\mathsf{outCP}_{q,\ell}$. One possible approach is to group the q message into $O(q/\ell^2)$ sets, each set consisting of about ℓ^2 messages. Now, the collision event for q messages implies that a collision occurred in the union of two sets. Since, `Ocoll` is a non-free event, Proposition 3.4.1 gives

$$\mathsf{outCP}_{k,\ell} = O\left(\frac{|\mathsf{Struct}_1(M^k)[\texttt{Ocoll}]|}{2^n}\right) + O\left(\frac{k^4\ell^4}{2^{2n}}\right),$$

for some $k \leq q$. Applying this with $k = \ell^2$ (i.e. messages within two sets), we have

$$\mathsf{outCP}_{q,\ell} = O\left(\frac{q^2}{\ell^4}\right) \times \mathsf{outCP}_{2\ell^2,\ell} = O\left(\frac{q^2N}{\ell^4 2^n}\right) + O\left(\frac{q^2\ell^8}{2^{2n}}\right)$$

where N denotes the number of accident-one structure graphs satisfying `Ocoll` for ℓ^2 messages with maximum length ℓ. The $O(q^2/\ell^4)$ term is due to the number of ways in

which we can choose two sets. In [163, Lemma 4], Pietrzak claimed that $N = O(\ell^4)$. More precisely, he showed that

$$N = \ell^4 \max_{\mathsf{lcp}(M_1,M_2)<m_1} |\mathsf{Struct}_1(M, M')[\mathtt{0coll}]| + \ell^4. \tag{3.17}$$

where the maximum is taken over all (M_1, M_2) such that $m_1 < m_2$. To get a bound in $O(\ell^4)$ Pietrzak made the following claim.

Claim 1 of [163]: If $\mathsf{lcp}(M_1, M_2) \neq m_1$, then $|\mathsf{Struct}_1(M_1, M_2)[\mathtt{0coll}]| = 1$.

If this claim happens to be true then $N = O(\ell^4)$. However, we have seen before that there exists M_1, M_2 such that $\mathsf{lcp}(M_1, M_2) = m_1 - 1$ (see Figure 3.4.1(c)) and $|\mathsf{Struct}_1(M, M')[\mathtt{0coll}]| = d'(\ell)$. Thus,

$$\max_{\mathsf{lcp}(M_1,M_2)<m_1} |\mathsf{Struct}_1(M, M')[\mathtt{0coll}]| = O(d'(\ell)).$$

If we plug in this, the modified bound is $N = O(\ell^4(d'(\ell)))$, whence the correct bound for collision probability becomes $O(q^2 d'(\ell)/2^n)$ which is not better than the bound in [19]. In fact, this is worse as the maximum length is restricted to $2^{n/8}$ as compared to a more relaxed bound of $2^{n/3}$ in [19].

3.5.2 Simple Proof of PRF Security for EMAC

We have seen in the last subsection that the influence of the flaw from [19, Lemma 10] is more serious in context of the tight bound analysis of EMAC given in [163]. One possible approach to fix the proof of [163], is by bounding N in a different way. For example, we can consider two cases: $\mathsf{lcp}(M_1, M_2) = m_1$, and $\mathsf{lcp}(M_1, M_2) = m_1 - 1$. For other pairs of messages which do not satisfy any of the two cases, the number of structure graphs can be shown to be one. However, we have to show that the the number of graphs is still at most ℓ^4 (see the second term of Eq. (3.17)).

We take a different and simpler approach. Instead of grouping the messages, we directly bound the number of structure graphs for a slightly different choices of permutations. We will ignore all those permutations (i.e. bad permutations) which induces one of the following:

1. For some pair of messages M_i and M_j the number of accident is two or more.
2. For some message M_i, the accident is one.

Let $\mathcal{G}$ be a structure graph associated to a q-tuple of messages. Recall that $\mathcal{G}$ is a union of q walks W_i. We write $\mathcal{G}_i$ and $\mathcal{G}_{i,j}$ to represent W_i and $W_i \cup W_j$. Note that, these are again structure graphs associated to M_i and (M_i, M_j) respectively. In this notation, all structure graphs generated by a good permutation must have two properties: for any $i \neq j \in [q]$, $\mathsf{Acc}(\mathcal{G}_i) = 0$ and $\mathsf{Acc}(\mathcal{G}_{i,j}) \leq 1$. We write $\mathtt{Bad}$ to denote the event that the structure graph is bad.

Lemma 3.5.1. *For any q-tuple of messages M^q and $\ell < 2^{n-2}$, we have*

$$\mathsf{outCP}_{q,\ell}(M^q) \leq \frac{q^2}{2^n} + \frac{q\ell^2}{2^n} + \frac{8q^2\ell^4}{2^{2n}}.$$

PROOF OVERVIEW — We want to bound the probability of realizing a structure graph for a pair of messages that satisfies $\mathtt{Ocoll}$. The main idea is to simplify the analysis by handling graphs in $\mathsf{Struct}(M^q)[\mathtt{Bad}]$ separately. Lets characterize the nature of any structure graph for a pair of messages that satisfies $\neg\mathtt{Bad} \wedge \mathtt{Ocoll}$. The number of accidents in the graph is at most 1, the individual walks corresponding to each message must be paths, and the two paths must share common first and last vertices. Now, the number of such graphs can be at most 1 (see Figure 3.4.1), leading to a probability bound of $q^2/2^n$ (due to $O(q^2)$ pairs of messages).

Proof. We first bound the probability that $\mathsf{Struct}^{\sqcap}(M^q) \in \mathsf{Struct}(M^q)[\mathtt{Bad}]$. For a bad graph (1) there exists i and j such that the accident for the pair of message M_i and M_j is at least 2, or (2) there exists i, such that the accident for M_i is at least one. The first event can happen with probability $8q^2\ell^4/2^{2n}$ using Corollary 3.2.8. Similarly, the second event can happen with $q\ell^2/2^n$. Now, we bound the probability that $\mathsf{Struct}^{\sqcap}(M^q) \in \mathsf{Struct}(M^q)[\mathtt{Ocoll} \wedge \neg\mathtt{Bad}]$. Note that, the collision event implies that there exists i and j such that collision event holds for the message M_i and M_j. Now $\neg\mathtt{Bad}$ implies that accident of $\mathcal{G}_{i,j}$ is one whereas accident of $\mathcal{G}_i$ and $\mathcal{G}_j$ are zero. In section 3.3, we have characterized all structure graphs for a pair of messages with accident one satisfying collision. Among all possibilities only one structure graph satisfies $\neg\mathtt{Bad}$. Hence, there is exactly one structure graph. This implies that

$$\Pr\left[\mathsf{Struct}^{\sqcap}(M_i, M_j) \in \mathsf{Struct}(M_i, M_j)[\mathtt{Ocoll} \wedge \neg\mathtt{Bad}]\right] \leq \frac{2}{2^n},$$

where we use the assumption that $m_i + m_j \leq 2\ell < 2^{n-1}$. By summing over all possible i and j, we have

$$\Pr\left[\mathsf{Struct}^{\sqcap}(M^q) \in \mathsf{Struct}(M^q)[\mathtt{Ocoll} \wedge \neg\mathtt{Bad}]\right] \leq \frac{q^2}{2^n}.$$

Now, we summarize the above discussion as follows

$$\begin{aligned}\mathsf{outCP}_{q,\ell}(M^q) &\leq \Pr\left[\mathsf{Struct}^{\Pi}(M^q) \in \mathsf{Struct}(M^q)[\mathtt{Ocoll} \wedge \neg\mathtt{Bad}]\right] \\ &\quad + \Pr\left[\mathsf{Struct}^{\Pi}(M^q) \in \mathsf{Struct}^{\Pi}(M^q)[\mathtt{Ocoll} \wedge \mathtt{Bad}]\right] \\ &\leq \sum_{i<j} \frac{2|\mathsf{Struct}(M_i, M_j)[\mathtt{Ocoll} \wedge \neg\mathtt{Bad}]|}{2^n} + \frac{q\ell^2}{2^n} + \frac{8q^2\ell^4}{2^{2n}} \\ &\leq \frac{q^2}{2^n} + \frac{q\ell^2}{2^n} + \frac{8q^2\ell^4}{2^{2n}}. \end{aligned} \tag{3.18}$$

This completes the proof. □

Using Eq. (3.3) and (3.5), and Lemma 3.5.1, we get the following result on the PRF security of EMAC.

Theorem 3.5.2. *For $\ell < 2^{n-2}$, we have*

$$\mathbf{Adv}^{\mathsf{prf}}_{\mathsf{EMAC}_{\mathsf{E},\mathsf{E}}}(q,\ell,\sigma,t) \leq 2\mathbf{Adv}^{\mathsf{prp}}_{\mathsf{E}}(\sigma,t) + \frac{1.5q^2}{2^n} + \frac{q\ell^2}{2^n} + \frac{8q^2\ell^4}{2^{2n}}.$$

Thus, for $\ell \leq \min\{q^{1/2}, 2^{n/4}\}$, we have

$$\mathbf{Adv}^{\mathsf{prf}}_{\mathsf{EMAC}_{\mathsf{E},\mathsf{E}}}(q,\ell,\sigma,t) \leq 2\mathbf{Adv}^{\mathsf{prp}}_{\mathsf{E}}(\sigma,t) + O\left(\frac{q^2}{2^n}\right).$$

Clearly, our result is a marked improvement over both [19] and [163]. The result in Theorem 3.5.2 is tight for $\ell \leq \min\{q^{1/2}, 2^{n/4}\}$, as one can obtain collisions in the output of EMAC in roughly $2^{n/2}$ queries, which leads to a easy length extension distinguishing event. The condition $q > \ell^2$ can be dropped if we assume that $\ell \leq 2^{n/4-k}$ for some small k such that 2^{-k} is negligible. More precisely, if $\ell \leq 2^{n/4-k}$, then the PRF advantage of EMAC is $O(q^2/2^n) + O(1/2^k)$.

3.5.3 Implications on the PRF Security of ECBC and FCBC

For any $M \in \{0,1\}^{\eta^*}$, recall that $\overline{M} = \mathsf{pad}(M) = M\|10^{n|M|_n-|M|-1}$ when $n \nmid |M|$, and $\overline{M} = \mathsf{pad}(M) = M$ otherwise. In [34], Black and Rogaway gave three variants for CBC-MAC, namely ECBC, FCBC, and XCBC, to handle arbitrary message lengths. Here, we describe ECBC and FCBC.

1. ECBC, based on a block cipher E instantiated thrice with keys $K_1, K_2, K_3 \in \mathcal{K}$, is defined as,

$$\mathsf{ECBC}_{\mathsf{E}_{K_1},\mathsf{E}_{K_2},\mathsf{E}_{K_3}}(M) := \begin{cases} \mathsf{E}_{K_2}(\mathsf{CBC}_{\mathsf{E}_{K_1}}(\overline{M})) & \text{if } |\overline{M}| = |M|, \\ \mathsf{E}_{K_3}(\mathsf{CBC}_{\mathsf{E}_{K_1}}(\overline{M})) & \text{otherwise.} \end{cases}$$

2. FCBC, based on a block cipher E instantiated thrice with keys $K_1, K_2, K_3 \in \mathcal{K}$, is defined as,

$$\mathsf{FCBC}_{\mathsf{E}_{K_1},\mathsf{E}_{K_2},\mathsf{E}_{K_3}}(M) := \begin{cases} \mathsf{E}_{K_2}(\mathsf{CBC}_{\mathsf{E}_{K_1}}(\overline{M}^{[m-1]}) \oplus \overline{M}_m) & \text{if } |\overline{M}| = |M|, \\ \mathsf{E}_{K_3}(\mathsf{CBC}_{\mathsf{E}_{K_1}}(\overline{M}^{[m-1]}) \oplus \overline{M}_m) & \text{otherwise,} \end{cases}$$

where $m = \left\lceil \frac{|M|}{n} \right\rceil$.

It should be clear that the two keys K_2 and K_3 are chiefly used to separate the processing of block sequences (key K_2 is used for finalization) from non-block sequences (key K_3 is used for finalization). If we instantiate the block cipher E with $\mathsf{K}^3 \leftarrow\!\!\$\, \mathcal{K}^3$, then clearly the finalization of block and non-block sequences is independent from each other. This observation coupled with the fact that `Ocoll` implies `Icoll` gives the following corollary on the security of ECBC and FCBC

Corollary 3.5.3. *For $\ell < 2^{n-2}$, we have*

1. $\mathbf{Adv}^{\mathsf{prf}}_{\mathsf{ECBC}_{\mathsf{E},\mathsf{E},\mathsf{E}}}(q,\ell,\sigma,t) \leq 3\mathbf{Adv}^{\mathsf{prp}}_{\mathsf{E}}(\sigma,t) + \dfrac{1.5q^2}{2^n} + \dfrac{q\ell^2}{2^n} + \dfrac{8q^2\ell^4}{2^{2n}}.$
2. $\mathbf{Adv}^{\mathsf{prf}}_{\mathsf{FCBC}_{\mathsf{E},\mathsf{E},\mathsf{E}}}(q,\ell,\sigma,t) \leq 3\mathbf{Adv}^{\mathsf{prp}}_{\mathsf{E}}(\sigma,t) + \dfrac{1.5q^2}{2^n} + \dfrac{q\ell^2}{2^n} + \dfrac{8q^2\ell^4}{2^{2n}}.$

The proof is quite straightforward given Theorem 3.5.2 and the above mentioned observations. We remark that the bounds in Corollary 3.5.3 are significant improvements over the applicable state-of-the-art bounds in [19, 163].

Chapter 4

Pseudorandom Function based CBC-MAC Family

In this chapter, we continue our investigations on the security of the CBC-MAC family. In all the previous works the underlying primitive of the CBC-MAC family was assumed to be a block cipher or a keyed family of random permutations. However, the CBC mode of operation can also be used to construct a MAC when the underlying primitive is a length-preserving pseudorandom function (PRF) or a keyed family of length-preserving random functions. In fact, the very first security proof of CBC-MAC in [15] essentially models the block cipher as a PRF. We believe, when it comes to CBC-MAC the choice of block ciphers or pseudorandom permutations (PRPs) over PRFs as the underlying primitive is primarily for historical reasons, i.e. fast, secure, standardized block cipher implementations are readily available compared to pseudorandom functions.

Given a lack of understanding of the exact security of PRF-based CBC-MAC constructions, the primary goal of this chapter is to understand the gap between PRP vs PRF instantiation of CBC-MAC and its derivative constructions.

4.1 PRF Analysis of PRF-based CBC-MAC Family

As before, we will be interested in the PRF security of CBC-MAC family. Most of the notations and terminologies used in this chapter are analogous (replace $\pi \in \mathsf{Perm}$ with $\gamma \in \mathsf{Func}$) to the ones defined in chapter 3. So, sometimes we will directly use these notations and terminologies without first explicitly defining them.

4.1.1 Function-based CBC-MAC Family

In this chapter, we always use length-preserving functions γ over the set $\mathbb{B}$, i.e., $\gamma \in \mathsf{Func}(\mathbb{B},\mathbb{B})$, and hence skip the parametrization in Func. The CBC function with a function $\gamma \in \mathsf{Func}$, viewed as a key of the construction, is defined analogous to CBC_π of section 3.1.1. In other words, CBC_γ takes as input a message $M \in \mathbb{B}^m$ with m blocks and outputs $\mathsf{CBC}_\gamma(M) := \mathfrak{v}^\gamma_m$. This is inductively computed as follows: $\mathfrak{v}^\gamma_0(M) = 0^n$, and for all $i \in [m]$, we have

$$\begin{aligned} \mathfrak{u}^\gamma_i(M) &:= \mathfrak{v}^\gamma_{i-1}(M) \oplus M_i, \\ \mathfrak{v}^\gamma_i(M) &:= \gamma(\mathfrak{u}^\gamma_i(M)). \end{aligned} \tag{4.1}$$

As before, we call $\mathfrak{u}^\gamma(M) := (\mathfrak{u}^\gamma_i(M))_{i\in[m]}$ and $\mathfrak{v}^\gamma(M) := (\mathfrak{v}^\gamma_i(M))_{i\in(m]}$, the *intermediate input* and *intermediate output* vectors, respectively, associated to γ and M. The intermediate input vector $\mathfrak{u}^\gamma(M)$ is uniquely determined by $\mathfrak{v}^\gamma(M)$ (and does not depend on the function γ). We can write down this association generically as a function $\mathsf{out2in}_M : \mathbb{B}^{m+1} \to \mathbb{B}^m$ mapping any block vector y^{m+1} to another block vector x^m where $x_i = y_{i-1} \oplus M_i$ for all $i \in [m]$. So for all functions $\gamma \in \mathsf{Func}$, we have $\mathsf{out2in}_M(\mathfrak{v}^\gamma(M)) = \mathfrak{u}^\gamma(M)$. *We drop the superscripts γ and M when they are obvious from the context.*

Given the definition of CBC_γ, one can easily define the CBC-MAC family. In the following description, $K^3 \in \mathcal{K}^3$, $\mathsf{F} \in \mathsf{BFunc}(\mathcal{K},\mathbb{B},\mathbb{B})$, $L, L' \in \mathbb{B}$, and $M \in \{0,1\}^{\eta^*}$ with $\lceil |M|/n \rceil = m$.

1. CBC-MAC, based on F instantiated with key K_1, is defined as,

$$\mathsf{CBC\text{-}MAC}_{\mathsf{F}_{K_1}}(M) := \mathsf{CBC}_{\mathsf{F}_{K_1}}(M) \text{ for all } M \in \mathbb{B}^{\eta^+}.$$

2. EMAC, based on F instantiated twice with key tuple $K^{[2]}$, is defined as,

$$\mathsf{EMAC}_{\mathsf{F}_{K_1},\mathsf{F}_{K_2}}(M) := \mathsf{F}_{K_2}(\mathsf{CBC}_{\mathsf{F}_{K_1}}(M)) \text{ for all } M \in \mathbb{B}^{\eta^+}.$$

3. ECBC, based on F instantiated thrice with key tuple $K^{[3]}$, is defined as,

$$\mathsf{ECBC}_{\mathsf{F}_{K_1},\mathsf{F}_{K_2},\mathsf{F}_{K_3}}(M) := \begin{cases} \mathsf{F}_{K_2}(\mathsf{CBC}_{\mathsf{F}_{K_1}}(\overline{M})) & \text{if } |\overline{M}| = |M|, \\ \mathsf{F}_{K_3}(\mathsf{CBC}_{\mathsf{F}_{K_1}}(\overline{M})) & \text{otherwise.} \end{cases}$$

4. FCBC, based on F instantiated thrice with key tuple $K^{[3]}$, is defined as,

$$\mathsf{FCBC}_{\mathsf{F}_{K_1},\mathsf{F}_{K_2},\mathsf{F}_{K_3}}(M) := \begin{cases} \mathsf{F}_{K_2}(\mathsf{CBC}_{\mathsf{F}_{K_1}}(\overline{M}^{[m-1]}) \oplus \overline{M}_m) & \text{if } |\overline{M}| = |M|, \\ \mathsf{F}_{K_3}(\mathsf{CBC}_{\mathsf{F}_{K_1}}(\overline{M}^{[m-1]}) \oplus \overline{M}_m) & \text{otherwise.} \end{cases}$$

5. XCBC, based on L, L', and F instantiated with key K_1, is defined as,

$$\mathsf{XCBC}_{L,L',\mathsf{F}_{K_1}}(M) := \begin{cases} \mathsf{F}_{K_1}(\mathsf{CBC}_{\mathsf{F}_{K_1}}(\overline{M}^{[m-1]}) \oplus \overline{M}_m \oplus L) & \text{if } |\overline{M}| = |M|, \\ \mathsf{F}_{K_1}(\mathsf{CBC}_{\mathsf{F}_{K_1}}(\overline{M}^{[m-1]}) \oplus \overline{M}_m \oplus L') & \text{otherwise.} \end{cases}$$

6. TMAC, based on L, and F instantiated with key K_1, is defined as,

$$\mathsf{TMAC}_{L,\mathsf{F}_{K_1}}(M) := \begin{cases} \mathsf{F}_{K_1}(\mathsf{CBC}_{\mathsf{F}_{K_1}}(\overline{M}^{[m-1]}) \oplus \overline{M}_m \oplus L) & \text{if } |\overline{M}| = |M|, \\ \mathsf{F}_{K_1}(\mathsf{CBC}_{\mathsf{F}_{K_1}}(\overline{M}^{[m-1]}) \oplus \overline{M}_m \oplus \mu \odot L) & \text{otherwise,} \end{cases}$$

where $\mu \odot L$ denotes the multiplication of $\mu \in \mathbb{F}_{2^n} \setminus \{0, 1\}$ and L over $\mathbb{F}_{2^n}$ (viewing $\mathbb{B}$ as $\mathbb{F}_{2^n}$).

7. OMAC, based on F instantiated with key K_1, is defined as,

$$\mathsf{OMAC}_{\mathsf{F}_{K_1}}(M) := \begin{cases} \mathsf{F}_{K_1}(\mathsf{CBC}_{\mathsf{F}_{K_1}}(\overline{M}^{[m-1]}) \oplus \overline{M}_m \oplus \mathsf{F}_{K_1}(0)) & \text{if } |\overline{M}| = |M|, \\ \mathsf{F}_{K_1}(\mathsf{CBC}_{\mathsf{F}_{K_1}}(\overline{M}^{[m-1]}) \oplus \overline{M}_m \oplus \mu \odot \mathsf{F}_{K_1}(0)) & \text{otherwise,} \end{cases}$$

where $\mu \odot \mathsf{F}_{K_1}(0)$ is defined as in case of TMAC.

We will mostly focus on EMAC, ECBC, FCBC, XCBC, and TMAC. However, our attack, given in Lemma 4.1.3, also works for OMAC and CBC-MAC with fixed length messages.

First, we replace the multiple instantiations of F with independent and uniform random functions $\Gamma_1, \Gamma_2, \Gamma_3 \xleftarrow{\$} \mathsf{Func}$. This incurs a cost of $\mathbf{Adv}^{\mathsf{prf}}_{\mathsf{F}}(\sigma, t)$, $2\mathbf{Adv}^{\mathsf{prf}}_{\mathsf{F}}(\sigma, t)$, and $3\mathbf{Adv}^{\mathsf{prf}}_{\mathsf{F}}(\sigma, t)$ in case of $\mathsf{XCBC}_{\mathsf{L},\mathsf{L}',\mathsf{F}_{\mathsf{K}_1}}$ and $\mathsf{TMAC}_{\mathsf{L},\mathsf{F}_{\mathsf{K}_1}}$, $\mathsf{EMAC}_{\mathsf{F}_{\mathsf{K}_1},\mathsf{K}_2}$, and $\mathsf{ECBC}_{\mathsf{K}_1,\mathsf{K}_2,\mathsf{K}_3}$ and $\mathsf{FCBC}_{\mathsf{K}_1,\mathsf{K}_2,\mathsf{K}_3}$, respectively. Formally, we have

$$\mathbf{Adv}^{\mathsf{prf}}_{\mathsf{EMAC}_{\mathsf{F},\mathsf{F}}}(q, \ell, \sigma, t) \leq 2\mathbf{Adv}^{\mathsf{prf}}_{\mathsf{F}}(\sigma, t) + \mathbf{Adv}^{\mathsf{prf}}_{\mathsf{EMAC}_{\Gamma_1,\Gamma_2}}(q, \ell, \sigma) \tag{4.2}$$

$$\mathbf{Adv}^{\mathsf{prf}}_{\mathsf{ECBC}_{\mathsf{F},\mathsf{F},\mathsf{F}}}(q, \ell, \sigma, t) \leq 3\mathbf{Adv}^{\mathsf{prf}}_{\mathsf{F}}(\sigma, t) + \mathbf{Adv}^{\mathsf{prf}}_{\mathsf{ECBC}_{\Gamma_1,\Gamma_2,\Gamma_3}}(q, \ell, \sigma) \tag{4.3}$$

$$\mathbf{Adv}^{\mathsf{prf}}_{\mathsf{FCBC}_{\mathsf{F},\mathsf{F},\mathsf{F}}}(q, \ell, \sigma, t) \leq 3\mathbf{Adv}^{\mathsf{prf}}_{\mathsf{F}}(\sigma, t) + \mathbf{Adv}^{\mathsf{prf}}_{\mathsf{FCBC}_{\Gamma_1,\Gamma_2,\Gamma_3}}(q, \ell, \sigma) \tag{4.4}$$

$$\mathbf{Adv}^{\mathsf{prf}}_{\mathsf{XCBC}_{\mathsf{L},\mathsf{L}',\mathsf{F}}}(q, \ell, \sigma, t) \leq \mathbf{Adv}^{\mathsf{prf}}_{\mathsf{F}}(\sigma, t) + \mathbf{Adv}^{\mathsf{prf}}_{\mathsf{XCBC}_{\mathsf{L},\mathsf{L}',\Gamma_1}}(q, \ell, \sigma) \tag{4.5}$$

$$\mathbf{Adv}^{\mathsf{prf}}_{\mathsf{TMAC}_{\mathsf{L},\mathsf{F}}}(q, \ell, \sigma, t) \leq \mathbf{Adv}^{\mathsf{prf}}_{\mathsf{F}}(\sigma, t) + \mathbf{Adv}^{\mathsf{prf}}_{\mathsf{TMAC}_{\mathsf{L},\Gamma_1}}(q, \ell, \sigma) \tag{4.6}$$

4.1.2 PRF Analysis of CBC-MAC Variants

In general, the PRF analysis of CBC-MAC variants can be reduced to the analysis of some collision events on the underlying CBC function. This has been the common technique for the PRF analysis of PRP instantiated CBC-MAC variants. We will follow the same technique here and establish a relationship between the PRF advantage of CBC-MAC variants and the collision probability of CBC function. Basically, we show that the PRF advantage of $\mathfrak{M}^+ \in \{\mathsf{EMAC}, \mathsf{ECBC}, \mathsf{FCBC}, \mathsf{XCBC}, \mathsf{TMAC}\}$ is asymptotically tight in the collision probability of the underlying CBC function. First, we develop some new terminologies to aid our discussions in rest of the chapter. Following that we build the core results of this chapter in shape of Lemma 4.1.1, 4.1.3, Theorem 4.1.4, 4.1.5 and 4.1.6.

Let M_1 and M_2 be two distinct tuple of blocks with block lengths m_1 and m_2, respectively. Analogous to $\mathtt{Icoll}^{\pi}(M_1, M_2)$ and $\mathtt{Ocoll}^{\pi}(M_1, M_2)$ for some $\pi \in \mathsf{Perm}$, we write $\mathtt{Icoll}^{\gamma}(M_1, M_2)$ and $\mathtt{Ocoll}^{\gamma}(M_1, M_2)$ to denote the input and output collision events, i.e., $\mathfrak{u}^{\gamma}_{m_1}(M_1) = \mathfrak{u}^{\gamma}_{m_2}(M_2)$ and $\mathfrak{v}^{\gamma}_{m_1}(M_1) = \mathfrak{v}^{\gamma}_{m_2}(M_2)$, respectively, for a pair of messages M_1 and M_2 and a function $\gamma \in \mathsf{Func}$. The collision events for a tuple of $q \geq 2$ distinct messages M^q is similarly defined as

$$\mathtt{Icoll}^{\Gamma}(M^q) = \bigcup_{i \neq j} \mathtt{Icoll}^{\Gamma}(M_i; M_j),$$

and

$$\mathtt{Ocoll}^{\Gamma}(M^q) = \bigcup_{i \neq j} \mathtt{Ocoll}^{\Gamma}(M_i; M_j).$$

We define **input-collision probability** as $\mathsf{inCP}^{\Gamma}(M^q) = \Pr[\mathtt{Icoll}^{\Gamma}(M^q)]$ and **output-collision probability** as $\mathsf{outCP}^{\Gamma}(M^q) = \Pr[\mathtt{Ocoll}^{\Gamma}(M^q)]$. It is easy to observe that $\mathsf{outCP}^{\Gamma}(M^q)$ can be bounded in terms of $\mathsf{inCP}^{\Gamma}(M^q)$. When $\mathtt{Icoll}$ is true $\mathtt{Ocoll}$ is trivially true. Otherwise, $\mathtt{Ocoll}$ is true when the underlying random function has a collision at the last intermediate output block. More specifically we have,

$$\mathsf{inCP}^{\Gamma}(M^q) \leq \mathsf{outCP}^{\Gamma}(M^q) \leq \mathsf{inCP}^{\Gamma}(M^q) + \frac{q(q-1)}{2^{n+1}}. \tag{4.7}$$

Compare this with the permutation case, where $\mathsf{inCP}^{\Pi}(M^q) = \mathsf{outCP}^{\Pi}(M^q)$. Thus, one can expect additional output collisions in function case. This will be our main attack idea.

Lemma 4.1.1 (PRF–CBC Upper Bound). *For $q, \ell, \sigma \geq 1$ we have,*

1. $\mathbf{Adv}^{\mathsf{prf}}_{\mathsf{EMAC}_{\Gamma_1,\Gamma_2}}(q, \ell, \sigma) \leq \mathsf{inCP}_{q,\ell,\sigma} + \dfrac{q(q-1)}{2^{n+1}}.$

2. $\mathbf{Adv}^{\mathsf{prf}}_{\mathsf{ECBC}_{\Gamma_1,\Gamma_2,\Gamma_3}}(q,\ell,\sigma) \leq \mathsf{inCP}_{q,\ell,\sigma} + \frac{q(q-1)}{2^{n+1}}$.
3. $\mathbf{Adv}^{\mathsf{prf}}_{\mathsf{FCBC}_{\Gamma_1,\Gamma_2,\Gamma_3}}(q,\ell,\sigma) \leq \mathsf{inCP}_{q,\ell,\sigma}$.
4. $\mathbf{Adv}^{\mathsf{prf}}_{\mathsf{XCBC}_{\mathsf{L},\mathsf{L}',\Gamma_1}}(q,\ell,\sigma) \leq \mathsf{inCP}_{q,\ell,\sigma} + \frac{\sigma q}{2^n} + \frac{q(q-1)}{2^{n+1}}$.
5. $\mathbf{Adv}^{\mathsf{prf}}_{\mathsf{TMAC}_{\mathsf{L},\Gamma_1}}(q,\ell,\sigma) \leq \mathsf{inCP}_{q,\ell,\sigma} + \frac{\sigma q}{2^n} + \frac{q(q-1)}{2^{n+1}}$.

The proof of this lemma is given in section 4.1.2.1 and 4.1.2.2.

4.1.2.1 Proof of 1, 2 and 3 of Lemma 4.1.1

Following [19, 34, 163], we view EMAC, ECBC, and FCBC as instances of the Hash-then-PRF paradigm [183] (see chapter 2 section 2.4.2.2). We discuss the implication of this for each of the schemes below:

1. EMAC and ECBC: In case of EMAC and ECBC, we consider the output of the underlying CBC function as the output of an almost-universal hash function. For EMAC the final random function Γ_2 acts on the output of this hash function.

   For ECBC the final random function is either Γ_2 or Γ_3 depending upon the padding result. Also observe that in this case the output collision on the CBC outputs for two messages, one each from $\mathbb{B}^{\eta^+}$ and $\{0,1\}^{\eta^*}$ is of no use for the adversary as the final outputs are independent due to the independence of Γ_2 and Γ_3. So we assume that the messages are from $\mathbb{B}^{\eta^+}$. Now using the Hash-then-PRF paradigm we have,

   $$\mathbf{Adv}^{\mathsf{prf}}_{\mathfrak{M}^+}(q,\ell,\sigma) \leq \mathsf{outCP}_{q,\ell,\sigma}. \tag{4.8}$$

   for $\mathfrak{M}^+ \in \{\mathsf{EMAC}, \mathsf{ECBC}\}$. The result follows by simple application of Eq. (4.7).

2. FCBC: In case of FCBC, we consider the input to the final random function as the output of an almost-universal hash function. Again using similar arguments as in the case of ECBC we assume that the messages are from $\mathbb{B}^{\eta^+}$. A small difference from EMAC and ECBC lies in the fact that we need to bound the probability of getting a collision on the final input, which is bounded by at most $\mathsf{inCP}_{q,\ell,\sigma}$, whence we have

   $$\mathbf{Adv}^{\mathsf{prf}}_{\mathsf{FCBC}}(q,\ell,\sigma) \leq \mathsf{inCP}_{q,\ell,\sigma}. \tag{4.9}$$

4.1.2.2 Proof of 4 and 5 of Lemma 4.1.1

We prove the result for XCBC here. The proof for TMAC can be obtained by slight modification of this proof. We use the coefficient-H technique to prove this proposition. Let Ω denote the set of all transcripts realizable via a uniform random function $\Gamma \xleftarrow{\$} \mathsf{Func}[\{0,1\}^{\eta^*}, \mathbb{B}]$. A typical transcript $\omega \in \Omega$ is of the form (M^q, T^q), where $M^q \in (\{0,1\}^{\eta^*})^q$ and $T^q \in \mathbb{B}^q$. Let $\overline{M}^q := (\overline{M}_i)_{i\in[q]} \in (\mathbb{B}^{\eta^+})^q$ be the padded version of M^q. Also, for all $i \in [q]$, $|\overline{M}_i| = m_i \leq \ell$, and $\sum_{i=1}^q m_i \leq \sigma$.

We take all transcripts $\omega \in \Omega$ to be good, i.e., $\Pr[\Theta_0 \in \Omega_{\mathsf{bad}}] = 0$. We fix a transcript $\omega = (M^q, T^q) \in \Omega$. In the ideal world, we have

$$\Pr[\Theta_0 = \omega] = \Pr[\Gamma(M^q) = T^q] = \frac{1}{2^{nq}}. \tag{4.10}$$

Let $\overline{\mathsf{u}}$ denote the random tuple $\left(\mathsf{u}_{m_1}^{\Gamma_1}(\overline{M}_1) \oplus \mathsf{K}_1, \ldots, \mathsf{u}_{m_q}^{\Gamma_1}(\overline{M}_q) \oplus \mathsf{K}_q\right)$, where $\mathsf{K}_i = \mathsf{L}$ if $M_i \in \mathbb{B}^{\eta^+}$, and $\mathsf{K}_i = \mathsf{L}'$, otherwise. Let $\mathtt{Fresh}$ denote the event,

$$\forall r, r' \in [q] \text{ and } i \in [m_{r'}], \left(\overline{\mathsf{u}}_r \neq \mathsf{u}_i^{\Gamma_1}(\overline{M}_{r'})\right) \wedge \left(r \neq r' \implies \overline{\mathsf{u}}_r \neq \overline{\mathsf{u}}_{r'}\right).$$

The output of XCBC is completely random when $\mathtt{Fresh}$ holds, as Γ_1 has not been sampled over $\overline{\mathsf{u}}$. Thus, we have,

$$\Pr\left[\mathsf{XCBC}_{\mathsf{L},\mathsf{L}',\Gamma_1}(M^q) = T^q | \mathtt{Fresh}\right] = \Pr\left[\Gamma_1(\overline{\mathsf{u}}) = T^q | \mathtt{Fresh}\right] = \frac{1}{2^{nq}}.$$

So the interpolation probability in the real world can be written as,

$$\begin{aligned}\Pr[\Theta_1 = \omega] &\geq \Pr[\Theta_1 = \omega | \mathtt{Fresh}] \times \Pr[\mathtt{Fresh}] \\ &\geq \frac{1}{2^{nq}}\left(1 - \Pr[\neg\mathtt{Fresh}]\right).\end{aligned}$$

Once we upper bound $\Pr[\neg\mathtt{Fresh}]$, we are done by the application of coefficient-H technique. The event $\neg\mathtt{Fresh}$ holds if one of the following three events occur:

1. $\mathtt{B}_1$: $\exists^* r, r' \in [q]$, such that $M_r, M_{r'} \in \mathbb{B}^{\eta^+}$ (or $M_r, M_{r'} \in \{0,1\}^{\eta^+}$), and $\overline{\mathsf{u}}_r = \overline{\mathsf{u}}_{r'}$.
2. $\mathtt{B}_2$: $\exists^* r, r' \in [q]$, such that $M_r \in \mathbb{B}^{\eta^+}$ and $M_{r'} \in \{0,1\}^{\eta^+}$, and $\overline{\mathsf{u}}_r = \overline{\mathsf{u}}_{r'}$.
3. $\mathtt{B}_3$: $\exists r, r' \in [q]$ and $i \in [m_{r'} - 1]$, such that $\overline{\mathsf{u}}_r = \mathsf{u}_i^{\Gamma_1}(\overline{M}_{r'})$.

BOUNDING $\Pr[\mathtt{B}_1]$: Note that, for $\mathtt{B}_1$ the masking keys do not play any role, and there is actually a collision of the form $\mathsf{u}_{m_r}^{\Gamma_1}(\overline{M}_r) = \mathsf{u}_{m_{r'}}^{\Gamma_1}(\overline{M}_{r'})$. So we can bound $\Pr[\mathtt{B}_1]$ in

terms of inCP, i.e,

$$\Pr[\mathtt{B}_1] \leq \mathsf{inCP}_{q,\ell,\sigma}.$$

BOUNDING $\Pr[\mathtt{B}_2]$: For a fix $\gamma \in \mathsf{Func}$, the conditional probability of $\mathtt{B}_2$ is dependent only on the randomness of the masking keys. Further the two masking keys L and L' are uniformly and independently distributed over $\mathbb{B}$. Thus, we have

$$\begin{aligned}\Pr[\mathtt{B}_2|\Gamma = \gamma] &\leq \sum_{r<r'\in[q]} \Pr\left[\mathsf{L} \oplus \mathsf{L}' = \mathfrak{u}^{\gamma}_{m_r}(\overline{M}_r) \oplus \mathfrak{u}^{\gamma}_{m_{r'}}(\overline{M}_{r'})|\Gamma = \gamma\right] \\ &\leq \frac{\binom{q}{2}}{2^n}.\end{aligned}$$

Since the conditional probability of $\mathtt{B}_2$ given γ is independent of the choice of γ, we can conclude that

$$\Pr[\mathtt{B}_2] \leq \frac{q(q-1)}{2^{n+1}}.$$

BOUNDING $\Pr[\mathtt{B}_3]$: For a fix $\gamma \in \mathsf{Func}$, the conditional probability of $\mathtt{B}_3$ is dependent only on the randomness of the masking keys (either one of them). Thus, we have

$$\begin{aligned}\Pr[\mathtt{B}_3|\Gamma = \gamma] &\leq \sum_{r\in[q]} \sum_{(r',i)\in[q]\times[m_{r'}-1]} \Pr\left[\mathsf{K}_r = \mathfrak{u}^{\gamma}_{m_r}(\overline{M}_r) \oplus \mathfrak{u}^{\gamma}_{i}(\overline{M}_{r'})|\Gamma = \gamma\right] \\ &\leq \frac{\sigma q}{2^n}.\end{aligned}$$

Again, since the conditional probability of $\mathtt{B}_3$ is independent of the choice of γ, so we have

$$\Pr[\mathtt{B}_3] \leq \frac{\sigma q}{2^n}.$$

Thus,

$$\Pr[\neg\mathtt{Fresh}] \leq \mathsf{inCP}_{q,\ell,\sigma} + \frac{\sigma q}{2^n} + \frac{q(q-1)}{2^{n+1}}.$$

The result follows from coefficient-H technique. □

Remark 4.1.2. Note that, the PRF analysis of OMAC is missing in lemma 4.1.1. Our proof technique cannot be applied directly in case of OMAC. We bound the probability of getting a collision at the input block of the final function. For CBC-MAC variants (other than OMAC) this can be argued using the randomness of the independent random functions or the auxiliary keys. In case of OMAC, $\Gamma_1(0)$ is used to mask the final internal input block. Whenever the first message block is 0, $\Gamma_1(0)$ is already defined, hence the current proof technique will not work. Having said that, we believe that identical upper bound should hold for OMAC also.

Lemma 4.1.3 (PRF–CBC Lower Bound). *Let $q, \ell \geq 1$ and $M^q := (M_1, \ldots, M_q)$ be a q-tuple of distinct messages such that for $i \in [q]$, $M_i \in \mathbb{B}\times(0^n)^{\ell-1}$. Then $\forall \mathfrak{M}^+ \in \{$CBC-MAC, EMAC, ECBC, FCBC,*

we have

$$\mathbf{Adv}^{\mathsf{prf}}_{\mathfrak{M}^+}(q,\ell) \geq \mathsf{inCP}(M^q)\left(1-\frac{q(q-1)}{2^{n+1}}\right).$$

Proof. To show the lower bound we present an adversary $\mathscr{A}$ that achieves the claimed PRF advantage using the given message tuple. Consider the following attack algorithm for $\mathfrak{M}^+$:

1. $\mathscr{A}$ queries $M_i \in M^q$ and observes the corresponding output T_i.
2. If $T_i = T_j$ for some $j < i$ then $\mathscr{A}$ returns 1.

Let $\mathbf{p}_{\mathfrak{M}^+}$ and $\mathbf{p}_\Gamma$ denote $\Pr\left[\mathscr{A}^{\mathfrak{M}^+}=1\right]$ and $\Pr\left[\mathscr{A}^{\Gamma}=1\right]$, respectively. Let $\mathbf{cp}_q$ denote the probability of having a collision on q elements chosen uniformly and independently from $\mathbb{B}$. It is well-known [78] that

$$1-\mathrm{e}^{-\frac{k(k-1)}{2^{n+1}}} \leq \mathbf{cp}_q \leq \frac{k(k-1)}{2^{n+1}}. \tag{4.11}$$

Now, we know that,

$$\mathbf{Adv}^{\mathsf{prf}}_{\mathfrak{M}^+}(q,\ell,\sigma) \geq \left|\mathbf{p}_{\mathfrak{M}^+} - \mathbf{p}_\Gamma\right|.$$

For $\mathfrak{M}^+ \in \{\mathsf{CBC\text{-}MAC}, \mathsf{EMAC}, \mathsf{ECBC}, \mathsf{FCBC}\}$ we have,

$$\begin{aligned}
\left|\mathbf{p}_{\mathrm{MAC}} - \mathbf{p}_\Gamma\right| &\overset{1}{=} \left|\mathsf{outCP}(M^q) + (1-\mathsf{outCP}(M^q))\cdot \mathbf{cp}_q - \mathbf{cp}_q\right| \\
&\overset{2}{=} \left|\mathsf{outCP}(M^q)\cdot(1-\mathbf{cp}_q)\right| \\
&\overset{3}{\geq} \left|\mathsf{inCP}(M^q)\cdot(1-\mathbf{cp}_q)\right| \\
&\overset{4}{\geq} \mathsf{inCP}(M^q)\cdot\left(1-\frac{q(q-1)}{2^{n+1}}\right)
\end{aligned}$$

We use Eq. (4.7) from 2 to 3 and Eq. (4.11) from 3 to 4. We can have similar analysis for $\mathfrak{M}^+ \in \{\mathsf{XCBC}, \mathsf{TMAC}, \mathsf{OMAC}\}$ by replacing outCP with inCP in the first equality above. □

Note that, due to the choice of messages (a non-zero block followed by $\ell-1$ zero blocks) in the attack, the CBC function can also be viewed as an iterated random function $\Gamma^{(\ell)}$ (denotes ℓ times composition of Γ). In other words, our attack also applies on the general iterated random function. Lemma 4.1.1 and 4.1.3 show that the PRF advantages of EMAC, ECBC, FCBC, XCBC, and TMAC are tight in inCP of CBC function.

4.1.3 Summary of Main Results

The following theorems are the main technical results of this chapter which quantify $\mathsf{inCP}_{q,\ell,\sigma}$. The proof of these theorems are postponed to section 4.2 and 4.3.

Theorem 4.1.4 (Upper Bound Theorem). *Let $q, \ell, \sigma \geq 1$. Let $M^q = (M_1, \ldots, M_q)$ be a q-tuple of distinct messages such that $M_i \in \mathbb{B}^{m_i}$, $m_i \leq \ell$ for all $i \in [q]$, and $\sum_{i=1}^{q} m_i \leq \sigma$. Then we have,*

$$\mathsf{inCP}_{q,\ell,\sigma}(M^q) \leq \frac{\sigma q}{2^n} + \frac{\sigma \ell}{2^n} + \frac{8\sigma q \ell^3}{2^{2n}}.$$

PROOF OVERVIEW — The proof strategy is identical to the proof of Lemma 3.5.1 in chapter 3. Note that, the bound in this case is worse than the bound in Lemma 3.5.1. As we will see in the proof (see section 4.2), this is due to the cumulative effect of internal input collisions.

Theorem 4.1.5 (Lower Bound Theorem). *Let $q, \ell, \sigma \geq 1$. Let $M^q = (M_1, \ldots, M_q)$ be a q-tuple of distinct messages such that $M_i \in \mathbb{B} \times (0^n)^{\ell-1}$. Then we have,*

$$\begin{aligned}\mathsf{inCP}(M^q) \geq \binom{q}{2}\frac{\ell-1}{2^n}\mathrm{e}^{-\frac{4\ell^2}{2^n}} - 3\binom{q}{3}\left(\frac{2\ell^2}{2^{2n}} + \frac{6\ell^6}{2^{3n}}\right) \\ - \frac{1}{2}\binom{q}{2}\binom{q-2}{2}\left(\frac{\ell^2}{2^{2n}} + \frac{6\ell^3 + 2\ell^5}{2^{3n}} + \frac{28\ell^8}{2^{4n}}\right)\end{aligned}$$

For $\ell, q \geq 3$, $\frac{q^2\ell}{2^n} < 1$ and $\ell < \min\{\frac{2^n}{5184}, \frac{2^{n/2}}{4\sqrt{3}}, \frac{2^{n/3}}{\sqrt[3]{36}}\}$, the above expression is at least $\frac{q^2\ell}{12\cdot 2^n}$. Further if we take $\ell = \frac{\sigma}{q}$, the expression is at least $\frac{\sigma q}{12\cdot 2^n}$.

Note that, in Lemma 4.1.3, the advantage can be lower bounded to $\frac{1}{2}\mathsf{inCP}(M^q)$ for $q < 2^{n/2}$. Using Lemma 4.1.1, 4.1.3, and Theorem 4.1.4, 4.1.5 we have the exact PRF security bounds.

Theorem 4.1.6 (PRF Bound). *Let $q, \ell, \sigma \geq 3$, such that $\frac{q^2\ell}{2^n} < 1, q < 2^{n/2}$, and $\ell < \min\{\frac{2^n}{5184}, \frac{2^{n/2}}{4\sqrt{3}}, \frac{2^{n/3}}{\sqrt[3]{36}}, q\}$. Then, $\forall \mathfrak{M}^+ \in \{\mathsf{EMAC}, \mathsf{ECBC}, \mathsf{FCBC}, \mathsf{XCBC}, \mathsf{TMAC}\}$, the PRF advantage of $\mathfrak{M}^+$ is asymptotically tight in terms of q, ℓ and σ, i.e.,*

$$\mathbf{Adv}^{\mathsf{prf}}_{\mathfrak{M}^+}(q, \ell, \sigma) = \Theta\left(\frac{\sigma q}{2^n}\right).$$

From the above discussion it is clear that, we are only left with the analysis of the collision probability of CBC. More specifically we have to prove Theorem 4.1.4 and 4.1.5. Following chapter 3, we use structure graphs to bound the collision probability, albeit with a slightly different definition.

In the rest of the chapter, we take M^q as the q-tuple of distinct messages such that $M_i \in \mathcal{B}^{m_i}$, $m_i \leq \ell$ for all $i \in [q]$, and $\sum_{i=1}^{q} m_i \leq \sigma$.

4.1.4 Input-Structure Graph

We mostly use the same set of notations and terminologies as used in section 3.2 of chapter 3. However, we define the block-vertex structure graphs over intermediate input vectors instead of output vectors. This small change is required as we will be mostly interested in `Icoll`, and in case of functions `Ocoll` does not imply `Icoll`. In the following subsections we briefly revisit the notations and definitions on structure graphs. Note that, we prepend the adjective "input" just to emphasize that the vertices in the graph correspond to the intermediate inputs.

4.1.4.1 Intermediate Inputs and Outputs

We define set $\mathsf{Func}_\perp$ as

$$\mathsf{Func}_\perp := \{\gamma \mid \gamma : \mathbb{B} \cup \{\perp\} \to \mathbb{B} \wedge \gamma(\perp) = 0^n\}.$$

Note that, the uniform distribution over $\mathsf{Func}_\perp$ has the same probability mass function (pmf) as Func, i.e., a constant function taking up the value 2^{-n2^n}.

We extend the definition of CBC function for any $\gamma \in \mathsf{Func}_\perp$ by setting $\mathfrak{u}_0^\gamma(M_i) = \perp$ for all $i \in [q]$, and then following the normal CBC computation.

Recall that, this extension does not hamper our analysis, as the uniform distribution over $\mathsf{Func}_\perp$ follows the same pmf as $\mathsf{Func}_\perp$. This extension may (or may not) look artificial at the moment, but it will greatly simplify some of the definitions and proofs that we discuss later. From now onward, γ and Γ will have their usual meaning, but over the set $\mathsf{Func}_\perp$.

INDEX SET: We define

$$\mathcal{I} = \{(r, i) : r \in [q], i \in (m_r]\}.$$

The dictionary order $\prec$ over $\mathcal{I}$ is defined analogous to $\mathcal{I}_0$ of section 3.2 of chapter 3. Note that, $\prec$ is a partial order. For $\omega, \omega' \in \mathcal{I}$, we say that ω' is a *successor* of ω if $\omega \prec \omega'$. In such cases, we say that ω is a *predecessor* of ω'. For all $\omega \in \mathcal{I}$ and $i \in \mathbb{N}$, we sometimes write $\omega + i$ to denote the i-th successor of ω in order.

SEQUENCES FOR INTERMEDIATE INPUTS AND OUTPUTS: For any $r \in [q]$, the subsequence $(x_{r,i})_{i \in (m_r]}$ is denoted x_{r*}. The sequences of intermediate inputs and outputs over $\mathcal{I}$ is denoted as $\mathfrak{u}^\gamma(M^q)$ and $\mathfrak{v}^\gamma(M^q)$, respectively, where $\forall r \in [q]$, $\mathfrak{u}_{r*}^\gamma(M^q) := \mathfrak{u}^\gamma(M_r)$ and $\mathfrak{v}_{r*}^\gamma(M^q) := \mathfrak{v}^\gamma(M_r)$.

The *block-vertex input-structure (BINS) graph* $\mathsf{BStruct}^{\gamma}(M^q) = (\mathcal{V}^{\gamma}, \mathcal{E}^{\gamma})$ for a function γ is defined by the labeled vertex set $\mathcal{V}^{\gamma} = \widehat{\mathfrak{u}}^{\gamma}(M^q)$, and the set of labeled edges

$$\mathcal{E} := \cup_{r=1}^{q}\{(\mathfrak{u}^{\gamma}_{(r,i-1)}, \mathfrak{u}^{\gamma}_{(r,i)}, M_{(r,i)}) : i \in [m_r]\}.$$

Note that, we skipped M^q from the parametrization as it is quite clear from the context. Clearly, $\perp \in \mathcal{V}^{\gamma}$ has zero in-degree and positive out-degree, and $\mathsf{BStruct}^{\gamma}$ is a union of M_i-walks, denoted W_i, for $i \in [q]$. Note that,

$$u \xrightarrow{A} v \;\Rightarrow\; \gamma(u) \oplus A = v. \tag{4.12}$$

So, *for every $u \in V$, all outward edges (similarly for inward edges) have distinct edge labels.* Using this property, it is easy to see that for all $i \in [q]$, the W_i walk is unique in a BINS graph. We denote the set of all BINS graphs by $\mathsf{BStruct}(M^q)$. Recall the minimum index mapping α of section 3.2.2, that maps a vertex $v \in \mathcal{V}^{\gamma}$ to the minimum index $(r, i) \in \mathcal{I}$ such that $W_{(r,i)} = v$.

Definition 4.1.7 (Input-Structure Graph)**.** The input-structure (INS) graph $\mathsf{Struct}^{\gamma} = (\widetilde{\mathcal{V}}^{\gamma}, \widetilde{\mathcal{E}}^{\gamma})$ associated to γ is the α-transformed $\mathsf{BStruct}^{\gamma}$, i.e. $\mathsf{Struct}^{\gamma} = \alpha(\mathsf{BStruct}^{\gamma})$.

The following example is a slight modification of example 3.1 given in chapter 3.

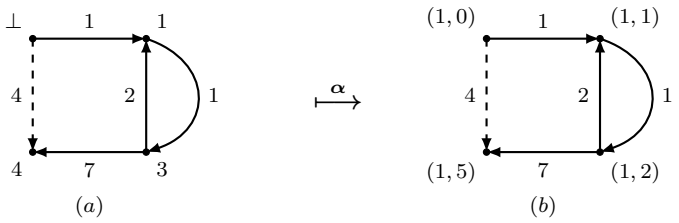

Figure 4.1.1: INS graph corresponding to a BINS graph.

Example 4.1. *Let $M_1 = (1, 1, 2, 1, 7)$ and $M_2 = (4)$ be two messages and $\gamma(1) = 2$; $\gamma(3) = 3$ for some $\gamma \in \mathsf{Func}_{\perp}$. Then, we have $\mathfrak{u}^{\gamma}(M_1) = (\perp, 1, 3, 1, 3, 4)$ and $\mathfrak{u}^{\gamma}(M_2) = (\perp, 4)$. The corresponding BINS graph $\mathsf{BStruct}^{\gamma}$, shown in Figure 4.1.1(a), has vertex set $\mathcal{V}^{\gamma} = \{\perp, 1, 3, 4\}$ and edge set*

$$\mathcal{E}^{\gamma} = \{(\perp, 1, 1), (1, 3, 1), (3, 1, 2), (3, 4, 7), (\perp, 4, 4)\}.$$

The corresponding INS graph Struct^{γ}, illustrated in Figure 3.2.2(b), has vertex set $\widetilde{\mathcal{V}}^{\gamma} = \{(1, 0), (1, 1), (1, 2), (1, 5)\}$ and edges set

$$\widetilde{\mathcal{E}}^{\gamma} = \{((1, 0), (1, 1), 1), ((1, 1), (1, 2), 1), ((1, 2), (1, 1), 2), ((1, 2), (1, 5), 7), ((1, 0), (1, 5), 4)\}.$$

It is easy to see that an INS graph is again a union of $\widetilde{W_r}$ walks corresponding to message M_r. Further, $(1,0)$ (i.e. $\alpha(\perp)$) has zero in-degree, and positive out-degree. We denote the set of all INS graphs for M^q by $\mathsf{Struct}(M^q)$.

Definition 4.1.8 (valid block labeling)**.** An injective function $\beta : \widetilde{\mathcal{V}} \to \mathbb{B} \cup \{\perp\}$ is called a *valid block labeling* for an INS graph $\widetilde{\mathcal{G}} = (\widetilde{\mathcal{V}}, \widetilde{\mathcal{E}})$ if the graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is a BINS graph where

1. $\mathcal{V} := \beta(\mathcal{V}) = \{\beta_v := \beta(v) : v \in \widetilde{\mathcal{V}}\}$, and
2. $\mathcal{E}$ is the edge set after relabeling v by β_v, i.e. $\mathcal{E} = \{(\beta_u, \beta_v, x) : (u, v, x) \in \mathcal{E}\}$.

NECESSARY CONDITIONS FOR A VALID BLOCK LABELING: Analogous to the valid block label in case of permutation-based structure graphs (see Definition 3.2.2) one can identify certain restrictions on valid block labeling for an INS graph. A valid block labeling must map $(1,0)$ to $\perp$, and $(1,0)$ should be the only preimage of $\perp$. Second, β must be injective as distinct vertices in a BINS graph have distinct block labels. Lastly, whenever we have edges $e_1 := (u, v, a), e_2 := (u, w, b) \in \widetilde{\mathcal{E}}$, we must have $\beta_v \oplus a = \beta_w \oplus b$, as these are equal to $\gamma(u)$ for some $\gamma \in \mathsf{Func}_\perp$.

Note that, if there exists an edge $e_1 := (u, w, a) \in \widetilde{\mathcal{E}}$, then $\gamma(\beta_u) \oplus a = \beta_w$ for some $\gamma \in \mathsf{Func}_\perp$. Now, if there exists another edge $e_2 := (v, w, b) \in \widetilde{\mathcal{E}}$, then we must have $\gamma(\beta_u) + \gamma(\beta_v) = a \oplus b$. A collision in $\widetilde{\mathcal{G}}$ is defined by such a triple $\delta = (u, v; w)$. The set $\{u, v\}$ is called the source of the collision whereas w is called the head of the collision. We also say the edges e_1 and e_2 are colliding edges. Thus, a collision $\delta = (u, v; z)$ induces a linear restriction $L_\delta : \gamma(\beta_u) \oplus \gamma(\beta_v) = c_\delta$, where $c_\delta = a \oplus b \in \mathbb{B}$. We denote the set of all collisions in $\widetilde{\mathcal{G}}$ by $\Delta(\widetilde{\mathcal{G}})$. Let $\mathsf{rank}(\widetilde{\mathcal{G}})$ denote the rank of the system of linear equations $L(\widetilde{\mathcal{G}}) := \{L_\delta : \delta \in \Delta(\widetilde{\mathcal{G}})\}$.

Definition 4.1.9 (Accident of an INS graph)**.** The accident of an INS graph $\widetilde{\mathcal{G}}$ is defined as $\mathsf{Acc}(\widetilde{\mathcal{G}}) := \mathsf{rank}(\widetilde{\mathcal{G}})$.

Now, we mention some important results on INS graphs which will be useful in our analysis ahead. These results have already been proved in [19, 163] and chapter 3 for the random permutation case. We prove these results for the random function case in section 4.4.

Lemma 4.1.10. *For any INS graph $\widetilde{\mathcal{G}}$ with a accidents,*

$$\Pr\left[\mathsf{Struct}^{\mathsf{\Gamma}} = \widetilde{\mathcal{G}}\right] \leq \frac{1}{2^{na}}.$$

Lemma 4.1.11. *The number of INS graphs with a accidents associated to $M^q = (M_1, \ldots, M_q)$ is at most $\binom{m}{2}^a$, where $\sum_{i=1}^q m_i = m$.*

Corollary 4.1.12. *Let $a \geq 1$ be an integer. Then,*

$$\Pr\left[\mathsf{Acc}(\mathsf{Struct}^{\Gamma}) \geq a\right] \leq \frac{m^{2a}}{2^{na}}.$$

4.2 Proof of Theorem 4.1.4 [Upper Bound Theorem]

In the previous chapter, we saw that `Ocoll` is a structure graph event. Similarly, one can show that `Icoll` is a structure graph event as well. For a fixed tuple of message M^q, `Icoll` is said to be true if there exists some pair of walks W_i and W_j (corresponding to some $M_i, M_j \in M^q$) in $\mathsf{Struct}^{\Gamma_1}(M^q)$ such that $W_{(i,m_i)} = W_{(j,m_j)}$.

Let $\mathsf{Struct}_a(M^q) \subseteq \mathsf{Struct}(M^q)$ denote the set of structure graphs with a accidents for M^q. By extending the notation we write $\mathsf{Struct}_a(M^q)[\texttt{Icoll}]$ to denote the subset of graphs which additionally satisfy `Icoll`. We follow the approach of section 3.5.2 to bound $\mathsf{Struct}_a(M^q)[\texttt{Icoll}]$.

Let us define the event `Bad` as,

1. for a pair of messages M_i, M_j, $\mathsf{Acc}(\mathsf{Struct}^{\Gamma}(M_i, M_j)) \geq 2$, or
2. for any message M_i, $\mathsf{Acc}(\mathsf{Struct}^{\Gamma}(M_i)) \geq 1$.

We aim to bound the inCP in terms of $\Pr[\texttt{Bad}]$ and $\Pr[\neg\texttt{Bad}]$. Specifically, we have

$$\mathsf{inCP}_{q,\ell,\sigma} \leq \Pr_{\Gamma}\left[\texttt{Icoll}^{\Gamma}(M^q) \cap \neg\texttt{Bad}\right] + \Pr_{\Gamma}[\texttt{Bad}].$$

So we just need to upper bound the following:

1. Bounding $\Pr_{\Gamma}[\texttt{Bad}]$: We use Corollary 4.1.12 to bound the probability of `Bad`. The probability of first condition is bounded by $\sum_{i<j\in[q]} \frac{(m_i+m_j)^4}{2^{2n}}$ (for all $\binom{q}{2}$ pairs of messages), and the probability of second conditions is bounded by $\sum_{i\in[q]} \frac{m_i^2}{2^n}$(for all q messages). Finally we have,

$$\begin{aligned}
\Pr[\texttt{Bad}] &\leq \sum_{i<j\in[q]} \frac{(m_i+m_j)^4}{2^{2n}} + \sum_{i\in[q]} \frac{m_i^2}{2^n} \\
&\leq \sum_{i<j\in[q]} \frac{(m_i+m_j)\cdot 8\ell^3}{2^{2n}} + \sum_{i\in[q]} \frac{m_i\cdot\ell}{2^n} \\
&\leq \frac{8\sigma q\ell^3}{2^{2n}} + \frac{\sigma\ell}{2^n}.
\end{aligned}$$

2. Bounding $\Pr_\Gamma\left[\texttt{Icoll}^\Gamma(M^q) \cap \neg\texttt{Bad}\right]$: $\neg\texttt{Bad}$ implies that $\mathsf{Acc}(W_i) = 0$ for all $i \in [q]$, or in other words W_i is acyclic. For any pair of messages M_i and M_j, we bound the set $|\mathsf{Struct}_1[\texttt{Icoll} \wedge \neg\texttt{Bad}]|$ to at most $\min\{m_i, m_j\}$ (see the following claim), whence we bound the probability to at most $\frac{\min\{m_i,m_j\}}{2^n}$. Hence for q messages we have
$$\Pr_\Gamma\left[\texttt{Icoll}^\Gamma(M^q) \cap \neg\texttt{Bad}\right] \leq \sum_{i<j\in[q]} \frac{\min\{m_i, m_j\}}{2^n} \leq \frac{\sigma q}{2^n}.$$

Combining 1 and 2, we have the desired result.

Claim 4.2.1. For any pair of messages $M_1, M_2 \in M^q$, we have
$$|\mathsf{Struct}_1[\texttt{Icoll} \wedge \neg\texttt{Bad}]| \leq \min\{m_1, m_2\}.$$

Proof. We prove the claim in two cases:

Case A: M_1 is a prefix of M_2: In this case M_1 must be a strict prefix of M_2 as M_1 and M_2 are distinct. Further W_1 is a subwalk of W_2 and we have $W_{1,i} = W_{2,i}$ for $i \in (m_1]$. The $\texttt{Icoll}$ event is equivalent to $W_{1,m_1} = W_{2,m_2}$, or $W_{2,m_2} = W_{2,m_1}$. Thus, W_2 must contain a cycle which is not possible. So, $|\mathsf{Struct}_1[\texttt{Icoll} \wedge \neg\texttt{Bad}]| = 0$.

Case B: M_1 is not a prefix of M_2: WLOG assume that $m_1 < m_2$. In this case we must have $p = \mathsf{lcp}(M_1, M_2) < m_1$. $\neg\texttt{Bad}$ implies that W_1 and W_2 are paths and $\texttt{Icoll}$ implies that $W_{1,m_1} = W_{2,m_2}$. To get $W_{1,m_1} = W_{2,m_2}$ we must have an accident $(W_{1,i}, W_{2,j}; W_{1,i+1})$ for some $p+1 \leq i \leq m_1$ and $j = m_2 - m_1 + i$. Therefore, summing over all values of i we have, $|\mathsf{Struct}_1(M_1, M_2)[\texttt{Icoll} \wedge \neg\texttt{Bad}]| \leq m_1 \leq \min\{m_1, m_2\}$.

The result follows from case 1 and 2. □

4.3 Proof of Theorem 4.1.5 [Lower Bound Theorem]

Let $M^q := (M_1, \ldots, M_q)$ be a q-tuple of messages such that for $i, j \in [q]$ $M_i = (X_i, 0, \cdots, 0) \in \mathcal{B}^\ell$ and $X_i \neq X_j \in \mathcal{B}$. We want to find a lower bound of collision probability that is,
$$\begin{aligned}\mathsf{inCP}(M^q) = \Pr\left[\texttt{Icoll}^\Gamma(M^q)\right] &= \Pr\left[\bigcup_{1\leq i<j\leq q} \texttt{Icoll}^\Gamma(M_i; M_j)\right] \\ &\geq \sum_{i<j} \Pr\left[\texttt{Icoll}^\Gamma(M_i, M_j)\right]\end{aligned}$$

$$- 3 \sum_{i<j<k} \Pr\left[\texttt{Icoll}^{\Gamma}(M_i, M_j) \wedge \texttt{Icoll}^{\Gamma}(M_j, M_k)\right]$$
$$- \frac{1}{2} \sum_{\substack{i<j,k<m \\ \{i,j\}\cap\{k,m\}=\emptyset}} \Pr\left[\texttt{Icoll}^{\Gamma}(M_i, M_j) \wedge \texttt{Icoll}^{\Gamma}(M_k, M_m)\right] \tag{4.13}$$

where the inequality follows from Principle of Inclusion-Exclusion and Bonferroni inequality (for details see section A.2.1 of appendix A). We have to compute the following bounds,

- Upper bound for $\Pr\left[\texttt{Icoll}^{\Gamma}(M_i; M_j) \wedge \texttt{Icoll}^{\Gamma}(M_j; M_k)\right]$, where i, j and k are distinct.
- Upper bound for $\Pr\left[\texttt{Icoll}^{\Gamma}(M_i; M_j) \wedge \texttt{Icoll}^{\Gamma}(M_k; M_m)\right]$, where i, j, k and m are distinct.
- Lower bound for $\Pr\left[\texttt{Icoll}^{\Gamma}(M_i; M_j)\right]$, where i and j are distinct.

We use structure graphs to bound the above mentioned probabilities. Before moving forward, we define a special graph which will be encountered frequently in this section.

Definition 4.3.1 (t-unicycle)**.** A *t-unicycle* is a connected directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ where $\forall v \in \mathcal{V}, \mathbf{deg_{out}}(v) \leq 1 \vee \mathbf{deg_{in}}(v) = 0$, and $\mathcal{E}$ is a union of 1 cycle $\mathcal{C}$ and t distinct paths $\mathcal{P}_1, \cdots, \mathcal{P}_t$ where exactly one endpoint of $\mathcal{P}_i$ is a vertex of $\mathcal{C}$ and the other endpoint must have zero in-degree.

The paths $\mathcal{P}_i$ may not be disjoint. Example 4.2 describes a 3-unicycle.

Observe that due to our choice of messages, we have the following property on $\mathsf{Struct}^{\Gamma}(M^q)$:

$$\forall v \in \mathcal{V}(\mathsf{Struct}^{\Gamma}) \setminus \{(1,0)\}, \mathbf{deg_{out}}(v) \leq 1.$$

This is obvious as all the edge labels (except those involving $(1,0)$) are identical (0^n). Therefore, Struct^{Γ} is either a union of paths or a union of unicycles or both. Note that, in either case the graph has no dependent collisions with distinct heads (as that requires $\mathbf{deg_{out}}(v) \geq 2$ for some v). Further, for a vertex with in-degree $a \geq 1$, we have $a - 1$ independent collisions or accidents.

Remark 4.3.2. We summarize some useful properties derived from the above discussion:

1. A union of k paths has at most $k-1$ collisions. So the number of accidents is $k-1$.
2. A k-unicycle has exactly k collisions. So the number of accidents is k.

3. A union of a k_1-unicycle and k_2 paths has at most $k_1 + k_2 - 1$ accidents.
4. A union of a k_1-unicycle and a k_2-unicycle has exactly $k_1 + k_2$ accidents.
5. In general k distinct walks (where each vertex v has $\mathbf{deg_{out}}(v) \leq 1$), can have at most k accidents.

Example 4.2. *In figure 4.3.1,*

1. *$\mathcal{G}$ is a 3-unicycle with* $\mathsf{Acc}(\mathcal{G}) = 3$.
2. *$\mathcal{G}'$ is a union of 3 paths with* $\mathsf{Acc}(\mathcal{G}') = 1$.

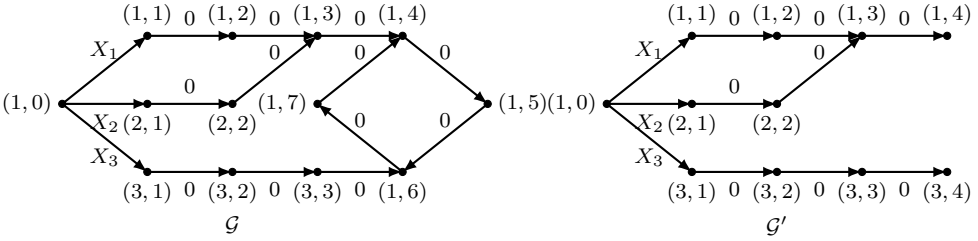

Figure 4.3.1: Unicycles and union of paths.

We bound the above mentioned probabilities in Lemma 4.3.3, 4.3.4, and 4.3.5. The proofs are postponed to section 4.3.1.

Lemma 4.3.3.

$$\Pr\left[\texttt{Icoll}^{\Gamma}(M_i; M_j) \wedge \texttt{Icoll}^{\Gamma}(M_j; M_k)\right] \leq \frac{2\ell^2}{2^{2n}} + \frac{6\ell^6}{2^{3n}}.$$

Lemma 4.3.4.

$$\Pr\left[\texttt{Icoll}^{\Gamma}(M_i; M_j) \wedge \texttt{Icoll}^{\Gamma}(M_k; M_m)\right] \leq \frac{\ell^2}{2^{2n}} + \frac{6\ell^3 + 2\ell^5}{2^{3n}} + \frac{28\ell^8}{2^{4n}}.$$

Lemma 4.3.5.

$$\Pr\left[\texttt{Icoll}^{\Gamma}(M_i; M_j)\right] \geq \frac{\ell}{2^n} \mathrm{e}^{-\frac{4\ell^2}{2^n}}.$$

Combining Equation 4.13 with Lemma 4.3.3, 4.3.4 and 4.3.5 we have,

$$\begin{aligned}\mathsf{inCP}(M^q) \geq \binom{q}{2}\frac{\ell-1}{2^n}\mathrm{e}^{-\frac{4\ell^2}{2^n}} - 3\binom{q}{3}\left(\frac{2\ell^2}{2^{2n}} + \frac{6\ell^6}{2^{3n}}\right) \\ - \frac{1}{2}\binom{q}{2}\binom{q-2}{2}\left(\frac{\ell^2}{2^{2n}} + \frac{6\ell^3+2\ell^5}{2^{3n}} + \frac{28\ell^8}{2^{4n}}\right).\end{aligned}$$

If $\ell, q \geq 3$, $\frac{q^2\ell}{2^n} < 1$ and $\ell < \min\{\frac{2^n}{5184}, \frac{2^{n/2}}{4\sqrt{3}}, \frac{2^{n/3}}{\sqrt[3]{36}}\}$, then using the inequality $e^{-x} \geq 1 - x$ and some algebraic manipulations one can show

$$\mathsf{inCP}(M^q) \geq \frac{q^2\ell}{122^n}.$$

Further for $\ell = \sigma/q$ this bound becomes $\frac{q\sigma}{122^n}$. □

4.3.1 Proofs Related to the Lower Bound Theorem

4.3.1.1 Proof of Lemma 4.3.3

Let $\texttt{Icoll}^{i,j,k}$ denote the event $\texttt{Icoll}^{\Gamma}(M_i, M_j) \wedge \texttt{Icoll}^{\Gamma}(M_j, M_k)$. From remark 4.3.2 we know that the number of accidents must be ≤ 3.

1. Observe that $\texttt{Icoll}^{i,j,k}$ requires at least 2 independent collisions (accidents) (in W_i and W_j paths and W_k and $W_i \cup W_j$), so

$$|\mathsf{Struct}_1[\texttt{Icoll}^{i,j,k}]| = 0.$$

2. Accident 2 graphs are possible only for union of paths, as the number of accidents correspond to the collision between the three paths. There are at most ℓ many choices for collision between W_i and W_j paths, and at most 2ℓ many choices for collision between W_k and $W_i \cup W_j$. This bounds

$$|\mathsf{Struct}_2[\texttt{Icoll}^{i,j,k}]| \leq 2\ell^2.$$

3. The graph can have 3 accidents iff it is a 3-unicycle. Suppose the cycle is in W_i. Then W_i is determined by the length of cycle and the distance of $W_{i,0}$ from the cycle which gives ℓ^2 choices for W_i. For each such choice we have at most $2\ell^2$ many choices for $W_i \cup W_j$, and at most $3\ell^2$ many choices for $W_i \cup W_j \cup W_k$. This bounds

$$|\mathsf{Struct}_3[\texttt{Icoll}^{i,j,k}]| \leq 6\ell^6.$$

The result follows by direct application of Lemma 4.1.10. □

4.3.1.2 Proof of Lemma 4.3.4

Let $\texttt{Icoll}^{i,j,k,m}$ denote the event $\texttt{Icoll}^{\Gamma}(M_i, M_j) \wedge \texttt{Icoll}^{\Gamma}(M_k, M_m)$. From remark 4.3.2 we know that the number of accidents must be ≤ 4. We bound the four sets corresponding to the number of accidents as below:

1. Accident 1 graphs are not possible. Hence,

$$|\mathsf{Struct}_2[\texttt{Icoll}^{i,j,k,m}]| = 0.$$

2. The accident 2 graphs are possible if and only if $(W_i \cup W_j) \cap (W_k \cup W_m) = \{(1,0)\}$, where the two accidents correspond to the collision between W_i and W_j, and collision between W_k and W_m. Now there are at most ℓ many choices for collision between W_i and W_j paths, and similarly at most ℓ many choices for collision between W_k and W_m paths. This gives

$$|\mathsf{Struct}_2[\mathtt{Icoll}^{i,j,k,m}]| \leq \ell^2.$$

3. Accident 3 graphs are possible if and only if,

   (a) W_i, W_j, W_k, W_m paths collide and the graph is a union of paths. In this case we have at most ℓ many choices for collision between W_i and W_j paths. Similarly we have at most 2ℓ and 3ℓ many choices for collision between W_k and $W_i \cup W_j$, and W_m and $W_i \cup W_j \cup W_m$ respectively. This gives

$$|\mathsf{Struct}_3[\mathtt{Icoll}^{i,j,k,m}]| \leq 6\ell^3.$$

   (b) $(W_i \cup W_j) \cap (W_k \cup W_m) = \{(1,0)\}$ and $W_i \cup W_j$ is a 2-unicycle and $W_k \cup W_m$ is a union of paths or vice versa. Without loss of generality assume that $W_i \cup W_j$ is a 2-unicycle. Then there exist a cycle in $W_i \cup W_j$. Suppose the cycle is in W_i. Then W_i is determined by the length of cycle and the distance of $W_{i,0}$ from the cycle which gives ℓ^2 choices for W_i. For each such choice we have at most $2\ell^2$ many choices for $W_i \cup W_j$. And there are at most ℓ many choices for collision between W_k and W_m. This gives,

$$|\mathsf{Struct}_3[\mathtt{Icoll}^{i,j,k,m}]| \leq 2\ell^5.$$

   Combining the two subcases we have,

$$|\mathsf{Struct}_3[\mathtt{Icoll}^{i,j,k,m}]| \leq 6\ell^3 + 2\ell^5.$$

4. Accident 4 graphs are possible if and only if,

   (a) $W_i \cup W_j$ and $W_k \cup W_m$ are distinct 2-unicycles. Using similar arguments as used in the previous cases we get a bound of $4\ell^8$.

   (b) W_i, W_j, W_k, W_m form a 4-unicycle. This case can be bounded to $24\ell^8$, using similar approach as used in the previous cases.

   Combining the two subcases we have,

$$|\mathsf{Struct}_4[\mathtt{Icoll}^{i,j,k,m}]| \leq 28\ell^8.$$

The result follows by direct application of Lemma 4.1.10. □

4.3.1.3 Proof of Lemma 4.3.5

Let $\texttt{Icoll}^{i,j}$ denote the event $\texttt{Icoll}^{\Gamma}(M_i; M_j)$. We are basically interested in the probability that $\mathsf{Struct}^{\Gamma} \in \mathsf{Struct}(M_i, M_j)[\texttt{Icoll}]$. Let $\texttt{Acyclic}$ denote the property that some graph $\mathcal{G} \in \mathsf{Struct}(M_i, M_j)[\texttt{Icoll}]$ is acyclic graph, and $\mathsf{Struct}(M_i, M_j)[\texttt{Icoll} \wedge \texttt{Acyclic}]$ denote the subset of graphs which satisfy both $\texttt{Icoll}$ and $\texttt{Acyclic}$. Thus, we have

$$\Pr\left[\mathsf{Struct}^{\Gamma} \in \mathsf{Struct}(M_i, M_j)[\texttt{Icoll}]\right] \geq \Pr\left[\mathsf{Struct}^{\Gamma} \in \mathsf{Struct}(M_i, M_j)[\texttt{Icoll} \wedge \texttt{Acyclic}]\right] \tag{4.14}$$

We will lower bound $\Pr\left[\mathsf{Struct}^{\Gamma} \in \mathsf{Struct}(M_i, M_j)[\texttt{Icoll} \wedge \texttt{Acyclic}]\right]$. First, convince yourself that for all $\mathcal{G} \in \mathsf{Struct}(M_i, M_j)[\texttt{Icoll} \wedge \texttt{Acyclic}]$, we must have $\mathsf{Acc}(\mathcal{G}) = 1$ (as M_i and M_j share a common suffix of length $\ell - 1$). Now, this accident can happen at any one of the index $i \in [2 \dots \ell]$, each contributing exactly one structure graph.

Now fix an index $i \in [2 \dots \ell]$ where the accident occurs and let the corresponding INS graph be $\mathcal{G}_i$. Then, a simple counting shows that the number of valid block labeling for $\mathcal{G}_i$ is exactly $(2^n - 2) \dots (2^n - 2i + 2)$. Each such labeling gives a BINS graph $\mathcal{S}$ with exactly $2i - 2$ positive out-degree vertices (excluding $\perp$ which is trivial) such that $\alpha(\mathcal{S}) = \mathcal{G}$. The probability of getting a BINS graph with $2i - 2$ many vertices having positive out-degree is equal to 2^{2n-2in} (as exactly these many outputs of Γ are fixed). Thus, we get

$$\begin{aligned}
\Pr\left[\mathsf{Struct}^{\Gamma} \in \mathsf{Struct}(M_i, M_j)[\texttt{Icoll} \wedge \texttt{Acyclic}]\right] &= \frac{1}{2^n} \sum_{i=2}^{\ell} \left(1 - \frac{2}{2^n}\right) \cdots \left(1 - \frac{2i-2}{2^n}\right) \\
&\geq \frac{\ell - 1}{2^n} \prod_{i=1}^{2\ell-2} \left(1 - \frac{i}{2^n}\right) \\
&\geq \frac{\ell - 1}{2^n} \left(1 - \frac{2\ell^2}{2^n}\right) \\
&\geq \mathrm{e}^{-\frac{4\ell^2}{2^n}} \frac{\ell - 1}{2^n},
\end{aligned} \tag{4.15}$$

where the last inequality follows from $(1 - x) \geq \mathrm{e}^{-2x}$ for $0 < x < 0.5$, and the assumption that $\ell < 2^{\frac{n}{2}-1}$. The result follows from Eq. (4.14) and (4.15). □

4.4 Proofs of Results on Structure Graph

Proof of Lemma 4.1.10: The proof is similar to the proof of Lemma 3.2.6. $\widetilde{\mathcal{G}}$ is an INS graph with a accidents, i.e., $\mathsf{rank}(\widetilde{\mathcal{G}}) = a$. Let s denote the number of vertices, excluding $(1,0)$, in $\widetilde{\mathcal{G}}$ with positive out-degree. Using linear algebra, we know that some $s-a$ choices of $\gamma(\beta_i)$ values will uniquely determine the rest of the block labels and so the number of valid block labeling is at most 2^{ns-na}. In fact, the number of block labeling should be much less due to the added restriction of distinctness. Any valid choice of β induces a block-vertex structure graph $\mathcal{S}(\mathcal{V},\mathcal{E})$ such that $\alpha(\mathcal{S}) = \widetilde{\mathcal{G}}$. Note that, s is the number of vertices $v \in \mathcal{V}$ with positive out-degree. So exactly $(2^n)^{2^n-s}$ number of functions can result in BINS graph $\mathcal{S}$. Therefore,

$$\Pr\left[\mathsf{BStruct}^{\Gamma} = \mathcal{S}\right] = \frac{1}{2^{ns}}. \tag{4.16}$$

The result follows by summing over all possible BINS graphs $\mathcal{S}$ such that the INS graph $\alpha(\mathcal{S}) = \widetilde{\mathcal{G}}$. □

For an INS graph $\widetilde{\mathcal{G}}$ we define *traversal* $\mathcal{T}(\widetilde{\mathcal{G}})$ as the sequence of vertices $\mathcal{T}(\widetilde{\mathcal{G}}) := (\widetilde{W}_{r,i})_{(r,i)\in\mathcal{I}}$. Note that, $\mathcal{T}(\widetilde{\mathcal{G}})$ implicitly stores the edges: for every $\omega \in \mathcal{I}$ such that $\omega \neq (r, m_r)$ we have $(\widetilde{W}_\omega, \widetilde{W}_{\omega+1}) \in \mathcal{E}$ with label $M_{\omega+1}$. We denote the set of edges in $\mathcal{T}(\widetilde{\mathcal{G}})$ by $\mathcal{E}(\mathcal{T}(\widetilde{\mathcal{G}}))$. The partial traversal $\mathcal{T}_\omega(\widetilde{\mathcal{G}})$ of $\mathcal{T}(\widetilde{\mathcal{G}})$ is defined as the subsequence till $\omega \in \mathcal{I}$ (using $\prec$ as the ordering). In $\mathcal{T}(\widetilde{\mathcal{G}})$, a collision $\delta := (u, v; w)$ can be equivalently written as,

$$\delta := (\widetilde{W}_i = u, \widetilde{W}_j = v; \widetilde{W}_{i+1} = \widetilde{W}_{j+1} = w),\ i \prec j \in \mathcal{I},$$

where i and j are the smallest such indices, and recall that $i+1$ and $j+1$ denote the immediate successor of i and j, respectively. Under this equivalent representation we can define a partial order $\prec_\Delta$ on $\Delta(\widetilde{\mathcal{G}})$ as follows:

For $i, j, i', j' \in \mathcal{I}$, $i \prec j$, and $i' \prec j'$, let $\delta = (\widetilde{W}_i, \widetilde{W}_j; \widetilde{W}_{i+1})$ and $\delta' = (\widetilde{W}_{i'}, \widetilde{W}_{j'}; \widetilde{W}_{i'+1})$. $\delta \prec_\Delta \delta'$ if either,

1. $j \prec j'$, or
2. $j = j'$ and $i \prec i'$.

Proposition 4.4.1. *Let $\widetilde{\mathcal{G}}_1, \widetilde{\mathcal{G}}_2 \in \mathsf{Struct}(M^q)$ be two INS graphs and $\mathcal{T}(\widetilde{\mathcal{G}}_1)$ and $\mathcal{T}(\widetilde{\mathcal{G}}_2)$ be their associated traversals. Then,*

$$\forall \omega \in \mathcal{I},\ \mathcal{T}_\omega(\widetilde{\mathcal{G}}_1) = \mathcal{T}_\omega(\widetilde{\mathcal{G}}_2) \iff \widetilde{\mathcal{G}}_1^{\mathcal{E}_\omega} = \widetilde{\mathcal{G}}_2^{\mathcal{E}_\omega},$$

where $\widetilde{\mathcal{G}}_i^{\mathcal{E}_\omega}$ *is the edge induced subgraph of* $\widetilde{\mathcal{G}}_i$ *with edge set* $\mathcal{E}(\mathcal{T}_\omega(\widetilde{\mathcal{G}}_i))$. *Particularly for* $\omega = (q, m_q)$ *we have,*

$$\mathcal{T}(\widetilde{\mathcal{G}}_1) = \mathcal{T}(\widetilde{\mathcal{G}}_2) \iff \widetilde{\mathcal{G}}_1 = \widetilde{\mathcal{G}}_2.$$

Proof. The necessary condition, i.e.,

$$\widetilde{\mathcal{G}}_1^{\mathcal{E}_\omega} = \widetilde{\mathcal{G}}_2^{\mathcal{E}_\omega} \implies \mathcal{T}_\omega(\widetilde{\mathcal{G}}_1) = \mathcal{T}_\omega(\widetilde{\mathcal{G}}_2)$$

is trivially true (by definition of traversals). So we focus on the sufficient condition, i.e.,

$$\mathcal{T}_\omega(\widetilde{\mathcal{G}}_1) = \mathcal{T}_\omega(\widetilde{\mathcal{G}}_2) \implies \widetilde{\mathcal{G}}_1^{\mathcal{E}_\omega} = \widetilde{\mathcal{G}}_2^{\mathcal{E}_\omega}.$$

Note that, $\mathcal{T}_\omega(\widetilde{\mathcal{G}}_1) = \mathcal{T}_\omega(\widetilde{\mathcal{G}}_2) \implies \mathcal{E}(\mathcal{T}_\omega(\widetilde{\mathcal{G}}_1)) = \mathcal{E}(\mathcal{T}_\omega(\widetilde{\mathcal{G}}_2))$. As the two edge sets are equal, the edge induced subgraphs must also be equal. Thus $\forall \omega \in \mathcal{I},\ \widetilde{\mathcal{G}}_1^{\mathcal{E}_\omega} = \widetilde{\mathcal{G}}_2^{\mathcal{E}_\omega}$. □

Definition 4.4.2 (Accident Basis and Dependent Collisions)**.** We define the accident basis $\Delta^{\mathsf{Acc}}(\widetilde{\mathcal{G}})$ of the INS graph $\widetilde{\mathcal{G}}$ as $\mathcal{C} \subseteq \Delta(\widetilde{\mathcal{G}})$ such that $\{L_\delta : \delta \in \mathcal{C}\}$ is the minimal spanning set of $L(\widetilde{\mathcal{G}})$ and the elements of $\mathcal{C}$ are smallest with respect to $\prec_\Delta$. Set $\mathcal{D} \subset \Delta(\widetilde{\mathcal{G}})$ is called a set of dependent collisions if $L(\mathcal{D})$ is linearly dependent. Note that, $\mathsf{Acc}(\widetilde{\mathcal{G}}) = |\Delta^{\mathsf{Acc}}(\widetilde{\mathcal{G}})|$ as $\Delta^{\mathsf{Acc}}(\widetilde{\mathcal{G}})$ is a basis of $L(\widetilde{\mathcal{G}})$.

It is obvious that $\Delta^{\mathsf{Acc}}(\widetilde{\mathcal{G}}^{\mathcal{E}_\omega})$ the accident basis corresponding to the edge induced subgraph $\widetilde{\mathcal{G}}\mathcal{E}_\omega$ (equivalently to the partial traversal $\mathcal{T}_\omega(\widetilde{\mathcal{G}})$) is a subset of $\Delta^{\mathsf{Acc}}(\widetilde{\mathcal{G}})$.

Example 4.3. *Recall example 4.1 and the corresponding INS graph, say* $\widetilde{\mathcal{G}}$, *in Figure 4.1.1(b). For* $\widetilde{\mathcal{G}}$, *we have*

1. $\mathcal{T}(\widetilde{\mathcal{G}}) = (\widetilde{W}_{(1,0)}, \widetilde{W}_{(1,1)}, \widetilde{W}_{(1,2)}, \widetilde{W}_{(1,3)}, \widetilde{W}_{(1,4)}, \widetilde{W}_{(1,5)}, \widetilde{W}_{(2,0)}, \widetilde{W}_{(2,1)})$, *where* $\widetilde{W}_{(1,0)} = (1,0)$, $\widetilde{W}_{(1,1)} = (1,1)$, $\widetilde{W}_{(1,2)} = (1,2)$, $\widetilde{W}_{(1,3)} = \widetilde{W}_{(1,1)}$, $\widetilde{W}_{(1,4)} = \widetilde{W}_{(1,2)}$, $\widetilde{W}_{(1,5)} = (1,5)$, $\widetilde{W}_{(2,0)} = \widetilde{W}_{(1,0)}$, $\widetilde{W}_{(2,1)} = \widetilde{W}_{(1,5)}$.
2. $\Delta(\widetilde{\mathcal{G}}) = \{((1,0),(1,2);(1,1)), ((1,2),(1,0);(1,5))\}$. *The equivalent representation in* $\mathcal{T}(\widetilde{\mathcal{G}})$ *is* $\left\{(\widetilde{W}_{(1,0)}, \widetilde{W}_{(1,2)}; \widetilde{W}_{(1,1)} = \widetilde{W}_{(1,3)}),\ (\widetilde{W}_{(1,4)}, \widetilde{W}_{(2,0)}; \widetilde{W}_{(1,5)} = \widetilde{W}_{(2,1)})\right\}$.
3. $L(\widetilde{\mathcal{G}}) = \left\{\gamma(\beta_{(1,0)}) \oplus \gamma(\beta_{(1,2)}) = 1 \oplus 2; \gamma(\beta_{(1,0)}) \oplus \gamma(\beta_{(1,2)}) = 7 \oplus 4\right\}$.
4. *Clearly,* $\mathsf{Acc}(\widetilde{\mathcal{G}}) = \mathsf{rank}(\widetilde{\mathcal{G}}) = 1$.
5. $\Delta^{\mathsf{Acc}}(\widetilde{\mathcal{G}}) = \{((1,0),(1,2);(1,1))\}$.

.

Proof of Lemma 4.1.11: The proof becomes trivial once we show that each structure graph has a unique accident basis and distinct graphs have distinct accident basis. In other words, we need to show that the mapping from the set of structure graphs to the set of accident basis is injective. It is easy to see that each structure graph has a unique accident basis (by the definition of accident basis). We now show that distinct structure graphs have distinct accident basis.

Claim 4.4.3. *Let $\widetilde{\mathcal{G}}_1$ and $\widetilde{\mathcal{G}}_2$ be two structure graphs. Then,*

$$\Delta^{\mathsf{Acc}}(\widetilde{\mathcal{G}}_1) = \Delta^{\mathsf{Acc}}(\widetilde{\mathcal{G}}_2) \implies \widetilde{\mathcal{G}}_1 = \widetilde{\mathcal{G}}_2.$$

Using Proposition 4.4.1 it is sufficient to show that

$$\Delta^{\mathsf{Acc}}(\widetilde{\mathcal{G}}_1) = \Delta^{\mathsf{Acc}}(\widetilde{\mathcal{G}}_2) \implies \mathcal{T}(\widetilde{\mathcal{G}}_1) = \mathcal{T}(\widetilde{\mathcal{G}}_2).$$

We prove the claim by induction on the dictionary order over the index set $\mathcal{I}$. Let $\omega \in \mathcal{I}$. Suppose $\mathcal{T}_\beta(\widetilde{\mathcal{G}}_1) = \mathcal{T}_\beta(\widetilde{\mathcal{G}}_2)$ for all $\beta \prec \omega$. If $\omega = (r, m_r)$ for some $r \in [q]$, then the next vertex on $\mathcal{T}(\widetilde{\mathcal{G}}_1)$, i.e., $\widetilde{W}^1_{\omega+1} = (1,0) = \widetilde{W}^2_{\omega+1}$, the next vertex on $\mathcal{T}(\widetilde{\mathcal{G}}_2)$. Note that, the superscript $b \in \{1,2\}$ is used to distinguish the vertices of $\widetilde{\mathcal{G}}_1$ and $\widetilde{\mathcal{G}}_2$. Thus, $\mathcal{T}_\omega(\widetilde{\mathcal{G}}_1) = \mathcal{T}_\omega(\widetilde{\mathcal{G}}_2)$. Suppose $\omega = (r,i)$ for some $r \in [q]$ and $i \in [m_r - 1]$. We show that the next edge in $\mathcal{T}(\widetilde{\mathcal{G}}_2)$, i.e., $e_2 := (\widetilde{W}^2_\omega, \widetilde{W}^2_{\omega+1})$ is same as $e_1 := (\widetilde{W}^1_\omega, \widetilde{W}^1_{\omega+1})$, the next edge in $\mathcal{T}(\widetilde{\mathcal{G}}_1)$. The next edge can lead to one of the following cases:

1. Suppose e_1 leads to a dependent collision δ in $\widetilde{\mathcal{G}}_1$. Then, the corresponding linear restriction L_δ must be spanned by $\Delta^{\mathsf{Acc}}(\widetilde{\mathcal{G}}_1^{\mathcal{E}_\omega})$. Now $\mathcal{T}_\omega(\widetilde{\mathcal{G}}_1) = \mathcal{T}_\omega(\widetilde{\mathcal{G}}_2) \implies \Delta^{\mathsf{Acc}}(\widetilde{\mathcal{G}}_1^{\mathcal{E}_\omega}) = \Delta^{\mathsf{Acc}}(\widetilde{\mathcal{G}}_2^{\mathcal{E}_\omega})$. So L_δ is also spanned by $\Delta^{\mathsf{Acc}}(\widetilde{\mathcal{G}}_2^{\mathcal{E}_\omega})$. As the message label is same, we must have $e_2 = e_1$.
2. Suppose e_1 leads to a new accident δ in $\widetilde{\mathcal{G}}_1$. As $\Delta^{\mathsf{Acc}}(\widetilde{\mathcal{G}}_1) = \Delta^{\mathsf{Acc}}(\widetilde{\mathcal{G}}_2)$, $\delta \in \Delta^{\mathsf{Acc}}(\widetilde{\mathcal{G}}_2)$. Thus e_2 also leads to same accident δ in $\widetilde{\mathcal{G}}_2$. Thus $e_1 = e_2$.
3. Suppose e_1 leads to a a new vertex, i.e., $\widetilde{W}^1_{\omega+1} \notin \mathcal{T}_\omega(\widetilde{\mathcal{G}}_1)$. As the labels of both e_1 and e_2 are same, e_2 must also lead to a new vertex. Then using the definition of INS graph we have $e_1 = e_2$.

In all three cases we have $e_1 = e_2$, i.e., $\mathcal{T}_\omega(\widetilde{\mathcal{G}}_1) = \mathcal{T}_\omega(\widetilde{\mathcal{G}}_2)$. This proves the claim. So $|\mathsf{Struct}(M^q)|$ is at most equal to the number of distinct accident basis of size a. Note that, we can have at most m vertices in the graph. So the number of distinct accident basis is at most $\binom{m}{2}^a$. The result follows. □

Corollary 4.1.12 can be easily obtained by combining Lemma 4.1.10 and 4.1.11 followed by some algebraic simplifications.

Chapter 5

Counter-based Input Encoding in MACs

In CRYPTO '95, Bellare et al. [16] proposed a method to construct non-deterministic MACs. The so-called XOR-MAC consisted of a stateful (nonce-based) MAC, XMACC, and a probabilistic MAC, XMACR. At high level, XOR-MAC consists of three sequential steps:

1. Counter-based encoding of input message and parsing as a sequence of encoded blocks;
2. Application of a pseudorandom function (or PRF) F to each of the encoded blocks followed by the accumulation of outputs into a hash value, say h, using simple XOR; and
3. XORing h with $\mathsf{F}'(N)$, where F' is a PRF independent of F, and N is either a nonce (non-repeating internal state) or a random salt.

The original security proof of XOR-MAC is based on the assumption that F and F' are (pseudo)random functions. Later, Bernstein [24] provided an improved analysis when F and F' are (pseudo)random permutations. It can be easily observed that XOR-MAC can be viewed as an instance of Wegman-Carter authenticators or Hash-then-Mask (see section 2.4.2.3), where steps 1 and 2 constitute the universal hashing (see Figure 5.0.1), and step 3 is the finalization. Later, Bernstein proposed a Hash-then-PRF variant for XOR-MAC, called *protected counter sum* (or PCS) [23], which applies a PRF F' on the hash value h in step 3 and returns the output as tag. LightMAC by Luykx et al. [126] is the Hash-then-PRP variant of XOR-MAC, where both F and F' are PRPs. We group all these schemes under a common family, called the XMAC family. All the MAC

schemes in the XMAC family are instances of Hash-then-{Mask/PRP/PRF}. Naturally, the security of XMAC family mainly relies on the universal property of h which in turn depends on the counter-based encoding of inputs. In this chapter, we formalize the notion of counters in symmetric-key settings, and characterize the properties required from them to guarantee security (under reasonable assumptions) for a large class of symmetric-key schemes, including the XMAC family.

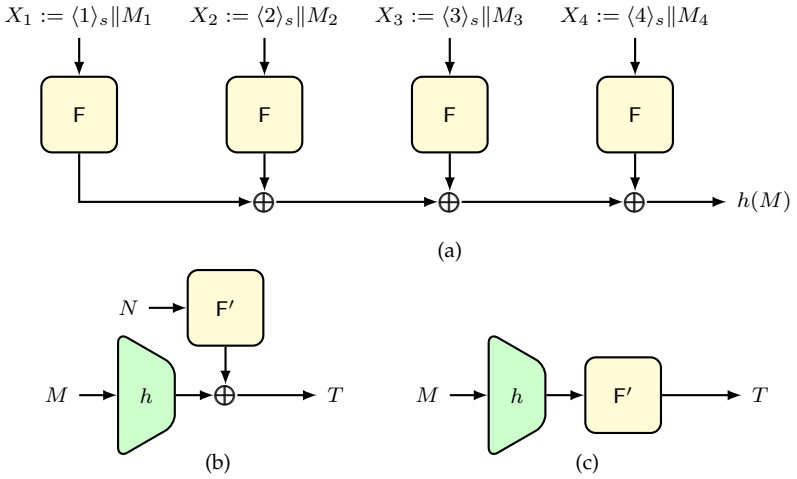

Figure 5.0.1: The XMAC family: (a) hash value computation; (b) finalization used in non-deterministic MACs, i.e., XMACC and XMACR; (c) finalization used in deterministic MACs, i.e., PCS and LightMAC. Here F denotes a PRF (or PRP).

5.1 Inefficiency of Classical Counters

Counter-based schemes can be classified into two categories: (1) *Counter-as-Input* or CaI, where the counter values are used as a standalone input to the underlying primitive; and (2) *Counter-as-Encoding* or CaE, where the counter values are encoded along with data bits within the input to the underlying primitive. Schemes based on CTR mode [153], such as GCM [130] and SIV [172], fall under the former category, whereas the XMAC family falls under the later category. Here we will concentrate only on CaE schemes.

Classical counter-based encodings are based on a fixed length counter. Let us fix an integer $s \in \mathbb{N}$ as the *counter size* and $L := 2^s$. Now, given a message $M \in \{0,1\}^\ell$ for some $\ell \leq L$, the encoding works as follows: Let $M' = M \| 1 \| 0^r$ where r is the smallest integer such that $n - s \mid |M'|$. Suppose $M' = (M_1, M_2, \ldots, M_b)$, such that for all $i \in [b]$, M_i is of bit-length $n - s$. Then $X := (X_1 := \langle 1 \rangle_s \| M_1, \ldots, X_b := \langle b \rangle_s \| M_b)$ is the counter-based encoded input corresponding to M.

The choice of s may depend on the nature of the application. More precisely, s should be chosen in such a way that the number of encoded blocks generated for the longest permissible message should not be more than 2^s, to avoid the possible reseting of counter values.

A typical choice for the maximum permissible message length would allow 2^{64} encoded blocks, i.e., s is exactly 64 bits. Suppose AES-128 cipher is used to instantiate LightMAC [126] scheme. Then, each call of AES-128 will process 64 bits of message, as each encoded block contains 64 bits of counter value and 64 bits of message. So, it would take 2 AES calls per 128 bits of message, independent of whether the message size is 1 kB (2^{13} bits), or 1 GB (2^{33} bits). In short, for a 1 kB message, LightMAC makes about 128 AES-128 calls. On the other hand when $s = 8$, the number of AES-128 calls reduces to 69 (almost twice faster). But this limits the maximum number of encoded blocks to 2^8. So, a classical counter cannot be efficient over a wide range of message lengths.

Contrary to fixed length counters, one can simply choose the best choice of s for the given message length, instead of fixing it throughout all the invocations. This will definitely speed up the performance for shorter messages and provide very similar performance when the message size approaches the maximum size. A disadvantage of the aforementioned message length dependent counter scheme is the requirement of prior knowledge of the message length. In many scenarios this may not be possible. What can be a good approach when the message length is not known beforehand?

Let us consider an analogous problem from athletics. Consider a race over an unknown distance, where the athletes would only know the finishing point once they actually reach there. What should be a good strategy for running this race? One may consider to run like a marathoner hoping that it is a marathon. In this case, clearly the athlete loses if the race is a hundred meter sprint. The better solution would be to start like a sprinter and then gradually reduce the speed as the race progresses. This could be optimal for short runs.

Our problem is similar to the one just described. We want to find a variable length counter that offers near optimal counter sizes for each message length. Although similar problems are well-studied in the field of prefix codes [92, 175] and data compression [179], to the best of our knowledge it has not seen any interest in cryptography. It would be interesting to investigate the techniques used in prefix coding and construct a near optimal counter scheme when the message length is not known.

Although these ideas are very natural, so far they have not been applied in any of the algorithms in the XMAC family. One possible reason is that the security guarantee is

not obvious for variable length messages. In [16], the authors have given a general framework for constructing XOR-MAC. But even this general framework is somewhat lacking and does not give a unified yet simple treatment for all possible counter-based encodings. In the main theorem of [23], it is assumed that given a message, the encoding produces distinct encoded blocks for distinct primitive calls. Although this points towards a general requirement from any counter-based encoding, the indication is rather implicit. Thus, a formal treatment of counters is required. Furthermore, this generalization should allow a generic proof technique that applies to all counter-based encodings.

5.2 A New Look at Counters

In computer science, counters are used to count the number of times a particular event or process has occurred. In addition to counting, counters have a very special role in cryptography. In XMAC family, the counter values are used to encode the message into distinct blocks. This freshness of inputs actually helps in (possibly improved) the security proof. The classical encoding of i is $\langle i \rangle_s$ which is the s-bit unsigned binary representation of i for a fixed parameter s. Whenever $i < 2^s$, we have distinct binary string corresponding to each counter value. For counter-based algorithms, the parameter s can be chosen as per the needs of the application domain. For example, if we know beforehand that for an application, the message size can not be more than 2^{32}, then one can choose $s = 32$. However, it must remain constant throughout all the executions of the algorithm. This might affect the performance for smaller length messages. In this section we introduce a new and general way of looking at counters which will provide some tools to improve the performance of these schemes without compromising with the security.

Throughout this chapter, we use a fixed $L \in \mathbb{N}$ to denote the maximum permissible bit-length. We will use $\widehat{\ell}$ to denote the block length corresponding to bit-length ℓ, i.e. $\widehat{\ell} := \lceil \frac{\ell}{n} \rceil$.

Definition 5.2.1 (Counter function family)**.** A counter function family (we also use CFF) CNTR is a family of counter functions $\{\mathsf{cntr}_\ell : \ell \leq L\}$, where for all $\ell \leq L$, $\mathsf{cntr}_\ell : \mathbb{N} \to \cup_{b \in (n-1]} \{0,1\}^b$.

We say that CNTR is *prefix-free* if for all $\ell \leq L$, cntr_ℓ is prefix-free (i.e., for all $i \neq j$, $\mathsf{cntr}_\ell(i)$ is not a prefix of $\mathsf{cntr}_\ell(j)$).

We use capital letters to denote a function family and small letters for individual functions. The classical (or standard) encoding, as mentioned above, can be viewed as a

CFF STD^s, where $\mathsf{std}^s_\ell(i) = \langle i \rangle_s$, for a fixed positive integer $s < n$. Note that, the standard counter function std_ℓ is actually independent of ℓ and hence for all ℓ, the counter functions are same. We call such CFFs *message length independent*. For a message length independent CFF CNTR, we simply write cntr to denote the counter function cntr_ℓ. Note that, the standard counter is a prefix-free counter. Prefix-free CFF is necessary to avoid repetitions among the inputs to the primitive (see Lemma 5.2.3 below). We also note that the size of the output of std_ℓ is fixed for all counter values. In general for a CFF CNTR, if $\forall \ell,\ \forall i, j\ \ |\mathsf{cntr}_\ell(i)| = |\mathsf{cntr}_\ell(j)|$, then we say that the CFF has *fixed size*. Otherwise, we call it a *variable size CFF*. We will see some examples of message length dependent and variable size CFFs later in the section.

5.2.1 Counter Function Family Based Message Encoding

Recall the encoding scheme discussed in section 5.1. When we use the standard counter STD^s with $(n-s)2^s \leq L$, we first parse a message $M \in \{0,1\}^\ell$ as $(M_1, \ldots, M_{b-1}, M'_b)$ where $b = \left\lceil \frac{\ell+1}{n-s} \right\rceil$, $|M_1| = \cdots = |M_{b-1}| = n - s$ and $0 \leq |M'_b| < (n-s)$ such that $M = M_1 \| \cdots \| M_{b-1} \| M'_b$. Let $M_b = M'_b \| 10^d$ where $d = n-s-1-|M'_b|$. Thus, $|M_b| = n-s$. Now, we define the b blocks encoding as $X_i = \mathsf{std}^s(i) \| M_i$ for all $i \in [b]$. These blocks are used as inputs to the underlying primitive, such as block cipher or compression function.

We extend the same methodology to define the encoding for any other CFF CNTR. For this, we first define the block function $b^{\mathsf{CNTR}}(\ell)$, which associates each message size to a unique number of blocks. We have seen that for standard counter, $b^{\mathsf{STD}}(\ell) = \lceil (\ell+1)/(n-s) \rceil$.

Definition 5.2.2 (Block function and Average counter length)**.** For any CFF CNTR, we define the *block function* $b^{\mathsf{CNTR}}(\ell)$ as the least integer b such that

$$\ell + 1 \leq \sum_{i=1}^{b} (n - |\mathsf{cntr}_\ell(i)|) \leq \ell + n.$$

We define the *average counter length* $\mu^{\mathsf{CNTR}}(\ell)$ as

$$\mu^{\mathsf{CNTR}}(\ell) := \frac{1}{b^{\mathsf{CNTR}}(\ell)} \sum_{i=1}^{b^{\mathsf{CNTR}}(\ell)} |\mathsf{cntr}_\ell(i)|.$$

It is easy to see that $\mu^{\mathsf{CNTR}}(\ell) \in (n-1]$ and $b^{\mathsf{CNTR}}(\ell) = \lceil (\ell+1)/(n - \mu^{\mathsf{CNTR}}(\ell)) \rceil$. Now given a message $M \in \{0,1\}^\ell$, $\ell \leq L$, we parse it as $M = M_1 \| \cdots \| M_{b-1} \| M'_b$ where $|M_i| = n - |\mathsf{cntr}_\ell(i)|$ for all $i < b$ and $0 \leq |M'_b| < n - |\mathsf{cntr}_\ell(b)|$. We similarly pad the last

block to make it compatible with the corresponding counter. More precisely, we define $M_b = M'_b \| 10^d$ where $d = n - |\mathsf{cntr}_\ell(b)| - 1 - |M'_b|$. Thus, $|M_b| = n - |\mathsf{cntr}_\ell(b)|$. We denote the parsing procedure as $(M_1, \ldots, M_b) \xleftarrow{\mathsf{CNTR}} M$. Finally, we define the encoding

$$\mathsf{CNTR}(M) := (X_1, \ldots, X_b),$$

where $X_i = \mathsf{cntr}_\ell(i) \| M_i$ for all $i \in [b]$. Note that, we abuse the notations slightly to view the CFF as an encoding function $\mathsf{CNTR} : \cup_{\ell \in [L]} \{0,1\}^\ell \to \mathbb{B}^+$.

Recall that one of the main purpose of counters in cryptography is to generate distinct blocks. Mathematically, a CFF CNTR is called *block-wise collision-free* if for all $M \in \{0,1\}^\ell$, $\ell \leq L$, $\mathsf{CNTR}(M) = X^b \in (\mathbb{B})_b$, i.e., X_i's are distinct for all $i \in [b]$. Now, we provide a characterization of block-wise collision-free counters.

Lemma 5.2.3. CNTR *is block-wise collision-free if and only if it is prefix-free.*

Proof. We first prove the "only if" direction. Suppose there exists $i < j \leq b$ such that $\mathsf{cntr}_\ell(i)$ is a prefix of $\mathsf{cntr}_\ell(j)$. Thus, we can find x and y such that $\mathsf{cntr}_\ell(i) \| x = \mathsf{cntr}_\ell(j) \| y \in \mathbb{B}$. Let $(M_1, \ldots, M_b) \xleftarrow{\mathsf{CNTR}} M \in \{0,1\}^\ell$, for some $\ell \in [L]$. Then, we define M' by replacing M_i and M_j in M by x and y, respectively. It is easy to see that $X'_i = X'_j$, where $\mathsf{CNTR}(M') = (X'_1, \ldots, X'_b)$.

To prove the other direction, let us assume $X_i = X_j$ for some $i \neq j$. Therefore, $\mathsf{cntr}_\ell(i) || M_i = \mathsf{cntr}_\ell(j) || M_j$. Clearly, either $\mathsf{cntr}_\ell(i)$ is a prefix of $\mathsf{cntr}_\ell(j)$, or $\mathsf{cntr}_\ell(j)$ is a prefix of $\mathsf{cntr}_\ell(i)$. □

5.2.2 Some (Efficient) Alternatives to the STD CFF

We have already demonstrated the classical or standard CFF STD^s. It is a message-length independent and fixed size CFF. In this section, we study two new examples of CFFs and see their advantages over the standard one.

5.2.2.1 OPT: A Message Length Dependent CFF

Our first CFF is just an optimization of the classical counters. In this case, the counter function maps any integer i to the s-bit unsigned bit representation of its argument, where s depends on the message length instead of a pre-determined fixed parameter. In fact, s is defined as the smallest possible value such that we can represent all the

counter values distinctly for the given message length. Formally, the counter function is defined as

$$\mathsf{opt}_\ell(i) = \langle i \rangle_s, \text{ where } s = \min\{g : 2^g(n-g) > \ell\}.$$

It is an example of message-length dependent and fixed size CFF. Clearly, it is a prefix-free counter.

5.2.2.2 VAR^r: A Variable Size CFF

Till now we have seen counter functions which map integers to their s-bit unsigned binary representation. In case of STD^s, s is fixed (approx. $\log_2 L$), whereas in case of OPT, s depends on the message length ℓ (approx. $\log_2 \ell$). Both STD^s and OPT are fixed size CFFs. Now we will construct a CFF which maps integers to binary strings of monotonically increasing lengths. A similar problem is well-known in the field of source coding. We briefly discuss these techniques assuming that the set of symbols is $\mathbb{N}$. Readers may refer to the references cited for a more detailed exposition.

PREFIX CODES: A mapping $\alpha : \mathbb{N} \to \{0,1\}^*$, is called a binary prefix code [175] over integers, if for all $x \neq y \in \mathbb{N}$, $\alpha(x)$ is not a prefix of $\alpha(y)$ and vice-versa. Here $\alpha(x)$ is called the codeword corresponding to x. Huffman codes [92, 175], Shannon-Fano codes [68, 175, 179] and Universal codes [66, 175] are some popular techniques for getting prefix codes.

Huffman codes and Shanon-Fano codes are in general better than universal codes, when the probability (or frequency) distribution over the integers is known. This is generally the case in static data compression settings, such as JPEG File Interchange Format [96] which employs a modified form of Huffman coding, and the ZIP file format [164] which uses a modified version of Shannon-Fano coding. Although Huffman codes and Shannon-Fano codes are better than universal codes, but they require exact probability (or frequency) distribution over the integers.

UNIVERSAL CODES: A universal code [66, 67, 175] is a binary prefix code, with the additional property that if the probability (or frequency) distribution over integers is monotonic, then the expected lengths of the codewords are within a constant factor of the optimal expected lengths for that distribution. Note that, a universal code works for any countably infinite set, and it only requires a monotonic distribution. This is one of the main motivations behind the study of universal codes. Our problem falls precisely in this case, as we can always assume a natural monotonic distribution over the number of blocks. For instance all messages must have at least one block, so 1 has the maximum frequency, followed by 2, and so on.

In 1974, Elias proposed the first prefix code with universal property [66]. In 1975, he followed it up by proposing several new examples of universal codes [67]. Some of the popular examples of universal codes are Elias gamma coding [67, 175], Elias delta coding [67, 175], Elias omega coding [67, 175], Fibonacci coding [73, 175], Levenshtein coding [175] and Exp-Golomb coding [175]. We describe two examples of universal codes, namely, Elias gamma and delta codings, related to our work.

Elias Gamma Coding, γ: A number $i \geq 1$ is encoded as follows:

1. Let $j = \lfloor \log_2 i \rfloor$, i.e., $2^j \leq i < 2^{j+1}$.
2. $\gamma(i) := 0^j \| \langle i \rangle_{j+1}$.

Elias Delta Coding, δ: A number $i \geq 1$ is encoded as follows:

1. Let $j = \lfloor \log_2 i \rfloor$, i.e., $2^j \leq i < 2^{j+1}$.
2. $\delta(i) := \gamma(j+1) \| \mathsf{lsb}_j(\langle i \rangle_{j+1})$.

To represent an integer i, Elias gamma uses $2\log_2 i + 1$ bits, and Elias delta uses $\log_2 i + 2\log_2(\log_2 i + 1) + 1$ bits. Clearly, Elias delta is more compact as compared to Elias gamma.

CFF FROM UNIVERSAL CODING: Theoretically, any universal code U can be used to construct a length-independent variable size prefix-free CFF U-CTR. Suppose U is a universal code, then we define U-CTR as follows:

$$\mathsf{U}\text{-CTR} = \{\mathsf{u}\text{-ctr}_\ell : \ell \leq L\} \text{ where } \mathsf{u}\text{-ctr}_\ell = \mathsf{U},\ 1 \leq \ell \leq L.$$

Clearly, U-CTR is prefix-free as U is universal. Although one might be tempted to use U-CTR straightaway, there is still some scope of improvement in this generic scheme. Note that, the motivation for universal coding is quite different than the cryptographic settings.

First, we are restricting the domain to some L, whereas a universal code is defined over $\mathbb{N}$. Intuitively it seems that the universal codes must have some redundant information that we can avoid. Second, all the CFFs discussed so far have efficient counter update (generating $\mathsf{u}\text{-ctr}_\ell(i+1)$ from $\mathsf{u}\text{-ctr}_\ell(i)$) mechanism, which is not a mandatory requirement for universal codes. So we should restrict our focus to only those universal codes which support efficient update mechanism, such as Elias delta code. Now we present our second CFF candidate, called VAR^r, which is a modified version of δ-CTR.

VAR^r COUNTER FUNCTION FAMILY: We first fix r, an application parameter chosen suitably (we will see very soon how to choose r). As VAR^r would be message length independent, it would be sufficient to define a function over the domain $\{1, 2, \ldots, L\}$. To begin with, we define $\mathsf{var}^r(0) = 0^r$. Now, for all $i \leq L$, we will recursively define $\mathsf{var}^r(i)$ given that $\mathsf{var}^r(j)$ has been defined for all $j < i$. Like STD^s and OPT, we increment the counter by one at every step. In other words, we first define $y = \mathsf{var}^r(i-1) + 1$. If the r most significant bits get altered (this is easy to check, as the remaining bits would all be zero), then we define $\mathsf{var}^r(i) = y \| 0$, otherwise we define $\mathsf{var}^r(i) = y$. Mathematically, we can write the recursive definition of var^r for $i \geq 1$, as

$$\mathsf{var}^r(i) = \begin{cases} x+1 & \textit{if } \mathsf{msb}_r(x) = \mathsf{msb}_r(x+1), \\ (x+1)\|0 & \textit{if } \mathsf{msb}_r(x) \neq \mathsf{msb}_r(x+1). \end{cases}$$

where $x = \mathsf{var}^r(i-1)$.

Clearly, size of the counter function output increases slowly but monotonically with i. So, VAR^r is an example of message length independent and variable size CFF. To understand this counter we demonstrate how the counter values are computed for $r = 3$. A boldface zero at the least significant bit represents the added zero bit on expansion. The underlined bits are the third most significant bits.

$$00\underline{0}, 00\underline{1}\mathbf{0}, 00\underline{1}1, 01\underline{0}0\mathbf{0}, 01\underline{0}01, 01\underline{0}10, 01\underline{0}11, 01\underline{1}00\mathbf{0}, \ldots$$

Now we will describe how one should choose the parameter r given that the limit on the message length is L. Note that, when the size of the counter reaches $r+i$ for the first time, the i least significant bits of the counter value are all zero. So the counter will be incremented 2^i many times before the next expansion in counter size. Since we need to keep the r most significant bits non-zero (to avoid repetitions), the following equation must be satisfied:

$$\sum_{i=0}^{2^r-1} (n-r-i)2^i > L.$$

After doing some algebraic simplifications one can show that the smallest r that satisfies the above equation is approx. $\log_2 \log_2 L$ for $L < 2^{c(n)\cdot n}$, where $\frac{1}{2} \leq c(n) < 1$. Note that, for any integer i,

$$\mathsf{var}^r_\ell(i) = \langle \lfloor \log_2 i \rfloor \rangle_r \| \mathsf{lsb}_{\log_2 i}(\langle i \rangle_{\log_2 i + 1}),$$

whereas

$$\delta(i) = \gamma(\lfloor \log_2 i + 1 \rfloor) \| \mathsf{lsb}_{\log_2 i}(\langle i \rangle_{\log_2 i + 1}).$$

Clearly $|\mathsf{var}^r_\ell(i)|$ is less than $|\delta(i)|$ when

$$i \geq 2^{2^{\frac{r-1}{2}}-1}.$$

We generally fix the maximum message length to be 2^{64} bits, which gives $r \approx 6$. So, the average counter size in VAR^r will be less than the average counter size in δ-CTR, when the number of generated blocks is at least 2^6. Specifically, for around 2^{16} blocks, the average counter size in VAR^r is shorter than the average counter size in δ-CTR by more than 3 bits. Now we show that VAR^r is a prefix-free CFF.

Theorem 5.2.4. *Let r be defined as above then VAR^r is prefix-free.*

Proof. Let $i \neq j$ be non-zero and $x = \mathsf{msb}_r(\mathsf{var}^r(i))$ and $y = \mathsf{msb}_r(\mathsf{var}^r(j))$. If $x \neq y$ then x can not be prefix of y. So assume $x = y$. Because of our choice of r, the r most significant bits does not become 0^r (except for the input 0). So, $\mathsf{var}^r(i)$ and $\mathsf{var}^r(j)$ have same size. As i and j are distinct, the rest of the bits must be different. This proves the prefix-free property. □

5.2.2.3 Word Oriented Adaptation of Our Counters

The counter functions described in the preceding section are aimed to minimize the counter size as much as possible, keeping all the counter values distinct (i.e., prefix-free). However, there are different practical issues while implementing these counter functions. The most important issue is to parse the message into small chunks which are compatible with the counter-based encoding. In practice, we mostly receive messages as a sequence of words (e.g., 8-bit (byte), or 32-bit words). It would be easier for implementation if we can parse the messages in multiples of word size. More formally, let us fix a parameter w (elements of $\{0,1\}^w$ are called words). We define OPT^w (the word oriented adaption of OPT) as follows:

$$\mathsf{opt}^w_\ell(i) = \langle i \rangle_s, \text{ where } s = \min\{g : w|g,\ (n-g)2^g > \ell\}.$$

Note that, $\mathsf{OPT} = \mathsf{OPT}^1$. Now we describe how we can generalize our second proposal to $\mathsf{VAR}^{r,w}$ which fits into word oriented implementation, for $r \leq w$. We define the counter function recursively as before except that we add 0^w instead of single 0. More formally, $\mathsf{var}^{r,w}(0) = 0^w$. For $i \geq 1$, let $x = \mathsf{var}^{r,w}(i-1)$. Then,

$$\mathsf{var}^{r,w}(i) = \begin{cases} x+1 & \text{if } \mathsf{msb}_r(x) = \mathsf{msb}_r(x+1), \\ (x+1)\|0^w & \text{if } \mathsf{msb}_r(x) \neq \mathsf{msb}_r(x+1). \end{cases}$$

For this choice of r, the counter function is prefix-free. In the following example we describe how the counter expands for $r = 4$ and $w = 8$.

$$000\underline{0}0000, \ldots, 000\underline{1}0000\mathbf{0^8}, \ldots, 001\underline{0}00000000000\mathbf{0^8}, \ldots$$

To the best of our knowledge such word-oriented definitions are not available for delta codes (for obvious reasons). Even if we use our word-oriented definition, the resulting code will not give better counter function. This can be argued by the simple fact that in case of $\mathsf{VAR}^{r,w}$, r is fixed, so we can simply start with some fixed w and then move as it is. But in case of delta code we have to consider the underlying gamma code also which is variable in nature. Handling two variable components may result in overheads in the counter size as well as the update mechanism.

5.2.3 Comparison of Rates of Counter Function Families

Recall that we have defined the number of blocks in a message of length ℓ as $\widehat{\ell} = \lceil \ell/n \rceil$. As we execute some costly primitive function on these blocks, it would be good to minimize the number of blocks as much as possible.

Definition 5.2.5. We define the rate function $\mathsf{rate}^{\mathsf{CNTR}}(\ell)$ associated with a counter-based encoding CNTR, as the ratio of the number of blocks in the message to the total number of blocks in the encoded message produced by CNTR, i.e., $\mathsf{rate}^{\mathsf{CNTR}}(\ell) = \frac{\widehat{\ell}}{b^{\mathsf{CNTR}}(\ell)}$.

Now, $\widehat{\ell} = \lceil \ell/n \rceil \leq \lceil (\ell+1)/n \rceil \leq \lceil (\ell+1)/(n-k) \rceil$ for any $k \in (n-1]$. In particular, we have $\widehat{\ell} \leq \lceil (\ell+1)/(n-\mu^{\mathsf{CNTR}}(\ell)) \rceil = b^{\mathsf{CNTR}}(\ell)$. Thus, we have $\mathsf{rate}^{\mathsf{CNTR}}(\ell) \leq 1$, where equality holds when n divides $(\ell+1)$ and the counter size is zero. Now we provide a comparison between the rate functions for the three CFFs defined above.

1. For the standard counter STD^s, the rate function is

$$\frac{\lceil \ell/n \rceil}{\lceil (\ell+1)/(n-s) \rceil} \approx \frac{n-s}{n},$$

   for $\ell \gg n > s$. Typically, $s \approx \log_2 L$, which reduces the rate when L is large.

2. For OPT, the rate function is

$$\frac{\lceil \ell/n \rceil}{\lceil (\ell+1)/(n-\log_2 \ell) \rceil} \approx \frac{n-\log_2 \ell}{n}.$$

   Clearly, the rate of this counter is better than STD^s for all choices of $\ell < L$.

3. For VAR^r, the rate function is a complex function in n and r. We skip the exact algebraic expression here and give an approximation,

$$\frac{\lceil \ell/n \rceil}{\lceil (\ell + n - r + 2)/(n - r + 2 - \log_2 \ell) \rceil} \approx \frac{n - r + 2 - \log_2 \ell}{n} \quad \Big(\text{for } \ell \gg n\Big).$$

Clearly, the rate of this counter is better than STD for small messages and large s. Further the rate is comparable with OPT for $n \gg r$.

5.2.4 Word Oriented Rates of Counter Function Families

Let $w \in \mathbb{N}$ denote the word size. We will assume that $n, \ell, L \in \mathbb{M}_w$, where $\mathbb{M}_w$ denotes the set of all multiples of w. We assume that $L = o(2^n)$. The word oriented rate functions for STD^s and OPT are pretty similar to their bit oriented counterparts. For instance, in STD^s, we can simply choose s to be some multiple of w. So we only derive the word oriented rate function for VAR^r.

5.2.4.1 Estimating Parameter r in $\mathsf{VAR}^{r,w}$

Let $c = \lceil r/w \rceil$. We know that the following inequality holds for the correct value of r,

$$\sum_{i=0}^{2^r - 1} (n - cw - iw) 2^{iw + cw - r} \geq L$$

Let $n' = n - cw$, $r' = cw - r$, $\dot{r} = 2^r$, and $\dot{w} = 2^w$. Now we will get a lower bound on r,

$$\begin{aligned}
\sum_{i=0}^{\dot{r}-1} (n' - iw) 2^{iw + r'} &\geq L \\
2^{r'} \left(n' \sum_{i=0}^{\dot{r}-1} \dot{w}^i - w \sum_{i=0}^{\dot{r}-1} i \cdot \dot{w}^i \right) &\geq L \\
2^{r'} \left[\left(n' + \frac{w\dot{w}}{\dot{w} - 1} \right) \left(\frac{\dot{w}^{\dot{r}} - 1}{\dot{w} - 1} \right) - \frac{w\dot{r}\dot{w}^{\dot{r}}}{\dot{w} - 1} \right] &\overset{1}{\geq} L \\
2^{r'} \left[\left(n' + 2w \right) \left(2\dot{w}^{\dot{r}-1} \right) - w\dot{w}^{\dot{r}-1} \right] &\overset{2}{\geq} L \\
\dot{w}^{\dot{r}} \left(2n - w \right) &\overset{3}{\geq} 2L
\end{aligned}$$

where 1 can be obtained by simple algebraic simplifications. As $\dot{w} \geq 2$, we have $\frac{1}{\dot{w}-1} \leq \frac{2}{\dot{w}}$. Using this we get 2. Also $c \geq 1$ and $r' \leq w - 1$, which gives us 3. Now simple

algebraic simplifications give us the upper bound on r,

$$r \geq \log_2 \left\lceil \frac{\log_2 \left(\frac{2L}{2n-w} \right)}{w} \right\rceil \tag{5.1}$$

For $w < n \ll L$, we get $r \approx \log_2 \log_2 L - \log_2 w$.

5.2.4.2 Estimating the Block Function of $\mathsf{VAR}^{r,w}$

For a given message length ℓ, we are interested in a lower bound on $b(\ell)$.[1] For simplicity we also assume $c = r/w$, i.e., $r' = 0$. We restrict our analysis to only those ℓ, for which $\exists\, k \leq 2^r - 1$ such that the number of blocks,

$$b(\ell) = 2^{r'} \sum_{i=0}^{k-1} \dot{w}^i = \left(\frac{\dot{w}^k - 1}{\dot{w} - 1} \right). \tag{5.2}$$

We find an upper bound for $b(\ell)$ in the following derivation,

$$\begin{aligned}
\left(n' \sum_{i=0}^{k-1} \dot{w}^i - w \sum_{i=0}^{k-1} i \cdot \dot{w}^i \right) &\leq \ell + n \\
\left(n' + \frac{w\dot{w}}{\dot{w}-1} - \frac{wk\dot{w}^k}{\dot{w}^k - 1} \right) \left(\frac{\dot{w}^k - 1}{\dot{w} - 1} \right) &\overset{1}{\leq} \ell + n \\
(n - r + w - wk) \left(\frac{\dot{w}^k - 1}{\dot{w} - 1} \right) &\overset{2}{\lesssim} \ell + n \\
(n - r + w - \log_2(\ell + n)) \left(\frac{\dot{w}^k - 1}{\dot{w} - 1} \right) &\overset{3}{\lesssim} \ell + n.
\end{aligned}$$

$\frac{1}{\dot{w}-1} \geq \frac{1}{\dot{w}}$, and for moderately large k, we get $\frac{\dot{w}^k}{\dot{w}^k - 1} \approx 1$. Using these facts we get 2 from 1. Similarly $wk \lesssim \log_2(\ell + n)$ (experimental results show that $wk \approx \log_2 \ell$), which gives 3 from 2. Using (5.2) and 3 we get,

$$b(\ell) \lesssim \frac{\ell + n}{n - r + w - \log_2(\ell + n)} \tag{5.3}$$

RATE FUNCTION FOR WORD ORIENTED $\mathsf{VAR}^{r,w}$: Finally we get the following approximation for lower bound on the rate function of word oriented $\mathsf{VAR}^{r,w}$,

$$\mathsf{rate}^{\mathsf{VAR}^{r,w}}(\ell) \gtrsim \frac{n + w - r - \log_2(\ell + n)}{n}$$

for $w < n \ll \ell$. Note that, we derived this function for some restricted ℓ values. But experimental results, given in Table 5.2.1, shows that the estimation is close to the actual

[1] Note that, we have dropped $\mathsf{VAR}^{r,w}$ from the superscript as it is obvious.

function. Figure 5.2.1 gives a graphical comparison between the rates of the three CFFs.

Table 5.2.1: Comparison between the rates offered by the three candidate CFFs for block size, $n = 128$ and $w = 8$.

Length	STD^s				$\mathsf{STD}^{opt,8}$	$\mathsf{VAR}^{4,8}$
	s = 8 bits	s = 16 bits	s = 32 bits	s = 64 bits		
128B	0.89	0.80	0.73	0.47	0.89	0.89
256B	0.89	0.84	0.73	0.48	0.89	0.89
512B	0.91	0.86	0.74	0.49	0.91	0.89
1kB	0.93	0.86	0.74	0.50	0.93	0.88
2kB	0.93	0.87	0.75	0.50	0.93	0.88
4kB	-	0.87	0.75	0.50	0.87	0.88
8kB	-	0.87	0.75	0.50	0.87	0.88
16kB	-	0.87	0.75	0.50	0.87	0.88
32kB	-	0.87	0.75	0.50	0.87	0.88
64kB	-	0.87	0.75	0.50	0.87	0.86
128kB	-	0.87	0.75	0.50	0.87	0.80
256kB	-	0.87	0.75	0.50	0.87	0.77
512kB	-	0.87	0.75	0.50	0.87	0.76
1MB	-	-	0.75	0.50	0.75	0.76

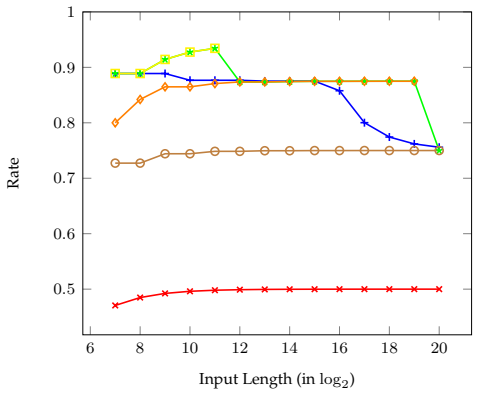

Figure 5.2.1: Rate plot for the three CFFs. Plots marked with $+$, $\star$, $\times$, $\circ$, $\diamond$, and $\square$ marked plots correspond to $\mathsf{VAR}^{4,8}$, OPT^8, STD^{64}, STD^{32}, STD^{16}, and STD^8, respectively.

5.3 Counter-as-Encoding Constructions

We propose some generic CaE schemes, such as AXU hash and MACs based on CFFs. We also propose a HAIFA [31] based cryptographic hash function. In the last section, we defined prefix-free counter function family. For any such prefix-free counter CNTR and for all M, we know that $X_1, \ldots, X_b$ are distinct where $\mathsf{CNTR}(M) = (X_1, \ldots, X_b)$. With a slight abuse of notation, we let $\mathsf{CNTR}(M)$ to denote the set $\{X_1, \ldots, X_b\}$ as well. Observe that prefix-free property does not say anything about the collisions between the encoded blocks of two distinct messages. Clearly, we will have some intersection between $\mathsf{CNTR}(M)$ and $\mathsf{CNTR}(M')$ (viewed as a set) for any counter function family.

Definition 5.3.1. A prefix-free CFF CNTR is called *injective* if for all $M \neq M'$, $\mathsf{CNTR}(M) \neq \mathsf{CNTR}(M')$ (as a set).

We provide a sufficient condition for injective counter function families.

Lemma 5.3.2. *Let* CNTR *be a prefix-free CFF. It is injective if it satisfies the following condition*

$$\forall \ell, \ell' \leq L, \;\; b(\ell) = b(\ell') \Rightarrow \mathsf{cntr}_\ell = \mathsf{cntr}_{\ell'}.$$

Proof. Suppose for some $M \in \{0,1\}^\ell, M' \in \{0,1\}^{\ell'}$, $\mathsf{CNTR}(M) = \mathsf{CNTR}(M')$. We require to show that $M = M'$. Note that, $\mathsf{CNTR}(M) = \mathsf{CNTR}(M') \Rightarrow \{X_1, \ldots, X_{b(\ell)}\} = \{X'_1, \ldots, X'_{b(\ell')}\}$. So $b(\ell) = b(\ell') = b$. By given condition, we have $\mathsf{cntr}_\ell = \mathsf{cntr}_{\ell'}$ and we denote it simply by cntr. Now, we first show that $X_i = X'_i$ for all i. As two sets are equal for any $i \leq b$, there must exist j so that $X_i = X'_j$. This implies that one of $\mathsf{cntr}(i)$ and $\mathsf{cntr}(j)$ is prefix of the other, and hence $i = j$ (as the counter function is block-wise collision-free). Since counter functions for ℓ and ℓ' are same we have $M_i = M'_i$ for all i. This proves that $M = M'$. □

Recall that all our candidate CFFs are prefix-free. Further it is obvious to see that they also satisfy the condition for injectivity. So we get the following corollary.

Corollary 5.3.3. $\mathsf{CNTR} \in \{\mathsf{STD}, \mathsf{OPT}, \mathsf{VAR}^r\}$ *is a prefix-free and injective CFF.*

In the following subsections we present various schemes based on CFFs. In particular, the MAC schemes with STD are precisely the MACs from the XMAC family. The security of all these schemes follow directly from the prefix-free and injective property of the underlying CFF. Specifically, all the security results in the following subsections hold, when we instantiate these schemes with $\mathsf{CNTR} \in \{\mathsf{STD}, \mathsf{OPT}, \mathsf{VAR}^r\}$.

5.3.1 Almost XOR Universal Hash Function

Let CNTR be a CFF. We define a counter based AXU-hash function, denoted CtH, based on a length-preserving[2] function Λ over $\mathbb{B}$, and a CFF CNTR. Let $M \in \{0,1\}^\ell$. We define

$$\mathsf{CtH}_{\Lambda,\mathsf{CNTR}}(M) = \Lambda(X_1) \oplus \cdots \Lambda(X_b),$$

where $\mathsf{CNTR}(M) = (X_1, \ldots, X_b)$. Notice that this hash function is a generalization of the hash functions used in the XMAC family [16, 23, 126]. It has a simple design that requires only Λ computation and XOR operations. Furthermore, the hash can be

[2] A function f is called length-preserving if $|f(x)| = |x|$, for all x.

computed completely in parallel. Now we show that CtH is an AXU hash function whenever Λ is a uniform random permutation or function over $\mathbb{B}$.

Theorem 5.3.4. *Let* CNTR *be a prefix-free and injective CFF and* $b^{\mathsf{CNTR}}(L) \leq 2^{n-2}$. *Then, we have*

1. $\mathsf{CtH}_{\Lambda,\mathsf{CNTR}}$ *is* 2^{-n}*-AXU hash if* $\Lambda \leftarrow\!\!\$\ \mathsf{Func}(\mathbb{B},\mathbb{B})$*; and*
2. $\mathsf{CtH}_{\Lambda,\mathsf{CNTR}}$ *is* 2^{1-n}*-AXU hash if* $\Lambda \leftarrow\!\!\$\ \mathsf{Perm}(\mathbb{B})$.

Proof. We prove the second part, while the first can be shown analogously. Let $M \neq M'$ be two messages of lengths ℓ and ℓ', $\mathsf{CTR}(M) = S = \{X_1, \ldots, X_b\}$, and $\mathsf{CTR}(M') = S' = \{X_1, \ldots, X_{b'}\}$. As CNTR is prefix-free and injective, there exists at least one block Y which appears exactly once in $S \cup S'$. Thus, for any $\delta \in \mathbb{B}$,

$$\begin{aligned}\Pr\left[\mathsf{CtH}_{\Lambda,\mathsf{CNTR}}(M) \oplus \mathsf{CtH}_{\Lambda,\mathsf{CNTR}}(M') = \delta\right] &= \Pr\left[\Lambda(Y) = \oplus_{y \in S_1 \cup S_2 \setminus \{Y\}} \Lambda(y)\right] \\ &\overset{1}{\leq} \frac{1}{2^n - b^{\mathsf{CNTR}}(\ell) - b^{\mathsf{CNTR}}(\ell')} \overset{2}{\leq} \frac{2}{2^n}.\end{aligned}$$

where inequality 1 follows by conditioning on the outputs of Λ on all the blocks except Y, and inequality 2 follows from $b^{\mathsf{CNTR}}(L) \leq 2^{n-2}$. □

Remark 5.3.5. The above result has been proved implicitly in [16, 23, 126] for the standard counter. Our result generalizes their results for any prefix-free and injective counter function family.

5.3.2 Message Authentication Codes

Now that we have an AXU hash function, we can apply standard methods discussed in section 2.4.2 to obtain (non-)deterministic MAC schemes from CtH.

5.3.2.1 Deterministic MACs

Let CNTR be a CFF. Given any $M \in \{0,1\}^\ell$ with $\ell > n$, we write $M = M' \| m$ where m is a block. We define

$$\mathsf{CtMAC1}_{\mathsf{E}_{\mathsf{K}_1},\mathsf{E}_{\mathsf{K}_2}}(M) := \mathsf{E}_{\mathsf{K}_2}(\mathsf{CtH}_{\mathsf{E}_{\mathsf{K}_1}}(M') \oplus m).$$

In the following theorem we show that CtMAC1 is a secure PRF.

Theorem 5.3.6. *Let* $\mathsf{CtMAC1}_{\mathsf{E}_{\mathsf{K}_1},\mathsf{E}_{\mathsf{K}_2},\mathsf{CNTR}}$ *be defined based on two independent instantiations of a keyed function* E*, and a prefix-free and injective CFF* CNTR. *We have*

1. *If* E *is a PRP, then we have*

$$\mathbf{Adv}^{\mathsf{prf}}_{\mathsf{CtMAC1}_{\mathsf{E},\mathsf{E},\mathsf{CNTR}}}(q,\hat{\ell},t) \leq 2\mathbf{Adv}^{\mathsf{prp}}_{\mathsf{E}}(q\hat{\ell},t) + \frac{1.5q^2}{2^n}.$$

2. *If* E *is a PRF, then we have*

$$\mathbf{Adv}^{\mathsf{prf}}_{\mathsf{CtMAC1}_{\mathsf{E},\mathsf{E},\mathsf{CNTR}}}(q,\hat{\ell},t) \leq 2\mathbf{Adv}^{\mathsf{prf}}_{\mathsf{E}}(q\hat{\ell},t) + \frac{q^2}{2^n}.$$

Proof. We prove the first part, the second part can be shown analogously. First, we replace the two instantiations of E with independent random permutations Π and Π', at the cost of $2\mathbf{Adv}^{\mathsf{prp}}_{\mathsf{E}}(q\hat{\ell},t)$. Now, Theorem 5.3.4 shows that CtH_{Π} is a $2/2^n$-AXU hash function when CNTR is prefix-free and injective. So, the modified scheme $\mathsf{CtH}_{\Pi}(M')\oplus m$ is a $2/2^n$-AU hash when CNTR is prefix-free and injective (using Proposition 2.3.5). The result follows by direct application of Proposition 2.4.3. □

Remark 5.3.7. The deterministic MAC schemes from XMAC family, namely PCS [23] and LightMAC [126], are instantiations of CtMAC1 with STD as the underlying CFF. Since, STD is prefix-free and injective, the security of PCS and LightMAC follows directly from Theorem 5.3.6.

5.3.2.2 Non-deterministic MACs

Let N be a seed (either random or nonce). We define

$$\mathsf{CtMAC2}_{\mathsf{E}_{\mathsf{K}_1},\mathsf{E}_{\mathsf{K}_2}}(N,M) = \mathsf{E}_{\mathsf{K}_2}(N) \oplus \mathsf{CtH}_{\mathsf{E}_{\mathsf{K}_1}}(M).$$

Depending upon the nature of N, we get either a nonce-based MAC, denoted $\mathsf{CtMAC2}^{\mathsf{st}}$, or a probabilistic MAC $\mathsf{CtMAC2}^{\$}$. Using Theorem 5.3.4 and Propositions 2.4.4 and 2.4.5 we get the following result on the security of $\mathsf{CtMAC2}^{\mathsf{st}}$ and $\mathsf{CtMAC2}^{\$}$.

Theorem 5.3.8. *Let* $\mathsf{CtMAC2}_{\mathsf{E}_{\mathsf{K}_1},\mathsf{E}_{\mathsf{K}_2},\mathsf{CNTR}}$ *be defined based on two independent instantiations of a keyed function* E*, and a prefix-free and injective CFF* CNTR*. Then,*

1. *If* E *is a PRP, then we have*
   - $\mathbf{Adv}^{\mathsf{mac}}_{\mathsf{CtMAC2}^{\mathsf{st}}{}_{\mathsf{E},\mathsf{E},\mathsf{CNTR}}}(q,\hat{\ell},t) \leq 2\mathbf{Adv}^{\mathsf{prp}}_{\mathsf{E}}(q\hat{\ell},t) + \frac{2q_v}{2^n}\left(1-\frac{q_m}{2^n}\right)^{-\frac{q_m+1}{2}}$*, and*
   - $\mathbf{Adv}^{\mathsf{mac}}_{\mathsf{CtMAC2}^{\$}{}_{\mathsf{E},\mathsf{E},\mathsf{CNTR}}}(q,\hat{\ell},t) \leq 2\mathbf{Adv}^{\mathsf{prp}}_{\mathsf{E}}(q\hat{\ell},t) + \frac{2q_v}{2^n}\left(1-\frac{q_m}{2^n}\right)^{-\frac{q_m+1}{2}} + \frac{0.5q_m^2}{2^n}$.
2. *If* E *is a PRF, then we have*
   - $\mathbf{Adv}^{\mathsf{mac}}_{\mathsf{CtMAC2}^{\mathsf{st}}{}_{\mathsf{E},\mathsf{E},\mathsf{CNTR}}}(q,\hat{\ell},t) \leq 2\mathbf{Adv}^{\mathsf{prf}}_{\mathsf{E}}(q\hat{\ell},t) + \frac{2q_v}{2^n}$*, and*

- $\mathbf{Adv}^{\mathsf{mac}}_{\mathsf{CtMAC2}^{\$}_{\mathsf{E},\mathsf{E},\mathsf{CNTR}}}(q,\hat{\ell},t) \leq 2\mathbf{Adv}^{\mathsf{prf}}_{\mathsf{E}}(q\hat{\ell},t) + \frac{0.5q_m^2}{2^n} + \frac{2q_v}{2^n}$.

Remark 5.3.9. The non-deterministic MAC schemes from XMAC family, namely XMACC and XMACR [16], are PRF-based instantiations of $\mathsf{CtMAC2}^{\mathsf{st}}$ and $\mathsf{CtMAC2}^{\$}$, respectively, with STD as the underlying CFF. Since, STD is prefix-free and injective, the security of XMACC and XMACR follows directly from Theorem 5.3.8.

5.3.2.3 Necessity of Prefix-free and Injectivity

In all the security results discussed till now, the security proof is implied by the prefix-free and injective property of the CFF CNTR. In other words, prefix-free and injective properties are sufficient for security. Now, we show that these properties are also necessary for CtMAC1 and CtMAC2, and hence for the XMAC family as well.

Case 1: Suppose CNTR doesn't have injective property. As CNTR is a deterministic function (encodings are deterministic), one can easily find two distinct messages M and M' such that $\mathsf{CNTR}(M) = \mathsf{CNTR}(M')$.

Case 2: Suppose CNTR is not prefix-free, i.e., for some $\ell \leq L$, we have $i < j \in \mathbb{N}$ such that $\mathsf{cntr}_\ell(i)$ is a prefix of $\mathsf{cntr}_\ell(j)$. Thus, we can find two pairs (c, c') and (d, d') such that $c \neq d$, $c' \neq d'$, $\mathsf{cntr}_\ell(i)\|c = \mathsf{cntr}_\ell(j)\|c'$ and $\mathsf{cntr}_\ell(i)\|d = \mathsf{cntr}_\ell(j)\|d'$. Using these two pairs we can construct two distinct messages M and M' which only differ at the i-th and j-th blocks. Further, suppose the i-th block of M and M' is c and d, respectively, and the j-th block of M and M' is c' and d', respectively. Clearly, the i-th encoded block of M collides with the j-th encoded block of M' and vice-versa, i.e., they cancel out each other in $\mathsf{CtH}(M)$ and $\mathsf{CtH}(M')$.

In both the cases the adversary can construct $M \neq M'$, such that

$$\Pr\left[\mathsf{CtH}(M) \oplus \mathsf{CtH}(M') = 0\right] = 1.$$

This leads to trivial forgery attacks on CtMAC1 and CtMAC2, as the adversary can make one MAC query with message M, and forge with M' and the tag T so obtained for M. Thus, prefix-free and injective properties are both necessary and sufficient for CtMAC1 and CtMAC2.

5.3.3 Counter-based Cryptographic Hash

As a side-result, we also propose a cryptographic hash function based on CFFs and block cipher. A hash function H is a function from $\cup_{i=1}^{L}\{0,1\}^i$ to $\mathbb{B}$, which aspires to achieve three chief security requirements:

1. *Preimage resistance*: Given a random challenge y, it is hard to find M such that $H(M) = y$. Here M is called *preimage* of y.
2. *Collision resistance*: It is hard to find a pair (M, M') of distinct elements, called *collision pair* for H, such that $H(M) = H(M')$.
3. *Second preimage (2PI) resistance*: Given a random challenge input M, it is hard to find a collision pair of the form (M, M'). In this case M' is also called a *second preimage* of M.

We employ a variant of *HAsh Iterative FrAmework* (HAIFA) [31] by Biham and Dunkelman to construct the hash function, which resists several second preimage attacks [6, 59, 116] applicable to a popular class of hash functions, the so-called Merkle-Damgård hash [55, 134]. The main idea of HAIFA is to process counter-encoded inputs in iteration. This roughly means that all primitive inputs are distinct (as counter differ) in a hash computation, which allows for higher security. In fact, Bouillaguet and Fouque [41] have shown that HAIFA has full (secure up to $o(2^n)$ queries) second preimage security in the random oracle model (the underlying primitive is assumed to be a public random function).

5.3.3.1 CtHAIFA Hash Function

Let CNTR be a CFF. We present a counter based HAIFA hash function, called CtHAIFA, based on a block cipher E-n/n (both block and key lengths are n bits). For a message $M \in \{0,1\}^{\eta^*}$, let $h_0 = \mathsf{IV}$, an n-bit constant, and

$$\forall\, i \in [b], \quad h_i := \mathsf{DM}_{\mathsf{E}}(h_{i-1}, X_i),$$

where X_i is the i-th element of $\mathsf{CNTR}(M)$, and $\mathsf{DM}_{\mathsf{E}} : \mathbb{B} \times \mathbb{B} \to \mathbb{B}$ is the Davies-Meyer compression function [166] which is defined by the mapping $(x, y) \mapsto \mathsf{E}_y(x) \oplus x$. We define CtHAIFA (illustrated in Figure 5.3.1) based on CNTR as,

$$\mathsf{CtHAIFA}_{\mathsf{E},\mathsf{CNTR}}(M) := h_\star = \mathsf{DM}_{\mathsf{E}}(h_b, \langle |M| \rangle_n).$$

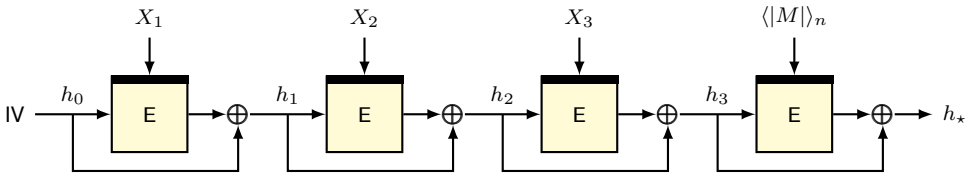

Figure 5.3.1: CtHAIFA hash computation over a 3-block message.

Since, we employ a block cipher instead of a public random function, the random oracle model as used in [41] is not appropriate to prove the security of CtHAIFA. The ideal cipher model (ICM) [50, 51] is widely used as the de facto model for a block cipher based hash function. In this model, the block cipher E is modeled as an *ideal cipher*, where for each $K \in \mathbb{B}$, $\mathsf{E}_K \leftarrow\!\!\$\, \mathsf{Perm}(\mathbb{B})$, and E_K is statistically independent of $\mathsf{E}_{K'}$ for all $K' \neq K \in \mathbb{B}$. Further, the adversary has access to E and it's inverse E^{-1}.

The preimage and collision security up to $o(2^n)$ and $o(2^{n/2})$ queries, respectively, can be easily shown using simple reductions to the preimage and collision resistance of DM in the ICM. We will concentrate on the second preimage security of CtHAIFA. Formally, the advantage of any adversary $\mathscr{A}$ in finding a second preimage of a hash function H_{E} under ICM is defined as

$$\mathbf{Adv}^{\mathsf{2PI}}_{H_{\mathsf{E}}}(\mathscr{A}) := \Pr_{\mathsf{E}}\left[\mathscr{A}^{\mathsf{E}^{\pm}}(M) = M' \;:\; M \leftarrow\!\!\$\, \cup_{i=1}^{L} \{0,1\}^i \wedge H_{\mathsf{E}}(M') = H_{\mathsf{E}}(M)\right].$$

Theorem 5.3.10 gives an upper bound on the second preimage advantage.

Theorem 5.3.10. *Let* CNTR *be a prefix-free and injective CFF. Then,* CtHAIFA *has full second preimage security. More specifically, for any second preimage adversary $\mathscr{A}$ that makes $q \leq 2^n/3$ many queries, we have*

$$\mathbf{Adv}^{\mathsf{2PI}}_{\mathsf{CtHAIFA}_{\mathsf{E},\mathsf{CNTR}}}(\mathscr{A}) \leq \frac{3q}{2^n}.$$

Proof. We use a similar approach as used in [41] for HAIFA based on random oracle. In the ideal cipher model, $\mathscr{A}$ can make both encryption and decryption queries to the underlying block cipher. We also assume that the adversary computes $\mathsf{CtHAIFA}(M)$. The adversary can do this by making encryption query (M_i, h_{i-1}) and storing $(h_{i-1}, M_i, h_i := \mathsf{E}_{M_i}(h_{i-1}) \oplus h_{i-1})$ for all $i \in [b]$. Actually $\mathscr{A}$ can compute any $\mathsf{DM}_{\mathsf{E}}(h, m)$ by making encryption query (m, h) to E. Let the sequence of intermediate chaining values for the challenge message M be $(h_0, h_1, \ldots, h_b, h_{b+1})$.

We denote the i-th query of $\mathscr{A}$ by the tuple $(x_i, m_i, c_i, y_i, o_i)$ such that c_i is the counter value, m_i is the message value, x_i is some chaining value, and $y_i = \mathsf{DM}(x_i, c_i \| m_i) \oplus x_i$. Further,

1. if the i-th query is an encryption query then $o_i = 0$ and $\mathscr{A}$ queried $c_i \| m_i, x_i$ and got $y_i := \mathsf{E}_{c_i \| m_i}(x_i)$.
2. if the i-th query is a decryption query then $o_i = 1$ and $\mathscr{A}$ queried $c_i \| m_i, y_i$ and got $x_i := \mathsf{E}^{-1}_{c_i \| m_i}(y_i)$.

Now suppose that $\mathscr{A}$ produces a valid second preimage M'. We denote this event by $\texttt{Win}$. Let $\texttt{Win}^{\neq}$ and $\texttt{Win}^{=}$ denote the events $\texttt{Win} \wedge |M| \neq |M'|$ and $\texttt{Win} \wedge |M| = |M'|$, respectively. Clearly $\texttt{Win} = \texttt{Win}^{\neq} \sqcup \texttt{Win}^{=}$. Therefore,

$$\Pr[\texttt{Win}] = \Pr\left[\texttt{Win}^{\neq}\right] + \Pr[\texttt{Win}^{=}].$$

We bound the probabilities of the two events on the R.H.S. as follows:

1. If $\texttt{Win}^{\neq}$ is true then $|M| \neq |M'|$. In this case we must have $\langle |M| \rangle_n \neq \langle |M'| \rangle_n$. Therefore, the adversary found a second preimage for the last block, i.e., there exist a query $i \in [q]$ such that $c_i \| m_i = \langle |M'| \rangle_n \neq \langle |M| \rangle_n$ and $y_i \oplus x_i = h_{b+1}$. This is possible with probability $\frac{1}{2^n - i + 1}$. Bounding over q queries we have $\Pr\left[\texttt{Win}^{\neq}\right] \leq \frac{q}{2^n - q}$.
2. If $\texttt{Win}^{=}$ is true then $|M| = |M'|$. In this case, if $\mathscr{A}$ succeeds, then $\mathscr{A}$ was successful in creating a connection to some intermediate chaining value h_j. Suppose this connection is established at the i-th query, then c_i must be equal to $\mathsf{ctr}_\ell(j)$ and $y_i \oplus x_i = h_j$. This probability is again $\frac{1}{2^n - i + 1}$. Bounding over q queries we have $\Pr[\texttt{Win}^{=}] \leq \frac{q}{2^n - q}$.

The result follows by combining 1 and 2. □

Remark 5.3.11. It is well-known that Davies-Meyer compression functions have efficiently computable fixed points, $\mathsf{DM}_{\mathsf{E}}(\mathsf{E}_y^{-1}(0), y) = \mathsf{E}_y^{-1}(0)$ for all $y \in \mathbb{B}$. However, this does not help in inverting any arbitrary hash value. This can be argued as follows: Let h_i be some hash value and the adversary aims to compute a pair (h_{i-1}, m) such that $\mathsf{DM}_{\mathsf{E}}(h_{i-1}, m) = h_i$. Suppose the adversary queries (m, y) to the decryption oracle, then the probability that $\mathsf{E}_m^{-1}(y) = x$ such that $\mathsf{E}_m(x) \oplus x = h_i$ is at most $1/(2^n - i + 1)$.

Remark 5.3.12. Dean [59] showed an attack on Merkle-Damgård hash functions, when the underlying compression function has efficiently computable fixed points for random message blocks. That attack cannot be extended to CtHAIFA as the underlying CFF has block-wise collision free property.

Remark 5.3.13. In Crypto 2005, Coron et al. [50] proved that Merkle-Damgård hash with prefix-free encoding is indifferentiable from a random oracle. Here prefix-free refers to

the property that the encoding of M is not a prefix of the encoding of M', if $M \neq M'$. Recall that a prefix-free CFF means $\mathsf{cntr}_\ell(i)$ is not a prefix of $\mathsf{cntr}_\ell(j)$ if $i \neq j$. Not all encoding schemes based on prefix-free CFFs are prefix-free encodings. In fact length padding is necessary to get prefix-free encodings.

5.4 Comparison and Empirical Result

In this section, we explore the feasibility of replacing the classical counter STD with the newly introduced OPT and VAR. We compare the performance of the three candidate counter function families via their application in different CaE schemes. In particular, we present software implementation of CtMAC1, CtMAC2$^{\mathsf{st}}$, and CtHAIFA. We instantiate the underlying primitive with AES-128 [152] block cipher.

5.4.1 Platform Setup

As mentioned earlier we use AES-128 (key size 128 bits) as the underlying block cipher. We use performance data for all inputs of length $\ell = 2^k$ bits where $7 \leq k \leq 20$. We implemented all the schemes on Intel's Sandy Bridge microarchitecture using AES-NI and SSE4 instructions. All measurements were taken on a single core of Intel Xeon E5-2640 processor at 2.5Ghz with Turbo Boost disabled. The warmup parameter is 250000 and the data is averaged over 1000000 repetitions. All results will be either in number of cycles per byte (or CPB) or number of cycles. The baseline performance for AES-128 using AES-NI instruction set is presented in Table 5.4.1.

Table 5.4.1: Baseline CPB value of AES-128 using AES-NI in Sandy Bridge architecture.

	encryption (cycles/byte)	**key scheduling** (cycles)
(AES, serial)	6.7	117
(AES, parallel)	0.69	117

5.4.2 Implementing Counter Function Families

We choose $w = 8$ as the word size in our implementations. The three counter function families basically differ in their selection of counter sizes. STD^s is a standard counter with a fixed size s; OPT computes the (optimal) counter size s based on the length of the input; and VAR^r dynamically changes the counter size s based on the number of input blocks processed so far. We choose $r = 4$ in our implementations.

In practice, STD^s requires a beforehand heuristic on the typical input lengths to be encountered. In situations where the input lengths are inconstant, this scheme may not be that efficient. If the input length is known beforehand, then OPT is the best option as it fixes the optimal (machine-dependent) counter size. VAR^r has an edge over the other two in the worst case scenario, i.e., when neither the input length is known nor the inputs have predictable lengths. This scenario is quite frequent in cloud-server applications.

Although OPT and VAR^r are much more flexible in terms of the changes in the input length, they do require some book-keeping operations. OPT requires the computation of a suitable counter size for the given input length. Similarly, VAR^r requires some kind of mechanism to update the counter size when the current counter reaches its limit.

STD^s does not incur such book-keeping overheads, as it is fixed for the construction. Thus, STD^s with $s = 32$ uses 32-bit counter irrespective of whether the input size is 2^{10} or 2^{20} bits. We implement STD^s for $s = w2^{i-1}$ and $i \in [4]$. Similar values are used in OPT and VAR^r while choosing (or expanding) the counter size. For OPT, we store the set of counter sizes ($\{8, 16, 32, 64\}$) along with their respective maximum input lengths. Although this incurs a small increment in code size, it reduces the number of micro-operations. For VAR^r, the overhead can be reduced significantly by reducing the number of times the counter size is increased. For example, we increase the size in steps of multiple of $8 \cdot 2^{i-1}$ (i.e. $8, 16, 32, 64$) (instead of $8 \cdot i$). All three counters were implemented in 64-bit registers.

Table 5.4.2 gives the book-keeping cost incurred by the three counters. Figure 5.4.1 presents these characteristics graphically. The cost for OPT is significantly lower than the other two candidates as in this case the major book-keeping operation is executed only once per message. For VAR^r and STD^s the check for last message block accounts for most of the book-keeping cost. VAR^r has a slightly more involved counter update mechanism as compared to STD^s. This leads to a further increase in its book-keeping cost. However, we remark that we have not tried to optimize the counter updation in VAR^r beyond the obvious. So, it is quite possible that the book-keeping cost can be further reduced.

5.4.3 Performance of MACs

We summarize the performance results of the deterministic MACs in Table 5.4.3 and the stateful MACs in Table 5.4.4. Figure 5.4.2 and 5.4.3 give graphical characteristics of the performance data.

Table 5.4.2: Cycles per byte comparison between the book-keeping operations performed by the three CFF candidates.

Length	STD^s				OPT^8	$\mathsf{VAR}^{4,8}$
	s = 8 bits	s = 16 bits	s = 32 bits	s = 64 bits		
128B	0.17	0.20	0.22	0.27	0.33	0.18
256B	0.15	0.17	0.18	0.27	0.18	0.17
512B	0.14	0.15	0.16	0.23	0.13	0.15
1kB	0.13	0.14	0.16	0.22	0.09	0.15
2kB	0.14	0.13	0.15	0.22	0.09	0.15
4kB	-	0.14	0.15	0.22	0.10	0.16
8kB	-	0.14	0.16	0.22	0.09	0.16
16kB	-	0.13	0.15	0.22	0.09	0.15
32kB	-	0.14	0.15	0.22	0.09	0.15
64kB	-	0.14	0.16	0.22	0.09	0.16
128kB	-	0.14	0.16	0.22	0.09	0.17
256kB	-	0.16	0.18	0.23	0.09	0.18
512kB	-	0.17	0.19	0.24	0.09	0.19
1MB	-	-	0.19	0.24	0.08	0.19

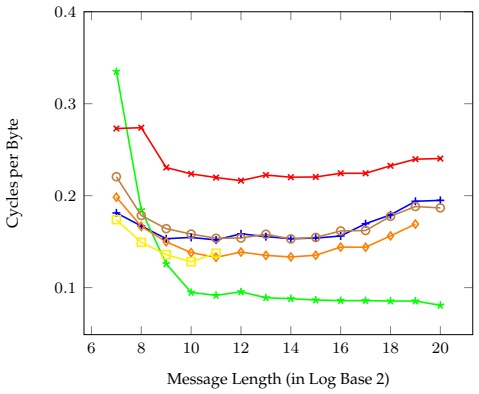

Figure 5.4.1: CPB plot for book-keeping operations of the three CFFs. Plots marked with $+$, $\star$, $\times$, $\circ$, $\diamond$, and $\square$ correspond to $\mathsf{VAR}^{4,8}$, OPT^8, STD^{64}, STD^{32}, STD^{16}, and STD^8, respectively.

Some notes on the characteristics: It is evident from Figure 5.4.2 and 5.4.3 that OPT gives the fastest MAC among all the three candidates. Also observe that when the input length is comparable to the counter size both STD^s and VAR^r closely resemble the performance curve for OPT. For instance, in Table 5.4.3, look at the entries corresponding to the range 8 kB (2^{16}) to 64 kB (2^{19} bits). In this range all three counters offer similar cpb of around 0.8-0.9.

For a fixed value of s in STD^s, VAR^r outperforms STD^s, until the input length is significantly smaller than 2^s, or the counter size in VAR^r is smaller than s. For example, for $s = 32$ VAR^r is faster than STD^s, when the input length $\ell \leq 128$kB (2^{20} bits). At 256 kB, the counter size in VAR^r expands to 32 bits which causes a change in the slope of the curve. Even when the input length lies in the optimal range the gain in using STD^s is marginal when compared to the flexibility offered by VAR^r over a diverse range of input lengths.

Table 5.4.3: Comparison between the CPB values of CtMAC1 based on the three CFF candidates.

Length	STD^s				OPT^8	$\mathsf{VAR}^{4,8}$
	s = 8 bits	s = 16 bits	s = 32 bits	s = 64 bits		
128B	1.26	1.74	1.79	1.96	1.37	1.24
256B	1.25	1.26	1.29	1.74	1.27	1.26
512B	1.00	1.02	1.18	1.63	1.00	1.03
1kB	0.88	0.98	1.07	1.57	0.88	0.91
2kB	0.83	0.92	1.05	1.54	0.79	0.91
4kB	-	0.90	1.02	1.52	0.86	0.89
8kB	-	0.88	1.01	1.51	0.83	0.87
16kB	-	0.88	1.00	1.51	0.83	0.88
32kB	-	0.88	1.01	1.51	0.82	0.87
64kB	-	0.88	1.00	1.51	0.81	0.89
128kB	-	0.87	1.00	1.51	0.82	0.95
256 kB	-	0.87	1.01	1.51	0.82	0.98
512kB	-	0.88	1.01	1.51	0.82	1.00
1MB	-	-	1.01	1.51	0.95	1.01

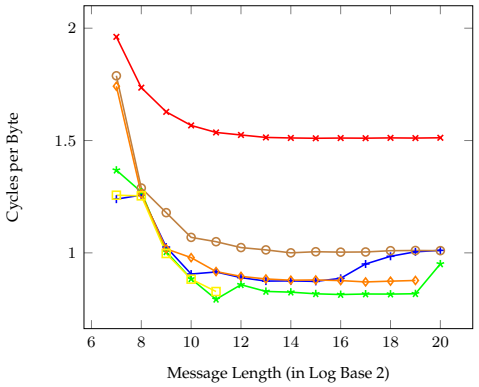

Figure 5.4.2: CPB plots for CtMAC1 based on the three CFF candidates. Plots marked with $+$, $\star$, $\times$, $\circ$, $\diamond$, and $\square$ correspond to CtMAC1 with $\mathsf{VAR}^{4,8}$, OPT^8, STD^{64}, STD^{32}, STD^{16}, and STD^8, respectively.

Note that, the effective counter size in VAR^r is always r bits less than the actual counter size. This is because the first r bits are reserved. For example when the counter size is 16 bits, only 12 bits are used. This reduces the maximum message length for 16-bit counter from 2^{23} bits to 2^{19} bits.

5.4.4 Performance of HAIFA

We summarize the performance results of CtHAIFA in Table 5.4.5. The graphical representation is illustrated in Figure 5.4.4. The comparison results are similar to those obtained earlier, in case of CtMAC1 and $\mathsf{CtMAC2}^{\mathsf{st}}$.

Table 5.4.4: Comparison between the CPB values of CtMAC2st based on the three CFF candidates.

Length	STD^s				OPT^8	VAR^{4,8}
	s = 8 bits	s = 16 bits	s = 32 bits	s = 64 bits		
128B	1.21	1.28	1.47	2.05	1.31	1.23
256B	1.03	1.11	1.30	1.74	1.07	1.04
512B	0.94	1.04	1.11	1.62	0.94	1.00
1kB	0.91	0.92	1.08	1.58	0.87	0.94
2kB	0.84	0.90	1.03	1.54	0.79	0.89
4kB	-	0.89	1.02	1.53	0.85	0.88
8kB	-	0.88	1.01	1.52	0.82	0.87
16kB	-	0.88	1.01	1.51	0.82	0.87
32kB	-	0.87	1.01	1.51	0.82	0.87
64kB	-	0.87	1.00	1.51	0.81	0.89
128kB	-	0.87	1.00	1.52	0.81	0.95
256kB	-	0.87	1.00	1.52	0.81	0.99
512kB	-	0.88	1.01	1.52	0.82	1.01
1MB	-	-	1.01	1.52	0.95	1.02

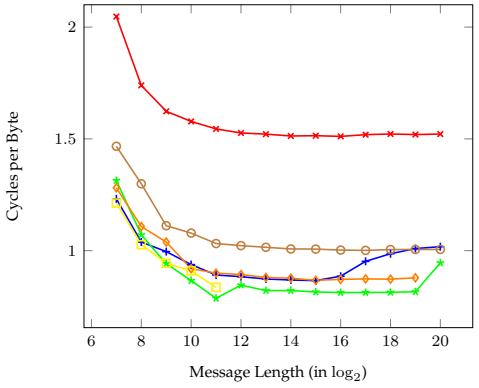

Figure 5.4.3: CPB plots for CtMAC2st based on the three CFF candidates. Plots marked with $+$, $\star$, $\times$, $\circ$, $\diamond$, and $\square$ correspond to CtMAC2st with VAR^{4,8}, OPT^8, STD^{64}, STD^{32}, STD^{16}, and STD^8, respectively.

Table 5.4.5: Comparison between the CPB values of CtHAIFA based on the three CFF candidates.

Length	STD^s				OPT^8	VAR^{4,8}
	s = 8 bits	s = 16 bits	s = 32 bits	s = 64 bits		
128B	7.42	8.16	8.96	13.83	7.64	7.58
256B	7.37	7.76	8.95	13.33	7.31	7.47
512B	7.17	7.55	8.69	13.15	7.00	7.53
1kB	7.02	7.55	8.71	13.07	6.92	7.65
2kB	6.95	7.48	8.63	13.03	6.88	7.62
4kB	-	7.43	8.65	13.00	7.43	7.59
8kB	-	7.40	8.61	12.92	7.42	7.57
16kB	-	7.42	8.61	12.95	7.42	7.60
32kB	-	7.43	8.63	12.99	7.41	7.62
64kB	-	7.41	8.64	12.97	7.41	7.78
128kB	-	7.41	8.65	12.98	7.41	8.37
256kB	-	7.42	8.66	12.99	7.42	8.66
512kB	-	7.44	8.67	12.99	7.43	8.82
1MB	-	-	8.66	12.99	8.65	8.90

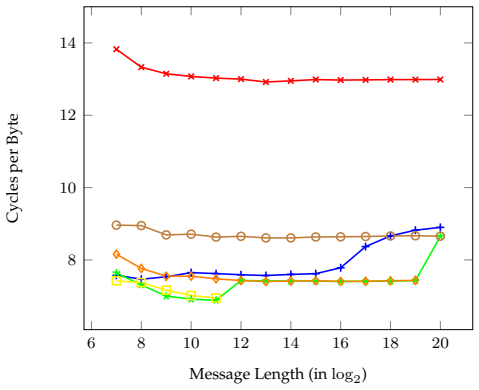

Figure 5.4.4: CPB plots for CtHAIFA based on the three CFF candidates. Plots marked with $+$, $\star$, $\times$, $\circ$, $\diamond$, and $\square$ marked plots correspond to CtHAIFA with $\mathsf{VAR}^{4,8}$, OPT^{8}, STD^{64}, STD^{32}, STD^{16}, and STD^{8}, respectively.

Part II

Analysis of Online and Tweakable Ciphers

Chapter 6

Tweakable Pseudorandom Permutation based Online Ciphers

Till now, we have seen schemes concerned with the authenticity and integrity of data, i.e., message authentication codes. In this chapter, we shift our focus to confidentiality. We study a special class of encryption schemes, called online ciphers [18, 39]. In particular, we study highly secure[1] and efficient[2] online cipher schemes based on tweakable block ciphers.

Andreeva et al. [7] showed that online ciphers are equivalent to arbitrary tweak length (ATL) TBCs. Although this result is more of theoretical interests, the TBC to online cipher converter can be coupled with XTX [139] to get an online cipher. In this chapter, we study the (un)suitability of straightforward application of this approach. Further, we study variants of this approach to construct efficient and highly secure online ciphers.

6.1 A Design Strategy for TBC-based BBB Online Cipher

The rate of any cryptographic scheme can be defined as the ratio of the number of blocks in any input to the number of primitive (TBC in this case) calls for that input. In most of the cases, rate is independent of the input. A rate ≥ 1 scheme will, in general, be more efficient then a rate $\frac{1}{2}$ scheme when identical primitives are employed. In this chapter, we use rate and the number of field multiplications as our parameters for efficiency.

[1] Beyond-the-birthday-bound security.

[2] Number of primitive calls is at most $\lceil \ell/n \rceil$ where ℓ denotes the message length and n denotes the primitive block size.

6.1.1 The Iterated TBC View of Online Ciphers

In a recent work [7] on a generic construction of offline ciphers (deterministic encryption scheme) using online ciphers as primitives, Andreeva et al. made the following useful observation:

An ideal online cipher is equivalent to an ideal arbitrary tweak length (ATL) TBC.

We reproduce their result in Theorem 6.1.1.

Theorem 6.1.1 ([7, Theorem 1]). *There is a security preserving one-to-one correspondence between online ciphers on $\mathbb{B}^+$ and tweakable block ciphers on $\mathbb{B}$ with tweak space $\mathbb{B}^*$.*

While a more detailed proof is available in [7], we describe their generic construction of online cipher based on a tweakable block cipher with arbitrary tweak space. Let $\widehat{\mathsf{E}}_K$ be a TBC with tweak space $\mathbb{B}^*$ and block space $\mathbb{B}$. We define an online cipher $\mathfrak{O}_{\widehat{\mathsf{E}}}$ over $\mathbb{B}^+$ as $\forall \ell \geq 1, \forall x := (x_1, \ldots, x_\ell) \in \mathbb{B}^\ell$,

$$\mathfrak{O}^+_{\widehat{\mathsf{E}}}(x) := \widehat{\mathsf{E}}^{x_0}_K(x_1) \| \widehat{\mathsf{E}}^{x^{(1)}}_K(x_2) \| \cdots \| \widehat{\mathsf{E}}^{x^{(i-1)}}_K(x_i) \| \cdots \| \widehat{\mathsf{E}}^{x^{(\ell-1)}}_K(x_\ell),$$

where x_0 is some constant. One can easily verify that,

$$\mathbf{Adv}^{\mathsf{osprp}}_{\mathfrak{O}_{\widehat{\mathsf{E}}}}(q, \sigma, t) \leq \mathbf{Adv}^{\mathsf{tsprp}}_{\widehat{\mathsf{E}}}(\sigma, t'), \tag{6.1}$$

where $t' = t + O(\sigma)$. Specifically, when we replace $\widehat{\mathsf{E}}$ with an ideal ATL tweakable random permutation $\widehat{\Pi}$ we get $\mathbf{Adv}^{\mathsf{osprp}}_{\mathfrak{O}[\widehat{\Pi}]}(q, \sigma, t) = 0$. A minor variant of the above mentioned construction may also consider the previous ciphertext blocks along with the plaintext blocks, i.e., tweak for the i-th block is $(x^{(i-1)}, \mathfrak{O}^+_{\widehat{\mathsf{E}}}(x^{(i-1)}))$, and the modified definition is,

$$\mathfrak{O}^+_{\widehat{\mathsf{E}}}(x) := \widehat{\mathsf{E}}^{x_0}_K(x_1) \| \widehat{\mathsf{E}}^{(x^{(1)}, \mathfrak{O}^+_{\widehat{\mathsf{E}}}(x^{(1)}))}_K(x_2) \| \cdots \| \widehat{\mathsf{E}}^{(x^{(\ell-1)}, \mathfrak{O}^+_{\widehat{\mathsf{E}}}(x^{(\ell-1)}))}_K(x_\ell).$$

It is not hard to see that this does not give any added security advantage. But the utility of this modification will become more apparent as we proceed. Thus any online cipher can be viewed as a chain of iterated TBC. We call this equivalent view of online ciphers, *the iterated TBC view of online cipher*.

6.1.2 $\mathfrak{O}_{\mathsf{XTX}}$: Moving from Theory to Practice

Given the strong security guarantee of Eq. (6.1) and efficiency (rate 1 construction) of $\mathfrak{O}_{\widetilde{\mathsf{E}}}$, it is only natural to look for practical instantiations. The next immediate question is: how can we instantiate an ATL TBC based on a fixed tweak length (FTL) TBC?

XTX [139] by Minematsu and Iwata is an elegant way of extending the tweak length of an FTL TBC. At the highest level, XTX employs a pAXU hash which takes a tweak value of arbitrary length as input and computes a fixed length tweak and input/output masking for the underlying FTL TBC. Formally, let $H : \mathbb{B}^* \to \mathbb{T} \times \mathbb{B}$ be an ϵ-pAXU. For convenience, we have dropped the key of the hash function from the notation, but keep in mind that the key is present. The hash output $H(T)$ is parsed into $H_{\mathrm{twk}}(T) \| H_{\mathrm{msk}}(T)$. Using this pAXU and a TBC $\widetilde{\mathsf{E}}$ over the tweak space $\mathbb{T}$, XTX is defined as $\mathsf{XTX}_K^T(x) = \widetilde{\mathsf{E}}_K^{H_{\mathrm{twk}}(T)}(x \oplus H_{\mathrm{msk}}(T)) \oplus H_{\mathrm{msk}}(T)$. In [139], Minematsu and Iwata have shown the following upper bound on the TSPRP advantage of XTX.

Theorem 6.1.2. $\mathbf{Adv}^{\mathsf{tsprp}}_{XTX}(\sigma, t') \leq \mathbf{Adv}^{\mathsf{tsprp}}_{\widetilde{\mathsf{E}}}(\sigma, t'') + \epsilon\sigma^2$, *where* $t'' = t' + \mathsf{Time}_H \times O(\sigma)$.

By using Eq. (6.1) and Theorem 6.1.2, we obtain a construction $\mathfrak{O}_{\mathsf{XTX}}$ with OSPRP advantage at most $\mathbf{Adv}^{\mathsf{tsprp}}_{\widetilde{\mathsf{E}}}(\sigma, t'') + \sigma^2\epsilon$. A recent proposal by Forler et al., called POEx [71, 72], is an implicit example of $\mathfrak{O}_{\mathsf{XTX}}$. Although $\mathfrak{O}_{\mathsf{XTX}}$ is simple (both in description and security proof), it requires very strong bound on ϵ, close to $2^{-(n+\tau)}$. Now we describe how this becomes an issue when coupled with the need for efficiency. In this chapter we focus on field multiplication based hash functions as the use of TBC in hash computation will make the overall construction inefficient (decreases the rate).

SECURITY VS HASH KEY SIZE: As mentioned before, we only consider algebraic hash functions. Note that, $H : \mathbb{B}^* \to \mathbb{T} \times \mathbb{B}$ is required to be ϵ-pAXU for some ϵ. If we keep hash key size to be $n + \tau$ then $\epsilon' \geq \ell'/2^{n+\tau}$ where $\ell' = \ell n/(n + \tau)$ (the number of $(n + \tau)$-bit strings present in ℓ blocks input). We can achieve this bound by considering polynomial hash defined over $(n + \tau)$-bit binary field. If we plug this ϵ we obtain a bound of the form $\sigma^2\ell/2^{n+\tau}$. To get rid of the ℓ factor we need to apply hash functions with larger hash key size, such as Pseudo dot product [36, 87, 188], multi-linear hash [43, 75, 177], and EHC [150], which is practically infeasible.

6.1.2.1 Towards A Possible Remedy

One possible way of solving or avoiding this degradation/inefficiency could be to use a fixed number of previous blocks information instead of the entire history of previous blocks. This will certainly replace the ℓ factor in case of poly-hash. But now we have

to use both plaintext and ciphertext pairs, otherwise the overall design will no longer be secure (the adversary can always keep the required plaintext blocks fixed). Further a direct security reduction like $\mathfrak{O}_{\mathsf{XTX}}$ will not be possible any more and an independent security analysis is required. In section 6.3 we use this design strategy to give an (almost) affirmative solution for (a) in shape of a new construction that achieves a security in the order of $\min(2^n/n, 2^{(n+\tau)/2})$ block-queries.

POEx, though an implicit instance of $\mathfrak{O}_{\mathsf{XTX}}$, tries to use this strategy. It fixes the input of the hash function to just the immediate previous input-output of the underlying TBC to avoid the ℓ factor while maintaining a key size of $n + \tau$ bits. Unfortunately, there is a flaw in their analysis (discussed in next section). It seems that the actual bound for POEx must have an ℓ factor.

6.2 Revisiting POEx

In this section, we review the security of POEx, a rate-1 online cipher based on tweakable block ciphers, which claims Beyond Birthday Security [71].

6.2.1 Description of POEx

POEx is an extended version of POE, the online cipher used under POET [1]. It constructs an online cipher by iterated usage of a fixed tweak length TBC and a pAXU hash function on two block input/output. The algorithmic description of POEx is given in Algorithm 6.2.1 and a schematic illustration of the encryption/decryption process is shown in Figure 6.2.1. On a macro level the construction looks neat and the security claim looks correct. But in the following subsections we describe a birthday bound attack on POEx that invalidates the security claim.

The security of POEx is claimed to be $2^{\frac{n+\tau}{2}}$ block-queries. More specifically we have the exact security claim in Theorem 6.2.1.

Theorem 6.2.1 (Theorem 3 in [71])**.**

$$\mathbf{Adv}^{\mathsf{osprp}}_{\mathsf{POEx}[\widetilde{\mathsf{E}},H]} \leq 2\mathbf{Adv}^{\mathsf{tsprp}}_{\widetilde{\mathsf{E}}^{\pm}}(\sigma, O(t)) + 2(\sigma+1)^2\epsilon \cdot \left(2 + \frac{2^{\tau}}{2^n - (\sigma+1)}\right)$$

6.2.1.1 Flaw in POEx Security Analysis

The security of POEx mainly rely on the pAXU bound (say ϵ) of the underlying hash function. For example one of the bad events in the security proof of POEx is $(T_{(i,j)}, X_{(i,j)})$

Algorithm 6.2.1 Definition of $\mathsf{POEx}_{\widetilde{\mathsf{E}},H}$. $\mu, \nu \in \mathbb{F}_{2^n}$, are two application specific constants.

1: **function** $\mathsf{POEx}^{+}_{\widetilde{\mathsf{E}}_K,H_L}(P)$	1: **function** $\mathsf{POEx}^{-}_{\widetilde{\mathsf{E}}_K,H_L}(C)$
2: $(X_0, Y_0) \leftarrow (\mu, \nu)$	2: $(X_0, Y_0) \leftarrow (\mu, \nu)$
3: $(P_1, \ldots, P_\ell) \xleftarrow{n} P$	3: $(C_1, \ldots, C_\ell) \xleftarrow{n} C$
4: **for** $i \leftarrow 1$ **to** ℓ **do**	4: **for** $i \leftarrow 1$ **to** ℓ **do**
5: $(T_i, U_i) \leftarrow H_L(X_{i-1}, Y_{i-1})$	5: $(T_i, U_i) \leftarrow H_L(X_{i-1}, Y_{i-1})$
6: $X_i \leftarrow P_i \oplus U_i$	6: $Y_i \leftarrow C_i \oplus U_i$
7: $Y_i \leftarrow \widetilde{\mathsf{E}}_K^{T_i}(X_i)$	7: $X_i \leftarrow \widetilde{\mathsf{E}}_K^{-T_i}(Y_i)$
8: $C_i \leftarrow Y_i \oplus U_i$	8: $P_i \leftarrow X_i \oplus U_i$
9: **end for**	9: **end for**
10: **return** $C := (C_1 \| \cdots \| C_\ell)$	10: **return** $P := (P_1 \| \cdots \| P_\ell)$
11: **end function**	11: **end function**

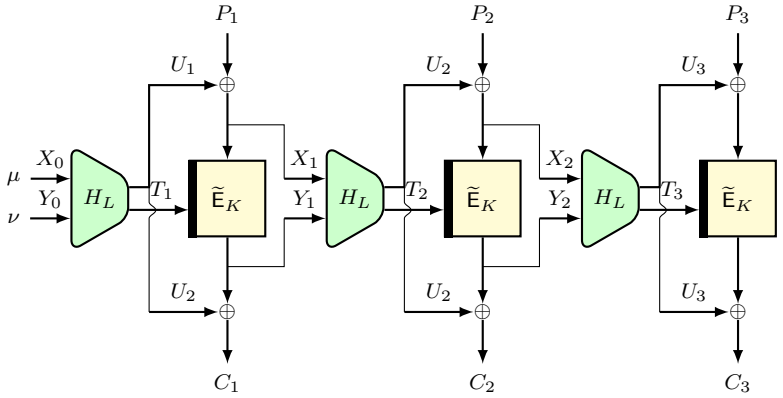

Figure 6.2.1: Schematic of the encryption/decryption process for a 3-block plaintext/-ciphertext using POEx construction.

(tweak-input) collision. This case has been bounded to $\sigma^2\epsilon/2$. The argument being: there can be at most $\binom{\sigma}{2}$ pairs and for each pair the pAXU bound is ϵ. But this argument only holds when all inputs to the hash function are independent of the hash key. In case of POEx this is not true as the input to the hash function is dependent on the previous hash values (linear combination of previous hash value and current message block). For example if we consider the candidate hash function given in [71], the hash-function inputs are dependent and the polynomial degree will accumulate through subsequent inputs to poly-hash. So the security claim is invalid. It seems that POEx requires a stronger assumption, i.e. it needs pAXU assumption on the iterated hash function. This may not be easy to achieve with algebraic hash functions.

6.2.2 Birthday Bound Attack on POEx

We substantiate the flaw discovered in the previous subsection by constructing an artificial example of a pAXU hash function that is not secure when used in iterated fashion.

For the sake of simplicity we take $\tau = n$. Similar attacks can be constructed for any arbitrary τ. The main idea is to construct an AXU hash, say SciFi, that on any given input converges to a fixed value when iterated a moderate number of times. By looking at the masking and tweak generator it can be observed that the adversary can use SciFi to fix the masking to an unknown but constant value. But the same is not possible for the tweak value as the Y_i value is not in the control of the adversary. Once the masking is fixed, the adversary can expect tweak collisions in birthday bound.

6.2.2.1 AXU Hash using Random Mapping over Ranked Nodes

Let $\mathsf{rank} : \mathbb{B} \to [0..n]$ be a surjective function which is defined as follows:

$$\forall x \in \mathbb{B}, \mathsf{rank}(x) := \begin{cases} 0 & x = 0, \\ i & 2^{i-1} \leq x \leq 2^i - 1. \end{cases}$$

Let $\mathbb{B}_i = \{x \in \mathbb{B} \ : \ \mathsf{rank}(x) = i\}$ and $\mathbb{B}_{<i} = \cup_{j<i}\mathbb{B}_j$. Clearly $\mathbb{B} = \sqcup_{i\in[n]}\mathbb{B}_i$. Let

$$\begin{aligned} \Psi &\leftarrow\!\!\$\ \mathsf{Perm}(\mathbb{B}) \\ \Phi_0 &\leftarrow\!\!\$\ \mathsf{Func}(\mathbb{B}_0 \times \mathbb{B}, \mathbb{B}_0) \\ \forall i \in [1..n],\ \Phi_i &\leftarrow\!\!\$\ \mathsf{Func}(\mathbb{B}_i \times \mathbb{B}, \mathbb{B}_{<i}) \end{aligned}$$

We write Φ to denote the random vector $(\Phi_0, \ldots, \Phi_n)$. We define a keyed hash function $\mathsf{SciFi}_{\Psi,\Phi} : \mathbb{B} \times \mathbb{B} \to \mathbb{B}$ as follows:

$$\forall (x, y) \in \mathbb{B} \times \mathbb{B},\ \mathsf{SciFi}_{\Psi,\Phi}(x, y) := \Psi^{-1} \circ \Phi_{\mathsf{rank}(\Psi(x))}(\Psi(x), y).$$

For the sake of simplicity, we drop the subscript Ψ, Φ from $\mathsf{SciFi}_{\Psi,\Phi}$. It is easy to verify that after at most n iterations SciFi returns a fixed value, i.e. for any x and $y^n := (y_1, \ldots, y_n) \in \mathbb{B}^n$ we have

$$\mathsf{SciFi}^n(x, y^n) := \mathsf{SciFi}(\cdots(\mathsf{SciFi}(\mathsf{SciFi}(x, y_1), y_2)\cdots, y_n) = c,$$

where c is some unknown but constant value. This can be argued as follows: after each iteration the rank of the output reduces by at least 1. Since there are finite number of ranks, $n + 1$ to be precise, we must be at rank 0 in n iterations. Once we reach rank 0 the output becomes fixed to $\Psi^{-1}(0)$. We summarize this property in Lemma 6.2.2.

Lemma 6.2.2. *For all $m \geq n$, and $(x, y^m) \in \mathbb{B} \times \mathbb{B}^m$, we have $\mathsf{SciFi}^m(x, y^m) = c$, where $c = \Psi^{-1}(0)$.*

In Lemma 6.2.3 below, we show that SciFi is an $O\left(\frac{n}{2^n}\right)$-AXU hash function. Thus, security wise it can be a good candidate for AXU hash.

Lemma 6.2.3. *For distinct* $(x_1, y_1) \neq (x_2, y_2) \in \mathbb{B} \times \mathbb{B}$ *and* $\delta \in \mathbb{B}$*, we have*

$$\Pr\left[\mathsf{SciFi}(x_1, y_1) \oplus \mathsf{SciFi}(x_2, y_2) = \delta\right] \leq \frac{n+2}{2^n}.$$

In other words, SciFi *is an* $(n+2)2^{-n}$*-AXU hash function.*

Proof. Let R_1 and R_2 denote $\mathsf{rank}(\Psi(x_1))$ and $\mathsf{rank}(\Psi(x_2))$, respectively. Let

$$\mathcal{NZ} := [n] \times [n] \text{ and } \mathcal{Z} := (n] \times (n] \setminus \mathcal{NZ}.$$

Now we have,

$$\begin{aligned}
&\Pr\left[\mathsf{SciFi}(x_1, y_1) \oplus \mathsf{SciFi}(x_2, y_2) = \delta\right] \\
&= \overbrace{\Pr\left[\mathsf{SciFi}(x_1, y_1) \oplus \mathsf{SciFi}(x_2, y_2) = \delta, (R_1, R_2) \in \mathcal{NZ}\right]}^{\epsilon_1} \\
&\quad + \overbrace{\Pr\left[\mathsf{SciFi}(x_1, y_1) \oplus \mathsf{SciFi}(x_2, y_2) = \delta, (R_1, R_2) \in \mathcal{Z}\right]}^{\epsilon_2}
\end{aligned} \tag{6.2}$$

We bound ϵ_1 and ϵ_2 below:

BOUND ON ϵ_1: For a fixed $\psi \in \mathsf{Perm}(\mathbb{B})$ if we consider inputs (x_1, y_1) and (x_2, y_2) such that $\psi(x_1) \neq 0$ and $\psi(x_2) \neq 0$, then the bound effectively reduces to bounding the probability that the sum of outputs of two (possibly) independent random functions on distinct inputs equals to δ. This can be bounded by $2^{-\max\{r_1, r_2\}}$ (by conditioning on the output of the function with smaller range). Using the preceding argument and conditioning on Ψ, we have

$$\begin{aligned}
\epsilon_1 &\leq \sum_{(r_1,r_2)\in\mathcal{NZ}} \Pr\left[\Psi^{-1}(\Phi_{r_1}(\Psi(x_1), y_1)) \oplus \Psi^{-1}(\Phi_{r_2}(\Psi(x_2), y_1)) = \delta\right] \times \frac{2^{r_1+r_2-2}}{2^{2n}} \\
&\leq \sum_{(r_1,r_2)\in\mathcal{NZ}} \frac{1}{2^{\max\{r_1,r_2\}-1}} \times \frac{2^{r_1+r_2-2}}{2^{2n}} \\
&\leq \sum_{(r_1,r_2)\in\mathcal{NZ}} \frac{2^{r_1-1}}{2^{2n}} = \frac{n}{2^n}
\end{aligned} \tag{6.3}$$

BOUND ON ϵ_2: In this case at least one of x_1 or x_2 is mapped to 0 by Ψ. Further $R_1 = R_2 = 0$ is possible if and only if $x_1 = x_2$, in which case we can simply bound the probability to at most 2^{-n}. So without loss of generality we assume that at least one of them is non-zero, say R_1. Now we can proceed as earlier and

we have

$$\begin{aligned}\epsilon_2 &\leq \frac{1}{2^n} + \sum_{r_1 \in [n]} \Pr\left[\Psi^{-1}(\Phi_{r_1}(\Psi(x_1), y_1)) = x_2 \oplus \delta\right] \times \frac{2^{r_1-1}}{2^{2n}} \\ &\leq \frac{1}{2^n} + \sum_{r_1 \in [n]} \frac{1}{2^{r_1-1}} \times \frac{2^{r_1-1}}{2^{2n}} \\ &\leq \frac{2}{2^n} \end{aligned} \tag{6.4}$$

The result follows from Eq. (6.2)-(6.4). □

6.2.2.2 Attack Description

Lemma 6.2.3 shows that SciFi is a good AXU hash function, whereas Lemma 6.2.2 shows that the iterated version $\mathsf{SciFi}^{\geq n}$ is a pathetic AXU hash. We use this later fact to construct a birthday bound attack on POEx. Suppose we instantiate the POEx construction using a tuple of AXU hash functions $H := (\Gamma, \mathsf{SciFi})$ where $\Gamma \leftarrow\!\!\$\ \mathsf{Func}(\mathbb{B}\times\mathbb{B}, \mathbb{B})$ is chosen independently of SciFi. Further the mapping is defined as follows:

$$H(X_i, Y_i) = (\Gamma(X_i, Y_i), \mathsf{SciFi}(X_i, Y_i))$$

Since Γ is a uniform random function over $\mathsf{Func}(\mathbb{B} \times \mathbb{B}, \mathbb{B})$ and hence a 2^{-n}-AXU hash function, using Proposition 2.3.5 we can conclude that H is an $\frac{n+2}{2^{2n}}$-pAXU hash function. Note that, the choice of universal hash is not rigid. We can take any good candidate of universal hash provided it is keyed independently of SciFi. Now consider an adversary $\mathscr{A}$ that works as follows:

1. $\mathscr{A}$ makes q queries of the form $(P_i \| 0^{\ell-1})$, such that $P^q \in (\mathbb{B})_q$ and $\ell > n$, and observes the outputs $(C_{(i,1)} \| \cdots \| C_{(i,\ell)})$.
2. If $(C_{(i,j)}, C_{(i,j+1)}) = (C_{(i',j')}, C_{(i',j'+1)})$ for two distinct block indices (i, j) and (i', j'), then $\mathscr{A}$ returns 1.

For an ideal online cipher this should require roughly 2^n many blocks. But for $\mathsf{POEx}_{\widetilde{\Pi},H}$ this would require roughly $2^{n/2}$ many blocks. This can be argued using Lemma 6.2.2 which, in this case, implies that for each query beyond block index $n-1$, the input becomes a constant value. So all that is required is a tweak collision which can be achieved if we have roughly $2^{n/2}$ blocks. *Hence* POEx *is at most birthday bound secure online cipher*.

Remark 6.2.4. We remark that the hash function notion (pAXU) used in the original POEx design [71] has been modified in a later journal version [72] with due acknowledgments to our observations. The authors have defined a new and stronger notion of hash function, called Chained-Partial-AXU Hash Function (an extension of the pAXU notion for iterated use), and based their security analysis on this notion.However, the authors have not given any practical instantiations for such hash function and we speculate that given the stringent conditions it will be difficult to construct an efficient candidate with close to $2^{-(n+\tau)}$ bound. We refer the readers to [72] for a more detailed exposition. In this chapter, we have considered the original POEx design from [71].

6.3 XTC: Rate-1 (Almost) Optimally Secure Online Cipher

In the preceding section we showed a subtle flaw in the security claim (and proof) of POEx. Here we explore the possibility of another approach towards a practical instantiation of Andreeva et al. [7] TBC to online cipher converter. As discussed in section 6.1, $\mathfrak{O}_{\mathsf{XTX}}$ cannot achieve both rate-1 and BBB security simultaneously. A possible remedy is the idea of using just a fixed number of previous input-output block-pairs information. Based on this idea, we propose XTC, that achieves $\min(2^n/n, 2^{\frac{n+\tau}{2}})$ block-queries security. The XTC construction is similar to POEx in the sense that it follows POEx's idea of generating the mask and tweak using a fixed number of previous blocks information. However, POEx uses pAXU property on hash function with at most ℓ-block input, whereas XTC uses pAXU property on hash function with at most three blocks input.

6.3.1 An Impossibility Result

In POEx [71] as well as XTC (see subsection 6.3.2), only a small number of previous blocks are used for tweak and mask computations. POEx uses the immediate previous TBC input and output blocks, where as XTC employs the previous two plaintext and ciphertext blocks. This is done to avoid the loss of ℓ factor in security. In Theorem 6.3.2, we show an impossibility result that would imply that this approach will be futile when the security goal is more than 2^n block-queries, which is a natural possibility for $\tau > n$.

Definition 6.3.1. A pair of distinct tuples $(a_1, \ldots, a_m)$ and $(b_1, \ldots, b_m)$ is called an m-block *all-but-first collision pair* if $a_i = b_i$ for all $i \in [2 \ldots m]$.

Theorem 6.3.2. *Given $m \geq 3$, for an ideal online cipher an m-block all-but-first collision pair can be constructed in $O(m^2 2^n)$ block-queries.*

Proof. Consider the following algorithm $\mathscr{A}$:

1. Make roughly $2^{n/2}$ distinct encryption queries (P_i, x) and stores the result $(C_{(i,1)}, C_{(i,2)})$ in $\mathcal{L}$. At least one collision on the second ciphertext block is expected, say (P_i, x) and (P_j, x) with the corresponding ciphertext $(C_{(i,1)}, y)$ and $(C_{(j,1)}, y)$.
2. Rename P_i, P_j, $C_{(i,1)}$, and $C_{(j,1)}$ as p_1, p'_1, c_1 and c'_1. Let $p'_2 = p_2 := x$ and $c'_2 = c_2 = y$.
3. For $i \in [3 \dots m]$:
   (a) For all $a \in \mathbb{B}$: Make 2 queries $(p_1, p_2, \dots, p_{i-1}, a)$ and $(p'_1, p'_2, \dots, p'_{i-1}, a)$, and check if $(c_2, \dots, c_i) = (c'_2, \dots, c'_i)$. If the equality holds for some a, then fix $p'_i = p_i := a$ and move to next i.
4. Set $p^m := (p_1, \dots, p_m)$, $c^m := (c_1, \dots, c_m)$, $p'^m := (p'_1, \dots, p'_m)$, and $c'^m = (c'_1, \dots, c'_m)$.
5. Returns (p^m, c^m) and (p'^m, c'^m) as the m-block all-but-first collision pair.

As the first block is distinct for all queries in step 1 above, there is very high chance of getting a collision pair. In step 3(a) at the i-th iteration $(p_1, p_2, \dots, p_{i-1})$ is distinct from $(p'_1, p'_2, \dots, p'_{i-1})$ so the outputs c_i and c'_i are drawn independently from $\mathbb{B}$, whence we expect one collision in 2^n tries. Hence the total block-query complexity is bounded by $O(m^2 \cdot 2^n)$. □

Step 1 of $\mathscr{A}$ in the proof of Theorem 6.3.2 gives a simple corollary that lower bounds the number of previous blocks information required for n-bit security.

Corollary 6.3.3. *To achieve n-bit security at least 2 previous plaintext-ciphertext blocks information is required.*

Remark 6.3.4. Although we constructed an all-but-first collision pair with start index as the first index of the queries, similar strategy can be applied if the collision pair has to be shifted to a later start index.

As long as m is a polynomial in n and n is sufficiently large, Theorem 6.3.2 and Remark 6.3.4 imply that an adversary can always create a collision on m previous input and output blocks in $O(2^n)$ (ignoring the polynomial factors), and thus the tweak value for the current block will collide leading to an easy distinguisher. Thus, to achieve significantly more than 2^n block-queries security one has to use more than m (polynomial in n) many previous blocks.

6.3.2 Description of XTC

For the sake of simplicity we assume tweak size $\tau = n$ while describing the scheme and its security. Later we give ways to extend the result for all τ. Using Corollary 6.3.3 we know that 2 previous plaintext-ciphertext block-pairs can be sufficient for the desired security goal. The algorithmic description of XTC is given in Algorithm 6.3.1, while Figure 6.3.1 gives a pictorial illustration of the encryption/decryption process. XTC can be viewed as an iteration of XTX, much like $\mathfrak{O}_{\mathsf{XTX}}$, albeit with tweak length fixed to three blocks. We call this equivalent view $\mathfrak{O}'_{\mathsf{XTX}}$. In Figure 6.3.1, this equivalent view is represented by dashed rectangles which denote the underlying XTX component.

Although $\mathfrak{O}_{\mathsf{XTX}}$ and XTC are similar in their use of XTX for tweak length extension, yet XTC is much more efficient while maintaining a satisfactory level of security. For a plaintext-ciphertext pair (P, C) we only use $(S_{i-2} = P_{i-2} \oplus C_{i-2}, P_{i-1}, C_{i-1})$ as the tweak input to the i-th block XTX. While this enables the application of efficient algebraic hash functions within XTX, the security analysis of the overall scheme becomes a bit more involved.

6.3.2.1 Design Choices and Rationale

We choose the pair $(S_{i-2}, P_{i-1}, C_{i-1})$ as it reduces the state size by one block, and is the simplest such pair. Further it can be easily verified that we cannot reduce this to 2 blocks without compromising on security. As far as the choice of hash function is concerned, we need universal property for the tweak part and XOR universal for the masking part. In other words we need H to be a pAXU hash over 3 blocks input. Since we are considering only rate-1 constructions we recommend algebraic hash functions for H.

BRW-BASED PAXU CANDIDATE: BRW hash function is an efficient candidate that requires just one multiplication when the input is restricted to three blocks. It was proposed by Bernstein [28] based on previous works by Rabin and Winograd [169]. For a 3-blocks input (a_1, a_2, a_3), $\mathsf{BRW}_x(a_1, a_2, a_3)$ is defined as:

$$\mathsf{BRW}_x(a_1, a_2, a_3) := (a_1 \oplus x) \odot (a_2 \oplus x^2) \oplus a_3,$$

where $\odot$ and $\oplus$ denote field multiplication and addition, respectively, over $\mathbb{F}_{2^n}$ generated by some predefined primitive element. It is well-known that BRW hash with three blocks input is $3 \cdot 2^{-n}$-universal [28, 46].

For $L \in \mathbb{B}^2$, let L_1 and L_2 be the most and least, respectively, significant n bits of L. We define the keyed hash function, $H_L : \mathbb{B} \times \mathbb{B} \times \mathbb{B} \to \mathbb{B} \times \mathbb{B}$ as follows

$$\forall S, P, C \in \mathbb{B}, \quad (T, U) := H_L(S, P, C) = (\mathsf{BRW}_{L_1}(C, S, P), \mathsf{BRW}_{L_2}(C, S, 0)).$$

Note that, the hash function definition is not arbitrary over the three block inputs. For example the security reduces to birthday bound if we just swap the values of T and U as after the swapping, U does not follow AXU. In Lemma 6.3.5 we bound the pAXU probability of H.

Lemma 6.3.5. *For distinct $(S, P, C), (S', P', C') \in \mathbb{B}^3$, and $\delta \in \mathbb{B}$ we have,*

$$\Pr_{\mathsf{L}}\left[H_{\mathsf{L}}(S, P, C) \oplus H_{\mathsf{L}}(S', P', C') = (0, \delta)\right] \leq \frac{9}{2^{2n}}.$$

Proof. To compute the output (T, U), H_{L} employs two calls to BRW hash with independent keys L_1 and L_2, which enables a universal bound of $9 \cdot 2^{-2n}$. But we need pAXU property, which in this case means that the second output should have AXU property. Note that, the last input block for the second call of BRW is always zero. This enables us to consider the difference block as part of the input and reduce the AXU bound to universal bound.

$$\begin{aligned}
&\Pr_{\mathsf{L}}\left[H_{\mathsf{L}}(S, P, C) \oplus H_{\mathsf{L}}(S', P', C') = (0, \delta)\right] \\
&\quad \leq \Pr_{\mathsf{L}_1}\left[\mathsf{BRW}_{\mathsf{L}_1}(C, S, P) = \mathsf{BRW}_{\mathsf{L}_1}(C', S', P')\right] \\
&\qquad \times \Pr_{\mathsf{L}_2}\left[\mathsf{BRW}_{\mathsf{L}_2}(C, S, 0) \oplus \mathsf{BRW}_{\mathsf{L}_2}(C', S', 0) = \delta\right] \\
&\quad = \Pr_{\mathsf{L}_1}\left[\mathsf{BRW}_{\mathsf{L}_1}(C, S, P) = \mathsf{BRW}_{\mathsf{L}_1}(C', S', P')\right] \\
&\qquad \times \Pr_{\mathsf{L}_2}\left[\mathsf{BRW}_{\mathsf{L}_2}(C, S, \delta) = \mathsf{BRW}_{\mathsf{L}_2}(C', S', 0)\right] \\
&\quad \leq \frac{9}{2^{2n}}.
\end{aligned}$$

□

We emphasize here that the above definition is not the only possibility. Indeed, one can use polynomial hash to further reduce the state size at the cost of two more multiplication. In general any good pAXU hash function can be employed. In this work we mainly focus on saving on the number of multiplications.

DERIVING HASH KEYS VIA TBC: XTC requires two hash keys. However the hash keys can be easily derived via TBC by reserving one tweak bit for key generation. For example one can use $(L_1, L_2) = (\widetilde{\mathsf{E}}_K^{1\|0}(0), \widetilde{\mathsf{E}}_K^{1\|1}(1))$ as the hash key and fix the most

Algorithm 6.3.1 Definition of $\mathsf{XTC}^{\widetilde{\mathsf{E}},H}$. We take $\mu = v = 0$ and $\nu = 1$.

function $\mathsf{XTC}^{+}_{\widetilde{\mathsf{E}}_K,H_L}(P)$	**function** $\mathsf{XTC}^{-}_{\widetilde{\mathsf{E}}_K,H_L}(C)$
$S_{-1} \leftarrow v$	$S_{-1} \leftarrow v$
$P_0 \leftarrow \mu$	$P_0 \leftarrow \mu$
$C_0 \leftarrow \nu$	$C_0 \leftarrow \nu$
$(P_1, \ldots, P_\ell) \xleftarrow{n} P$	$(C_1, \ldots, C_\ell) \xleftarrow{n} C$
for $i \leftarrow 1$ **to** ℓ **do**	**for** $i \leftarrow 1$ **to** ℓ **do**
$(T_i, U_i) \leftarrow H_L(S_{i-2}, P_{i-1}, C_{i-1})$	$(T_i, U_i) \leftarrow H_L(S_{i-2}, P_{i-1}, C_{i-1})$
$X_i \leftarrow P_i \oplus U_i$	$Y_i \leftarrow C_i \oplus U_i$
$Y_i \leftarrow \widetilde{\mathsf{E}}_K^{T_i}(X_i)$	$X_i \leftarrow \widetilde{\mathsf{E}}_K^{-T_i}(Y_i)$
$C_i \leftarrow Y_i \oplus U_i$	$P_i \leftarrow X_i \oplus U_i$
$S_i \leftarrow P_i \oplus C_i$	$S_i \leftarrow P_i \oplus C_i$
end for	**end for**
return $C := (C_1 \| \cdots \| C_\ell)$	**return** $P := (P_1 \| \cdots \| P_\ell)$
end function	**end function**

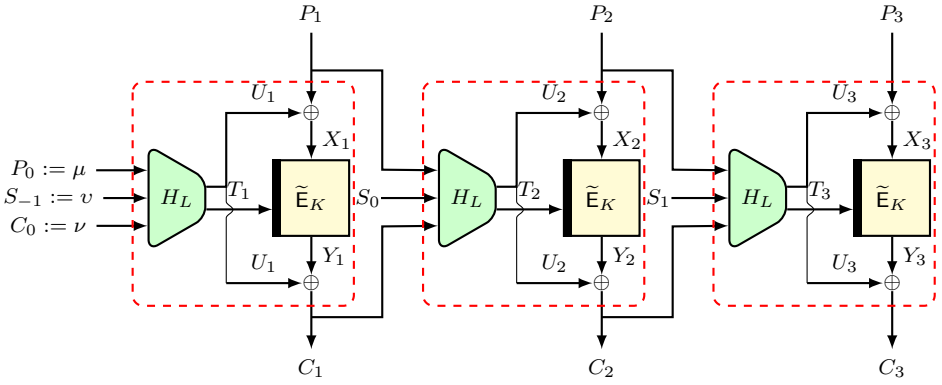

Figure 6.3.1: Schematic of the encryption/decryption process for a 3-block plaintext/-ciphertext using XTC construction. For each $i \geq 0$, $S_i := P_i \oplus C_i$. Dashed rectangles denote the XTX components of XTC and can be used to view XTC as $\mathfrak{O}'_{\mathsf{XTX}}$.

significant bit (MSB) of the tweaks for each TBC in block processing to 0. This will lead to a nominal loss of 1 bit security.

6.3.3 Security of XTC

We show that $\mathsf{XTC}^{\widetilde{\Pi},H}$ construction is secure up to $2^n/n$ block-queries given the hash function H is $O(2^{-2n})$-pAXU hash and $\widetilde{\Pi}$ is a uniform tweakable random permutation. More generally, we prove the upper bound result on the OSPRP advantage of $\mathsf{XTC}^{\widetilde{\mathsf{E}},H}$ in Theorem 6.3.6.

Theorem 6.3.6. *If* XTC *is defined as above,* H *is an* ϵ*-pAXU, and* $\sigma \leq 2^{n-3}$*, then we have*

$$\mathbf{Adv}^{\mathsf{osprp}}_{\mathsf{XTC}^{\widetilde{\mathsf{E}},H}}(q, \ell, \sigma, t) \leq \mathbf{Adv}^{\mathsf{tsprp}}_{\widetilde{\mathsf{E}}}(\sigma, t'') + \sigma^2\epsilon + \frac{2(n+2)\sigma}{2^n}.$$

PROOF OVERVIEW — Before proceeding with the proof it would be better to discuss the main crux of the proof briefly. We employ the iterated TBC view (see Section 6.1) of an online cipher. The main steps of the proof are shown in Figure 6.3.2. Basically,

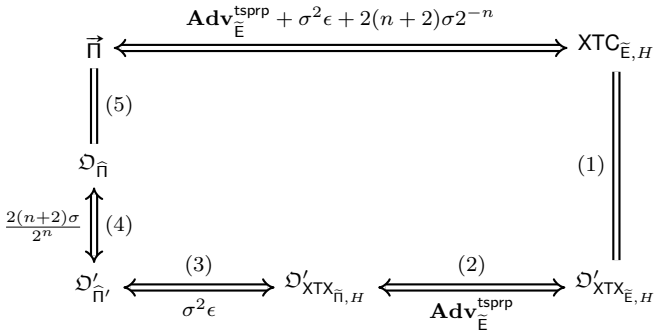

Figure 6.3.2: Main steps in the proof are given in (1)–(5). $\Longleftrightarrow$ with overlying text denotes the distance between the corresponding schemes and $=$ denotes the equivalence between the corresponding schemes. Sequence of steps: (1) XTX based view of XTC; (2) and (3) TSPRP security of XTX; (4) Distance between $\mathfrak{O}'_{\widehat{\Pi}}$ and $\mathfrak{O}_{\widehat{\Pi}}$.

we reduce the original XTC construction based on $\widetilde{\mathsf{E}}$ and H to a variant of $\mathfrak{O}_{\widehat{\mathsf{E}}}$, that we call $\mathfrak{O}'_{\widehat{\Pi}'}$ based on an ATL tweakable uniform random permutation $\widehat{\Pi}'$. $\mathfrak{O}'_{\widehat{\Pi}'}$ behaves exactly as $\mathfrak{O}_{\widehat{\Pi}}$, except that it restricts the i-th block tweak to $(S_{i-2}, P_{i-1}, C_{i-1})$. Finally, we bound the distance between $\mathfrak{O}_{\widehat{\Pi}}$ and $\mathfrak{O}'_{\widehat{\Pi}'}$. Intuitively, the two oracles will behave identically until there is a tweak collision in one of them which is not reciprocated by the other one. Now a tweak collision in $\mathfrak{O}_{\widehat{\Pi}}$ always implies a tweak collision in $\mathfrak{O}'_{\widehat{\Pi}'}$. But the converse is not true, and we bound the probability of this event to complete the proof.

Proof. We follow the series of steps shown in Figure 6.3.2. Step (1) simply transforms XTC to $\mathfrak{O}'_{\mathsf{XTX}}$ and step (5) transforms $\vec{\Pi}$ to its iterated TBC view $\mathfrak{O}_{\widehat{\Pi}}$. Steps (2) and (3) are used to replace the underlying $\mathsf{XTX}_{\widetilde{\mathsf{E}},H}$ with an ATL TRP, $\widehat{\Pi}$. Using Theorem 6.1.2, steps (2) and (3) are bounded by $\mathbf{Adv}^{\mathsf{tsprp}}_{\widetilde{\mathsf{E}}}(\sigma, t'') + \epsilon\sigma^2$. In step (4) we upper bound the distance between $\mathfrak{O}_{\widehat{\Pi}}$ and $\mathfrak{O}'_{\widehat{\Pi}'}$. We employ coefficient-H technique (see Corollary 2.2.2).

Let Ω denote the set of all attainable transcripts for the ideal oracle $\mathcal{O}_0 := \mathfrak{O}_{\widehat{\Pi}}$. A typical transcript $\omega \in \Omega$ is of the form (P^q, C^q), where $P^q, C^q \in (\mathbb{B}^{\eta^+})_q$, $|P_i|_n = p_i \leq \ell$ and $\sum_{i\in[q]} p_i \leq \sigma$. In addition, (P^q, C^q) satisfies the online property.

We say that a transcript $\omega \in \Omega$ is bad, denoted $\omega \in \Omega_{\mathsf{bad}}$, if the following condition holds:

$$\texttt{TColl}: \exists (i,j) \in [q] \times [p_i],\ (i'j') \in [q] \times [p_{i'}],\ \text{such that}$$
$$P_i^{(j-1]} \neq P_{i'}^{(j'-1]} \wedge (S_{(i,j-2)}, P_{(i,j-1)}, C_{(i,j-1)}) = (S_{(i',j'-2)}, P_{(i',j'-1)}, C_{(i',j'-1)}).$$

It is quite straightforward to see that for all $\omega \in \Omega \setminus \Omega_{\mathsf{bad}}$, we have

$$\Pr[\Theta_1 = \omega] = \Pr[\Theta_0 = \omega],$$

since $\neg\texttt{TColl}$ implies that a tweak collision in the real world means a tweak collision in the ideal world and vice-versa. So, all that we are left with is bounding the probability of TColl (viewed as the event associated with the predicate TColl).

BOUND ON $\Pr[\texttt{TColl}]$: In the following discussion we assume that all the queries are of encryption type. This will not hamper the correctness of our analysis due to the online property, rather it will greatly simplify the analysis. The basic idea is to first bound the number of multicollisions on $(\mathsf{S}_{(i,j-2)}, \mathsf{P}_{(i,j-1)})$, and then for each (i,j) bound the probability of $\mathsf{C}_{(i,j-1)}$ collisions over the multicollision set that contains $(\mathsf{S}_{(i,j-2)}, \mathsf{P}_{(i,j-1)})$. Note that, $(\mathsf{P}_i^{(j-1]} \neq \mathsf{P}_{i'}^{(j'-1]}) \wedge (\mathsf{P}_{(i,j-1)} = \mathsf{P}_{(i',j'-1)}) \implies (\mathsf{P}_i^{(j-2]} \neq \mathsf{P}_{i'}^{(j'-2]})$, otherwise the probability of TColl is zero. Let

$$\mathcal{I} := \left\{ (u,v) \in [q] \times [p_u] \ : \forall (u',v') < (u,v) \in [q] \times [p_{u'}],\ \mathsf{P}_u^{(v]} \neq \mathsf{P}_{u'}^{(v']} \right\}.$$

We define the multicollision relation $\sim$ on $\mathcal{I}$ as follows:

$$\forall (u,v), (u',v') \in \mathcal{I},\ (u,v) \sim (u',v') \iff (\mathsf{S}_{(u,v-1)}, \mathsf{P}_{(u,v)}) = (\mathsf{S}_{(u',v'-1)}, \mathsf{P}_{(u',v')}).$$

Clearly $\sim$ is an equivalence relation. Let $\mathcal{P}_\alpha$ denote the equivalence class containing $\alpha \in \mathcal{I}$ and $\#\texttt{mc} := \max_{\alpha \in \mathcal{I}} |\mathcal{P}_\alpha|$. Let $\texttt{MC}_n$ denote the event $\#\texttt{mc} > n$. We make the following claim on $\#\texttt{mc}$.

Claim 6.3.7. For $\sigma \leq 2^{n-3}$, we have

$$\Pr[\texttt{MC}_n] \leq \frac{4\sigma}{2^n}.$$

We are interested in the conditional probability of TColl given $\neg\texttt{MC}_n$. It is clear that for each $(i, j-1) \in \mathcal{I}$ we have $\left|\mathcal{P}_{(i,j-1)}\right| \leq n$, and we have to bound the probability of $\mathsf{C}_{(i,j-1)}$ collisions for at most these many pairs.

$$\begin{aligned}
\Pr[\texttt{TColl}] &\leq \Pr[\texttt{MC}_n] + \Pr[\texttt{TColl} \mid \neg\texttt{MC}_n] \\
&\overset{1}{\leq} \Pr[\texttt{MC}_n] + \sum_{(i,j-1) \in \mathcal{I}} \sum_{(i',j') \in \mathcal{P}_{(i,j-1)}} \Pr\left[\mathsf{C}_{(i,j-1)} = \mathsf{C}_{(i',j')}\right] \\
&\overset{2}{\leq} \Pr[\texttt{MC}_n] + \sum_{(i,j-1) \in \mathcal{I}} \sum_{(i',j') \in \mathcal{P}_{(i,j-1)}} \frac{1}{2^n - s_{(i,j-2)}} \\
&\overset{3}{\leq} \Pr[\texttt{MC}_n] + \sum_{(i,j-1) \in \mathcal{I}} \sum_{(i',j') \in \mathcal{P}_{(i,j-1)}} \frac{1}{2^n - \sigma}
\end{aligned}$$

$$\overset{4}{\leq} \Pr\left[\mathtt{MC}_n\right] + \sum_{(i,j-1)\in\mathcal{I}} \frac{n}{2^n - \sigma}$$
$$\overset{5}{\leq} \frac{4\sigma}{2^n} + \frac{2n\sigma}{2^n}.$$

Note that, $\mathsf{P}_i^{[j-2]} \neq \mathsf{P}_{i'}^{[j'-1]}$, so the tweaks are distinct, whence the transition from 1 to 2, where $s_{(i,j-2)}$ denotes the number of (i_1, j_1) such that $\mathsf{P}_{i_1}^{[j_1]} = \mathsf{P}_i^{[j-2]}$. Using $s_{(i,j-2)} \leq \sigma$ we have 2 to 3. Using $\#\mathtt{mc} \leq n$, $|\mathcal{I}| \leq \sigma$ and Claim 5.1 we have 3 to 5. □

Proof of Claim 6.3.7: Let $\#\mathtt{mc} = m$ and for some $\alpha \in \mathcal{I}$, let $|\mathcal{P}_\alpha| = m$. For each $(u, v) \in \mathcal{P}_\alpha$, let $s_{(u,v)}$ denote the number of $(u', v') \in \mathcal{I}$ such that $\mathsf{P}_{u'}^{[v']} = \mathsf{P}_u^{[v]}$. Let us fix one index, say lexicographically first one, (u_1, v_1) as our reference index. Then we get a system of $(m-1)$ equations of the form $\{\mathsf{S}_{(u_1,v_1-1)} = \mathsf{S}_{(u_a,v_a-1)}\}_{a\in[2\ldots m]}$. We can argue that all these equations are independent as $\mathsf{P}_{u_a}^{[v_a-1]} \neq \mathsf{P}_{u_b}^{[v_b-1]}$ where $a \neq b \in [m]$. So, we have

$$\Pr\left[\#\mathtt{mc} = m\right] \leq \frac{\binom{\sigma}{m}}{(2^n - s_{(i_2,j_2)}) \cdots (2^n - s_{(i_m,j_m)})} \leq \frac{\binom{\sigma}{m}}{(2^n - \sigma)^{m-1}}.$$

Summing over all $m \geq n+1$, we have

$$\begin{aligned}
\Pr\left[\mathtt{MC}_n\right] &= \sum_{m=n+1}^{\infty} \Pr\left[\#\mathtt{mc} = m\right] \\
&\overset{1}{\leq} \sum_{m=n+1}^{\infty} \frac{\binom{\sigma}{m}}{(2^n - \sigma)^{m-1}} \\
&\overset{2}{\leq} \frac{1}{2} \sum_{m=n+1}^{\infty} \left(\frac{4\sigma}{2^n}\right)^m \\
&\overset{3}{\leq} \frac{4\sigma}{2^n}.
\end{aligned}$$

From 1 to 2 we use the fact that $\sigma < 2^{n-1}$, and $n(m-1) > (n-1)m$ whenever $m > n$. Using the convergence result on infinite geometric series and assuming $\sigma \leq 2^{n-3}$, we get the result from 2 to 3. □

As a direct consequence of Theorem 6.3.6 and Lemma 6.3.5 we get the following corollary for the BRW-based instantiation of H.

Corollary 6.3.8. *If H is defined as in Section 6.3.2.1, and $\sigma \leq 2^{n-3}$, then we have*

$$\mathbf{Adv}^{\mathsf{osprp}}_{\mathsf{XTC}_{\widetilde{\mathsf{E}},H}}(q, \ell, \sigma, \tau) \leq \mathbf{Adv}^{\mathsf{tsprp}}_{\widetilde{\mathsf{E}}}(\sigma, \tau'') + \frac{9\sigma^2}{2^{2n}} + \frac{2(n+2)\sigma}{2^n}.$$

6.3.4 Extending XTC to Handle Arbitrary Tweak Lengths

In the following discussion, we extend the initial XTC scheme with BRW based hash function to handle arbitrary tweak lengths.

- For $\tau < n$, we replace the tweak generating part of H by $\mathsf{msb}_\tau(\mathsf{BRW}_{L_1}(C, S, 0) \odot (P \oplus L_1^3))$. It can be easily shown that $\mathsf{BRW}_{L_1}(C, S, 0) \odot (P \oplus L_1^3)$ is a $6/2^n$-AXU hash. Using Proposition 2.3.5, we can establish that $\mathsf{msb}_t(\mathsf{BRW}_{L_1}(C, S, 0) \odot (P \oplus L_1^3))$ is a $6/2^\tau$-AU hash. Finally, this gives a bound of $\min(2^n/n, 2^{\frac{n+\tau}{2}})$ block-queries.
- For $\tau > n$, we already know a 2^n block-queries attack (using Theorem 6.3.2) on XTC. While we cannot improve over it without incorporating significantly more previous blocks, we can still get a sub-optimal upper bound of $2^n/n$ block-queries. This is achieved by padding the n-bit tweak generated by the hash function with zeros to make it τ-bit. Clearly, the previous bounds are directly applicable.

Combining the two cases we conclude that XTC is secure while $\sigma \ll \min(2^n/n, 2^{\frac{n+\tau}{2}})$.

Chapter 7

Security of Cascaded LRW2

In the previous chapter, we discussed online ciphers based on tweakable block ciphers (TBCs). This chapter studies the security of a popular way of constructing TBCs using standard block ciphers, called the *Cascaded LRW2* or CLRW2. Landecker et al. [123] first suggested the cascading of two independent LRW2 [125] instances to get a beyond the birthday bound (BBB) secure TBC, called CLRW2. They proved that CLRW2 is a secure TBC up to approx. $2^{2n/3}$ queries. Later, Procter [168] pointed out a flaw in the security proof, which was subsequently fixed by both Landecker et al. and Procter to recover the claimed security bounds. Lampe and Seurin [122] studied the $\ell \geq 2$ independent cascades of LRW2, and proved that it is secure up to approx. $2^{\frac{\ell n}{\ell+2}}$ queries.

In [132], Mennink presented an improved security analysis of CLRW2. The major contribution was an attack in approx. $n^{1/2}2^{3n/4}$ queries (see section 7.2.2). Following on the insights from the attack, he also gave a security proof of the same order under three assumptions:

1. The hash functions are 4-wise independent AXU.
2. The maximum number of tweak repetitions is restricted to $2^{n/4}$.
3. A limited variant of Patarin's mirror theory [160, 161] is true for $q < 2^{3n/4}$.

Among the three assumptions, the first two are at least plausible. But the last assumption is questionable as barring certain restricted cases, the proof of mirror theory has several critical gaps which are still open or unproven [54, 58]. In this chapter we explore the possibility of relaxing the above mentioned assumptions while maintaining a similar security bound.

7.1 A Note on Patarin's Mirror Theory

In [160] Patarin defines Mirror theory as a technique to estimate the number of solutions of linear systems of equalities and linear non equalities in finite groups. In its most general case, the mirror theory proof is tractable up to the order of $2^{2n/3}$ security bound, but it readily becomes complex and extremely difficult to verify, as one aims for the optimal bound [54, 58]. We remark here that this in no way suggests that the result is incorrect, and in future, we might even get some independent verifications of the result.

We restrict ourselves to the binary field $\mathbb{B}$ with $\oplus$ as the group operation. We will use Mennink and Neves interpretation [133] of mirror theory. For ease of understanding and notational coherency, we sometime use different parametrization and naming conventions. Let $q \geq 1$ and let $\mathcal{L}$ be the system of linear equations

$$\{e_1 : Y_1 \oplus V_1 = \lambda_1,\ \ e_2 : Y_2 \oplus V_2 = \lambda_2,\ \ \ldots,\ \ e_q : Y_q \oplus V_q = \lambda_q\}$$

where Y^q and V^q are unknowns, and $\lambda^q \in (\mathbb{B})^q$ are knowns. In addition there are (in)equality restrictions on Y^q and V^q, which uniquely determine $\widehat{Y^q}$ and $\widehat{V^q}$. We assume that $\widehat{Y^q}$ and $\widehat{V^q}$, are indexed in an arbitrary order by the index sets $[q_Y]$ and $[q_V]$, where $q_Y = |\widehat{Y^q}|$ and $q_V = |\widehat{V^q}|$. This assumption is without any loss of generality as this does not affect the system $\mathcal{L}$. Given such an indexing, we can define two surjective index mappings:

$$\varphi_Y : \begin{cases} [q] \to [q_Y] \\ i \mapsto j \text{ if and only if } Y_i = \widehat{Y}_j. \end{cases} \qquad \varphi_V : \begin{cases} [q] \to [q_V] \\ i \mapsto j \text{ if and only if } V_i = \widehat{V}_j. \end{cases}$$

It is easy to verify that $\mathcal{L}$ is uniquely determined by $(\varphi_Y, \varphi_V, \lambda^q)$, and vice-versa. Consider a labeled bipartite graph $\mathcal{G}(\mathcal{L}) = ([q_Y], [q_V], \mathcal{E})$ associated with $\mathcal{L}$, where $\mathcal{E} = \{(\varphi_Y(i), \varphi_V(i), \lambda_i) : i \in [q]\}$, λ_i being the label of edge. Clearly, each equation in $\mathcal{L}$ corresponds to a unique labeled edge (assuming no duplicate equations). We give three definitions with respect to the system $\mathcal{L}$ using $\mathcal{G}(\mathcal{L})$.

Definition 7.1.1 (cycle-freeness)**.** $\mathcal{L}$ is said to be cycle-free if and only if $\mathcal{G}(\mathcal{L})$ is acyclic.

Definition 7.1.2 ($\xi_{\max}$-component)**.** Two distinct equations (or unknowns) in $\mathcal{L}$ are said to be in the same component if and only if the corresponding edges (res. vertices) in $\mathcal{G}(\mathcal{L})$ are in the same component. The size of any component $\mathcal{C}$ in $\mathcal{L}$, denoted $\xi(\mathcal{C})$, is the number of vertices in the corresponding component of $\mathcal{G}(\mathcal{L})$, and the maximum component size is denoted by $\xi_{\max}(\mathcal{L})$ (or simply $\xi_{\max}$).

Definition 7.1.3 (non-degeneracy)**.** $\mathcal{L}$ is said to be non-degenerate if and only if there does not exist a path of length at least 2 in $\mathcal{G}(\mathcal{L})$ such that the labels along the edges on this path sum up to zero.

Theorem 7.1.4 (Fundamental Theorem of Mirror Theory [160])**.** *Let $\mathcal{L}$ be a system of equations over the unknowns $(\widehat{Y^q}, \widehat{V^q})$, that is (i) cycle-free, (ii) non-degenerate, and (iii) $\xi_{\max}^2 \cdot \max\{q_Y, q_V\} \leq 2^n$. Then, the number of solutions $(y_1, \ldots, y_{q_Y}, v_1, \ldots, v_{q_V})$ of $\mathcal{L}$, denoted h_q, such that $y_i \neq y_j$ and $v_i \neq v_j$ for all $i \neq j$, satisfies*

$$h_q \geq \frac{(2^n)_{q_Y} \cdot (2^n)_{q_V}}{2^{nq}}. \tag{7.1}$$

A sketchy proof for this theorem is given in [160]. As mentioned before, the proof is quite involved with some claims remaining open or unproved. On the other hand, the same paper contains results for various other cases. For instance, for $\xi = 2$, several sub-optimal bounds have been shown. By sub-optimal, we mean that a factor of $(1 - \epsilon)$, for some $\epsilon > 0$, is multiplied to the right hand side of Eq. (7.1). Inspired by this, we give the following terminology which will be useful in later references to mirror theory.

> For $\xi \geq 2$, $\epsilon > 0$, we write (ξ, ϵ)-restricted mirror theory theorem to denote the mirror theory result in which the number of solutions, h_q, of a system of equations with $\xi_{\max} = \xi$, satisfies $h_q \geq (1 - \epsilon) \frac{(2^n)_{q_Y} \cdot (2^n)_{q_V}}{2^{nq}}$.

Here ϵ can be viewed as the degree of deviation from random function behavior. Mirror theory has been primarily used for bounding the pseudorandomness of some random system with respect to a random function. Accordingly, one finds a term of the form 2^{nq} in mirror theory bounds. When combined with the coefficient-H technique, we get an ϵ term in the distinguishing advantage bound.

In [132], $(4, q^4/2^{3n})$-restricted mirror theory theorem is used. In section 7.4, we study the $(\xi, q^4/2^{3n})$ case, for $\xi \in \{2, 3\}$ and present a variant of mirror theory suitable for tweakable permutation scenario.

7.2 Revisiting Mennink's Improved Bound on CLRW2

Recall the notion of ℓ-wise independent XOR universal hash functions defined in Definition 2.3.3 of chapter 2. This notion will be used for the description of CLRW2 (for $\ell = 2$), as well as Mennink's improved bound on CLRW2 (for $\ell = 4$).

7.2.1 Description of the Cascaded LRW2 Construction

Let $\mathsf{E} \in \mathsf{BPerm}(\{0,1\}^\kappa, \mathbb{B})$ be a block cipher. Let $\mathcal{H}$ be a hash function family from $\mathbb{T}$ to $\mathbb{B}$. We define the tweakable block cipher $\mathsf{LRW2}_{\mathsf{E},\mathcal{H}}$, based on the block cipher E and the hash function family $\mathcal{H}$, by the following mapping: $\forall (k,h,t,m) \in \{0,1\}^\kappa \times \mathcal{H} \times \mathbb{T} \times \mathbb{B}$,

$$\mathsf{LRW2}_{\mathsf{E},\mathcal{H}}(k,h,t,m) := \mathsf{E}_k(m \oplus h(t)) \oplus h(t). \tag{7.2}$$

For $r \in \mathbb{N}$, the r-round cascaded LRW2 construction, denoted $\mathsf{CLRW2}_{\mathsf{E},\mathcal{H},r}$, is a cascade of r independent LRW2 instances, i.e. $\mathsf{CLRW2}_{\mathsf{E},\mathcal{H},r}$ is a tweakable block cipher, based on the block cipher E and the hash function family $\mathcal{H}$, defined as follows: $\forall (k^r, h^r, t, m) \in \{0,1\}^{r\kappa} \times \mathcal{H}^r \times \mathbb{T} \times \mathbb{B}$,

$$y_i := \begin{cases} \mathsf{LRW2}_{\mathsf{E},\mathcal{H}}(t,m) & \text{for } i = 1, \\ \mathsf{LRW2}_{\mathsf{E},\mathcal{H}}(t,y_{i-1}) & \text{otherwise.} \end{cases}$$
$$\mathsf{CLRW2}_{\mathsf{E},\mathcal{H},r}(k^r,h^r,t,m) := y_r. \tag{7.3}$$

The 2-round CLRW2, was first analyzed by Landecker et al. [123], whereas the $r > 2$ case was studied by Lampe and Seurin [122]. Since we mainly focus on the $r = 2$ case, we use the nomenclatures, CLRW2 and cascaded LRW2, interchangeably with 2-round CLRW2. Figure 7.2.1 gives a pictorial description of the cascaded LRW2 construction. In [123] the CLRW2 construction was shown to be a BBB secure (up to $2^{2n/3}$ queries)

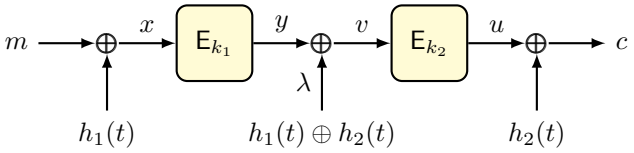

Figure 7.2.1: The cascaded LRW2 construction.

TSPRP, provided the underlying block cipher is an SPRP, and the hash function families are AXU. However, an attack with matching bounds eluded up until quite recently.

7.2.2 Mennink's Attack on CLRW2

In [132] Mennink gave an $O(n^{1/2}2^{3n/4})$ query attack on CLRW2. The attack is generic in nature as it does not exploit the weaknesses in the underlying block cipher. Rather it assumes that the block cipher instances are independent random permutations. Also the attack works for any hash function. We briefly describe the attack and refer the readers to [132] for a more concrete and formal description, analysis and experimental verification of the attack.

ATTACK DESCRIPTION: Suppose in the transcript generated by a distinguisher, there exist four queries (t, m_1, c_1), (t', m_2, c_2), (t, m_3, c_3), and (t', m_4, c_4), such that the following equations hold:

$$\begin{aligned} m_1 \oplus h_1(t) &= m_2 \oplus h_1(t') \\ c_2 \oplus h_2(t') &= c_3 \oplus h_2(t) \\ m_3 \oplus h_1(t) &= m_4 \oplus h_1(t') \end{aligned} \tag{7.4}$$

Using notations analogous to Figure 7.2.1, we equivalently have, $x_1 = x_2$; $u_2 = u_3$; and $x_3 = x_4$. Since $x^4 \leftrightsquigarrow y^4$ and $v^4 \leftrightsquigarrow u^4$, looking at the equations generated by the corresponding y and v values, we have $v_1 = y_1 \oplus \lambda(t) = y_2 \oplus \lambda(t) = v_2 \oplus \lambda(t') \oplus \lambda(t) = v_3 \oplus \lambda(t) \oplus \lambda(t') = y_3 \oplus \lambda(t') = v_4$. This immediately gives $u_1 = u_4$, i.e.

$$c_4 \oplus h_2(t') = c_1 \oplus h_2(t). \tag{7.5}$$

In other words, Eq. (7.5) is implied by the existence of Eq. (7.4), and by combining all four equations, we have

$$\begin{aligned} m_1 \oplus m_2 = m_3 \oplus m_4 = \alpha, \\ c_1 \oplus c_4 = c_2 \oplus c_3 = \beta, \end{aligned}$$

where $\alpha = h_1(t) \oplus h_1(t')$ and $\beta = h_2(t) \oplus h_2(t')$. While the distinguisher does not know α and β, it can exploit the relations:

$$m_1 \oplus m_2 = m_3 \oplus m_4, \tag{7.6}$$

$$c_1 \oplus c_4 = c_2 \oplus c_3, . \tag{7.7}$$

If for some value a we have about 2^n many quadruples satisfying

$$m_1 \oplus m_2 = m_3 \oplus m_4 = a, \tag{7.8}$$

then, for CLRW2, the expected number of solutions for Eq. (7.6)-(7.7) is approximately 2 for $a = \alpha$. On the other hand, for $\widetilde{\Pi}$, the expected number of solutions is always close to 1 for any $a \in \mathbb{B}$. In [132], it has been shown that approximately $2n^{1/2}2^{3n/4}$ queries are sufficient for the distinguisher to ensure that Eq. (7.8) has about 2^n solutions. Given these many queries the distinguisher can attack by observing the number of solutions for Eq. (7.6)-(7.7) for each value of a.

7.2.3 Mennink's Proof Approach

The proof in [132] applies coefficient-H technique coupled with mirror theory. The main focus is to identify a suitable class of bad events on (x^q, u^q), where q is the number of queries, which makes mirror theory inapplicable. Crudely, the bad events correspond to cases where for some query there is no randomness left (in the sampling of y^q and v^q) in the ideal world. Given a good transcript, mirror theory is applied to bound the number of solutions of the system of equation $\{Y_i \oplus V_i = \lambda_i : i \in [q]\}$, where Y_i and V_i are unknowns satisfying $x^q \leftrightsquigarrow Y^q$ and $V^q \leftrightsquigarrow u^q$, and λ^q is fixed. The proof relies on three major assumptions:

Assumption 1. $\mathcal{H}$ is AXU_4 hash function family.

Assumption 2. For any $t' \in \mathbb{T}$, $\mu_{t'} = \mu(t^q, t') \leq \gamma = 2^{n/4}$.

Assumption 3. $\left(4, \frac{q^4}{2^{3n}}\right)$-restricted mirror theory theorem is correct.

TRANSCRIPT GRAPH: A graphical view on x^q and u^q was used to characterize all bad events. Basically, each transcript is mapped to a unique bipartite graph on x^q, u^q, as defined in Definition 7.2.1.

Definition 7.2.1 (Transcript Graph)**.** A transcript graph $\mathcal{G} = (\mathcal{X}^q, \mathcal{U}^q, \mathcal{E}^q)$ associated with (x^q, u^q), denoted $\mathcal{G}(x^q, u^q)$, is defined as a bipartite graph with vertex sets $\mathcal{X} := \{(x_i, 0) : i \in [q]\}$; $\mathcal{U} := \{(u_i, 1) : i \in [q]\}$; and edge set $\mathcal{E} := \{((x_i, 0), (u_i, 1)) : i \in [q]\}$. We also associate the value $\lambda_i = h_1(t_i) \oplus h_2(t_i)$ with edge $((x_i, 0), (u_i, 1)) \in \mathcal{E}$.

Note that, the graph may not be simple, i.e., it can contain parallel edges. Here $\mathcal{X}$ and $\mathcal{U}$ are just the disjointified representations of x^q and u^q, respectively. For all practical purposes we may drop the 0 and 1 for $(x, 0) \in \mathcal{X}$ and $(u, 1) \in \mathcal{U}$, as they can be easily distinguished. Further, for some $i, j \in [q]$, if $x_i = x_j$ (or $u_i = u_j$) , then they share the same vertex $x_i = x_j = x_{i,j}$ (or $u_i = u_j = u_{i,j}$). The event $x_i = x_j$ and $u_i = u_j$, although extremely unlikely, will lead to a parallel edge in $\mathcal{G}$. Finally each edge $(x_i, u_i) \in \mathcal{E}$ corresponds to a query index $i \in [q]$, so we can equivalently view (and call) the edge (x_i, u_i) as index i. Figure 7.2.2 gives an example graph for $\mathcal{G}$.

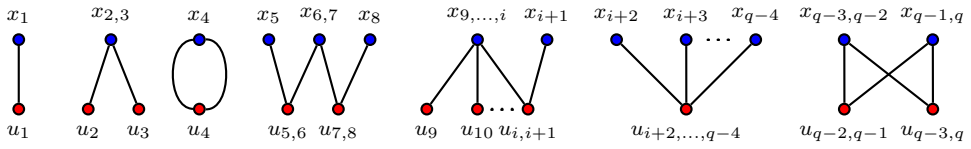

Figure 7.2.2: A possible transcript graph $\mathcal{G}(x^q, u^q)$ associated with (x^q, u^q). Vertices in x^q are colored blue and vertices in u^q are colored red, for illustration only.

BAD TRANSCRIPTS: A transcript graph $\mathcal{G}(x^q, u^q)$ is called bad if:

1. it has a cycle of size $= 2$.
2. it has two adjacent edges i and j such that $\lambda_i \oplus \lambda_j = 0$.
3. it has a component of length ≥ 4 edges.

All subgraphs in Figure 7.2.2, except the first two from left, are considered bad in [132]. Conditions 1 and 2 correspond to the cases which might lead to degeneracy in the real world. Condition 3 may lead to a cycle of length ≥ 4 edges. The non-fulfillment of condition 1,2 and 3 satisfies the cycle-free and non-degeneracy properties required in mirror theory. It also bounds $\xi_{\max} \leq 4$. Condition 1 and 2 contribute small and insignificant terms and can be ignored from this discussion. We focus on the major bottleneck, i.e. condition 3. The subgraphs corresponding to condition 3 are given in Figure 7.2.3. Configuration (D), (E), and (F) are symmetric to (A), (B), and (C). So we can study (A), (B), and (C), and the other three can be similarly analyzed.

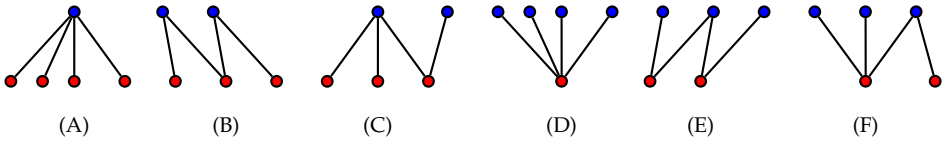

Figure 7.2.3: Possible configuration of size $= 4$ edge subgraphs. Vertices in x^q are colored blue and vertices in u^q are colored red, and vertex labels are omitted for brevity.

BOTTLENECK 1: BOUND ON THE PROBABILITY OF (A), (C), (D) AND (F) — This can be divided into two parts:

(a) Configuration (A) arises for the event

$$\exists^* i, j, k, l \text{ such that } x_i = x_j = x_k = x_\ell.$$

This event is upper bounded to $q^4\epsilon^3$ using assumption 1 on hash functions. Similar argument holds for (D).

(b) Configuration (C) (similarly for F) arises for the event

$$\exists^* i, j, k, \ell \in [q] \text{ such that } x_i = x_j = x_k \wedge u_k = u_\ell.$$

In this case we can apply assumption 1 (even AXU_3 would suffice) to get an upper bound of $q^4\epsilon^3$.

BOTTLENECK 2: BOUND ON THE PROBABILITY OF (B) — Configuration (B) arises for the event

$$\exists^* i, j, k, l \text{ such that } x_i = x_j \wedge u_j = u_k \wedge x_k = x_\ell.$$

This is probably the trickiest case, which requires assumption 2, i.e. restriction on tweak repetition. Specifically consider the case $t_i = t_k$ and $t_j = t_\ell$. This is precisely the case exploited in Mennink's attack on CLRW2 [132] (see subsection 7.2.2). In this case for a fixed i, j, k, ℓ the probability is bounded by ϵ^2. There are at most q^2 many choices for (i, j), at most $(\mu_{t_i} - 1)$ choices for k and a single choice for ℓ given i, j and k. Thus the probability is bounded by $q^2\gamma\epsilon^2$ (using assumption 2). Similar argument holds for (E).

BOTTLENECK 3: MIRROR THEORY BOUND — The final hurdle is the use of mirror theory in computation of real world interpolation probability, which requires assumption 3. Yet another issue is the nature of the mirror theory bound. A straightforward application of mirror theory bound leads to a term of the form

$$\boxed{\frac{\prod_{t' \in \hat{t}^q} (2^n)_{\mu_{t'}}}{(2^n)^q}} (1 - O(q/2^n)) \,,$$

in the ratio of interpolation probabilities, where $\sum_{t' \in \hat{t}^q} \mu_{t'} = q$. Here the numerator is due to the tweakable random permutation. In the worst case, $\mu_{t'} = O(q)$, which gives a lower bound of the form $1 - q^2/2^n$. But using assumption 2, we get a bound of $1 - q\gamma/2^n$ as $\mu_{t'} \leq \gamma$.

7.2.3.1 Severity of the Assumptions in [132]

Among the three assumptions, assumption 1 and 2 are quite plausible in the sense that real life use-cases exist for assumption 2 and practical instantiations are possible for assumption 1. Another point of note is the fact that $\gamma < 2^{n/4}$ is imposed only due to bottleneck 3. Otherwise a better bound of $\gamma < 2^{n/2}$ could have been used. While assumption 1 and 2 are plausible to a large extent, assumption 3 is disputable.

Although the proof in [132] requires the above mentioned assumptions, the proof approach seems quite simple and in some cases it highlights the bottlenecks in getting tight security. In the remainder of this chapter, we aim to resolve all the bottlenecks discussed here, while relaxing all the assumptions made in [132]. Specifically, bottleneck 1(a) is incorporated within good transcripts, bottleneck 2 is resolved using the tools from section 7.3, and bottleneck 3 is resolved using the tools from section 7.4 and a careful application of the expectation method in section 7.5. The only unresolved bottleneck is 1(b) for which we will use the AXU_3 assumption on hash functions.

7.3 The Alternating Collisions and Events Lemmata

Let $\mathcal{H} = \{h \mid h : \mathcal{T} \to \mathcal{B}\}$ be an ϵ-AU hash function family. A pair (t_i, t_j) is called a colliding pair for some $h \in \mathcal{H}$ if $h(t_i) = h(t_j)$. The probability of having at least one colliding pair from q distinct elements $t^q \in (\mathcal{T})_q$ for a randomly chosen hash function $\mathsf{H} \xleftarrow{\$} \mathcal{H}$ is at most $\binom{q}{2} \cdot \epsilon$. This is straightforward from the union bound.

Suppose $\mathsf{H}_1, \mathsf{H}_2 \xleftarrow{\$} \mathcal{H}$ are two independently drawn universal hash functions and $t^q \in (\mathcal{T})_q$. Then, by applying independence and union bound, we have

$$\Pr\left[\exists^* i, j, k \in [q],\ \mathsf{H}_1(t_i) = \mathsf{H}_1(t_j) \ \wedge\ \mathsf{H}_2(t_j) = \mathsf{H}_2(t_k)\right] \leq q(q-1)(q-2) \cdot \epsilon^2.$$

Now we go one step further. We would like to bound the probability of the following event:

$$\exists^* i, j, k, l \in [q],\ \mathsf{H}_1(t_i) = \mathsf{H}_1(t_j) \ \wedge\ \mathsf{H}_2(t_j) = \mathsf{H}_2(t_k) \ \wedge\ \mathsf{H}_1(t_k) = \mathsf{H}_1(t_l).$$

For any fixed distinct i, j, k and l, we cannot claim that the probability of the event $\mathsf{H}_1(t_i) = \mathsf{H}_1(t_j) \ \wedge\ \mathsf{H}_2(t_j) = \mathsf{H}_2(t_k) \ \wedge\ \mathsf{H}_1(t_k) = \mathsf{H}_1(t_l)$ is ϵ^3 as the first and last event are no longer independent. In this section we show how we can get an improved bound even in the dependent situation. In particular, we prove the following lemma.

Lemma 7.3.1 (Alternating Collisions Lemma)**.** *Suppose* $\mathsf{H}_1, \mathsf{H}_2 \xleftarrow{\$} \mathcal{H}$ *are two independently drawn* ϵ*-universal hash functions and* $t^q \in (\mathcal{T})_q$*. Then,*

$$\Pr\left[\exists^*\ i, j, k, l \in [q],\ \mathsf{H}_1(t_i) = \mathsf{H}_1(t_j) \ \wedge\ \mathsf{H}_1(t_k) = \mathsf{H}_1(t_l) \ \wedge\ \mathsf{H}_2(t_j) = \mathsf{H}_2(t_k)\right] \leq q^2\epsilon^{1.5}.$$

Proof. For any $h \in \mathcal{H}$, we define the following useful set:

$$\mathcal{I}_h = \{(i, j) : h(t_i) = h(t_j)\}.$$

Let us denote the size of the above set by I_h. So, I_h is the number of colliding pairs for the hash functions h. We also define a set $\mathcal{H}_{\leq} = \{h : I_h \leq \frac{1}{\sqrt{\epsilon}}\}$ which collects all hash functions having a small number of colliding pairs. We denote the complement set by $\mathcal{H}_{>}$. Now, by using double counting of the set $\{(h, i, j) : h(t_i) = h(t_j)\}$ we get

$$\sum_h I_h \leq q(q-1) \cdot \epsilon \times |\mathcal{H}|. \tag{7.9}$$

Basically for every h, we have exactly I_h many choices of (i, j) and so the size of the set $\{(h, i, j) : h(t_i) = h(t_j)\}$ is exactly $\sum_h I_h$. On the other hand, for any $i < j \in [q]$, there are at most $\epsilon \cdot |\mathcal{H}|$ many hash functions h, such that (t_i, t_j) is a colliding pair for h.

This follows from the definition of the universal hash function. From Eq. (7.9) and the definition of $\mathcal{H}_{\leq}$, we have

$$\frac{|\mathcal{H}_{>}|}{\sqrt{\epsilon}} + \sum_{h \in \mathcal{H}_{\leq}} I_h \leq \sum_{h} I_h \leq q(q-1) \cdot \epsilon \cdot |\mathcal{H}|. \tag{7.10}$$

Let $\mathtt{E}$ denote the event that there exists distinct i, j, k, l such that $\mathsf{H}_1(t_i) = \mathsf{H}_1(t_j) \wedge \mathsf{H}_1(t_k) = \mathsf{H}_1(t_l) \wedge \mathsf{H}_2(t_j) = \mathsf{H}_2(t_k)$. Now we proceed to bound the probability of this event.

$$\begin{aligned}
\Pr[\mathtt{E}] &= \sum_{h} \Pr[\mathtt{E} \wedge \mathsf{H}_1 = h] \\
&= \sum_{h} \Pr[\mathsf{H}_1 = h] \times \Pr[\mathtt{E} \wedge \mathsf{H}_1 = h \mid \mathsf{H}_1 = h] \\
&\leq \sum_{h} \Pr[\mathsf{H}_1 = h] \times \min\{1, I_h^2 \cdot \epsilon\} \\
&= \Pr[\mathsf{H}_1 \in \mathcal{H}_{>}] + \sum_{h \in \mathcal{H}_{\leq}} \Pr[\mathsf{H}_1 = h] \cdot I_h^2 \cdot \epsilon \\
&\leq \frac{|\mathcal{H}_{>}|}{|\mathcal{H}|} + \sum_{h \in \mathcal{H}_{\leq}} \frac{I_h \cdot \sqrt{\epsilon}}{|\mathcal{H}|}. \\
&= \frac{\sqrt{\epsilon}}{|\mathcal{H}|} \times \left(\frac{|\mathcal{H}_{>}|}{\sqrt{\epsilon}} + \sum_{h \in \mathcal{H}_{\leq}} I_h \right) \\
&\leq q(q-1)\epsilon^{1.5}.
\end{aligned}$$

We justify the first inequality. Let $\mathcal{I}_h$ be the set of pairs (i, j) with $h_1(t_i) = h_1(t_j)$ (i.e. $|\mathcal{I}_h| = I_h$). Given $\mathsf{H}_1 = h$, the probability of the event $\mathtt{E}$ is same as the probability of the following event:

$$\exists^{*}(i, j), (k, l) \in \mathcal{I}_h,\ \mathsf{H}_2(t_j) = \mathsf{H}_2(t_k).$$

There are at most I_h^2 many pairs of pairs and for each pair of pairs and the collision probability of $\mathsf{H}_2(t_j) = \mathsf{H}_2(t_k)$ is at most ϵ. So probability of the above event can be at most $\min\{1, I_h^2 \cdot \epsilon\}$. The last inequality follows from Eq. (7.10). □

Now we generalize the above result for a more general setting which will also be used in this chapter. The proof of this result is similar to the previous proof.

Lemma 7.3.2 (Alternating Events Lemma)**.** *Suppose for all $i < j \in [q]$, $\mathtt{E}_{i,j}$ are events, possibly dependent. Each event holds with probability at most ϵ. For any distinct i, j, k, l, $\mathtt{F}_{i,j,k,l}$ are events which holds with probability at most ϵ'. Moreover, the collection of events $(\mathtt{F}_{i,j,k,l})_{i,j,k,l}$ is independent with the collection of event $(\mathtt{E}_{i,j})_{i,j}$. Then,*

$$\Pr[\exists^{*} i, j, k, l \in [q],\ \mathtt{E}_{i,j} \wedge \mathtt{E}_{k,l} \wedge \mathtt{F}_{i,j,k,l}] \leq q^2 \cdot \epsilon \cdot \sqrt{\epsilon'}.$$

Note that, Lemma 7.3.1 is a direct corollary of the above Lemma (the event $\mathtt{E}_{i,j}$ denotes that (t_i, t_j) is a colliding pair of H_1 and $\mathtt{F}_{i,j,k,l}$ denotes that (t_j, t_k) is a colliding pair of H_2).

Proof. We follow a similar proof approach as considered in Lemma 7.3.1. We define a indicator random vector $\mathsf{I} = (\mathsf{I}_{i,j})_{i<j\in[q]}$ where $\mathsf{I}_{i,j}$ takes value 1 if $\mathtt{E}_{i,j}$ holds, otherwise zero. The sample space of the random vector is Ω, the set of all binary vectors indexed by all pairs (i, j). For any vector $w \in \Omega$, we write $\#w$ to represent the number of 1's appeared in w. Let $\Omega_{\leq} = \{w : \#w \leq \frac{1}{\sqrt{\epsilon'}}\}$ and its complement set by $\Omega_{>}$.

We define a random variable $\mathsf{N} = \sum_{i\neq j} \mathsf{I}_{i,j}$, i.e., the number of $\mathtt{E}$-events hold. As $\mathtt{E}_{i,j}$ holds with probability at most ϵ,

$$\begin{aligned} q(q-1)\epsilon &\geq \mathsf{Ex}\left[\mathsf{N}\right] \\ &= \sum_{w} \#w \cdot \Pr\left[\mathsf{I} = w\right] \\ &\geq \sum_{w\in\Omega_{\leq}} \#w \cdot \Pr\left[\mathsf{I} = w\right] + \frac{\Pr\left[\mathsf{I} \in \Omega_{>}\right]}{\sqrt{\epsilon'}}. \end{aligned} \tag{7.11}$$

Let $\mathtt{E}$ denote the event that there exists distinct i, j, k, l such that $\mathtt{E}_{i,j} \wedge \mathtt{E}_{k,l} \wedge \mathtt{F}_{i,j,k,l}$. Now we proceed for bounding the probability of the event.

$$\begin{aligned} \Pr\left[\mathtt{E}\right] &= \sum_{w} \Pr\left[\mathtt{E} \wedge \mathsf{I} = w\right] \\ &= \sum_{w} \Pr\left[\mathsf{I} = w\right] \times \Pr\left[\mathtt{E} \wedge \mathsf{I} = w \mid \mathsf{I} = w\right] \\ &\leq \sum_{w} \Pr\left[\mathsf{I} = w\right] \times \min\{1, (\#w)^2 \cdot \epsilon'\} \\ &= \Pr\left[\mathsf{I} \in \Omega_{>}\right] + \sum_{w\in\Omega_{\leq}} \Pr\left[\mathsf{I} = w\right] \cdot (\#w)^2 \cdot \epsilon' \\ &\leq \Pr\left[\mathsf{I} \in \Omega_{>}\right] + \sum_{w\in\Omega_{\leq}} \Pr\left[\mathsf{I} = w\right] \cdot \#w \cdot \sqrt{\epsilon'} \\ &= \sqrt{\epsilon'} \cdot \left(\sum_{w\in\Omega_{\leq}} \#w \cdot \Pr\left[\mathsf{I} = w\right] + \frac{\Pr\left[\mathsf{I} \in \Omega_{>}\right]}{\sqrt{\epsilon'}} \right) \\ &\leq q(q-1)\epsilon \cdot \sqrt{\epsilon'}. \end{aligned}$$

The first inequality follows exactly by the same reason argued in the proof of Lemma 7.3.1. The last inequality follows from Eq. (7.11). This completes the proof. □

7.4 Mirror Theory for Tweakable Permutations

As evident from bottleneck 4 of section 7.2.3, a straightforward application of mirror theory bound would lead to a sub-optimal bound. In order to circumvent this sub-optimality, [132] uses a restriction on tweak repetitions (assumption 2 of section 7.2.3). Specifically, a bound of the form $O(q/2^{3n/4})$ requires $\mu(t^q, t') < 2^{n/4}$ for all $t' \in \widehat{t^q}$, where t^q is the q-tuple denoting the tweaks used in the q queries. In order to avoid this assumption, we need a different approach.

A closer inspection of the mirror theory proof reveals that we can actually avoid the restrictions on tweak repetitions. In fact, rather surprisingly, we will see that tweak repetitions are actually helpful in the sense that mirror theory bound is good. In the remainder of this section, we develop a modified version of mirror theory, apt for applications in tweakable permutation settings. We will only consider $\xi_{\max}(\mathcal{L}) = 3$ case.

7.4.1 General setup and notations

Following the notations and definitions from section 7.1, consider a system of equation $\mathcal{L}$ with $\xi_{\max}(\mathcal{L}) = 3$. For brevity, we simply write "sub-system of $\mathcal{L}$ with $\xi = x$", for some $x \in \{2, 3\}$, to denote the sub-system of $\mathcal{L}$ consisting of equations belonging to components with $\xi = x$.

Following this nomenclature, let $\mathcal{C}_1$ denote the sub-system of $\mathcal{L}$ with $\xi = 2$. Let $\mathcal{C}_2$ (res. $\mathcal{C}_3$) denote the sub-system of $\mathcal{L}$ with $\xi = 3$ such that for $i \in [q]$, if equation $e_i \in \mathcal{C}_2$ (res. $e_i \in \mathcal{C}_3$) then $\mu(V^q, V_i) = 1$ (res. $\mu(Y^q, Y_i) = 1$).

For $i \in [3]$, let $c_i = |\mathcal{C}_i|$ and q_i be the number of equations in c_i, i.e. $q_1 = c_1$, $q_2 = 2c_2$, $q_3 = 2c_3$, and $q_1 + q_2 + q_3 = q$. We also write $c = c_2 + c_3$. For $i \in [q]$, let $\delta_i := \mu(\lambda^{i-1}, \lambda_i)$, where $\delta_1 = 0$ by convention.

Note that, the equations in $\mathcal{L}$ can be arranged in any arbitrary order without affecting the number of solutions. For the sake of simplicity, we fix the following order on $\mathcal{L}$:

$$\underbrace{\begin{array}{c} Y_1 \oplus V_1 = \lambda_1 \\ \vdots \\ Y_{q_1} \oplus V_{q_1} = \lambda_{q_1} \end{array}}_{\mathcal{C}_1} \Bigg| \underbrace{\begin{array}{c} Y_{q_1+1} \oplus V_{q_1+1} = \lambda_{q_1+1} \\ \vdots \\ Y_{q_1+q_2} \oplus V_{q_1+q_2} = \lambda_{q_1+q_2} \end{array}}_{\mathcal{C}_2} \Bigg| \underbrace{\begin{array}{c} Y_{q_1+q_2+1} \oplus V_{q_1+q_2+1} = \lambda_{q_1+q_2+1} \\ \vdots \\ Y_{q_1+q_2+q_3} \oplus V_{q_1+q_2+q_3} = \lambda_{q_1+q_2+q_3} \end{array}}_{\mathcal{C}_3}$$

where for all $i \in [c_1], j \in [c_2]$, $Y_{q_1+2i-1} = Y_{q_1+2i}$, and $V_{q_1+q_2+2j-1} = V_{q_1+q_2+2j}$, otherwise all Y's are pairwise distinct and V's are pairwise distinct. Therefore, we get the following equalities on φ_Y and φ_V:

1. for all $i \in [c_2]$, $\varphi_Y(q_1 + 2i - 1) = \varphi_Y(q_1 + 2i)$.
2. for all $i \in [c_3]$, $\varphi_V(q_1 + q_2 + 2i - 1) = \varphi_V(q_1 + q_2 + 2i)$.

Also, for all $i \in [c]$, $\lambda_{q_1+2i-1} \neq \lambda_{q_1+2i}$. So all the equations in $\mathcal{C}_1$ come first, followed by all the equations in $\mathcal{C}_2$, and finally all the equations in $\mathcal{C}_3$. Now, our goal is to give a lower bound on the number of solutions of $\mathcal{L}$, such that the $\widehat{Y}_i$ values are pairwise distinct and $\widehat{V}_i$ values are pairwise distinct. More formally we aim to prove the following result.

Theorem 7.4.1. *Let $\mathcal{L}$ be the system of linear equations satisfying the restrictions as above in section 7.4.1 with $q < 2^{n-2}$. Then the number of solutions $(y_1, \ldots, y_{q_Y}, v_1, \ldots, v_{q_V})$ of $\mathcal{L}$, denoted h_q, such that $y_i \neq y_j$ and $v_i \neq v_j$, for all $i \neq j$, satisfies:*

$$h_q \geq \left(1 - \frac{604q^2c}{2^{2n}} - \frac{128qc}{2^{2n}} - \frac{192q^3c}{2^{3n}} - \frac{36q^4}{2^{3n}} - \frac{204q^2}{2^{2n}} - \frac{64q^3}{2^{3n}} - \frac{56q}{2^{2n}}\right) \frac{(2^n)_{q_1+c_2+q_3}(2^n)_{q_1+q_2+c_3}}{\prod_{\lambda' \in \widehat{\lambda}^q}(2^n)_{\mu(\lambda^q,\lambda')}}.$$

The following corollary is just a simplified variant of Theorem 7.4.1 that is more suitable for applications. We will use this variant later on in our main result on CLRW2.

Corollary 7.4.2. *Let $\mathcal{L}$ be the system of linear equations satisfying the restrictions as above in section 7.4.1 with $q < 2^{n-2}$. Then the number of solutions $(y_1, \ldots, y_{q_Y}, v_1, \ldots, v_{q_V})$ of $\mathcal{L}$, denoted h_q, such that $y_i \neq y_j$ and $v_i \neq v_j$, for all $i \neq j$, satisfies:*

$$h_q \geq \left(1 - \frac{780q^2c}{2^{2n}} - \frac{100q^4}{2^{3n}} - \frac{260q^2}{2^{2n}}\right) \frac{(2^n)_{q_1+c_2+q_3}(2^n)_{q_1+q_2+c_3}}{\prod_{\lambda' \in \widehat{\lambda}^q}(2^n)_{\mu(\lambda^q,\lambda')}}.$$

We note here that the bound in Theorem 7.4.1 and Corollary 7.4.2 are parametrized in q and c. This is a bit different from the traditional mirror theory bounds. Further, we note that the bounds in Theorem 7.4.1 and Corollary 7.4.2, become $1 - O(q^4/2^{3n})$ in average case, when the expected value of c is $O(q^2/2^n)$. The proof of Theorem 7.4.1 uses an inductive approach similar to the one in [160]. We postpone the complete proof to section 7.6.

7.5 Improved Security Bound of CLRW2

Based on the tools we developed in section 7.3 and 7.4, we now show that the CLRW2 construction achieves security up to the query complexity approximately $2^{3n/4}$, given

that the underlying hash functions are AXU_3. Given Mennink's attack [132] (see section 7.2.2) in roughly these many queries we can conclude that the bound is tight.

Theorem 7.5.1. *Let $\kappa, \tau, n \in \mathbb{N}$ and $\epsilon > 0$. Let $\mathsf{E} \in \mathsf{BPerm}(\{0,1\}^\kappa, \mathbb{B})$, and let $\mathcal{H}$ be an ϵ-AXU$_3$ hash function family from $\mathbb{T}$ to $\mathbb{B}$. Consider*

$$\mathsf{CLRW2}_{\mathsf{E},\mathcal{H}} : \{0,1\}^{2\kappa} \times \mathcal{H}^2 \times \mathbb{T} \times \mathbb{B} \to \mathbb{B}.$$

For $q \leq 2^{n-2}$ and $t > 0$, the TSPRP *security of $\mathsf{CLRW2}_{\mathsf{E},\mathcal{H}}$ against $\mathbb{A}(q,t)$ is given by*

$$\mathbf{Adv}^{\mathsf{tsprp}}_{\mathsf{CLRW2}_{\mathsf{E},\mathcal{H}}}(q,t) \leq 2\mathbf{Adv}^{\mathsf{sprp}}_{\mathsf{E}}(q,t') + \Delta,$$

where $t' = O(t + qt_{\mathcal{H}})$, $t_{\mathcal{H}}$ being the time complexity for computing the hash function $\mathcal{H}$, and

$$\Delta \leq \frac{780q^4\epsilon}{2^{2n}} + \frac{100q^4}{2^{3n}} + \frac{260q^2}{2^{2n}} + 2q^2\epsilon^2 + 2q^2\epsilon^{1.5} + q^4\epsilon^3 + \frac{8q^2\epsilon}{2^{n/2}} + \frac{4q^4\epsilon^2}{2^n}. \tag{7.12}$$

On putting $\epsilon = 1/2^n$, in Eq. (7.12) and further simplifying, we get

Corollary 7.5.2. *For $q \leq 2^{n-2}$ and $\epsilon = \frac{1}{2^n}$, we have*

$$\mathbf{Adv}^{\mathsf{tsprp}}_{\mathsf{CLRW2}_{\mathsf{E},\mathcal{H}}}(q,t) \leq 2\mathbf{Adv}^{\mathsf{sprp}}_{\mathsf{E}}(q,t') + \frac{885q^4}{2^{3n}} + \frac{10q^2}{2^{3n/2}} + \frac{262q^2}{2^{2n}}. \tag{7.13}$$

Specifically, the advantage is negligible up to $q = \min(2^{\frac{3n}{4}-2.5}, 2^{n-4.5})$ queries.

PROOF OVERVIEW — The proof of Theorem 7.5.1 employs the Expectation method (see Lemma 2.2.1) coupled with an adaptation of $(3, q^4/2^{3n})$-restricted mirror theory theorem [160] in tweakable permutation settings. While our use of mirror theory is somewhat inspired by its recent use in [132], in contrast to [132], we apply the modified version of mirror theory developed in section 7.4, and that too for a restricted subset of queries. The alternating collisions lemma of section 7.3 is used to bound the probability of the most crucial bad event (Bottleneck 2 in section 7.2.3). The complete proof of Theorem 7.5.1 is given in the remainder of this section.

7.5.1 Initial Step

Consider the instantiation $\mathsf{CLRW2}_{\mathsf{E}_{\mathsf{K}_1},\mathsf{E}_{\mathsf{K}_2},\mathsf{H}_1,\mathsf{H}_2}$ of $\mathsf{CLRW2}_{\mathsf{E},\mathcal{H}}$, where K_1, K_2, H_1, H_2 are independent and $(\mathsf{K}_1, \mathsf{K}_2) \leftarrow\!\$ (\{0,1\}^\kappa)^2$, $(\mathsf{H}_1, \mathsf{H}_2) \leftarrow\!\$ \mathcal{H}^2$. As the first step, we switch to the information-theoretic setting, i.e. we replace $(\mathsf{E}_{\mathsf{K}_1}, \mathsf{E}_{\mathsf{K}_2})$ with $(\Pi_1, \Pi_2) \leftarrow\!\$ \mathsf{Perm}(\mathbb{B})^2$. For the sake of simplicity, we write the modified instantiation $\mathsf{CLRW2}_{\Pi_1,\Pi_2,\mathsf{H}_1,\mathsf{H}_2}$ as

CLRW2, i.e. without any parametrization. This switching is done via a standard hybrid argument that incurs a cost of $2\mathbf{Adv}^{\mathsf{sprp}}_{\mathsf{E}}(q, t')$ where $t' = O(t + qt_{\mathcal{H}})$. Thus, we have

$$\mathbf{Adv}^{\mathsf{tsprp}}_{\mathsf{CLRW2}_{\mathsf{E},\mathcal{H}}}(q, t) \leq 2\mathbf{Adv}^{\mathsf{sprp}}_{\mathsf{E}}(q, t') + \mathbf{Adv}^{\mathsf{tsprp}}_{\mathsf{CLRW2}}(q). \tag{7.14}$$

So, in Eq. (7.14), we have to give an upper bound on $\mathbf{Adv}^{\mathsf{tsprp}}_{\mathsf{CLRW2}}(q)$. At this point, we are in the information-theoretic setting. In other words, we consider computationally unbounded distinguisher $\mathscr{A}$. Without loss of generality, we assume that $\mathscr{A}$ is deterministic and non-trivial. Under this setup, we are now ready to apply the expectation method.

7.5.2 Oracle Description

The two oracles of interest are: $\mathcal{O}_1$, the real oracle, that implements CLRW2; and, $\mathcal{O}_0$, the ideal oracle, that implements $\widetilde{\Pi} \leftarrow\!\!\$\, \mathsf{BPerm}(\tau, n)$. We consider an extended version of these oracles, the one in which they release some additional information. We use notations analogously as given in Figure 7.2.1 to describe the transcript generated by $\mathscr{A}$'s interaction with its oracle.

7.5.2.1 Description of the Real Oracle, $\mathcal{O}_1$

The real oracle $\mathcal{O}_1$ faithfully runs CLRW2. We denote the transcript random variable generated by $\mathscr{A}$'s interaction with $\mathcal{O}_1$ by the usual notation Θ_1, which is a 10-ary q-tuple

$$(\mathsf{T}^q, \mathsf{M}^q, \mathsf{C}^q, \mathsf{X}^q, \mathsf{Y}^q, \mathsf{V}^q, \mathsf{U}^q, \lambda^q, \mathsf{H}_1, \mathsf{H}_2),$$

defined as follows: The initial transcript consists of $(\mathsf{T}^q, \mathsf{M}^q, \mathsf{C}^q)$, where for all $i \in [q]$:

T_i: i-th tweak value, M_i: i-th plaintext value, C_i: i-th ciphertext value

where $\mathsf{C}^q = \mathsf{CLRW2}(\mathsf{T}^q, \mathsf{M}^q)$. At the end of the query-response phase $\mathcal{O}_1$ releases some additional information $(\mathsf{X}^q, \mathsf{Y}^q, \mathsf{V}^q, \mathsf{U}^q, \lambda^q, \mathsf{H}_1, \mathsf{H}_2)$, where for all $i \in [q]$:

- $(\mathsf{X}_i, \mathsf{Y}_i)$: i-th input-output pair for Π_1,
- $(\mathsf{V}_i, \mathsf{U}_i)$: i-th input-output pair for Π_2,
- λ_i: i-th internal masking, $\mathsf{H}_1, \mathsf{H}_2$: the hash keys.

Note that, X^q, U^q, and λ^q are completely determined by the hash keys $\mathsf{H}_1, \mathsf{H}_2$, and the initial transcript $(\mathsf{T}^q, \mathsf{M}^q, \mathsf{C}^q)$. But, we include them anyhow to ease the analysis.

7.5.2.2 Description of the ideal oracle, $\mathcal{O}_0$

The ideal oracle $\mathcal{O}_0$ has access to $\widetilde{\Pi}$. Since $\mathcal{O}_1$ releases some additional information, $\mathcal{O}_0$ must generate these values as well. The ideal transcript random variable Θ_0 is also a 10-ary q-tuple

$$(\mathsf{T}^q, \mathsf{M}^q, \mathsf{C}^q, \mathsf{X}^q, \mathsf{Y}^q, \mathsf{V}^q, \mathsf{U}^q, \lambda^q, \mathsf{H}_1, \mathsf{H}_2),$$

defined below. Note that, we use the same notation to represent the variables of transcripts in the both world. However, the probability distributions of the these would be determined from their definitions. The initial transcript consists of $(\mathsf{T}^q, \mathsf{M}^q, \mathsf{C}^q)$, where for all $i \in [q]$:

T_i: i-th tweak value, M_i: i-th plaintext value, C_i: i-th ciphertext value,

where $\mathsf{C}^q = \widetilde{\Pi}(\mathsf{T}^q, \mathsf{M}^q)$. Once the query-response phase is over $\mathcal{O}_0$ samples $(\mathsf{H}_1, \mathsf{H}_2) \xleftarrow{\$} \mathcal{H}^2$ and computes $\mathsf{X}^q, \mathsf{U}^q, \lambda^q$, where for all $i \in [q]$:

$$\mathsf{X}_i := \mathsf{H}_1(\mathsf{T}_i) \oplus \mathsf{M}_i, \;\; \mathsf{U}_i := \mathsf{H}_2(\mathsf{T}_i) \oplus \mathsf{C}_i, \;\; \lambda_i := \mathsf{H}_1(\mathsf{T}_i) \oplus \mathsf{H}_2(\mathsf{T}_i).$$

This means that X^q, U^q, and λ^q are defined honestly. Given the partial transcript $\Theta_0' := (\mathsf{T}^q, \mathsf{M}^q, \mathsf{C}^q, \mathsf{X}^q, \mathsf{U}^q, \lambda^q, \mathsf{H}_1, \mathsf{H}_2)$ we wish to characterize the hash key $\mathsf{H} := (\mathsf{H}_1, \mathsf{H}_2)$ as good or bad. We write $\mathcal{H}_{\mathsf{bad}}$ for the set of bad hash keys, and $\mathcal{H}_{\mathsf{good}} = \mathcal{H}^2 \setminus \mathcal{H}_{\mathsf{bad}}$. We say that a hash key $\mathsf{H} \in \mathcal{H}_{\mathsf{bad}}$ (or H is bad) if and only if one of the following predicates is true:

1. $\mathtt{H}_1$: $\exists^* i, j \in [q]$ such that $\mathsf{X}_i = \mathsf{X}_j \wedge \mathsf{U}_i = \mathsf{U}_j$.
2. $\mathtt{H}_2$: $\exists^* i, j \in [q]$ such that $\mathsf{X}_i = \mathsf{X}_j \wedge \lambda_i = \lambda_j$.
3. $\mathtt{H}_3$: $\exists^* i, j \in [q]$ such that $\mathsf{U}_i = \mathsf{U}_j \wedge \lambda_i = \lambda_j$.
4. $\mathtt{H}_4$: $\exists^* i, j, k, \ell \in [q]$ such that $\mathsf{X}_i = \mathsf{X}_j \wedge \mathsf{U}_j = \mathsf{U}_k \wedge \mathsf{X}_k = \mathsf{X}_\ell$.
5. $\mathtt{H}_5$: $\exists^* i, j, k, \ell \in [q]$ such that $\mathsf{U}_i = \mathsf{U}_j \wedge \mathsf{X}_j = \mathsf{X}_k \wedge \mathsf{U}_k = \mathsf{U}_\ell$.
6. $\mathtt{H}_6$: $\exists^* i, j, k, \ell \in [q]$ such that $\mathsf{X}_i = \mathsf{X}_j = \mathsf{X}_k \wedge \mathsf{U}_k = \mathsf{U}_\ell$.
7. $\mathtt{H}_7$: $\exists^* i, j, k, \ell \in [q]$ such that $\mathsf{U}_i = \mathsf{U}_j = \mathsf{U}_k \wedge \mathsf{X}_k = \mathsf{X}_\ell$.

CASE 1: H IS BAD — If the hash key H is bad, then Y^q and V^q values are sampled arbitrarily.

CASE 2: H IS GOOD — To characterize the transcript corresponding to a good hash key it will be useful to study a graph, similar to the one in section 7.2, associated with

$(\mathsf{X}^q, \mathsf{U}^q)$. Specifically, we consider the random transcript graph $\mathcal{G}(\mathsf{X}^q, \mathsf{U}^q)$ arising due to $\mathsf{H} \in \mathcal{H}_{\mathsf{good}}$. Lemma 7.5.3 and Figure 7.5.1 characterizes the different types of possible components in $\mathcal{G}(\mathsf{X}^q, \mathsf{U}^q)$. Note that, type-4 and type-5 graphs are the same as configuration (A) and (D) of Figure 7.2.3, for size ≥ 4 edges. These graphs are considered as bad in [132], whereas we allow such components.

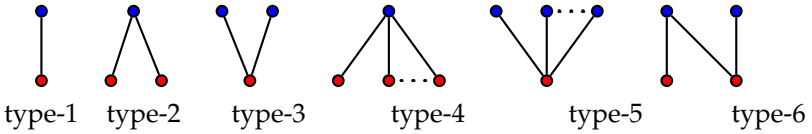

Figure 7.5.1: Enumerating all possible types of components of a transcript graph corresponding to a good hash key: type-1 is the only possible component of size $= 1$ edge; type-2 and type-3 are the only possible components of size $= 2$ edges; type-4, type-5 and type-6 are the only possible components of size $= 3$ edges; type-4 and type-5 can have size > 3 edges, and type-6 can have degree 2 vertices in both $\mathcal{X}$ and $\mathcal{U}$.

Lemma 7.5.3. *The transcript graph $\mathcal{G}$ corresponding to $(\mathsf{X}^q, \mathsf{U}^q)$ generated by a good hash key H has the following properties:*

1. *$\mathcal{G}$ is simple, acyclic and has no isolated vertices.*
2. *$\mathcal{G}$ has no two adjacent edges i and j such that $\lambda_i \oplus \lambda_j = 0$.*
3. *$\mathcal{G}$ has no subgraph $\mathcal{G}'$ of size ≥ 4 edges such that $\exists \mathsf{X} \in \mathcal{X}$ and $\exists \mathsf{U} \in \mathcal{U}$ with $\mathbf{deg}(\mathsf{X}), \mathbf{deg}(\mathsf{U}) \geq 2$ and $(\mathsf{X}, \mathsf{U}) \in \mathcal{E}(\mathcal{G}')$.*

In fact the all possible types of components of $\mathcal{G}$ are enumerated in Figure 7.5.1.

Proof. Property 1 holds by definition and the non-existence of bad hash key conditions 1, 4, and 5. Property 2 holds due to the non-existence of bad hash key conditions 2, and 3. Property 3 holds due to the non-existence of bad hash key conditions 4, 5, 6, and 7. It is easy to verify that given Property 1, 2, and 3, Figure 7.5.1 enumerates all possible types of components of $\mathcal{G}$. □

In what follows we describe the sampling of Y^q and V^q when $\mathsf{H} \in \mathcal{H}_{\mathsf{good}}$. We collect the indices $i \in [q]$ corresponding to the edges in all type-1, type-2, type-3, type-4, type-5 and type-6 components, in the index sets $\mathcal{I}_1, \mathcal{I}_2, \mathcal{I}_3, \mathcal{I}_4, \mathcal{I}_5$, and $\mathcal{I}_6$, respectively. Clearly, the six sets are disjoint, and $[q] = \mathcal{I}_1 \sqcup \mathcal{I}_2 \sqcup \mathcal{I}_3 \sqcup \mathcal{I}_4 \sqcup \mathcal{I}_5 \sqcup \mathcal{I}_6$. Let $\mathcal{I} = \mathcal{I}_1 \sqcup \mathcal{I}_2 \sqcup \mathcal{I}_3$. Consider the system of equation

$$\mathcal{L} = \{Y_i \oplus V_i = \lambda_i \ : \ i \in \mathcal{I}\},$$

where $Y_i = Y_j$ (res. $V_i = V_j$) if and only if $\mathsf{X}_i = \mathsf{X}_j$ (res. $\mathsf{U}_i = \mathsf{U}_j$) for all $i, j \in [q]$. The solution set of $\mathcal{L}$ is precisely the set

$$\mathcal{S} = \{(y^{\mathcal{I}}, v^{\mathcal{I}}) : y^{\mathcal{I}} \leftrightsquigarrow \mathsf{X}^{\mathcal{I}} \wedge v^{\mathcal{I}} \leftrightsquigarrow \mathsf{U}^{\mathcal{I}} \wedge y^{\mathcal{I}} \oplus v^{\mathcal{I}} = \lambda^{\mathcal{I}}\}.$$

Given these definitions, the ideal oracle $\mathcal{O}_0$ samples $(\mathsf{Y}^q, \mathsf{V}^q)$ as follows:

- $(\mathsf{Y}^{\mathcal{I}}, \mathsf{V}^{\mathcal{I}}) \xleftarrow{\$} \mathcal{S}$, i.e. $\mathcal{O}_0$ uniformly samples one valid assignment from the set of all valid assignments.
- Let $\mathcal{G} \setminus \mathcal{I}$ denote the subgraph of $\mathcal{G}$ after the removal of edges and vertices corresponding to $i \in \mathcal{I}$. For each component $\mathcal{C}$ of $\mathcal{G} \setminus \mathcal{I}$:
  - Let $(\mathsf{X}_i, \mathsf{U}_i)$ be the edge in $\mathcal{C}$ with the smallest index $i \in [q] \setminus \mathcal{I}$. Then $\mathsf{Y}_i \xleftarrow{\$} \mathbb{B}$ and $\mathsf{V}_i = \mathsf{Y}_i \oplus \lambda_i$.
  - For all other indices $i' \in [q] \setminus \mathcal{I}$, corresponding to edges $(\mathsf{X}_{i'}, \mathsf{U}_{i'}) \in \mathcal{C}$, $(\mathsf{Y}_{i'}, \mathsf{V}_{i'})$ is defined in the following way:
    - Suppose $i, i' \in \mathcal{I}_4$ (res. $\mathcal{I}_5$), then $\mathsf{Y}_{i'} = \mathsf{Y}_i$ (res. $\mathsf{V}_{i'} = \mathsf{V}_i$), and $\mathsf{V}_{i'} = \mathsf{Y}_i \oplus \lambda_{i'}$ (res. $\mathsf{Y}_{i'} = \mathsf{V}_i \oplus \lambda_{i'}$).
    - Suppose $i, i' \in \mathcal{I}_6$, and assume that i corresponds to the edge where $\mathsf{X}_i = \mathsf{X}_{i'}$ and $\mathsf{U}_i = \mathsf{U}_{i''}$, where i, i', i'' are the three edges in the component. Then $\mathsf{Y}_{i'} = \mathsf{Y}_i$ and $\mathsf{V}_{i'} = \mathsf{Y}_i \oplus \lambda_{i'}$. The other cases can be handled in a similar fashion.

At this point, $\Theta_0 = (\mathsf{T}^q, \mathsf{M}^q, \mathsf{C}^q, \mathsf{X}^q, \mathsf{Y}^q, \mathsf{V}^q, \mathsf{U}^q, \lambda^q, \mathsf{H}_1, \mathsf{H}_2)$ is completely defined. In this way we maintain both the consistency of equations of the form $\mathsf{Y}_i \oplus \mathsf{V}_i = \lambda_i$ (as in the case of real world), and the permutation consistency within each component, when $\mathsf{H} \in \mathcal{H}_{\mathsf{good}}$. However, there might be collisions among Y or V values from different components.

7.5.3 Definition and Analysis of Bad Transcripts

Given the description of the transcript random variable corresponding to the ideal oracle we can define the set of attainable transcripts Ω as the set of all tuples $\omega = (t^q, m^q, c^q, x^q, y^q, v^q, u^q, \lambda^q, h_1, h_2)$, where $t^q \in (\mathbb{T})^q$; $m^q, c^q, y^q, v^q \in (\mathbb{B})^q$; $(h_1, h_2) \in \mathcal{H}^2$; $x^q = h_1(t^q) \oplus m^q$; $y^q = h_2(t^q) \oplus c^q$; $\lambda^q = h_1(t^q) \oplus h_2(t^q)$; and $(t^q, m^q) \leftrightsquigarrow (t^q, c^q)$.

Our bad transcript definition is inspired by two requirements:

1. Eliminate all x^q, u^q, and λ^q tuples such that y^q and v^q are trivially restricted by way of linear dependence. For example, a cycle in the transcript graph would lead to such a restriction.
2. Eliminate all x^q, u^q, y^q, v^q tuples such that $x^q \not\leftrightsquigarrow y^q$ or $u^q \not\leftrightsquigarrow v^q$.

Among the two, requirement 2 is trivial as $x^q \leftrightsquigarrow y^q$ and $u^q \leftrightsquigarrow v^q$ is always true for real world transcript. Requirement 1 is more of a technical one that helps in the ideal world sampling of y^q and v^q.

BAD TRANSCRIPT DEFINITION: We first define certain attainable transcripts as bad depending upon the characterization of hash keys. Inspired by the ideal world description, we say that a hash key $(h_1, h_2) \in \mathcal{H}_{\mathsf{bad}}$ (or (h_1, h_2) is bad) if and only if the following predicate is true:

$$\mathtt{H}_1 \vee \mathtt{H}_2 \vee \mathtt{H}_3 \vee \mathtt{H}_4 \vee \mathtt{H}_5 \vee \mathtt{H}_6 \vee \mathtt{H}_7.$$

We say that ω is *hash induced bad* transcript, if $(h_1, h_2) \in \mathcal{H}_{\mathsf{bad}}$. We write this event as BAD-HASH, and by a slight abuse of notations,[1] we have

$$\texttt{BAD-HASH} = \bigcup_{i=1}^{7} \mathtt{H}_i. \tag{7.15}$$

This takes care of the first requirement. For the second one we have to enumerate all the conditions which might lead to $x^q \not\leftrightsquigarrow y^q$ or $u^q \not\leftrightsquigarrow v^q$. Since we sample arbitrarily when the hash key is bad, we assume that the transcript is *trivially inconsistent* in this case. For good hash keys, if $x_i = x_j$ (or $u_i = u_j$) then we always have $y_i = y_j$ (res. $v_i = v_j$); hence the inconsistency will not arise. So, given that the hash key is good, we say that ω is *sampling induced bad* transcript, if one of the following conditions is true: for some $\alpha \in [6]$ and $\beta \in [\alpha \dots 6]$, we have

- $\mathtt{Ycoll}_{\alpha\beta}$: $\exists i \in \mathcal{I}_\alpha, j \in \mathcal{I}_\beta$, such that $x_i \neq x_j \wedge y_i = y_j$, and
- $\mathtt{Vcoll}_{\alpha\beta}$: $\exists i \in \mathcal{I}_\alpha, j \in \mathcal{I}_\beta$, such that $u_i \neq u_j \wedge v_i = v_j$,

where $\mathcal{I}_i$ is defined as before in section 7.5.2. By varying α and β over all possible values, we get all 42 conditions which might lead to $x^q \not\leftrightsquigarrow y^q$ or $u^q \not\leftrightsquigarrow v^q$. Here we remark that some of these 42 conditions are never satisfied due to the sampling mechanism prescribed in section 7.5.2. These are $\mathtt{Ycoll}_{11}$, $\mathtt{Ycoll}_{12}$, $\mathtt{Ycoll}_{13}$, $\mathtt{Ycoll}_{22}$, $\mathtt{Ycoll}_{23}$, $\mathtt{Ycoll}_{33}$, $\mathtt{Vcoll}_{11}$, $\mathtt{Vcoll}_{12}$, $\mathtt{Vcoll}_{13}$, $\mathtt{Vcoll}_{22}$, $\mathtt{Vcoll}_{23}$, and $\mathtt{Vcoll}_{33}$. We listed them here only for the sake of completeness. We write the combined event that one of

[1] We use the notation $\mathtt{H}_i$ to denote the event that the predicate $\mathtt{H}_i$ is true.

the 42 conditions hold as BAD−SAMP. Again by an abuse of notations, we have

$$\text{BAD−SAMP} = \bigcup_{\alpha\in[6],\beta\in[\alpha\ldots6]} (\texttt{Ycoll}_{\alpha\beta} \cup \texttt{Vcoll}_{\alpha\beta}) \,. \tag{7.16}$$

Finally, a transcript ω is called bad, i.e. $\omega \in \Omega_{\mathsf{bad}}$, if it is either a hash or a sampling induced bad transcript. All other transcripts are called good.

BAD TRANSCRIPT ANALYSIS: We analyze the probability of realizing a bad transcript in the ideal world. By definition, this is possible if and only if one of BAD−HASH or BAD−SAMP occurs. So, we have

$$\begin{aligned}\epsilon_{\mathsf{bad}} = \Pr\left[\Theta_0 \in \Omega_{\mathsf{bad}}\right] &= \Pr_{\Theta_0}\left[\text{BAD−HASH} \cup \text{BAD−SAMP}\right] \\ &= \underbrace{\Pr_{\Theta_0}\left[\text{BAD−HASH}\right]}_{\epsilon_{\mathsf{hash}}} + \underbrace{\Pr_{\Theta_0}\left[\text{BAD−SAMP}\right]}_{\epsilon_{\mathsf{samp}}} .\end{aligned} \tag{7.17}$$

Lemma 7.5.4 upper bounds ϵ_{hash} to $2q^2\epsilon^2+2q^2\epsilon^{1.5}+q^4\epsilon^3$ and Lemma 7.5.5 upper bounds ϵ_{samp} to $8q^2\epsilon 2^{-n/2} + 4q^4\epsilon^2 2^{-n}$. Substituting these values in Eq. (7.17), we get

$$\epsilon_{\mathsf{bad}} \le 2q^2\epsilon^2 + 2q^2\epsilon^{1.5} + q^4\epsilon^3 + \frac{8q^2\epsilon}{2^{n/2}} + \frac{4q^4\epsilon^2}{2^n}. \tag{7.18}$$

Lemma 7.5.4. $\epsilon_{\mathsf{hash}} \le 2q^2\epsilon^2 + 2q^2\epsilon^{1.5} + q^4\epsilon^3$.

Proof. Using Eq. (7.15) and (7.17), we have

$$\epsilon_{\mathsf{hash}} = \Pr\left[\mathtt{H}_1 \cup \mathtt{H}_2 \cup \mathtt{H}_3 \cup \mathtt{H}_4 \cup \mathtt{H}_5 \cup \mathtt{H}_6 \cup \mathtt{H}_7\right] \le \sum_{i=1}^{7} \Pr\left[\mathtt{H}_i\right].$$

$\mathtt{H}_1$ is true if for some distinct i, j both $\mathsf{X}_i = \mathsf{X}_j$, and $\mathsf{U}_i = \mathsf{U}_j$. Now $\mathsf{T}_i = \mathsf{T}_j \implies \mathsf{M}_i \neq \mathsf{M}_j$. Hence $\mathsf{X}_i \neq \mathsf{X}_j$ and $\mathtt{H}_1$ is not true. So suppose $\mathsf{T}_i \neq \mathsf{T}_j$. Then for a fixed i, j we get an upper bound of ϵ^2 as $\mathcal{H}$ is ϵ-AXU, and we have at most $\binom{q}{2}$ pairs of i, j. Thus, $\Pr\left[\mathtt{H}_1\right] \le \binom{q}{2}\epsilon^2$. Following a similar line of argument one can bound $\Pr\left[\mathtt{H}_2\right] \le \binom{q}{2}\epsilon^2$ and $\Pr\left[\mathtt{H}_3\right] \le \binom{q}{2}\epsilon^2$.

In the remaining, we bound the probability of $\mathtt{H}_4$ and $\mathtt{H}_6$, while the probability of $\mathtt{H}_5$ and $\mathtt{H}_7$ can be bounded analogously. For any function $f : \mathbb{T} \in \mathbb{B}$, let $f' : \mathbb{T} \times \mathbb{B} \to \mathbb{B}$ be defined as $f'(t, m) = f(t) \oplus m$. So $\mathsf{X}_i = \mathsf{H}'_1(\mathsf{T}_i, \mathsf{M}_i)$, and $\mathsf{U}_i = \mathsf{H}'_2(\mathsf{T}_i, \mathsf{C}_i)$, for all $i \in [q]$. It is easy to see that H'_b is ϵ-universal if H_b is ϵ-AXU for $b \in \{0, 1\}$. Using the renewed description, $\mathtt{H}_4$ is true if for some distinct i, j, k, ℓ,

$$\mathsf{H}'_1(\mathsf{T}_i, \mathsf{M}_i) = \mathsf{H}'_1(\mathsf{T}_j, \mathsf{M}_j) \wedge \mathsf{H}'_2(\mathsf{T}_j, \mathsf{C}_j) = \mathsf{H}'_2(\mathsf{T}_k, \mathsf{C}_k) \wedge \mathsf{H}'_1(\mathsf{T}_k, \mathsf{M}_k) = \mathsf{H}'_1(\mathsf{T}_\ell, \mathsf{M}_\ell).$$

Since $(t_i, m_i) \neq (t_j, m_j)$ and $(t_i, c_i) \neq (t_j, c_j)$ for distinct i and j, we can apply the alternating collisions lemma of Lemma 7.3.1 to get $\Pr[\mathtt{H}_4] \leq q^2\epsilon^{1.5}$.

For $\mathtt{H}_6$ we use the AXU_3 property of H_1. Basically for some distinct i, j, k the probability that $\mathsf{H}'_1(\mathsf{T}_i, \mathsf{M}_i) = \mathsf{H}'_1(\mathsf{T}_j, \mathsf{M}_j) = \mathsf{H}'_1(\mathsf{T}_i, \mathsf{M}_i)$ is upper bounded by ϵ^2 (as H_1 is ϵ-AXU_3 hash). Now for each such triple, the probability that $\mathsf{H}'_2(\mathsf{T}_k, \mathsf{C}_k) = \mathsf{H}'_2(\mathsf{T}_\ell, \mathsf{C}_\ell)$ is upper bounded by ϵ for a fixed ℓ. We have at most $\binom{q}{4}$ such i, j, k, ℓ and hence $\Pr[\mathtt{H}_6] \leq \binom{q}{4}\epsilon^3$. The result follows by summing up all the individual probabilities followed by some algebraic simplifications. □

Lemma 7.5.5. $\epsilon_{\mathsf{samp}} \leq \dfrac{8q^2\epsilon}{2^{n/2}} + \dfrac{4q^4\epsilon^2}{2^n}$.

Proof. Using Eq. (7.16) and (7.17), we have

$$\begin{aligned}\epsilon_{\mathsf{samp}} &= \Pr\left[\bigcup_{\alpha\in[6],\beta\in[\alpha\ldots6]} (\mathtt{Ycoll}_{\alpha\beta} \cup \mathtt{Vcoll}_{\alpha\beta})\right] \\ &\leq \sum_{\alpha\in[6]} \sum_{\beta\in[\alpha\ldots6]} \left(\Pr[\mathtt{Ycoll}_{\alpha\beta}] + \Pr[\mathtt{Vcoll}_{\alpha\beta}]\right).\end{aligned}$$

We bound the probabilities of the events on the right hand side in groups as given below:

Bounding $\sum_{\alpha\in[3],\beta\in[\alpha\ldots3]} \Pr[\mathtt{Ycoll}_{\alpha\beta}] + \Pr[\mathtt{Vcoll}_{\alpha\beta}]$: Recall that the sampling of Y and V values is always done consistently for indices belonging to $\mathcal{I} = \mathcal{I}_1 \sqcup \mathcal{I}_2 \sqcup \mathcal{I}_3$. Hence,

$$\sum_{\alpha\in[3],\beta\in[\alpha\ldots3]} \Pr[\mathtt{Ycoll}_{\alpha\beta}] + \Pr[\mathtt{Vcoll}_{\alpha\beta}] = 0. \tag{7.19}$$

Bounding $\sum_{\beta\in\{4,5\}} \Pr[\mathtt{Ycoll}_{1\beta}] + \Pr[\mathtt{Vcoll}_{1\beta}]$: Let's consider the event $\mathtt{Ycoll}_{14}$, which translates to there exist indices $i \in \mathcal{I}_1$ and $j \in \mathcal{I}_4$ such that $\mathsf{X}_i \neq \mathsf{X}_j \wedge \mathsf{Y}_i = \mathsf{Y}_j$. Since $j \in \mathcal{I}_4$, there must exist $k, \ell \in \mathcal{I}_4 \setminus \{j\}$, such that $\mathsf{X}_j = \mathsf{X}_k = \mathsf{X}_\ell$. To bound the probability of $\mathtt{Ycoll}_{14}$, we can thus look at the joint event

$$\exists i \in \mathcal{I}_1, \exists^* j, k, \ell \in \mathcal{I}_4, \text{ such that } \mathsf{Y}_i = \mathsf{Y}_j \wedge \mathsf{X}_j = \mathsf{X}_k \wedge \mathsf{X}_k = \mathsf{X}_\ell.$$

Note that, the event $\mathsf{Y}_i = \mathsf{Y}_j$ is independent of $\mathsf{X}_j = \mathsf{X}_k \wedge \mathsf{X}_k = \mathsf{X}_\ell$, as both Y_i and Y_j are sampled independent of the hash key. Thus, we get

$$\begin{aligned}\Pr[\mathtt{Ycoll}_{14}] &= \Pr[\exists i \in \mathcal{I}_1, \exists^* j, k, \ell \in \mathcal{I}_4, \text{ such that } \mathsf{Y}_i = \mathsf{Y}_j \wedge \mathsf{X}_j = \mathsf{X}_k \wedge \mathsf{X}_k = \mathsf{X}_\ell] \\ &= \sum_{i\in\mathcal{I}_1} \sum_{j<k<\ell\in\mathcal{I}_4} \Pr[\mathsf{Y}_i = \mathsf{Y}_j] \cdot \Pr[\mathsf{X}_j = \mathsf{X}_k \wedge \mathsf{X}_k = \mathsf{X}_\ell]\end{aligned}$$

$$\leq q\binom{q}{3}\frac{\epsilon^2}{2^n},$$

where the last inequality follows from the uniform randomness of Y_j and the ϵ-AXU$_3$ property of H_1 (by viewing $\mathsf{X}_m = \mathsf{H}_m(\mathsf{T}_m) \oplus \mathsf{M}_m$ for $m \in \{j, k, \ell\}$). We can bound the probabilities of $\texttt{Ycoll}_{15}$, $\texttt{Vcoll}_{14}$, $\texttt{Vcoll}_{15}$ in a similar manner as in the case of $\texttt{Ycoll}_{14}$. So we skip the argumentation for these cases, and summarize the probability for this group as

$$\sum_{\beta\in\{4,5\}} \Pr\left[\texttt{Ycoll}_{1\beta}\right] + \Pr\left[\texttt{Vcoll}_{1\beta}\right] \leq 4q\binom{q}{3}\frac{\epsilon^2}{2^n}. \tag{7.20}$$

Bounding $\sum_{\alpha\in[6]} \Pr\left[\texttt{Ycoll}_{\alpha 6}\right] + \Pr\left[\texttt{Vcoll}_{\alpha 6}\right]$: Let's consider the event $\texttt{Ycoll}_{16}$ which translates to there exist indices $i \in \mathcal{I}_1$ and $j \in \mathcal{I}_6$ such that $\mathsf{X}_i \neq \mathsf{X}_j \wedge \mathsf{Y}_i = \mathsf{Y}_j$. Here as $j \in \mathcal{I}_6$, there must exist $k, \ell \in \mathcal{I}_6 \setminus \{j\}$ such that one of the following happens

$$\mathsf{X}_j = \mathsf{X}_k \wedge \mathsf{U}_k = \mathsf{U}_\ell \quad \mathsf{U}_j = \mathsf{U}_k \wedge \mathsf{X}_k = \mathsf{X}_\ell \quad \mathsf{X}_j = \mathsf{X}_k \wedge \mathsf{U}_j = \mathsf{U}_\ell$$

Without loss of generality we assume that the first configuration happens. So to bound the probability of $\texttt{Ycoll}_{16}$, we can look at the joint event

$$\exists i \in \mathcal{I}_1, \exists^* j, k, \ell \in \mathcal{I}_6, \text{ such that } \mathsf{Y}_i = \mathsf{Y}_j \wedge \mathsf{X}_j = \mathsf{X}_k \wedge \mathsf{U}_k = \mathsf{U}_\ell.$$

In this case the three events $\mathsf{Y}_i = \mathsf{Y}_j$, $\mathsf{X}_j = \mathsf{X}_k$, and $\mathsf{U}_k = \mathsf{U}_\ell$ are independent as Y_j, H_1, and H_2 are all independent. Thus, we get

$$\begin{aligned}
\Pr\left[\texttt{Ycoll}_{16}\right] &= \Pr\left[\exists i \in \mathcal{I}_1, \exists^* j, k, \ell \in \mathcal{I}_6, \text{ such that } \mathsf{Y}_i = \mathsf{Y}_j \wedge \mathsf{X}_j = \mathsf{X}_k \wedge \mathsf{U}_k = \mathsf{U}_\ell\right] \\
&= \sum_{i\in\mathcal{I}_1} \sum_{j<k<\ell\in\mathcal{I}_6} \Pr\left[\mathsf{Y}_i = \mathsf{Y}_j\right] \cdot \Pr\left[\mathsf{X}_j = \mathsf{X}_k\right] \cdot \Pr\left[\mathsf{U}_k = \mathsf{U}_\ell\right] \\
&\leq q\binom{q}{3}\frac{\epsilon^2}{2^n},
\end{aligned}$$

where the last inequality follows from the uniform randomness of Y_j and the ϵ-AXU$_2$ property of H_1 and H_2. The probabilities of all the remaining events in this group can be bounded in a similar fashion. We emphasize here that, in case of $\texttt{Ycoll}_{66}$ and $\texttt{Vcoll}_{66}$, it is of note that both i and j cannot lie in the same component due to the sampling mechanism. In summary, we have

$$\sum_{\alpha\in[6]} \Pr\left[\texttt{Ycoll}_{\alpha 6}\right] + \Pr\left[\texttt{Vcoll}_{\alpha 6}\right] \leq 12q\binom{q}{3}\frac{\epsilon^2}{2^n}. \tag{7.21}$$

Bounding $\sum_{\alpha\in\{2,4\}} \Pr[\texttt{Ycoll}_{\alpha 4}] + \Pr[\texttt{Vcoll}_{\alpha 4}]$: In this case we consider the probability of event $\texttt{Ycoll}_{24}$, while the probabilities of all other events can be similarly upper bounded. Now $\texttt{Ycoll}_{24}$ translates to there exist $i \in \mathcal{I}_2$ and $j \in \mathcal{I}_4$ such that $\mathsf{X}_i \neq \mathsf{X}_j \wedge \mathsf{Y}_i = \mathsf{Y}_j$. Since $i \in \mathcal{I}_2$ and $j \in \mathcal{I}_4$, there must exist $k \in \mathcal{I}_2 \setminus \{i\}$ and $\ell \in \mathcal{I}_4 \setminus \{j\}$ such that $\mathsf{X}_i = \mathsf{X}_k \wedge \mathsf{X}_j = \mathsf{X}_\ell$. To bound $\texttt{Ycoll}_{24}$, we look at the joint event

$$\exists^* i, k \in \mathcal{I}_2, \exists^* j, \ell \in \mathcal{I}_4, \text{ such that } \mathsf{X}_k = \mathsf{X}_i \wedge \mathsf{Y}_i = \mathsf{Y}_j \wedge \mathsf{X}_j = \mathsf{X}_\ell.$$

Here, the event $\mathsf{Y}_i = \mathsf{j}$ is independent of $\mathsf{X}_k = \mathsf{X}_i$ and $\mathsf{X}_j = \mathsf{X}_\ell$ (Y_i and Y_j are sampled independently of H_1). So we can apply the alternating events lemma, which gives

$$\begin{aligned}\Pr[\texttt{Ycoll}_{24}] &= \Pr[\exists^* i, k \in \mathcal{I}_2, \exists^* j, \ell \in \mathcal{I}_4, \text{ such that } \mathsf{X}_k = \mathsf{X}_i \wedge \mathsf{Y}_i = \mathsf{Y}_j \wedge \mathsf{X}_j = \mathsf{X}_\ell] \\ &\leq q^2 \frac{\epsilon}{2^{n/2}},\end{aligned}$$

where the inequality follows from Lemma 7.3.2, and the fact $|\mathcal{I}_2|, |\mathcal{I}_4| \leq q$. The probability of this group is summarized below

$$\sum_{\alpha\in\{2,4\}} \Pr[\texttt{Ycoll}_{\alpha 4}] + \Pr[\texttt{Vcoll}_{\alpha 4}] \leq 4q^2 \frac{\epsilon}{2^{n/2}}. \tag{7.22}$$

Bounding $\sum_{\alpha\in\{3,5\}} \Pr[\texttt{Ycoll}_{\alpha 5}] + \Pr[\texttt{Vcoll}_{\alpha 5}]$: This is analogous to the previous case, and can be bounded as below

$$\sum_{\alpha\in\{3,5\}} \Pr[\texttt{Ycoll}_{\alpha 5}] + \Pr[\texttt{Vcoll}_{\alpha 5}] \leq 4q^2 \frac{\epsilon}{2^{n/2}}. \tag{7.23}$$

Bounding $\sum_{(\alpha,\beta)\in\{(2,5),(3,4),(4,5)\}} \Pr[\texttt{Ycoll}_{\alpha\beta}] + \Pr[\texttt{Vcoll}_{\alpha\beta}]$: We consider the probability of the event $\texttt{Ycoll}_{25}$ which translates to there exist $i \in \mathcal{I}_2$ and $j \in \mathcal{I}_5$, such that $\mathsf{X}_i \neq \mathsf{X}_j \wedge \mathsf{Y}_i = \mathsf{Y}_j$. As $i \in \mathcal{I}_2$ and $j \in \mathcal{I}_5$, there must exist $k \in \mathcal{I}_2 \setminus \{i\}$ and $\ell \in \mathcal{I}_5 \setminus \{j\}$, such that $\mathsf{X}_i = \mathsf{X}_k \wedge \mathsf{U}_j = \mathsf{U}_\ell$. We consider the joint event

$$\exists^* i, k \in \mathcal{I}_2, j, \ell \in \mathcal{I}_5, \text{ such that } \mathsf{X}_k = \mathsf{X}_i \wedge \mathsf{Y}_i = \mathsf{Y}_j \wedge \mathsf{U}_j = \mathsf{U}_\ell.$$

Using a similar line of argument as in the case of group 3, we get

$$\Pr[\texttt{Ycoll}_{25}] \leq q \binom{q}{3} \frac{\epsilon^2}{2^n}.$$

And finally, we have

$$\sum_{(\alpha,\beta)\in\{(2,5),(3,4),(4,5)\}} \Pr[\texttt{Ycoll}_{\alpha\beta}] + \Pr[\texttt{Vcoll}_{\alpha\beta}] \leq 6q \binom{q}{3} \frac{\epsilon^2}{2^n}. \tag{7.24}$$

The result follows by combining Eq. (7.19-7.24), followed by some algebraic simplifications. □

7.5.4 Good Transcript Analysis

From section 7.5.2, we know the types of components present in the transcript graph corresponding to a good transcript ω are exactly as in Figure 7.5.1. Let

$$\omega = (t^q, m^q, c^q, x^q, y^q, v^q, u^q, \lambda^q, h_1, h_2)$$

be the good transcript at hand. From the bad transcript description of section 7.5.3, we know that for a good transcript $(t^q, m^q) \leftrightsquigarrow (t^q, c^q)$, $x^q \leftrightsquigarrow y^q$, $v^q \leftrightsquigarrow u^q$, and $y^q \oplus v^q = \lambda^q$.

We add some new parameters with respect to ω to aid our analysis of good transcripts. For $i \in [6]$, let $c_i(\omega)$ and $q_i(\omega)$ denote the number of components and number of indices (corresponding to the edges), respectively of type-i in ω. Note that, $q_1(\omega) = c_1(\omega)$, $q_i(\omega) = 2c_i(\omega)$, for $i \in \{2, 3\}$, and $q_6(\omega) = 3c_6(\omega)$. Obviously, for a good transcript $q = \sum_{i=1}^{6} q_i(\omega)$. Let $c(\omega) = c_2(\omega) + c_3(\omega)$. For all these parameters, we will drop the ω parametrization whenever it is understood from the context.

INTERPOLATION PROBABILITY FOR THE REAL ORACLE: In the real oracle, $\mathsf{H}^2 \xleftarrow{\$} \mathcal{H}^2$, Π_1 is called exactly $q_1 + c_2 + q_3 + c_4 + q_5 + 2c_6$ times and Π_2 is called exactly $q_1 + q_2 + c_3 + q_4 + c_5 + 2c_6$ times. Thus, we have

$$\Pr\left[\Theta_1 = \omega\right] = \frac{1}{|\mathcal{H}|^2} \cdot \frac{1}{(2^n)_{q_1+c_2+q_3+c_4+q_5+2c_6}} \cdot \frac{1}{(2^n)_{q_1+q_2+c_3+q_4+c_5+2c_6}}. \tag{7.25}$$

INTERPOLATION PROBABILITY FOR THE IDEAL ORACLE: In the ideal oracle, the sampling is done in parts:

I. $\widetilde{\Pi}$ *sampling*: Let $(t'_1, t'_2, \cdots, t'_r)$ denote the tuple of distinct tweaks in t^q, and for all $i \in [r]$, let $a_i = \mu(t^q, t'_i)$, i.e. $r \leq q$ and $\sum_{i=1}^{r} a_i = q$. Then, we have

$$\Pr\left[\widetilde{\Pi}(t^q, m^q) = c^q\right] \leq \frac{1}{\prod_{i=1}^{r} (2^n)_{a_i}}.$$

II. *Hash key sampling*: The hash keys are sampled uniformly from $\mathcal{H}^2$, whence

$$\Pr\left[(\mathsf{H}_1, \mathsf{H}_2) = (h_1, h_2)\right] = \frac{1}{|\mathcal{H}|^2}.$$

III. *Internal variables sampling*: The internal variables Y^q and V^q are sampled in two stages.

(A). *type*-1, *type*-2 *and type*-3 *sampling*: Recall the sets $\mathcal{I}_1$, $\mathcal{I}_2$, and $\mathcal{I}_3$, from section 7.5.3. Consider the system of equation

$$\mathcal{L} = \{\mathsf{Y}_i \oplus \mathsf{V}_i = \lambda_i \ : \ i \in \mathcal{I}\}.$$

Let $(\lambda'_1, \lambda'_2, \cdots, \lambda'_s)$ denote the tuple of distinct elements in $\lambda^{\mathcal{I}}$, and for all $i \in [s]$, let $b_i = \mu(\lambda^{\mathcal{I}}, \lambda'_i)$. From Figure 7.5.1 we know that $\mathcal{L}$ is cycle-free, non-degenerate and has $\xi_{\max} \leq 3$. So, we can apply Corollary 7.4.2 to get a lower bound on the the number of valid solutions, $|S(\mathcal{L})|$ for $\mathcal{L}$. Using the fact that $(\mathsf{Y}^{\mathcal{I}}, \mathsf{V}^{\mathcal{I}}) \leftarrow_{\$} S(\mathcal{L})$, and Corollary 7.4.2, we have

$$\Pr\left[(\mathsf{Y}^{\mathcal{I}}, \mathsf{V}^{\mathcal{I}}) = (y^{\mathcal{I}}, v^{\mathcal{I}})\right] \leq \frac{\prod_{i=1}^{s}(2^n)_{b_i}}{\zeta(\omega)(2^n)_{q_1+c_2+q_3}(2^n)_{q_1+q_2+c_3}},$$

where

$$\zeta(\omega) = \left(1 - \frac{780q^2c(\omega)}{2^{2n}} - \frac{100q^4}{2^{3n}} - \frac{260q^2}{2^{2n}}\right),$$

(B). *type*-4, *type*-5 *and type*-6 *sampling*: For the remaining indices, one value is sampled uniformly for each of the components, i.e. we have

$$\Pr\left[\left(\mathsf{Y}^{[q]\setminus\mathcal{I}}, \mathsf{V}^{[q]\setminus\mathcal{I}}\right) = \left(y^{[q]\setminus\mathcal{I}}, v^{[q]\setminus\mathcal{I}}\right)\right] = \frac{1}{(2^n)^{c_4+c_5+c_6}}.$$

By combining I, II, III, and rearranging the terms, we have

$$\Pr\left[\Theta_0 = \omega\right] \leq \frac{1}{|\mathcal{H}|^2} \cdot \frac{1}{\zeta(\omega)} \cdot \frac{\prod_{i=1}^{s}(2^n)_{b_i}}{\prod_{i=1}^{r}(2^n)_{a_i}(2^n)_{p_1}(2^n)_{p_2}(2^n)^{c_4+c_5+c_6}}, \tag{7.26}$$

where $p_1 = q_1 + c_2 + q_3$, and $p_2 = q_1 + q_2 + c_3$.

7.5.5 Ratio of Interpolation Probabilities

First, we give give two inequality results in Propositions 7.5.7 and 7.5.8, which will be used to compute the ratio of interpolation probabilities. The proofs of these results are postponed to the end of this section.

Definition 7.5.6. For $r \geq s$, let $a = (a_i)_{i\in[r]}$ and $b = (b_j)_{j\in[s]}$ be two sequences over $\mathbb{N}$. We say that a *compresses to* b, if there exists a partition $\mathcal{P}$ of $[r]$ such that $\mathcal{P}$ contains exactly s many cells, say $\mathcal{P}_1, \ldots, \mathcal{P}_s$, and $\forall i \in [s]$, $\ b_i = \sum_{j\in\mathcal{P}_i} a_j$.

Proposition 7.5.7 (Fact 1 in [108]). *For $r \geq s$, let $a = (a_i)_{i\in[r]}$ and $b = (b_j)_{j\in[s]}$ be sequences over $\mathbb{N}$, such that a compresses to b. Then for any $n \in \mathbb{N}$, such that $2^n \geq \sum_{i=1}^{r} a_i$, we have $\prod_{i=1}^{r}(2^n)_{a_i} \geq \prod_{j=1}^{s}(2^n)_{b_j}$.*

We remark here that Fact 1 in [108] is a variant of Proposition 7.5.7, which is in fact false. However, the proof of Lemma 3 in [108] implicitly used Proposition 7.5.7, and hence stands correct.

Proposition 7.5.8. *For $r \geq 2$, let $c = (c_i)_{i\in[r]}$ and $d = (d_i)_{i\in[r]}$ be two sequences over $\mathbb{N}$. Let $a_1, a_2, b_1, b_2 \in \mathbb{N}$, such that $c_i \leq a_j$, $c_i + d_i \leq a_j + b_j$ for all $i \in [r]$ and $j \in [2]$, and $\sum_{i=1}^{r} d_i = b_1 + b_2$. Then, for any $n \in \mathbb{N}$, such that $a_j + b_j \leq 2^n$ for $j \in [2]$, we have $\prod_{i=1}^{r}(2^n - c_i)_{d_i} \geq (2^n - a_1)_{b_1}(2^n - a_2)_{b_2}$.*

Proposition 7.5.8 is quite intuitive, in the sense, that the starting value in each of the falling factorial term on the left is at least as much as the starting values on the right, and the total number of terms are same on both the sides.

Now, on dividing Eq. (7.25) by Eq. (7.26), and simplifying the expression, we get

$$\begin{aligned}
\frac{\Pr\left[\Theta_1 = \omega\right]}{\Pr\left[\Theta_0 = \omega\right]} &\overset{1}{\geq} \zeta(\omega) \cdot \frac{\prod_{i=1}^{r}(2^n)_{a_i}}{\prod_{i=1}^{s}(2^n)_{b_i}(2^n - p_1 - c_4 - c_6)_{q_5+c_6}(2^n - p_2 - c_5)_{q_4+2c_6}} \\
&\overset{2}{\geq} \zeta(\omega) \cdot \frac{\prod_{i=1}^{r}(2^n)_{d_i}\prod_{i=1}^{r}(2^n - d_i)_{a_i-d_i}}{\prod_{i=1}^{s}(2^n)_{b_i}(2^n - p_1 - c_4 - c_6)_{q_5+c_6}(2^n - p_2 - c_5)_{q_4+2c_6}} \\
&\overset{3}{\geq} \zeta(\omega) \cdot \frac{\prod_{i=1}^{r}(2^n - d_i)_{a_i-d_i}}{(2^n - p_1 - c_4 - c_6)_{q_5+c_6}(2^n - p_2 - c_5)_{q_4+2c_6}} \Bigg\} A \\
&\overset{4}{\geq} \zeta(\omega).
\end{aligned} \tag{7.27}$$

From inequality 1 to 2, we rewrite the numerator such that $d_i = \mu(t^{\mathcal{I}}, t'_i)$ for $i \in [r]$. At inequality 2, $r \geq s$, as number of distinct internal masking values is at most the number of distinct tweaks, and $\widehat{t^{\mathcal{I}}}$ compresses to $\widehat{\lambda^{\mathcal{I}}}$. So using Proposition 7.5.7, we can justify the transition from 2 to 3. At inequality 3, for $i \in \{2, 3, 4, 5, 6\}$, $c_i(\omega) > 0$ if and only if $r \geq 2$. Also, $d_i \leq c_1 + c_2 + c_3 \leq p_1 + c_4 + c_6$ and $d_i \leq p_2 + c_5$ for $i \in [r]$. Similarly, $a_i \leq c_1 + c_2 + c_3 + c_4 + c_5 + 2c_6 \leq p_1 + c_4 + q_5 + 2c_6$, and $a_i \leq p_2 + q_4 + c_5 + 2c_6$. Thus, A satisfies the conditions given in Proposition 7.5.8, and hence $A \geq 1$. This justifies the transition from 2 to 3.

We define $\epsilon_{\mathsf{ratio}} : \Omega \to [0, \infty)$ by the mapping

$$\epsilon_{\mathsf{ratio}}(\omega) = 1 - \zeta(\omega).$$

Clearly $\epsilon_{\mathsf{ratio}}$ is non-negative and the ratio of real to ideal interpolation probabilities is at least $1 - \epsilon_{\mathsf{ratio}}(\omega)$ (using Eq. (7.27)). Let C, C_2, and C_3 denote the random variables $c(\Theta_0)$, $c_2(\Theta_0)$, and $c_3(\Theta_0)$. Thus, we can use Lemma 2.2.1 to get

$$\mathbf{Adv}^{\mathsf{tsprp}}_{\mathsf{CLRW2}}(q) \leq \frac{780q^2}{2^{2n}}\mathsf{Ex}\left[\mathsf{C}\right] + \frac{100q^4}{2^{3n}} + \frac{260q^2}{2^{2n}} + \epsilon_{\mathsf{bad}}$$

$$\leq \frac{780q^4\epsilon}{2^{2n}} + \frac{100q^4}{2^{3n}} + \frac{260q^2}{2^{2n}} + \epsilon_{\mathsf{bad}}, \tag{7.28}$$

where the second inequality follows from the fact that $\mathsf{Ex}\,[\mathsf{C}] \leq q^2\epsilon$. This is due to the fact that $\mathsf{C} = \mathsf{C}_2 + \mathsf{C}_3$, and for $i \in \{2,3\}$, $\mathsf{Ex}\,[\mathsf{C}_i] \leq q^2\epsilon/2$, which itself follows from the ϵ-AXU$_2$ property of $\mathcal{H}$. Theorem 7.5.1 follows from Eq. (7.14), (7.18), and (7.28). □

Proof of Proposition 7.5.7: Suppose a compresses to b due to a partition $\mathcal{P}$. Then, we call $\mathcal{P}$ the compressing partition of a and b. For $s \geq 1$, let $p(s)$ denote the claimed statement. We prove the result by induction on s. We first handle the base case, $s = 1$. In this case, we have $b_1 = \sum_{i=1}^{r} a_i$. Thus, $a_i \leq b_1$ for all $i \in [r]$. Now, a term by term comparison gives

$$\prod_{i=1}^{r} (2^n)_{a_i} \geq (2^n)_{b_1},$$

which shows that the base case $p(1)$ is true. Suppose $p(s)$ is true for all $s = n$, for some $n > 1$. We now show that $p(n+1)$ is true.

Let $a = (a_i)_{i\in[r]}$ and $b = (b_j)_{j\in[s+1]}$ be two sequences over $\mathbb{N}$, such that $r \geq s+1$ and a compresses to b. Suppose $\mathcal{P}$ is a compressing partition of a and b. Consider the sequences $a' = (a_i)_{i\in\mathcal{P}_{s+1}}$ and $b' = (b_{s+1})$. We have $|\mathcal{P}_{s+1}| \geq 1$, and $b_{s+1} = \sum_{i\in\mathcal{P}_{s+1}} a_i$, which means a' compresses to b'. Further, $2^n \geq \sum_{i\in\mathcal{P}_{s+1}} a_i$. Thus, we can apply $p(1)$ result on a' and b' to get

$$\prod_{i\in\mathcal{P}_{s+1}} (2^n)_{a_i} \geq (2^n)_{b_{s+1}}. \tag{7.29}$$

For the remaining, let $a'' = (a_i)_{i\in[r]\setminus\mathcal{P}_{s+1}}$ and $b'' = (b_j)_{j\in[s]}$. Again, we have $r - |\mathcal{P}_{s+1}| \geq s$, and $b_i = \sum_{j\in\mathcal{P}_i} a_j$ for all $i \in [s]$. Thus, we can apply the induction hypothesis for $p(s)$ on a'' and b'' to get

$$\prod_{i\in[r]\setminus\mathcal{P}_{s+1}} (2^n)_{a_i} \geq \prod_{j\in[s]} (2^n)_{b_j}. \tag{7.30}$$

The combination of Eq. (7.29) and (7.30) shows that $p(s+1)$ is true. The result follows by induction. □

Proof of Proposition 7.5.8: For $r \geq 2$, let $p(r)$ denote the claimed statement. We prove the result by induction on r. For now, assume $p(2)$ to be true, as we handle this case later. Suppose the proposition statement, denoted $p(r)$, is true for all $r \geq 2$. We show that the statement $p(r+1)$ is true. Fix some arbitrary $n \in \mathbb{N}$.

Let $a_1, a_2, b_1, b_2, c_1, \ldots, c_{r+1}, d_1, \ldots, d_{r+1} \in \mathbb{N}$, such that $c_i \leq a_i$ and $c_i + d_i \leq a_i + b_j \leq 2^n$, for all $i \in [r+1]$ and $j \in [2]$. Let i' be the smallest index in $[r+1]$, such that $d_{i'} = \min\{d_1, \ldots, d_{r+1}\}$ (such an element exist by well ordering principle). Without loss of generality, we assume that $b_1 \geq b_2$. We compare the terms, $(2^n - c_{i'} - j + 1)$ and $(2^n - a_1 - j + 1)$, for all $j \in [d_{i'}]$. Since $c_{i'} \leq a_1$, we must have $(2^n - c_{i'} - j + 1) \geq (2^n - a_1 -$

$j+1)$, for all $j \in [d_{i'}]$. Now, we must have $d_{i'} \leq b_1$, otherwise $d_{i'} > b_1 \geq b_2$ which leads to $\sum_{i\in[r]} d_i > b_1 + b_2$. Suppose $d_{i'} < b_1$, then using $(2^n - c_{i'} - j + 1)/(2^n - a_1 - j + 1) \geq 1$, we remove all the $(2^n - c_{i'} - j + 1)$, $(2^n - a_1 - j + 1)$ terms for all $j \in [d_{i'}]$. This reduces the claimed statement to $p(r)$, which is true by hypothesis. If $d_{i'} = b_1$, then we are left with $\prod_{i\in[r+1]\setminus\{i'\}}(2^n - c_i)\cdots(2^n - c_i - d_i + 1)$ on the left, where $r \geq 2$, and $(2^n - a_2)\cdots(2^n - a_2 - b_2 + 1)$ on the right. Using a similar line of argument as above we can again reduce the claimed statement to $p(r)$, which is true by hypothesis. So $p(r+1)$ is true.

Now the base case $p(2)$ can be handled in a similar manner. In this case we assume without loss of generality that $d_1 \leq d_2$ and $b_1 \geq b_2$, where $d_1 + d_2 = b_1 + b_2$. Since $c_1 \leq a_1$, we must have $(2^n - c_1 - j + 1) \geq (2^n - a_1 - j + 1)$, for all $j \in [d_1]$. Now, we must have $d_1 \leq b_1$, otherwise $d_1 > b_1 \geq b_2$ which leads to $d_1 + d_2 > b_1 + b_2$. If $d_1 = b_1$, then after removing all the terms corresponding to (c_1, d_1) and (a_1, b_1), we have $(2^n - c_2)\cdots(2^n - c_2 - d_2 + 1)$ on the left and $(2^n - a_2)\cdots(2^n - a_2 - b_2 + 1)$, where $c_2 \leq a_2$ and $c_2 + b_2 \leq a_2 + b_2$, whence $(2^n - c_2)\cdots(2^n - c_2 - d_2 + 1) \geq (2^n - a_2)\cdots(2^n - a_2 - b_2 + 1)$. If $d_1 < b_1$, then we compare terms from $(2^n - c_2)\cdots(2^n - c_2 - d_2 + 1)$ with $(2^n - a_1 - d_1)\cdots(2^n - a_1 - b_1 + 1)(2^n - a_2)\cdots(2^n - a_2 - b_2 + 1)$. First $(2^n - c_2 - d_2 + j) \geq (2^n - a_2 - b_2 + j)$ for $j \in [b_2]$, as $c_2 + d_2 \leq a_2 + b_2$. We remove all these terms to get $(2^n - c_2)\cdots(2^n - c_2 - d_2 + b_2 + 1)$ on the left and $(2^n - a_1 - d_1)\cdots(2^n - a_1 - b_1 + 1)$ on the right, where the number of terms $d_2 - b_2 = b_1 - d_1$. Since $c_2 \leq a_1$, $(2^n - c_2 - j + 1) \geq (2^n - a_1 - d_1 - j + 1)$ for all $j \in [b_1 - d_1]$. This shows that $p(2)$ is true. □

7.6 Proof of Mirror Theory for Tweakable Permutations

The induction is defined on the number of components. Apropos to this, we consider the parameter $\mathfrak{h}_i$ for $i \in [q_1 + c]$, which denotes the number of solutions for the sub-system consisting of the first i components of $\mathcal{L}$, denoted $\mathcal{L}_{|i}$. Note that, $h_i = \mathfrak{h}_i$ for $i \in [q_1]$, and $h_q = \mathfrak{h}_{q_1+c}$. We describe the proof in two steps. First, in section 7.6.1, we bound the number of solutions for sub-system $\mathcal{C}_1$. Given the number of solutions for $\mathcal{C}_1$, we then bound the number of solutions for sub-systems $\mathcal{C}_2$ and $\mathcal{C}_3$, in section 7.6.2, which essentially gives a bound for the complete system $\mathcal{L}$.

7.6.1 Bound for Sub-System with $\xi = 2$

For all $i \in [q_1]$, denote

$$H_i = \prod_{\lambda' \in \widehat{\lambda}^i} (2^n)_{\mu(\lambda^i, \lambda')} \cdot \mathfrak{h}_i,$$

and $J_i = (2^n)_i^2$. As noted before, we want to bound $\mathfrak{h}_i$ by induction on i, i.e. we want to evaluate $\mathfrak{h}_{i+1}$ from $\mathfrak{h}_i$. Inspired by Patarin's mirror theory argument [142, 160], we will study the relation between H_i and J_i for all $i \in [q_1]$. First, we give two supplementary results in Lemma 7.6.1 and 7.6.3, which will be used later on to prove the main result.

Lemma 7.6.1. *For $i \in [q_1]$,*

$$\mathfrak{h}_{i+1} = \mathfrak{h}_i \left(2^n - 2i + \delta_{i+1}\right) + \sum_{(j,k)\in\mathcal{M}} \mathfrak{h}'_i(j,k,\lambda_{i+1}),$$

where

$$\mathcal{M} = \{(j,k) : j,k \in [i], j \neq k, \lambda_{i+1} \neq \lambda_j, \lambda_{i+1} \neq \lambda_k\},$$

and $\mathfrak{h}'_i(j,k,\lambda_{i+1})$ denotes the number of solutions of $\mathcal{L}'_{|i}(j,k,\lambda_{i+1}) := \mathcal{L}_{|i} \cup \{Y_j \oplus V_k = \lambda_{i+1}\}$, for some $j,k \in [i]$.

Proof. Let $\mathcal{S}_i$ denote the solution space of $\mathcal{L}_{|i}$, i.e. $\mathfrak{h}_i = |\mathcal{S}_i|$. For a fix $(y^i, v^i) \in \mathcal{S}_i$, we want to compute the number of (y_{i+1}, v_{i+1}) pairs such that $(y^{i+1}, v^{i+1}) \in \mathcal{S}_{i+1}$. Now, some pair $(x, x \oplus \lambda_{i+1})$ is valid if $x \neq y_j$ and $x \oplus \lambda_{i+1} \neq v_k$, for $j,k \in [i]$. This means that $x \notin \mathcal{Y} \cup \mathcal{V}$, where $\mathcal{Y} = \{y_j : j \in [i]\}$ and $\mathcal{V} = \{v_j \oplus \lambda_{i+1} : j \in [i]\}$. As all y_j values are pairwise distinct and v_j values are pairwise distinct, we must have $|\mathcal{Y}| = |\mathcal{V}| = i$. Thus, we have

$$\begin{aligned}
\mathfrak{h}_{i+1} &= \sum_{(y^i,v^i)\in\mathcal{S}_i} (2^n - |\mathcal{Y} \cup \mathcal{V}|) \\
&= \sum_{(y^i,v^i)\in\mathcal{S}_i} (2^n - |\mathcal{Y}| - |\mathcal{V}| + |\mathcal{Y} \cap \mathcal{V}|) \\
&= \mathfrak{h}_i \cdot (2^n - 2i) + \sum_{(y^i,v^i)\in\mathcal{S}_i} |\mathcal{Y} \cap \mathcal{V}| \\
&\overset{1}{=} \mathfrak{h}_i \cdot (2^n - 2i) + \sum_{(y^i,v^i)\in\mathcal{S}_i} \sum_{j,k\in[i]} \phi(j,k) \\
&\overset{2}{=} \mathfrak{h}_i \cdot (2^n - 2i) + \sum_{j,k\in[i]} \mathfrak{h}'_i(j,k,\lambda_{i+1}) \\
&\overset{3}{=} \mathfrak{h}_i \cdot (2^n - 2i) + \mathfrak{h}_i \cdot \delta_{i+1} + \sum_{(j,k)\in\mathcal{M}} \mathfrak{h}'_i(j,k,\lambda_{i+1}) \\
&= \mathfrak{h}_i \cdot (2^n - 2i + \delta_{i+1}) + \sum_{(j,k)\in\mathcal{M}} \mathfrak{h}'_i(j,k,\lambda_{i+1}),
\end{aligned} \tag{7.31}$$

where $\phi(j,k)$ is the indicator variable that takes the value of 1 when $y_j \oplus v_k = \lambda_{i+1}$, and 0 otherwise. The equality from 1 to 2 follows from the definition of $\mathfrak{h}'_i(j,k,\lambda_{i+1})$, and the equality from 2 to 3 follows from the fact that exactly δ_{i+1} many (j,k) pairs exist such that $k = j$, $\lambda_{i+1} = \lambda_j$, and $y_j \oplus v_j = \lambda_{i+1}$. For these the number of solutions is

exactly the same as $\mathfrak{h}_i$ (since $Y_j \oplus V_k = \lambda_{i+1}$ is already in $\mathcal{L}_{|i}$). The remaining valid (j,k) pairs, must have $\lambda_j, \lambda_k \neq \lambda_{i+1}$, else they contradict $\mathcal{L}$. The set of these remaining (j,k) pairs is the set $\mathcal{M}$. □

The following corollary of Lemma 7.6.1 will be quite useful. The proof is immediate from the proof of Lemma 7.6.1.

Corollary 7.6.2. *For $i \geq 1$, $(2^n - 2i)\mathfrak{h}_i \leq \mathfrak{h}_{i+1} \leq (2^n - i)\mathfrak{h}_i$.*

Lemma 7.6.3. *For all $(j,k) \in \mathcal{M}$, and for all $\beta \in \{0,1\}^n$,*

$$\mathfrak{h}'_i(j,k,\beta) \geq \frac{\mathfrak{h}_i}{2^n - i + 1} \cdot \left(1 - \frac{2(i-2)}{2^n - 2(i-2)}\right).$$

Proof. We are interested in $\mathfrak{h}'_i(j,k,\beta)$, which is the number of solutions of $\mathcal{L}'_{|i}(j,k,\beta)$, $j,k \in \mathcal{M}$. The sub-system containing j and k equations is of the form

$$Y_j \oplus V_j = \lambda_j, \quad Y_j \oplus V_k = \beta, \quad Y_k \oplus V_k = \lambda_k,$$

where once we fix $Y_j = y_j$, all other unknowns are completely determined by linearity. Thus, $\mathfrak{h}'_i(j,k,\beta)$ is at most $\mathfrak{h}_{i-1}$, where $\mathfrak{h}_{i-1}$ is the number of solutions of $\mathcal{L}_{|i-1}$, the system obtained by removing the equations $Y_j \oplus V_k = \beta$ and $Y_k \oplus V_k = \lambda_k$ from $\mathcal{L}'_{|i}(j,k,\beta)$. Now a solution among the $\mathfrak{h}_{i-1}$ solutions of $\mathcal{L}_{i-1}$ is not valid to be counted in $\mathfrak{h}'_i(j,k,\beta)$, if there exists $\ell \in [i] \setminus \{k\}$, such that $y_j \oplus v_\ell = \beta$ or $y_j \oplus v_\ell = \beta \oplus \lambda_k \oplus \lambda_\ell$. The first case leads to $V_k = V_\ell$, and the second case leads to $Y_k = Y_\ell$, where $k \neq \ell$ is obvious. Therefore, the two cases correspond to the terms $\mathfrak{h}'_{i-1}(j,\ell,\beta)$ and $\mathfrak{h}'_{i-1}(j,\ell,\beta \oplus \lambda_k \oplus \lambda_\ell)$, whence we have

$$\mathfrak{h}'_i(j,k,\beta) \geq \mathfrak{h}_{i-1} - \sum_{\ell \in [i]\setminus\{j,k\}} \mathfrak{h}'_{i-1}(j,\ell,\beta) - \sum_{\ell \in [i]\setminus\{j,k\}} \mathfrak{h}'_{i-1}(j,\ell,\beta \oplus \lambda_k \oplus \lambda_\ell)$$

Using similar line of argument as above we bound $\mathfrak{h}'_{i-1}(j,\ell,\beta) \leq \mathfrak{h}_{i-2}$ and $\mathfrak{h}'_{i-1}(j,\ell,\beta \oplus \lambda_k \oplus \lambda_\ell) \leq \mathfrak{h}_{i-2}$, where the two $\mathfrak{h}_{i-2}$ terms correspond to the number of solutions of $\mathcal{L}'_{|i-2}(j,\ell,\beta)$ and $\mathcal{L}'_{|i-2}(j,\ell,\beta \oplus \lambda_k \oplus \lambda_\ell)$. Finally, we have

$$\begin{aligned}
\mathfrak{h}'_i(j,k,\beta) &\geq \mathfrak{h}_{i-1} - \sum_{\ell \in [i]\setminus\{j,k\}} \mathfrak{h}_{i-2} - \sum_{\ell \in [i]\setminus\{j,k\}} \mathfrak{h}_{i-2} \\
&\overset{1}{\geq} \mathfrak{h}_{i-1} - 2(i-2)\mathfrak{h}_{i-2} \\
&\overset{2}{\geq} \mathfrak{h}_{i-1}\left(1 - \frac{2(i-2)}{2^n - 2(i-2)}\right) \\
&\overset{3}{\geq} \frac{\mathfrak{h}_i}{2^n - i + 1}\left(1 - \frac{2(i-2)}{2^n - 2(i-2)}\right),
\end{aligned}$$

where 1-3 follows from Corollary 7.6.2. Note that, we switch from $\mathfrak{h}_{i-2}$ to $\mathfrak{h}_{i-1}$ by reintroducing the equation $Y_\ell \oplus V_\ell = \lambda_\ell$, and from $\mathfrak{h}_{i-1}$ to $\mathfrak{h}_i$ by reintroducing the equation $Y_k \oplus V_k = \lambda_k$. The readers may use Figure 7.6.1 to get a pictorial view of the switchings between different systems of equations. □

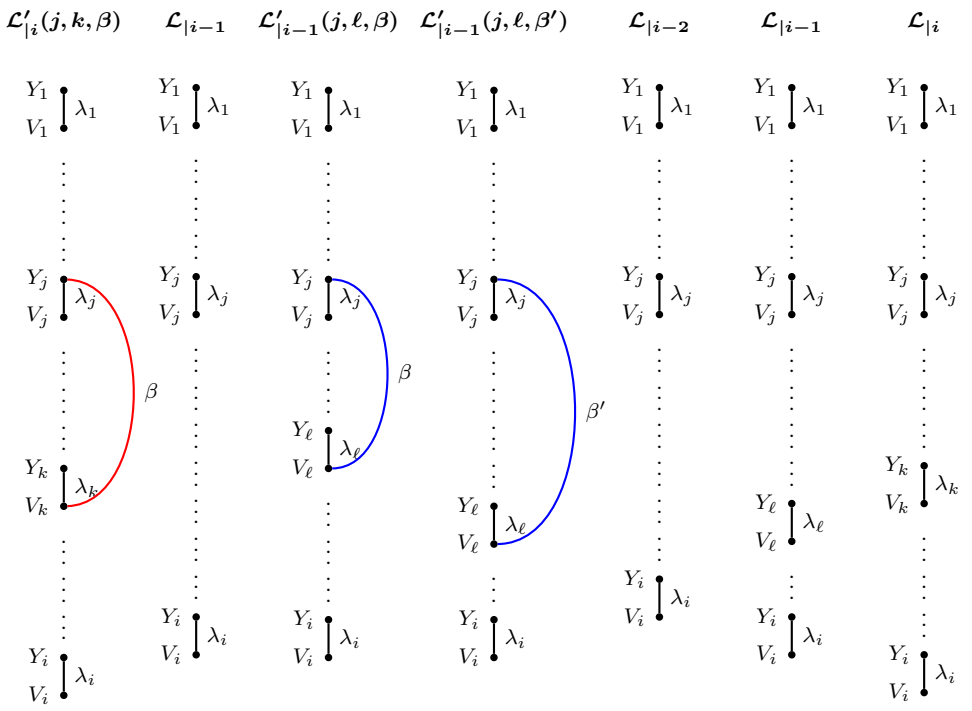

Figure 7.6.1: The switchings used in the proof of Lemma 7.6.3. From left to right: $\mathcal{L}'_{|i}(j,k,\beta)$ is the system $\mathcal{L}_{|i} \cup \{Y_j \oplus V_k = \beta\}$; $\mathcal{L}_{|i-1}$ is obtained by removing the equations involving V_k from $\mathcal{L}'_{|i}(j,k,\beta)$; $\mathcal{L}'_{|i-1}(j,\ell,\beta)$ is the system $\mathcal{L}_{|i-1} \cup \{Y_j \oplus V_\ell = \beta\}$; $\mathcal{L}'_{|i-1}(j,\ell,\beta')$ is the system $\mathcal{L}_{|i-1} \cup \{Y_j \oplus V_\ell = \beta'\}$, where $\beta' = \beta \oplus \lambda_k \oplus \lambda_\ell$; $\mathcal{L}_{|i-2}$ is obtained by removing the equations involving V_ℓ from $\mathcal{L}'_{|i-1}(j,\ell,\beta)$ or $\mathcal{L}'_{|i-1}(j,\ell,\beta')$;. Note that, there should have been two $\mathcal{L}_{|i-2}$ switchings, one each for $\mathcal{L}'_{|i-1}(j,\ell,\beta)$ and $\mathcal{L}'_{|i-1}(j,\ell,\beta')$. We have drawn just once for economical reasons. Similar clarification applies to switchings from $\mathcal{L}_{|i-2}$ to $\mathcal{L}_{|i-1}$, and from $\mathcal{L}_{|i-1}$ to $\mathcal{L}_{|i}$.

Remark 7.6.4. In [160, Theorem 11] a result similar to Lemma 7.6.3 has been proved for random function scenario. While the proof of that theorem is correct, there is a notational issue which is worth pointing out. The $\mathfrak{h}'$ notation is used in an unparameterized fashion, with an explicit hint in [160, Theorem 8] that this is done for simplification. But this simplification leads to a rather peculiar technical issue in [160, Theorem 11], where both lower and upper bounds are required on $\mathfrak{h}'$ values, requiring different switchings. Without the parametrization it is difficult to understand (and verify) the switchings.

Remark 7.6.5. The proof of Lemma 7.6.3 should also give an idea of the proof complexity. Since we only want $\epsilon = O(q^4/2^{3n})$, we needed a somewhat crude estimate of $\mathfrak{h}'$ values.

In actual mirror theory as we move towards $\epsilon = O(q/2^n)$, we have to make a good estimate of $\mathfrak{h}'$ values, which does not seem to be easy.

Now, we state the main result of this section.

Lemma 7.6.6. *For $q_1 < 2^{n-2}$, we have*

$$\frac{H_{q_1}}{J_{q_1}} \geq \left(1 - \frac{4q_1^2}{2^{2n}} - \frac{36q_1^4}{2^{3n}}\right).$$

Proof. We prove by induction on $i \in [q_1]$, the number of components. First, $H_1 = 2^2 n = J_1$. So the statement is true for $i = 1$. By definition, the ratio $\frac{H_{i+1}}{H_i} = (2^n - \delta_{i+1}) \cdot \frac{\mathfrak{h}_{i+1}}{\mathfrak{h}_i}$, and $J_{i+1} = (2^n - i)^2 J_i$. So we have

$$\frac{H_{i+1}}{J_{i+1}} = \frac{(2^n - \delta_{i+1})\frac{\mathfrak{h}_{i+1}}{\mathfrak{h}_i}}{(2^n - i)^2} \frac{H_i}{J_i}. \tag{7.32}$$

From Lemma 7.6.1 and 7.6.3, we have

$$\mathfrak{h}_{i+1} \geq \mathfrak{h}_i \left((2^n - 2i + \delta_{i+1}) + \frac{|\mathcal{M}|}{2^n - i + 1}\left(1 - \frac{2(i-2)}{2^n - 2(i-2)}\right)\right). \tag{7.33}$$

Recall that $\mathcal{M} = \{(j,k) : j,k \in [i], j \neq k, \lambda_j, \lambda_k \neq \lambda_{i+1}\}$. As there are δ_{i+1} many $i' \in [i]$ such that $\lambda_{i+1} = \lambda_{i'}$, we must have $|\mathcal{M}| \geq (i - \delta_{i+1})(i - \delta_{i+1} - 1)$. On substituting this value for $|\mathcal{M}|$ in Eq. (7.33), and using the resulting lower bound for $\mathfrak{h}_{i+1}$ in Eq. (7.32), we get

$$\begin{aligned}
\frac{H_{i+1}}{J_{i+1}} &\overset{1}{\geq} \frac{(2^n - \delta_{i+1})\left((2^n - 2i + \delta_{i+1}) + \frac{(i-\delta_{i+1})(i-\delta_{i+1}-1)}{2^n - i + 1}\left(1 - \frac{2(i-2)}{2^n - 2(i-2)}\right)\right)}{(2^n - i)^2} \frac{H_i}{J_i} \\
&\overset{2}{\geq} \frac{(2^n - i)^2 - i - \frac{\delta_{i+1}(i-\delta_{i+1})(i-\delta_{i+1}-1)}{2^n} - \frac{2(i-2)(i-\delta_{i+1})(i-\delta_{i+1}-1)}{2^n - 2(i-2)}}{(2^n - i)^2} \frac{H_i}{J_i} \\
&\overset{3}{\geq} \left(1 - \frac{4i}{2^{2n}} - \frac{36i^3}{2^{3n}}\right) \frac{H_i}{J_i} \\
&\overset{4}{\geq} \left(1 - \frac{4i}{2^{2n}} - \frac{36i^3}{2^{3n}}\right)^i \frac{H_1}{J_1} \\
&\overset{5}{\geq} \left(1 - \frac{4i^2}{2^{2n}} - \frac{36i^4}{2^{3n}}\right).
\end{aligned} \tag{7.34}$$

Here 1 to 2 is just a simplification of the expression; from 2 to 3 we use $\delta_{i+1} \leq i$, and $i \leq 2^{n-2}$; and from 3 to 4 we recursively apply the induction hypothesis. The result follows by induction. □

7.6.2 Bound for Sub-System with $\xi = 3$

At this point, we have the bound for the sub-system consisting the first q_1 many components, and we want to extend it to get the bound on $\mathfrak{h}_{q_1+c}$. We first extend to get $\mathfrak{h}_{q_1+c_2}$, and then further extend to get $\mathfrak{h}_{q_1+c}$. Before moving forward, we add some more notations. For $i \in [c_2]$, let $i_- = q_1 + 2i - 1$, $i_+ = i_- + 1$, $i' = q_1 + i$. Now we give a natural modification in the definition of $H_{i'}$ and $J_{i'}$ to accommodate for two equations per component. For all $i \in [c_2]$, denote

$$H_{i'} = \prod_{\lambda' \in \widehat{\lambda}^{i_+}} (2^n)_{\mu(\lambda^{i_+}, \lambda')} \cdot \mathfrak{h}_{i'},$$

and $J_{i'} = (2^n)_{q_1+i}(2^n)_{q_1+2i}$. Then, we have the following relations for $i \in \{0, \ldots, c_2\}$:

$$H_{q_1+i+1} = (2^n - \delta_{(i+1)_-})(2^n - \delta_{(i+1)_+}) \frac{\mathfrak{h}_{q_1+i+1}}{\mathfrak{h}_{q_1+i}} H_{q_1+i} \tag{7.35}$$

$$J_{q_1+i+1} = (2^n - q_1 - i)(2^n - q_1 - 2i)(2^n - q_1 - 2i - 1) J_{q_1+i}. \tag{7.36}$$

Our approach will be similar to the previous case, and we will first establish a lower bound on $\mathfrak{h}_{i'+1}$ in terms of $\mathfrak{h}_{i'}$ in Lemma 7.6.7.

Lemma 7.6.7. *For $i \geq 0$,*

$$\mathfrak{h}_{i'+1} \geq (2^n - 3q_1 - 5i + \delta_{(i+1)_-} + \delta_{(i+1)_+}) \mathfrak{h}_{i'},$$

with the convention that $\mathfrak{h}_{0'} = \mathfrak{h}_{q_1}$.

Proof. Let $\mathcal{S}_{i'}$ denote the solution space of $\mathcal{L}_{|i'}$, i.e. $\mathfrak{h}_{i'} = |\mathcal{S}_{i'}|$. For a fix $(y^{i_+}, v^{i_+}) \in \mathcal{S}_{i'}$, we want to compute the number of $(y_{(i+1)_-}, v_{(i+1)_-})$ and $(y_{(i+1)_+}, v_{(i+1)_+})$ pairs such that $(y^{(i+1)_+}, v^{(i+1)_+}) \in \mathcal{S}_{i'+1}$. Clearly the two tuples should be of the form $(x, x \oplus \lambda_-)$ and $(x, x \oplus \lambda_+)$ for some $x \in \mathbb{B}$. Now, some x is valid if $x \neq y_j$ and $x \oplus \lambda_{i+1} \neq v_k$, for $j, k \in [i_+]$. This means that $x \notin \mathcal{Y} \cup \mathcal{V}$, where $\mathcal{Y} = \{y_j : j \in [i_+]\}$ and $\mathcal{V} = \{v_j \oplus \lambda_{(i+1)_-}, v_j \oplus \lambda_{(i+1)_+} : j \in [i_+]\}$. Now $|\mathcal{Y}| = q_1 + i$, since exactly q_1 components from sub-system $\mathcal{C}_1$, and i many components from sub-system $\mathcal{C}_2$ have been evaluated so far, each contributing 1 y value in $\mathcal{Y}$. And, $|\mathcal{V}| \leq 2q_1 + 4i$, since each component in $\mathcal{C}_1$ and $\mathcal{C}_2$ contributes at most 2 and 4 v values, respectively. Further, exactly $\delta_{(i+1)_-} + \delta_{(i+1)_+}$ many equations share λ value with one of $\lambda_{(i+1)_-}$, or $\lambda_{(i+1)_+}$, whence $|\mathcal{Y} \cap \mathcal{V}| \geq \delta_{(i+1)_-} + \delta_{(i+1)_+}$. Thus, we have

$$\mathfrak{h}_{i'+1} = \sum_{(y^{i_+}, v^{i_+}) \in \mathcal{S}_{i'}} (2^n - |\mathcal{Y} \cup \mathcal{V}|)$$

$$
\begin{aligned}
&= \sum_{(y^{i+},v^{i+})\in\mathcal{S}_{i'}} (2^n - |\mathcal{Y}| - |\mathcal{V}| + |\mathcal{Y}\cap\mathcal{V}|) \\
&\geq \sum_{(y^{i+},v^{i+})\in\mathcal{S}_{i'}} (2^n - 3q_1 - 5i + \delta_{(i+1)_-} + \delta_{(i+1)_+}) \\
&= (2^n - 3q_1 - 5i + \delta_{(i+1)_-} + \delta_{(i+1)_+}) \cdot \mathfrak{h}_{i'}.
\end{aligned}
$$

□

Now, we state the lemma that bounds the number of solutions of $\mathcal{C}_1 \cup \mathcal{C}_2$ given the bound for $\mathcal{C}_1$.

Lemma 7.6.8. *For $q' = q_1 + q_2 \leq 2^{n-2}$, we have*

$$
\frac{H_{c_2'}}{J_{c_2'}} \geq \left(1 - \frac{196{q'}^2 c_2}{2^{2n}} - \frac{40q'c_2}{2^{2n}} - \frac{24{q'}^3 c_2}{2^{3n}} - \frac{48{q'}^2}{2^{2n}} - \frac{16q'}{2^{2n}} - \frac{8{q'}^3}{2^{3n}}\right) \frac{H_{q_1}}{J_{q_1}}.
$$

Proof. We use induction on $i \in [c_2]$. In the base case for $i = 1$, it can be easily verified that we have

$$
\begin{aligned}
\frac{H_{1'}}{J_{1'}} &\geq \frac{(2^n - \delta_{1_-})(2^n - \delta_{+})(2^n - 3q_1 + \delta_{1_-} + \delta_{1_+})}{(2^n - q_1)(2^n - q_1)(2^n - q_1 - 1)} \frac{H_{q_1}}{J_{q_1}} \\
&\geq \left(1 - \frac{48q_1^2}{2^{2n}} - \frac{16q_1}{2^{2n}} - \frac{8q_1^3}{2^{3n}}\right) \frac{H_{q_1}}{J_{q_1}},
\end{aligned} \tag{7.37}
$$

where we have used the fact that $\delta_{1_-}, \delta_{1_+} \leq q_1$, and $q_1 \leq 2^{n-1}$. Note that, when $q_1 = 0$, $H_1' = 2^{3n} > 2^{2n}(2^n - 1) = J_{1'}$, by definition. For $i > 1$, on dividing Eq. (7.35) by Eq. (7.36), we have

$$
\frac{H_{i'+1}}{J_{i'+1}} = \frac{(2^n - \delta_{(i+1)_-})(2^n - \delta_{(i+1)_+})\frac{\mathfrak{h}_{i'+1}}{\mathfrak{h}_{i'}}}{(2^n - q_1 - i)(2^n - q_1 - 2i)(2^n - q_1 - 2i - 1)} \frac{H_{i'}}{J_{i'}}. \tag{7.38}
$$

On substituting the lower bound on $\mathfrak{h}_{i'+1}$ from Lemma 7.6.7 in Eq. (7.38), we get

$$
\begin{aligned}
\frac{H_{i'+1}}{J_{i'+1}} &\overset{1}{\geq} \frac{(2^n - \delta_{(i+1)_-})(2^n - \delta_{(i+1)_+})(2^n - 3q_1 - 5i + \delta_{(i+1)_-} + \delta_{(i+1)_+})}{(2^n - q_1 - i)(2^n - q_1 - 2i)(2^n - q_1 - 2i - 1)} \frac{H_{i'}}{J_{i'}} \\
&\overset{2}{\geq} \left(1 - \frac{(24{q'}^2 + 5q')2^n + 8{q'}^3}{(2^n - q_1 - i)(2^n - q_1 - 2i)(2^n - q_1 - 2i - 1)}\right) \frac{H_{i'}}{J_{i'}} \\
&\overset{3}{\geq} \left(1 - \frac{196{q'}^2}{2^{2n}} - \frac{40q'}{2^{2n}} - \frac{24{q'}^3}{2^{3n}}\right) \frac{H_{i'}}{J_{i'}} \\
&\overset{4}{\geq} \left(1 - \frac{196{q'}^2}{2^{2n}} - \frac{40q'}{2^{2n}} - \frac{24{q'}^3}{2^{3n}}\right)^i \frac{H_{1'}}{J_{1'}} \\
&\overset{5}{\geq} \left(1 - \frac{196{q'}^2 i}{2^{2n}} - \frac{40q'i}{2^{2n}} - \frac{24{q'}^3 i}{2^{3n}} - \frac{48{q'}^2}{2^{2n}} - \frac{16q'}{2^{2n}} - \frac{8{q'}^3}{2^{3n}}\right) \frac{H_{q_1}}{J_{q_1}},
\end{aligned} \tag{7.39}
$$

where, $q' = q_1 + q_2$, 1-2 follows from $\delta_{(i+1)_-}, \delta_{(i+1)_+}, i, q_1 \leq q$; 2-3 follows from $q_1 + 2i < 2^{n-1}$; 3-4 follows from recursive application of induction hypothesis; 4-5 follows from Eq. (7.37) and $q_1 \leq q$. □

At this point, we have $\mathfrak{h}_{q_1+c_2}$, and we want to extend it to $\mathfrak{h}_{q_1+c}$. The extension follows from exactly the same line of argument as above. We skip the formal analysis and just provide the final result in Lemma 7.6.9. For the sake of verification we provide some modified definitions. For $i \in [c_3]$, let $i_- = q_1 + q_2 + 2i - 1$, $i_+ = i_- + 1$, $i' = q_1 + c_1 + i$. For all $i \in [c_3]$, denote

$$H_{i'} = \prod_{\lambda' \in \widehat{\lambda}^{i_+}} (2^n)_{\mu(\lambda^{i_+}, \lambda')} \cdot \mathfrak{h}_{i'},$$

and $J_{i'} = (2^n)_{q_1+2c_2+i}(2^n)_{q_1+c_2+2i}$.

Lemma 7.6.9. *For $q = q_1 + q_2 + q_3 < 2^{n-2}$, and $c = c_2 + c_3$, we have*

$$\frac{H_{q_1+c}}{J_{q_1+c}} \geq \left(1 - \frac{604q^2c}{2^{2n}} - \frac{128qc}{2^{2n}} - \frac{192q^3c}{2^{3n}} - \frac{200q^2}{2^{2n}} - \frac{56q}{2^{2n}} - \frac{64q^3}{2^{3n}}\right) \frac{H_{q_1}}{J_{q_1}}.$$

Theorem 7.4.1 follows from the definitions of H_{q_1+c} and J_{q_1+c}, and Lemmata 7.6.6 and 7.6.9.

Part III

Analysis of Authenticated Encryptions

Chapter 8

Random Read Access in OCB

Now that we have studied the security of various message authentication and encryption schemes, we shift our focus to authenticated encryption schemes. In this chapter, we study one of the most popular AE family, called *Offset Codebook* or OCB. The first version of OCB, called OCB1, was proposed by Rogaway et al. [174] in 2001. Later, Rogaway [171] proposed a second version of OCB, called OCB2. The latest version of OCB, called OCB3, was proposed by Krovetz and Rogaway [119]. OCB3 is one of the winners in the "high performance applications" category of the recently concluded [42] competition. OCB achieves both authentication and encryption with minimal performance overhead compared to classical single-pass encryption only modes such as CTR mode [153]. In fact OCB3 [119] has, to a larger extent, already achieved the theoretical bounds for efficiency in software implementation.

Recall the random read access and out-of-sequence decryption properties (see section 1.3.3 of chapter 1). The efficiency of OCB family in random read access depends on the efficiency of the underlying mask generating function or MGF in direct computation. OCB masks the input and output of each block cipher call with the output of an MGF. So, if the MGF is efficient in direct computation[1] then OCB can achieve random read access almost as efficiently as CTR [153] based designs, most notably CCM [154] and GCM [130]. Next, we revisit some popular choices of MGFs.

8.1 Revisiting Small-Domain AXU Hash Functions

SOME MORE NOTATIONS: For $m \in \mathbb{N}$, I_m denotes the identity matrix of size m. We use $\oplus$ and $\odot$ to denote the field addition (XOR) and field multiplication, respectively,

[1] Computation over arbitrary inputs without following any particular order.

over the finite field $\mathbb{F}_{2^n} = \mathbb{B}$. We use α to denote a primitive element of $\mathbb{F}_{2^n}$, and for any $x \in \mathbb{F}_{2^n}$, the operation $x \odot \alpha$ will be referred to as the α-multiplication operation on x. For some $m \in \mathbb{N}$, for $x \in \{0,1\}^m$, $b \in [m]$, $x \gg b$ denotes x shifted b bits to the right, i.e., $0^b \| x^{[m-b]}$, while $x \ll b$ denotes x shifted b bits to the left. For a binary string x, $\mathsf{ntz}(x)$ denotes the number of trailing zero bits in x.

8.1.1 Examples of Small-Domain AXU Hash Function

Symmetric-key cryptographic literature is rich in AXU hash function candidates. In this work we concentrate only on *small-domain* hash functions that can be used to generate masks in OCB. Further, the range of these hash functions is $\mathbb{B}$. In what follows, we discuss some examples of AXU hash functions, with a summary given in Table 8.1.1.

Table 8.1.1: Feature summary of some small-domain AXU hash function candidates, where $\oplus$, $\odot$, $\ll$, $\|$, $\oplus_?$ denote addition, multiplication, shifts, concatenation and conditional addition over appropriate fields; AESRD denotes AES round function, wt and ntz denote the hamming weight and trailing zeros function.

Hash Family	Sequential (operations)	Direct (operations)	Key Size	Domain Size in $\log_2$	AXU Bound
xtimes [174]	$1 \ll, 1 \oplus_?$	$O(\log_2 i) \odot$	128	128	2^{-128}
gray [35, 119]	1 ntz(i), $1 \oplus$	$1 \oplus, 1 \ll, 1 \odot$	128	128	2^{-128}
mtrx [45, 81]	$2 \ll, 1 \oplus$ [#]	$O(\log_2 i) \odot$	128	128	2^{-128}
mlin	wt$(i) \oplus$	wt$(i) \oplus$	$128r$	r	2^{-128}
clh [137]	wt$(i) \ll$, wt$(i) \oplus$	wt$(i) \ll$, wt$(i) \oplus$	128	127	2^{-128}
sts [119]	$1 \|, 1 \oplus, 3 \ll$	$1 \|, 1 \oplus, 3 \ll$	128	6	2^{-128}
lcube	$1 \oplus, 2 \odot$	$1 \oplus, 2 \odot$	128	128	2^{-128}
linv [174]	$1 \oplus, 1 \odot$	$1 \oplus, 1 \odot$	128	128	2^{-128}
aes4 [135, 141]	4 AESRD	4 AESRD	512	128	2^{-113}
aes2 (Section 8.2)	2 AESRD	2 AESRD	256	32	2^{-113}
aes1 (Section 8.2)	1 AESRD	1 AESRD	128	8	2^{-96}

[#] Note that, [81] defines many three-operation maskings for 128 bit binary field.

Example 8.1 (α-multiplication based hash [174])**.** *For some reasonably large $\ell \in [2^n - 1]$, let $\mathcal{X} = [\ell]$ and $\mathcal{K} = \mathbb{B}$. The hash function* xtimes*, defined as $\forall L \in \mathbb{B}, i \in [\ell]$,*

$$\mathit{xtimes}_L(i) = \mathit{xtimes}(L, i) := \alpha^i \odot L, \tag{8.1}$$

is a 2^{-n}-AXU hash. xtimes *has been used to generate position dependent input masks in many symmetric-key designs including the* OCB *family [119, 171, 174],* OTR *[138],* PMAC *[35],* PMAC+*[190],* EME *[88],* COPA *[3] and* EIMD *[40]. The popularity of* xtimes *is mainly due to its efficiency in sequential computation, as we have*

$$\forall i \geq 1,\ L \in \mathbb{B}, \mathit{xtimes}_L(i+1) = \alpha \odot \mathit{xtimes}_L(i),$$

and α-multiplication is efficient (one shift and one conditional bit-wise XOR operation). While the sequential computation of $\mathit{xtimes}_L(i)$ is very efficient, the same is not reflected in direct

computation of $\mathsf{xtimes}_L(i)$ *when* i *is a large integer. In general it may require* $O(\log_2 i)$ *many field multiplications. Even when the* α^i *values are precomputed, we need one field multiplication which is not as efficient as one* α*-multiplication. Indeed we can even avoid the precomputation and simply use* $i \odot L$ *as the hash definition.*

Example 8.2 (Gray-code based hash [35, 119])**.** *For reasonably large* $\ell \in [2^n - 1]$*, let* $\mathcal{X} = [\ell]$ *and* $\mathcal{K} = \mathbb{B}$*. The hash function* **gray** *is defined as,*

$$\forall L \in \mathbb{B},\ i \in [\ell],\ \mathsf{gray}_L(i) = \mathsf{gray}(L, i) := \gamma(i) \odot L \tag{8.2}$$

where $\gamma(i) := i \oplus (i \gg 1)$ *is the gray code value for* i*. The Gray code sequence has a very nice feature that for* $i \geq 1$*, the* i*-th gray code can be written in terms of the* $(i-1)$*-th gray code, i.e*

$$\gamma(i) = \gamma(i-1) \oplus 2^{\mathsf{ntz}(i)},$$

When we substitute this in (8.2) we get an alternate definition $\forall L \in \mathbb{B},\ i \in [\ell]$*,*

$$\mathsf{gray}_L(i) = \mathsf{gray}(L, i) := \oplus_{j=1}^{i} L_{\mathsf{ntz}(j)}. \tag{8.3}$$

where $L_j = \alpha^j \odot L$ *for all* $j < n$*. In [119] and RFC7253 [120],* **gray** *has been shown to be a* 2^{-n} *AXU hash function. Note that, we have*

$$\forall i \geq 1,\ L \in \mathbb{B}, \mathsf{gray}_L(i+1) = \mathsf{gray}_L(i) \oplus L_{\mathsf{ntz}(i+1)}.$$

So if the L_j *values are precomputed, then in sequential computation* **gray** *hash only requires,* ntz *computation along with a single* n*-bit XOR operation. Hence it is very efficient in sequential computations. This hash function is applied in* **OCB3** *[119] (RFC7253 [120]) and* **PMAC** *[35].*

Example 8.3 (Matrix-powered hash [45, 81])**.** *Let* $n = wn'$ *for some positive integer* w*, called word size, whose values are generally software friendly such as* 8*,* 16*,* 32*,* 64 *etc. Let* $\mathcal{W} = \{0,1\}^w$ *and* $\mathcal{K} = \mathcal{W}^{n'}$ *and* $\mathcal{X} = [\ell]$ *for some reasonably large* $\ell \in [2^n - 1]$*. Let* M *be an invertible* $n' \times n'$ *matrix whose entries are from the field*[2] $\mathcal{W}$ *such that for all* $i \in [2^n - 1]$*,* $I_{n'} + M^i$ *is invertible. Then the hash function* **mtrx** *(called matrix-powered hash), defined as*

$$\forall L \in \mathcal{W}^{n'}, i \in [\ell],\ \mathsf{mtrx}_L(i) = \mathsf{mtrx}(L, i) := M^i \cdot L, \tag{8.4}$$

is a 2^{-n}*-AXU. Note here that* L *is viewed as a vector over* $\mathcal{W}$*. Various matrix-powered hash candidates are given in [81] and [45]. But, similar to* **xtimes** *hash, all those hash functions are efficient in sequential computation and inefficient in direct computation.*

[2] $\{0,1\}^m$ can be viewed as the binary field by fixing a $\deg m$ primitive polynomial.

The matrix-powered hash function is a very general candidate. For example, xtimes and gray hash functions can be viewed as instances of matrix-powered hash for appropriate choices of the matrix M. The readers are directed to [81] and [45] for a detailed exposition on matrix-powered hash functions. A table of various word-oriented matrices has been listed in [81].

Example 8.4. *Some other finite field based constructions, with $\mathcal{K} = \mathcal{X} = \mathbb{B}$, are the following:* $\forall L \in \mathbb{B},\ x \in \mathbb{B}$

- $\mathit{lcube}_L(x) = (L \oplus x)^3$;
- $\mathit{linv}_L(x) = (L \oplus x)^{-1}$ *(if* $x \neq L$*),* 0 *otherwise.*

It is easy to verify that lcube *is* 2^{1-n}*-AXU and* linv *is* 2^{2-n}*-AXU hash function.*

Example 8.5 (Multi-linear hash)**.** *For $\ell \in [2^n - 1]$, let $\mathcal{X} = (\ell]$ and $\mathcal{K} = \mathbb{B}^{\log_2 \ell}$. The hash function* mlin *(called multi-linear hash), is defined as*

$$\forall L \in \mathcal{K},\ b \in \mathcal{X},\ \ \mathit{mlin}_L(b) = \mathit{mlin}(L, b) := \oplus_{i=1}^{r} b_i L_i, \tag{8.5}$$

is a 2^{-n}-AXU hash function, where $L_1, \ldots, L_r \in \mathbb{B}$ and $b_1, \ldots, b_r \in \{0, 1\}$. In Grain-128a *[2], the tag is computed using* mlin*, where the keys are derived using Toeplitz matrix.*

Example 8.6 (Circulant hash [137])**.** *In [137] Minematsu presented a small-domain hash function based on data-dependent rotations. For $\ell \in [2^{n-1} - 1]$, let $\mathcal{X} = (\ell]$ and $\mathcal{K} = \mathbb{B}$. The hash function* clh *(called circulant hash) is defined as* $\forall K \in \mathcal{K},\ x \in \mathcal{X}$*,*

$$\mathit{clh}_L(x) = \mathit{clh}(L, x) := \bigoplus_{i \in [\log_2 \ell]:\ x_i = 1} (L \ll (i-1)). \tag{8.6}$$

For direct computation, the hash function might require $n - 1$ rotations and $n - 2$ XORs in the worst case; for sequential computation some optimization is possible for the i-th input given the $(i-1)$-th output. In [137], clh *is shown to be both 2^{-128}-AXU and 2^{-128}-uniform hash function.*

Example 8.7 (Stretch-then-shift hash [119])**.** *In [119] Krovetz and Rogaway presented a highly efficient AXU hash function over a small domain $\mathcal{X} = [64]$. It follows the general data-dependent rotation technique by Minematsu [137]. Fix $c \in \mathbb{N}$. For $\mathcal{X} = (\psi(c) - 1]$ and $\mathcal{K} = \mathbb{B}$, where $\psi(c)$ is a positive integer denoting the maximum domain size for a fixed c. The hash function* sts *(called stretch-then-shift hash) is defined as* $\forall L \in \mathbb{B},\ i \in [0..\psi(c) - 1]$*,*

$$\mathit{sts}_L(i) = \mathit{sts}(L, i) := \mathsf{msb}_n(\mathit{stretch}_c(L) \ll i), \tag{8.7}$$

where $\forall L \in \mathbb{B}$, $\mathsf{stretch}_c(L) := L \| (L \oplus (L \ll c))$. *The hash function first stretches the key based on a parameter* c*, and then makes data-dependent shift before outputting the most significant* n *bits; hence the name. For* $n = 128$, $c = 8$ *and* $\psi(c) = 64$, *Krovetz and Rogaway showed that* **sts** *achieves* 2^{-128}*-AXU and* 2^{-128}*-uniform bound [119].*

Table 8.1.1 clearly shows that most of the above given small-domain hash functions are indeed quite efficient when computed sequentially over the indices; but all of them are inefficient for direct computation at any arbitrary index. A possible way out is to use iterations of some cryptographic round function with good differential probability. In the next section we take this approach.

8.2 AES-based Small-Domain AXU Hash Functions

In the previous section we listed some hash functions based on finite field operations and data dependent rotations. Now we will propose some hash functions based on the differential properties of the AES rounds. Before moving forward, we give the standard and well-known [141, 157] definition for differential probability of a (keyed) permutation.

Definition 8.2.1. Let π be a permutation over $\mathbb{F}_{2^n}$ and Π_{K} be a keyed permutation over $\mathbb{F}_{2^n}$ with key $\mathsf{K} \leftarrow\!\!\$\, \mathcal{K}$. For any given non-zero $a, b \in \mathbb{F}_{2^n}$ the differential probability of π, is defined as

$$\mathsf{dp}_\pi(a, b) := \Pr_{\mathsf{X}}[\pi(X) \oplus \pi(\mathsf{X} \oplus a) = b].$$

The maximum differential probability (MDP) of π is defined as

$$\mathsf{MDP}(\pi) := \max_{a \neq 0, b} \mathsf{dp}_\pi(a, b).$$

Similarly, the expected differential probability (EDP) of Π_{K}, is defined as

$$\begin{aligned}\mathsf{EDP}_{\Pi_{\mathsf{K}}}(a, b) &:= \Pr_{\mathsf{K},\mathsf{X}}[\Pi_{\mathsf{K}}(\mathsf{X}) \oplus \Pi_{\mathsf{K}}(\mathsf{X} \oplus a) = b] \\ &= \sum_{k \in \mathcal{K}} \Pr_{\mathsf{X}}[\Pi_k(\mathsf{X}) \oplus \Pi_k(\mathsf{X} \oplus a) = b \mid \mathsf{K} = k] \times \Pr[\mathsf{K} = k] \\ &= \sum_{K \in \mathcal{K}} \mathsf{dp}_{\Pi_k}(a, b) \times \Pr[\mathsf{K} = k] = \mathsf{Ex}_{\mathsf{K}}\left[\mathsf{dp}_{\Pi_{\mathsf{K}}}(a, b)\right],\end{aligned}$$

The maximum expected differential probability (MEDP) of Π_{K} is defined as

$$\mathsf{MEDP}(\Pi_{\mathsf{K}}) := \max_{a \neq 0, b} \mathsf{EDP}_{\Pi_{\mathsf{K}}}(a, b).$$

We have also used the term differential probability in the context of AXU hash functions in Definition 2.3.1. Now that we have formally defined the differential probability of a (keyed) permutation, it is imperative that we discuss the similarity and differences between the two probabilities. The major difference is the source of randomness. In the AXU view of differential probability, the randomness is due to the hash key, whereas here one of the input is chosen randomly. Even though this seems to be a major difference, but for certain class of functions the two views have a very nice relationship. Suppose π is a (possibly keyed) permutation over $\mathbb{F}_{2^n}$ with keyspace $\mathcal{K}$ (empty set when π is keyless). We define a function $\pi' : \mathbb{F}_{2^n} \times \mathcal{K} \times \mathbb{F}_{2^n} \to \mathbb{F}_{2^n}$ as a keyed family of function indexed by $K, K' \in \mathbb{F}_{2^n} \times \mathcal{K}$, such that

$$\forall K, K', x \in \mathbb{F}_{2^n} \times \mathcal{K} \times \mathbb{F}_{2^n},\ \pi'_{K,K'}(x) := \pi_{K'}(K \oplus x).$$

We will drop K' from the subscript, whenever π is unkeyed. Lemma 8.2.2 is a well-known result that we prove here just for the sake of completeness.

Lemma 8.2.2. *For some $\epsilon \geq 0$,*

1. *If π is unkeyed and $\mathsf{MDP}(\pi) = \epsilon$ then π' is an ϵ-AXU hash for $\mathsf{K} \xleftarrow{\$} \mathbb{F}_{2^n}$.*
2. *If π is keyed and $\mathsf{MEDP}(\pi) = \epsilon$ then π' is an ϵ-AXU hash for $(\mathsf{K}, \mathsf{K}') \xleftarrow{\$} \mathbb{F}_{2^n} \times \mathcal{K}$.*

Proof. We prove the unkeyed version of the lemma. The keyed version can be proved in similar fashion. Let $x, x' \in \mathbb{F}_{2^n}$ such that $x \neq x'$ and $\delta \in \mathbb{F}_{2^n}$. If $\delta = 0$ then the result is vacuously true as π' is a permutation. Suppose $\delta \neq 0$. Then, we have

$$\begin{aligned}
\Pr_{\mathsf{K}}\left[\pi'_{\mathsf{K}}(x) \oplus \pi'_{\mathsf{K}}(x) = \delta\right] &= \Pr_{\mathsf{K}}\left[\pi(\mathsf{K} \oplus x) \oplus \pi(\mathsf{K} \oplus x') = \delta\right] \\
&= \Pr_{\mathsf{X}}[\pi(\mathsf{X}) \oplus \pi(\mathsf{X} \oplus x' \oplus x) = \delta] \\
&\leq \mathsf{dp}(\pi) = \epsilon.
\end{aligned}$$

□

8.2.1 Revisiting MEDP bounds for the AES

AES is a Substitution Permutation Network (SPN) with block size $n = 128$, internal s-box input size $b = 8$, where all s-boxes are identical, denoted by S. The permutation layer, denoted by ϕ, consists of a bytewise permutation followed by four identical 32-bit linear transformations applied in parallel. One round of AES, denoted by 1-AESRD (without the subkey mixing) is nothing but $\phi \circ S$. Using the 1-AESRD round function, we

can define the keyed function i-AESRD for all integers $i > 1$ as follows: $\forall x \in \mathbb{F}_{2^{128}}, K \in \mathbb{F}_{2^{128}}^{i-1}$,

For $i = 2$:

$$2\text{-AESRD}_K(x) = 1\text{-AESRD}(K_1 \oplus 1\text{-AESRD}(x))$$

For $i > 2$:

$$i\text{-AESRD}_K(x) = 1\text{-AESRD}(K_{i-1} \oplus (i-1)\text{-AESRD}_{K_{[i-2]}}(x))$$

The differential properties of AES is a well-studied [52, 53, 114, 115, 157] topic in symmetric-key cryptography. In fact, one of the design criteria for AES [52, 152] was protection against linear [127] and differential cryptanalysis [32].

In [53], Daemen and Rijmen first observed that due to the specific nature of the AES permutation layer ϕ, the differential probabilities over 2 AES rounds are equivalent to those over a reduced SPN structure, which they called a *super box*. Basically the two rounds of AES over 128-bit input can be viewed as four parallel invocations of the super box with independent keys (disjoint substrings of a uniform string are independent) each working with distinct 32 bits of the input. We denote an AES super box instantiated with a subkey K by $\texttt{aesSB}_K$.

Keliher and Sui [114, 115] later used this idea to derive a tight MEDP bound for 2-AESRD. Specifically, in [114, Theorem 1, Theorem 2] and [115, section 4.1] they proved that the MEDP for AES super box is 1.656×2^{-29}, which is equivalent to the MEDP for 2-AESRD. We refer the readers to [53, 115] for further exposition on AES super box and its relation with 2-AESRD. Using the fact that the upper bound on the MEDP for 4 or more rounds of AES is equal to the 4-th power of the upper bound on the MEDP for 2-AESRD, Keliher and Sui [115] further showed that MEDP for t-AESRD for $t \geq 4$ is upper bounded by 1.881×2^{-114}. We summarize these results in Proposition 8.2.3.

Proposition 8.2.3. *[115, section 4.1] Let* $\mathsf{X}^3 \xleftarrow{\$} (\mathbb{F}_{2^{128}})^3$ *and* $\mathsf{Y} \xleftarrow{\$} \mathbb{F}_{2^{32}}$*. Then,*

1. $\mathsf{MEDP}(\texttt{aesSB}_{\mathsf{Y}}) \approx 1.656 \times 2^{-29}$.
2. $\mathsf{MEDP}(2\text{-}\texttt{AESRD}_{\mathsf{X}_1}) \approx 1.656 \times 2^{-29}$.
3. $\mathsf{MEDP}(4\text{-}\texttt{AESRD}_{\mathsf{X}^3}) \leq 1.881 \times 2^{-114}$.

8.2.2 Our AES-based proposals

In the following discussion we fix $n = 128$, i.e. $\mathbb{B} = \{0,1\}^{128}$. Before presenting our proposals we start with a simple 4-AESRD based hash function in the following example.

Example 8.8. *Let $\mathcal{X} = \mathbb{B}$ and $\mathcal{K} = (\mathbb{B})^4$. For $L^4 \in (\mathbb{B})^4$ and $x \in \mathbb{B}$, the 4-AESRD based hash function aes4 is defined as*

$$\mathsf{aes4}_{L^4}(x) = \mathsf{aes4}(L^4, x) := \mathsf{4\text{-}AESRD}_{L^{[2\ldots4]}}(L_1 \oplus x).$$

From Lemma 8.2.2 and Proposition 8.2.3, it is straightforward to see that aes4 is a 1.88×2^{-114}-AXU hash function. Minematsu used such a hash function in [135]. In [141] Minematsu and Tsunoo used a variant of aes4 to construct a universal hash function and subsequently extended it to construct a MAC. Jakimoski and Subbalakshmi [100] further built upon their work to get more efficient hash functions and MAC. Bellare et al. [20] proposed a cryptographic hash based on 4-AESRD.

8.2.2.1 1-Round AES based Hash Function

The MDP for AES s-box S is known to be 2^{-6} [157]. We define a keyed function $S' : \mathbb{F}_{2^8} \times \mathbb{F}_{2^8} \to \mathbb{F}_{2^8}$ as

$$\forall K, x \in \mathbb{F}_{2^8}^2, \quad S'_K(x) := S(K \oplus x).$$

Clearly S and S' satisfy the conditions in Lemma 8.2.2, hence S' is 2^{-6}-AXU hash function. Now, let $\mathcal{X} = \{0,1\}^8$ (sometimes viewed as $[256]$) and $\mathcal{K} = \mathbb{B}$ (viewed as $\mathcal{X}^{16}$). Given a key L, and an input $i \in \mathcal{X}$, we define the one round AES hash, aes1 as

$$\begin{aligned} \mathsf{aes1}_L(i) = \mathsf{aes1}(L, i) &:= \mathsf{1\text{-}AESRD}(L \oplus i^{16}) \\ &= \phi(S'_{L_1}(i) \| \cdots \| S'_{L_{16}}(i)), \end{aligned} \tag{8.8}$$

where i^{16} denotes i duplicated 16 times. Recall that ϕ is the linear transformation or permutation layer used in AES. It is well-known that the AXU property is preserved under linear composition over an AXU has function. Thus ϕ being a linear transformation does not affect the AXU bound for aes1 and can be ignored. When L is uniformly drawn from $\mathcal{K}$ and viewed as an element of $\mathcal{X}^{16}$ (i.e. as a 16-tuple with elements from $\mathcal{X}$), then L_i's are mutually independent. So by repeatedly using Proposition 2.3.5 and Lemma 8.2.2, we get the AXU bound for aes1 in Lemma 8.2.4.

Lemma 8.2.4. *aes1 as defined above is 2^{-96}-AXU hash function.*

8.2.2.2 2-Round AES based Hash Function

From Proposition 8.2.3 we already now that the MDP for $\texttt{aesSB}$ is upper bounded by 1.656×2^{-29}. Again we define a keyed function $\texttt{aesSB}' : \mathbb{F}_{2^{32}} \times \mathbb{F}_{2^{32}} \times \mathbb{F}_{2^{32}} \to \mathbb{F}_{2^{32}}$ as

$$\forall K^2, x \in (\mathbb{F}_{2^{32}})^3, \;\; \texttt{aesSB}'_{K^2}(x) := \texttt{aesSB}_{K_2}(K_1 \oplus x).$$

Clearly $\texttt{aesSB}'$ is 1.656×2^{-29}-AXU hash function (using Lemma 8.2.2). Now let $\mathcal{X} = \{0,1\}^{32}$ (viewed as $[2^{32}]$) and $\mathcal{K} = \mathbb{B} \times \mathbb{B}$ (viewed as $\mathcal{X}^4 \times \mathcal{X}^4$). Given a key $L = (L, L') \in \mathcal{X}^4 \times \mathcal{X}^4$, and an input $i \in \mathcal{X}$, we define the two round AES hash, aes2 as

$$\begin{aligned} \textsf{aes2}_{L,L'}(i) = \textsf{aes2}(L, L', i) &:= \text{2-}\texttt{AESRD}_{L'}(L \oplus i^4)) \\ &= \phi(\texttt{aesSB}'_{L'_1}(L_1 \oplus i) \| \cdots \| \texttt{aesSB}'_{L'_4}(L_4 \oplus i)), \end{aligned}$$

where i^4 denotes i duplicated 4 times and each L_i/L'_i is a 32-bit strings. Using a similar line of argument as used in case of aes1 we get the AXU bound for aes2 in Lemma 8.2.5.

Lemma 8.2.5. *aes2 as defined above is* 1.881×2^{-114}*-AXU hash function.*

So we see that aes1 and aes2 are sub-optimum, but decent AXU hashes given the domains they are applied upon. In addition, they would allow efficient random read access. In fact it is quite surprising that this fact was not discovered yet. However, going by the existing security proofs, the straightforward application of these hash functions in OCB is not advisable as they give sub-optimum security bounds. This is due to AXU assumption on the underlying MGF.

8.3 Generic view of OCB

In previous two sections we have seen various examples of AXU hashes over small domain $[\ell]$, where ℓ is typically a large positive integer in $[2^n]$. These hash functions are useful when the goal is to embed position-based dependency (called masks) to the input blocks of the underlying primitive, typically a block cipher. This type of embedding is used in many encryption, MAC and AE algorithms. Some prominent schemes among these belong to the OCB like design paradigms. We call the embedding function — *mask-generating function* (MGF).

MASK GENERATING FUNCTION: Typically, an MGF, $\lambda : \mathcal{L} \times \mathcal{N} \times \mathbb{N} \to \mathbb{B}$ is a $(\mathcal{L}, \mathcal{N} \times \mathbb{N})$-hash function family indexed by the key space $\mathcal{L}$. Here $\mathcal{N}$ is typically called the

nonce space. For notational simplicity we will sometime write $\lambda(i) = \lambda_L(N, i)$, i.e $N \in \mathcal{N}$ and $L \in \mathcal{L}$ will be clear from the context. Next we define generic abstraction for OCB, called GOCB, based on $\Pi \xleftarrow{\$} \mathsf{Perm}(\mathbb{B})$ and λ. Figure 8.3.1 illustrates the schematic view of the encryption-decryption algorithms for GOCB and the complete algorithmic descriptions is given in Algorithm 8.3.1. For $i \in \mathbb{N}$, we will refer M_i, A_i and C_i as message block, associated data block and ciphertext block. For $i \in \mathbb{N}$, we will refer U_i and X_i as input blocks; V_i and Y_i as output blocks; and S_i as final block. In GOCB $X_\oplus$ and $Y_\oplus$ will be referred as the tag input and output block respectively. We will use the term GPHash[3] for the associated data processing phase of GOCB.

Any binary string X can be mapped uniquely to a block sequence $\hat{X} = (\hat{X}_1, \ldots, \hat{X}_m, \hat{X}_*) \xleftarrow{n} X$, where $\ell = \lfloor |X|/n \rfloor$, i.e., $\hat{X}_i$ are complete blocks ($|\hat{X}_i| = n$) for all $i \in [\ell]$, and $\hat{X}_*$ is an incomplete (possibly empty). In the rest of this chapter, we use this methodology to parse any $X \in \{0,1\}^{\eta^*}$ into block sequences.

Note that, the constructions employ different MGFs for special blocks (last block, checksum block etc.) processing. Usually for concrete constructions these MGFs are simple variants of the usual MGF λ. Similarly the key for the MGF is usually derived from the underlying random permutation. For the sake of simplicity we will assume that the MGF key is drawn independently, and the MGFs for special blocks are also defined independently.

In this chapter, we only focus on the mask generation component and its properties. This is mainly because GOCB greatly resemble simple variant of ECB mode when the masking is ignored. Hence both efficiency and security mostly rely on the mask-generation phase, albeit the two issues are mutually orthogonal. We discuss the two issues separately starting with efficiency.

8.3.0.1 Mask-generation and Efficiency

As noted before, if we ignore the input masking, GOCB is simply a variant of ECB, which is arguably the most efficient cryptographic mode of operation, both in hardware and software. So the MGF should preferably have the following properties:

- *Sequential Computation* — MGF should be fast enough to compute the mask outputs for consecutive inputs. More formally, $\lambda(i), \lambda(i+1), \ldots, \lambda(i+r-1)$ should be efficiently computable for some r and for any i. Here r can be viewed as a parameter. One may observe the performance for different choices of r and then choose the best possible one.

[3] PHash is a commonly used terminology for the hash component in OCB.

Algorithm 8.3.1 GOCB encryption/decryption process applied over associated data A and plaintext/ciphertext M/C having a and m complete blocks and an incomplete (possibly empty) block, respectively; λ_1 is a keyed (i.e. the key is implicit) mask-generating function, and λ_i for $1 < i \leq 6$ are minor variants of λ_1, deployed in various boundary conditions, like incomplete block processing, tag generation etc.

function $\mathsf{GOCB}^{+}_{\Pi,\lambda}(N, A, M)$
- $S \leftarrow 0^n$
- $M_{\oplus} \leftarrow 0^n$
- $(A_1, \ldots, A_a, A_*) \xleftarrow{n} A$
- $(M_1, \ldots, M_m, M_*) \xleftarrow{n} M$
- **for** $i \leftarrow 1$ **to** a **do**
  - $U_i \leftarrow A_i \oplus \lambda_5(N, i)$
  - $V_i \leftarrow \Pi(U_i)$
  - $S \leftarrow S \oplus V_i$
- **end for**
- **if** $|A_*| > 0$ **then**
  - $U_* \leftarrow \mathsf{pad}(A_*) \oplus \lambda_6(N, a)$
  - $V_* \leftarrow \Pi(U_*)$
  - $S \leftarrow S \oplus V_*$
- **end if**
- **for** $i \leftarrow 1$ **to** m **do**
  - $X_i \leftarrow M_i \oplus \lambda_1(N, i)$
  - $Y_i \leftarrow \Pi(X_i)$
  - $C_i \leftarrow Y_i \oplus \lambda_1(N, i)$
  - $M_{\oplus} \leftarrow M_{\oplus} \oplus M_i$
- **end for**
- **if** $|M_*| = 0$ **then**
  - $X_{\oplus} \leftarrow M_{\oplus} \oplus \lambda_3(N, m)$
- **else**
  - $X_* \leftarrow \lambda_2(N, *)$
  - $Y_* \leftarrow \Pi(X_*)$
  - $\overline{C}_* \leftarrow Y_* \oplus \mathsf{pad}(M_*)$
  - $C_* \leftarrow \mathsf{msb}_{|M_*|}(\overline{C}_*)$
  - $M_{\oplus} \leftarrow M_{\oplus} \oplus \mathsf{pad}(M_*)$
  - $X_{\oplus} \leftarrow M_{\oplus} \oplus \lambda_4(N, m)$
- **end if**
- $C \leftarrow (C_1, \ldots, C_m, C_*)$
- $Y_{\oplus} \leftarrow \Pi(X_{\oplus})$
- $T \leftarrow Y_{\oplus} \oplus S$
- **return** (C, T)

end function

function $\mathsf{GOCB}^{-}_{\Pi,\lambda}(N, A, C, T)$
- $S \leftarrow 0^n$
- $M_{\oplus} \leftarrow 0^n$
- $(A_1, \ldots, A_a, A_*) \xleftarrow{n} A$
- $(C_1, \ldots, C_m, C_*) \xleftarrow{n} C$
- **for** $i \leftarrow 1$ **to** a **do**
  - $U_i \leftarrow A_i \oplus \lambda_5(N, i)$
  - $V_i \leftarrow \Pi(U_i)$
  - $S \leftarrow S \oplus V_i$
- **end for**
- **if** $|A_*| > 0$ **then**
  - $U_* \leftarrow \mathsf{pad}(A_*) \oplus \lambda_6(N, a)$
  - $V_* \leftarrow \Pi(U_*)$
  - $S \leftarrow S \oplus V_*$
- **end if**
- **for** $i \leftarrow 1$ **to** m **do**
  - $Y_i \leftarrow C_i \oplus \lambda_1(N, i)$
  - $X_i \leftarrow \Pi^{-1}(Y_i)$
  - $M_i \leftarrow X_i \oplus \lambda_1(N, i)$
  - $M_{\oplus} \leftarrow M_{\oplus} \oplus M_i$
- **end for**
- **if** $|C_*| = 0$ **then**
  - $X_{\oplus} \leftarrow M_{\oplus} \oplus \lambda_3(N, m)$
- **else**
  - $X_* \leftarrow \lambda_2(N, m)$
  - $Y_* \leftarrow \Pi(X_*)$
  - $\overline{M}_* \leftarrow Y_* \oplus \mathsf{pad}(C_*)$
  - $M_* \leftarrow \mathsf{msb}_{|C_*|}(\overline{M}_*)$
  - $M_{\oplus} \leftarrow M_{\oplus} \oplus \mathsf{pad}(M_*)$
  - $X_{\oplus} \leftarrow M_{\oplus} \oplus \lambda_4(N, m)$
- **end if**
- $M \leftarrow (M_1, \ldots, M_m, M_*)$
- $Y_{\oplus} \leftarrow \Pi(X_{\oplus})$
- $T' \leftarrow Y_{\oplus} \oplus S$
- **if** $T' = T$ **then**
  - **return** M
- **else**
  - **return** $\perp$
- **end if**

end function

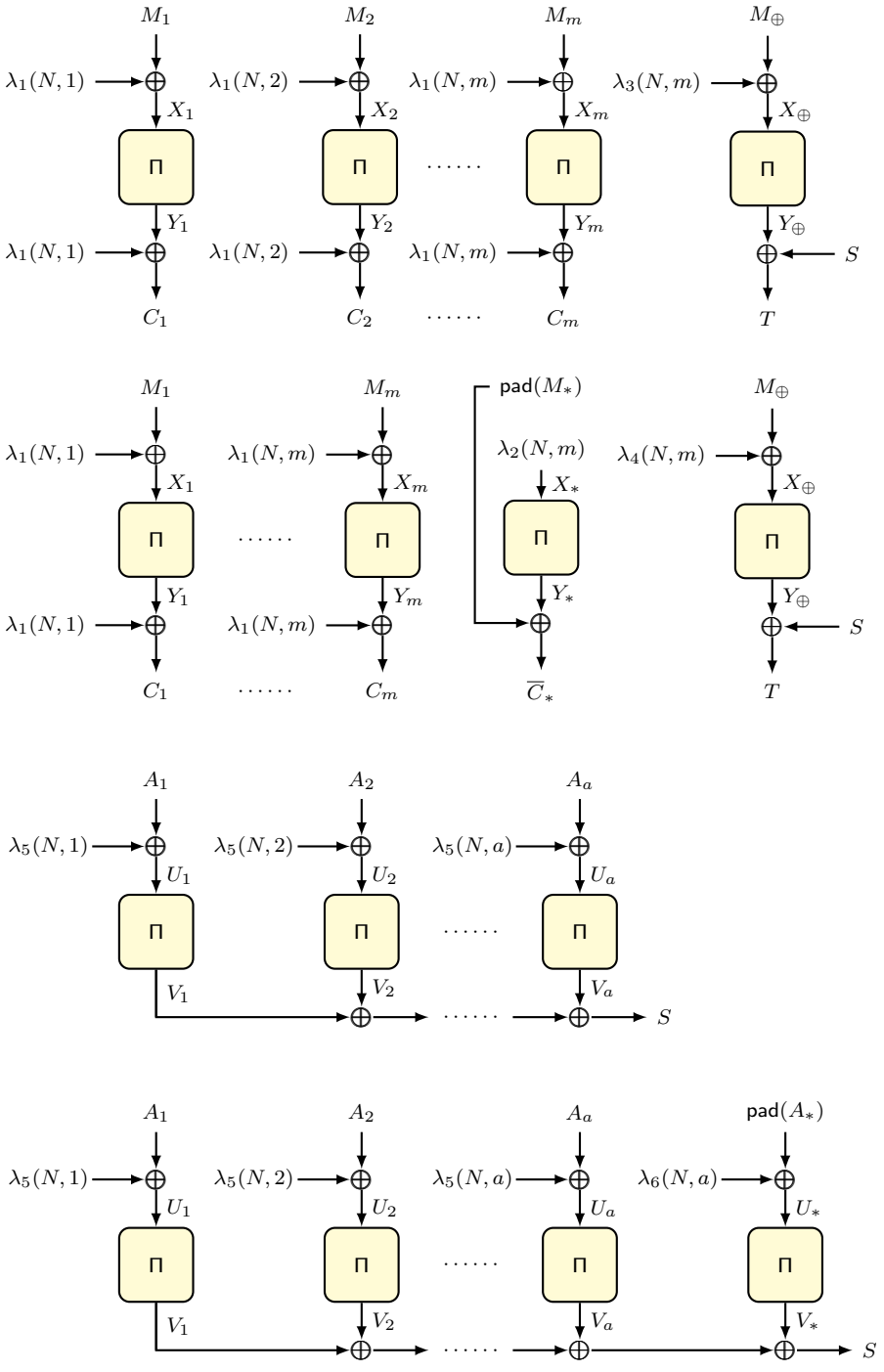

Figure 8.3.1: Schematic of GOCB— **Top to Bottom:** encrypting M when $n \mid |M|$; encrypting M when $n \nmid |M|$; hashing A when $n \mid |A|$; hashing A when $n \nmid |A|$.

- *Direct Computation* — MGF should be easily computable on any arbitrary input. In other words, for all i, $\lambda(i)$ should be efficiently computable without the knowledge of any other mask outputs. This is the usual constraint for hash functions. Direct computation of masks is an important feature as it leads to features like random read access and massive parallelism. Given large number of parallel processing units (multiple CPU cores or GPUs) GOCB is completely parallel when the underlying MGF is efficient in direct computation. This holds even when the message length is not known beforehand, a scenario that is common in data streams.
- *Constant-time Computation* — A secondary requirement for any MGF is constant computation time for all inputs. While this is a secondary requirement as far as efficiency is concerned, it helps in mitigating a form of side channel attacks called timing attack [26, 82].

8.3.0.2 Mask-Generation and Security

From the security point of view, earlier works [30, 119, 174] required strong (almost 2^{-n}) AXU and 1-universal (or regular, i.e $\Pr[H(x) = y] = O(2^{-n})$) property (some time implicitly) from the underlying MGFs. Since allmost all of the hash functions discussed in this work are 1-regular, we will implicitly assume that MGF has this property. We will focus on the more dominant, AXU property. In this setting the advantage is generally bounded in terms of ϵ, the AXU bound of MGF. For instance the bound for OCB is roughly $O(\sigma^2\epsilon)$ respectively; hence the need for $O(2^{-n})$-AXU bound. Naturally, both aes1 and aes2 would suffer significant security loss in the existing setting. This is probably one of the reasons, these hash functions have not got much attention. Yet another reason is the fact that aes1 and aes2 are defined over restricted domains $\{0,1\}^8$ and $\{0,1\}^{32}$ respectively.

8.3.1 Locally-Imperfect XOR Universal Hash Functions

The immediate question in light of the above discussion is as follows:

> *Can we relax the AXU condition on MGFs to use efficient directly computable functions like* aes1 *and* aes2*, while maintaining comparable security?*

We answer this in affirmative by proposing a slightly different universal notion — *Locally-Imperfect XOR Universal (or LIXU) Hash Functions*. The idea is to model a hash

function H whose differential probability depends on the grouping of the given inputs. More precisely, we partition the inputs in several disjoint subsets with bounded cardinality, say at most r.[4] Now the additional property introduced by LIXU is as follows: for two inputs say x and y from distinct partitions the differential probability is at most 2^{-n}, and the probability degrades to $\epsilon \geq 2^{-n}$, when x and y belong to the same partition. Formally we define the notion of Locally-Imperfect XOR Universal in Definition 8.3.1.

Definition 8.3.1 (Locally-Imperfect XOR Universal Hash Function)**.** Let $H : \mathcal{L} \times \mathcal{X} \to \mathbb{B}$ be a $(\mathcal{L}, \mathcal{X})$-hash function family indexed by the keyspace $\mathcal{L}$. For a fixed $r \in \mathbb{N}$ and $\epsilon \geq 2^{-n}$, we say that H is an (ϵ, r)-locally-imperfect XOR universal, or LIXU, hash function if and only if there exists a partitioning $\mathcal{X} = \mathcal{P}_1 \sqcup \mathcal{P}_2 \sqcup \cdots \sqcup \mathcal{P}_k$ with $k \geq 2$, such that $|\mathcal{P}_i| \leq r$ for $i \in [k]$, and for distinct $x, y \in \mathcal{X}$, and $\delta \in \mathbb{B}$,

$$\Pr_{\mathsf{L} \xleftarrow{\$} \mathcal{L}}[H_{\mathsf{L}}(x) + H_{\mathsf{L}}(y) = \delta] \leq \begin{cases} \epsilon & \text{if } x, y \in \mathcal{P}_i, \\ \frac{1}{2^n} & \text{if } x \in \mathcal{P}_i,\ y \in \mathcal{P}_j, i \neq j. \end{cases}$$

Here r is called the *width* of H. We say that $x, y \in \mathcal{X}$ are *local pairs* (to each other and as a pair as well) if and only if they belong to the same partition. So a LIXU hash function behaves as a perfect XOR universal hash function when the inputs are not local pairs, and behaves as an imperfect (or almost) XOR universal hash function when the inputs are local pairs. Note that, an (ϵ, r)-LIXU hash implies an ϵ-AXU hash, but the vice-versa may not be true. Further it is rather trivial to establish that any 2^{-n}-AXU hash function over the domain $\mathcal{X}$ is also a $(2^{-n}, r)$-LIXU hash function for all $r \leq |\mathcal{X}| - 1$. We note this simple fact in Proposition 8.3.2.

Proposition 8.3.2. *A 2^{-n}-AXU hash H is a $(2^{-n}, r)$-LIXU hash for all $r \leq |\mathcal{X}| - 1$, where $\mathcal{X}$ is the domain of H. Notably, H is a $(2^{-n}, 1)$-LIXU hash function.*

In this chapter, we use a specific type of LIXU hash functions defined over the domain $\mathcal{N} \times \mathbb{N}$, where the domain is partitioned as follows: $(N, i) \neq (N', i') \in \mathcal{N} \times \mathbb{N}$ are said to be local pairs if and only if $N = N'$ and $\lceil i/r \rceil = \lceil i'/r \rceil$. Here, $\mathcal{N}$ denotes the nonce space. We sometime use chunk and partition interchangeably, and for any $i \in \mathbb{N}$, we call $\lceil i/r \rceil$ the chunk number of i.

8.3.2 LIXU Hash as MGF in GOCB

Previous works on OCB have assumed that the underlying MGF is an AXU hash. We depart from this setting and employ LIXU hash as MGF. Since LIXU and AXU are not

[4] In fact we try to minimize the value of r.

exactly compatible, we will require a slightly different analysis. Particularly the TBC based abstraction [171] for OCB is not useful here. We now present the main security result of the chapter, along with an overview of our proof approach. The formal proof is postponed to Section 8.4.

Theorem 8.3.3 (GOCB NAEAD Bound). *For a fixed $r \leq \ell \in \mathbb{N}$ and $\epsilon \geq 0$, let λ be an (ϵ, r)-LIXU hash function over $\mathcal{N} \times \mathbb{N}$. Let $\mathbb{A}(q, \tilde{q}, \ell, \sigma, \tilde{\sigma})$ be the class of all adversaries that make q encryption queries consisting of μ message blocks in all with at most ℓ blocks per query, and ν associated data blocks in all, and $\tilde{q}$ decryption queries in a NAEAD security game against $\mathsf{GOCB}_{\Pi,\lambda}$. Then $\forall \mathscr{A} \in \mathbb{A}(q, \tilde{q}, \ell, \sigma)$ we have, $\mathbf{Adv}^{\mathsf{naead}}_{\mathsf{GOCB}_{\Pi,\lambda}}(\mathscr{A}) = \epsilon_{\mathsf{naead}}$, where*

$$\epsilon_{\mathsf{naead}} \leq \frac{0.5\sigma^2 + \widetilde{\sigma}^2}{2^n} + \frac{q\sigma + \tilde{q} + 0.5q^2}{2^n - \sigma} + \frac{\tilde{\sigma}(\sigma + 3)}{2^n} + (\sigma + 2\tilde{\sigma})r\epsilon + \widetilde{q}\epsilon.$$

Here $\sigma = \mu + \nu$ is the total number of blocks in messages and associated data queried in the encryption queries.

PROOF OVERVIEW — The proof of Theorem 8.3.3 is given in Section 8.4. While we do get a factor of $\sigma r\epsilon$ in the bound for GOCB, the original bound of OCB remains intact asymptotically if $r\epsilon = o(2^{-\frac{n}{2}})$. To understand why we don't get any degradation in security we have to look into the bound more precisely. Generally the OCB bound is dominated by the probability of collision among the input/output of the underlying block cipher. As per the earlier analyses the collisions can be bounded in terms of the AXU bound of λ, which gives $\sigma^2\epsilon/2$ bound. This evaluates to $\sigma^2 2^{1-n}$ when $\epsilon = 2^{-n}$. However when we replace AXU with LIXU the straightforward analysis will result in security degradation. Instead we first bound the probability of collision between input blocks which are non-local pairs. Since LIXU hashes are perfectly XOR universal on non-local pairs we get a bound of $\sigma^2 2^{1-n}$. Now for each input block there are at most $r - 1$ local pairs and there are at most σ input blocks which gives a bound of roughly $r\sigma\epsilon$.

8.3.3 Instantiating LIXU hash in GOCB

8.3.3.1 Deriving OCB3 from GOCB

We define the mask-generating function used in OCB3 in LIXU setting and derive concrete bound for $n = 128$. OCB3 employs gray code based masking defined using the gray hash function discussed in Section 8.1. As noted earlier gray is a 2^{-128}-AXU hash function.

For OCB3, Krovetz and Rogaway defined the gray function as: For any $i \in \mathbb{N}$, $\mathsf{gray}_{\mathsf{L}}(i) = 4\gamma(i) \odot \mathsf{L}$. For a fixed $j \in [n]$ (fixed as an application parameter) and $N \in \mathbb{B} \setminus \{0\}$, $f(N) = \mathsf{msb}_j(N) \| 0^{n-j}$, and $g(N) = \mathsf{lsb}_j(N)$. The OCB3 NAEAD advantage [119] can be obtained from GOCB by instantiating the definition of GOCB as follows:

- $\mathsf{L} = (\mathsf{L}_1, \mathsf{L}_2)$, where L_2 is a keyed function from $\mathbb{B}$ to $\mathbb{B}$, with $\mathsf{L}_1 := \mathsf{E}_{\mathsf{K}}(0)$ and $\forall N \in \mathbb{B}, \mathsf{L}_2(N) := \mathsf{sts}_{\mathsf{E}_{\mathsf{K}}(f(N))}(g(N))$.
- For $i \in [2^n]$, and $N \in \mathbb{B} \setminus \{0\}$:
  - $\lambda_1(\mathsf{L}, N, i) := \mathsf{L}_2(N) \oplus \mathsf{gray}_{\mathsf{L}_1}(i)$; $\lambda_5(\mathsf{L}, N, i) := \mathsf{gray}_{\mathsf{L}_1}(i)$;
  - $\lambda_2(\mathsf{L}, N, i) := \mathsf{L}_2(N) \oplus \mathsf{gray}_{\mathsf{L}_1}(i) \oplus \mathsf{L}_1$; $\lambda_6(\mathsf{L}, N, i) := \mathsf{gray}_{\mathsf{L}_1}(i) \oplus \mathsf{L}_1$;
  - $\lambda_3(\mathsf{L}, N, i) := \mathsf{L}_2(N) \oplus \mathsf{gray}_{\mathsf{L}_1}(i) \oplus 2\mathsf{L}_1$.
  - $\lambda_4(\mathsf{L}, N, i) := \mathsf{L}_2(N) \oplus \mathsf{gray}_{\mathsf{L}_1}(i) \oplus 3\mathsf{L}_1$.

  We write λ_i as GR_i to emphasize that λ_i is defined via gray hash function.

Here we have derived L through K, which adds a insignificant factor in the overall advantage. In [119], all variants of GR were shown to be 2^{-n}-AXU hash over the domain $\{0,1\}^{128}$. This in combination with Proposition 8.3.2, establishes that GR variants are $(2^{-n}, 1)$-LIXU hash. Using Theorem 8.3.3, $q \le \sigma$, $\widetilde{q} \le \widetilde{\sigma}$, and assuming $\sigma, \widetilde{\sigma} < 2^{n-1}$, we get the security bound for the original OCB3 design [119].

Corollary 8.3.4. *$\forall \mathscr{A} \in \mathbb{A}(q, \ell, \sigma)$ we have,*

$$\mathbf{Adv}^{\mathsf{naead}}_{\mathsf{OCB3}_{\mathsf{E}^{\pm}_{\mathsf{K}},\mathsf{GR}}}(\mathscr{A}) \le \mathbf{Adv}^{\mathsf{sprp}}_{\mathsf{E}^{\pm}_{\mathsf{K}}}(\sigma + \widetilde{\sigma}) + \frac{6.5\sigma^2 + 4\widetilde{\sigma}^2 + 7\sigma\widetilde{\sigma} + 3\sigma + 10\widetilde{\sigma}}{2^n}.$$

8.3.3.2 Instantiating GOCB with aes1 and aes2

We set $n = 128$ and $\mathcal{N} \subset \mathbb{B}$, and define

- $\mathsf{A1} : \mathsf{Func}(\mathbb{B}, \mathbb{B}) \times \mathcal{N} \times \mathbb{F}_{2^8} \to \mathbb{B}$ as, $\forall F \in \mathsf{Func}(\mathbb{B}, \mathbb{B})$, $N \in \mathcal{N}$, $i \in \mathbb{F}_{2^8}$,

$$\mathsf{A1}_F(N, i) := \mathsf{aes1}_{F(N)}(i).$$

- $\mathsf{A2} : \mathsf{Func}(\mathbb{B}, \mathbb{B}) \times \mathbb{B} \times \mathcal{N} \times \mathbb{F}_{2^{32}} \to \mathbb{B}$ as, $\forall (F, L, N, i) \in \mathsf{Func}(\mathbb{B}, \mathbb{B}) \times \mathbb{B} \times \mathcal{N} \times \mathbb{F}_{2^{32}}$,

$$\mathsf{A2}_{(F,L)}(N, i) := \mathsf{aes2}_{(F(N),L)}(i).$$

Next we show that A1 and A2 are LIXU hash functions for appropriate values of ϵ and r. In Lemma 8.3.5 and Lemma 8.3.6 we show that A1 and A2 are $(2^{-96}, 2^8)$-LIXU and $(2^{-113}, 2^{32})$-LIXU hash functions, respectively. The proofs of these lemmata are given in section 8.4.

Lemma 8.3.5. *If* $\Gamma \xleftarrow{\$} \mathsf{Func}(\mathbb{B}, \mathbb{B})$, *then* $\mathsf{A1}_\Gamma$ *is a* $(2^{-96}, 2^8)$*-LIXU hash function.*

Lemma 8.3.6. *If* $(\Gamma, \mathsf{L}) \xleftarrow{\$} \mathsf{Func}(\mathbb{B}, \mathbb{B}) \times \mathbb{B}$, *then* $\mathsf{A2}_{\Gamma,\mathsf{L}}$ *is a* $(2^{-113}, 2^{32})$*-LIXU hash function.*

Description of $\mathsf{GOCB}_{\mathsf{E},\mathsf{A1}}$ — We define GOCB based on A1 as follows:

- The maximum message and associated data size per query is limited to 2^6 blocks, i.e. 1 Kilobytes, and the nonce space is $\mathbb{B} \setminus \{0\}$.
- $\mathsf{L} = \mathsf{F}$, where F is a keyed function from $\mathbb{B}$ to $\mathbb{B}$, where $\forall N \in \mathbb{B}$, $\mathsf{F}(N) := \mathsf{E}_\mathsf{K}(N)$.
- For $i \in [2^6]$, and $N \in \mathbb{B} \setminus \{0\}$:
  - $\lambda_1(\mathsf{F}, N, i) := \mathsf{A1}_\mathsf{F}(N, 4i + 0)$; and $\lambda_5(\mathsf{F}, N, i) := \mathsf{A1}_\mathsf{F}(N, 4i + 0) \oplus \mathsf{E}_K(0)$.
  - $\lambda_2(\mathsf{F}, N, i) := \mathsf{A1}_\mathsf{F}(N, 4i + 1)$; and $\lambda_6(\mathsf{F}, N, i) := \mathsf{A1}_\mathsf{F}(N, 4i + 1) \oplus \mathsf{E}_K(0)$.
  - $\lambda_3(\mathsf{F}, N, i) := \mathsf{A1}_\mathsf{F}(N, 4i + 2)$;
  - $\lambda_4(\mathsf{F}, N, i) := \mathsf{A1}_\mathsf{F}(N, 4i + 3)$;

  where $4i$ denotes the integer multiplication of i with 4.

Here we need to add a factor of $(q + 1)(q + \sigma)2^{-128}$ to bound the probability that some masked input matches with some nonce value. Further a factor of $q^2 2^{-129}$ is required for PRF-PRP switch. Using Theorem 8.3.3, $q < \sigma$, $\widetilde{q} < \widetilde{\sigma}$, and $\sigma, \widetilde{\sigma} < 2^{127}$, we get the security result for $\mathsf{GOCB}_{\mathsf{E}_K,\mathsf{A1}}$ as in Corollary 8.3.7.

Corollary 8.3.7. $\forall \mathscr{A} \in \mathbb{A}(q, \tilde{q}, \sigma)$ *we have,*

$$\mathbf{Adv}^{\mathsf{naead}}_{\mathsf{GOCB}_{\mathsf{E}^{\pm}_{\mathsf{K}},\mathsf{A1}}}(\mathscr{A}) \leq \mathbf{Adv}^{\mathsf{sprp}}_{\mathsf{E}^{\pm}_{\mathsf{K}}}(\sigma + \widetilde{\sigma}) + \frac{6.5\sigma^2 + 4\widetilde{\sigma}^2 + 7\sigma\widetilde{\sigma} + 2\sigma + 7\widetilde{\sigma}}{2^{128}} + \frac{\sigma + 2\widetilde{\sigma}}{2^{88}} + \frac{\widetilde{q}}{2^{96}}.$$

Description of $\mathsf{GOCB}_{\mathsf{E},\mathsf{A2}}$ — $\mathsf{GOCB}_{\mathsf{E},\mathsf{A2}}$ can be defined analogously using mask hash key $L = (L_1, L_2, F)$, where $L_1 = \mathsf{E}_K(0)$ and $L_2 = \mathsf{E}_K(1)$. We define GOCB based on A2 as follows:

- The maximum message and associated data size per query is limited to 2^{30} blocks, i.e. 16 Gigabytes, and the nonce space is $\mathbb{B} \setminus \{0, 1\}$.

- $\mathsf{L} = (\mathsf{L}_1, \mathsf{F})$, where $\mathsf{L}_1 = \mathsf{E}_\mathsf{K}(0)$ and F is a keyed function from $\mathbb{B}$ to $\mathbb{B}$. For all $N \in \mathbb{B} \setminus \{0,1\}$, $\mathsf{F}(N) := \mathsf{E}_\mathsf{K}(N)$.
- For $i \in [2^{30}]$, and $N \in \mathbb{B}$:
  - $\lambda_1(\mathsf{L}, N, i) := \mathsf{A2}_\mathsf{L}(N, 4i+0)$; and $\lambda_1(\mathsf{L}, N, i) := \mathsf{A2}_\mathsf{L}(N, 4i+0) \oplus \mathsf{E}_K(1)$.
  - $\lambda_2(\mathsf{L}, N, i) := \mathsf{A2}_\mathsf{L}(N, 4i+1)$; and $\lambda_2(\mathsf{L}, N, i) := \mathsf{A2}_\mathsf{L}(N, 4i+1) \oplus \mathsf{E}_K(1)$.
  - $\lambda_3(\mathsf{L}, N, i) := \mathsf{A2}_\mathsf{L}(N, 4i+2)$;
  - $\lambda_4(\mathsf{L}, N, i) := \mathsf{A2}_\mathsf{L}(N, 4i+3)$;

  where $4i$ denotes the integer multiplication of i with 4.

Using similar argument as above we can derive the security bound, given in Corollary 8.3.8, for GOCB instantiated with A2.

Corollary 8.3.8. *$\forall \mathscr{A} \in \mathbb{A}(q, \tilde{q}, \sigma)$ we have,*

$$\mathbf{Adv}^{\mathsf{naead}}_{\mathsf{GOCB}_{\mathsf{E}^{\pm}_\mathsf{K},\mathsf{A2}}}(\mathscr{A}) \leq \mathbf{Adv}^{\mathsf{sprp}}_{\mathsf{E}^{\pm}_\mathsf{K}}(\sigma + \widetilde{\sigma}) + \frac{6.5\sigma^2 + 4\widetilde{\sigma}^2 + 7\sigma\widetilde{\sigma} + 4\sigma + 9\widetilde{\sigma}}{2^{128}} + \frac{\sigma + 2\widetilde{\sigma}}{2^{81}} + \frac{\widetilde{q}}{2^{113}}.$$

8.3.3.3 Extending the Domain of $\mathsf{GOCB}_{\mathsf{E},\mathsf{A1}}$ and $\mathsf{GOCB}_{\mathsf{E},\mathsf{A2}}$

The A1 and A2 based instantiations have a restriction on the maximum input length. This is more prominent in case of $\mathsf{GOCB}_{\mathsf{E},\mathsf{A1}}$, which allows message lengths up to 1 KB. Here we give some ways to extend the domain of $\mathsf{GOCB}_{\mathsf{E},\mathsf{A1}}$ and $\mathsf{GOCB}_{\mathsf{E},\mathsf{A2}}$. Our general idea remains the same in all the methods. We improve the maximum input size restriction to roughly 2^{50} bytes or 2^{46} blocks, a usual limit on input size [156]. We need 46 bits to represent each block of input uniquely. We divide the 46-bit length representation into two parts $i \| j$, where j is 6-bit long for $\mathsf{GOCB}_{\mathsf{E},\mathsf{A1}}$ and j is 30-bit long for $\mathsf{GOCB}_{\mathsf{E},\mathsf{A2}}$. We will handle the j-value in the same way as before, and employ different techniques to handle (N, i) input of the mask generating function.

1. DOMAIN EXTENSION VIA THE F FUNCTION: We can consider a keyed function over larger domain, say $\mathcal{N} \times [2^{128}]$. For example, consider the definition $\mathsf{F}(N, i) := \mathsf{E}_\mathsf{K}(i \oplus \mathsf{E}_\mathsf{K}(N))$. This would add an extra $O(\sigma^2/2^n)$ term, a rough upper bound to avoid block cipher input collisions involving $i \oplus \mathsf{E}_\mathsf{K}(N)$ for some (N, i). This method can clearly extend the message size to 2^{46} but at the cost of an extra block cipher call every time the j value resets to zero, plus it needs one more encryption call for the nonce.

2. DOMAIN EXTENSION VIA NONCE SIZE REDUCTION: Another way is to reduce the nonce size to 88 bits and 112 bits for $\mathsf{GOCB}_{\mathsf{E},\mathsf{A1}}$ and $\mathsf{GOCB}_{\mathsf{E},\mathsf{A2}}$, respectively, and let $\mathsf{F}(N,i) := \mathsf{E}_{\mathsf{K}}(N\|i)$. This is slightly better than the previous method as we can save one block cipher call. As before, this adds an extra $O(\sigma^2/2^n)$ term, a rough upper bound to avoid block cipher input collisions.

3. DOMAIN EXTENSION VIA SUM OF LIXU HASH: We explain the method for $\mathsf{GOCB}_{\mathsf{E},\mathsf{A2}}$. But similar technique is also applicable to $\mathsf{GOCB}_{\mathsf{E},\mathsf{A1}}$. We modify the λ function as follows:

- First, we redefine $\mathsf{L} = (\mathsf{L}_1, \mathsf{L}_2, \mathsf{F}, \mathsf{F}')$, where $\mathsf{L}_1 = \mathsf{E}_{\mathsf{K}}(0)$ and $\mathsf{L}_2 = \mathsf{E}_{\mathsf{K}}(1)$. The nonce space is $\mathcal{N} = \{0,1\}^{n-1} \setminus \{0,1\}$, and F and F' are keyed functions from $\mathcal{N}$ to $\mathbb{B}$, defined as $\mathsf{F}_{\mathsf{K}}(N) = \mathsf{E}_{\mathsf{K}}(N\|0)$ and $\mathsf{F}'_{\mathsf{K}}(N) = \mathsf{E}_{\mathsf{K}}(N\|1)$.
- Second, $\lambda_1(\mathsf{L}, N, i\|j) := \mathsf{A2}_{\mathsf{L}_1,\mathsf{F}}(N,i) \oplus \mathsf{A2}_{\mathsf{L}_2,\mathsf{F}}(N,j)$. The other variants are defined analogously as before. Further for AD processing, we XOR $\mathsf{E}_{\mathsf{K}}(2)$ (instead of $\mathsf{E}_{\mathsf{K}}(1)$) to the modified λ_1 and λ_2 values.

It can be easily shown that this modification offers asymptotically similar security as before.

8.4 Security Proofs

In this section we provide the proof for our main result Theorem 8.3.3, along with the proofs for Lemma 8.3.5 and Lemma 8.3.6.

8.4.1 Inaptness of the Existing TBC-based Abstraction

In [171], Rogaway introduced a very nice abstraction for OCB like constructions based on tweakable block ciphers (TBCs). More formally, the TBC abstraction of [171] views the masked input-output block cipher as a way to construct TBCs from block ciphers and efficient mask generating functions, called XEX. In this view any OCB like construction can be viewed as an instance of the TBC-based authenticated encryption called ΘCB, where each encryption TBC call takes a tuple of tweak values (N,i), where N denotes the nonce and i denotes the block index.

This abstraction simplifies as well as modularizes the proof, as it can be shown (using results from [171]) that the NAEAD advantage is bounded by the tweakable strong pseudorandom permutation (TSPRP) advantage of XEX, which is bounded by at most $\sigma^2\epsilon$, where σ denotes the total number of TBC calls and ϵ denotes the AXU bound of

the underlying MGF. Consequently, this approach was used in several previous works, most notably [45] and [81].

While the reduction from the NAEAD game for OCB to the TSPRP game for XEX gives tight estimate on the privacy bounds (exactly same as the TSPRP bound for XEX), the reduction is rather loose in case of authenticity bound, as noted in [30]. A straightforward approach gives an authenticity bound of the form $O(\sigma^2\epsilon) + O(1/2^n)$ for single forgery. The first term here is due to the TSPRP advantage of XEX. Here the $O(\sigma^2\epsilon)$ term is due to the XEX to tweakable random permutation reduction. When extended to multiple verification queries, this results in a bound of the form $O(\tilde{q}\sigma^2\epsilon) + O(\tilde{q}/2^n)$, where $\tilde{q}$ denotes the number of verification queries. Clearly the security degrades for multiple verification queries.

Yet another approach gives a security bound of the form $O(\sigma^2\epsilon) + O(\tilde{\sigma}^2\epsilon) + O(\tilde{q}/2^n)$, where $\tilde{\sigma}$ denotes the number of TBC calls in all decryption queries. Here the $O(\sigma^2\epsilon)$ and $O(\tilde{\sigma}^2\epsilon)$ terms are due to the XEX to tweakable random permutation reduction.

A constrained variant of the XEX based abstraction could also be used to get a modularized privacy bound for GOCB. In this abstraction the adversary against tweakable block cipher is restricted to at most r many queries per chunk number. This restriction is perfectly fine, as the NAEAD adversary is nonce-respecting for encryption queries and for each nonce the message length is at most r. This gives a privacy bound of the form $O(\sigma^2/2^n) + O(\sigma r\epsilon)$. But the same is not true for authenticity, as the adversary can make all the decryption queries with tweaks within the same chunk. This results in a term of the form $O(\tilde{\sigma}^2\epsilon)$, which is clearly sub-optimal.

So, we use a more direct approach as also employed in [30]. The proof for Theorem 8.3.3 uses coefficient-H technique.

MORE NOTATIONS: We say a function $f : \mathcal{X} \to \mathcal{Y}$ is *partial* or *restricted*, if we know the values of f on a strict subset of $\mathcal{X}$. This subset is called $\mathsf{domain}(f)$. A partial function f can be viewed as a restriction of f to $\mathsf{domain}(f)$. The range of this restricted function is called $\mathsf{range}(f)$. We will treat a partial function as *updatable*: for some $x \in \mathbb{B} \setminus \mathsf{domain}(f)$ and some $y \in \mathcal{Y}$, (x, y) may be added to f, so that $\mathsf{domain}(f)$ expands to $\mathsf{domain}(f) \cup \{x\}$, and $\mathsf{range}(f)$ becomes $\mathsf{range}(f) \cup \{y\}$.

8.4.2 Proof of GOCB NAEAD Bound (Theorem 8.3.3)

Let $\mathcal{O}_1^{\pm}$ denote the real oracle corresponding to GOCB, and $\mathcal{O}_0^{\pm}$ denote the ideal oracle corresponding to $(\Gamma, \perp)$. We start off by setting up the necessary notations related to the query-response transcript of $\mathscr{A}$. The transcript generated by $\mathscr{A}$ consists of

- *Encryption query-response tuples:* for $i \in [q]$, the i-th encryption query-response tuple is built of
  - N_i: the queried nonce block, such that for any $i' < i$, $N_i \neq N_{i'}$.
  - A_i: the queried associated data, consisting of a_i complete blocks and an incomplete (possibly empty) block $A_{(i,*)}$ at the end;
  - M_i: the queried message, consisting of m_i complete blocks and an incomplete (possibly empty) block $M_{(i,*)}$ at the end;
  - C_i: the response ciphertext, such that $|C_i| = |M_i|$;
  - T_i: the response tag block.
- *Decryption query-response tuples:* for $i \in [\widetilde{q}]$ the i-th decryption query-response tuple is built of
  - $\widetilde{N}_i$: the queried nonce block. Note that, $\widetilde{N}_i$'s can be repeated.
  - $\widetilde{A}_i$: the queried associated data, consisting of $\widetilde{a}_i$ complete blocks and an incomplete (possibly empty) block $\widetilde{A}_{(i,*)}$ at the end;
  - $\widetilde{C}_i$: the queried ciphertext, consisting of $\widetilde{m}_i$ complete blocks and an incomplete (possibly empty) block $\widetilde{C}_{(i,*)}$ at the end;
  - $\widetilde{T}_i$: the queried tag block.
  - $\widetilde{M}_i$: the response message, such that $|\widetilde{M}_i| = |\widetilde{C}_i|$. Note that, the decryption oracle may return $\perp$, in which case $\widetilde{M}_i$ just denotes the unauthenticated decrypted message to be used internally.

The internal variables arising in one call to the encryption oracle are analogously as given in Algorithm 8.3.1 and Figure 8.3.1, while the internal variables from the decryption oracle are defined identically, but topped with a tilde to differentiate them from the encryption variables.

Let $\mathcal{I}$ and $\mathcal{J}$ denote the encryption query indices with incomplete-block message and associated data respectively. The decryption counterparts are $\widetilde{\mathcal{I}}$ and $\widetilde{\mathcal{J}}$. Let $I = I_1 \cup I_2$ and $O = O_1 \cup O_2$ be multisets, where for all $i \in [2]$, I_i and O_i are defined as follows

$$
\begin{aligned}
I_1 &:= \left\{U_{(i,j)} : (i,j) \in [q] \times [a_i]\right\} \cup \left\{U_{(i,*)} : i \in \mathcal{J}\right\}, \\
I_2 &:= \left\{X_{(i,j)} : (i,j) \in [q] \times ([m_i] \cup \{\oplus\})\right\} \cup \left\{X_{(i,*)} : i \in \mathcal{I}\right\}, \\
O_1 &:= \left\{V_{(i,j)} : (i,j) \in [q] \times [a_i]\right\} \cup \left\{V_{(i,*)} : i \in \mathcal{J}\right\}, \\
O_2 &:= \left\{Y_{(i,j)} : (i,j) \in [q] \times ([m_i] \cup \{\oplus\})\right\} \cup \left\{Y_{(i,*)} : i \in \mathcal{I}\right\}.
\end{aligned}
$$

8.4.2.1 Oracle Behavior and Transcript Extension

Consider a modified security game where we let the real oracle $\mathcal{O}_1$ reveal all the internal input and output blocks appearing in the encryption query phase. Consequently we have to make appropriate behavioral changes in the ideal oracle $\mathcal{O}_0$ to release these internal variables. We describe the sampling behavior of $\mathcal{O}_0$ in greater detail in the subsequent paragraphs. During the sampling $\mathcal{O}_0$ might set $\mathsf{bad} = 1$, whereafter its behavior is undefined.

SAMPLING BEHAVIOR OF $\mathcal{O}_0$

Query Phase: For $i \in [q]$, on the i-th encryption query, for each $j \in [m_i]$, $\mathsf{C}_{(i,j)} \leftarrow\!\!\$\, \mathbb{B}$; $\mathsf{T}_i \leftarrow\!\!\$\, \mathbb{B}$; and if $i \in \mathcal{I}$, $\overline{\mathsf{C}}_{(i,*)} \leftarrow\!\!\$\, \mathbb{B}$, set $\mathsf{C}_{(i,*)} = \mathsf{msb}_{|\mathsf{M}_{(i,*)}|}(\overline{\mathsf{C}}_{i,*})$; return $\mathsf{C}_i = \mathsf{C}_{(i,1)} \| \cdots \| \mathsf{C}_{(i,*)}$ and T_i to $\mathscr{A}$. For $i \in [\tilde{q}]$, on the i-th decryption query, return $\perp$ to $\mathscr{A}$.

Post-query Phase: Let $\mathsf{L} \leftarrow\!\!\$\, \mathcal{L}$ and Π, a permutation over $\mathbb{B}$, be undefined on all blocks, i.e $\mathsf{domain}(\Pi) = \mathsf{range}(\Pi) = \emptyset$.

Step 1: Extending the encryption query-response tuple — Set the following values:

- for $i \in [q]$, $j \in [m_i]$ set $\mathsf{X}_{(i,j)} = \mathsf{M}_{(i,j)} \oplus \lambda_1(\mathsf{L}, \mathsf{N}_i, j)$ and $\mathsf{Y}_{(i,j)} = \mathsf{C}_{(i,j)} \oplus \lambda_1(\mathsf{L}, \mathsf{N}_i, j)$;
- for $i \in \mathcal{I}$, set $\mathsf{X}_{(i,*)} = \lambda_2(\mathsf{L}, \mathsf{N}_i, m_i)$ and $\mathsf{Y}_{(i,*)} = \overline{\mathsf{C}}_{(i,*)} \oplus \mathsf{pad}(\mathsf{M}_{(i,*)})$;
- for $i \in [q] \setminus \mathcal{I}$, set $\mathsf{M}_{(i,\oplus)} = \oplus_{j=1}^{m_i} \mathsf{M}_{(i,j)}$ and $\mathsf{X}_{(i,\oplus)} = \mathsf{M}_{(i,\oplus)} \oplus \lambda_3(\mathsf{L}, \mathsf{N}_i, m_i)$;
- for $i \in \mathcal{I}$, set $\mathsf{M}_{(i,\oplus)} = \oplus_{j=1}^{m_i} \mathsf{M}_{(i,j)} \oplus \mathsf{pad}(\mathsf{M}_{(i,*)})$ and $\mathsf{X}_{(i,\oplus)} = \mathsf{M}_{(i,\oplus)} \oplus \lambda_4(\mathsf{L}, \mathsf{N}_i, m_i)$;
- for $i \in [q]$, $j \in [a_i]$ set $\mathsf{U}_{(i,j)} = \mathsf{A}_{(i,j)} \oplus \lambda_5(\mathsf{L}, \mathsf{N}_i, j)$;
- for $i \in \mathcal{J}$, set $\mathsf{U}_{(i,*)} = \mathsf{pad}(\mathsf{A}_{(i,a_i)}) \oplus \lambda_6(\mathsf{L}, \mathsf{N}_i, a_i)$;
- set $\mathsf{bad} = 1$, if I or O (the partial multiset containing $\mathsf{Y}_{(i,j)}$ values for $i \in [q]$ and $j \in [m_i] \cup \{*\}$) contains a non-trivial colliding pair (pair of duplicate elements). Note that, all colliding pairs are non-trivial at this stage;
- for all $i \in [q],\ j \in [m_i]$, fix $\Pi(\mathsf{X}_{(i,j)}) = \mathsf{Y}_{(i,j)}$, and for all $i \in \mathcal{I}$, fix $\Pi(\mathsf{X}_{(i,*)}) = \mathsf{Y}_{(i,*)}$;
- for all $i \in [q]$, $j \in [a_i]$, fix $\Pi(\mathsf{U}_{(i,j)}) = \mathsf{V}_{(i,j)} \leftarrow\!\!\$\, \mathbb{B} \setminus \mathsf{range}(\Pi)$, and for all $i \in \mathcal{J}$, fix fix $\Pi(\mathsf{U}_{(i,*)}) = \mathsf{V}_{(i,*)} \leftarrow\!\!\$\, \mathbb{B} \setminus \mathsf{range}(\Pi)$;
- for $i \in [q] \setminus \mathcal{J}$, set $\mathsf{S}_i = \oplus_{j=1}^{a_i} \mathsf{V}_{(i,j)}$;
- for $i \in \mathcal{J}$, set $\mathsf{S}_i = \oplus_{j=1}^{a_i} \mathsf{V}_{(i,j)} \oplus \mathsf{V}_{(i,*)}$;
- for $i \in [q]$ set $\mathsf{Y}_{(i,\oplus)} = \mathsf{S}_i \oplus \mathsf{T}_i$.

- set $\mathsf{bad} = 1$, if O (the complete multiset) contains a non-trivial colliding pair (pair of duplicate elements).
- for $i \in [q]$, fix $\Pi(\mathsf{X}_{(i,\oplus)}) = \mathsf{Y}_{(i,\oplus)}$.

Note that, the partial sampling of Π remains permutation-compatible in step 1 as long as $\mathsf{bad} = 0$.

Step 2: Extending the decryption query-response tuple — Set the following values:

- at the start all intermediate variables are set as undefined.
- for $i \in [\widetilde{q}]$, $j \in [\widetilde{m}_i]$, set $\widetilde{\mathsf{Y}}_{(i,j)} = \widetilde{\mathsf{C}}_{(i,j)} \oplus \lambda_1(\mathsf{L}, \widetilde{\mathsf{N}}_i, j)$; if $i \in \widetilde{\mathcal{I}}$, set $\widetilde{\mathsf{X}}_{(i,*)} = \lambda_2(\mathsf{L}, \widetilde{\mathsf{N}}_i, \widetilde{m}_i)$.
- for $i \in [\widetilde{q}]$, $j \in [\widetilde{m}_i]$, if $\widetilde{\mathsf{Y}}_{(i,j)} \in \mathsf{range}(\Pi)$ then set $\widetilde{\mathsf{X}}_{(i,j)} = \Pi^{-1}(\widetilde{\mathsf{Y}}_{(i,j)})$ and $\widetilde{\mathsf{M}}_{(i,j)} = \widetilde{\mathsf{X}}_{(i,j)} \oplus \lambda_1(\mathsf{L}, \widetilde{\mathsf{N}}_i, j)$; if $i \in \widetilde{\mathcal{I}}$ and $\widetilde{\mathsf{X}}_{(i,*)} \in \mathsf{domain}(\Pi)$, then set $\widetilde{\mathsf{Y}}_{(i,*)} = \Pi(\widetilde{\mathsf{X}}_{(i,*)})$ and $\widetilde{\mathsf{M}}_{(i,*)} = \mathsf{msb}_{|\widetilde{\mathsf{C}}_{(i,*)}|}(\widetilde{\mathsf{Y}}_{(i,*)}) \oplus \widetilde{\mathsf{C}}_{(i,*)}$; if $\widetilde{\mathsf{M}}_i$ is defined then set $\widetilde{\mathsf{X}}_{(i,\oplus)} = \widetilde{\mathsf{M}}_{(i,\oplus)} \oplus \lambda_3(\mathsf{L}, \widetilde{\mathsf{N}}_i, \widetilde{m}_i)$.
- for $i \in [\widetilde{q}]$, $j \in [\widetilde{a}_i]$, set $\widetilde{\mathsf{U}}_{(i,j)} = \widetilde{\mathsf{A}}_{(i,j)} \oplus \lambda_5(\mathsf{L}, \widetilde{\mathsf{N}}_i, j)$; if $i \in \widetilde{\mathcal{J}}$, set $\widetilde{\mathsf{U}}_{(i,*)} = \lambda_6(\mathsf{L}, \widetilde{\mathsf{N}}_i, \widetilde{a}_i)$.
- for $i \in [\widetilde{q}]$, $j \in [\widetilde{a}_i]$, if $\widetilde{\mathsf{U}}_{(i,j)} \in \mathsf{domain}(\Pi)$ then set $\widetilde{\mathsf{V}}_{(i,j)} = \Pi(\widetilde{\mathsf{U}}_{(i,j)})$; if $\widetilde{\mathsf{V}}_i$ is defined then set $\widetilde{\mathsf{Y}}_{(i,\oplus)} = \widetilde{\mathsf{S}}_i \oplus \widetilde{\mathsf{T}}_i$.
- At this stage all those intermediate variables, which can be derived through the MGF key L, and adversary's query, are completely determined. Also, some other variables are trivially derived due to the extended encryption transcript.
- set $\mathsf{bad} = 1$, if there exists $(i,j) \in [\widetilde{q}] \times ([\widetilde{m}_i] \cup \{*, \oplus\})$, and one of the following is true:
  - $\exists (i', j') \in [q] \times ([m_{i'}] \cup \{*, \oplus\})$ such that $(\widetilde{\mathsf{N}}_i, \widetilde{\mathsf{C}}_{(i,j)}) \neq (\mathsf{N}_{i'}, \mathsf{C}_{(i',j')})$ and $\widetilde{\mathsf{Y}}_{(i,j)} = \mathsf{Y}_{(i',j')}$.
  - $\exists (i', j') \in [q] \times ([a_{i'}] \cup \{*\})$ such that $\widetilde{\mathsf{Y}}_{(i,j)} = \mathsf{V}_{(i',j')}$.
- set $\mathsf{bad} = 1$, if there exists $i \in [\widetilde{q}] \times ([\widetilde{a}_i] \cup \{*\})$, and one of the following is true:
  - $\exists (i', j') \in [q] \times ([m_{i'}] \cup \{*, \oplus\})$ such that $\widetilde{\mathsf{U}}_{(i,j)} = \mathsf{X}_{(i',j')}$.
  - $\exists (i', j') \in [q] \times ([a_{i'}] \cup \{*\})$ such that $(\widetilde{\mathsf{N}}_i, \widetilde{\mathsf{A}}_{(i,j)}) \neq (N_{i'}, \widetilde{\mathsf{A}}_{(i',j')})$ and $\widetilde{\mathsf{U}}_{(i,j)} = \mathsf{U}_{(i',j')}$.

Extend the transcript by including all the internal variables computed thus far and return the extended transcript to $\mathscr{A}$.

IDENTIFYING THE BAD TRANSCRIPTS: Let Ω denote the set of all realizable transcripts. We say that $\omega \in \Omega$ is a bad transcript if it causes bad $= 1$. In other words, we say that ω is bad if one of the following events occur:

`EEcoll`:

- $\exists\, (i,j) \in [q] \times [m_i] \cup \{*, \oplus\}\,,\ (i',j') \in [q] \times [m_{i'}] \cup \{*, \oplus\}\ :\ (i,j) \neq (i',j') \wedge \mathsf{X}_{(i,j)} = \mathsf{X}_{(i',j')}$.
- $\exists\, (i,j) \in [q] \times [a_i] \cup \{*\}\,,\ (i',j') \in [q] \times [a_{i'}] \cup \{*\}\ :\ (i,j) \neq (i',j') \wedge \mathsf{U}_{(i,j)} = \mathsf{U}_{(i',j')}$.
- $\exists\, (i,j) \in [q] \times [m_i] \cup \{*, \oplus\}\,,\ (i',j') \in [q] \times [a_{i'}] \cup \{*\}\ :\ \mathsf{X}_{(i,j)} = \mathsf{U}_{(i',j')}$.
- $\exists\, (i,j) \in [q] \times [m_i] \cup \{*\}\,,\ (i',j') \in [q] \times [m_{i'}] \cup \{*\}\ :\ (i,j) \neq (i',j') \wedge \mathsf{Y}_{(i,j)} = \mathsf{Y}_{(i',j')}$.

`ETcoll`:

- $\exists\, i \in [q],\ (i',j') \in [q] \times [m_{i'}] \cup \{*\}\ :\ \mathsf{Y}_{(i,\oplus)} = \mathsf{Y}_{(i',j')}$.
- $\exists^*\, i, i' \in [q]\ :\ \mathsf{Y}_{(i,\oplus)} = \mathsf{Y}_{(i',\oplus)}$.
- $\exists\, i \in [q],\ (i',j') \in [q] \times [a_{i'}] \cup \{*\}\ :\ \mathsf{Y}_{(i,\oplus)} = \mathsf{V}_{(i',j')}$.

`DEcoll`:

- $\exists\, (i,j) \in [\widetilde{q}] \times [\widetilde{m}_i] \cup \{*\}\,,\ (i',j') \in [q] \times [m_{i'}] \cup \{*, \oplus\}\ :\ (\widetilde{\mathsf{N}}_i, \widetilde{\mathsf{C}}_{(i,j)}) \neq (\mathsf{N}_{i'}, \mathsf{C}_{(i',j')}) \wedge \widetilde{\mathsf{Y}}_{(i,j)} = \mathsf{Y}_{(i',j')}$.
- $\exists\, (i,j) \in [\widetilde{q}] \times [\widetilde{m}_i] \cup \{*\}\,,\ (i',j') \in [q] \times [a_{i'}] \cup \{*\}\ :\ \widetilde{\mathsf{Y}}_{(i,j)} = \mathsf{V}_{(i',j')}$.
- $\exists\, (i,j) \in [\widetilde{q}] \times [\widetilde{a}_i] \cup \{*\}\,,\ (i',j') \in [q] \times [a_{i'}] \cup \{*\}\ :\ (\widetilde{\mathsf{N}}_i, \widetilde{\mathsf{A}}_{(i,j)}) \neq (\mathsf{N}_{i'}, \mathsf{A}_{(i',j')}) \wedge \widetilde{\mathsf{U}}_{(i,j)} = \mathsf{U}_{(i',j')}$.
- $\exists\, (i,j) \in [\widetilde{q}] \times [\widetilde{a}_i] \cup \{*\}\,,\ (i',j') \in [q] \times [m_{i'}] \cup \{*, \oplus\}\ :\ \widetilde{\mathsf{U}}_{(i,j)} = \mathsf{X}_{(i',j')}$.

`DEFcoll`:

- $\exists\, i \in [\widetilde{q}],\ i' \in [q]\ :\ (\widetilde{\mathsf{N}}_i, \widetilde{\mathsf{A}}_i, \widetilde{\mathsf{T}}_i) = (\mathsf{N}_{i'}, \mathsf{A}_{i'}, \mathsf{T}_{i'}) \wedge \widetilde{\mathsf{C}}_i = \mathsf{C}_{i'}^{[\widetilde{m}_i]} \wedge \widetilde{\mathsf{X}}_{(i,\oplus)} \in I$.

`DDFcoll`:

- $\exists\, (i,j) \in [\widetilde{q}] \times [\widetilde{m}_i] \cup \{*\}\,,\ i' \in [q]\ :\ (\widetilde{\mathsf{N}}_i, \widetilde{\mathsf{A}}_i, \widetilde{\mathsf{C}}_i, \widetilde{\mathsf{T}}_i) = (\mathsf{N}_{i'}, \mathsf{A}_{i'}, \mathsf{C}_{i'}^{[\widetilde{m}_i]}, \mathsf{T}_{i'}) \wedge \widetilde{\mathsf{X}}_{(i,\oplus)} = \widetilde{\mathsf{X}}_{(i,j)}$.
- $\exists\, (i,j) \in [\widetilde{q}] \times [\widetilde{a}_i] \cup \{*\}\,,\ i' \in [q]\ :\ (\widetilde{\mathsf{N}}_i, \widetilde{\mathsf{A}}_i, \widetilde{\mathsf{C}}_i, \widetilde{\mathsf{T}}_i) = (\mathsf{N}_{i'}, \mathsf{A}_{i'}, \mathsf{C}_{i'}^{[\widetilde{m}_i]}, \mathsf{T}_{i'}) \wedge \widetilde{\mathsf{X}}_{(i,\oplus)} = \widetilde{\mathsf{U}}_{(i,j)}$.
- $\exists\, i \in [\widetilde{q}],\ \exists^* j, j' \in [\widetilde{m}_i] \cup \{*, \oplus\}\ :\ \widetilde{\mathsf{Y}}_{(i,j)} = \widetilde{\mathsf{Y}}_{(i,j')}$.

Let Ω_{bad} be the set of all transcripts which are bad and $\Omega_{\mathsf{good}} = \Omega \setminus \Omega_{\mathsf{bad}}$. We bound $\mathsf{ip}_{\mathcal{O}_0}[\Omega_{\mathsf{bad}}]$ to $O(\sigma^2 2^{-n} + r\sigma\epsilon + q\sigma 2^{-n} + q^2 2^{-n} + \widetilde{\sigma}\sigma 2^{-n} + \widetilde{\sigma} r\epsilon + \widetilde{\sigma} 2^{-n} + \widetilde{q}\epsilon)$ in Lemma 8.4.1.

Lemma 8.4.1.

$$\Pr\left[\Theta_0 \in \Omega_{\mathsf{bad}}\right] \leq \frac{0.5\sigma^2}{2^n} + \frac{\widetilde{\sigma}^2}{2^n} + \frac{q\sigma + 0.5q^2}{2^n - \sigma} + \frac{\widetilde{\sigma}(\sigma+3)}{2^n} + (\sigma + 2\widetilde{\sigma})r\epsilon + \widetilde{q}\epsilon.$$

Proof. We bound the probability of getting a bad transcript as follows:

$$\begin{aligned}
\Pr\left[\Theta_0 \in \Omega_{\mathsf{bad}}\right] &= \Pr\left[\mathtt{EEcoll} \vee \mathtt{ETcoll} \vee \mathtt{DEcoll} \vee \mathtt{DEFcoll} \vee \mathtt{DDFcoll}\right] \\
&\leq \Pr\left[\mathtt{EEcoll}\right] + \Pr\left[\mathtt{ETcoll} \mid \neg\mathtt{EEcoll}\right] \\
&+ \Pr\left[\mathtt{DEcoll} | \neg(\mathtt{EEcoll} \vee \mathtt{ETcoll})\right] \\
&+ \Pr\left[\mathtt{DEFcoll} | \neg(\mathtt{EEcoll} \vee \mathtt{ETcoll} \vee \mathtt{DEcoll})\right] \\
&+ \Pr\left[\mathtt{DDFcoll} | \neg(\mathtt{EEcoll} \vee \mathtt{ETcoll} \vee \mathtt{DEcoll} \vee \mathtt{DEFcoll})\right]
\end{aligned}$$

BOUNDING $\Pr\left[\mathtt{EEcoll}\right]$: Let P and Q be a colliding pair. We bound the probability in two cases:

Case 1: The colliding pair is a non-local pair — As λ is a LIXU hash function, we can bound the probability of this case by 2^{-n}. Further we have at most $\sigma(\sigma-1)/2$ many such pairs. Hence

$$\Pr\left[\mathtt{EEcoll} \wedge \text{Case 1}\right] \leq \frac{\sigma^2}{2^{n+1}}.$$

Case 2: The colliding pair is a local pair — This means that the colliding pair must belong to a single message or associated data. We can have two cases: (2.1) The colliding pair belongs to the i-th message; (2.2) The colliding pair belongs to the i-th associated data. Without loss of generality we assume case (2.1). Let $n_i = \lfloor m_i/r \rfloor$ and $r_i = m_i - n_i r$. As λ is an (ϵ, r)-LIXU hash function, we can bound the probability of a fixed pair by ϵ. Hence

$$\begin{aligned}
\Pr\left[\mathtt{EEcoll} \wedge \text{Case 2.1}\right] &\leq \sum_{i=1}^{q}\sum_{j=1}^{n_i}\binom{r}{2}\epsilon + \sum_{i=1}^{q}\binom{r_i}{2}\epsilon \\
&\overset{1}{\leq} r'\epsilon\sum_{i=1}^{q} r \cdot n_i + \sum_{i=1}^{q}\binom{r_i}{2}\epsilon \\
&\overset{2}{\leq} r'\mu\epsilon - \left(r'\epsilon\sum_{i=1}^{q} r_i - \sum_{i=1}^{q}\binom{r_i}{2}\epsilon\right) \\
&\overset{3}{\leq} r'\mu\epsilon.
\end{aligned}$$

Here $r' = (r-1)/2$; 1 to 2 follows from $rn_i = m_i - r_i$ and $\sum m_i = \mu$; 2 to 3 follows from the fact that $(r-1)r_i \geq r_i(r_i - 1)$ for all i. Similarly the probability in case (2.2) can be bounded to $r'\nu\epsilon$. Hence the total probability in case 2 is bounded by $r'\sigma\epsilon$.

Combining case 1 and 2, we have

$$\Pr\left[\texttt{EEcoll}\right] \leq \frac{\sigma^2}{2^{n+1}} + r\sigma\epsilon.$$

BOUNDING $\Pr\left[\texttt{ETcoll} \mid \neg\texttt{EEcoll}\right]$: This event bounds the probability that the checksum output block collides with some previous output block given that no other input/output collision occurred. At this instant at most $\sigma = \mu + \nu$ many points are defined for Π, whence we can have at most $q\sigma + q^2/2$ many possible colliding pairs. So we have,

$$\Pr\left[\texttt{ETcoll} \mid \neg\texttt{EEcoll}\right] \leq \frac{q\sigma}{2^n - \sigma} + \frac{q^2}{2(2^n - \sigma)}.$$

Bounding $\Pr\left[\texttt{DEcoll}|\neg(\texttt{EEcoll} \vee \texttt{ETcoll})\right]$: This event bounds the probability that some intermediate output ($\widetilde{\mathsf{Y}}_{(i,j)}$) or input ($\widetilde{\mathsf{U}}_{(i,j)}$) input block non-trivially belongs in O or I, respectively. First, suppose $\widetilde{\mathsf{Y}}_{(i,j)}$ belongs to O for some $(i,j) \in [\widetilde{q}] \times [\widetilde{m}_i]$. Now as in case of EEcoll above: i) the colliding encryption block can either be a non-local pair of (i,j), in which case we bound the probability to at most $\sigma/2^n$; or ii) the colliding encryption block can be a local pair of (i,j), in which case we bound the probability to at most $r\epsilon$. Since there are at most $\widetilde{\mu}$ many decryption ciphertext blocks, the probability that there exists $(i,j) \in [\widetilde{q}] \times [\widetilde{m}_i]$, such that $\widetilde{\mathsf{Y}}_{(i,j)}$ belongs in O non-trivially is bounded by $\widetilde{\mu}\sigma/2^n + \widetilde{\mu}r\epsilon$. Similarly, one can bound the probability that there exists $(i,j) \in [\widetilde{q}] \times [\widetilde{a}_i]$, such that $\widetilde{\mathsf{U}}_{(i,j)}$ belongs in I non-trivially to at most $\widetilde{\nu}\sigma/2^n + \widetilde{\nu}r\epsilon$. Combining the two cases, we have

$$\Pr\left[\texttt{DEcoll}|\neg(\texttt{EEcoll} \vee \texttt{ETcoll})\right] \leq \frac{\widetilde{\sigma}\sigma}{2^n} + \widetilde{\sigma}r\epsilon.$$

Bounding $\Pr\left[\texttt{DEFcoll}|\neg(\texttt{EEcoll} \vee \texttt{ETcoll} \vee \texttt{DEcoll})\right]$: This case bounds the probability that some decryption checksum input block collides with some encryption input block with a restriction that the decryption input blocks must all be defined. In this case we have $\widetilde{\mathsf{X}}_{(i,\oplus)} = \sum_{j=1}^{\widetilde{m}_i} \widetilde{\mathsf{M}}_{(i,j)} \oplus \lambda_3(\mathsf{L}, \widetilde{\mathsf{N}}, \widetilde{m}_i)$. Now we can have two cases: i) the checksum block collides with $\mathsf{X}_{(i',\oplus)}$, which can be generously bounded by ϵ (assuming $(\widetilde{\mathsf{N}}_i, \widetilde{m}_i)$ and $(N_{i'}, m_{i'})$ are local pair); and ii) the checksum block collides with any other $\mathsf{X}_{(i',j')}$, in which case we bound the probability by 2^{-n} using the 1-universal property of λ. In summary, we have

$$\Pr\left[\texttt{DEFcoll}|\neg(\texttt{EEcoll} \vee \texttt{ETcoll} \vee \texttt{DEcoll})\right] \leq \frac{\widetilde{q}}{2^n} + \widetilde{q}\epsilon.$$

Bounding $\Pr[\texttt{DDFcoll}|\neg(\texttt{EEcoll} \vee \texttt{ETcoll} \vee \texttt{DEcoll} \vee \texttt{DEFcoll})]$: This case simply bound the probability that there is no collisions within a decryption query. Let us fix a decryption query index $i \in \widetilde{q}$. Then the first case is bounded by at most $\widetilde{m}_i 2^{-n}$; the second case is bounded by at most $\widetilde{a}_i 2^{-n}$; the third case is bounded by at most $\widetilde{m}_i(\widetilde{m}_i - r)2^{-n} + \widetilde{m}_i r\epsilon$. On combining the three cases, we have

$$\Pr[\texttt{DDFcoll}|\neg(\texttt{EEcoll} \vee \texttt{ETcoll} \vee \texttt{DEcoll} \vee \texttt{DEFcoll})] \leq \frac{2\widetilde{\sigma}}{2^n} + \frac{\widetilde{\sigma}^2}{2^n} + \widetilde{\sigma} r\epsilon.$$

The result follows by combining all the individual bounds above, and simplifying using $\widetilde{q} \leq \widetilde{\sigma}$. □

RATIO OF INTERPOLATION PROBABILITIES: Fix an arbitrary transcript $\omega \in \Omega \setminus \Omega_{\mathsf{bad}}$. In Lemma 8.4.2, we show that the ratio of real to ideal interpolation probabilities for ω is at least $(1 - O(\widetilde{q}2^{-n}))$.

Lemma 8.4.2. *For any $\omega \in \Omega \setminus \Omega_{\mathsf{bad}}$, we have*

$$\frac{\Pr[\Theta_1 = \omega]}{\Pr[\Theta_0 = \omega]} \geq \left(1 - \frac{\widetilde{q}}{2^n - \sigma}\right).$$

Proof. In $\mathcal{O}_0$, for encryption phase: first $\mu + q$ ciphertext and tag outputs are sampled in with replacement fashion, followed by the sampling of ν output blocks in without replacement manner from a subset of $\mathbb{B}$ of size $(2^n - \mu)$ (Π is already defined on μ many input-output blocks); for decryption phase the oracle always returns $\perp$. Hence

$$\mathsf{ip}_{\mathcal{O}_0}[\omega] = \frac{1}{(2^n)^{\mu+q}(2^n - \mu)_\nu}. \tag{8.9}$$

In $\mathcal{O}_1$, for any ω we denote the encryption tuples by ω_e and the decryption tuples by ω_d. Then we have

$$\begin{aligned}
\Pr[\Theta_1 = \omega] &= \Pr_{\mathcal{O}_1}[\omega_e, \omega_d] \\
&= \Pr_{\mathcal{O}_1}[\omega_e] \times \Pr_{\mathcal{O}_1}[\omega_d \mid \omega_e] \\
&= \Pr_{\mathcal{O}_1}[\omega_e] \times \left(1 - \Pr_{\mathcal{O}_1}[\neg\omega_d \mid \omega_e]\right)
\end{aligned} \tag{8.10}$$

where $\omega_d = (\widetilde{\mathsf{N}}_i, \widetilde{\mathsf{A}}_i, \widetilde{\mathsf{C}}_i, \widetilde{\mathsf{T}}_i, \perp)_{i \in [\widetilde{q}]}$, as the ideal oracle always returns $\perp$ on decryption queries. Note that, we have slightly abused the notation to use $\neg\omega_d$ as the event that: for some $i \in [\widetilde{q}]$ the i-th decryption query successfully decrypts.

For encryption tuples, exactly $\mu + \nu + q$ many calls are made to Π: one for each of the μ message blocks, one for each of the ν associated data blocks; and one for each of the q

tags. As $\sigma = \mu + \nu$ we have

$$\Pr_{\mathcal{O}_1}[\omega_e] = \frac{1}{(2^n)_{\sigma+q}}. \tag{8.11}$$

Now we upper bound the probability of $\neg\omega_d$. It is enough to bound the probability for a fixed index $i \in [\widetilde{q}]$ corresponding to a successful decryption query. By union bound, the probability of $\neg\omega_d$ is at most $\widetilde{q}$ times more than the single forgery probability. For simplicity, we bound the probability for $i \in [\widetilde{q}] \setminus (\widetilde{\mathcal{I}} \cup \widetilde{\mathcal{J}})$. For $i \in \widetilde{\mathcal{I}} \cup \widetilde{\mathcal{J}}$, we get similar bounds. Note that, the adversary succeeds in forgery if the adversary somehow makes the i-th query in such a way that the following equation holds.

$$\Pi(\widetilde{\mathsf{X}}_{(i,\oplus)}) \oplus \widetilde{\mathsf{Y}}_{(i,\oplus)} = 0. \tag{8.12}$$

Now based on the i-th decryption query we can have different cases:

Case 1: $\exists j \in [q],\ \widetilde{\mathsf{N}}_i = \mathsf{N}_j$ — Based on $\widetilde{\mathsf{C}}_i$ we can have two subcases:

Subcase 1.1: $\widetilde{\mathsf{C}}_i = \mathsf{C}_{i'}^{[\widetilde{m}^i]}$ — In this case $\widetilde{\mathsf{M}}_{(i,\oplus)}$ is pre-determined as $\oplus_{k=1}^{\widetilde{m}_i}\mathsf{M}_{(i',k)}$. Suppose $\widetilde{m}_i = m_{i'}$, i.e. the two ciphertexts are exactly the same. Then we must have some j such that $\widetilde{\mathsf{A}}_{(i,j)} \neq \mathsf{A}_{(i',j)}$, otherwise i-th decryption query is a duplicate of i'-th encryption query. Since the transcript is good, $\widetilde{\mathsf{U}}_{(i,j)}$ is fresh. So, by exploiting the conditional randomness of Π we can bound the probability to at most $1/(2^n - \sigma)$. Now suppose $\widetilde{\mathsf{C}}_i$ is a proper prefix of $\mathsf{C}_{i'}$, then we must have a fresh $\widetilde{\mathsf{X}}_{(i,\oplus)}$ (as the transcript is good), whence we can again bound the probability to at most $1/(2^n - \sigma)$.

The two cases discussed above are mutually exclusive, hence the probability that the adversary forges in subcase 2.1 is at most $1/(2^n - \sigma)$.

Subcase 1.2: $\widetilde{\mathsf{C}}_i \neq \mathsf{C}_{i'}^{[\widetilde{m}^i]}$ — In this case there exists at least one $k \in [\widetilde{m}_i]$ such that $\widetilde{\mathsf{C}}_{(i,k)} \neq \mathsf{C}_{(i',k)}$. We consider first such block $\widetilde{\mathsf{C}}_{(i,k)}$. Since the transcript is good, we must have a fresh $\widetilde{\mathsf{Y}}_{(i,k)}$. Observe that we can rewrite Eq. (8.12) as

$$\Pi^{-1}(\widetilde{\mathsf{Y}}_{(i,k)}) = \Pi^{-1}(\widetilde{\mathsf{Y}}_{(i,\oplus)}) \oplus \delta \oplus \lambda_3(\mathsf{L}, \widetilde{\mathsf{N}}_i, \widetilde{m}_i),$$

where δ denotes the checksum of all the blocks of i-th query, except the k-th block. Again, by conditioning on all points except $\widetilde{\mathsf{Y}}_{(i,k)}$, we bound the probability of this case by $1/(2^n - \sigma)$.

Case 2: $\forall j \in [q],\ \widetilde{\mathsf{N}}_i \neq \mathsf{N}_j$*:* In this case $\widetilde{\mathsf{N}}_i$ has not been used for mask generation till now. Without loss of generality, we assume that $\widetilde{m}_i \geq 1$. Since the transcript is good, we must have a fresh $\widetilde{\mathsf{Y}}_{(i,1)}$. Now using a similar line of argument as in sub case 1.2 we bound the probability of this event by $1/(2^n - \sigma)$.

Note that, case 1.1, case 1.2 and case 2 are all mutually exclusive so we can take the maximum over all three. As $\sigma \geq a_i + a^j$ we have

$$\Pr_{\mathcal{O}_1}[\neg\omega_d \mid \omega_e] \leq \sum_{i=1}^{\widetilde{q}} \frac{1}{2^n - \sigma} \leq \frac{\widetilde{q}}{2^n - \sigma} \tag{8.13}$$

On substituting (8.11) and (8.13) in (8.10) and dividing the result by (8.9) we have

$$\begin{aligned}
\frac{\mathsf{ip}_{\mathcal{O}_1}[\omega]}{\mathsf{ip}_{\mathcal{O}_0}[\omega]} &\geq \frac{(2^n)^{\underline{\mu+q}} \cdot (2^n - \mu)^{\underline{\nu}}}{(2^n)^{\underline{\sigma+q}}} \left(1 - \frac{\widetilde{q}}{2^n - \sigma}\right) \\
&\geq \frac{(2^n)^{\underline{q}}}{(2^n - \sigma)^{\underline{q}}} \left(1 - \frac{\widetilde{q}}{2^n - \sigma}\right) \\
&\geq \left(1 - \frac{\widetilde{q}}{2^n - \sigma}\right).
\end{aligned}$$

□

We get the desired result using Lemma 8.4.1, Lemma 8.4.2 and Corollary 2.2.2.

8.4.3 Proof of Lemma 8.3.5 and Lemma 8.3.6

The proofs of these lemmata are quite similar. We give the proof for Lemma 8.3.5, while the proof for Lemma 8.3.6 can be obtained analogously.

Fix distinct $(N, i), (N', i') \in \mathcal{N} \times \mathbb{F}_{2^8}$ and $\delta \in \mathbb{B}$. Now we have two cases:

Case 1: (N, i) and (N', i') are local pairs — We know that $N = N'$ and $i \neq i'$. Hence,

$$\Pr\left[\mathsf{A1}_\Gamma(N, i) \oplus \mathsf{A1}_\Gamma(N', i') = \delta\right] = \Pr\left[\mathsf{aes1}_{\Gamma(N)}(i) \oplus \mathsf{aes1}_{\Gamma(N)}(i') = \delta\right] \leq \frac{1}{2^{96}},$$

where the last inequality follows from Lemma 8.2.4.

Case 2: (N, i) and (N', i') are not local pairs — We know that $N \neq N'$. Hence,

$$\begin{aligned}
\Pr\left[\mathsf{A1}_\Gamma(N, i) \oplus \mathsf{A1}_\Gamma(N', i') = \delta\right] &= \Pr\left[\mathsf{aes1}_{\Gamma(N)}(i) \oplus \mathsf{aes1}_{\Gamma(N')}(i') = \delta\right] \\
&\leq \Pr\left[\Gamma(N) = \text{1-}\mathtt{AESRD}^{-1}(\mathsf{aes1}_{\Gamma(N')}(i') \oplus \delta) \oplus i\right] \\
&\leq \frac{1}{2^{128}}.
\end{aligned}$$

The result follows from the bounds in case 1 and case 2. □

8.5 Software Performance

In this section, we present software implementation of GOCB instantiated with AES-128 as the underlying block cipher, and A1 and A2 as the underlying MGFs, and compare the performance of the proposed designs against the standard OCB3-AES-128. We do the benchmarking in two cases: i) sequential processing (the conventional implementation), and ii) random read access (introduced in this chapter).

IMPLEMENTATION — We reuse the optimized C code of OCB3 [119] by Krovetz. Further, we introduce minimal changes, as required, over the OCB3 code to generate the C code for $\mathsf{GOCB}_{\mathsf{AES\text{-}128,A1}}$, and $\mathsf{GOCB}_{\mathsf{AES\text{-}128,A2}}$. We also reuse the time measurement mechanism as used in [119]. In summary, we utilized the hardware support on our benchmarking platform for Intel's SSE4 vector instructions [94], Intel's `pclmulqdq` instruction for multiplication over $\mathbb{F}_{2^{128}}$, and Intel's AES-NI library [82] for operations involving AES-128 round functions. We made some simplifying assumptions for the implementations:

- message length is treated to be equal to the length of the associated data, and
- all messages and associated data are assumed to have complete blocks.

PLATFORM — We performed the benchmarking on:

- Intel Core i7-6500U "Skylake" (2.5 GHz with 64 KB L1 cache, 256 KB L2 cache, 4 MB L3 cache) with Ubuntu 16.04 LTS (kernel version 4.4),

which supports Intel's SSE4 vector instructions [94], standard AES-NI instructions [82] and `pclmuldq` [83], the carry-less multiplication instruction.

SETUP — The tests for benchmark were compiled using `gcc` version 5.4, with optimization flag `-O3`, and instruction set architecture flag `-march=native` (as instructed in [119]). All the tests were run on isolated core, after turning off processor frequency scaling and power management options such as Intel's Turbo Boost and Hyper-Threading technologies. The reference baseline performance for AES-128, using AES-NI instruction set, is presented in Table 8.5.1. All performance figures presented are throughputs, in units of cycles-per-byte (CPB).

TESTING METHODOLOGY — Following [119], we measure the CPB value for every message length from 1 to 1024 bytes, as well as 1500 and 4096 bytes. To avoid any discrepancies arising due to memory subsystem, we execute warm up runs so that the all

Table 8.5.1: Baseline CPB value of AES-128 using AES-NI in Skylake architecture.

Microarchitecture	Encryption (CPB)
Intel Core i7-6500U "Skylake"	0.66

code and data is in the cache before benchmarking begins. To capture the average cost of encryption, we use the `rdtsc` instruction — a time-stamp counter available on Skylake processors — to time encryption of the same message: i) 64 times for sequential processing; ii) 1536 times for random access. Note that, we average over a larger number of repetitions in random access case to get an appreciable measurement for a single block encryption. The median of 15 such averaged values is reported as the number of cycles. The cycles per byte value is obtained by dividing the median value by the length of the corresponding message.

Table 8.5.2: Performance comparison between sequential implementations of OCB3-AES-128, $\text{GOCB}_{\text{AES-128,A1}}$, and $\text{GOCB}_{\text{AES-128,A2}}$. The performance figures presented are throughputs, in units of cycles-per-byte (CPB).

Length	OCB3-AES-128	$\text{GOCB}_{\text{AES-128,A1}}$	$\text{GOCB}_{\text{AES-128,A2}}$
128 bytes	1.48	2.11	2.15
256 bytes	1.15	1.41	1.45
512 bytes	0.85	1.07	1.10
1024 bytes	0.75	0.90	0.93
4096 bytes	0.68	0.77	0.81

Table 8.5.3: Summary of IPI values for OCB3-AES-128, $\text{GOCB}_{\text{AES-128,A1}}$, and $\text{GOCB}_{\text{AES-128,A2}}$.

	OCB3-AES-128	$\text{GOCB}_{\text{AES-128,A1}}$	$\text{GOCB}_{\text{AES-128,A2}}$
IPI:	1.05	1.24	1.28

SEQUENTIAL PROCESSING: RESULTS AND DISCUSSION — Table 8.5.2 summarizes the CPB values for OCB3-AES-128, $\text{GOCB}_{\text{AES-128,A1}}$, and $\text{GOCB}_{\text{AES-128,A2}}$, for moderate to large message lengths, in sequential processing. Following [119], we also present IPI (Internet Performance Index) values — a weighted average CPB vlaue for usual message lengths observed over internet — in Table 8.5.3, for OCB3-AES-128, $\text{GOCB}_{\text{AES-128,A1}}$, and $\text{GOCB}_{\text{AES-128,A2}}$. As per our experiments, OCB3 with 10 rounds of AES-128 is approximately, 9% and 10% faster than OCB3 with 11 and 12 rounds, respectively, of AES-128. One may think that $\text{GOCB}_{\text{AES-128,A1}}$ and $\text{GOCB}_{\text{AES-128,A2}}$ should have similar degradation in performance. But the IPI clearly indicates that the A1 and A2 variants are much more slower, approximately 18% and 22%, respectively, than OCB3 over usual internet data. This is mainly due to the lower masking (just after encryption), which is the contentious instruction that breaks the pipelining benefits.

However, the important point to note is that for large messages (i.e. ≥ 4096 bytes), which is our main area of focus, the CPB values for $\text{GOCB}_{\text{AES-128,A1}}$ and $\text{GOCB}_{\text{AES-128,A2}}$

tend towards the CPB values for OCB3-AES-128 with 11 and 12 rounds, respectively. Specifically, we have observed that OCB3-AES-128 with 11 and 12 rounds have 0.74 and 0.81 CPBs, respectively, for 4096 bytes message. This is quite close to the CPB values achieved by $\text{GOCB}_{\text{AES-128,A1}}$ and $\text{GOCB}_{\text{AES-128,A2}}$, i.e. 0.77 and 0.81, respectively. Beyond this point the CPB values for $\text{GOCB}_{\text{AES-128,A1}}$ and $\text{GOCB}_{\text{AES-128,A2}}$ are expected to saturate and follow the CPB values of OCB3-AES-128 with 11 and 12 rounds, respectively. In other words, for messages with length ≥ 4096 bytes, $\text{GOCB}_{\text{AES-128,A1}}$ and $\text{GOCB}_{\text{AES-128,A2}}$ are only $\approx 10\%$ and $\approx 20\%$ slower than OCB3.

Table 8.5.4: Summary of CPB values for OCB3-AES-128, $\text{GOCB}_{\text{AES-128,A1}}$, and $\text{GOCB}_{\text{AES-128,A2}}$ to process a single block of data in random access.

OCB3-AES-128	$\text{GOCB}_{\text{AES-128,A1}}$	$\text{GOCB}_{\text{AES-128,A2}}$
3.93	3.12	3.29

RANDOM ACCESS: RESULTS AND DISCUSSION — To the best of our knowledge, optimized 128-bit field multiplication algorithms (using `pclmulqdq`) are very close to 5 rounds of AES (using AES-NI) in terms of performance. Thus, theoretically one would expect that $\text{GOCB}_{\text{AES-128,A1}}$, and $\text{GOCB}_{\text{AES-128,A2}}$ will have significantly better random access performance as compared to OCB3, which requires a 128-bit field multiplication.

$\text{GOCB}_{\text{AES-128,A1}}$ and $\text{GOCB}_{\text{AES-128,A2}}$ would require close to 44 and 48 cycles, respectively, for AES round calls, as 1-round AES call requires 4 cycles (latency) in Skylake. Further, roughly 4 cycles are required to create the initial masking state. So, theoretically $\text{GOCB}_{\text{AES-128,A1}}$ and $\text{GOCB}_{\text{AES-128,A2}}$ are expected to have approximately 3-3.1 and 3.2-3.3 CPB. On the other hand, OCB3 requires close to 60 cycles for field multiplication and the AES call. Apart from this approx. 3 cycles are required for gray code generation. So, OCB3 is expected to have around 3.9-4 CPB. In other words, $\text{GOCB}_{\text{AES-128,A1}}$ and $\text{GOCB}_{\text{AES-128,A2}}$ are expected to be close to 25% and 20%, respectively, better than OCB3 in terms of random access. Note that, we have ignored the overhead due to nonce processing. This is reasonable given the assumption that random read/write does not span across different messages.

Table 8.5.4 summarizes the CPB values to process an arbitrary (random access) block using OCB3-AES-128, $\text{GOCB}_{\text{AES-128,A1}}$, and $\text{GOCB}_{\text{AES-128,A2}}$. Clearly, the experimental data justifies the theoretical speedup described above.

Appendix A

Graphs and Probability Theory

A.1 Directed Edge-Labeled Graphs

DIRECTED EDGE-LABELED GRAPH: A directed edge-labeled graph is a pair $\mathcal{G} := (\mathcal{V}, \mathcal{E})$ with $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V} \times \mathcal{L}$ where $\mathcal{V}$ is the set of vertices, $\mathcal{L}$ is the set of edge labels, and $\mathcal{E}$ is the set of edges along with their corresponding labels. We write $u \xrightarrow{a} v$ to mean that $(u, v, a) \in \mathcal{E}$. By a slight abuse of notation, $\mathcal{E}$ also denotes the set of unlabeled edges. For any graph $\mathcal{G}$, we write $\mathcal{V}(\mathcal{G})$ and $\mathcal{E}(\mathcal{G})$ to denote the vertex set and edge set of $\mathcal{G}$.

An *undirected (edge-labeled) graph* is a special case of directed (edge-labeled) graph where $\mathcal{E}$ is a set of two-sets,[1], i.e., the edges (u, v) and (v, u) are denoted by a two-set $\{u, v\}$. For brevity, we extend the notation of ordered edge (u, v) for undirected graphs as well.

When viewed as the set of unlabeled edges, $\mathcal{E}$ is a multiset. Recall that μ denotes the multiplicity function 2.1.

We say, that a graph has *parallel edges* between two vertices u and v, if $(u, v) \in \mathcal{E}$ and $\mu(\mathcal{E}, (u, v)) \geq 2$. We say, that a graph has *self loop* on vertex u, if $(u, u) \in \mathcal{E}$.

SIMPLE GRAPH: A graph $\mathcal{G}$ is called simple if $\mathcal{G}$ has no parallel edges and self loops.

A graph $\mathcal{H}$ is called the *subgraph* of another graph $\mathcal{G}$ if $\mathcal{V}(\mathcal{H}) \subseteq \mathcal{V}(\mathcal{G})$ and $\mathcal{E}(\mathcal{G}) \subseteq \mathcal{E}(\mathcal{G})$.

Definition A.1.1 (Isomorphism)**.** Let $\mathcal{G}_1 = (\mathcal{V}_1, \mathcal{E}_1)$ and $\mathcal{G}_2 = (\mathcal{V}_2, \mathcal{E}_2)$ be two directed edge-labeled graphs. A function $\alpha : \mathcal{V}_1 \to \mathcal{V}_2$ is an isomorphism from $\mathcal{G}_1$ to $\mathcal{G}_2$ if α is a bijection and $(u, v, a) \in \mathcal{E}_1$ if and only if $(\alpha(u), \alpha(v), a) \in \mathcal{E}_2$. In this case, we write $G_1 \cong G_2$.

[1] A set with cardinality 2.

A.1.1 Degree of a Vertex

For a directed edge $e := (u, v)$, vertex u is called the *predecessor* of v, and v is called the *successor* of u. For all vertex v, we define two sets:

1. Predecessor set of a vertex v is $\mathsf{nbd}(* \to v) := \{u : (u, v) \in \mathcal{E}\}$.
2. Successor set of a vertex v is $\mathsf{nbd}(v \to *) := \{u : (v, u) \in \mathcal{E}\}$.

By extending the definitions to undirected graph, we see that $\mathsf{nbd}(* \to v) = \mathsf{nbd}(v \to *)$ (the order does not matter).

Definition A.1.2 (Degree)**.** In-degree $\mathbf{deg_{in}}(v)$ of a vertex v is defined as $\mathbf{deg_{in}}(v) = |\mathsf{nbd}(* \to v)|$. Similarly, out-degree $\mathbf{deg_{out}}(v)$ is defined as $\mathbf{deg_{out}}(v) = |\mathsf{nbd}(v \to *)|$.

In a directed graph $\mathcal{G}$, degree $\mathbf{deg}(v)$ of $v \in \mathcal{V}(\mathcal{G})$ is defined as the sum of $\mathbf{deg_{in}}(v)$ and $\mathbf{deg_{out}}(v)$, i.e., $\mathbf{deg}(v) = \mathbf{deg_{in}}(v) + \mathbf{deg_{out}}(v)$.

In an undirected graph $\mathcal{G}$, degree $\mathbf{deg}(v)$ of $v \in \mathcal{V}(\mathcal{G})$ is defined as $\mathbf{deg}(v) = \mathbf{deg_{in}}(v) = \mathbf{deg_{out}}(v)$.

A.1.2 Walk, Path, Cycle and Component

We define these terms for directed edge-labeled graphs. They can be analogously defined for undirected graphs as well.

Definition A.1.3 (Walk)**.** A *walk of length s* is defined as a vertex sequence $W := (W_0, \ldots, W_s)$, such that $W_{i-1} \xrightarrow{a_i} W_i$ for all $i \in [s]$. The sequence $a^s = (a_1, \ldots, a_s)$ is called the label of W. We also say that W is an a^s-walk. Any *subwalk* of $W^{(s]}$ is a contiguous subsequence $W^{[a \ldots b]}$ where $a \leq b \in (s]$.

A walk $W^{(s]}$ is said to be: a *path* if it has $s + 1$ distinct vertices, i.e., $W_i \neq W_j$ for all $i < j \in (s]$; a *cycle* if it has at most s distinct nodes and $W_s = W_0$. Note that, a self loop (or simply a loop) is also a cycle with $s = 1$.

A graph $\mathcal{G}$ is called *connected* if for any two distinct vertices $u, v \in \mathcal{V}(\mathcal{G})$, there exists a path $W^{(s]}$ such that $W_0 = u$ and $W_s = v$. Any connected subgraph $\mathcal{C}$ of $\mathcal{G}$ is called a *connected component* (or simply component).

A.2 Some Basic Results in Probability Theory

In the following, we assume reader's acquaintance with basic discrete probability theory. See Feller [69] for a primer on basic probability theory.

A.2.1 Bonferroni's Inequalities

For a finite[2] collection of events $\mathtt{A}_1, \mathtt{A}, \ldots, \mathtt{A}_n$, we define $S_1 := \sum_{i\in[n]} \Pr[\mathtt{A}_i]$, $S_2 := \sum_{i<j\in[n]} \Pr[\mathtt{A}_i \cap \mathtt{A}_j]$, and $S_k := \sum_{i_1<\cdots<i_k\in[n]} \Pr[\mathtt{A}_{i_1} \cap \cdots \cap \mathtt{A}_{i_k}]$ for all $k \in [3 \ldots n]$. Then, for odd $k \in [n]$, we have

$$\Pr\left[\bigcup_{i\in[n]} \mathtt{A}_i\right] \leq \sum_{j\in[k]} (-1)^{j-1} S_j,$$

and for even $k \in [2 \ldots n]$, we have

$$\Pr\left[\bigcup_{i\in[n]} \mathtt{A}_i\right] \geq \sum_{j\in[k]} (-1)^{j-1} S_j.$$

The equality holds for $k = n$, and the resulting identity is the principle of inclusion-exclusion.

[2] The inequalities hold for countable collections. However, for our uses just the finite case will suffice.

Bibliography

[1] Farzaneh Abed, Scott R. Fluhrer, Christian Forler, Eik List, Stefan Lucks, David A. McGrew, and Jakob Wenzel. Pipelineable on-line encryption. In *Fast Software Encryption – FSE '14, Revised Selected Papers*, pages 205–223, 2014.

[2] Martin Ågren, Martin Hell, Thomas Johansson, and Willi Meier. Grain-128a: a new version of Grain-128 with optional authentication. *IJWMC*, 5(1):48–59, 2011.

[3] Elena Andreeva, Andrey Bogdanov, Atul Luykx, Bart Mennink, Elmar Tischhauser, and Kan Yasuda. Parallelizable and authenticated online ciphers. In *Advances in Cryptology – ASIACRYPT '13, Proceedings, Part I*, pages 424–443, 2013.

[4] Elena Andreeva, Atul Luykx, Bart Mennink, and Kan Yasuda. COBRA: A parallelizable authenticated online cipher without block cipher inverse. In *Fast Software Encryption – FSE '14, Revised Selected Papers*, pages 187–204, 2014.

[5] Elena Andreeva, Andrey Bogdanov, Nilanjan Datta, Atul Luykx, Bart Mennink, Mridul Nandi, Elmar Tischhauser, and Kan Yasuda. COLM. Specification document v1, CAESAR Submission, 2016. Online: `https://competitions.cr.yp.to/round3/colmv1.pdf` (Accessed: 02 September, 2019).

[6] Elena Andreeva, Charles Bouillaguet, Orr Dunkelman, Pierre-Alain Fouque, Jonathan J. Hoch, John Kelsey, Adi Shamir, and Sébastien Zimmer. New second-preimage attacks on hash functions. *J. Cryptol.*, 29:657–696, 2016.

[7] Elena Andreeva, Guy Barwell, Ritam Bhaumik, Mridul Nandi, Dan Page, and Martijn Stam. Turning online ciphers off. *IACR Trans. Symmetric Cryptol.*, 2017(2), 2017.

[8] Subhadeep Banik, Sumit Kumar Pandey, Thomas Peyrin, Yu Sasaki, Siang Meng Sim, and Yosuke Todo. GIFT: A small present - towards reaching the limit of lightweight encryption. In *Cryptographic Hardware and Embedded Systems – CHES '17, Proceedings*, pages 321–345, 2017.

[9] Subhadeep Banik, Andrey Bogdanov, Atul Luykx, and Elmar Tischhauser. SUNDAE: small universal deterministic authenticated encryption for the internet of things. *IACR Trans. Symmetric Cryptol.*, 2018(3):1–35, 2018.

[10] M. Baugher, D. McGrew, M. Naslund, E. Carrara, and K. Norrman. The secure real-time transport protocol (SRTP). RFC 3711, Internet Engineering Task Force (IETF), 2004. Online: `https://tools.ietf.org/html/rfc3711` (Accessed: 02 May, 2020).

[11] Ray Beaulieu, Douglas Shors, Jason Smith, Stefan Treatman-Clark, Bryan Weeks, and Louis Wingers. The Simon and Speck Block Ciphers on AVR 8-Bit Microcontrollers. In *Lightweight Cryptography for Security and Privacy – LightSec '14, Revised Selected Papers*, pages 3–20, 2014.

[12] Christof Beierle, Jérémy Jean, Stefan Kölbl, Gregor Leander, Amir Moradi, Thomas Peyrin, Yu Sasaki, Pascal Sasdrich, and Siang Meng Sim. The SKINNY family of block ciphers and its low-latency variant MANTIS. In *Advances in Cryptology – CRYPTO '16, Proceedings, Part II*, pages 123–153, 2016.

[13] Mihir Bellare. Practice-oriented provable security. In *Lectures on Data Security, Modern Cryptology in Theory and Practice, Summer School*, pages 1–15, 1998.

[14] Mihir Bellare and Phillip Rogaway. The security of triple encryption and a framework for code-based game-playing proofs. In *Advances in Cryptology – EUROCRYPT '06, Proceedings*, pages 409–426, 2006.

[15] Mihir Bellare, Joe Kilian, and Phillip Rogaway. The security of cipher block chaining. In *Advances in Cryptology – CRYPTO '94, Proceedings*, pages 341–358, 1994.

[16] Mihir Bellare, Roch Guérin, and Phillip Rogaway. XOR macs: New methods for message authentication using finite pseudorandom functions. In *Advances in Cryptology – CRYPTO '95, Proceedings*, pages 15–28, 1995.

[17] Mihir Bellare, Oded Goldreich, and Hugo Krawczyk. Stateless evaluation of pseudorandom functions: Security beyond the birthday barrier. In *Advances in Cryptology – CRYPTO '99, Proceedings*, pages 270–287, 1999.

[18] Mihir Bellare, Alexandra Boldyreva, Lars R. Knudsen, and Chanathip Namprempre. Online ciphers and the hash-CBC construction. In *Advances in Cryptology – CRYPTO 2001, Proceedings*, pages 292–309, 2001.

[19] Mihir Bellare, Krzysztof Pietrzak, and Phillip Rogaway. Improved Security Analyses for CBC MACs. In *Advances in Cryptology - CRYPTO '05, Proceedings*, pages 527–545, 2005.

[20] Mihir Bellare, Viet Tung Hoang, and Sriram Keelveedhi. Cryptography from compression functions: The UCE bridge to the ROM. In *Advances in Cryptology – CRYPTO '14, Proceedings, Part I*, pages 169–187, 2014.

[21] A. Berendschot, B. den Boer, J. Boly, A. Bosselaers, J. Brandt, D. Chaum, I. Damgård, M. Dichtl, W. Fumy, M. van der Ham, C. Jansen, P. Landrock, B. Preneel, G. Roelofsen, P. de Rooij, and J Vandewalle. Final Report of RACE Integrity Primitives. *Lecture Notes in Computer Science, Springer-Verlag, 1995*, 1007, 1995.

[22] Robert Berke. On the security of iterated macs. Diploma thesis, ETH Zurich, 2003.

[23] Daniel J. Bernstein. How to stretch random functions: The security of protected counter sums. *J. Cryptol.*, 12(3):185–192, 1999.

[24] Daniel J. Bernstein. The poly1305-aes message-authentication code. In *Fast Software Encryption – FSE '05, Revised Selected Papers*, pages 32–49, 2005.

[25] Daniel J. Bernstein. Stronger Security Bounds for Wegman-Carter-Shoup Authenticators. In *Advances in Cryptology – EUROCRYPT '05, Proceedings*, pages 164–180, 2005.

[26] Daniel J Bernstein. Cache-timing attacks on AES. Online: `https://cr.yp.to/antiforgery/cachetiming-20050414.pdf`, 2005. (Accessed: 02 September, 2019).

[27] Daniel J. Bernstein. A short proof of the unpredictability of cipher block chaining. Online: `https://cr.yp.to/antiforgery/easycbc-20050109.pdf`, 2005. (Accessed: 02 September, 2019).

[28] Daniel J. Bernstein. Polynomial evaluation and message authentication. Online: `http://cr.yp.to/antiforgery/pema-20071022.pdf`, 2007. (Accessed: 02 September, '19).

[29] Ritam Bhaumik and Mridul Nandi. OleF: an inverse-free online cipher. an online SPRP with an optimal inverse-free construction. *IACR Trans. Symmetric Cryptol.*, 2016(2):30–51, 2016.

[30] Ritam Bhaumik and Mridul Nandi. Improved security for OCB3. In *Advances in Cryptology – ASIACRYPT '17, Proceedings, Part II*, pages 638–666, 2017.

[31] Eli Biham and Orr Dunkelman. A Framework for Iterative Hash Functions - HAIFA. *IACR Cryptology ePrint Archive*, 2007(278), 2007. Online: `https://eprint.iacr.org/2007/278.pdf` (Accessed: 02 May, 2020).

[32] Eli Biham and Adi Shamir. Differential cryptanalysis of des-like cryptosystems. In *Advances in Cryptology – CRYPTO '90, Proceedings*, pages 2–21, 1990.

[33] John Black and Martin Cochran. MAC reforgeability. In *Fast Software Encryption – FSE '09, Proceedings*, pages 345–362, 2009.

[34] John Black and Phillip Rogaway. CBC macs for arbitrary-length messages: The three-key constructions. In *Advances in Cryptology - CRYPTO '00, Proceedings*, pages 197–215, 2000.

[35] John Black and Phillip Rogaway. A block-cipher mode of operation for parallelizable message authentication. In *Advances in Cryptology – EUROCRYPT '02, Proceedings*, pages 384–397, 2002.

[36] John Black, Shai Halevi, Hugo Krawczyk, Ted Krovetz, and Phillip Rogaway. UMAC: fast and secure message authentication. In *Advances in Cryptology – CRYPTO '99, Proceedings*, pages 216–233, 1999.

[37] Manuel Blum and Silvio Micali. How to generate cryptographically strong sequences of pseudo random bits. In *Foundations of Computer Science – FOCS '82, Proceedings*, pages 112–117, 1982.

[38] Andrey Bogdanov, Lars R. Knudsen, Gregor Leander, Christof Paar, Axel Poschmann, Matthew J. B. Robshaw, Yannick Seurin, and C. Vikkelsoe. PRESENT: an ultra-lightweight block cipher. In *Cryptographic Hardware and Embedded Systems – CHES '07, Proceedings*, pages 450–466, 2007.

[39] Alexandra Boldyreva and Nut Taesombut. Online encryption schemes: New security notions and constructions. In *Topics in Cryptology – CT-RSA '04, Proceedings*, pages 1–14, 2004.

[40] Lilian Bossuet, Nilanjan Datta, Cuauhtemoc Mancillas-López, and Mridul Nandi. ELmD: A pipelineable authenticated encryption and its hardware implementation. *IEEE Trans. Computers*, 65(11):3318–3331, 2016.

[41] Charles Bouillaguet and Pierre-Alain Fouque. Practical Hash Functions Constructions Resistant to Generic Second Preimage Attacks Beyond the Birthday Bound, Submitted for publication, (2010).

[42] CAESAR. Competition for Authenticated Encryption: Security, Applicability, and Robustness. Online: `https://competitions.cr.yp.to/caesar.html`, 2014. (Accessed: 02 May, 2020).

[43] Larry Carter and Mark N. Wegman. Universal classes of hash functions. *J. Comput. Syst. Sci.*, 18(2):143–154, 1979.

[44] Avik Chakraborti, Nilanjan Datta, Mridul Nandi, and Kan Yasuda. Beetle family of lightweight and secure authenticated encryption ciphers. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2018(2):218–241, 2018.

[45] Debrup Chakraborty and Palash Sarkar. A general construction of tweakable block ciphers and different modes of operations. *IEEE Trans. Information Theory*, 54(5):1991–2006, 2008.

[46] Debrup Chakraborty, Sebati Ghosh, and Palash Sarkar. A fast single-key two-level universal hash function. *IACR Trans. Symmetric Cryptol.*, 2017(1):106–128, 2017.

[47] Donghoon Chang and Mridul Nandi. A Short Proof of the PRP/PRF Switching Lemma. *IACR Cryptology ePrint Archive*, 2008(78), 2008. Online: `https://eprint.iacr.org/2008/78.pdf` (Accessed: 02 May, 2020).

[48] Shan Chen and John P. Steinberger. Tight security bounds for key-alternating ciphers. In *Advances in Cryptology – EUROCRYPT '14, Proceedings*, pages 327–350, 2014.

[49] Benoît Cogliati and Yannick Seurin. EWCDM: an efficient, beyond-birthday secure, nonce-misuse resistant MAC. In *Advances in Cryptology – CRYPTO '16, Proceedings*, pages 121–149, 2016.

[50] Jean-Sébastien Coron, Yevgeniy Dodis, Cécile Malinaud, and Prashant Puniya. Merkle-Damgård Revisited: How to Construct a Hash Function. In *Advances in Cryptology – CRYPTO '05, Proceedings*, pages 430–448, 2005.

[51] Jean-Sébastien Coron, Jacques Patarin, and Yannick Seurin. The random oracle model and the ideal cipher model are equivalent. In *Advances in Cryptology – CRYPTO '08, Proceedings*, pages 1–20, 2008.

[52] Joan Daemen and Vincent Rijmen. *The Design of Rijndael: AES - The Advanced Encryption Standard*. Information Security and Cryptography. Springer, 2002.

[53] Joan Daemen and Vincent Rijmen. Understanding two-round differentials in AES. In *Security and Cryptography for Networks – SCN '06, Proceedings*, pages 78–94, 2006.

[54] Wei Dai, Viet Tung Hoang, and Stefano Tessaro. Information-theoretic indistinguishability via the chi-squared method. In *Advances in Cryptology – CRYPTO '17, Proceedings, Part III*, pages 497–523, 2017.

[55] Ivan Damgård. A design principle for hash functions. In *Advances in Cryptology – CRYPTO '89, Proceedings*, pages 416–427, 1989.

[56] Nilanjan Datta and Mridul Nandi. ELmE: A misuse resistant parallel authenticated encryption. In *Information Security and Privacy – ACISP '14, Proceedings*, pages 306–321, 2014.

[57] Nilanjan Datta, Avijit Dutta, Mridul Nandi, and Goutam Paul. Double-block hash-then-sum: A paradigm for constructing bbb secure prf. *IACR Trans. Symmetric Cryptol.*, 2018(3):36–92, 2018.

[58] Nilanjan Datta, Avijit Dutta, Mridul Nandi, and Kan Yasuda. Encrypt or decrypt? to make a single-key beyond birthday secure nonce-based MAC. In *Advances in Cryptology – CRYPTO '18, Proceedings, Part I*, pages 631–661, 2018.

[59] R.D. Dean. *Formal Aspects of Mobile Code Security*. PhD thesis, Princeton University, 1999.

[60] Whitfield Diffie and Martin E. Hellman. New Directions in Cryptography. *IEEE Trans. Information Theory*, 22(6):644–654, 1976.

[61] Christoph Dobraunig, Maria Eichlseder, Florian Mendel, and Martin Schläffer. Ascon. Specification document v1.2, CAESAR Submission, 2016. Online: `https://competitions.cr.yp.to/round3/asconv12.pdf` (Accessed: 02 September, 2019).

[62] Yevgeniy Dodis, Rosario Gennaro, Johan Håstad, Hugo Krawczyk, and Tal Rabin. Randomness extraction and key derivation using the CBC, cascade and HMAC modes. In *Advances in Cryptology – CRYPTO '04, Proceedings*, pages 494–510, 2004.

[63] Avijit Dutta, Ashwin Jha, and Mridul Nandi. A new look at counters: Don't run like marathon in a hundred meter race. *IEEE Trans. Computers*, 66(11):1851–1864, 2017.

[64] Avijit Dutta, Mridul Nandi, and Suprita Talnikar. Beyond birthday bound secure MAC in faulty nonce model. In *Advances in Cryptology – EUROCRYPT '19, Proceedings, Part I*, pages 437–466, 2019.

[65] William F. Ehrsam, Carl H. W. Meyer, John L. Smith, and Walter L. Tuchman. Message Verification and Transmission Error Detection by Block Chaining. Patent 4074066, USPTO, 1976.

[66] Peter Elias. Minimum times and memories needed to compute the values of a function. *J. Comput. Syst. Sci.*, 9:196–212, 1974.

[67] Peter Elias. Universal codeword sets and representations of the integers. *IEEE Trans. Information Theory*, 21(2):194–203, 1975.

[68] Robert M. Fano. The transmission of information. Technical Report 65, Research Laboratory of Electronics at MIT, 1949. Online: `https://hcs64.com/files/fano-tr65-ocr-only.pdf` (Accessed: 02 May, 2020).

[69] William Feller. *An Introduction to Probability Theory and Its Applications*. Wiley, 1970.

[70] Ewan Fleischmann, Christian Forler, and Stefan Lucks. Mcoe: A family of almost foolproof on-line authenticated encryption schemes. In *Fast Software Encryption – FSE '12, Revised Selected Papers*, pages 196–215, 2012.

[71] Christian Forler, Eik List, Stefan Lucks, and Jakob Wenzel. POEx: A beyond-birthday-bound-secure on-line cipher. ArcticCrypt '16, 2016. Online: `https://www.researchgate.net/publication/299565944_POEx_A_Beyond-Birthday-Bound-Secure_On-Line_Cipher` (Accessed: 02 May, 2020).

[72] Christian Forler, Eik List, Stefan Lucks, and Jakob Wenzel. Poex: A beyond-birthday-bound-secure on-line cipher. *Cryptogr. Commun.*, 10(1):177–193, 2018.

[73] Aviezri S. Fraenkel and Shmuel T. Klein. Robust universal complete codes for transmission and compression. *Discrete Applied Mathematics*, 64:31–55, 1996.

[74] Peter Gazi, Krzysztof Pietrzak, and Michal Rybár. The exact security of PMAC. *IACR Trans. Symmetric Cryptol.*, 2016(2):145–161, 2016.

[75] Edgar N. Gilbert, F. Jessie MacWilliams, and Neil J. A. Sloane. Codes which detect deception. *Bell System Technical Journal*, 53:405–424, 1974.

[76] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions (extended abstract). In *Annual Symposium on Foundations of Computer Science – FOCS '84, Proceedings*, pages 464–479, 1984.

[77] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. On the cryptographic applications of random functions. In *Advances in Cryptology – CRYPTO '84, Proceedings*, pages 276–288, 1984.

[78] Shafi Goldwasser and Mihir Bellare. Lecture Notes on Cryptography. Online: `https://cseweb.ucsd.edu/~mihir/papers/gb.pdf`, 2008. (Accessed: 02 May, 2020).

[79] Shafi Goldwasser and Silvio Micali. Probabilistic encryption and how to play mental poker keeping secret all partial information. In *ACM Symposium on Theory of Computing – STOC '82, Proceedings*, pages 365–377, 1982.

[80] Sergey Gorbunov and Charles Rackoff. On the security of cipher block chaining message authentication code. Online: `https://cs.uwaterloo.ca/~sgorbuno/publications/securityOfCBC.pdf`, 2016. (Accessed: 02 September, 2019).

[81] Robert Granger, Philipp Jovanovic, Bart Mennink, and Samuel Neves. Improved masking for tweakable blockciphers with applications to authenticated encryption. In *Advances in Cryptology – EUROCRYPT '16, Proceedings, Part I*, pages 263–293, 2016.

[82] Shay Gueron. Intel® Advanced Encryption Standard (AES) new instructions set. White paper, Intel Corporation, 2010. Online: `https://www.intel.com/content/dam/doc/white-paper/advanced-encryption-standard-new-instructions-set-paper.pdf` (Accessed: 02 May, 2020).

[83] Shay Gueron and Michael E. Kounavis. Intel® carry-less multiplication instruction and its usage for computing the GCM mode. White paper, Intel Corporation, 2011. Online: `https://www.intel.com/content/dam/www/public/us/en/documents/white-papers/carry-less-multiplication-instruction-in-gcm-mode-paper.pdf` (Accessed: 02 May, 2020).

[84] Shay Gueron and Yehuda Lindell. GCM-SIV: full nonce misuse-resistant authenticated encryption at under one cycle per byte. In *ACM Conference on Computer and Communications Security - CCS '15, Proceedings*, pages 109–119, 2015.

[85] Chun Guo and Lei Wang. Revisiting key-alternating feistel ciphers for shorter keys and multi-user security. *IACR Cryptology ePrint Archive*, 2018(816), 2018. Online: `https://eprint.iacr.org/2018/816` (Accessed: 02 May, 2020).

[86] Jian Guo, Jérémy Jean, Nicky Mouha, and Ivica Nikolic. More rounds, less security? *IACR Cryptology ePrint Archive*, 2015(484), 2015. Online: `https://eprint.iacr.org/2015/484` (Accessed: 02 May, 2020).

[87] Shai Halevi and Hugo Krawczyk. MMH: software message authentication in the gbit/second rates. In *Fast Software Encryption – FSE '97, Proceedings*, pages 172–189, 1997.

[88] Shai Halevi and Phillip Rogaway. A tweakable enciphering mode. In *Advances in Cryptology – CRYPTO '03, Proceedings*, pages 482–499, 2003.

[89] Shai Halevi and Phillip Rogaway. A parallelizable enciphering mode. In *Topics in Cryptology – CT-RSA '04, Proceedings*, pages 292–304, 2004.

[90] Viet Tung Hoang and Stefano Tessaro. Key-alternating ciphers and key-length extension: Exact bounds and multi-user security. In *Advances in Cryptology – CRYPTO '16, Proceedings, Part I*, pages 3–32, 2016.

[91] Viet Tung Hoang and Stefano Tessaro. The multi-user security of double encryption. In *Advances in Cryptology – EUROCRYPT '17, Proceedings, Part II*, pages 381–411, 2017.

[92] David A. Huffman. A method for the construction of minimum-redundancy codes. *IRE, Proceedings*, 40:1098–1101, 1952.

[93] K Igoe and J Solinas. AES Galois Counter Mode for the Secure Shell Transport Layer Protocol. RFC 5647, Internet Engineering Task Force (IETF), 2009. Online: `https://tools.ietf.org/html/rfc5647` (Accessed: 02 May, 2020).

[94] *Intel® SSE4 (AES) Programming Reference*. Intel Corporation, 2007.

[95] ISO/IEC. Information Technology – Security Techniques – Message Authentication Codes (MACs) – Part 1: Mechanisms Using A Block Cipher. International Standard ISO/IEC 9797-1:2011, International Organization for Standardization, 2011.

[96] ISO/IEC. Information technology – Digital compression and coding of continuous-tone still images: JPEG File Interchange Format (JFIF). International Standard ISO/IEC 10918-5:2013, International Organization for Standardization, 2013.

[97] Tetsu Iwata and Kaoru Kurosawa. OMAC: One-Key CBC MAC. In *Fast Software Encryption - FSE '03, Revised Papers*, pages 129–153, 2003.

[98] Tetsu Iwata and Kaoru Kurosawa. Stronger Security Bounds for OMAC, TMAC, and XCBC. In *Progress in Cryptology – INDOCRYPT '03, Proceedings*, pages 402–415, 2003.

[99] Tetsu Iwata and Kazuhiko Minematsu. Stronger security variants of GCM-SIV. *IACR Trans. Symmetric Cryptol.*, 2016(1):134–157, 2016.

[100] Goce Jakimoski and K. P. Subbalakshmi. On efficient message authentication via block cipher design techniques. In *Advances in Cryptology – ASIACRYPT '07, Proceedings*, pages 232–248, 2007.

[101] Éliane Jaulmes and Reynald Lercier. FRMAC, a proc. fast randomized message authentication code. *IACR Cryptology ePrint Archive*, 2004(166), 2004. Online: `https://eprint.iacr.org/2004/166.pdf` (Accessed: 02 May, 2020).

[102] Éliane Jaulmes, Antoine Joux, and Frédéric Valette. On the Security of Randomized CBC-MAC Beyond the Birthday Paradox Limit: A New Construction. In *Fast Software Encryption – FSE '02, Proceedings*, pages 237–251, 2002.

[103] Jérémy Jean, Ivica Nikolic, and Thomas Peyrin. Tweaks and keys for block ciphers: The TWEAKEY framework. In *Advances in Cryptology – ASIACRYPT '14, Proceedings, Part II*, pages 274–288, 2014.

[104] Ashwin Jha and Mridul Nandi. Revisiting Structure Graphs: Applications to CBC-MAC and EMAC. *J. Mathematical Cryptol.*, 10(3–4):157–180, 2016.

[105] Ashwin Jha and Mridul Nandi. On rate-1 and beyond-the-birthday bound secure online ciphers using tweakable block ciphers. *Cryptography and Communications*, 10(5):731–753, 2018.

[106] Ashwin Jha and Mridul Nandi. A survey on applications of H-technique: Revisiting security analysis of PRP and PRF. *IACR Cryptology ePrint Archive*, 2018 (1130), 2018. Online: `https://eprint.iacr.org/2018/1130` (Accessed: 02 May, 2020).

[107] Ashwin Jha and Mridul Nandi. Tight security of cascaded LRW2. *J. Cryptol.*, 2020.

[108] Ashwin Jha, Eik List, Kazuhiko Minematsu, Sweta Mishra, and Mridul Nandi. XHX - A framework for optimally secure tweakable block ciphers from classical block ciphers and universal hashing. In *Progress in Cryptology – LATINCRYPT '17, Revised Selected Papers*, pages 207–227, 2017.

[109] Ashwin Jha, Avradip Mandal, and Mridul Nandi. On the exact security of message authentication using pseudorandom functions. *IACR Trans. Symmetric Cryptol.*, 2017(1):427–448, 2017.

[110] Ashwin Jha, Cuauhtemoc Mancillas-López, Mridul Nandi, and Sourav Sen Gupta. On random read access in OCB. *IEEE Trans. Information Theory*, 65(12): 8325–8344, 2019.

[111] I. Johansson and M. Westerlund. Support for reduced-size real-time transport control protocol (RTCP): Opportunities and consequences. RFC 5506, Internet Engineering Task Force (IETF), 2009. Online: `https://tools.ietf.org/html/rfc5506` (Accessed: 02 May, 2020).

[112] Charanjit S. Jutla. PRF domain extension using dags. In *Theory of Cryptography, – TCC '06, Proceedings*, pages 561–580, 2006.

[113] David Kahn. *The Codebreakers: The Comprehensive History of Secret Communication from Ancient Times to the Internet*. Scribner, 1996.

[114] Liam Keliher. Refined analysis of bounds related to linear and differential cryptanalysis for the AES. In *Advanced Encryption Standard – AES '04, Revised Selected and Invited Papers*, pages 42–57, 2004.

[115] Liam Keliher and Jiayuan Sui. Exact maximum expected differential and linear probability for two-round Advanced Encryption Standard. *IET Information Security*, 1(2):53–57, 2007.

[116] John Kelsey and Bruce Schneier. Second preimages on n-bit hash functions for much less than 2^n work. In *Advances in Cryptology – EUROCRYPT '05, Proceedings*, pages 474–490, 2005.

[117] Hugo Krawczyk. LFSR-based hashing and authentication. In *Advances in Cryptology – CRYPTO '94, Proceedings*, pages 129–139, 1994.

[118] Ted Krovetz. Message authentication on 64-bit architectures. In *Selected Areas in Cryptography – SAC '06, Revised Selected Papers*, pages 327–341, 2006.

[119] Ted Krovetz and Phillip Rogaway. The software performance of authenticated-encryption modes. In *Fast Software Encryption – FSE '11, Revised Selected Papers*, pages 306–327, 2011.

[120] Ted Krovetz and Phillip Rogaway. The OCB authenticated-encryption algorithm. RFC 7253, Internet Engineering Task Force (IETF), 2014. Online: `https://tools.ietf.org/html/rfc7253` (Accessed: 02 May, 2020).

[121] Kaoru Kurosawa and Tetsu Iwata. TMAC: Two-key cbc mac. In *Topics in Cryptology - CT-RSA '03, Proceedings*, pages 33–49, 2003.

[122] Rodolphe Lampe and Yannick Seurin. Tweakable blockciphers with asymptotically optimal security. In *Fast Software Encryption – FSE '13, Revised Selected Papers*, pages 133–151, 2013.

[123] Will Landecker, Thomas Shrimpton, and R. Seth Terashima. Tweakable blockciphers with beyond birthday-bound security. In *Advances in Cryptology – CRYPTO '12, Proceedings*, pages 14–30, 2012.

[124] J. Lennox. Encryption of header extensions in the secure real-time transport protocol (SRTP). RFC 6904, Internet Engineering Task Force (IETF), 2013. Online: `https://tools.ietf.org/html/rfc6904` (Accessed: 02 May, 2020).

[125] Moses Liskov, Ronald L. Rivest, and David A. Wagner. Tweakable block ciphers. In *Advances in Cryptology – CRYPTO '02, Proceedings*, pages 31–46, 2002.

[126] Atul Luykx, Bart Preneel, Elmar Tischhauser, and Kan Yasuda. A MAC mode for lightweight block ciphers. In *Fast Software Encryption – FSE '16, Revised Selected Papers*, pages 43–59, 2016.

[127] Mitsuru Matsui and Atsuhiro Yamagishi. A new method for known plaintext attack of FEAL cipher. In *Advances in Cryptology – EUROCRYPT '92, Proceedings*, pages 81–91, 1992.

[128] Ueli M. Maurer. Indistinguishability of Random Systems. In *Advances in Cryptology – EUROCRYPT '02, Proceedings*, pages 110–132, 2002.

[129] D McGrew and D Bailey. AES-CCM Cipher Suites for Transport Layer Security (TLS). RFC 6655, Internet Engineering Task Force (IETF), 2012. Online: `https://tools.ietf.org/html/rfc6655` (Accessed: 02 May, 2020).

[130] David A. McGrew and John Viega. The security and performance of the galois/-counter mode (GCM) of operation. In *Progress in Cryptology – INDOCRYPT '04, Proceedings*, pages 343–355, 2004.

[131] Bart Mennink. Insuperability of the standard versus ideal model gap for tweakable blockcipher security. In *Advances in Cryptology – CRYPTO '17, Proceedings, Part II*, pages 708–732, 2017.

[132] Bart Mennink. Towards tight security of cascaded LRW2. In *Theory of Cryptography – TCC '18, Proceedings, Part II*, pages 192–222, 2018.

[133] Bart Mennink and Samuel Neves. Encrypted davies-meyer and its dual: Towards optimal security using mirror theory. In *Advances in Cryptology – CRYPTO '17, Proceedings, Part III*, pages 556–583, 2017.

[134] Ralph C. Merkle. One way hash functions and DES. In *Advances in Cryptology – CRYPTO '89, Proceedings*, pages 428–446, 1989.

[135] Kazuhiko Minematsu. Improved security analysis of XEX and LRW modes. In *Selected Areas in Cryptography – SAC '06, Revised Selected Papers*, pages 96–113, 2006.

[136] Kazuhiko Minematsu. How to thwart birthday attacks against macs via small randomness. In *Fast Software Encryption – FSE '10, Proceedings*, pages 230–249, 2010.

[137] Kazuhiko Minematsu. A short universal hash function from bit rotation, and applications to blockcipher modes. In *Provable Security – ProvSec '13, Proceedings*, pages 221–238, 2013.

[138] Kazuhiko Minematsu. Parallelizable rate-1 authenticated encryption from pseudorandom functions. In *Advances in Cryptology – EUROCRYPT '14, Proceedings*, pages 275–292, 2014.

[139] Kazuhiko Minematsu and Tetsu Iwata. Tweak-length extension for tweakable blockciphers. In *Cryptography and Coding – IMACC '15, Proceedings*, pages 77–93, 2015.

[140] Kazuhiko Minematsu and Toshiyasu Matsushima. New Bounds for PMAC, TMAC, and XCBC. In *Fast Software Encryption – FSE '07, Revised Selected Papers*, pages 434–451, 2007.

[141] Kazuhiko Minematsu and Yukiyasu Tsunoo. Provably secure MACs from differentially-uniform permutations and AES-based implementations. In *Fast Software Encryption – FSE '06, Revised Selected Papers*, pages 226–241, 2006.

[142] Valérie Nachef, Jacques Patarin, and Emmanuel Volte. *Feistel Ciphers - Security Proofs and Cryptanalysis*. Springer, 2017.

[143] Yusuke Naito. Blockcipher-based macs: Beyond the birthday bound without message length. In *Advances in Cryptology – ASIACRYPT '17, Proceedings, Part III*, pages 446–470, 2017.

[144] Mridul Nandi. A simple and unified method of proving indistinguishability. In *Progress in Cryptology - INDOCRYPT '06, Proceedings*, pages 317–334, 2006.

[145] Mridul Nandi. A simple security analysis of hash-cbc and a new efficient one-key online cipher. *IACR Cryptology ePrint Archive*, 2007(158), 2007. Online: `https://eprint.iacr.org/2007/158` (Accessed: 02 May, 2020).

[146] Mridul Nandi. Two new efficient cca-secure online ciphers: MHCBC and MCBC. In *Progress in Cryptology – INDOCRYPT '08, Proceedings*, pages 350–362, 2008.

[147] Mridul Nandi. Fast and Secure CBC-type MAC Algorithms. In *Fast Software Encryption – FSE '09, Revised Selected Papers*, pages 375–393, 2009.

[148] Mridul Nandi. Improved Security Analysis for OMAC as a Pseudorandom Function. *J. Mathematical Cryptol.*, 3(2):133–148, 2009.

[149] Mridul Nandi. A Unified Method for Improving PRF Bounds for a Class of Blockcipher Based MACs. In *Fast Software Encryption – FSE '10, Revised Selected Papers*, pages 212–229, 2010.

[150] Mridul Nandi. On the minimum number of multiplications necessary for universal hash functions. In *Fast Software Encryption – FSE '14, Revised Selected Papers*, pages 489–508, 2014.

[151] Mridul Nandi and Avradip Mandal. Improved security analysis of PMAC. *J. Mathematical Cryptol.*, 2(2):149–162, 2008.

[152] NIST. Announcing the ADVANCED ENCRYPTION STANDARD (AES). FIPS 197, National Institute of Standards and Technology, U. S. Department of Commerce, 2001. Online: `https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.197.pdf` (Accessed: 02 May, 2020).

[153] Morris Dworkin (NIST). Recommendation for block cipher modes of operation – methods and techniques. NIST Special Publication 800-38A, National Institute of Standards and Technology, U. S. Department of Commerce, 2001. Online: `https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-38a.pdf` (Accessed: 02 May, 2020).

[154] Morris Dworkin (NIST). Recommendation for Block Cipher Modes of Operation: The CCM Mode for Authentication and Confidentiality. NIST Special Publication 800-38C, National Institute of Standards and Technology, U. S. Department of Commerce, 2001. Online: `https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-38c.pdf` (Accessed: 02 May, 2020).

[155] Morris Dworkin (NIST). Recommendation for Block Cipher Modes of Operation: The CMAC Mode for Authentication. NIST Special Publication 800-38B, National Institute of Standards and Technology, U. S. Department of Commerce, 2005. Online: `https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-38b.pdf` (Accessed: 02 May, 2020).

[156] NIST LwC Standardization Team. Submission requirements and evaluation criteria for the lightweight cryptography standardization process. `https://csrc.nist.gov/CSRC/media/Projects/Lightweight-Cryptography/documents/final-lwc-submission-requirements-august2018.pdf`, 2018. (Accessed: 02 September, 2019).

[157] Sangwoo Park, Soo Hak Sung, Sangjin Lee, and Jongin Lim. Improving the upper bound on the maximum differential and the maximum linear hull probability for SPN structures and AES. In *Fast Software Encryption – FSE '03, Revised Selected Papers*, pages 247–260, 2003.

[158] Jacques Patarin. *Etude des Générateurs de Permutations Pseudo-aléatoires Basés sur le Schéma du DES*. PhD thesis, Université de Paris, 1991.

[159] Jacques Patarin. The "coefficients H" technique. In *Selected Areas in Cryptography - SAC '08, Revised Selected Papers*, pages 328–345, 2008.

[160] Jacques Patarin. Introduction to mirror theory: Analysis of systems of linear equalities and linear non equalities for cryptography. *IACR Cryptology ePrint Archive*, 2010(287), 2010. Online: `https://eprint.iacr.org/2010/287` (Accessed: 02 May, 2020).

[161] Jacques Patarin. Mirror theory and cryptography. *Appl. Algebra Eng. Commun. Comput.*, 28(4):321–338, 2017.

[162] Erez Petrank and Charles Rackoff. CBC MAC for Real-Time Data Sources. *J. Cryptol.*, 13(3):315–338, 2000.

[163] Krzysztof Pietrzak. A Tight Bound for EMAC. In *Automata, Languages and Programming - ICALP '06, Proceedings, Part II*, pages 168–179, 2006.

[164] *ZIP File Format Specification*. PKWARE Inc., 2014.

[165] J. Postel. User datagram protocol. RFC 768, Internet Engineering Task Force (IETF), 1980. Online: `https://tools.ietf.org/html/rfc768` (Accessed: 02 May, 2020).

[166] Bart Preneel. Davies–Meyer. In *Encyclopedia of Cryptography and Security*. Springer-Verlag, 2011.

[167] Bart Preneel and Paul C. van Oorschot. On the security of iterated message authentication codes. *IEEE Trans. Information Theory*, 45(1):188–199, 1999.

[168] Gordon Procter. A note on the CLRW2 tweakable block cipher construction. *IACR Cryptology ePrint Archive*, 2014(111), 2014. Online: `https://eprint.iacr.org/2014/111` (Accessed: 02 May, 2020).

[169] Michael O. Rabin and Shmuel Winograd. Fast evaluation of polynomials by rational preparation. *Communications on Pure and Applied Mathematics*, 25(4):433–458, 1972.

[170] Phillip Rogaway. Bucket hashing and its application to fast message authentication. *J. Cryptol.*, 12(2):91–115, 1999.

[171] Phillip Rogaway. Efficient instantiations of tweakable blockciphers and refinements to modes OCB and PMAC. In *Advances in Cryptology – ASIACRYPT '04, Proceedings*, pages 16–31, 2004.

[172] Phillip Rogaway and Thomas Shrimpton. Deterministic authenticated-encryption: A provable-security treatment of the key-wrap problem. In *Advances in Cryptology – EUROCRYPT '06, Proceedings*, pages 373–390, 2006.

[173] Phillip Rogaway and Haibin Zhang. Online ciphers from tweakable blockciphers. In *Topics in Cryptology - CT-RSA '11, Proceedings*, pages 237–249, 2011.

[174] Phillip Rogaway, Mihir Bellare, John Black, and Ted Krovetz. OCB: a block-cipher mode of operation for efficient authenticated encryption. In *ACM Conference on Computer and Communications Security – CCS '01, Proceedings*, pages 196–205, 2001.

[175] David Salomon. *Variable-length Codes for Data Compression*. Springer-Verlag New York, Inc., 2007.

[176] Palash Sarkar. Improving upon the TET mode of operation. In *Information Security and Cryptology – ICISC '07, Proceedings*, pages 180–192, 2007.

[177] Palash Sarkar. A new multi-linear universal hash family. *Des. Codes Cryptography*, 69(3):351–367, 2013.

[178] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson. RTP: A transport protocol for real-time applications. RFC 3550, Internet Engineering Task Force (IETF), 2003. Online: `https://tools.ietf.org/html/rfc3550` (Accessed: 02 May, 2020).

[179] Claude Elwood Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 27:379–423, 1948.

[180] Claude Elwood Shannon. Communication theory of secrecy systems. *The Bell System Technical Journal*, 28:656–715, 1949.

[181] C Shao, H Deng, R Pazhyannur, F Bari, R Zhang, and S Matsushima. IEEE 802.11 Medium Access Control (MAC) Profile for Control and Provisioning of Wireless Access Points (CAPWAP). RFC 7494, Internet Engineering Task Force (IETF), 2015. Online: `https://tools.ietf.org/html/rfc7494` (Accessed: 02 May, 2020).

[182] Victor Shoup. On fast and provably secure message authentication based on universal hashing. In *Advances in Cryptology – CRYPTO '96, Proceedings*, pages 313–328, 1996.

[183] Victor Shoup. Sequences of games: a tool for taming complexity in security proofs. *IACR Cryptology ePrint Archive*, 2004(332), 2004. Online: `https://eprint.iacr.org/2004/332` (Accessed: 02 May, 2020).

[184] Peng Wang, Dengguo Feng, and Wenling Wu. HCTR: A variable-input-length enciphering mode. In *Information Security and Cryptology – CISC '05, Proceedings*, pages 175–188, 2005.

[185] Mark N. Wegman and Larry Carter. New hash functions and their use in authentication and set equality. *J. Comput. Syst. Sci.*, 22(3):265–279, 1981.

[186] D. Whiting, R. Housley, and N. Ferguson. Counter with CBC-MAC (CCM). RFC 3610, Internet Engineering Task Force (IETF), 2003. Online: `https://tools.ietf.org/html/rfc3610` (Accessed: 02 September, 2019).

[187] S. Wigert. Sur l'ordre de grandeur du nombre des diviseurs d'un entier. *Ark. Mat. Astron. Fys.*, 3(18), 1907.

[188] Shmuel Winograd. A new algorithm for inner product. *IEEE Trans. Computers*, 17(7):693–694, 1968.

[189] Kan Yasuda. The sum of CBC MACs is a secure prf. In Josef Pieprzyk, editor, *Topics in Cryptology – CT-RSA '10, Proceedings*, pages 366–381, 2010.

[190] Kan Yasuda. A new variant of PMAC: beyond the birthday bound. In *Advances in Cryptology – CRYPTO '11, Proceedings*, pages 596–609, 2011.

[191] Kan Yasuda. PMAC with parity: Minimizing the query-length influence. In *Topics in Cryptology – CT-RSA '12, Proceedings*, pages 203–214, 2012.