On the Security of Hash Function Combiners

Vom Fachbereich Informatik der
Technischen Universität Darmstadt genehmigte

Dissertation

zur Erlangung des Grades
Doctor rerum naturalium (Dr.rer.nat.)
von

Dipl.-Inf. Anja Lehmann
geboren in Dresden

Referenten:	Dr. Marc Fischlin Prof. Dr. Yevgeniy Dodis
Tag der Einreichung:	25. Januar 2010
Tag der mündlichen Prüfung:	19. März 2010

Darmstadt, 2010
Hochschulkennziffer: D17

Erklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit – abgesehen von den in ihr ausdrücklich genannten Hilfen – selbständig verfasst habe.

Wissenschaftlicher Werdegang

Oktober 2000 – September 2002

Studium der Medieninformatik an der Technischen Universität Dresden

Oktober 2002 – August 2006

Weiterführung des Studiums im Studiengang Informatik mit Nebenfach Neuroinformatik

September 2004 – März 2005

Auslandssemester an der University of Bristol, England

seit August 2006

Wissenschaftliche Mitarbeiterin in der Emmy-Noether-Forschungsgruppe "MiniCrypt" an der Technischen Universität Darmstadt

List of Publications

[1] Christina Brzuska, Marc Fischlin, Anja Lehmann and Dominique Schröder. *Unlinkability of Sanitizable Signatures*. To appear in Public Key Cryptography (PKC) 2010, Lecture Notes in Computer Science. Springer-Verlag, 2010.

[2] Marc Fischlin, Anja Lehmann and Daniel Wagner. *Hash Function Combiners in TLS and SSL*. Topics in Cryptology – Cryptographers Track, RSA Conference (CT-RSA) 2010, Volume 5985 of Lecture Notes in Computer Science, pages 268–283. Springer-Verlag, 2010.

[3] Marc Fischlin and Anja Lehmann. *Delayed-Key Message Authentication for Streams*. Theory of Cryptography Conference (TCC) 2010, Volume 5978 of Lecture Notes in Computer Science, pages 288–305. Springer-Verlag, 2010.

[4] Anja Lehmann and Stefano Tessaro. *A Modular Design for Hash Functions: Towards Making the Mix-Compress-Mix Approach Practical*. Advances in Cryptology – Asiacrypt 2009, Volume 5912 of Lecture Notes in Computer Science, pages 364–381. Springer-Verlag, 2009.

[5] Christina Brzuska, Marc Fischlin, Anja Lehmann and Dominique Schröder. *Sanitizable Signatures: How to Partially Delegate Control for Authenticated Data.* Biometrics and Electronic Signatures — Research and Applications (BIOSIG) 2009, Volume 155 of Lecture Notes in Informatics, pages 117–129. Gesellschaft für Informatik, 2009.

[6] Christina Brzuska, Marc Fischlin, Tobias Freudenreich, Anja Lehmann, Marcus Page, Jakob Schelbert, Dominique Schröder, Florian Volk. *Security of Sanitizable Signatures Revisited*. Public Key Cryptography (PKC) 2009, Volume 5443 of Lecture Notes in Computer Science, pages 317–336. Springer-Verlag, 2009.

[7] Marc Fischlin, Anja Lehmann and Krzysztof Pietrzak. *Robust Multi-Property Combiners for Hash Functions Revisited*. International Colloquium on Automata, Languages, and Programming (ICALP) 2008, Volume 5126 of Lecture Notes in Computer Science, pages 655–666. Springer-Verlag, 2008.

[8] Marc Fischlin and Anja Lehmann. *Robust Multi-Property Combiners for Hash Functions*. Theory of Cryptography Conference (TCC) 2008, Volume 4948 of Lecture Notes in Computer Science, pages 375–392. Springer-Verlag, 2008.

[9] Marc Fischlin and Anja Lehmann. *Security-Amplifying Combiners for Hash Functions*. Advances in Cryptology—Crypto 2007, Volume 4622 of Lecture Notes in Computer Science, pages 224–243. Springer-Verlag, 2007.

[10] Daniel Dönigus, Stefan Endler, Marc Fischlin, Andreas Hülsing, Patrick Jäger, Anja Lehmann, Sergey Podrazhansky, Sebastian Schipp, Erik Tews, Sven Vowe, Matthias Walthart, Frederik Weidemann. *Security of Invertible Media Authentication Schemes Revisited*. Information Hiding 2007. Volume 4567 of Lecture Notes in Computer Science, pages 189–203. Springer-Verlag, 2007.

Acknowledgments

Many people have contributed in various ways to this thesis. First and foremost, I want to acknowledge the guidance and support of my advisor Marc Fischlin. It was my privilege and pleasure to be his first PhD student, to work with and learn from him. Marc freely shared his research ideas and guided me with ongoing encouragement and patience throughout my studies. He also gave me the opportunity to attend quite a few conferences, thereby traveling half the world. For all of this, I am deeply thankful to him.

I am also grateful to Yevgeniy Dodis for agreeing to be the co-referee of this thesis. Furthermore, I would like to thank all my collaborators, and in particular Krzysztof Pietrzak for his contributions to this work and Stefano Tessaro for the many fruitful discussions we had.

My time at the TU Darmstadt would certainly have been less enjoyable without my fellow students. Among them, I would especially like to thank my officemates Erik Dahmen and Richard Lindner for providing such a fun and friendly environment and also for proofreading parts of the thesis. I also want to thank Lucie Langer and Axel Schmidt for many relaxing coffee breaks and introducing me to all the nice spots of the city. In addition, I owe a big thank you to all my non-academic friends who accompanied me during the last years. The fun hours we spent (especially on Wednesdays) helped getting my mind off work and recharge my batteries.

Finally, I am deeply grateful to my family for providing me with endless support (and Knusperflocken) and for cheering me up, whenever I needed it. Vielen Dank, für Alles!

Anja Lehmann
Dresden, January 2010

Abstract

A hash function is an algorithm that compresses messages of arbitrary length into short digests of fixed length. If the function additionally satisfies certain security properties, it becomes a powerful tool in the design of cryptographic protocols. The most important property is collision-resistance, which requires that it should be hard to find two distinct messages that evaluate to the same hash value. When a hash function deploys secret keys, it can also be used as a pseudorandom function or message authentication code.

However, recent attacks on collision-resistant hash functions [WLF+05, WYY05, WY05, SSA+09] caused a decrease of confidence that today's candidates really have this property and have raised the question how to devise constructions that are more tolerant to cryptanalytic results. Hence, approaches like robust combiners [Her05, Her09, HKN+05] which "merge" several candidate functions into a single failure-tolerant one, are of great interest and have triggered a series of research [BB06, Pie07, CRS+07, FL07, Pie08, FLP08].

In general, a hash combiner takes two hash functions H_0, H_1 and combines them in such a way that the resulting function remains secure as long as at least one of the underlying candidates H_0 or H_1 is secure. For example, the classical combiner for collision-resistance simply concatenates the outputs of both hash functions $\mathsf{Comb}(M) = H_0(M)||H_1(M)$ in order to ensure collision-resistance as long as either of H_0, H_1 obeys the property.

However, this classical approach is complemented by two negative results: On the one hand, the combiner requires twice the output length of an ordinary hash function and this was even shown to be optimal for collision-resistance [BB06, Pie07, CRS+07, Pie08]. On the other hand, the security of the combiner does not increase with the enlarged output length, i.e., the combiner is not significantly stronger than the sum of its components [Jou04]. In this thesis we address the question if there are *security-amplifying* combiners where the combined hash function provides a higher security level than the building blocks, thus going beyond the additive limit. We show that one can indeed have such combiners and propose a solution that is essentially as efficient as the concatenated combiner.

Another issue is that, so far, hash function combiners only aim at preserving a single property such as collision-resistance or pseudorandomness. However, when hash functions are used in protocols like TLS to secure http

and email communication, they are often required to provide several properties simultaneously. We therefore introduce the notion of *robust multi-property* combiners and clarify some aspects on different definitions for such combiners. We also propose constructions that are multi-property robust in the strongest sense and provably preserve important properties such as (target) collision-resistance, one-wayness, pseudorandomness, message authentication, and indifferentiability from a random oracle.

Finally, we analyze the (ad-hoc) hash combiners that are deployed in the TLS and SSL protocols. Nowadays, both protocols are ubiquitous as they provide secure communication for a variety of applications in untrusted environments. Therein, hash function combiners are deployed to derive shared secret keys and to authenticate the final step in the key-agreement phase. As those established secret parameters are subsequently used to protect the communication, their security is of crucial importance. We therefore formally fortify the security guarantees of the TLS/SSL combiner constructions and provide the sufficient requirements on the underlying hash functions that make those combiners suitable for their respective purposes.

Zusammenfassung

Hash Funktionen verarbeiten Eingaben beliebiger Länge und bilden diese auf Zeichenketten mit kurzer, fester Länge ab. Besitzen solche Funktionen zusätzlich bestimmte Sicherheitseigenschaften, sind sie ein wichtiger Bestandteil von zahlreichen kryptographischen Protokollen. Die wohl wichtigste Eigenschaft von Hash Funktionen ist Kollisionsresistenz. Diese verlangt, dass es schwierig ist, zwei verschiedene Nachrichten zu finden, die durch die Funktion auf den selben Hashwert abgebildet werden. Setzen Hash Funktionen zudem geheime Schlüssel ein, können sie auch als Pseudozufallsfunktionen oder Message Authentication Codes dienen.

Erfolgreiche Angriffe gegen kollisionsresistente Hash Funktionen [WLF+05, WYY05, WY05, SSA+09] ließen allerdings die Frage aufkommen, wie solche Funktionen besser vor kryptanalytischen Resultaten geschützt werden können. Eine Möglichkeit stellen sogenannte *Robust Combiner* [Her05, Her09, HKN+05] dar, die verschiedene Varianten eines kryptographischen Verfahrens kombinieren, um so die gewünschte Robustheit gegen neue Angriffe zu bieten.

Im Allgemeinen besteht ein Hash Combiner aus zwei Hash Funktionen, die so zu einer Funktion zusammengesetzt werden, dass deren Sicherheit garantiert ist, solange mindestens eine der unterliegenden Funktionen sicher ist. Für die Eigenschaft der Kollisionsresistenz besteht der klassische Combiner aus dem einfachen Konkatenieren zweier Hashwerte $\mathsf{Comb}(M) = H_0(M)||H_1(M)$. Die so kombinierte Hash Funktion ist kollisionsresistent, solange mindestens eine der Funktionen H_0, H_1 diese Eigenschaft besitzt.

Der klassische Combiner für Kollisionsresistenz hat allerdings auch Nachteile: Zum einen, erfordert er eine Ausgabe, die doppelt so lang ist wie die einer einzelnen Hash Funktion [BB06, Pie07, CRS+07, Pie08]. Zum anderen steigt die Sicherheit nicht im gleichen Masse wie die Ausgabelänge, denn der Combiner ist im Wesentlichen nur so stark wie die Summe der Einzelsicherheiten [Jou04]. In dieser Arbeit betrachten wir daher die Frage, ob Combiner existieren, welche die Sicherheit beider unterliegenden Funktionen sogar verstärken können. Dabei stellen wir eine Konstruktion vor, die diese Eigenschaft erfüllt und dabei nahezu genauso effizient ist wie der klassische Ansatz.

Weiterhin wurden Hash Combiner bisher nur so konzipiert, dass sie einzelne Eigenschaften, wie z.B. Kollisionsresistenz oder Pseudozufälligkeit, erhalten. Wenn Hash Funktionen allerdings in Protokollen wie TLS eingesetzt wer-

den, müssen sie darin oft mehrere Eigenschaften gleichzeitig erfüllen. Aus diesem Grund führen wir den Begriff der *Robust Multi-Property Combiner* ein und diskutieren zunächst verschiedene Definitionsmöglichkeiten und deren Auswirkungen. Anschließend werden Konstruktionen für solche Combiner vorgestellt, die bis zu sechs wichtige Eigenschaften gleichzeitig absichern.

Im letzten Teil der Arbeit untersuchen wir die Hash Combiner die in den TLS und SSL Protokollen eingesetzt werden. Beide Protokolle ermöglichen eine sichere Kommunikation in nicht-vertrauenswürdigen Umgebungen und sind daher in zahlreichen Anwendungen zu finden. Zur Erzeugung des notwendigen Schlüsselmaterials setzen sowohl TLS als auch SSL eigene Hash Combiner ein. Da die so ausgehandelten Schlüssel anschließend die Grundlage der abgesicherten Kommunikation bilden, ist deren Sicherheit von großer Bedeutung. Aus diesem Grund analysieren wir die vorgeschlagenen Combiner Konstruktionen und zeigen, unter welchen Annahmen die gewünschte Sicherheit erreicht werden kann.

Contents

Chapter 1

Introduction

A hash function is an algorithm that compresses messages of arbitrary length into short digests of fixed length. Originally, they were used in the context of data storage, where they can provide a speed-up for searching an entry in a set of stored elements. If a hash function H additionally satisfies certain security properties, it becomes a powerful tool in the design of cryptographic protocols. The most important property is collision-resistance, which requires that it should be hard to find two distinct messages $M \neq M'$ that evaluate to the same hash value $H(M) = H(M')$. When a hash function gets keyed it can, for example, be used as a pseudorandom function, where its outputs should be indistinguishable from truly random values. In fact, hash functions are nowadays employed in a broad spectrum of cryptographic protocols, such as message authentication codes, digital signatures, encryption schemes and key-agreement in the TLS/SSL protocols.

However, recent attacks [WYY05, WY05, CR08, SSA+09] against the most widely deployed hash functions MD5 and SHA1 caused a decrease of confidence, especially concerning long-term security. Consider for instance a practical signature scheme like RSA-PSS [BR96] that follows the "hash-and-sign" paradigm, i.e., it first hashes a message of arbitrary length M and then signs the short digest $H(M)$ using a cryptographic trapdoor function like RSA. Then the ability of efficiently finding collisions on H would immediately break this signature scheme, independently of the strength of the applied trapdoor function. The threat of insecure hash functions even prompted NIST, the American (National) Institute of Standards and Technology, to announce a call for a new hash function [NIS]. Thus, in the last two years the cryptographic community came up with several proposals [LMPR08, FLS+09, GKM+09, GKK+09] for new, hopefully secure, hash functions from which one will be selected in 2012 to be the new standard hash function — SHA-3.

Robust Combiners. An independent approach to achieve hash constructions that are more tolerant to cryptanalytic results is to use so-called combiners.

That is, combining multiple (hash) functions in such a way that the resulting function remains secure as long as at least one of the underlying candidates is secure. Actually, this concept is somewhat folklore and by no means limited to cryptography. It even exists in the real physical world: For instance, in the morning of an important appointment (like a PhD defense) one might tend to set two alarm clocks instead of a single one, in case one fears that a battery dies overnight. Another example, more related to security, is the use of several and different locks to protect a valuable bike. Clearly, that complicates the work of a thief in the sense that it increases his time by a factor of k, when k locks instead of 1 need to be broken. Moreover, as long as at least one lock resists the attack, the bike cannot be stolen and its security is guaranteed.

In cryptography, the approach of using several implementations for some primitive in order to hedge one's bets against new attacks or implementation failures has been subject to research for a long time. The early work on combiners mostly considered encryption schemes and analyzed the security of multiple (cascade) encryption when it incorporates potentially untrusted ciphers [AB81, EG85, MM93]. However, the first explicit and formal studies of combiners were initiated only recently by Herzberg [Her05, Her09] and Harnik et al. [HKN+05]. Therein the authors coined the term *robust combiner* and also proposed combiner constructions for several cryptographic primitives such as one-way functions, commitment schemes or key-agreement.

Collision-Resistant Hash Combiners. For many primitives very straightforward robust combiners exist. This includes collision-resistant hash functions, where the combiner simply concatenates the outputs of two hash functions, invoked on the same message:

$$\mathsf{Comb}_{||}(M) = H_0(M)||H_1(M).$$

This classical approach provides collision-resistance as long as at least one of the two underlying hash functions is secure, since any collision on the combiner can be traced back to collisions on both candidates. On the negative side, the combiner increases the output of the hash function from n to $2n$ bits, which limits its suitability for practical applications where the output length is a crucial parameter. Yet, it was shown that the output of a (black-box) collision-resistant combiner cannot be shorter than the concatenation of both outputs [BB06, Pie07, CRS+07, Pie08]. In this light it was disappointing to learn that the (necessary) longer output length does not come with significantly higher security guarantees. Since the adversary against the concatenated combiner needs to find a message pair that collides simultaneously under both hash functions, it was expected that the provided security is clearly beyond the sum of the individual securities. Regarding our example where a bike is protected by multiple locks, this would mean that the thief has to find a single key or tool that opens all locks at the same time. However, Joux showed in [Jou04] that, if at least one of the deployed hash functions has an

iterative structure (which is the de-facto standard design), then one is able to generate collisions for the concatenated combiner in time $\mathcal{O}(n2^{n/2})$ where n denotes the output length of a single hash function. This is far less than the expected time $\mathcal{O}(2^n)$ given by the birthday attack for finding a collision on an ideal hash function that directly outputs $2n$ bits. In other words, the combiner provides a satisfactory hedge against a total lapse of one of the underlying hash functions but it does not increase the security for iterative hash functions. One part of this thesis presents *security-amplifying combiners* that withstand the attack of Joux and thus, are stronger than the sum of their components.

The Problem with Multiple Properties. Note that the statements above were given with respect to the property of collision-resistance only. However, as already mentioned, hash functions are currently used for various tasks that require numerous properties beyond collision-resistance, e.g., the HMAC construction [BCK96a] based on a keyed hash function is used (amongst others) in the IPSec and TLS protocols as a pseudorandom function and as a MAC. Other schemes, like the standardized protocols RSA-OAEP [BR94] and RSA-PSS [BR96] are only proven secure assuming that the applied hash function behaves like a random oracle, i.e., a public and truly random function (cf. [BF05, BF06]).

While one could in principle always employ a suitable hash combiner tailored to the individual security property needed by one particular cryptographic scheme, common practices such as code re-use, call for the design of a *single* (combiner) function satisfying as many properties as possible. On the level of hash functions this point of view has also been adopted by NIST in its on-going SHA-3 competition [NIS] and motivated a series of works [BR06a, ANPS07, LT09] that, e.g., show how to lift multiple properties provided by a compression functions to a full-grown hash function.

Thus, also for hash combiners one would ideally like to have a single construction that is robust for many properties simultaneously. Combiners which preserve a single property such as collision-resistance or pseudorandomness are quite well understood. Robust multi-property combiners, on the other hand, are not covered by these strategies and require new techniques instead. As an example we discuss this issue for the case of collision-resistance and pseudorandomness. Recall that the classical combiner for collision-resistance simply concatenates the outputs of both hash functions. Yet, it does not guarantee, for example, pseudorandomness (assuming that the hash functions are keyed) if only one of the underlying hash functions is pseudorandom. An adversary can immediately distinguish the concatenated output from a truly random value by simply examining the part of the insecure hash function. An obvious approach to obtain a hash combiner that is robust for pseudorandomness is to set

$$\mathsf{Comb}_{\oplus}(M) = H_0(M) \oplus H_1(M).$$

However, this combiner is not known to preserve collision-resistance anymore, since a collision for the combiner does not necessarily require collisions on both hash functions. In fact, this combiner also violates the above mentioned condition that for collision-resistance the output cannot be shorter than $2n$ bits. Thus, already the attempt of combining only two properties in a robust manner indicates that finding a robust multi-property combiner is far from trivial. Therefore, we initiate the study of *robust multi-property combiners* in this thesis and propose constructions that are simultaneously robust for many important properties, including collision-resistance and pseudorandomness.

Hash Combiners in Practice. Finally, we remark that hash function combiners are not only an interesting subject for theoretical investigations, but also entered practical applications. In fact, the possibility that combiners give better security assurances has been acknowledged by the designers of TLS [TLS99, TLS06] and its predecessor SSL [SSL94], long before they have been studied more thoroughly by theoreticians.

The TLS and SSL protocols are widely used to ensure secure communication over an untrusted network. Therein, a client and server first engage in the so-called handshake protocol to establish shared keys that are subsequently used to encrypt and authenticate the data transfer. Both, TLS and SSL use various combinations of MD5 and SHA1 instead of relying only on a single hash function. The specification of TLS even explicitly states:

> "*In order to make the PRF as secure as possible, it uses two hash algorithms in a way which* should *guarantee its security if either algorithm remains secure*" [TLS99] .

While the combiners in TLS mostly follow the classical approaches, SSL employs somewhat non-standard constructions. Interestingly, despite its practical importance, TLS and SSL have not undergone a comprehensive analysis for a relatively long time. An important step was done only recently by Morrissey et al. [MSW08] who gave the first security analysis of the handshake protocol of TLS. However, the combiner constructions of both protocols in particular, are not backed up with security proofs yet. We close that gap by giving the first formal treatment of the TLS and SSL hash combiners.

Contributions of this Thesis

In this thesis we address all the aforementioned issues of hash function combiners. We start by giving the foundations for our research in Chapter 2.

Chapter 3 then deals with the problem that hash combiners for collision-resistance require doubling of the output length while retaining roughly the security of a single output. Ideally, one would like to have *security-amplifying combiners* where the security of the building blocks increases the security of the combined hash function, thus going beyond the bound of Joux. To this

end we first propose a formal model that captures this intuition of security-amplification. Then we show that the classical combiner and similar proposals are *not* security amplifying according to the previous discussion. Finally, we present a (input-restricted) construction that is not only a secure combiner in the traditional sense, but even security-amplifying assuming that the underlying compression functions behave ideally. Somewhat surprisingly in light of recent attacks [NS04, HS06] that extend Joux's approach to a broader class of hash functions and combiners, our solution is essentially as efficient as the classical combiner.

In Chapter 4 we put forward the notion of *robust multi-property combiners* and elaborate on different definitions for such combiners. We then propose a combiner that provably preserves (target) collision-resistance, pseudorandomness, and being a secure message authentication code, if each property is provided by at least one underlying hash function. This construction has output length $2n$ only, which matches the lower bound of black-box combiners for collision-resistance, showing that the other properties can be achieved without penalizing the length of the hash values. We then propose a combiner which also preserves the property of being indifferentiable from a random oracle, slightly increasing the output length to $2n + \omega(\log n)$. Moreover, we show how to augment our constructions in order to make them also robust for one-wayness.

Chapter 5 shows our results for the proposed hash combiners in the TLS and SSL protocols. In order to ensure that the obtained keys are as secure as possible, both protocols deploy hash function combiners for key derivation and the authentication step in the handshake protocol. We therefore analyze the security of the proposed TLS/SSL combiner constructions with respect to the property of being a secure pseudorandom function and message authentication code respectively. Our results essentially show that the choices in TLS are sound as they follow common design criteria for such combiners whereas the SSL combiners require much stronger assumptions. However, the TLS construction that is used as pseudorandom function is neither optimal in terms of security nor efficiency. We therefore also discuss possible tweaks to obtain better security bounds while saving on computation.

Chapter 2

Definitions

In this chapter we provide some general notation and introduce the basic definitions and known results for hash functions and combiners that will be used in this work.

2.1 General Notation

Throughout this thesis, $\{0,1\}^n$ denotes the set of bit-strings x of length $|x| = n$, and 1^n stands for n in unary encoding, i.e., the string that consist of n ones. For two strings x, y we write $x||y$ for the concatenation and $x \oplus y$ for the bitwise exclusive-or of x and y. For the latter we assume that x and y have equal length.

An *adversary* $\mathcal{A}$ is a probabilistic algorithm. We write $\mathcal{A}^{\mathcal{O}}(y)$ for an adversary that runs on input y and has oracle access to $\mathcal{O}$. The shorthand $x \leftarrow X$ denotes that x is sampled from the random variable X. Similarly we write $x \leftarrow \mathcal{A}(y)$ for the output of $\mathcal{A}$ for input y. We say an adversary is *efficient* if it runs in polynomial-time. That is, if there exists a polynomial $p(n)$ such that $\mathcal{A}$ takes at most $p(n)$ steps where n is the length of the input.

A function is called *negligible* (in n) if it vanishes faster then the inverse of any polynomial. More formally, we say a function $\epsilon(n)$ is negligible if for every positive polynomial $p(\cdot)$ there exists a constant n_0, such that $\epsilon(n) < 1/p(n)$ for all $n > n_0$.

Let $X = (X_n)_{n\in\mathbb{N}}$ and $Y = (Y_n)_{n\in\mathbb{N}}$ be distribution ensembles, i.e., sequences of random variables. We say that X and Y are (computationally) *indistinguishable* if no efficient adversary can decide whether it sees an input sampled from X or from Y. Thus, for any efficient adversary $\mathcal{A}$ the advantage

$$|\text{Prob}[\mathcal{A}(1^n, x) = 1] - \text{Prob}[\mathcal{A}(1^n, y) = 1]|$$

must be negligible in n, where the probabilities are over $\mathcal{A}$'s coin tosses and the random choice of $x \leftarrow X_n$, resp. $y \leftarrow Y_n$.

2.2 Hash Functions

Loosely speaking, a hash function is a cryptographic primitive that compresses arbitrary length messages into short, fixed-length strings. More formally, a hash function $\mathcal{H} = (\mathsf{HKGen}, \mathsf{H})$ is a pair of efficient algorithms such that HKGen for input 1^n returns (the description of) a hash function H (which contains 1^n), and H for input H and $M \in \{0,1\}^*$ deterministically outputs a digest $H(M)$. Often, the hash function is also based on a public initial value IV and we therefore occasionally write $H(\mathrm{IV}, M)$ instead of $H(M)$. Similarly, we often identify the hash function with its digest values $H(\cdot)$ if the key generation algorithm is clear from the context.

Most recent hash functions such as $\mathsf{MD5}$ and $\mathsf{SHA1}$ apply the *Merkle-Damgård* construction [Mer89, Dam89] to obtain a variable input-length function out of a fixed input-length compression function $h : \{0,1\}^n \times \{0,1\}^\ell \to \{0,1\}^n$ and an initial vector IV (see also Figure 2.1). To compute a digest one divides (and possibly pads) the message $M = m_0 m_1 \dots m_{k-1}$ into blocks m_i of ℓ bits and computes the digest $H(M) = \mathrm{iv}_k$ as

$$\mathrm{iv}_0 = \mathrm{IV}, \qquad \mathrm{iv}_{i+1} = h(\mathrm{iv}_i, m_i) \qquad \text{for } i = 0, 1, \dots, k-1.$$

In this case the description of the hash function simply consists of the pair (h, IV).

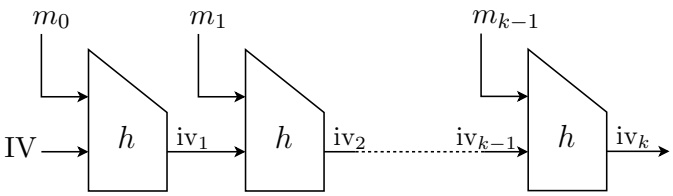


Figure 2.1: The Merkle-Damgård Construction

2.3 Properties of Hash Functions

In this section we present formal definitions of the six important security properties for hash functions (cf. [BR07]) we consider in this work: the unkeyed properties of (target) collision-resistance and one-wayness and the keyed properties of being a pseudorandom function or a message authentication code. The final property – indifferentiability from a random oracle – is special, as one considers idealized components. In particular, there is no efficient key-generation algorithm, but rather the hash function is given directly by an oracle.

Depending on the security property we are interested in, the access of the adversary to the hash function is modeled differently. For unkeyed primitives,

the description of H is given to the adversary. Whereas for keyed primitives the adversary only gets black-box access to the hash function. We could also consider a somewhat more general notion, where the key-generation algorithm outputs a pair H^p, H^s of values, which together define the hash function H, and where in the keyed setting, only H^s (but not H^p) is kept secret. For example in the HMAC construction, H^p would define the underlying compression function, and the secret key H^s would be the randomly chosen initial value IV. All our results also hold in this setting, but we avoid using such a fine-grained definition as to save on notation which would only distract from the main ideas.

Collision-Resistance (CR): Informally, collision-resistance of a hash function H requires that it should be infeasible to find two distinct messages $M \neq M'$ that map under H to the same value $H(M) = H(M')$. For the formal treatment we consider families of hash functions and call a hash function *collision-resistant* if for any efficient adversary $\mathcal{A}$ the advantage

$$\mathbf{Adv}^{\mathrm{cr}}_{\mathcal{A}}(n) = \mathrm{Prob}[H \leftarrow \mathsf{HKGen}(1^n); (M, M') \leftarrow \mathcal{A}(H) : M \neq M' \wedge H(M) = H(M')]$$

is negligible (as a function of n).

Merkle and Damgård showed that by iterating a collision-resistant compression function, as described in Section 2.2, one gets a hash function that is CR for variable input-lengths as well. An upper bound for collision-resistance for any hash function is given by the *birthday attack*. This generic attack states that for any hash function with n bits output, an attacker can find a collision in $\mathcal{O}(2^{n/2})$ steps.

Target Collision-Resistance (TCR): Target collision-resistance is a weaker security notion than collision-resistance which obliges the adversary to first commit to a target message M before getting the description $H \leftarrow \mathsf{HKGen}(1^n)$ of the hash function. For the given H the adversary must then find a second message $M' \neq M$ such that $H(M) = H(M')$.

More formally, a hash function is *target collision-resistant* if for any efficient adversary $\mathcal{A} = (\mathcal{A}^1, \mathcal{A}^2)$ the following advantage is negligible in n:

$$\mathbf{Adv}^{\mathrm{tcr}}_{\mathcal{A}}(n) = \mathrm{Prob}\left[\begin{array}{c|c} (M, \mathsf{st}) \leftarrow \mathcal{A}^1(1^n); H \leftarrow \mathsf{HKGen}(1^n); & M \neq M' \wedge \\ M' \leftarrow \mathcal{A}^2(H, M, \mathsf{st}) & H(M) = H(M') \end{array}\right].$$

The literature sometimes refer to target collision-resistance also as second-preimage resistance [RS04] or universal one-wayness [NY89].

One-Wayness (OW): The definition of one-wayness intuitively requires that it is infeasible to determine the preimage of a hash value. A hash function is called *one-way*, if for any efficient algorithm $\mathcal{A}$ the advantage

$$\mathbf{Adv}^{\mathrm{owf}}_{\mathcal{A}}(n) = \mathrm{Prob}\left[H \leftarrow \mathsf{HKGen}(1^n); M \leftarrow \{0,1\}^*; M' \leftarrow \mathcal{A}(H, H(M)) : H(M') = H(M)\right]$$

is negligible in n.

Note that, in general, one-wayness is not implied by collision-resistance. However, for hash functions that substantially compress their inputs, it was shown that CR as well as TCR imply OW [RS04]. The strengths of both implications then depend on the difference between the domain and range of the hash function.

Pseudorandomness (PRF): A hash function can be used as a pseudorandom function if, e.g., the initial value IV is replaced by a randomly chosen key K of the same size. We capture such a keyed setting by granting the adversary only black-box access to the (randomly chosen) hash function $H(\cdot)$. The hash function is then called *pseudorandom*, if no efficient adversary can distinguish H from a uniformly random function f (with the same range and same domain) with noticeable advantage. More formally, we require that for any efficient adversary $\mathcal{A}$ the advantage

$$\mathbf{Adv}^{\mathrm{prf}}_{\mathcal{A}}(n) = \left|\mathrm{Prob}\left[\mathcal{A}^{H(\cdot)}(1^n) = 1\right] - \mathrm{Prob}\left[\mathcal{A}^{f}(1^n) = 1\right]\right|$$

is negligible, where the probability in the first case is over $\mathcal{A}$'s coin tosses and the choice of $H \leftarrow \mathsf{HKGen}(1^n)$, and in the second case over $\mathcal{A}$'s coin tosses and the choice of the random function $f : \{0,1\}^* \to \{0,1\}^n$.

Message Authentication (MAC): A message authentication code is a symmetric primitive which allows a sender and receiver, both sharing a secret, to exchange information in an authenticated manner. When a hash function is used as a MAC, the description $H \leftarrow \mathsf{HKGen}(1^n)$ constitutes the shared secret, and the sender augments a message M by the tag $\tau \leftarrow H(M)$. The receiver of (M, τ) then verifies whether $\tau = H(M)$ holds.

A MAC is considered secure, if it is unforgeable under chosen message attacks, i.e., an adversary after adaptively learning several tags $(M_1, \tau_1), (M_2, \tau_2), \ldots, (M_q, \tau_q)$ should not be able to compute a forgery for a fresh message M^*. Note that the adversary has again only oracle access to $H(\cdot)$. More compactly, a hash function is called a *secure* MAC, if for any efficient adversary $\mathcal{A}$ the following advantage is negligible in n

$$\mathbf{Adv}^{\mathrm{mac}}_{\mathcal{A}}(n) = \mathrm{Prob}\left[H \leftarrow \mathsf{HKGen}(1^n), (M, \tau) \leftarrow \mathcal{A}^{H(\cdot)} : H(M) = \tau \wedge M \text{ not queried}\right].$$

A pseudorandom function always gives a secure message authentication code, while vice-versa a concrete MAC may not directly yield a full-fledged PRF. However, existentially a MAC and a PRF have been shown to be equivalent [NR98].

Indifferentiability from Random Oracle (IRO): Some cryptographic protocols, e.g., RSA-OAEP [BR94] and RSA-PSS [BR96], require stronger properties from hash functions than the ones considered so far. In those cases, a hash function is assumed to be a *a random oracle*, i.e., a public random function that is accessible by all parties in a black-box manner and returns truly random values for each query (cf. [BR93]).

While random oracles are modeled as monolithic entities, hash functions are usually highly structured due to the Merkle-Damgård design as described in Section 2.2. Coron et al. [CDMP05] bridged that gap by considering a hash transform H as secure, when the underlying compression function f is given as a fixed input-length random oracle and the resulting hash function H^f "behaves like" a (variable input-length) random oracle. The formalization of that idea is based on the indifferentiability notion [MRH04] which is a generalization of indistinguishability allowing to consider random oracles that are used as public components.

According to [MRH04, CDMP05] a hash function H^f is *indifferentiable* from a random oracle $\mathcal{F}$ if for any efficient adversary $\mathcal{A}$ there exists an efficient algorithm $\mathcal{S}$ such that the advantage

$$\mathbf{Adv}^{\mathrm{ind}}_{\mathcal{A}}(n) = \left|\mathrm{Prob}\left[\,\mathcal{A}^{H^f,f}(1^n) = 1\right] - \mathrm{Prob}\left[\,\mathcal{A}^{\mathcal{F},\mathcal{S}^{\mathcal{F}}}(1^n) = 1\right]\right|$$

is negligible in n, where the probability in the first case is over $\mathcal{A}$'s coin tosses and the choice of the random function f, and in the second case over the coin tosses of $\mathcal{A}$ and $\mathcal{S}$, and over the choice of $\mathcal{F}$.

The goal of the simulator $\mathcal{S}^{\mathcal{F}}$ is to mimic the ideal compression function f, such that no adversary $\mathcal{A}$ can decide whether its interacting with H^f and f or with $\mathcal{F}$ and $\mathcal{S}^{\mathcal{F}}$. To this end, $\mathcal{S}^{\mathcal{F}}$ has to produce output that is random but consistent with the values the adversary can obtain from the random oracle $\mathcal{F}$. Note that the simulator has oracle access to $\mathcal{F}$ too, but it does *not* get to see the queries $\mathcal{A}$ issues to $\mathcal{F}$.

Roughly speaking, indifferentiability of a hash function states that the design has no structural flaws and provides security against generic attacks. Furthermore, when a hash function H^f is proven to be indifferentiable from a random oracle $\mathcal{F}$, then H^f can replace $\mathcal{F}$ in any cryptographic scheme, while the scheme remains secure. For a comprehensive treatment of the indifferentiability framework we refer to [MRH04].

2.4 Robust Combiners

As discussed earlier, a combiner for a cryptographic primitive is a function that "merges" two candidate implementations into a single one. The combiner is called *property-preserving* for some property P if it enjoys this property given that *both* underlying functions have P. In a sense, this ensures a minimalistic security guarantee. The combiner is called *robust* if it obeys the property if at least *one* of the two functions has the corresponding property. The idea of such constructions is to provide robustness against insecure implementations or wrong assumptions of the underlying functions. We refer to Herzberg [Her05, Her09] and Harnik et al. [HKN+05] for a broad introduction of robust combiners for various cryptographic primitives.

Note that the concept of robust combiners naturally extends to a more general setting, where (k, l)-robust combiners are considered. Such combiners are guaranteed to securely implement a property P, if at least k of the l deployed components obey P. However, as most of our results are given for the $(1, 2)$ setting, we avoid that general notation.

2.4.1 Hash Function Combiners

In this thesis we scrutinize robust combiners for hash functions. A hash function combiner $\mathcal{C}$ for hash functions $\mathcal{H}_0, \mathcal{H}_1$ itself is also a hash function which combines the two functions $\mathcal{H}_0$ and $\mathcal{H}_1$ such that it securely guarantees property P as long as $\mathcal{H}_0$ or $\mathcal{H}_1$ obey P. More formally, a hash function combiner $\mathcal{C} = (\mathsf{CKGen}, \mathsf{Comb})$ is a pair of efficient algorithms, where $\mathsf{CKGen}(1^n)$ generates $H_0 \leftarrow \mathsf{HKGen}_0(1^n)$ and $H_1 \leftarrow \mathsf{HKGen}_1(1^n)$ and outputs (H_0, H_1). In addition, Comb for hash functions $\mathcal{H}_0, \mathcal{H}_1$ is an efficient deterministic algorithm such that, for input $H_0 \leftarrow \mathsf{HKGen}_0(1^n)$, $H_1 \leftarrow \mathsf{HKGen}_1(1^n)$ and $M \in \{0,1\}^*$, it returns a digest $\mathsf{Comb}(H_0, H_1, M)$.

As intuitive examples we briefly discuss the somewhat classical hash function combiners which guarantee collision-resistance resp. pseudorandomness in a robust way.

Classical Combiner for CR. The standard approach to obtain a robust combiner for collision-resistance is to invoke two hash functions H_0, H_1 on the same message M and concatenate their outputs:

$$\mathsf{Comb}_{||}^{H_0,H_1}(M) = H_0(M)||H_1(M).$$

It is easy to see that a collision $M \neq M'$ for the combiner is always also a collision for both components H_0 or H_1. Thus if either of the hash function H_0 or H_1 is collision-resistant, then so is the combined function. A similar argumentation can be used to show that $\mathsf{Comb}_{||}^{H_0,H_1}$ robustly preserves the property of being target collision-resistant and a secure MAC [Her05].

As this combiner doubles the output length from n to $2n$ bits, the question whether more efficient constructions exist arose. However, [BB06, Pie07, CRS+07, Pie08] gave a negative answer to that question by showing that the output of a (black-box) collision-resistant combiner cannot be significantly shorter than the concatenation of the outputs from all employed hash functions.

Moreover, Joux [Jou04] presented the so-called *multi-collision attack*, which states that the concatenation of Merkle-Damgård hash functions is not much more secure than the individual functions. The generic attack exploits the iterative structure of a hash function H and allows to obtain many collisions on H (roughly) for the price of one. More precisely, one first searches for k consecutive collisions $(m_0, m'_0), (m_1, m'_1), \ldots (m_{k-1}, m'_{k-1})$ on the underlying compression function of H, e.g., by running each time the birthday attack with complexity $\mathcal{O}(2^{n/2})$ (where n denotes the output length). As observed in [Jou04], those k collisions found in time $\mathcal{O}(k2^{n/2})$ immediately give 2^k distinct messages that all hash to the same value on H as depicted in Figure 2.2. As a consequence, collisions on the concatenated combiner can be found by simply generating $2^{n/2}$ multi-collisions on one of the hash functions H_b for $b \in \{0, 1\}$. Then, due to the birthday attack, two messages from that multi-collision set are also expected to collide under $H_{\bar{b}}$, and hence under $\mathsf{Comb}_{||}^{H_0,H_1}$. Thus, a collision on the combiner can be obtained in expected time $\mathcal{O}(n2^{n/2})$, which is significantly below the generic birthday bound of $\mathcal{O}(2^n)$ for an ideal hash function with $2n$ bits of output. More generally speaking, if an adversary can find collisions for H_0 and H_1 in time T_0 and T_1, respectively, then Joux's multi-collision attack allows to break the concatenated combiner in $\frac{n}{2} \cdot T_0 + T_1$ steps.

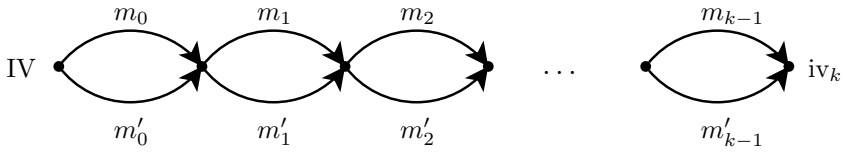


Figure 2.2: Multi-Collision Construction

Classical Combiner for PRF. An obvious approach to construct a robust hash combiner that preserves pseudorandomness is to compute the exclusive-or of the outputs of two independently chosen hash functions:

$$\mathsf{Comb}_{\oplus}^{H_0,H_1}(M) = H_0(M) \oplus H_1(M).$$

The above combiner is also robust for the IRO-property, and, as we show in Section 5.4, for MAC as well. However, $\mathsf{Comb}_{\oplus}^{H_0,H_1}$ is not suitable to guar-

antee collision-resistance, since a collision for the combiner does not necessarily require collisions on both hash functions. The combiner is not even CR-preserving, i.e., two collision-resistant hash functions might complement in a way, such that $H_0(M) \oplus H_1(M) = H_0(M') \oplus H_1(M')$ but no collisions on *both* underlying functions occurred.

2.4.2 Combiners for Other Primitives

The investigation of combiners is of course not limited to hash functions. In fact, the work on robust combiners was implicitly triggered by studies on secure combinations of encryption schemes. Those date back to 1981 where Asmuth and Blakely [AB81] considered a variant of the sequential (cascade) application of two encryption systems in order to guarantee security even if one cannot be trusted. Cascade encryption in general then became subject for further research: Even and Goldreich [EG85] showed that for multiple ciphers the cascade is at least as strong as the weakest cipher in the chain. Later, Maurer and Massey [MM93] proved that for a weaker attack model, the cascade combination is at least as strong as the first cipher in the cascade. A robust combiner for chosen ciphertext secure encryption was recently proposed by Dodis and Katz [DK05].

As already mentioned, Herzberg [Her05, Her09] and Harnik et al. [HKN+05] considered combiners for various cryptographic primitives including commitment schemes and key-agreement. The latter work then fostered a line of research concerning robust combiners for private information retrieval [MP06] and oblivious transfer [MPW07, HIKN08, PW08]. In [MP06] also *cross-primitive combiner* were proposed, which can be seen as the combination of a reduction and a combiner, as the combined primitive is different from the underlying components.

Chapter 3

Amplifying Collision-Resistance

This chapter deals with hash function combiners that are robust for collision-resistance. The classical combiner for this purpose concatenates the output of two hash functions H_0, H_1 and provides collision-resistance as long as at least one of the two underlying functions is secure. This statement is complemented by the multi-collision attack of Joux [Jou04] for iterated hash functions H_0, H_1 with n-bit outputs. He shows that one can break the classical combiner in $\frac{n}{2} \cdot T_0 + T_1$ steps if one can find collisions for H_0 and H_1 in time T_0 and T_1, respectively.

Here we introduce *security-amplifying* combiners where the security of the building blocks increases the security of the combined hash function, thus beating the bound of Joux. We start by defining our model and security amplifying combiners in Section 3.2. Next, in Section 3.3, we discuss that the classical combiner and similar proposals are not security amplifying. Section 3.4 present some general conclusions in our model. The main result appears in Section 3.5 and its proof is given in Section 3.6.

This work has been presented at Crypto 2007 [FL07].

3.1 Introduction

A hash function combiner takes two hash functions H_0 and H_1 and combines them into a single, failure-resistant hash function. For collision-resistance the classical combiner is $\mathsf{Comb}_{||}^{H_0,H_1}(M) = H_0(M)||H_1(M)$, concatenating the outputs of the two hash functions. Any collision $M \neq M'$ on the combiner then immediately gives collisions for both hash functions H_0 and H_1.

From a more quantitative viewpoint, the classical combiner provides the following security guarantee: If breaking H_0 and H_1 requires T_0 and T_1 steps, respectively, finding a collision for the classical combiner takes at least $T_0 + T_1$ steps. This almost matches an upper bound by Joux [Jou04], showing that for Merkle-Damgård hash functions H_0, H_1 with n-bit outputs the classical

combiner can be broken in $\frac{n}{2} \cdot T_0 + T_1$ steps. This means that if the security level of each hash function is degraded only moderately through a new attack method, e.g., from 2^{80} to 2^{60}, then the classical combiner, too, merely warrants a reduced security level of $T_0 + T_1 = 2 \cdot 2^{60}$. Ideally, we would like to have a better security bound for combiners and such moderate degradations, going beyond the $T_0 + T_1$ limit and the bound due to Joux.

Our Results. Here we introduce the notion of security-amplifying combiners for collision-resistant hash functions. Such combiners guarantee a security level $\alpha \cdot (T_0 + T_1)$ for some $\alpha > 1$ and, in a sense, are therefore stronger than the sum of their components. Note that the classical combiner (and similar proposals) are *not* security amplifying according to the previous discussion, indicating that constructing such security-amplifying combiners is far from trivial.

We next discuss how to achieve security amplification. Consider two Merkle-Damgård hash functions H_0, H_1 (given by compression functions f_0, f_1) and the classical combiner, but limited to input messages $M = m_0||\ldots||m_{t-1}$ of $t < \frac{n}{4}$ blocks exactly:

$$\mathsf{Comb}^{H_0,H_1}_{\mathrm{amp},t}(M) = H_0(m_0||\ldots||m_{t-1}) \,||\, H_1(m_0||\ldots||m_{t-1})$$

This is clearly a secure combiner in the traditional sense, guaranteeing collision-resistance if at least one of both hash functions is collision-resistant. But we show that it is even a security-amplifying combiner, assuming that the underlying compression functions behave ideally. More precisely, we consider an attack model in which the compression functions f_0, f_1 are given by random functions, but where the adversary against the combiner can use subroutines $\mathsf{Coll}_0, \mathsf{Coll}_1$ to generate collisions for the corresponding compression function. Intuitively, these collision finder oracles implement the best known strategy to find collisions, and each time the adversary calls Coll_b to get a collision for f_b, we charge T_b steps. The adversary's task is now to turn such collisions derived through $\mathsf{Coll}_0, \mathsf{Coll}_1$ into one against the combiner.

We note that the adversary against the combiner in our model is quite powerful. For each query to the collision finders the adversary can significantly bias the outcome, e.g., by presetting parts of the colliding messages. To give further support of the significance of our model, we show that we can implement the attack of Joux on the classical combiner $\mathsf{Comb}_{||}$ in our model. We can also realize similar attacks for more advanced combiners like $\mathsf{Comb}^{H_0,H_1}(M) = H_0(M)||H_1(H_0(M) \oplus M)$.

Our main result is to certify the security amplification of our combiner $\mathsf{Comb}_{\mathrm{amp},t}$. The proof is basically split into two parts: one covering general statements about our model (such as pre-image resistance, even in presence of the collision finders), and the other part uses the basic facts to prove our specific combiner $\mathsf{Comb}_{\mathrm{amp},t}$ to be security-amplifying. In our security proof we show that calling each collision finder $\mathsf{Coll}_0, \mathsf{Coll}_1$ only polynomially many

times does not help to find a collision for $\mathsf{Comb}_{\mathrm{amp},t}$. Therefore, successful attacks on the combiner require more than $\mathrm{poly}(n) \cdot (T_0 + T_1)$ steps.

Viewed from a different perspective we can think of our result as a supplementary lower bound to the attack of Joux. His attack breaks the classical combiner in $\frac{n}{2} \cdot T_0 + T_1$ steps if the hash functions allow to process $t \geq \frac{n}{2}$ message blocks. Our result indicates that restricting the input to $t < \frac{n}{4}$ many blocks suffices to make the combiner security-amplifying and to overcome the bound by Joux. The situation for t in between $\frac{n}{4}$ and $\frac{n}{2}$ remains open.

Finally, recall that our proposal at this point only allows to hash messages of $t < \frac{n}{4}$ blocks. To extend the combiner to handle arbitrarily long messages one can use hash trees in a straightforward way (with our combiner placed at every node of the tree). Since finding collisions in such hash trees requires to come up with collisions in one of the nodes, our security amplification result carries over instantaneously. For messages of k blocks the classical combiner takes about $2k$ applications of the compression functions, compared to roughly $\frac{t}{t-1} \cdot 2k$ applications for our tree-based combiner (but coming with the stronger security amplification guarantee).

Limitations of the Model. Our hash combiner guarantees security amplification in an idealized world where the underlying compression functions behave like random functions. In this model only generic attacks on the hash function are allowed, in the sense that the adversary cannot take advantage of weaknesses of the compression functions beyond the ability to generate collisions (albeit the collision finders are quite flexible). It remains open if similar results can be obtained in a non-idealized setting at all.

Currently, our collision finders return two values mapping to the same compression function output. A recent work of Yu and Wang [YW07], however, shows that very weak compression functions as in MD4 may allow K-multi-collision attacks, where one is able to find K instead of 2 simultaneous collisions for the compression functions. We expect our results to transfer to this case, when restricting the number of message blocks further to $t < \frac{n}{4 \log_2 K}$. However, since such strong attacks are only known for specific compression functions that were already considered insecure, we refrain from a thorough treatment of K-multi-collisions in our general setting.

Related Work. Interestingly, the idea of security amplification for cryptographic combiners already appears implicitly in Yao's work [Yao82]. He shows that the existence of weak one-way functions —where inversion may succeed with probability $1 - 1/\mathrm{poly}(n)$— can be turned into strong one-way functions where inversion almost surely fails. The construction can be viewed as a security-amplifying self-combiner for one-way functions. See also [GIL+90] for improvements and [LTW05] for related results.

Other relevant works are the upper bounds of Nandi and Stinson [NS04] and of Hoch and Shamir [HS06]. They extend the attack of Joux to arbitrary combiners for iterated hash functions, where each message block is possibly

processed via the compression function more than once but at most a constant number of times, e.g., $\mathsf{Comb}^{H_0,H_1}(M) = H_0(m_1 m_1 || \ldots || m_k m_k) || H_1(m_1 || \ldots || m_k || m_1 || \ldots || m_k)$. They also transfer their results to tree-based constructions. However, in their model the output of one compression function must not serve as an input to the other compression function, thus disallowing mixes of intermediate hash values. By this, the hash-tree based extension of our combiner circumvents their bounds.

In a recent work, Hoch and Shamir [HS08] provide a lower bound for the concatenated combiner based on weak hash functions. The security is analyzed in Liskov's model [Lis06], where the underlying compression functions are assumed to be ideal, but the adversary has also access to a "breaking" oracle. There the additional oracle cannot only provide collisions but even fully invert the compression function on a given input. It is shown that this does not significantly weaken the combiner, as a collision on the concatenated output still requires at least $2^{n/2}$ steps.

Finally we remark that, in a concurrent work, Canetti et al. [CRS+07] also consider amplification of collision-resistance. In contrast to our idealized setting they use a complexity-theoretic approach.

3.2 Our Model

Note that our results are given for *idealized* Merkle-Damgård (MD) constructions where we assume that the compression function f behaves like a random function (drawn from the set of all functions mapping $(l+n)$-bit strings to n-bit strings). In particular, if an algorithm now gets as input the description of such an idealized MD hash function then it is understood that this algorithms gets IV as input string and oracle access to the random function f. This holds also for a combiner Comb of such idealized MD hash function, i.e., Comb gets oracle access to f_0, f_1 and receives the strings $\mathrm{IV}_0, \mathrm{IV}_1$ as input. We then often write $\mathsf{Comb}^{H_0,H_1}(\cdot)$ instead of $\mathsf{Comb}^{f_0,f_1}(\mathrm{IV}_0, \mathrm{IV}_1, \cdot)$. *We emphasize that the combiner may assemble a solution from the compression functions and the initial vectors which is not necessarily an iterated hash function.*

To analyze the security amplification of a combiner for two idealized MD hash functions (f_0, IV_0) and (f_1, IV_1) we consider an adversary $\mathcal{A}$ with oracle access to f_0, f_1 and input $\mathrm{IV}_0, \mathrm{IV}_1$. The task of this algorithm is to find a collision for the combiner. Since finding collisions for the random compression function directly is restricted to the birthday attack, we adapt the approach of Liskov[1] [Lis06] and allow $\mathcal{A}$ additional oracle access to two *collision finder oracles* Coll_0, Coll_1 generating collisions for each compression function (both oracles themselves have access to f_0, f_1). These collision finders can be viewed

[1] Liskov introduced in [Lis06] the concept of *weak compression functions*, which are modeled as random oracles but also capture vulnerabilities by giving the adversary access to an additional inversion oracle.

as the best known algorithm to generate collision for the compression function. See Figure 3.1.

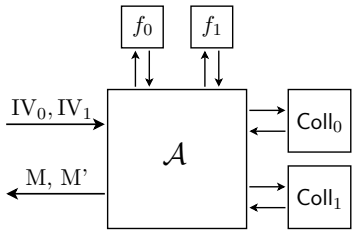
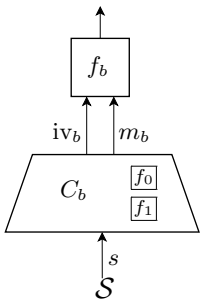


Figure 3.1: Attack Model

In its most simple form, algorithm $\mathcal{A}$ can query the collision finder Coll_b by forwarding values $\mathrm{iv}_b, \mathrm{iv}'_b$ and getting a collision (m_b, m'_b) with $f_b(\mathrm{iv}_b, m_b) = f_b(\mathrm{iv}'_b, m'_b)$ from Coll_b. More generally, the adversary may want to influence the colliding messages or enforce dependencies between the initial values $\mathrm{iv}_b, \mathrm{iv}'_b$ and the messages m_b, m'_b. To model such advanced collision finding strategies we allow the adversary to pass (the description of) a circuit $C_b : \{0,1\}^i \to \{0,1\}^{l+n}$ (possibly containing f_0- and f_1-gates) to Coll_b instead of $\mathrm{iv}_b, \mathrm{iv}'_b$ only. The collision finder then applies an internal stateful source $\mathcal{S} = \mathcal{S}(C_b)$ to continuously generate i-bit strings $s \leftarrow \mathcal{S}$ and successively provides each s as input to the circuit C_b. See Figure 3.2a.[2]



(a)

SAMPLES$_b(C_b)$ contains all tested pairs $(C_b(s), f_b(C_b(s)))$ in Coll_b's collision search for input circuit C_b

CVAL$_b$ contains all collisions returned by collision finder Coll_b

FVAL$_b$ contains all pairs $(x, f_b(x))$ appearing in direct f_b-box queries of $\mathcal{A}$ or in an evaluation of a circuit C_b

(b)

Figure 3.2: Operation of collision finder Coll_b (a), Sets of function values (b)

For the circuit's output $(\mathrm{iv}_b, m_b) = C_b(s)$ to the next input value s, the collision finder computes $f_b(\mathrm{iv}_b, m_b)$ and checks if for some previously computed value (iv'_b, m'_b) a collision $f_b(\mathrm{iv}_b, m_b) = f_b(\mathrm{iv}'_b, m'_b)$ occurs. If so, the finder Coll_b immediately stops and outputs the collision $((\mathrm{iv}_b, m_b), f_b(\mathrm{iv}_b, m_b), s)$ and $((\mathrm{iv}'_b, m'_b), f_b(\mathrm{iv}'_b, m'_b), s')$. Otherwise it stores the new triple $((\mathrm{iv}_b, m_b), f_b(\mathrm{iv}_b,$

[2]The source $\mathcal{S}$ can be thought of the collision finder's strategy to generate collisions for the input circuit, and is possibly even known by $\mathcal{A}$. Since we will later quantify over all collision finders we do not specify this distribution; the reader may for now think of $\mathcal{S}$ sequentially outputting the values $0, 1, 2, \ldots$ in binary.

$m_b), s)$ and continues its computations. If Coll_b does not find a collision among all i-bit inputs s to the circuit it returns $\perp$. We assume that the adversary implicitly gets to know all consulted input values s, gathered in an ordered set $\text{SVAL}(C_b)$. Note that we leave it essentially up to the adversary and his choice for C_b to minimize the likelihood of undefined outputs or trivial collisions (i.e., for the same pre-image).

3.2.1 Lucky Collisions

The collision finders should be the only possibility to derive collisions, i.e., we exclude accidental collisions (say, $\mathcal{A}$ ignoring the collision finders and finding an f_0-collision by querying the f_0-oracle many times). To capture such *lucky collisions* we assume that each answer $((\text{iv}_b, m_b), f_b(\text{iv}_b, m_b), s)$, $((\text{iv}'_b, m'_b), f_b(\text{iv}'_b, m'_b), s')$ of Coll_b is augmented by all pre-image/image pairs (x, y) of f_0- and f_1-gate evaluations in the circuit computations during the search. We stress that this excludes all samples $(C_b(s), f_b(C_b(s)))$ which the collision finder probes to find the collision, unless the sample also appears in one of the circuit evaluations (see also the discussion below).

For a query C_b to Coll_b we denote the set of the pre-image/image pairs returned to $\mathcal{A}$ by $\text{FVAL}_b^{\text{cf}}(C_b)$ and by $\text{FVAL}_b^{\text{cf}}$ we denote the union of $\text{FVAL}_b^{\text{cf}}(C_b)$ over all queries C_b made to Coll_b during $\mathcal{A}$'s computation. Here we assume that the set $\text{FVAL}_b^{\text{cf}}$ is updated immediately after each function gate evaluation during a circuit evaluation. Similarly, $\text{FVAL}_b^{\text{box}}$ stands for the pre-image/image pairs generated by $\mathcal{A}$ as queries and answers to the f_b-box directly. We now set FVAL as the union of $\text{FVAL}_b^{\text{cf}}$ and $\text{FVAL}_b^{\text{box}}$ for both $b = 0, 1$.

Definition 3.1 (Lucky Collision) *A pair (x, x') is called a* lucky collision *if for an execution of $\mathcal{A}$ we have $x \neq x'$ and $(x, y), (x', y) \in \text{FVAL}$ for some y.*

In the definition below $\mathcal{A}$ will not be considered successful if a lucky collision occurs during an execution. It therefore lies in $\mathcal{A}$'s responsibility to prevent lucky collisions when querying f-boxes or the collision finders.

For notational convenience we collect the pre-image/image pairs of collisions generated by the collision-finders in the set CVAL, which is the union of all answers $\text{CVAL}_b(C_b)$ of collision-finder Coll_b for query C_b, over all queries C_b and $b = 0, 1$. We also let $\text{SAMPLES}_b(C_b)$ denote all samples $(C_b(s), f_b(C_b(s)))$ which the collision finder Coll_b collects to find a collision for query C_b, and SAMPLES stands for the union over all $\text{SAMPLES}_b(C_b)$ for all queries C_b and $b \in \{0, 1\}$. Clearly, $\text{CVAL}_b(C_b) \subseteq \text{SAMPLES}_b(C_b)$. An informal overview about the sets is given in Figure 3.2b.

We remark that we do not include the pairs $(C_b(s), f_b(C_b(s)))$ which the collision finder probes in FVAL_b (unless they appear in the circuit's evaluations). This is in order to not punish the adversary for the collision finder's search and strengthens the model, as lucky collisions become less likely. However, for an answer of the collision finder the adversary $\mathcal{A}$ can re-compute

all or some of those values by browsing through the ordered set $\text{SVAL}(C_b)$, containing all inspected s-values, and submitting $C_b(s)$ to the f_b-oracle. This value is then added to the set FVAL_b, of course.

3.2.2 Security Amplification

As for the costs of each oracle call to collision finder Coll_b we charge the adversary $\mathcal{A}$ a pre-determined number T_b of steps for each call (e.g., $T_b = 2^{n/2}$ if Coll_b implements the birthday attack, ignoring the fact that the collision finder may even fail with some probability in this case). We do not charge the adversary for other steps than these calls. In the definition below we make no restriction on the number of calls to the collision finders, yet one might often want to limit this number in some non-trivial way, e.g., for our main result we assume that the adversary makes at most a polynomial number of calls.

Definition 3.2 *A hash combiner* Comb *for idealized Merkle-Damgård hash functions* $\mathcal{H}_0, \mathcal{H}_1$ *is called* $\alpha(n)$-security amplifying *if for any oracles* $\mathsf{Coll}_0, \mathsf{Coll}_1$ *(with running times* $T_0(n)$ *and* $T_1(n)$*, respectively) and any algorithm* $\mathcal{A}$ *making at most* $\alpha(n) \cdot (T_0(n) + T_1(n))$ *steps we have*

$$\text{Prob}\left[\mathbf{Exp}^{\textit{amp-comb}}_{\mathcal{A},\mathcal{H}_0,\mathcal{H}_1,\mathsf{Coll}_0,\mathsf{Coll}_1}\mathsf{Comb}(n) = 1\right] \approx 0$$

where

Experiment $\mathbf{Exp}^{\textit{amp-comb}}_{\mathcal{A},\mathcal{H}_0,\mathcal{H}_1,\mathsf{Coll}_0,\mathsf{Coll}_1}\mathsf{Comb}(n)$*:*
initialize $(f_0, IV_0) \leftarrow \mathsf{HKGen}_0(1^n)$*,* $(f_1, IV_1) \leftarrow \mathsf{HKGen}_1(1^n)$
let $(M, M') \leftarrow \mathcal{A}^{f_0,f_1,\mathsf{Coll}_0,\mathsf{Coll}_1}(IV_0, IV_1)$
output 1 *iff*
$M \neq M'$*, and*
$\mathsf{Comb}^{f_0,f_1}(IV_0, IV_1, M) = \mathsf{Comb}^{f_0,f_1}(IV_0, IV_1, M')$*, and*
no lucky collisions during $\mathcal{A}$*'s computation occurred.*

The combiner is called security amplifying *if it is* $\alpha(n)$*-security amplifying for some function* $\alpha(n)$ *with* $\alpha(n) > 1$ *for all sufficiently large* n*'s.*

Our definition allows $\alpha(n)$ to converge to 1 rapidly, e.g., $\alpha(n) = 1 + 2^{-n}$. We do not exclude such cases explicitly, but merely remark that, as long as $T_0(n)$ and $T_1(n)$ are polynomially related and the combiner is security-amplifying, one can always find a suitable function $\alpha(n)$ bounded away from 1 by a polynomial fraction. The definition also captures the more general running time requirement $\alpha_0(n) \cdot T_0(n) + \alpha_1(n) \cdot T_1(n)$, where both collision finders may be called a different number of times, when we consider $\alpha(n) = \min\{\alpha_0(n), \alpha_1(n)\}$.

For simplicity we have defined compression functions f_0, f_1 of equal output length n (which is also the security parameter). We remark that all our definitions and results remain valid for different output lengths n_0, n_1 by considering $n = \min\{n_0, n_1\}$.

3.3 Warming Up: Attack on the Classical Combiner

In this section, to get accustomed to our model, we first present the attack of Joux on the classical combiner, showing that this one is not security amplifying (even though it is a secure combiner in the traditional sense). This also proves that finding such security-amplifying combiners is far from trivial.

Recall that the classical combiner is given by

$$\mathsf{Comb}_{||}^{H_0 H_1}(M) = H_0(M)||H_1(M)$$

for idealized Merkle-Damgård hash functions. Obviously this combiner is collision-resistant as long as at least one of the hash functions has this property. Yet, it does not have the desired security-amplification property, because an adversary $\mathcal{A}$ can use the strategy of Joux [Jou04] to find a collision rapidly. The idea is to build a multi-collision set of size $2^{\frac{n}{2}}$ for H_0 by calling Coll_0 only $\frac{n}{2}$ times, and then to let Coll_1 search for a pair among those messages in the multi-collision set which also constitutes a collision under H_1.

Adversary $\mathcal{A}^{f_0,f_1,\mathsf{Coll}_0,\mathsf{Coll}_1}(\mathrm{IV}_0,\mathrm{IV}_1)$:
for $i = 0,1,\ldots,k$ with $k = \frac{n}{2} - 1$:
  let $C_{0,i} : \{0,1\}^l \to \{0,1\}^{l+n}$ be the circuit $C_{0,i}(s) = (\mathrm{iv}_{0,i}, s)$, where $\mathrm{iv}_{0,0} = \mathrm{IV}_0$
  get $((\mathrm{iv}_{0,i}, m_i), y_i, s), ((\mathrm{iv}_{0,i}, m_i'), y_i, s') \leftarrow \mathsf{Coll}_0(C_{0,i})$
    where $m_i \neq m_i'$ by the choice of $C_{0,i}$
  set $\mathrm{iv}_{0,i+1} = y_i$
end of for

construct circuit $C_1 : \{0,1\}^{n/2} \to \{0,1\}^{l+n}$, that contains all received collisions (m_i, m_i') from the first stage, as follows:
  for $i = 0,1,\ldots,k$ with $k = \frac{n}{2} - 1$:
    for the i-th input bit s_i let $\widehat{m}_i = m_i$ if $s_i = 0$, and $\widehat{m}_i = m_i'$ otherwise
    except for the last round, compute $\mathrm{iv}_{1,i+1} = f_1(\mathrm{iv}_{1,i}, \widehat{m}_i)$, where $\mathrm{iv}_{1,0} = \mathrm{IV}_1$
  end of for
  let the circuit output $(\mathrm{iv}_{1,k}, \widehat{m}_k)$

get $((\mathrm{iv}_{1,k}, \widehat{m}_k), y_k, s), ((\mathrm{iv}'_{1,k}, \widehat{m}'_k), y_k, s') \leftarrow \mathsf{Coll}_1(C_1)$

reconstruct the successful combination M, M' of Coll_1 by using the values s, s' for the pairs (m_i, m_i') as above, and output M, M'

First, the collision finder Coll_0 is called $\frac{n}{2}$ times by the adversary to derive $\frac{n}{2}$ pairs of colliding message blocks (m_i, m_i') where $f_0(\mathrm{iv}_{0,i}, m_i) = f_0(\mathrm{iv}_{0,i}, m_i')$ for $i = 0,1,\ldots,k$. Since the circuit $C_{0,i}$ passed to Coll_0 does not evaluate the functions f_0, f_1, no lucky collision can occur in this stage. The query to collision finder Coll_1 then requires $\frac{n}{2}$ compression function evaluations in the circuit C_1 for each input $s \in \{0,1\}^{n/2}$, which selects one of the $2^{\frac{n}{2}}$ multi-collisions derived from Coll_0's answers. Yet, for each common prefix of the s-values the same function evaluations are repeated, and the set $\textsc{fval}_1^{\mathrm{cf}}$ therefore

contains at most $2^{\frac{n}{2}}$ pre-image/image pairs (x, y) from the circuit evaluations. This implies that the probability for a lucky collision is at most $\frac{1}{2}$.

On the other hand, given that no collision in FVAL_1 occurs, all circuit outputs are distinct and the set of probed values of the collision finder is at least $2^{\frac{n}{2}}$. But then, Coll_0 will find a collision among the values with constant probability (which is roughly equal to $1 - e^{-1/2}$ for the Euler constant e). Hence, the adversary succeeds with constant probability, taking only $\frac{n}{2} \cdot T_0(n) + T_1(n)$ steps. This implies that the classical combiner is *not* security amplifying, because no appropriate function $\alpha(n) > 1$ exists.

Our model also allows to implement attacks on more sophisticated hash combiners such as $\mathsf{Comb}^{H_0,H_1}(M) = H_0(M)||H_1(H_0(M) \oplus M)$, which may seem to be more secure than the classical combiner at first glance due to the dependency of both hash functions. However, by using the circuit C_1 to compute valid inputs for H_1 we can realize a similar attack as the one for $\mathsf{Comb}_{||}$.

3.4 Basic Conclusions

In this section we provide some basic conclusions in our model, e.g., that the functions f_0, f_1 are still pre-image resistant in presence of the collision finders. These results will also be useful when proving our combiner to be security amplifying.

The first lemma basically restates the well-known birthday paradox that, if the adversary $\mathcal{A}$ in experiment $\mathbf{Exp}^{\text{amp-comb}}_{\mathcal{A},\mathcal{H}_0,\mathcal{H}_1,\mathsf{Coll}_0,\mathsf{Coll}_1}\mathsf{Comb}(n)$ makes too many f_0- and f_1-queries (either directly or through the collision-finders), then most likely a lucky collision will occur and $\mathcal{A}$ cannot succeed anymore. This result —like all results in this section— hold for arbitrary combiners (based on the idealized Merkle-Damgård model):

Lemma 3.3 (Birthday Paradox) *Consider the security-amplification experiment* $\mathbf{Exp}^{amp\text{-}comb}_{\mathcal{A},\mathcal{H}_0,\mathcal{H}_1,\mathsf{Coll}_0,\mathsf{Coll}_1}\mathsf{Comb}(n)$ *and assume that* $|\text{FVAL}_b| > 2^{dn}$ *for* $b \in \{0,1\}$ *and a constant* $d > \frac{1}{2}$. *Then the probability that no lucky collisions occur is negligible (and, in particular, the probability that the experiment returns* 1 *is negligible, too).*

Proof. Suppose $|\text{FVAL}_b| > 2^{dn}$ for some b. Then the birthday paradox implies that with probability at most $\exp(-\binom{2^{dn}+1}{2}/2^n) \leq \exp(-2^{(2d-1)n-1})$ there would be *no* lucky collision. Since $d > \frac{1}{2}$ the term $2^{(2d-1)n-1}$ grows exponentially in n. But if a lucky collision occurs, then the experiment outputs 0. □

We next show that the images of sample values $\text{SAMPLES} \setminus \text{CVAL}$ appearing during the search of the collision finder (but which are not returned to $\mathcal{A}$) are essentially uniformly distributed from $\mathcal{A}$'s viewpoint (i.e., given the sets

FVAL, CVAL). This holds at any point in the execution and even if $\mathcal{A}$ does not win:

Lemma 3.4 (Image Uncertainty) *Assume that algorithm $\mathcal{A}$ in experiment $\mathbf{Exp}^{amp\text{-}comb}_{\mathcal{A},\mathcal{H}_0,\mathcal{H}_1,\mathsf{Coll}_0,\mathsf{Coll}_1}\mathsf{Comb}(n)$ makes at most 2^{cn} calls to each collision-finder $\mathsf{Coll}_0, \mathsf{Coll}_1$ and that $\text{FVAL}_0, \text{FVAL}_1$ each contain at most 2^{cn} elements for a constant $c < 1$. Then for any $(iv, m), y$ and $b \in \{0,1\}$ such that $((iv, m), f_b(iv, m)) \notin \text{FVAL}_b \cup \text{CVAL}_b$, we have $\text{Prob}[\, f_b(iv, m) = y \mid \text{FVAL}, \text{CVAL}] \leq 2 \cdot 2^{-n}$ (for sufficiently large n's).*

Proof. Consider the information about the image of a value (iv, m) (not appearing in FVAL $\cup$ CVAL) available through FVAL, CVAL. Suppose that this value (iv, m) appears in the course of a collision search —else the claim already follows because the image is completely undetermined— and thus the image belongs to SAMPLES $\setminus$ (FVAL $\cup$ CVAL). This only leaks the information that the image of (iv, m) must be distinct from other images in such a collision search, or else the collision finder would have output (iv, m) as part of the collision. Hence, the information available through FVAL, CVAL only exclude the images in SAMPLES $\cap$ $(\text{FVAL}_b \cup \text{CVAL}_b)$ —values for the other bit $\bar{b}$ are not relevant— which is a set of size at most $|\text{FVAL}_b \cup \text{CVAL}_b| \leq 3 \cdot 2^{cn}$ (since each of the 2^{cn} calls to Coll_b yields at most two entries in CVAL_b). Thus, for large n's there are at least $2^n - 3 \cdot 2^{cn} \geq \frac{1}{2} \cdot 2^n$ candidate images left, each one being equally like. □

The next lemma says that the collision-finders cannot be used to break pre-image resistance, i.e., despite the ability to find collisions via $\mathsf{Coll}_0, \mathsf{Coll}_1$, searching for a pre-image to a chosen value is still infeasible. Below we formalize this by executing an adversary $\mathcal{B}$ in mode challenge first, in which $\mathcal{B}$ explicitly determines an image y for which a pre-image should be found under f_b. To avoid trivial attacks we also presume that no (iv, m) with $f_b(\text{iv}, m) = y$ has been found up to this point. Then, we continue $\mathcal{B}$'s execution in mode find in which $\mathcal{B}$ tries to find a suitable pre-image (iv, m). This assumes that $\mathcal{B}$ cannot try out too many collision-finder replies (i.e., at most 2^{cn} many for some constant $c < \frac{1}{2}$):

Lemma 3.5 (Chosen Pre-Image Resistance) *For any algorithm $\mathcal{B}$ and any constant $c < \frac{1}{2}$ the following experiment $\mathbf{Exp}^{pre,\mathsf{Comb}}_{\mathcal{B},\mathcal{H}_0,\mathcal{H}_1,\mathsf{Coll}_0,\mathsf{Coll}_1}(n)$ has negligible probability of returning 1:*

Experiment $\mathbf{Exp}^{pre,\mathsf{Comb}}_{\mathcal{B},\mathcal{H}_0,\mathcal{H}_1,\mathsf{Coll}_0,\mathsf{Coll}_1}(n)$*:*
initialize $(f_0, IV_0) \leftarrow \mathsf{HKGen}_0(1^n)$, $(f_1, IV_1) \leftarrow \mathsf{HKGen}_1(1^n)$
let $(y, b, state) \leftarrow \mathcal{B}^{f_0,f_1,\mathsf{Coll}_0,\mathsf{Coll}_1}(\mathsf{challenge}, IV_0, IV_1)$
let $\text{VAL}^{ch}_b = \text{FVAL}_b \cup \text{CVAL}_b$ *at this point*
let $(iv, m) \leftarrow \mathcal{B}^{f_0,f_1,\mathsf{Coll}_0,\mathsf{Coll}_1}(\mathsf{find}, state)$
return 1 *iff*

$f_b(iv, m) = y$ and $((iv, m), y) \notin \text{VAL}_b^{\mathsf{ch}}$, and
$\mathcal{B}$ made at most 2^{cn} calls to collision-finder Coll_b (in both phases together), and no lucky collisions occurred during $\mathcal{B}$'s computation (in both phases together)

The proof idea is as follows. For any value appearing in $\text{FVAL}_b \setminus \text{CVAL}_b$ during the find phase the probability of matching y is at most $2 \cdot 2^{-n}$ by the image uncertainty. Furthermore, according to the Birthday Lemma 3.3 the set FVAL_b cannot contain more than 2^{dn} elements for some $d > \frac{1}{2}$ (or else a lucky collision is very likely). But then the probability of finding another pre-image among those values is negligible.

The harder part is to show that $\mathcal{B}$ cannot significantly influence the collision finder Coll_b to search for a collision with image y (which would then appear in CVAL_b and could be output by $\mathcal{B}$). Here we use the property of our model saying that the circuit's output $C_b(s)$ for each sample is essentially determined by $\mathcal{B}$ (or, to be precise, by the previous values in FVAL and CVAL). But then the Image Uncertainty Lemma applies again, and each sample $C_b(s)$ yields y with probability at most $2 \cdot 2^{-n}$. The final step is to note that each collision search most likely requires approximately $2^{\frac{n}{2}}$ or less samples, and $\mathcal{B}$ initiates at most 2^{cn} many searches for $c < \frac{1}{2}$. Hence, with overwhelming probability there is no value with image y in SAMPLES in the find phase at all, and thus no such value in CVAL_b. This shows Chosen Pre-Image Resistance. More formally:

Proof. Let d be a constant with $\frac{1}{2} < d < 1$. Assume that FVAL_b contains more than 2^{dn} elements at the end. Then Lemma 3.3 implies that such executions can only contribute with negligible probability to $\mathcal{B}$'s success. From now on we can therefore condition on this bound 2^{dn} of number on elements in FVAL_b.

By the image uncertainty we can conclude that the probability that any of the values $((\text{iv}, m), f_b(\text{iv}, m)) \in \text{FVAL}_b \setminus \text{CVAL}_b$ in $\mathcal{B}$'s find phase yields y, is at most $2 \cdot 2^{-n}$. Here we use the fact that any function evaluation adding to $\text{FVAL}_b \setminus \text{CVAL}_b$ is either via a direct call to the f_b-box, or via an f_b-gate evaluation in the computation of a circuit $C(s)$, carried out through one of the collision finders. In any case, the input to the function only depends on the values in FVAL and CVAL before the corresponding query; for f_b-box queries this is clear and for circuit computations it follows as the circuit is chosen by $\mathcal{B}$ and all previous function evaluations immediately appear in FVAL_b. Therefore, the uncertainty bound applies. Summing over all at most 2^{dn} many values in FVAL_b shows that the probability of hitting y is bounded from above by $2 \cdot 2^{(d-1)n}$ and is thus negligible. In the sequel we therefore presume that no $((\text{iv}, m), y) \in \text{FVAL}_b \setminus \text{CVAL}_b$ appears (unless it has been in $\text{VAL}_b^{\mathsf{ch}}$ before, in which case $\mathcal{B}$ cannot use it anymore for a successful run).

We next investigate the effect of collision finder calls on CVAL_b, addressing the question if $\mathcal{B}$ can force the collision finder to bias collisions towards y in some way. Recall that the collision finder makes at most 2^{cn} many runs for

$c < \frac{1}{2}$. Let $e = \frac{3}{4} - \frac{c}{2} > \frac{1}{2}$. Then we can assume that each run probes at most 2^{en} new elements previously not in SAMPLES. This is so since, for a single run, the probability of finding no collisions after 2^{en} many trials for fresh values, is double-exponentially small (see Lemma 3.3 and note that this remains true for a slightly larger probability of $2 \cdot 2^{-n}$). The probability that any of the 2^{cn} calls would require more fresh samples, is therefore still negligible. From now on we thus presume that each call adds at most 2^{en} new entries to SAMPLES.

Consider the j-th call C_b to the collision finder Coll_b in the find stage. Let $\text{CVAL}_{b,j}^{\mathsf{before}}$ be the set CVAL_b before this call, such that $\text{CVAL}_{b,1}^{\mathsf{before}}$ denotes the set CVAL_b at the beginning of the find phase. Note that $\text{CVAL}_{b,j}^{\mathsf{before}}$ does not change during the collision search, but only when the finder returns the collision. Suppose further that $\text{CVAL}_{b,j}^{\mathsf{before}}$ does not contain any element $((\text{iv}, m), y)$ which is not already in $\text{VAL}_b^{\mathsf{ch}}$. This is obviously true for $\text{CVAL}_{b,1}^{\mathsf{before}}$.

A crucial aspect in our consideration is that all circuit values $C_b(s)$ during the collision search are fully determined given FVAL_b (containing the pairs of the entire execution but whose images are distinct from y by assumption) as well as $\text{CVAL}_{b,j}^{\mathsf{before}}$. Hence, the uncertainty bound applies again, and the probability that a specific sample $C_b(s)$ gives a new pair $(C_b(s), y) \notin \text{CVAL}_{b,j}^{\mathsf{before}} \cup \text{VAL}_b^{\mathsf{ch}}$, is at most $2 \cdot 2^{-n}$ (noting that any entry $(C_b(s), f_b(C_b(s))) \in (\text{FVAL}_b \cup \text{CVAL}_{b,j}^{\mathsf{before}}) \setminus \text{VAL}_b^{\mathsf{ch}}$ has an image different from y by assumption). Since there are at most 2^{en} new samples, only with probability at most $2 \cdot 2^{(e-1)n}$ some new sample $C_b(s)$ in Coll_b's search yields y. It follows that, except with probability $2 \cdot 2^{(e-1)n}$, the set $\text{CVAL}_{b,j+1}^{\mathsf{before}}$ including the new collisions will not contain a suitable entry.

Finally, sum over all at most 2^{cn} many calls to Coll_b to derive that CVAL_b does not contain a new entry $((\text{iv}, m), y) \in \text{CVAL}_b \setminus \text{VAL}_b^{\mathsf{ch}}$, except with probability $2 \cdot 2^{(c+e-1)n}$ for $c + e = \frac{3}{4} + \frac{c}{2} < 1$ which is negligible. Since the same holds for $\text{FVAL}_b \setminus \text{CVAL}_b$ the overall probability of finding a suitable preimage (iv_0, m), including possibly the final output which is not a member in $\text{FVAL}_b \cup \text{CVAL}_b$, is negligible. □

For the final conclusions about our model, we prove that, given a collision (iv, m), (iv', m') produced by a collision finder Coll_b, generating another preimage also mapping to $f_b(\text{iv}, m) = f_b(\text{iv}', m')$, is infeasible. The proof is in two steps, first showing that one cannot use the f_b-boxes to find such an additional value, and the second lemma shows that this remains true if one tries to use the collision finder (if one does not call the collision finder more than a polynomial number of times). We remark that this aspect refers to collisions *for the compression functions* only; given a collision generated by the finders one can of course extend this to further collisions *for the iterated hash function* by appending message blocks:

Lemma 3.6 (f-Replication Resistance) *Suppose that the adversary $\mathcal{A}$ during experiment $\mathbf{Exp}^{\textit{amp-comb}}_{\mathcal{A},\mathcal{H}_0,\mathcal{H}_1,\mathsf{Coll}_0,\mathsf{Coll}_1}\mathsf{Comb}(n)$ makes at most 2^{cn} calls to each*

collision-finder $\mathsf{Coll}_0, \mathsf{Coll}_1$ *and that each set* $\text{FVAL}_0, \text{FVAL}_1$ *contains at most* 2^{dn} *elements for constants* c, d *with* $c+d<1$*. Then the probability that there exist values* $((iv,m),y) \in \text{CVAL}_b$ *and* $((iv',m'),y) \in \text{FVAL}_b \setminus \text{CVAL}_b$ *for* $b \in \{0,1\}$*, is negligible.*

Proof. Fix a bit b. Since $\mathcal{A}$ makes at most 2^{cn} calls to Coll_b and each reply returns two elements, the set CVAL_b is of size at most $2 \cdot 2^{cn}$. Consider any value $((\text{iv},m),y) \in \text{CVAL}_b$ and any value $((\text{iv}',m'),y') \in \text{FVAL}_b \setminus \text{CVAL}_b$. Then, because $((\text{iv}',m'),y') \notin \text{CVAL}_b$, we must have $y' \neq y$ or $(\text{iv},m) \neq (\text{iv}',m')$. In the first case we have no match, in the second case a match can occur with probability at most $2 \cdot 2^{-n}$ by the image uncertainty (considering the point in the execution where the second of the two values appears for the first time).

Now sum over all $2 \cdot 2^{cn} \cdot 2^{dn} = 2 \cdot 2^{(c+d)n}$ combinations, such that the probability of finding any match is at most $4 \cdot 2^{(c+d-1)n}$. Since $c+d<1$ this is negligible, and stays negligible if we sum over both choices for b. □

Note that the fact above indicates that, after having generated collisions through the finder, finding other matching function values through the f-boxes is infeasible. This holds at any point in the execution, i.e., $\mathcal{A}$ may not even successfully produce a collision but rather stop prematurely. Next, we use this fact (together with chosen pre-image resistance) to prove replication resistance with respect to the collision finders:

Lemma 3.7 (Coll-Replication Resistance) *Suppose that the adversary* $\mathcal{A}$ *during experiment* $\mathbf{Exp}^{\textit{amp-comb}}_{\mathcal{A},\mathcal{H}_0,\mathcal{H}_1,\mathsf{Coll}_0,\mathsf{Coll}_1}\mathsf{Comb}(n)$ *makes at most poly*(n) *calls to each collision-finder* $\mathsf{Coll}_0, \mathsf{Coll}_1$ *and that* $\text{FVAL}_0, \text{FVAL}_1$ *each contain at most* 2^{dn} *elements for a constant* $d<1$*. Then the probability that there exist values* $((iv,m),y)$*,* $((iv',m'),y)$*,* $((iv^*,m^*),y) \in \text{CVAL}_b$ *for* $b \in \{0,1\}$ *with pairwise distinct* $(iv,m),(iv',m'),(iv^*,m^*)$*, is negligible.*

Proof. We discuss that if $\mathcal{A}$ could find three (or more) of those values then this would contradict either f-replication resistance or chosen pre-image resistance. Consider adversary $\mathcal{B}$ against the chosen pre-image resistance which basically runs a black-box simulation of $\mathcal{A}$. In the challenge-phase, $\mathcal{B}$ initially makes a guess for a specific call j adversary $\mathcal{A}$ makes to one of the collision finders. Then $\mathcal{B}$ runs $\mathcal{A}$ up to the point where $\mathcal{A}$ receives the answer $((\text{iv},m),y)$, $((\widehat{\text{iv}},\widehat{m}),y)$ of Coll_b for this j-th call. Then $\mathcal{B}$ outputs y,b (and all internal information of $\mathcal{A}$ as state) and concludes this stage. In the find-phase $\mathcal{B}$ continues $\mathcal{A}$'s simulation and waits to see a value $((\text{iv}^*,m^*),y)$ in the execution, and then outputs (iv^*,m^*) and stops.

We next analyze $\mathcal{B}$'s success probability. Since each call to the collision-finders adds at most two new values to CVAL_b, there must be a point in $\mathcal{A}$'s execution where there is $(\text{iv},m) \in \text{CVAL}_b$ (and possibly $(\text{iv}',m') \in \text{CVAL}_b$) and only the next call to Coll_b adds the value (iv^*,m^*) to CVAL_b, i.e., so far $(\text{iv}^*,m^*) \notin \text{CVAL}_b$. Suppose that the conditional probability (given such a

value (iv^*, m^*) with the same image really appears in the execution) that this value belongs to FVAL_b after the corresponding call to Coll_b, was noticeable. Then this would clearly contradict the f-replication resistance (bounding the polynomial number of calls by 2^{cn} for the constant $c = \frac{1}{2} - \frac{d}{2}$ with $c + d < 1$). We may therefore assume that $(\text{iv}^*, m^*) \notin \text{VAL}_b^{\mathsf{ch}} = \text{CVAL}_b \cup \text{FVAL}_b$ at this point. But then $\mathcal{B}$ guesses the right call j with probability $1/\text{poly}(n)$, and thus predicts a function value with noticeable probability. This, however, contradicts the chosen pre-image resistance. □

3.5 A Security-Amplifying Combiner

Our (input-restricted) security-amplifying combiner takes messages $M = m_0 || \ldots || m_{t-1}$ of exactly t blocks with $t \leq en$ for some constant $e < \frac{1}{4}$ and applies each of the two hash functions H_0, H_1 to the message $m_0 || \ldots || m_t$ and outputs the concatenation:

Theorem 3.8 *Let $\mathcal{H}_0, \mathcal{H}_1$ be idealized Merkle-Damgård hash functions. Let $e < \frac{1}{4}$ be a constant and assume that $t \leq en$. Then the combiner*

$$\mathsf{Comb}_{amp,t}^{H_0,H_1}(M) = H_0(m_0 || \ldots || m_{t-1}) \,||\, H_1(m_0 || \ldots || m_{t-1})$$

of $\mathcal{H}_0$ and $\mathcal{H}_1$ is $\alpha(n)$-security-amplifying for $\alpha(n) = \text{poly}(n)$ if the adversary in experiment $\mathbf{Exp}^{amp\text{-}comb}_{\mathcal{A},\mathcal{H}_0,\mathcal{H}_1,\mathsf{Coll}_0,\mathsf{Coll}_1}\mathsf{Comb}_{amp,t}(n)$ makes at most $\alpha(n) = \text{poly}(n)$ calls to each collision finder.

We also remark that our combiner is obviously a (classically) secure combiner in the non-idealized setting. The theorem shows that we get the improved security-amplification guarantee against attacks in the idealized world.

For the proof idea it is instructive to investigate why the straightforward application of the attack of Joux for the case of at most $t \leq \frac{n}{4}$ message blocks fails. In this case one would again build a multi-collision set for either hash function of size at most $2^t \leq 2^{\frac{n}{4}}$. But this time the probability that any of the $2^{2t} < 2^{\frac{n}{2}}$ pairs in such a multi-collision set also collides under the other hash function, should be approximately $2^{\frac{n}{2}} \cdot 2^{-n} = 2^{-\frac{n}{2}}$. Most likely, even approximately $2^{\frac{n}{2}}$ multi-collsion sets should therefore not help to find a collision under both hash functions. Our proof follows these lines of reasoning, i.e., bounding the size of multi-collision sets and the probability that message pairs in such a multi-collision set also collide under the other hash function. We stress, however, that a full proof in our model still needs to deal with more general adversaries, possibly taking advantage of the collision finders through "clever" queries.

To process messages of arbitrary length without losing the property of security-amplification we apply a hash-tree construction [Mer89] to our combiner. Below we outline the somewhat standard construction for $t = 2$. For

a similar and more formal treatment see for instance [BR97]. An example is given in Figure 3.3.

One first divides and possibly pads the message M into blocks m_j, $j = 0, 1, \ldots, k-1$, of l bits each (say, by appending $10\ldots0$ for a sufficient number of 0's). Now build a hash tree recursively by putting the message blocks in the leaves and applying our basic combiner in each node to its two children. Specifically, let $h_j^0 = m_j$, $k^0 = k$ and, for each $i = 1, 2, \ldots, \lceil \log_2 k \rceil$, let

$$\begin{aligned} k^i &= \lceil k^{i-1}/2 \rceil \quad \text{and} \\ h_j^i &= \begin{cases} \mathsf{Comb}_{\mathrm{amp},2}^{H_0,H_1}(h_{2j}^{i-1} || h_{2j+1}^{i-1}) & \text{for } j = 0, 1, \ldots, \lfloor k^{i-1}/2 \rfloor - 1 \\ h_{2j}^{i-1} & \text{if } j = \lfloor k^{i-1}/2 \rfloor = \lceil k^{i-1}/2 \rceil - 1 \end{cases} \end{aligned}$$

where we pad each h_j^i to l bits if necessary. In particular, we must have that the output length of our combiner satisfies $2n \leq l$, which is not a significant restriction for common hash functions today. Note that we simply lift values at "odd" positions to the next level. Finally, we apply the basic combiner once more to the final value $h_0^{\lceil \log_2 k \rceil}$ and the length $|M|$ of the original message, given as an l-bit string.

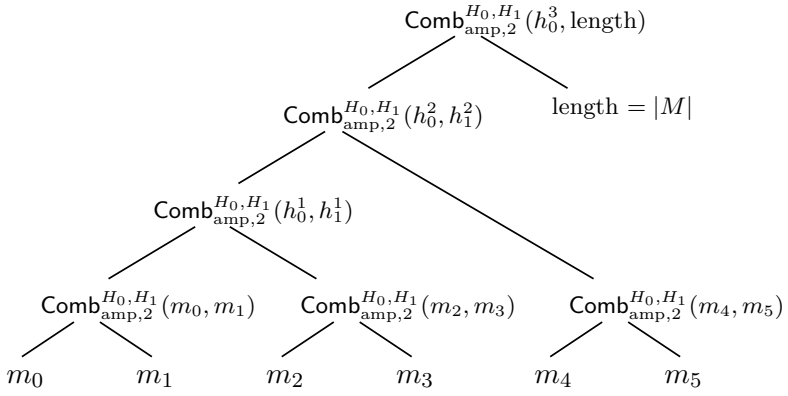


Figure 3.3: Example of a hash tree construction for our combiner ($t = 2, k = 6$)

If two messages $M \neq M'$ lead to a collision in the root of the hash tree, it can be either the result of a non-trivial collision in the final application of the combiner for different message lengths $|M| \neq |M'|$ (in which case we get a non-trivial collision for the basic combiner), or else the tree structures must be identical. In the latter case the collision can always be traced back to a collision for an earlier application of the combiner. Hence, in both cases the reason for the tree collision is at least one collision for the basic combiner.

As for the efficiency, for a full t-ary tree (with $k = t^r$, the number of message blocks, being a power of t) we apply our basic combiner $\frac{k-1}{t-1} + 1$ times. Each time we need $2t$ applications of the compression functions, making our

solution about $\frac{t}{t-1}$ times slower than the classical combiner with $2k$ applications (but with the advantage of security amplification for our combiner).

3.6 Proof of Security Amplification

Before giving the proof we first show a technical conclusion stating that the adversary against our (input-restricted) combiner essentially cannot win if the function values of the output are undetermined:

Lemma 3.9 (Output Knowledge) *Assume the adversary $\mathcal{A}$ in experiment $\mathbf{Exp}^{amp\text{-}comb}_{\mathcal{A},\mathcal{H}_0,\mathcal{H}_1,\mathsf{Coll}_0,\mathsf{Coll}_1}\mathsf{Comb}_{amp,t}(n)$ makes at most 2^{cn} calls to each collision-finder $\mathsf{Coll}_0, \mathsf{Coll}_1$ for some constant $c < 1$. Assume that $\mathcal{A}$ eventually outputs $M = m_0||\ldots||m_{t-1} \neq M' = m'_0||\ldots||m'_{t-1}$ such that*

$$iv_{b,0} = iv'_{b,0} = IV_b, \quad iv_{b,i+1} = f_b(iv_{b,i}, m_i),$$
$$iv'_{b,i+1} = f_b(iv'_{b,i}, m'_i) \quad \text{for } b \in \{0,1\},\ i \in \{0,1,\ldots,t-1\}$$

Suppose further that $((iv_{b,i}, m_i), iv_{b,i+1})$ or $((iv'_{b,i}, m'_i), iv'_{b,i+1}))$ does not belong to $\text{FVAL}_b \cup \text{CVAL}_b$ for some $b \in \{0,1\}$ and some $i \in \{0,1,\ldots,t-1\}$. Then the probability that the experiment returns 1 is negligible.

Proof. Suppose $\mathcal{A}$ outputs such values M, M' and succeeds with noticeable probability. Assume for simplicity that $((\text{iv}_{b,i}, m_i), \text{iv}_{b,i+1}) \notin \text{FVAL}_b \cup \text{CVAL}_b$; the case $((\text{iv}'_{b,i}, m'_i), \text{iv}'_{b,i+1})$ is treated analogously. Let i be maximal and fix the bit b.

By Lemma 3.3 we can assume $|\text{FVAL}_b| \leq 2^{dn}$ for $d = \max\{\frac{3}{4}, c\}$, except with negligible probability. Hence, from now on we can condition on $|\text{FVAL}_b \cup \text{CVAL}_b| \leq 3 \cdot 2^{dn}$. For a success the messages M and M' must collide under H_b. If $i = t-1$ then $f_b(\text{iv}_{b,i}, m_i) = \text{iv}_{b,i+1}$ is the output of the hash function, and since this value does not appear in $\text{FVAL}_b \cup \text{CVAL}_b$, the probability of matching $\text{iv}'_{b,i+1}$ is bounded from above by $2 \cdot 2^{-n}$ by the image uncertainty.

If $i < t-1$ then there must exist an entry $((\text{iv}_{b,i+1}, m_{i+1}), \text{iv}_{b,i+2}) \in \text{FVAL}_b \cup \text{CVAL}_b$ (because i is chosen to be maximal). However, the probability that the value $f_b(\text{iv}_{b,i}, m_i)$ appears as a prefix in any of the $3 \cdot 2^{dn}$ values in $\text{FVAL}_b \cup \text{CVAL}_b$, is at most $6 \cdot 2^{(d-1)n}$ and thus negligible. On the other hand, if the prefix $f_b(\text{iv}_{b,i}, m_i)$ does not appear in $\text{FVAL}_b \cup \text{CVAL}_b$, then this contradicts the maximal choice of i. Doubling the probability for both choices of b concludes the proof. □

The following lemma proves that, for t message blocks there can only be 2^t multi-collisions, as long as each collision finder is only called a polynomial number of times:

Lemma 3.10 (Multi-Collisions) *Assume that the attacker $\mathcal{A}$ in experiment $\mathbf{Exp}^{amp\text{-}comb}_{\mathcal{A},\mathcal{H}_0,\mathcal{H}_1,\mathsf{Coll}_0,\mathsf{Coll}_1}\mathsf{Comb}_{amp,t}(n)$ makes at most poly(n) calls to each collision-finder $\mathsf{Coll}_0, \mathsf{Coll}_1$ and that the experiment returns 1. Then, the probability that for some $b \in \{0,1\}$ and some $iv_{b,t}$, the set*

$$\text{MULTI}_b(iv_{b,t}) = \left\{ M = m_0||\ldots||m_{t-1} \;:\; \begin{array}{l} iv_{b,i+1} = f_b(iv_{b,i}, m_i) \in \text{FVAL}_b \cup \text{CVAL}_b \\ \text{for } i = 0,1,\ldots,t-1, \text{ where } iv_{b,0} = IV_b \end{array} \right\}$$

contains more than 2^t elements, is negligible.

Proof. Assume that the experiment returns 1 (such that, except with negligible probability, $\text{FVAL}_0, \text{FVAL}_1$ are of size at most 2^{dn} each, for some constant $d < 1$). If some set $\text{MULTI}_b(\text{iv}_{b,t})$ contains more than 2^t elements then there must be an index i such that there are (at least) three distinct values $(\text{iv}_{b,i}, m_i)$, $(\text{iv}'_{b,i}, m'_i)$ and $(\text{iv}^*_{b,i}, m^*_i)$ mapping to the same image under f_b. If two or more of those values belong to $\text{FVAL}_b \setminus \text{CVAL}_b$ then this constitutes a lucky collision and refutes the fact that the experiment returns 1. If one of the values lies in $\text{FVAL}_b \setminus \text{CVAL}_b$, whereas the other two values belong to CVAL_b, then this contradicts the f-replication resistance and this can only happen with negligible probability. Finally, the case that all three values belong to CVAL_b can only happen with negligible probability, too, under the Coll-replication resistance. □

With these two lemmas we can now prove that our combiner is security-amplifying. For an outline consider the multi-collision sets defined in the previous lemma. Lemma 3.9 implies that, in order to win, the adversary must know the images of the final output $M \neq M'$. Hence, each of the two messages must appear in some multi-collision set, and to constitute a collision under hash function H_b, they must appear in the same multi-collision set $\text{MULTI}_b(y_b)$ for some y_b. Moreover, since the messages must collide under both hash functions simultaneously they must belong to an intersection $\text{MULTI}_0(y_0) \cap \text{MULTI}_1(y_1)$ for some y_0, y_1.

Lemma 3.10 now says that each multi-collision set has at most 2^t elements. Thus, there are at most $2^{2t} \leq 2^{2en}$ such pairs in each multi-collision set. Furthermore, we can bound the number of multi-collision sets by the number of elements in $\text{FVAL}_b \cup \text{CVAL}_b$, and therefore by $3 \cdot 2^{dn}$ for a constant $d > \frac{1}{2}$ with $d + 2e < 1$ (here we use the fact that $e < \frac{1}{4}$). We therefore have at most $3 \cdot 2^{(d+2e)n}$ possible pairs $M \neq M'$. The proof then shows that, by the image uncertainty, any of the pairs M, M' in some multi-collision set $\text{MULTI}_b(y_b)$ also collides under the other hash function $H_{\bar{b}}$, with probability at most $6 \cdot 2^{(d+2e-1)n}$ which is negligible. Put differently, with overwhelming probability the intersections of multi-collision sets for both hash functions are empty and the adversary cannot find appropriate messages M, M'.

Finally, we give the full proof that our combiner is security amplifying:

Proof (of Theorem 3.8). According to our definition a combiner is called security-amplifying if for any algorithm $\mathcal{A}$ making at most $\alpha(n)\cdot(T_0(n)+T_1(n))$ steps the probability of finding a collision is negligible (for some $\alpha(n) > 1$). Hence we will show that, with overwhelming probability, no collisions for $\mathsf{Comb}_{\mathrm{amp},t}$ (with $t < en$ for constant $e < \frac{1}{4}$) can be computed for any $\alpha(n) = \mathrm{poly}(n)$ when calling each collision finders at most $\alpha(n) = \mathrm{poly}(n)$ many times.

Let $d = \frac{3}{4} - e$ such that the constant d is at larger than $\frac{1}{2}$ and $d + 2e < 1$. Then we can assume that $\mathrm{FVAL}_0, \mathrm{FVAL}_1$ in $\mathcal{A}$'s attack each contain at most 2^{dn} elements, otherwise the probability of winning would be negligible. Also assume that the number of collision finder calls is bounded by $2 \cdot \mathrm{poly}(n) \leq 2^{dn}$ (for sufficiently large n's). Hence, in the following, we can assume that $\mathrm{FVAL}_b \cup \mathrm{CVAL}_b$ contains at most $3 \cdot 2^{dn}$ many elements for $b \in \{0,1\}$.

For any $b \in \{0,1\}$ and any $\mathrm{iv}_{b,t}$ we again consider all sets of multi-collisions $\mathrm{MULTI}_b(\mathrm{iv}_{b,t})$, but this time we divide them into different stages (depending on the calls to the collision finders). We denote by $\mathrm{MULTI}_{b,j}^{\mathsf{before}}(y)$ the set of multi-collisions *before* the j-th call to one of the two collision finders. The transition to the next phase therefore adds all messages with respect to the new function values from the collision finder's reply as well as all subsequent function evaluations through the f-boxes. Clearly, $\mathrm{MULTI}_{b,j}^{\mathsf{before}}(y) \subseteq \mathrm{MULTI}_{b,j+1}^{\mathsf{before}}(y)$ for all j and $\mathrm{MULTI}_{b,2\cdot\mathrm{poly}(n)+1}^{\mathsf{before}}(y)$ —which we denote by $\mathrm{MULTI}_b^{\mathsf{end}}(y)$— contains all multi-collisions for y under H_b at the end of the execution.

By Lemma 3.9 adversary $\mathcal{A}$ must "know" all function values in the final output, i.e., they must belong to $\mathrm{FVAL}_b \cup \mathrm{CVAL}_b$ for some $b \in \{0,1\}$. Hence, both messages of the collision $M \neq M'$ for H_b output by $\mathcal{A}$ must also appear in the same set $\mathrm{MULTI}_b^{\mathsf{end}}(y_b)$ for some y_b. This basically reduces the task of showing that $\mathcal{A}$ fails, to the proof that no $M \neq M'$ and y_0, y_1 with $M, M' \in \mathrm{MULTI}_0^{\mathsf{end}}(y_0) \cap \mathrm{MULTI}_1^{\mathsf{end}}(y_1)$ exist (except with some very small probability or if one of the success requirements such as the absence of lucky collisions is violated).

We will show that, given that no success requirements are violated, with overwhelming probability the intersection of multi-collision sets for $b = 0, 1$ will be empty in the course of the execution. This is done by a careful inductive argument, where we use the invariant that for no y_b the set $\mathrm{MULTI}_{b,j}^{\mathsf{before}}(y_b)$ contains $M \neq M'$ such that they collide under $H_{\overline{b}}$. This is clearly true for $\mathrm{MULTI}_{b,1}^{\mathsf{before}}(y_b)$ because up to the point where the first collision finder is called, only f-queries have been made, and each set $\mathrm{MULTI}_{b,1}^{\mathsf{before}}(y_b)$ can contain only one element (or a lucky collision already occurs).

We also use that, according to the Multi-Collision Lemma 3.10, each set $\mathrm{MULTI}_{b,j}^{\mathsf{before}}(y_b)$ can contain at most 2^t elements (with overwhelming probability). Additionally, we always have at most $3 \cdot 2^{dn}$ non-empty multi-collision sets, because there can only be an element in a such set if there is at least one value from $\mathrm{FVAL}_b \cup \mathrm{CVAL}_b$. Hence, at any point there are at most $2^{2t} \cdot 3 \cdot 2^{dn} \leq 3 \cdot 2^{(d+2e)n}$ many collision pairs (M, M') appearing together

in one of the multi-collision sets, for the constant $d + 2e < 1$.

Now suppose we make the j-th call to one of the collision finders, Coll_b. After this call (and all subsequent f-function evaluations) take any pair $M \neq M'$ belonging to the same set $\text{MULTI}^{\mathsf{before}}_{b,j+1}(y_b)$ for some y_b. The next step is to note that, most likely, this pair M, M' cannot belong to some $\text{MULTI}^{\mathsf{before}}_{\overline{b},j+1}(y_{\overline{b}})$. Note that if M and M' lie in multi-collision sets $\text{MULTI}^{\mathsf{before}}_{\overline{b},j+1}(y_{\overline{b}})$ and $\text{MULTI}^{\mathsf{before}}_{\overline{b},j+1}(y'_{\overline{b}})$ for $y_{\overline{b}} \neq y'_{\overline{b}}$ then they clearly do not collide under $H_{\overline{b}}$ as those sets must be disjoint.

Assume, towards contradiction, that M, M' appear in a single multi-collision set for $\overline{b}$. We already know that M, M' cannot belong to some $\text{MULTI}^{\mathsf{before}}_{\overline{b},j}(y_{\overline{b}})$ of the previous stage, because none of these pairs constitutes a collision under $H_{\overline{b}}$, except with negligible probability. Hence, at least one of the two messages (say, M) must have been added to $\text{MULTI}^{\mathsf{before}}_{\overline{b},j+1}(y_{\overline{b}})$ because of an $f_{\overline{b}}$-function evaluation of Coll_b or via a direct evaluation of $f_{\overline{b}}$, taking into account that $\text{CVAL}_{\overline{b}}$ does not change between the two points in time.

Suppose that M is added to some set $\text{MULTI}^{\mathsf{before}}_{\overline{b},j+1}(y_{\overline{b}})$ via a new $f_{\overline{b}}$-value (which has not been in $\text{CVAL}_{\overline{b}}$), and assume that either M' is added only now or has already been in this set before the call. Consider the maximal i for which a new function value is added (when one would process the blocks m_i of message M through the iterated hash function). If the final value $\text{iv}_{\overline{b},t} = f_{\overline{b}}(\text{iv}_{\overline{b},t-1}, m_{t-1})$ is added ($i = t-1$) then, if for M' processing the final message block $\text{iv}_{\overline{b},t} = f_{\overline{b}}(\text{iv}'_{\overline{b},t-1}, m'_{t-1})$ has been in $\text{FVAL}_{\overline{b}}$ before or is added to $\text{FVAL}_{\overline{b}}$ now, we would have a lucky collision. So $\text{iv}_{\overline{b},t} = f_{\overline{b}}(\text{iv}'_{\overline{b},t-1}, m'_{t-1})$ must have been in $\text{CVAL}_{\overline{b}}$ before. But then this would contradict the f-replication resistance. For any other $i < t-1$ we note that, if $f_{\overline{b}}(\text{iv}_{\overline{b},j}, m_i)$ has not been determined before by $\mathcal{A}$, the probability that it matches any prefix of the at most $3 \cdot 2^{dn}$ previous values in $\text{FVAL}_{\overline{b}} \cup \text{CVAL}_{\overline{b}}$, is negligible (namely, at most $6 \cdot 2^{(d-1)n}$ by the image uncertainty). But this would contradict the maximal choice of i.

In conclusion, for any of the pairs M, M' there must still be an $f_{\overline{b}}$-value not in $\text{FVAL}_{\overline{b}} \cup \text{CVAL}_{\overline{b}}$ at this point, and the probability that the pair M, M' collides under $H_{\overline{b}}$ at all, is thus at most $2 \cdot 2^{-n}$. Therefore, the probability that any of the at most $3 \cdot 2^{(d+2e)n}$ pairs M, M' for $d + 2e < 1$ constitutes a collision under $H_{\overline{b}}$, is negligible. The same argument applies now vice versa, no pair M, M' from a set $\text{MULTI}^{\mathsf{before}}_{\overline{b},j+1}(y_{\overline{b}})$ yields a collision under H_b, except for some negligible error. This gives us the invariant.

The argument can now be set forth to the at most $2 \cdot \text{poly}(n) + 1$ many phases, showing that the final multi-collision sets for $b = 0, 1$ never intersect in more than one element. This proves the theorem. □

Chapter 4

Multi-Property Robustness

This chapter discusses robust combiners that simultaneously preserve multiple properties such as (target) collision-resistance (CR, TCR), pseudorandomness (PRF), message authentication (MAC), one-wayness (OW) or indifferentiability from a random oracle (IRO), from the underlying hash functions.

We start by defining three notions of multi-property robustness (MPR) for combiners in Section 4.2. In Section 4.3 we give the construction of our most efficient MPR combiner that is robust for CR, TCR, PRF and MAC according to our strongest notion. A combiner which additionally preserves the IRO property, slightly increasing the output length and computational costs, is then discussed in Section 4.4. In Section 4.5 we show that a twist on our combiners also makes them robust for one-wayness (but at the price of a fixed input length). Section 4.6 deals again with the different notions of multi-property robustness by showing the correlations between the three variants. We finally address the issue of composing combiners resp. multi-hash combiners in Section 4.7.

The results in this chapter are based on joint work with Marc Fischlin and Krzysztof Pietrzak which appeared in [FL08, FLP08].

4.1 Introduction

Nowadays hash functions are often deployed in many facets, e.g., as pseudorandom functions in TLS or message authentication codes in IPSec. In some standardized protocols as RSA-OAEP [BR94] and RSA-PSS [BR96], even stronger assumptions on the underlying hash-functions are made [BF05, BF06]. A further example for the need of multiple properties is given by Katz and Shin [KS05], where collision-resistant pseudorandom functions are required in order to protect authenticated group key exchange protocols against insider

attacks.[1]

Adhering to the usage of hash functions as "swiss army knives" Bellare and Ristenpart [BR06a, BR07] have shown how to preserve multiple properties in the design of hash functions. In contrast to their approach, which starts with a compression function and aims at constructing a single multi-property preserving hash function, a combiner takes two full-grown hash functions and tries to build a hash function which should preserve the properties, even if one of the underlying hash functions is already broken.

While the concatenation combiner (see Section 2.4.1 for details) preserves the CR, TCR and MAC property, the property of being a PRF is in general not conserved. In contrast, the "XOR combiner" $\mathsf{Comb}_{\oplus}^{H_0,H_1}(M) = H_0(M) \oplus M_1(M)$ is robust with respect to PRF, and also for indifferentiability from a random oracle (IRO), but neither preserves the CR nor the TCR property. Ideally, however, one would like to have a single combiner preserving many properties simultaneously.

In this chapter we show how to build combiners that provably preserve multiple properties in a robust manner. We concentrate on the most common properties as proposed in [BR07], namely, collision-resistance (CR), target collision-resistance (TCR), pseudorandomness (PRF), message authentication (MAC), one-wayness (OW) and indifferentiability from a random oracle (IRO). For formal definitions of those properties we refer to Section 2.3.

The Combiner $\mathsf{Comb}_{4\mathsf{P}}$. We first propose a combiner $\mathsf{Comb}_{4\mathsf{P}}$ with optimal output length of $2n$ bits which robustly preserves the four properties collision-resistance, target collision-resistance, pseudorandomness and message authentication. The basic idea of this construction is to use the concatenation combiner $\mathsf{Comb}_{||}$, and to apply a three-round Feistel permutation to its output. In the first round of the Feistel permutation no round function is applied, whereas the two subsequent rounds are constructed by using the XOR-combiner $\mathsf{Comb}_{\oplus}$ (cf. Figure 4.1). The round functions are made somewhat independent by prepending the round number to the input.

The rationale here is that applying the Feistel (or any other) permutation to the output of $\mathsf{Comb}_{||}$ still preserves the CR, TCR and MAC properties, e.g., collisions for $\mathsf{Comb}_{||}$ are pulled through the downstream permutation and can be traced back to collisions for $\mathsf{Comb}_{||}$. At the same time, one achieves robustness for the PRF property. The latter can be seen as follows: if either H_0 or H_1 is pseudorandom, then the round functions in the Feistel network are pseudorandom as $\mathsf{Comb}_{\oplus}$ is a secure combiner for pseudorandom functions. The Luby-Rackoff [LR88] result now states that a three-round Feistel-network, instantiated with quasi independent pseudorandom functions, is a pseudorandom permutation. We note that the formal argument also needs to take into

[1] Technically, they require *statistical* collision-resistance for the keys of the pseudorandom function.

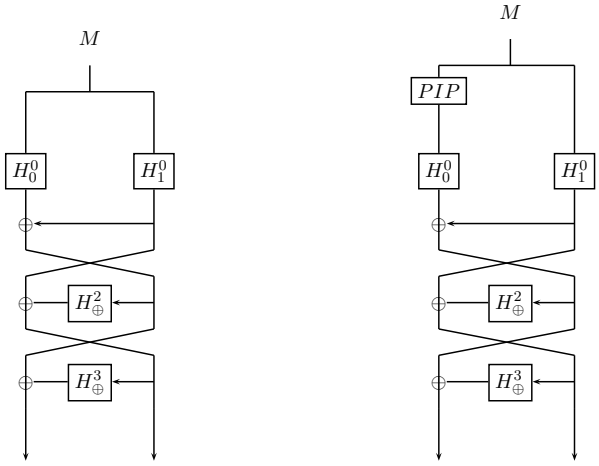


Figure 4.1: Illustration of the basic construction $\mathsf{Comb}_{\mathsf{4P}}$ (left) preserving $\mathsf{CR}, \mathsf{PRF}, \mathsf{TCR}$ and MAC. Here $H_b^i(\cdot)$ denotes $H_b(\langle i \rangle_2 \| \cdot)$ where $\langle i \rangle_2$ is the binary representation of the integer i with two bits. $H_\oplus^i(\cdot)$ denotes $H_0^i(\cdot) \oplus H_1^i(\cdot)$. By applying a pairwise independent permutation (PIP) to the input of H_0^0 we get our construction $\mathsf{Comb}_{\mathsf{4P\&OW}}$ (right), which also preserves OW. Because of the PIP, the input length of the construction must now be fixed.

account that finding collisions in the keyed version of the initial $\mathsf{Comb}_{||}$ computation is infeasible.

Preserving IRO. In Section 4.4 we modify the $\mathsf{Comb}_{\mathsf{4P}}$ construction such that it also preserves indifferentiability from a random oracle. The obstruction of the IRO robustness in the $\mathsf{Comb}_{\mathsf{4P}}$ combiner stems from the invertibility of the Feistel permutation: an adversary trying to distinguish the output of the combiner from a random function (given access to the underlying hash functions, as opposed to the case of pseudorandom functions for example) can partly "reverse engineer" images under the combiner. Hence, we introduce a "signature" value α_M (depending on the input message M), entering the round functions in the Feistel network and basically allowing combiner computations in the forward direction only.

The description of our enhanced combiner $\mathsf{Comb}_{\mathsf{4P\&IRO}}$ is given in Figure 4.2. The signature α_M is taken as (a prefix of) the XOR of the output halves of the $\mathsf{Comb}_{||}$ combiner and is used as additional input parameter in the Feistel round functions, allowing us to also save one round of the Feistel structure. Note that this essentially means that different Feistel permutations may be used for different inputs M, M', because the signatures $\alpha_M, \alpha_{M'}$ may be distinct. In order to apply again the argument that the Feistel permutation does not interfere with the CR,TCR and MAC robustness of the concatenating combiner, we therefore also need to ensure that finding "bad" pairs α_M and $\alpha_{M'}$ is infeasible. To this end we introduce another output branch which basi-

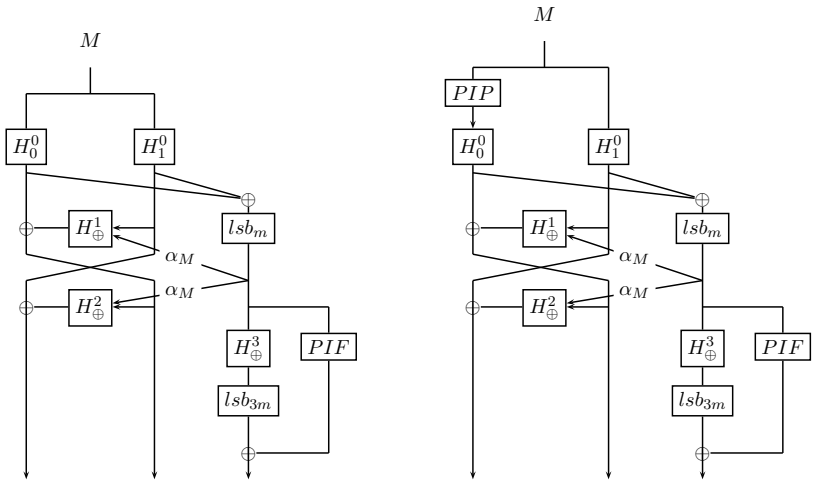


Figure 4.2: Illustration of the construction $\mathsf{Comb}_{\mathsf{4P\&IRO}}$ (left), which (besides the four properties preserved by $\mathsf{Comb}_{\mathsf{4P}}$) also preserves the IRO property, at the prize of an increased output length. The third branch of the construction operates on a signature value α_M depending on input M and applies a pairwise independent function. On the right side the construction $\mathsf{Comb}_{\mathsf{6P}}$ is illustrated which simultaneously preserves all six properties considered.

cally guarantees collision-resistance of the signatures. This additional output is of length $3m$ for some $m = \omega(\log n)$, yielding an overall output length of $2n + \omega(\log n)$.

Preserving One-Wayness. Even though both our solutions are robust for an important set of properties they are not good combiners for one-wayness. Our results so far merely show that they are one-way functions making for example the potentially stronger assumption that one of the two hash functions is collision-resistance. In Section 4.5 we therefore show how to augment our constructions such that they also preserve the one-wayness property.

The idea is that applying a pairwise-independent permutation (PIP) to the input of H_0 (or H_1) in the concatenation combiner $\mathsf{Comb}_{||}$ makes this combiner also robust for one-wayness. Then we can use this modified concatenation combiner in the initial stages of our previous constructions, noting again the subsequent Feistel permutations do not interfere with this property either. Yet, as the description length of a PIP is linear in its input length, the input length of the derived combiners must be fixed, too, giving one-wayness as an additional property.

Weak vs. Strong Robustness. We prove our constructions to be *strongly* robust multi-property combiners for different sets of properties. That is, it suffices that each property is provided by at least one hash function, e.g., if H_0 or H_1 has property MAC, then so does the combiner, independently of the

other properties. We also introduce further relaxations of MPR, denoted by weakly MPR and mildly MPR. In the weak case the combiner only inherits a set of multiple properties if they are all provided by at least one hash function (i.e., if there is a strong candidate which has all properties at the same time). Mildly MPR combiners are between strongly MPR and weakly MPR combiners, where all properties are granted, but different hash functions may cover different properties.

Our work then addresses several questions related to the different notions of multi-property robustness. Namely, we show that strongly MPR is indeed strictly stronger than mildly MPR which, in turn, implies weakly MPR (but not vice versa). We finally discuss the case of general tree-based combiners for more than two hash functions built out of combiners for two hash functions, as suggested in a more general setting by Harnik et al. [HKN+05]. As part of this result we show that such tree-combiners inherit the weakly and strongly MPR property of two-function combiners, whereas mildly MPR two-function combiners surprisingly do not propagate their security to trees.

4.2 Robust Multi-Property Hash Combiners

First, we need to augment the notion of robust combiners, as given in Section 2.4, to the case that multiple properties $\mathsf{P}_1, \mathsf{P}_2, \ldots, \mathsf{P}_N$ instead of a single property P are considered. Recall that a combiner $\mathcal{C} = (\mathsf{CKGen}, \mathsf{Comb})$ for hash functions $\mathcal{H}_0, \mathcal{H}_1$ itself is also a hash function which combines the two functions $\mathcal{H}_0, \mathcal{H}_1$ such that, if at least one of the hash functions obeys property P, then so does the combiner.

For multiple properties $\text{PROP} = \{\mathsf{P}_1, \mathsf{P}_2, \ldots, \mathsf{P}_N\}$ one can either demand that the combiner inherits the properties if one of the candidate hash functions is strong and has all the properties (weakly robust), or that for each property at least one of the two hash functions has the property (strongly robust). We also consider a notion in between but somewhat closer to the weak case, called mildly robust, in which case all properties from PROP must hold, albeit different functions may cover different properties (instead of one function as in the case of weakly robust combiners). In the following, we denote by $\text{PROP}(\mathcal{H}) \subseteq \text{PROP}$ for a set $\text{PROP} = \{\mathsf{P}_1, \mathsf{P}_2, \ldots, \mathsf{P}_N\}$ the properties which a hash function $\mathcal{H}$ has.[2] More formally,

Definition 4.1 (Multi-Property Robustness) *For a set* $\text{PROP} = \{\mathsf{P}_1, \mathsf{P}_2, \ldots, \mathsf{P}_N\}$ *of properties a hash function combiner* $\mathcal{C} = (\mathsf{CKGen}, \mathsf{Comb})$ *for hash functions* $\mathcal{H}_0, \mathcal{H}_1$ *is called:*

weakly multi-property robust *(wMPR) for* PROP *iff*

$$\text{PROP} = \text{PROP}(\mathcal{H}_0) \text{ or } \text{PROP} = \text{PROP}(\mathcal{H}_1) \quad \Longrightarrow \quad \text{PROP} = \text{PROP}(\mathcal{C}),$$

[2] One may also refine these notions further. We focus on these three "natural" cases.

mildly multi-property robust *(mMPR) for* PROP *iff*

$$\text{PROP} = \text{PROP}(\mathcal{H}_0) \cup \text{PROP}(\mathcal{H}_1) \quad \Longrightarrow \quad \text{PROP} = \text{PROP}(\mathcal{C}),$$

strongly multi-property robust *(sMPR) for* PROP *iff for all* $\mathsf{P}_i \in$ PROP*,*

$$\mathsf{P}_i \in \text{PROP}(\mathcal{H}_0) \cup \text{PROP}(\mathcal{H}_1) \quad \Longrightarrow \quad \mathsf{P}_i \in \text{PROP}(\mathcal{C}).$$

We remark that for weak and mild robustness all individual properties $\mathsf{P}_1, \mathsf{P}_2, \ldots, \mathsf{P}_N$ from PROP are guaranteed to hold, either by a single function as in weak robustness, or possibly by different functions as in mild robustness. The combiner may therefore depend on some strong property $\mathsf{P}_i \in$ PROP which one of the hash functions has, and which helps to implement some other property P_j in the combined hash function. But then, for a subset PROP$' \subseteq$ PROP which, for instance, misses this strong property P_i, the combiner may no longer preserve the properties PROP$'$. This is in contrast to strongly robust combiners which support such subsets of properties by definition.

Note that for a singleton PROP $= \{\mathsf{P}\}$ all notions coincide and we simply say that $\mathcal{C}$ is P-robust in this case. However, for two or more properties the notions become strictly stronger from weak to mild to strong, as we show in Section 4.6. Finally, we remark that our definition allows the case $\mathcal{H}_0 = \mathcal{H}_1$, which may require some care when designing combiners, especially if the hash functions are based on random oracles.

4.3 The $\mathcal{C}_{4\mathsf{P}}$ Combiner for CR, PRF, TCR and MAC

In this section we introduce the construction of our basic combiner $\mathcal{C}_{4\mathsf{P}}$ as illustrated in Figure 4.1. Recall that the idea of this combiner is to apply a Feistel permutation (with quasi independent round functions given by the XOR combiner) to the concatenating combiner to ensure CR, PRF, TCR and MAC robustness.

4.3.1 Our Construction

The three-round Feistel permutation P^3 over $\{0,1\}^{2n}$ is given by the round functions $H^i_{\oplus}(\cdot) = H^i_0(\cdot) \oplus H^i_1(\cdot)$ for $i = 2, 3$, with $H^i_b(\cdot)$ denoting the function $H_b(\langle i \rangle_2 \,||\cdot)$ where $\langle i \rangle_2$ is the binary representation of the integer i with two bits. The first round function is the identity function, which we denote for notational convenience as $H^1_{\oplus}(X) = X$. In the i-th round the input (L_i, R_i) is mapped to the output $(R_i, L_i \oplus H^i_{\oplus}(R_i))$. We occasionally denote this Feistel permutation more explicitly by $P^3 = \psi[H^1_{\oplus}, H^2_{\oplus}, H^3_{\oplus}](\cdot)$.

Our combiner, instantiated with hash functions $\mathcal{H}_0, \mathcal{H}_1$, is a pair of efficient algorithms $\mathcal{C}_{4\mathsf{P}} = (\mathsf{CKGen}_{4\mathsf{P}}, \mathsf{Comb}_{4\mathsf{P}})$ where the key generation algorithm $\mathsf{CKGen}_{4\mathsf{P}}(1^n)$ samples $H_0 \leftarrow \mathsf{HKGen}_0(1^n)$ and $H_1 \leftarrow \mathsf{HKGen}_1(1^n)$. The

evaluation algorithm $\mathsf{Comb}_{4\mathsf{P}}^{H_0,H_1}$ for parameters H_0, H_1 and input message M outputs

$$\mathsf{Comb}_{4\mathsf{P}}^{H_0,H_1}(M) = P^3(H_0^0(M)||H_1^0(M)).$$

4.3.2 Multi-Property Robustness

We next show that the construction satisfies the strongest notion for robust multi-property combiners:

Theorem 4.2 *$\mathcal{C}_{4\mathsf{P}}$ is a strongly robust multi-property combiner for* PROP $=$ $\{\mathit{CR}, \mathit{PRF}, \mathit{TCR}, \mathit{MAC}\}$.

Recall that a strong robust multi-property combiner inherits all properties that are provided by at least one of the underlying hash functions. Thus, we have to prove that each property $\mathsf{CR}, \mathsf{PRF}, \mathsf{TCR}$ and MAC is preserved independently.

Lemma 4.3 *The combiner $\mathcal{C}_{4\mathsf{P}}$ is CR-robust.*

Proof. Observe that any collision $M \neq M'$ for $\mathsf{Comb}_{4\mathsf{P}}^{H_0,H_1}(\cdot)$ directly gives a collision $00\|M \neq 00\|M'$ for for $H_0(\cdot)$ and $H_1(\cdot)$. Thus any adversary that finds collisions for $\mathsf{Comb}_{4\mathsf{P}}$ when instantiated with H_0, H_1 with non-negligible probability, can be used to find collision (with the same probability) for H_0 and H_1 respectively: to find a collision for $H_b \leftarrow \mathsf{HKGen}_b(1^n)$ with $b \in \{0,1\}$, run $H_{\bar{b}} \leftarrow \mathsf{HKGen}_{\bar{b}}(1^n)$ and then invoke the adversary on input $H_b, H_{\bar{b}}$. If the adversary outputs a collision for $\mathsf{Comb}_{4\mathsf{P}}^{H_0,H_1}(\cdot)$, this is also a collision for $H_b(\cdot)$. □

Lemma 4.4 *The combiner $\mathcal{C}_{4\mathsf{P}}$ is TCR-robust.*

Proof. Assume towards contradiction that there exist an efficient adversary $\mathcal{A}_{\mathsf{Comb}} = (\mathcal{A}^1_{\mathsf{Comb}}, \mathcal{A}^2_{\mathsf{Comb}})$ that commits to a message M before getting H_0 and H_1 and then finds some M' such that $\mathsf{Comb}_{4\mathsf{P}}^{H_0,H_1}(M) = \mathsf{Comb}_{4\mathsf{P}}^{H_0,H_1}(M')$ with noticeable probability. Then we can use this attacker to construct a successful target-collision adversary $\mathcal{A}_b = (\mathcal{A}^1_b, \mathcal{A}^2_b)$ against the underlying hash functions H_b for $b \in \{0,1\}$ which contradicts the assumption that at least one of the two hash functions is target collision-resistant.

First, the adversary $\mathcal{A}^1_b(1^n)$ runs $\mathcal{A}^1_{\mathsf{Comb}}(1^n)$ to receive the target message M and some state information st. $\mathcal{A}^1_b$ then commits to $00||M$. On input H_b the adversary $\mathcal{A}^2_b$ samples the second hash function $H_{\bar{b}} \leftarrow \mathsf{HKGen}_{\bar{b}}(1^n)$ and passes $H_b, H_{\bar{b}}$ together with (M, st) to $\mathcal{A}^2_{\mathsf{Comb}}$. When $\mathcal{A}^2_{\mathsf{Comb}}$ outputs a message $M' \neq M$ with $\mathsf{Comb}_{4\mathsf{P}}^{H_0,H_1}(M) = \mathsf{Comb}_{4\mathsf{P}}^{H_0,H_1}(M')$ the adversary $\mathcal{A}^2_b$ returns $00||M'$.

Due to the permutation a collision of M, M' for the combiner can be traced back to the input of $P(\cdot)$ and thus we have

$$H_0(00||M)||H_1(00||M) = H_0(00||M')||H_1(00||M').$$

Hence, both adversaries $\mathcal{A}_b$ for $b = 0, 1$ succeed in finding a message $00||M'$ that together with the target message $00||M$ leads to a collision under H_b with the same noticeable probability as $\mathcal{A}_{\mathsf{Comb}}$. □

Lemma 4.5 *The combiner $\mathcal{C}_{4\mathsf{P}}$ is PRF-robust.*

Proof. As the XOR combiner is a good combiner for pseudorandom functions (PRFs), the round functions $H^2_{\oplus}, H^3_{\oplus}$ in the Feistel network are instantiated with PRFs, as long as at least H_0 or H_1 is a PRF. Prepending the unique prefix $\langle i \rangle_2$ for $i = 2, 3$ to the input of $H^i_{\oplus}(\cdot) = H_{\oplus}(\langle i \rangle_2 \,\|\cdot)$ in each round ensures that the functions in different rounds are never invoked on the same input, which means they are indistinguishable from two independent random functions. The first round of our Feistel permutation, that does not apply a round function, simply prepares the input for the second round function $H^2_{\oplus}(\cdot)$ by xoring both input halves $H^0_0(M) \oplus H^0_1(M)$. Thus, if at least one hash function is a PRF then the input to the second round function is already a pseudorandom value, which prevents an adversary from directly choosing the inputs to the second Feistel round.

We can now apply the results due to Luby-Rackoff [LR88] and Naor-Reingold [NR99] which state that a two-round Feistel-network invoked on an unpredictable input and instantiated with independent pseudorandom functions is a pseudorandom permutation (PRP).

Further, if either H_0 or H_1 is a PRF, then the initial concatenation combiner $\mathsf{Comb}^{H_0,H_1}_{||}$ is weakly collision-resistant[3], thus the probability that the adversary will invoke the combiner on distinct inputs M, M' where a collision $H^0_0(M)\|H^0_1(M) = H^0_0(M')\|H^0_1(M')$ occurs, is negligible. So with overwhelming probability, all the adversary sees is the output of a PRP on distinct inputs. This distribution is indistinguishable from uniformly random (this follows from the PRP/PRF switching lemma [BR06b]), thus $\mathcal{C}_{4\mathsf{P}}$ is PRF robust.

More precisely, from any adversary $\mathcal{A}_{\mathsf{Comb}}$ who has advantage ϵ in distinguishing $\mathsf{Comb}^{H_0,H_1}_{4\mathsf{P}}$ making q queries, we can construct an attacker $\mathcal{A}_b$ for $b \in \{0, 1\}$, that distinguishes $H_b \leftarrow \mathsf{HKGen}_b(1^n)$ from random with advantage $\epsilon - \mathcal{O}(q^2/2^n)$. For $b = 0$ (the case $b = 1$ is symmetric) the adversary $\mathcal{A}_0$ first samples $H_1 \leftarrow \mathsf{HKGen}_1(1^n)$ and then simulates the experiment of $\mathcal{A}_{\mathsf{Comb}}$ using this knowledge of H_1 and its oracle access to H_0. Finally, $\mathcal{A}_0$ returns the output of $\mathcal{A}_{\mathsf{Comb}}$. If H_0 is a uniformly random function $f : \{0, 1\}^* \to \{0, 1\}^n$, then any (even computationally unbounded) adversary making q queries has advantage at most $\mathcal{O}(q^2/2^n)$ in distinguishing $\mathsf{Comb}^{f,H_1}_{4\mathsf{P}}$ from a random function (as the advantage from the PRP/PRF switching lemma and the advantage in the Luby-Rackoff result are both $\mathcal{O}(q^2/2^n)$). Thus, if $\mathcal{A}_{\mathsf{Comb}}$ distinguishes

[3]Weak collision-resistance is defined similarly to collision resistance, except that here the function is keyed and the key is secret, i.e., the adversary only gets black-box access to the function.

$\mathsf{Comb}_{4P}^{H_0,H_1}$ from a truly random function $F : \{0,1\}^* \to \{0,1\}^{2n}$ with advantage ϵ, it has advantage $\epsilon - \mathcal{O}(q^2/2^n)$ to distinguish $\mathsf{Comb}_{4P}^{H_0,H_1}$ from $\mathsf{Comb}_{4P}^{f,H_1}$. The latter is by definition also $\mathcal{A}_0$'s advantage for f and H_0.

□

Lemma 4.6 *The combiner $\mathcal{C}_{4P}$ is MAC-robust.*

Proof. Assume towards contradiction that an adversary $\mathcal{A}_{\mathsf{Comb}}$ with oracle access to the combiner $\mathsf{Comb}_{4P}^{H_0,H_1}(\cdot)$ finds with non-negligible probability a valid pair (M,τ), such that $\tau = \mathsf{Comb}_{4P}^{H_0,H_1}(M)$ but the message M was never queried to the MAC-oracle. Given $\mathcal{A}_{\mathsf{Comb}}$ we can construct a successful adversary $\mathcal{A}_b$ against the underlying hash function H_b for $b \in \{0,1\}$. To forge $H_b(\cdot)$, the adversary $\mathcal{A}_b$ first samples $H_{\bar{b}} \leftarrow \mathsf{HKGen}_{\bar{b}}(1^n)$, and then lets $\mathcal{A}_{\mathsf{Comb}}$ attack $\mathsf{Comb}_{4P}^{H_0,H_1}(\cdot)$, and let $\mathcal{A}_b$ use his oracle access to $H_b(\cdot)$ and the knowledge of $H_{\bar{b}}$ to compute the answers to $\mathcal{A}_{\mathsf{Comb}}$'s oracle queries. When finally $\mathcal{A}_{\mathsf{Comb}}$ outputs (M,τ), the adversary $\mathcal{A}_b$ computes its forgery $(00||M,\tau_b)$ by inverting the permutation $P^3 = \psi[H^1_\oplus, H^2_\oplus, H^3_\oplus]$ (recall that $H^i_\oplus(\cdot) = H_0(\langle i\rangle_2 \,||\cdot) \oplus H_1(\langle i\rangle_2 \,||\cdot)$ for $i = 2,3$ and that the required hash function evaluations can be made with the help of the MAC oracle):

$$\tau_0||\tau_1 = P^{3^{-1}}(\tau).$$

The adversary $\mathcal{A}_b$ then outputs the message $00||M$ and τ_b. If M was not previously queried by $\mathcal{A}_{\mathcal{C}}$, then $00||M$ is distinct from all of $\mathcal{A}_b$'s previous queries, because all additional queries are prepended by $\langle i\rangle_2$ where $i \in \{2,3\}$. By construction, if (M,τ) is a valid forgery for $\mathsf{Comb}_{4P}^{H_0,H_1}(\cdot)$, then $H_0^0(M)||H_1^0(M) = \tau_0||\tau_1$ and thus $(00\|M,\tau_b)$ is a valid forgery for $H_b(\cdot)$. □

4.4 Preserving Indifferentiability: the $\mathcal{C}_{4P\&\mathsf{IRO}}$ Combiner

First, we give a brief idea why our $\mathcal{C}_{4P}$ combiner does not guarantee the IRO property. To be IRO-robust the combiner has to be indifferentiable from a random oracle for any efficient adversary $\mathcal{A}$, if H_b is a random oracle for some $b \in \{0,1\}$. Thereby the adversary $\mathcal{A}$ has oracle access either to the combiner $\mathsf{Comb}_{4P}^{H_0,H_1}$ and the random oracle H_b, or to $\mathcal{F}$ and a simulator $\mathcal{S}^{\mathcal{F}}$. The simulator's goal is to mimic H_b such that $\mathcal{A}$ cannot have a significant advantage on deciding whether its interacting with $\mathsf{Comb}_{4P}^{H_0,H_1}$ and H_b, or with $\mathcal{F}$ and $S^{\mathcal{F}}$.

Usually, the strategy for designing such a simulator is to check if a query is a potential attempt of $\mathcal{A}$ to imitate the construction of the combiner and then to precompute further answers that are consistent with the information $\mathcal{A}$ can get from $\mathcal{F}$. However, for $\mathsf{Comb}_{4P}^{H_0,H_1}$ the simulator may be unable to precompute those consistent values, because an adversary $\mathcal{A}$ can compute the

permutation part of the combiner backwards such that $\mathcal{S}^{\mathcal{F}}$ has to commit to its round values used in the permutation P^3 before knowing the initial input M. To this end, $\mathcal{A}$ first queries the random oracle $\mathcal{F}$ on input M and uses the response $Y \leftarrow \mathcal{F}(M)$ to compute $X = P^{3^{-1}}(Y)$ with the help of $\mathcal{S}^{\mathcal{F}}$ simulating H_b and the function $H_{\overline{b}}$ which is accessible in a black-box manner. Then the answers of $\mathcal{S}^{\mathcal{F}}$, in order to be indistinguishable from those of H_b, must lead to a value $X = S(00||M)||H_1(00||M)$ if $b = 0$, and $X = H_0(00||M)||S(00||M)$ else.

While the part of X corresponding to $S(00||M)$ can simply be set as response to a further query $00||M$ by the simulator, the part of $H_{\overline{b}}(00||M)$ is determined by the oracle $H_{\overline{b}}(\cdot)$ and the message M. However, since the simulator does not know the message M when answering $\mathcal{A}$'s queries for computing $P^{3^{-1}}$, it is not able to call the $H_{\overline{b}}$ oracle about $00||M$ and to choose those answers accordingly. Thus, the probability that the responses provided by $\mathcal{S}^{\mathcal{F}}$ will lead in $P^{3^{-1}}(Y)$ to a value that is consistent with the structure of the combiner, is negligible and the adversary $\mathcal{A}$ can distinguish between $(\mathsf{Comb}_{4\mathsf{P}}^{H_0,H_1}, H_b)$ and $(\mathcal{F}, \mathcal{S}^{\mathcal{F}})$ with noticeable probability.

In order to guarantee the IRO property, we modify the $\mathsf{Comb}_{4\mathsf{P}}^{H_0,H_1}$ combiner such that the adversary is forced to query the message M before he can create meaningful queries aiming to imitate the construction. By this the simulator becomes able to switch to the common strategy of preparing consistent answers in advance. As explained in the introduction, adding a signature value α_M into the computation does the job.

4.4.1 The Combiner $\mathcal{C}_{4\mathsf{P}\&\mathsf{IRO}}$

In this section we consider the modified combiner $\mathcal{C}_{4\mathsf{P}\&\mathsf{IRO}}$ as illustrated in Figure 4.2. The combiner $\mathcal{C}_{4\mathsf{P}\&\mathsf{IRO}} = (\mathsf{CKGen}_{4\mathsf{P}\&\mathsf{IRO}}, \mathsf{Comb}_{4\mathsf{P}\&\mathsf{IRO}})$ is defined as follows: $\mathsf{CKGen}_{4\mathsf{P}\&\mathsf{IRO}}$ first samples $H_0 \leftarrow \mathsf{HKGen}_0(1^n), H_1 \leftarrow \mathsf{HKGen}_1(1^n)$ and a pairwise independent function $g : \{0,1\}^m \rightarrow \{0,1\}^{3m}$ for some $m \leq n/3$ (the larger m, the better the security level, but the longer the output, too):

Definition 4.7 (Pairwise-Independent Function/Permutation) *A family of functions $G : A \rightarrow B$ from domain A to range B is called pairwise independent iff for all $x \neq x' \in A$ and $z \neq z' \in B$ we have* $\mathrm{Prob}_{g \in G}[g(x) = z \wedge g(x') = z'] = |B|^{-2}$.

A family of function $\Pi : A \rightarrow A$ is a pairwise independent permutation, if for $x \neq x'$ and $z \neq z' \in A$ we have $\mathrm{Prob}_{g \in G}[g(x) = z \wedge g(x') = z'] = \frac{1}{|B|(|B|-1)}$.

One gets a simple construction of a pairwise independent function (PIF) mapping $\{0,1\}^n$ to $\{0,1\}^n$, by sampling $a, b \in \{0,1\}^n$ at random, which then defines the function $g_{(a,b)}(x) = (ax + b)$, where addition and multiplication are in the field $GF(2^n)$ (if we want a smaller range $\{0,1\}^m, m < n$, one can simply drop $n - m$ bits of the output). This construction is also a pairwise-

independent *permutation* (PIP), if a is chosen at random from $\{0,1\}^n \setminus 0^n$ (instead of $\{0,1\}^n$).

The evaluation algorithm $\mathsf{Comb}^{H_0,H_1,g}_{4P\&IRO}(M)$ first computes $\mathsf{Comb}^{H_0,H_1}_{||}(M) = H_0^0(M)||H_1^0(M)$ and a value α_M – which we call the "signature of M" – as $\alpha_M = lsb_m(H^0_\oplus(M))$ where $H^0_\oplus(M) = H_0^0(M) \oplus H_1^0(M)$ and $lsb_a(x)$ denotes the a least significant bits of x. The value α_M is used as an extra prefix in the round functions of the two-round Feistel permutation $P^2_\alpha(\cdot) = \psi[H^1_\oplus(\alpha_M\|\cdot), H^2_\oplus(\alpha_M\|\cdot)]$. Applying P^2_α on $H_0^0(M)||H_1^0(M)$ then gives the first part of the combiners output.

The construction as described so far, is already a robust combiner for IRO and PRF, but not for CR and TCR. The reason is that now distinct input messages M, M' where $\alpha_M \neq \alpha_{M'}$ lead to distinct Feistel permutations $P^2_{\alpha_M} \neq P^2_{\alpha_{M'}}$, and thus we cannot compute a collision for Comb^{H_0,H_1} (and thus for H_0 and H_1) from a collision $\mathsf{Comb}^{H_0,H_1}_{||}(P^2_{\alpha_M}(M)) = \mathsf{Comb}^{H_0,H_1}_{||}(P^2_{\alpha'_M}(M'))$.

To solve this problem, we could append the signature to the output of the combiner, which would enforce that two inputs can only collide if they have the same signature. Unfortunately, outputting the signature α directly would make the permutation P^2_α invertible, and ruin the IRO robustness of our construction again. This is why we only output a "blinded" version of the signature computed as $lsb_{3m}(H^3_\oplus(\alpha_M)) \oplus g(\alpha_M)$. This way the signature α_M gets not leaked when H_0 or H_1 is a random oracle, which is necessary for the combiner to be IRO robust. Moreover with high probability (over the choice of the pairwise-independent function g) the blinding, which maps $\{0,1\}^m$ to $\{0,1\}^{3m}$, will be injective (i.e., contain no collisions), which as explained before is necessary to get robustness for CR and TCR.

Overall, the combiner – as illustrated in Figure 4.2 – computes for input message M and its corresponding signature $\alpha_M = lsb_m(H^0_\oplus(M))$ the following output:

$$\mathsf{Comb}^{H_0,H_1,g}_{4P\&IRO}(M) = P^2_\alpha(H_0^0(M)||H_1^0(M)) \;||\; lsb_{3m}(H^3_\oplus(\alpha_M)) \oplus g(\alpha_M).$$

4.4.2 $\mathcal{C}_{4P\&IRO}$ is IRO-Robust

We show that our combiner is indifferentiable from a random oracle when instantiated with two functions H_0, H_1, where one of them is a random oracle (we refer to it as $H_b, b \in \{0,1\}$), and the other function $H_{\bar{b}}$ is arbitrary[4]. Like the random oracle H_b, also $H_{\bar{b}}$ is given as an oracle and accessible by all parties. The pairwise independent function g that comes up in this construction is only needed to prove that $\mathcal{C}_{4P\&IRO}$ still preserves the CR and TCR properties; for the IRO property this function can be arbitrary.

[4]There is a small caveat here. Our definition of combiners allows to use the same hash function $H_0 = H_1$, albeit our combiner samples independent instances of the hash functions then. In this sense, it is understood that, if hash function H_b is given by a random oracle, then in case $H_b = H_{\bar{b}}$ the other hash function instance uses an independent random oracle.

Lemma 4.8 *The combiner $\mathcal{C}_{\mathsf{4P\&IRO}}$ is IRO-robust.*

Remark. Note that the security of $\mathsf{Comb}_{\mathsf{4P\&IRO}}$ as a random oracle combiner depends on m, and thus on the output length, which is $2n+3m$. This can be slightly improved to $2n+2m+m'$ for some $m' < m$ (by simply replacing $3m$ with $2m+m'$ in Figure 4.2), though m' should not be too small, as $\mathcal{C}_{\mathsf{4P\&IRO}}$ is a good combiner for the CR and TCR with probability $2^{-m'}$ (this probability is over the choice of the PIF, as we explain later in Section 4.4.3).

Proof. For the proof we assume that $b = 0$, i.e., the hash function $H_0 : \{0,1\}^* \to \{0,1\}^n$ is a random oracle. The case $b = 1$ is proved analogously. The adversary $\mathcal{A}$ has then access either to the combiner $\mathsf{Comb}_{\mathsf{4P\&IRO}}$ and H_0 or to a random oracle $\mathcal{F} : \{0,1\}^* \to \{0,1\}^{2n+3m}$ and a simulator $\mathcal{S}^{\mathcal{F}}$. Our combiner is indifferentiable from a random oracle $\mathcal{F}$ if there exists a simulator $\mathcal{S}^{\mathcal{F}}$, such that the adversary $\mathcal{A}$ can distinguish between $\mathsf{Comb}_{\mathsf{4P\&IRO}}, H_0$ and $\mathcal{F}, \mathcal{S}^{\mathcal{F}}$ only with negligible probability. The proof consists of two parts: we first provide the description of our simulator $\mathcal{S}^{\mathcal{F}}$ and then we show that $\mathcal{A}$ has only negligible advantage in distinguishing the ideal setting (with $\mathcal{S}^{\mathcal{F}}$) and the real setting.

The simulator keeps as state the function table of a (partially defined) function $\hat{H}_0 : \{0,1\}^* \to \{0,1\}^n$, which initially is empty, i.e., $\hat{H}_0(X) = \perp$ for all X. We define $\hat{H}_0^i(M) = \hat{H}_0(\langle i \rangle_2 \,||M)$ to mimic the notion used in Figure 4.2. The goal of $\mathcal{S}^{\mathcal{F}}$ is to define $\hat{H}_0$ in such a way that, from $\mathcal{A}$'s point of view, $(\mathcal{F}, \hat{H}_0)$ look like $(\mathsf{Comb}_{\mathsf{4P\&IRO}}^{H_0,H_1,g}, H_0)$, i.e., the output of $\hat{H}_0$ has to be random and consistent to what the distinguisher can obtain from $\mathcal{F}$. Therefore, our simulator $\mathcal{S}^{\mathcal{F}}$ parses each query X it is invoked on as $X = \langle i \rangle_2 \,||M$ and proceeds as follows:

Simulator $\mathcal{S}^{\mathcal{F}}(X)$:
on query X check if some entry $Y \leftarrow \hat{H}_0(X)$ already exists
  if $Y = \perp$ //no entry so far
    if $X = \langle 0 \rangle_2 \,||M$ for some M
      set $\hat{H}_0^0(M) = y_0$ where y_0 is randomly chosen from $\{0,1\}^n$ $(*)$
      get $y_1 \leftarrow H_1^0(M)$ and compute $\alpha_M = lsb_m(y_0 \oplus y_1)$
      get $U \leftarrow \mathcal{F}(M)$ for query M and parse U as $U_1||U_2||U_3$ $(*)$
        where $|U_1| = |U_2| = n$ and $|U_3| = 3m$.
      set $\hat{H}_0^1(\alpha_M||y_1) = U_2 \oplus y_0 \oplus H_1^1(\alpha_M||y_1)$
      set $\hat{H}_0^2(\alpha_M||U_2) = U_1 \oplus y_1 \oplus H_1^2(\alpha_M||U_2)$
      set $\hat{H}_0^3(\alpha_M) = (U_3||z) \;\oplus\; (g(\alpha_M)||0^{n-3m}) \;\oplus\; H_1^3(\alpha_M)$
        where z is randomly chosen from $\{0,1\}^{n-3m}$
    if $X \neq \langle 0 \rangle_2 \,||M$, choose a random $Y \in \{0,1\}^n$ $(*)$
        and save the value by setting $\hat{H}_0(X) = Y$
output $Y \leftarrow \hat{H}_0(X)$

Figure 4.3: Description of the Simulator

Whenever $\mathcal{S}^{\mathcal{F}}$ is invoked on a query X where $\hat{H}_0(X) \neq \perp$, $\mathcal{S}^{\mathcal{F}}$ simply outputs $\hat{H}_0(M)$. Thus from now on we only consider queries X where $\hat{H}_0(X) = \perp$. In this case, $\mathcal{S}^{\mathcal{F}}$ will define the output of $\hat{H}_0(X)$, and in some cases also on some additional inputs. On a query $X = \langle i \rangle_2 ||M$ where $\hat{H}_0^i(M) = \perp$ and $i \neq 0$, the simulator samples a random $Y \in \{0,1\}^n$, sets $\hat{H}_0^i(M) = Y$ and outputs Y.

The interesting queries are the queries of the form $X = \langle 0 \rangle_2 ||M$ which could be an attempt of $\mathcal{A}$ to simulate the construction of the combiner, such that the simulator has to compute in addition consistent answers to potential subsequent queries of $\mathcal{A}$. The simulator starts by sampling a random $y_0 \in \{0,1\}^n$ and sets $\hat{H}_0^0(M) = y_0$. To define the "signature" α_M of M, $\mathcal{S}^{\mathcal{F}}$ queries its oracle H_1 on $\langle 0 \rangle_2 ||M$ and uses the answer $y_1 = H_1^0(M)$ to compute $\alpha_M = lsb_m(y_0 \oplus y_1)$. The simulator then defines the outputs of the intermediate functions $\hat{H}_0^1, \hat{H}_0^2$ and $\hat{H}_0^3$ such that $\mathsf{Comb}_{4P\&IRO}^{\hat{H}_0,H_1,g}(M) = \mathcal{F}(M)$. Therefore $\mathcal{S}^{\mathcal{F}}$ invokes its random oracle $\mathcal{F}$ on input M and computes the corresponding outputs of $\hat{H}_0$ by retracing the combiners construction as defined in the simulators description. Note that this is possible in a unique way, except for the $n - 3m$ last bits of $\hat{H}_0^3(\alpha_M)$, which must be chosen uniformly at random.

We now prove that from $\mathcal{A}$'s point of view $(\mathsf{Comb}_{4P\&IRO}^{H_0,H_1,g}, H_0)$ and $(\mathcal{F}, \mathcal{S}^{\mathcal{F}})$ are indistinguishable, when making at most q queries to each oracle. To this end we consider a sequence of hybrid games, starting with a game where $\mathcal{A}$ interacts with $(\mathsf{Comb}_{4P\&IRO}^{H_0,H_1,g}, H_0)$ and ending in the ideal setting where the distinguisher has access to $(\mathcal{F}, \mathcal{S}^{\mathcal{F}})$. The game structure of this proof is depicted in Figure 4.4.

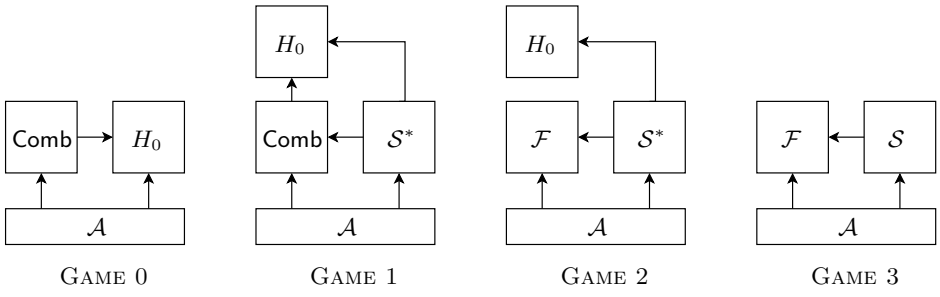


Figure 4.4: Games used in the Indifferentiability Proof

Game 0: The adversary interacts with $\mathsf{Comb}_{4P\&IRO}^{H_0,H_1,g}$ and H_0.

Game 1: We change the way $\mathcal{A}$'s queries to H_0 are answered, by giving $\mathcal{A}$ access to an algorithm $\mathcal{S}^*$ instead of direct access to the random oracle. The algorithm $\mathcal{S}^*$ works as our simulator $\mathcal{S}$, except that it queries H_0 instead of simulating it via lazy sampling, and it calls $\mathsf{Comb}_{4P\&IRO}^{H_0,H_1,g}(M)$ instead of $\mathcal{F}(M)$. Thus, $\mathcal{S}^*$ basically relays all queries of $\mathcal{A}$ to H_0 but also keeps a table of answered values. For all queries of the form $X = \langle 0 \rangle_2 ||M$ the algorithm

additionally precomputes further values as described in Figure 4.3 (the lines where $\mathcal{S}^*$ deviates from $\mathcal{S}$ are marked with $*$). Note that $\mathcal{S}^*$'s answers and stored values are (with one exception) exactly the same as the values one would obtain directly from H_0. In particular, the values $\mathcal{S}^*$ defines for $\hat{H}_0^1, \hat{H}_0^2$ by querying $\mathsf{Comb}_{\mathsf{4P\&IRO}}^{H_0,H_1,g}$ are identical to the real values of H_0^1, H_0^2. The only difference occurs when in the precomputations of $\mathcal{S}^*$ a value for $\hat{H}_0^3(\alpha_M)$ is set, since only the first $3m$ bits will equal the value of $H_0^3(\alpha_M)$. However, the final $n - 3m$ bits are set to random such that also $\hat{H}_0^3(\alpha_M)$ is a truly random string. Thus, the only way for $\mathcal{A}$ to recognize the discrepancy to the real $H_0^3(\alpha_M)$ value, is by querying $\langle 3 \rangle_2 \,||\alpha_M$ *before* sending a query $\langle 0 \rangle_2 \,||M$ that will lead to α_M. We denote this event by Bad_1. As all signature values α_M that originate from a query to H_0^0 are uniform random values of length m and $\mathcal{A}$ makes at most q queries to its $\mathcal{S}^*$ oracle, this event happens with overall probability at most $q^2 \cdot 2^{-m}$. Unless $\mathcal{A}$ provokes Bad_1, Game 0 and Game 1 are identical (we denote by Game $i \Rightarrow 1$ the event that $\mathcal{A}$ outputs 1 in Game i):

$$|\,\mathrm{Prob}[\,\textsc{Game}\ 1 \Rightarrow 1] - \mathrm{Prob}[\,\textsc{Game}\ 0 \Rightarrow 1]\,| \leq \mathrm{Prob}[\,\mathsf{Bad}_1] \leq q^2 \cdot 2^{-m}$$

Game 2: In our second game we replace the combiner $\mathsf{Comb}_{\mathsf{4P\&IRO}}^{H_0,H_1,g}$ with the random oracle $\mathcal{F}$. Due to that change, the algorithm $\mathcal{S}^*$ now obtains $\mathcal{F}(M)$ instead of $\mathsf{Comb}_{\mathsf{4P\&IRO}}^{H_0,H_1,g}(M)$ when doing its precomputations. Thus, the additional values that $\mathcal{S}^*$ stores in $\hat{H}_0^i$ for $i \in \{1,2,3\}$ when responding to a $\langle 0 \rangle_2 \,||M$ query, are now consistent with $\mathcal{F}(M)$ and thereby with high probability different from the real values of H_0^i for $i \in \{1,2,3\}$. Again, this only matters if $\mathcal{A}$ manages to first issue a query $\langle i \rangle_2 \,||\alpha_M||*$ and subsequently invokes $\mathcal{S}^*$ on $\langle 0 \rangle_2 \,||M$ that will lead to α_M. Otherwise, all $\mathcal{A}$ gets to see from $\mathcal{S}^*$ are random and consistent answers. To capture that case where $\mathcal{S}^*$ "fails", we consider by Bad_2 the event that the function $\hat{H}_0^i$ for $i \in \{1,2\}$ is already defined on any input of the form $\alpha_M||*$ when $\mathcal{S}^*$ wants to set a value in the course of a precomputation. (Note that the case for $i = 3$ is already handled by Bad_1 in Game 1.) As $\alpha_M \in \{0,1\}^m$ is uniformly random, the probability that Bad_2 occurs in the q-th query is at most $2q \cdot 2^{-m}$ (as each $\hat{H}_0^i$ for $i \in \{1,2\}$ is defined on at most $q-1$ inputs). Then the overall probability that Bad_2 in any of $\mathcal{A}$'s queries happens is at most $2q^2 \cdot 2^{-m}$.

Furthermore, the outputs provided by $\mathsf{Comb}_{\mathsf{4P\&IRO}}^{H_0,H_1,g}$ are indistinguishable from $\mathcal{F}$, as long as no collision on the signature values occurs, i.e., $M \neq M'$ but $\alpha_M = \alpha'_M$ (we omit a formal proof, as it follows the argumentation of Lemma 4.10 closely). Since $\mathcal{A}$ sends at most q queries to $\mathsf{Comb}_{\mathsf{4P\&IRO}}^{H_0,H_1,g}$, such a collision occurs with probability at most $q^2 \cdot 2^{-m}$. By adding the probabilities of both events we obtain

$$|\,\mathrm{Prob}[\,\textsc{Game}\ 2 \Rightarrow 1] - \mathrm{Prob}[\,\textsc{Game}\ 1 \Rightarrow 1]\,| \leq 3q^2 \cdot 2^{-m}$$

GAME 3: In the final game the adversary interacts with $\mathcal{F}$ and $\mathcal{S}^{\mathcal{F}}$. That is, GAME 2 and GAME 3 only differ in the fact that $\mathcal{S}^{\mathcal{F}}$ simulates the random responses from H_0 by using lazy sampling instead of querying H_0. Thus, from $\mathcal{A}$'s viewpoint both games are identical:

$$\text{Prob}[\,\text{GAME } 3 \Rightarrow 1] = \text{Prob}[\,\text{GAME } 2 \Rightarrow 1]$$

Overall, we have

$$|\,\text{Prob}[\,\text{GAME } 3 \Rightarrow 1] - \text{Prob}[\,\text{GAME } 0 \Rightarrow 1]\,| \leq 4q^2 \cdot 2^{-m}.$$

Hence, the advantage of $\mathcal{A}$ in distinguishing $(\mathsf{Comb}^{H_0,H_1,g}_{4P\&IRO}, H_0)$ from $(\mathcal{F}, \mathcal{S}^{\mathcal{F}})$ is negligible. This proves our claim.

□

4.4.3 $\mathcal{C}_{4P\&IRO}$ is Robust for CR, TCR, MAC, PRF

We now prove that, like the $\mathcal{C}_{4P}$ combiner, $\mathcal{C}_{4P\&IRO}$ also preserves the CR, TCR, MAC and PRF property in a robust manner.

Lemma 4.9 *The combiner $\mathcal{C}_{4P\&IRO}$ is CR- and TCR-robust.*

Proof. We will prove that for any H_0, H_1, with probability $1 - 2^{-m}$ over the choice of the pairwise independent function g, any collision for $\mathsf{Comb}^{H_0,H_1,g}_{4P\&IRO}$ is simultaneously a collision for H_0^0 and H_1^0. To this end, let $M \neq M'$ be a collision for $\mathsf{Comb}^{H_0,H_1,g}_{4P\&IRO}$ and let α_M and $\alpha_{M'}$ denote their signatures. Let $Y\|Y' = \mathsf{Comb}^{H_0,H_1,g}_{4P\&IRO}(M)$ where $Y \in \{0,1\}^{2n}$ and $Y' \in \{0,1\}^{3m}$.

If $\alpha_M = \alpha_{M'}$, then M, M' must be a collision for H_0^0 and H_1^0, as we have

$$H_0^0(M)\|H_1^0(M) = P_\alpha^{2\,-1}(Y) = P_{\alpha'}^{2\,-1}(Y) = H_0^0(M')\|H_1^0(M') \tag{4.1}$$

and the Feistel permutations $P^2_\alpha, P^2_{\alpha'}$ are identical if $\alpha_M = \alpha_{M'}$.

For M, M' where $\alpha_M \neq \alpha_{M'}$, a collision on the combiner $\mathsf{Comb}^{H_0,H_1,g}_{4P\&IRO}(M) = \mathsf{Comb}^{H_0,H_1,g}_{4P\&IRO}(M')$ does not imply (4.1), and thus will in general not be a collision for H_0 and H_1. Yet, as with probability $1 - 2^{-m}$ over the choice of the pairwise independent function $g : \{0,1\}^m \to \{0,1\}^{3m}$, there does not exist a collision M, M' for $\mathsf{Comb}^{H_0,H_1,g}_{4P\&IRO}$ where $\alpha_M \neq \alpha_{M'}$. Note that for this it is sufficient to prove that for any two potential signatures $\alpha \neq \alpha' \in \{0,1\}^m$, we have

$$lsb_{3m}(H^3_\oplus(\alpha)) \oplus g(\alpha) \neq lsb_{3m}(H^3_\oplus(\alpha')) \oplus g(\alpha') \tag{4.2}$$

as this implies that the final outputs are distinct for any two messages with different signatures. As g is pairwise independent, for any particular $\alpha \neq \alpha'$, equation (4.2) holds with probability $1 - 2^{-3m}$. Taking the union bound over

all $2^m(2^m-1)/2 < 2^{2m}$ distinct values $\alpha \neq \alpha'$, we get that the probability that there exists some $\alpha \neq \alpha'$ not satisfying (4.2) is at most $2^{2m}/2^{3m} = 2^{-m}$.

The proof of TCR-robustness follows a similar argumentation. A collision $M \neq M'$ on the combiner implies with overwhelming probability a collision $H_0^0(M)||H_1^0(M) = H_0^0(M')||H_1^0(M')$ on the first evaluation of both hash functions. Thus, given an adversary $\mathcal{A}_{\mathsf{Comb}}$ against the combiner that commits to a target message M and later outputs a colliding message M', one can build an adversary $\mathcal{A}_b$ against hash function H_b that commits to $00||M$ and outputs in the second stage $00||M'$. □

Lemma 4.10 *The combiner $\mathcal{C}_{\mathsf{4P\&IRO}}$ is PRF-robust.*

Remark. To compute the first part of the output, our combiner $\mathsf{Comb}_{\mathsf{4P\&IRO}}^{H_0,H_1,g}$ applies a two-round Feistel network, which in general does not preserve the (pseudo)-randomness from an underlying round function $H_{\oplus}^i$, because it maps an input (L_0, R_0) to (L_2, R_2) where $R_2 = H_{\oplus}^1(R_0) \oplus L_0$ depends only on the given input values. When evaluating the Feistel network with two distinct inputs (L_0, R_0) and (L_0', R_0), the difference $L_0 \oplus L_0'$ then propagates to the outputs, i.e., $L_0 \oplus L_0' = R_2 \oplus R_2'$, which can be exploited by an adversary. In our construction we destroy this dependence by prepending the value α_M to the input of each round function, where $\alpha_M = lsb_m(H_{\oplus}^0(M))$ is a uniformly random value if $H_b, b \in \{0,1\}$ is a uniformly random function. Thus we have $R_2 = H_{\oplus}^1(\alpha_M||R_0) \oplus L_0$ with $L_0 = H_0^0(M)$ and $R_0 = H_1^0(M)$ such that for two distinct inputs $M \neq M'$, the probability for $R_2 \oplus R_2' = H_0^0(M) \oplus H_0^0(M')$ is $\mathrm{Prob}[\alpha_M = \alpha_{M'}] = 2^{-m}$.

Proof. Assume that the hash function H_0 is a pseudorandom function, but the combiner $\mathsf{Comb}_{\mathsf{4P\&IRO}}^{H_0,H_1,g}$ is not (the proof for H_1 can be done analogously). Hence, there exists a successful adversary $\mathcal{A}_{\mathsf{Comb}}$ which can distinguish the construction $\mathsf{Comb}_{\mathsf{4P\&IRO}}^{H_0,H_1,g}$ from a truly random function $F : \{0,1\}^* \to \{0,1\}^{2n+3m}$ with non-negligible probability. We show that this allows to construct an adversary $\mathcal{A}_0$ that can distinguish H_0 from a random function $f : \{0,1\}^* \to \{0,1\}^n$.

Algorithm $\mathcal{A}_0$ simulates the oracle of $\mathcal{A}_{\mathsf{Comb}}$, which is either $\mathsf{Comb}_{\mathsf{4P\&IRO}}^{H_0,H_1,g}$ or F, with his own oracle and the knowledge of $H_1 \leftarrow \mathsf{HKGen}_1$ and g that he samples accordingly. For each query of $\mathcal{A}_{\mathsf{Comb}}$, the adversary $\mathcal{A}_0$ computes an answer by emulating the combiner $\mathsf{Comb}_{\mathsf{4P\&IRO}}$ using $H_1(\cdot), g$ and his oracle which serves as H_0.

For the analysis recall that the underlying oracle of $\mathcal{A}_0$ is either a random function f or the hash function $H_0(\cdot)$. In the latter case $\mathcal{A}_0$ provides outputs that are identically distributed to the values $\mathcal{A}_{\mathsf{Comb}}$ would obtain from $\mathsf{Comb}_{\mathsf{4P\&IRO}}^{H_0,H_1,g}$. Hence, we have

$$\mathrm{Prob}\left[\mathcal{A}_0^{H_0}(1^n) = 1\right] = \mathrm{Prob}\left[\mathcal{A}_{\mathsf{Comb}}^{\mathsf{Comb}_{\mathsf{4P\&IRO}}^{H_0,H_1,g}}(1^n) = 1\right].$$

If the underlying oracle is the random function f, then the computed answers of $\mathcal{A}_0$ have to look like a truly random function as well. We show that this is true if, for q queries $M_1 \dots M_q$ and for all $i \neq j$, we have $\alpha_{M_i} \neq \alpha_{M_j}$. The probability of this not being the case is at most $q^2 \cdot 2^{-m}$, since $\alpha_M = lsb_m(H^0_{\oplus}(M))$ is a random value when H_0 gets replaced by the random function f.

Hence, with high probability $\mathcal{A}_0$ will create for each query M_i of $\mathcal{A}_{\mathsf{Comb}}$ a fresh signature α_{M_i}. To analyze the corresponding output of $\mathcal{A}_0$ we parse his answer in three parts, namely $\mathsf{Comb}^{f,H_1,g}_{4P\&IRO}(M_i) = U_1||U_2||U_3$ with $|U_1| = |U_2| = n$ and $|U_3| = 3m$. The last part U_3 results from the computation $lsb_{3m}(f(\langle 3\rangle_2 \,||\alpha_{M_i}) \oplus H^3_1(\alpha_{M_i})) \oplus g(\alpha_{M_i})$. Since α_{M_i} is uniformly distributed and gets extended by the unique prefix $\langle 3\rangle_2$, the input value of $f(\langle 3\rangle_2 \,||\alpha_{M_i})$ is distinct from all other queries to f during the $\mathsf{Comb}^{f,H_1,g}_{4P\&IRO}(M_i)$ computation, and hence the corresponding output is an independently and uniformly distributed value. As xor-ing is a good combiner for random functions, the randomness of f gets preserved in the computation of U_3. For the second part U_2 we just consider the final calculation, i.e., $U_2 = f(\langle 0\rangle_2 \,||||M_i) \oplus f(\langle 1\rangle_2 \,||\alpha_{M_i}||Y) \oplus H^1_1(\alpha_{M_i}||Y)$ for some $Y \in \{0,1\}^n$. Here we prepend the bits $\langle 1\rangle_2$ to the random value α_{M_i}, such that we have again distinct evaluations of f which gives us uniformly random images. A similar argumentation holds for $U_1 = Y' \oplus f(\langle 2\rangle_2 \,||\alpha_{M_i}||Y'') \oplus H^2_1(\alpha_{M_i}||Y'')$ for $Y', Y'' \in \{0,1\}^n$, where we use the unique prefix $\langle 2\rangle_2$ when querying f in order to obtain values that are independently and uniformly distributed. Thus, if for all queried messages $M_i \neq M_j$ of $\mathcal{A}_{\mathsf{Comb}}$ there occurs no collision on the signatures, i.e., $\alpha_{M_i} \neq \alpha_{M_j}$, the values $U_1||U_2||U_3$ are independent random strings.

Overall, the output distribution of $\mathcal{A}_{\mathsf{Comb}}$ satisfies

$$\mathrm{Prob}\Big[\,\mathcal{A}^f_0(1^n) = 1\Big] \leq \mathrm{Prob}\big[\,\mathcal{A}^F_{\mathsf{Comb}}(1^n) = 1\big] + q^2 \cdot 2^{-m}.$$

Thus, the probability that $\mathcal{A}_0$ can distinguish H_0 from f is not negligible, which contradicts the assumption that H_0 is a pseudorandom function. □

Lemma 4.11 *The combiner $\mathcal{C}_{4P\&IRO}$ is MAC-robust.*

Proof. The proof is by contradiction. Assume that an adversary $\mathcal{A}_{\mathsf{Comb}}$ with oracle access to the combiner $\mathsf{Comb}^{H_0,H_1,g}_{4P\&IRO}$ outputs with noticeable probability a valid pair (M,τ) where $\tau = \mathsf{Comb}^{H_0,H_1,g}_{4P\&IRO}(M)$ and M is distinct from all previous queries to the MAC-oracle. This allows to construct an adversary $\mathcal{A}_b$ against the hash function H_b for $b \in \{0,1\}$.

Adversary $\mathcal{A}_b$ first samples $H_{\overline{b}} \leftarrow \mathsf{HKGen}_1$ that it uses together with its own oracle $H_b(\cdot)$ to answer all queries by $\mathcal{A}_{\mathsf{Comb}}$ in a black-box simulation. When $\mathcal{A}_{\mathsf{Comb}}$ returns a valid forgery (M,τ), where $M \neq M_1, M_2 \dots M_q$, the adversary $\mathcal{A}_b$ flips a coin $c \leftarrow \{0,1\}$ and proceeds as follows:

- If $c = 0$, then $\mathcal{A}_b$ randomly chooses an index k between 1 and q and looks up the corresponding signature value α_{M_k}. It then computes $\tau_0||\tau_1 = P_\alpha^{2-1}(lsb_{2n}(\tau))$ using α_{M_k} and stops with the output $(\langle 0\rangle_2 ||M, \tau_b)$.

- If $c = 1$, then $\mathcal{A}_b$ queries its oracle about $\langle 0\rangle_2 ||M$ to receive an answer y_0 and computes $\alpha_M = y_0 \oplus y_1$ with $y_1 = H_1^0(M)$. It then calculates the first round of the Feistel permutation, i.e., until the evaluation of $H_\oplus^2$ where $x = y_0 \oplus H_0^1(\alpha_M||y_1)$ would be used as input to this function. It outputs as forgery the message $(\langle 2\rangle_2 ||\alpha_M||x)$ with tag $\tau' = \tau_b \oplus H_{\bar{b}}(\langle 2\rangle_2 ||\alpha_M||x) \oplus y_1$ where $\tau_0||\tau_1 = lsb_n(\tau)$.

For the analysis we have to consider two cases of an successful adversary $\mathcal{A}_{\mathsf{Comb}}$. In the first case, $\mathcal{A}_{\mathsf{Comb}}$ returns a pair (M, τ), such that $\alpha_M = \alpha_{M_j}$ for some $j = 1, 2, \ldots, q$, i.e., the signature value of M has already been computed for another message $M_j \neq M$ during A_b's process of simulating the combiner. Then, if $c = 0$, the adversary $\mathcal{A}_b$ obtains a valid forgery $(\langle 0\rangle_2 ||M, \tau_b)$ if it guesses the index j correctly and then inverts the Feistel step for input $lsb_{2n}(\tau)$ and α_{M_j}. The message $\langle 0\rangle_2 ||M$ is distinct from all of $\mathcal{A}_b$'s queries, because $\langle 0\rangle_2 ||M$ is distinct from all $\langle 0\rangle_2 ||M_i$ and the additional queries of $\mathcal{A}_b$ start with a prefix $\langle i\rangle_2$ where $i \in 1, 2, 3$. Hence, if $\mathcal{A}_{\mathsf{Comb}}$ forges such a MAC with non-negligible probability ϵ, then $\mathcal{A}_b$ succeeds with probability $\epsilon/2q$.

In the second case, $\mathcal{A}_{\mathsf{Comb}}$ outputs (M, τ) where α_M has not occurred in $\mathcal{A}_b$'s computations, i.e., $\alpha_M \neq \alpha_{M_j}$ for all $j = 1, 2, \ldots, q$. In this case, we have $c = 1$ with probability $1/2$ where $\mathcal{A}_b$ starts its forgery by computing the first round of the Feistel permutation for input $H_0^0(M)||H_1^0(M)$ and $\alpha_M = lsb_m(H_\oplus^0(M))$, which requires a further oracle query about $00||M$. The left part of the computed Feistel output is then $x = H_0^0(M) \oplus H_0^1(\alpha_M||H_1^0(M))$ and would serve as input for $H_\oplus^2$. The adversary uses this value together with the fresh signature α_M as its output message $(\langle 2\rangle_2 ||\alpha_M||x)$ and reconstructs the corresponding tag with the knowledge about the other parameters. Since α_M is distinct from all α_{M_j}, the message $(\langle 2\rangle_2 ||\alpha_M||x)$ was never queried by $\mathcal{A}_b$ before.

In both cases a successful attack against the combiner $\mathsf{Comb}_{\mathsf{4P\&IRO}}^{H_0,H_1,g}$ allows successful attacks on H_0 and H_1, contradicting the assumption that at least one hash function is a secure MAC. □

4.5 Preserving One-Wayness and the $\mathcal{C}_{\mathsf{4P\&OW}}$ Combiner

In this section we first propose a combiner which simultaneously is a combiner for CR and OW. At the end of this section we discuss how to plug this combiner into our combiners $\mathsf{Comb}_{\mathsf{4P}}$ and $\mathsf{Comb}_{\mathsf{4P\&IRO}}$ to get our construction $\mathsf{Comb}_{\mathsf{4P\&OW}}$ (cf. Figure 4.1) and $\mathsf{Comb}_{\mathsf{6P}}$ (cf. Figure 4.2), respectively.

Recall that the concatenation combiner

$$\mathsf{Comb}_{\|}^{H_0,H_1}(M) = H_0(M)\|H_1(M)$$

is a robust combiner for the CR property, but its not hard to see that this combiner is not robust for the one-wayness property OW. On the other hand, the following combiner

$$\mathsf{Comb}_{\mathsf{OW}}^{H_0,H_1}(M_L\|M_R) = H_0(M_L)\|H_1(M_R)$$

is robust for tho OW property, i.e. $\mathsf{Comb}_{\mathsf{OW}}^{H_0,H_1}(M_L\|M_R)$ is hard to invert on a random input from $\{0,1\}^{2m}$, if either H_0 or H_1 is hard to invert on $\{0,1\}^m$. Unfortunately, this combiner is not robust for CR.

The basic idea to construct a combiner which is robust for CR and OW is to use the $\mathsf{Comb}_{\|}^{H_0,H_1}$ combiner, but to apply a pairwise independent permutation (PIP) to the input of one of the two components. As the length of a description of a PIP is twice its input length, we have to assume an upper bound on the input length of the components. We fix the domain of H_0 and H_1 to $\{0,1\}^{5n}$, but let us mention that any longer input length $kn, k > 5$ will work too (but then we will also need $2kn$ bits for the description of P). Allowing shorter input length $kn, k < 5$ is not possible, as we use the fact that the input is (at least) $5n$ bits in the proof.

4.5.1 A Combiner for CR and OW

We define the combiner $\mathcal{C}_{\mathsf{CR}\&\mathsf{OW}}$ for preserving collision-resistance and one-wayness in a robust manner as follows. The key generation algorithm of the combiner $\mathsf{CKGen}_{\mathsf{CR}\&\mathsf{OW}}(1^n)$ generates $H_0 \leftarrow \mathsf{HKGen}_0(1^n)$ and $H_1 \leftarrow \mathsf{HKGen}_1(1^n)$ and picks a pairwise independent permutation $\pi : \{0,1\}^{5n} \to \{0,1\}^{5n}$. It outputs (H_0, H_1, π). The evaluation algorithm $\mathsf{Comb}_{\mathsf{CR}\&\mathsf{OW}}^{H_0,H_1,\pi}$ on input $M \in \{0,1\}^{5n}$ returns $H_0(\pi(M))||H_1(M)$. By the following theorem $\mathcal{C}_{\mathsf{CR}\&\mathsf{OW}}$ preserves the properties of $\mathsf{Comb}_{||}$ and $\mathsf{Comb}_{\mathsf{OW}}$ simultaneously.

Theorem 4.12 *The combiner $\mathcal{C}_{CR\&OW}$ is a strongly robust multi-property combiner for* PROP $= \{CR, TCR, MAC, OW\}$.

The proof is again split into lemmas for the individual properties.

Lemma 4.13 *The combiner $\mathcal{C}_{CR\&OW}$ is CR-, TCR- and MAC-robust.*

Proof. As for the CR and TCR properties, note that given any collision $M \neq M'$ for $\mathsf{Comb}_{\mathsf{CR}\&\mathsf{OW}}^{H_0,H_1,\pi}$, we get a collision M, M' for H_1 and a collision $\pi(M), \pi(M')$ for H_0. Note that $\pi(M) \neq \pi(M')$ as π is a permutation.

To see that the MAC property is preserved, observe that given any forgery (M, τ) for $\mathsf{Comb}_{\mathsf{CR}\&\mathsf{OW}}^{H_0,H_1,\pi}$, we get a forgery $(\pi(M), \tau_0)$ for H_0 and a forgery (M, τ_1) for H_1 where $\tau_0\|\tau_1 = \tau$. □

Lemma 4.14 *The combiner $\mathcal{C}_{\mathsf{CR\&OW}}$ is OW-robust.*

More precisely, we show that for any functions H_0, H_1 and any $T = T(n)$, the following is true for all but a $1/2T$ fraction of the π's: an adversary who inverts $\mathsf{Comb}^{H_0,H_1,\pi}_{\mathsf{CR\&OW}}$ with probability $1/2T$, can be used to invert H_0 and H_1 with probability $1/2T^3$.[5]

Proof. We first need to relate the output of our combiner $\mathsf{Comb}^{H_0,H_1,\pi}_{\mathsf{CR\&OW}}$ to the one of $\mathsf{Comb}^{H_0,H_1}_{\mathsf{OW}}$, depending on T. For this we call a tuple $(\pi_0, y_0||y_1)$ bad if it is more than $2T^2$ times more likely to be a key/output pair of $\mathsf{Comb}^{H_0,H_1,\pi_0}_{\mathsf{CR\&OW}}$, compared to the combiner $\mathsf{Comb}^{H_0,H_1}_{\mathsf{OW}}(\cdot) = H_0(\cdot)||H_1(\cdot)$ and random permutation π. That is, $(\pi, y_0||y_1)$ is called *bad* iff

$$\begin{aligned}\mathrm{Prob}_M[\mathsf{Comb}^{H_0,H_1,\pi}_{\mathsf{CR\&OW}}(M) = y_0||y_1] \\ \geq \quad 2 \cdot T^2 \cdot \mathrm{Prob}_{M_0,M_1}[\mathsf{Comb}^{H_0,H_1}_{\mathsf{OW}}(M_0||M_1) = y_0||y_1].\end{aligned}$$

Equivalently,

$$\begin{aligned}\mathrm{Prob}_M[H_0(\pi(M)) = y_0 | H_1(M) = y_1] \\ \geq \quad 2 \cdot T^2 \cdot \mathrm{Prob}_{M_0,M_1}[H_0(M_0) = y_0 | H_1(M_1) = y_1]. \qquad (4.3)\end{aligned}$$

We next bound the likelihood of a tuple to be bad in terms of the adversary's success probability (and running time):

Claim 1: $\mathrm{Prob}_{\pi,M}[(\pi, \mathsf{Comb}^{H_0,H_1,\pi}_{\mathsf{CR\&OW}}(M)) \text{ is bad}] \leq 2/T^2$, where the probability is over the choice of the PIP $\pi : \{0,1\}^{5n} \to \{0,1\}^{5n}$ and $M \in \{0,1\}^{5n}$.

Proof. Letting $\mathcal{M}_0 = H_0^{-1}(y_0)$ and $\mathcal{M}_1 = H_1^{-1}(y_1)$ denote the pre-images of y_0 and y_1 under H_0 and H_1, respectively, and $\pi(\mathcal{M}_0)$ be the set of all $\pi(x)$ for $x \in \mathcal{M}_0$, we can bound the terms in (4.3) as:

$$\mathrm{Prob}_{M_0,M_1}[H_0(M_0) = y_0 | H_1(M_1) = y_1] = \frac{|\mathcal{M}_0|}{2^{5n}} \qquad (4.4)$$

$$\mathrm{Prob}_M[H_0(\pi(M)) = y_0 | H_1(M) = y_1] = \frac{|\mathcal{M}_0 \cap \pi(\mathcal{M}_1)|}{|\mathcal{M}_1|} \qquad (4.5)$$

The former equation is clear as we hit a pre-image of y_0 for the random M_0 with the given probability, and the latter follows as each of the possible pre-images of y_1 must be mapped via π to a pre-image of y_0.[6]

With equations (4.4),(4.5) and (4.3) we can rewrite the statement of the claim as

$$\mathrm{Prob}_{\pi,M}\left[\frac{|\mathcal{M}_0 \cap \pi(\mathcal{M}_1)|}{|\mathcal{M}_1|} \geq T^2 \cdot 2\frac{|\mathcal{M}_0|}{2^{5n}}\right] \leq \frac{2}{T^2}. \qquad (4.6)$$

[5] Note that this statement implies that if either H_0 or H_1 is one-way and π is chosen at random, then $\mathsf{Comb}^{H_0,H_1,\pi}_{\mathsf{CR\&OW}}$ is one-way with overwhelming probability.

[6] Note that $\mathcal{M}_1$ contains at least the element $H_1(M)$, so division by 0 cannot occur.

In order to prove this we consider for any $\mathcal{M}_0, \mathcal{M}_1$ and $\pi(M)$ the expected size of $|\mathcal{M}_0 \cap \pi(\mathcal{M}_1)|/|\mathcal{M}_1|$ (over the choice of π). First note that at least one element, namely $\pi(M)$, lies in $\mathcal{M}_0 \cap \pi(\mathcal{M}_1)$. For any other of the $|\mathcal{M}_1| - 1$ possible values $M' \in \mathcal{M}_1, M' \neq M$, the value $\pi(M')$ is uniformly distributed in $\{0,1\}^{5n} \setminus \pi(M)$, because π is a pairwise independent permutation. So the probability that $\pi(M')$ hits $\mathcal{M}_0$ is $(|\mathcal{M}_0|-1)/(2^{5n}-1)$ (observe that the term $|\mathcal{M}_0| - 1$ comes from the fact that $\pi(M) \in \mathcal{M}_0$ cannot be hit). Hence,

$$\mathbb{E}\left[\frac{|\mathcal{M}_0 \cap \pi(\mathcal{M}_1)|}{|\mathcal{M}_1|}\right] = \frac{1}{|\mathcal{M}_1|}\left(1 + \frac{(|\mathcal{M}_0|-1)(|\mathcal{M}_1|-1)}{2^{5n}-1}\right). \tag{4.7}$$

For large $\mathcal{M}_0$ and $\mathcal{M}_1$ the right hand side of the previous equation converges towards (4.4). We are therefore interested in the probability that $\mathcal{M}_0$ and $\mathcal{M}_1$ are large. To derive this probability first note that for any function $f : \{0,1\}^{5n} \to \{0,1\}^n$ there are at most 2^n images y with $|f^{-1}(y)| \leq 2^{3n}$, and a random input M falls into such a bad set with probability at most $2^{4n}/2^{5n} = 2^{-n}$. As M and $\pi(M)$ are uniformly distributed, it follows that

$$\text{Prob}[|\mathcal{M}_0| < 2^{3n} \vee |\mathcal{M}_1| < 2^{3n}] \leq 2 \cdot 2^{-n}. \tag{4.8}$$

Hence, except with probability $2 \cdot 2^{-n}$ (which becomes smaller than $1/T^2$ for sufficiently large n's), we have $|\mathcal{M}_0| \geq 2^{3n}$ and $|\mathcal{M}_1| \geq 2^{3n}$, let us call this event $\mathcal{E}$. In this case

$$\frac{1}{|\mathcal{M}_1|}\left(1 + \frac{(|\mathcal{M}_0|-1)(|\mathcal{M}_1|-1)}{2^{5n}-1}\right) \leq 2\frac{|\mathcal{M}_0|}{2^{5n}}, \tag{4.9}$$

We can now prove (4.6) as (below $Z = |\mathcal{M}_0 \cap \pi(\mathcal{M}_1)|/|\mathcal{M}_1|$)

$$\begin{aligned} \text{Prob}\left[Z \geq T^2 \cdot 2\frac{|\mathcal{M}_0|}{2^{5n}}\right] &\leq \text{Prob}[Z \geq T^2 \cdot \mathbb{E}[Z]|\mathcal{E}] + \text{Prob}[\neg\mathcal{E}] \\ &\leq 1/T^2 + 2 \cdot 2^{-n} \leq 2/T^2 \end{aligned} \tag{4.10}$$

where we used (4.7)-(4.9) in the first and Markov's inequality in the second step. □

Using Markov's inequality once more the claim implies

$$\text{Prob}_\pi[\text{Prob}_M[(\pi, \mathsf{Comb}^{H_0,H_1,\pi}_{\mathsf{CR\&OW}}(M)) \text{ is bad}] \leq 1/T] \geq 1 - 2/T. \tag{4.11}$$

We say that the permutation π is *good* if $\text{Prob}_M[(\pi, \mathcal{C}_{\mathsf{CR\&OW}}(\pi, M)) \text{ is bad}] \leq 1/T]$, thus by the above equation, a random π is good with probability at least $1 - 2/T$.

To conclude the proof, assume there exists an adversary $\mathcal{A}$ which inverts $\mathsf{Comb}^{H_0,H_1,\pi}_{\mathsf{CR\&OW}}(.)$ with noticeable probability $\epsilon = 2/T$ for more than a $2/T$ fraction of the π's. Thus by equation (4.11), this must be the case for at least

one good π. For this π, the output of $\mathsf{Comb}_{\mathsf{CR\&OW}}^{H_0,H_1,\pi}(.)$ is bad with probability at most $1/T$, thus $\mathcal{A}$ must invert with probability at least $\epsilon - 1/T$ even on outputs that are not bad. But then, by equation (4.3), it must also invert of $\mathsf{Comb}_{\mathsf{OW}}^{H_0,H_1}(M_0\|M_1)$ for random $M_0\|M_1$ with probability at least $(\epsilon - 1/T)/2T^2 = 1/2T^3$. □

4.5.2 Combining Things

We can now plug the combiner $\mathcal{C}_{\mathsf{CR\&OW}}$ into the initial computation of our combiner $\mathcal{C}_{\mathsf{4P}}$. That is, we replace the initial computation $H_0^0(M)||H_1^0(M)$ in our original combiner by $H_0^0(\pi(M))||H_1^0(M)$ for messages of $5n$ bits. Note that if $H_b(\cdot)$ is one way on inputs of length $5n+2$, then also $H_b^0(\cdot)$ is one-way on inputs of length $5n$, and we only lose a factor of 4 in the security.

More formally, in our combiner $\mathcal{C}_{\mathsf{4P\&OW}} = (\mathsf{CKGen}_{\mathsf{4P\&OW}}, \mathsf{Comb}_{\mathsf{4P\&OW}})$ for functions $\mathcal{H}_0, \mathcal{H}_1$ the key generation algorithm generates a tuple (π, H_0, H_1) consisting of a pairwise independent permutation π (over $\{0,1\}^{5n}$) and two hash functions $H_0 \leftarrow \mathsf{HKGen}_0(1^n)$ and $H_1 \leftarrow \mathsf{HKGen}_1(1^n)$. The evaluation algorithm $\mathsf{Comb}_{\mathsf{4P\&OW}}^{H_0,H_1,\pi}$ for input $M \in \{0,1\}^{5n}$ computes $P^3(H_0^0(\pi(M))||H_1^0(M))$ where P^3 is the Feistel permutation $P^3 = \psi[H_\oplus^1, H_\oplus^2, H_\oplus^3]$. Note that applying a permutation to the output of a one-way function does not violate the one-way property. We have already proved that the other three properties CR,TCR,MAC which are preserved by $\mathcal{C}_{\mathsf{CR\&OW}}$ are not affected by applying a permutation in Section 4.3.

Theorem 4.15 *The combiner $\mathcal{C}_{\mathsf{4P\&OW}}$ is a strongly robust multi-property combiner for* PROP $= \{\mathsf{CR}, \mathsf{PRF}, \mathsf{TCR}, \mathsf{MAC}, \mathsf{OW}\}$.

When we apply the modifications from Section 4.5 and the combiner $\mathsf{Comb}_{\mathsf{4P\&IRO}}$ from Section 4.4 together, we get our construction $\mathcal{C}_{\mathsf{6P}}$ (cf. Figure 4.2). This construction is defined like $\mathcal{C}_{\mathsf{4P\&IRO}}$, where one additionally applies a pairwise-independent permutation over $\{0,1\}^{kn}$ (with $k \geq 5$) to the input of H_0^0.

Theorem 4.16 *The combiner $\mathcal{C}_{\mathsf{6P}}$ is a strongly robust multi-property combiner for* PROP $= \{\mathsf{CR}, \mathsf{TCR}, \mathsf{PRF}, \mathsf{MAC}, \mathsf{OW}, \mathsf{IRO}\}$.

4.6 Weak vs. Mild vs. Strong Robustness

In this section we revert to our different notions of multi-property robustness as introduced in Section 4.2, and analyze the relations among the three variants. The first proposition shows that strong robustness implies mild robustness which, in turn, implies weak robustness. The proof is straightforward and given only for sake of completeness:

Proposition 4.17 *Let* PROP *be a set of properties. Then any strongly robust multi-property combiner for* PROP *is also mildly robust for* PROP*, and any mildly robust combiner for* PROP *is also weakly robust for* PROP*.*

Proof. Assume that the combiner is sMPR for PROP. Suppose further that $\text{PROP}(\mathcal{C}) \not\subseteq \text{PROP}$ such that there is some property $\mathsf{P}_i \in \text{PROP} - \text{PROP}(\mathcal{C})$. Then, since the combiner is sMPR, we must also have $\mathsf{P}_i \notin \text{PROP}(\mathcal{H}_0) \cup \text{PROP}(\mathcal{H}_1)$, else we derive a contradiction to the strong robustness. We therefore have $\text{PROP} \not\subseteq \text{PROP}(\mathcal{H}_0) \cup \text{PROP}(\mathcal{H}_1)$, implying mild robustness via the contrapositive statement.

Now consider an mMPR combiner and assume $\text{PROP} = \text{PROP}(\mathcal{H}_0)$ or $\text{PROP} = \text{PROP}(\mathcal{H}_1)$. Then, in particular, $\text{PROP} = \text{PROP}(\mathcal{H}_0) \cup \text{PROP}(\mathcal{H}_1)$ and the mMPR property says that also $\text{PROP} = \text{PROP}(\mathcal{C})$. This proves sMPR. □

To separate the notions we consider the collision-resistance property CR and the property NZ (*n*on-*z*ero output) that the hash function should return $0 \cdots 0$ with small probability only. This may be for example required if the hash value should be inverted in a field:

non-zero output (NZ): A hash function $\mathcal{H}$ has property NZ if for any efficient adversary $\mathcal{A}$ the probability that for $H \leftarrow \mathsf{HKGen}(1^n)$ and $M \leftarrow \mathcal{A}(H)$ we have $H(M) = 0 \cdots 0$, is negligible.

Lemma 4.18 *Let* $\text{PROP} = \{\mathsf{CR}, \mathsf{NZ}\}$ *and assume that collision-intractable hash functions exist. Then there is a hash function combiner which is weakly multi-property robust for* PROP*, but not mildly multi-property robust for* PROP*.*

Proof. Consider the following combiner (with the standard key generation, $(H_0, H_1) \leftarrow \mathsf{CKGen}(1^n)$ for $H_0 \leftarrow \mathsf{HKGen}_0(1^n)$ and $H_1 \leftarrow \mathsf{HKGen}_1(1^n)$):

> The combiner for input M first checks that the length of M is even, and if so, divides $M = L||R$ into halves L and R, and checks
>
> - that $H_0(L) \neq H_0(R)$ if $L \neq R$, and that $H_0(M) \neq 0 \cdots 0$,
> - that $H_1(L) \neq H_1(R)$ if $L \neq R$, and that $H_1(M) \neq 0 \cdots 0$.
>
> If the length of M is odd or any of the two properties above holds, then the combiner outputs $H_0(M)||H_1(M)$. In any other case, it returns 0^{2n}.

We first show that the combiner is weakly robust. For this assume that the hash function H_b for $b \in \{0, 1\}$ has both properties. Then the combiner returns the exceptional output 0^{2n} only with negligible probability, namely, if one finds an input with a non-trivial collision under H_b and which also refutes property NZ. In any other case, the combiner's output $H_0(M)||H_1(M)$ inherits the properties CR and NZ from hash function H_b.

Next we show that the combiner is not mMPR. Let H_1' be a collision-resistant hash function with $n-1$ bits output (and let H_1 include a description of H_1'). Define the following hash functions:

$$H_0(M) = 1^n, \qquad H_1(M) = \begin{cases} 0^n & \text{if } M = 0^n1^n \\ 1||H_1'(M) & \text{else} \end{cases}.$$

Clearly, H_0 has property NZ but is not collision-resistant. On the other hand, H_1 obeys CR but not NZ, as 0^n1^n is mapped to zeros. But then we have $\text{PROP} = \{\mathsf{CR}, \mathsf{NZ}\} = \text{PROP}(H_0) \cup \text{PROP}(H_1)$ and mild robustness now demands that the combiner, too, has these two properties. Yet, for input $M = 0^n1^n$ the combiner returns 0^{2n} since the length of M is even, but $L = 0^n$ and $R = 1^n$ collide under H_0, and M is thrown to 0^n under H_1. This means that the combiner does not obey property NZ. □

Lemma 4.19 *Let* $\text{PROP} = \{\mathsf{CR}, \mathsf{NZ}\}$*. Then there exists a hash function combiner which is mildly multi-property robust for* PROP*, but not strongly multi-property robust for* PROP*.*

Proof. Consider the following combiner (again with standard key generation):

> The combiner for input M first checks that the length of M is even, and if so, divides $M = L||R$ into halves L and R and then verifies that $H_0(L) \neq H_1(R)$ or $H_1(L) \neq H_1(R)$ or $L = R$. If any of the latter conditions holds, or the length of M is odd, then the combiner outputs $H_0(M)||H_1(M)$. In any other case it returns 0^{2n}.

We first prove that the combiner is mMPR. Given that $\text{PROP} \subseteq \text{PROP}(H_0) \cup \text{PROP}(H_1)$ at least one of the two hash functions is collision-resistant. Hence, even for $M = L||R$ with even length and $L \neq R$, the hash values only collide with negligible probability. In other words, the combiner outputs $H_0(M)||H_1(M)$ with overwhelming probability, implying that the combiner too has properties CR and NZ.

Now consider the constant hash functions $H_0(M) = H_1(M) = 1^n$ for all M. Clearly, both hash functions obey property $\mathsf{NZ} \in \text{PROP}(H_0) \cup \text{PROP}(H_1)$. Yet, for input 0^n1^n the combiner returns 0^{2n} such that $\mathsf{NZ} \notin \text{PROP}(\mathcal{C})$, implying that the combiner is not strongly robust. □

The proof indicates how mildly (or weakly) robust combiners may take advantage of further properties to implement other properties. It remains open if one can find similar separations for the popular properties like CR and PRF, or for CR and IRO.

4.7 Multiple Hash Functions and Tree-Based Composition of Combiners

So far we have considered combiners for two hash functions. The multi-property robustness definition extends to the case of more hash functions as follows:

Definition 4.20 *For a set* PROP $= \{\mathsf{P}_1, \mathsf{P}_2, \ldots, \mathsf{P}_N\}$ *of properties an m-function combiner* $\mathcal{C} = (\mathsf{CKGen}, \mathsf{Comb})$ *for hash functions* $\mathcal{H}_0, \mathcal{H}_1, \ldots, \mathcal{H}_{m-1}$ *is called*

weakly multi-property robust *(wMPR) for* PROP *iff*

$$\exists j \in \{0, 1, \ldots, m-1\} \text{ s.t. } \text{PROP} = \text{PROP}(\mathcal{H}_j) \quad \Longrightarrow \quad \text{PROP} = \text{PROP}(\mathcal{C}),$$

mildly multi-property robust *(mMPR) for* PROP *iff*

$$\text{PROP} = \bigcup_{j=0}^{m-1} \text{PROP}(\mathcal{H}_j) \quad \Longrightarrow \quad \text{PROP} = \text{PROP}(\mathcal{C}),$$

and strongly multi-property robust *(sMPR) for* PROP *iff for all* $\mathsf{P}_i \in$ PROP,

$$\mathsf{P}_i \in \bigcup_{j=0}^{m-1} \text{PROP}(\mathcal{H}_j) \quad \Longrightarrow \quad \mathsf{P}_i \in \text{PROP}(\mathcal{C}).$$

For the above definitions we still have that sMPR implies mMPR and mMPR implies wMPR. The proof is a straightforward adaption of the case of two hash functions.

Given a combiner for two hash functions one can build a combiner for three or more hash functions by considering the two-function combiner itself as a hash function and applying it recursively. For instance, to combine three hash functions $\mathcal{H}_0, \mathcal{H}_1, \mathcal{H}_2$ one may define the "cascaded" combiner by $\mathcal{C}_2(\mathcal{C}_2(\mathcal{H}_0, \mathcal{H}_1), \mathcal{H}_2)$, where we assume that the output of $\mathcal{C}_2$ allows to be used again as input to the combiner on the next level.

More generally, given m hash functions and a two-function combiner $\mathcal{C}_2$ we define an m-function combiner $\mathcal{C}_{\text{multi}}$ as a binary tree, as suggested for general combiners by [HKN+05]. Each leaf is labeled by one of the m hash functions (different leaves may be labeled by the same hash function). Each inner node, including the root, with two descendants labeled by $\mathcal{F}_0$ and $\mathcal{F}_1$, is labeled by $\mathcal{C}_2(\mathcal{F}_0, \mathcal{F}_1)$.

The key generation algorithm for this tree-based combiner now runs the key generation algorithm for the label at each node (each run independent of the others, even if two nodes contain the same label). To evaluate the multi-hash function combiner one inputs M into each leaf and computes the functions outputs recursively up to the root. The output of the root node is then the output of $\mathcal{C}_{\text{multi}}$. We call this a *combiner tree for* $\mathcal{C}_2$ *and* $\mathcal{H}_0, \mathcal{H}_1, \ldots, \mathcal{H}_{m-1}$.

For efficiency reasons we assume that there are at most polynomially many combiner evaluations in a combiner tree. Also, to make the output dependent on all hash functions we assume that each hash function appears in (at least) one of the leaves. If a combiner tree obeys these properties, we call it an *admissible combiner tree for* $\mathcal{C}_2$ *and* $\mathcal{H}_0, \mathcal{H}_1, \ldots, \mathcal{H}_{m-1}$.

We first show that weak MPR and strong MPR preserve their properties for admissible combiner trees:

Proposition 4.21 *Let* $\mathcal{C}_2$ *be a weakly (resp. strongly) multi-property robust two-function combiner for* PROP. *Then any admissible combiner tree for* $\mathcal{C}_2$ *and functions* $\mathcal{H}_0, \mathcal{H}_1, \ldots, \mathcal{H}_{m-1}$ *for* $m \geq 2$ *is also weakly (resp. strongly) multi-property robust for* PROP.

Proof. We give the proof by induction for the depth of the tree. For depth $d = 1$ we have $m = 2$ and $\mathcal{C}_{\text{multi}}(\mathcal{H}_0, \mathcal{H}_1) = \mathcal{C}_2(\mathcal{H}_0, \mathcal{H}_1)$ or $\mathcal{C}_{\text{multi}}(\mathcal{H}_0, \mathcal{H}_1) = \mathcal{C}_2(\mathcal{H}_1, \mathcal{H}_0)$ and the claim follows straightforwardly for both cases.

Now assume $d > 1$ and that combiner $\mathcal{C}_2$ is wMPR. Then the root node applies $\mathcal{C}_2$ to two nodes N_0 and N_1, labeled by $\mathcal{F}_0$ and $\mathcal{F}_1$. Note that by the wMPR prerequisite we assume that there exists one hash function $\mathcal{H}_j$ which has all properties in PROP. Since this hash functions appears in at least one of the subtrees under N_0 or N_1, it follows by induction that at least one of the functions $\mathcal{F}_0$ and $\mathcal{F}_1$, too, has properties PROP. But then the combiner application in the root node also inherits these properties from its descendants.

Now consider $d > 1$ and the case of strong MPR. It follows analogoulsy to the previous case that for each property $\mathsf{P}_i \in$ PROP, one of the hash functions in the subtrees rooted at N_0 and N_1 must have property P_i as well. This carries over to the combiners at nodes N_0 or N_1 by induction, and therefore to the root combiner. □

Somewhat surprisingly, mild MPR in general does not propagate security for tree combiners, as we show by a counter-example below. Note that we still obtain, via the previous proposition, that the mMPR combiner is also wMPR and that the resulting tree combiner is thus also wMPR. Yet, it loses its mMPR property.

Proposition 4.22 *Let* PROP $= \{\mathsf{CR}, \mathsf{NZ}\}$ *and assume that there are collision-intractable hash functions. Then there exists a two-function weakly robust multi-property combiner* $\mathcal{C}_2$ *for* PROP*, and an admissible tree combiner for* $\mathcal{C}_2$ *and hash functions* $\mathcal{H}_0, \mathcal{H}_1, \mathcal{H}_2$ *which is not mildly multi-property robust for* PROP.

Proof. Consider the following two-function combiner $\mathcal{C}_2$ for hash functions $\mathcal{H}_0, \mathcal{H}_1$ (again with standard key generation):

> For input M check that the length of M is even and, if so, divide $M = L||R$ into halves L and R. If $H_0(L) = H_0(R)$ and $H_1(L) =$

$H_1(R)$ and $L \neq R$, or we have $H_0(M) = 0 \cdots 0$ and $H_1(M) = 0 \cdots 0$, then output $0^{|H_0(M)|+|H_1(M)|}$. Else, or if the length of M is odd, return $H_0(M)||H_1(M)$.

It is easy to verify that this is an mMPR two-function combiner for PROP. Now consider the following hash functions, where H_2' is a collision-resistant hash function with $n-1$ bits output:

$$H_0(M) = 1^n, \qquad H_1(M) = 1^n, \qquad H_2(M) = \begin{cases} 0^n & \text{if } M = 0^n 1^n \\ 1||H_2'(M) & \text{else} \end{cases}.$$

Then $\text{PROP}(\mathcal{H}_0) = \text{PROP}(\mathcal{H}_1) = \{\mathsf{NZ}\}$ and $\text{PROP}(\mathcal{H}_2) = \{\mathsf{CR}\}$ such that $\text{PROP} = \bigcup \text{PROP}(\mathcal{H}_j)$.

Consider the following tree combiner defined through $\mathcal{C}(\mathcal{H}_0, \mathcal{H}_1, \mathcal{H}_2) = \mathcal{C}_2(\mathcal{C}_2(\mathcal{H}_0, \mathcal{H}_1), \mathcal{H}_2)$, i.e., which cascades the three hash functions. Then the inner application of $\mathcal{C}_2$ yields a hash function which returns 0^{2n} for message $M = 0^n 1^n$. Since this message also causes H_2 to return 0^n the tree combiner runs into the exception case and returns 0^{3n} for this input. Hence, the tree combiner does not have property NZ. □

Note that the cascading combiner can also be applied to all our proposed MPR combiners to compose three or more hash functions. The derived combiner, however, is less efficient than the direct construction sketched there.

Chapter 5

Hash Function Combiners in TLS and SSL

In theory, theory and practice are the same. In practice, they are not.
— Albert Einstein

In this last chapter we analyze the proposed combiner constructions in the TLS and SSL protocols. Both protocols are widely used to ensure secure communication over an untrusted network, and deploy combinations of MD5 and SHA1 to establish shared keys that are as secure as possible.

We first present the preliminaries for our investigation of the combiners in SSL/TLS in Section 5.2. In Section 5.3 we analyze the hash combiners that are used to derive the shared master secret regarding their robustness for pseudorandomness. The combiner constructions for message authentication that both protocols employ to strengthen the final authentication step in the handshake phase, are discussed in Section 5.4.

An extended abstract of this chapter is published in [FLW10].

5.1 Introduction

The SSL protocol [SSL94] was published in 1994 by Netscape to provide secure communication between two parties over an untrusted network, and subsequently formed the basis for the TLS protocol [TLS99, TLS06]. Nowadays, both protocols are ubiquitously present in various applications such as electronic banking, online shopping or secure data transfer. Interestingly, TLS and SSL use various combinations of MD5 and SHA1 instead of relying only on a single hash function in order to strengthen their security guarantees. However, neither TLS nor SSL were accompanied with rigorous security proofs. An important step was recently done by Morrissey et al. [MSW08] who gave the first security analysis of the handshake protocol of TLS. The handshake protocol is the essential part of TLS/SSL as it allows a client and server to

negotiate security parameters, such as shared symmetric keys or trusted ciphers, without having any common secrets yet. The established keys and cryptographic algorithms are subsequently used to protect the data transfer, i.e., the confidentiality and authenticity of the entire communication relies on the security of the key agreement. Thus, it is of crucial importance that the handshake protocol provides reliable parameters. Ideally, this statement should be fortified by comprehensive security proofs.

Our Results. In this chapter, we scrutinize the design of the (non-standard) hash combiners, deployed in the TLS and SSL handshake protocols, regarding their suitability for the respective purposes. As already mentioned, secure key derivation is one of the main tasks of the handshake phase. Both TLS and SSL use hash combiners to compute the master secret out of the pre-master secret, which is assumed to be a shared random string. To achieve secure key-derivation, robustness with respect to pseudorandomness is required.

While TLS (mainly) reverts to the standard design for PRF combiners, i.e., it xors the outputs of the two hashes, SSL applies the cascade $H_0(k, (H_1(k, m))$ as the pseudorandom function for key derivation. For SSL we prove that the combiner is not robust and not even preserving, i.e., even two secure PRFs may yield an insecure combiner. This stems from the fact that both hash functions are invoked with the same (pre) master key. By using individual keys for each underlying function, we show that the security of the SSL combiner is somewhat between robustness and property-preservation. In the case of TLS, we prove that the combiner is a secure PRF if either of H_0, H_1 is a pseudorandom function. Interestingly, the construction is neither optimal in terms of security nor efficiency. We therefore also discuss possible tweaks to obtain better security bounds while saving on computation.

TLS and SSL also use hash combiners for the finished message in the handshake protocol, which is basically a message authentication code generated for the shared master secret and all previous handshake messages. This concludes the key exchange phase in TLS/SSL and authenticates the previous communication. Ideally, the combiners used for this purpose should be robust for MACs, i.e., rely only on unforgeability instead of pseudorandomness of the hash function.[1]

We show that in TLS the combiner for authentication requires the additional assumption of at least one hash function being collision-resistant. The combiner used in SSL is again neither robust nor preserving, due to the same problem of using the master secret as key for both functions. We discuss

[1]The devil's advocate may claim that we can already start from the assumption that one of the hash function is a PRF, as we require this for the key derivation step anyway. However, it is a common principle to revert to the minimal requirements for such sub protocols and their designated purpose. Suppose, for example, that both hash functions turn out to be *not* pseudorandom, that key derivation becomes insecure and confidentiality of the subsequently transmitted data is breached. Then a secure authentication step in the finished message via the MAC-robust combiner would still guarantee authenticity of the designated partner.

that the modified version which splits the key into independent halves, is a secure MAC when at least one hash function is simultaneously unforgeable and collision-resistant.

In summary, we give the first formal treatment of the hash combiners deployed in the TLS and SSL protocols. Our results essentially show that the choices in TLS are sound as they follow common design criteria for such combiners (but still leave space for improvements), whereas the SSL design for combiners requires much stronger assumptions. Our result, together with other steps like the security proofs in [MSW08, GMP+08], strengthen the confidence in the important protocols TLS and SSL.

5.2 Preliminaries

So far, this thesis mainly dealt with conceptual results which we analyzed in terms of asymptotic security. That is, we usually showed the existence of a polynomial time security reduction from the combiner to the underlying hash functions. However, as we consider practical protocols in this chapter, we switch to the *concrete security approach* where the definition of the advantage already measures the quality of the transformation from one primitive to another. To this end, we first restate the notions of hash functions and their properties (which are relevant in this chapter) in terms of concrete security.

5.2.1 Hash Function and Their Properties

Since we give all results in terms of concrete security we adopt Rogaway's approach [Rog06] of defining hash functions as single instances (instead of families) and considering constructive reductions between security properties. For security notions without secret keys like collision-resistance the adversary is implicitly based on (the description of) H, whereas for security properties involving secret keys like pseudorandomness or the MAC property, the adversary also gets black-box access to the hash function $H(k,\cdot)$ with secret key k. In this case we call H a keyed hash function and usually denote the key space by K.

Collision-Resistance. Let $H : \{0,1\}^* \to \{0,1\}^n$ be a hash function. The *collision-finding advantage* of an adversary $\mathcal{A}$ is

$$\mathbf{Adv}^{\mathrm{cr}}_H(\mathcal{A}) = \mathrm{Prob}\left[(M, M') \leftarrow \mathcal{A}() : M \neq M' \wedge H(M) = H(M')\right].$$

We note that, formally, for any hash function there is a very efficient algorithm $\mathcal{A}$ with advantage 1, namely, the one which has a collision hardwired into it and simply outputs this collision. However, based on current knowledge it is usually infeasible to specify this algorithm constructively.

Pseudorandomness. Let $H : K \times \{0,1\}^* \to \{0,1\}^n$ be a keyed hash function with key space K. We define the advantage of a distinguisher $\mathcal{A}$ as

$$\mathbf{Adv}_H^{\text{prf}}(\mathcal{A}) = \left|\text{Prob}\left[\mathcal{A}^{H(k,\cdot)} = 1\right] - \text{Prob}\left[\mathcal{A}^{f(\cdot)} = 1\right]\right|$$

where the probability in the first case is over $\mathcal{A}$'s coin tosses and the random choice of $k \leftarrow K$, and in the second case over $\mathcal{A}$'s coin tosses and the choice of the random function $f : \{0,1\}^* \to \{0,1\}^n$.

Message Authentication (Unforgeability). Let $H : K \times \{0,1\}^* \to \{0,1\}^n$ be a keyed (deterministic) hash function with key space K. We define the *forgeability advantage* of an adversary $\mathcal{A}$ as

$$\mathbf{Adv}_H^{\text{mac}}(\mathcal{A}) = \\ \text{Prob}\left[k \leftarrow K, (M,\tau) \leftarrow \mathcal{A}^{H(k,\cdot)} : H(k,M) = \tau \wedge M \text{ not queried}\right].$$

Hash Function Combiners Recall, that a combiner for hash functions H_0, H_1 is called *robust* if it obeys the property if at least *one* of the two underlying functions has the corresponding property. In terms of our concrete security statements, collision-resistance robustness for example is formulated by demanding that the probability of finding collisions in a combiner is bounded from above by the minimum of finding collisions for the individual hash functions.

5.2.2 HMAC

Each hash function can be used as a pseudorandom function or MAC by replacing the initial value IV with a randomly chosen key k of the same size. A more convenient technique was proposed by Bellare et al. [BCK96a] with the HMAC/NMAC algorithms, which are message authentication codes built from iterated hash functions. Recall that a MAC takes a secret key k, message M and outputs a tag τ. The HMAC algorithm takes, in its more general version, two keys $k_{\text{in}}, k_{\text{out}}$ and applies an iterated hash function H like MD5 and SHA1 in a nested manner:

$$\mathsf{HMAC}(k_{\text{in}}, k_{\text{out}})(M) = H(IV, k_{\text{out}} || H(IV, k_{\text{in}} || M)) \tag{5.1}$$

In practice, HMAC typically uses only a single key k from which it derives dependent keys $k_{\text{in}} = k \oplus \mathsf{ipad}$ and $k_{\text{out}} = k \oplus \mathsf{opad}$ for fixed constants $\mathsf{ipad} = 0x3636\ldots36, \mathsf{opad} = 0x5c5c\ldots5c$.

Originally, Bellare et al. [BCK96a] proved HMAC – resp. its theoretical counterpart NMAC – to be pseudorandom functions when the underlying compression function h is pseudorandom and collision-resistant. Subsequently, the proof was restated on the sole assumption that the compression function is pseudorandom [Bel06] or non-malleable [Fis08]. As the security claims are

given for NMAC, Bellare [Bel06] introduced the notion of a "dual" pseudo-random function $\bar{h} : \{0,1\}^n \times K \to \{0,1\}^n$ with $\bar{h}(m,k) = h(k,m)$. If both $\bar{h}$ and h are pseudorandom, the security of NMAC carries over to HMAC:

Lemma 5.1 *Let $h : \{0,1\}^n \times \{0,1\}^\ell \to \{0,1\}^n$ be a compression function with key space $\{0,1\}^n$. Let $IV \in \{0,1\}^n$ be a fixed initialization vector, and let $\mathsf{HMAC} : \{0,1\}^n \times \{0,1\}^* \to \{0,1\}^n$ be defined as in (5.1). For any adversary $\mathcal{A}$ against HMAC that makes q queries each of at most l blocks and runs in time at most t, there exist adversaries $\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3$ such that*

$$\mathbf{Adv}^{prf}_{\mathsf{HMAC}}(\mathcal{A}) \;\leq\; 2\mathbf{Adv}^{prf}_{\bar{h}}(\mathcal{A}_1) + \mathbf{Adv}^{prf}_{h}(\mathcal{A}_2) + \binom{q}{2}\left[2l \cdot \mathbf{Adv}^{prf}_{h}(\mathcal{A}_3) + 2^{-n}\right]$$

where $\mathcal{A}_1$ makes a single query IV and runs in time at most t. $\mathcal{A}_2$ makes at most q queries and runs in time at most t, while $\mathcal{A}_3$ makes at most 2 oracle queries and runs in time at most $\mathcal{O}(lT_h)$ where T_h denotes the time required for one evaluation of h.

For the single-keyed HMAC-version, the security of $\bar{h}$ must hold against related-key attacks as well. That is, the adversary is granted access to two oracles $\bar{h}(k \oplus \mathsf{opad}, \cdot), \bar{h}(k \oplus \mathsf{ipad}, \cdot)$ with dependent keys.

5.2.3 The SSL/TLS Handshake Protocol

The SSL and TLS protocols consist of two layers: the record layer and the handshake protocol. The record layer encrypts all data with a cipher and session key that have been negotiated by the handshake protocol. Thus the handshake protocol is a key-exchange protocol layered above the record layer and initializes and synchronizes a cryptographic state between a server and a client. Both versions of the handshake protocol, for TLS and for SSL, vary mainly in the implementation of the exchanged messages, i.e., the overall structure of the handshake part is the same and can be summarized as the sequence of the following steps [Res01] (see also Figure 5.1):

(1) The client conveys its willingness to engage in the protocol by sending a list of supported cipher algorithms and a random number, that is subsequently used for key-derivation.

(2) The server responds by choosing one of the proposed ciphers, and sending its certified public key as well as a random nonce.

(3) The client verifies the validity of the received certificate and sends a randomly chosen *pre-master secret* encrypted under the server's public key.

An alternative to having the client choose the pre-master secret is to engage in a key exchange protocol like signed Diffie-Hellman. Since our analysis below only assumes that the pre-master secret is random we omit the details about its generation.

(4) Both client and server derive independently a *master secret* from the exchanged random nonces and the pre-master secret. Once the master key is computed, it can be used to obtain further application keys.

(5+6) Finally, the master secret is confirmed by the *finished message*, where each party sends a MAC over the transcript of the conversation using the new master key. This is also the first transmission which uses the secure channel for the derived keys.

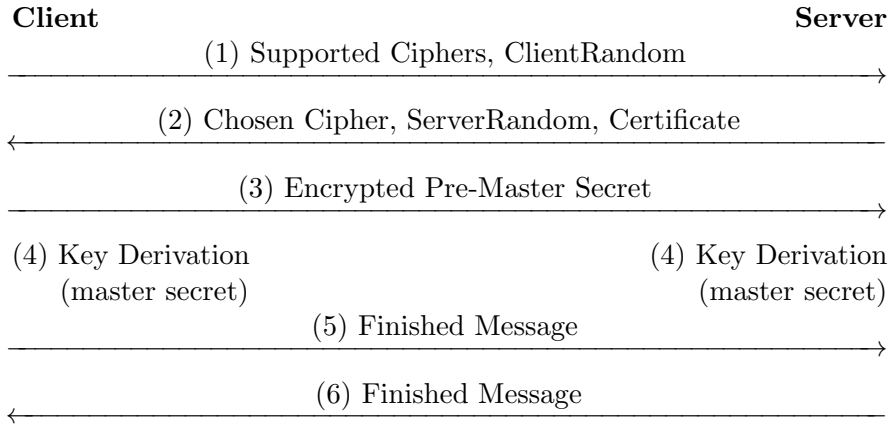


Figure 5.1: Overview of the TLS/SSL handshake protocol [Res01]

5.3 Derivation of the Master Secret

In this section we analyze the functions that are deployed by TLS and SSL to derive a secret master key from a shared pre-master key. The basic requirement of key derivation is that the obtained key should be indistinguishable from a randomly chosen one. In particular, the key-derivation function must be pseudorandom. For more discussions see [Kra08]. We will show that the combiner proposed by TLS is PRF-robust, i.e., security of one of the underlying hash function suffices, whereas the SSL combiner requires assumptions on both hash functions in order to produce random looking output.

5.3.1 The PRF-Combiner used in TLS

The TLS key derivation obtains the master secret (ms) from the pre-master secret (pms) by invoking the following hash combiner:

$$ms = \mathsf{Comb}^{\mathsf{MD5,SHA1}}_{\mathsf{TLS-prf}}(pms, \text{“master secret”}, \text{ClientRandom}||\text{ServerRandom})[0..47]$$

The pre-master secret is assumed to be a random value both parties have agreed upon, and *ClientRandom* and *ServerRandom* are public random nonces exchanged in the handshake protocol. By introducing a specific label (here "master secret") to the input, the combiner can subsequently be used for further (key-derivation) computations, while guaranteeing distinct inputs for each application. The appendix [0..47] indicates that the master secret consists of the first 48 bytes of the combiners output.

Basically, the combiner $\mathsf{Comb}^{H_0,H_1}_{\mathsf{TLS-prf}}$ xors the output of a function T which gets called twice based on two distinct hash functions H_0 and H_1. To this end, the combiner also splits the key $K = k_0||k_1$ with $|k_1| = |k_0|$ into independent halves:

$$\mathsf{Comb}^{H_0,H_1}_{\mathsf{TLS-prf}}(k_0||k_1, M) = \mathsf{T}_{H_0}(k_0, M) \oplus \mathsf{T}_{H_1}(k_1, M) \tag{5.2}$$

The underlying function T_{H_b} makes several queries to HMAC_{H_b} and produces byte strings of (arbitrary) length that is a positive multiple of n.

$$\mathsf{T}_{H_b}(k, M) = \mathsf{HMAC}_{H_b}(k, A(1)||M) \;||\; \mathsf{HMAC}_{H_b}(k, A(2)||M)|| \ldots \tag{5.3}$$

with $A(0) = M$ and $A(i) = \mathsf{HMAC}_{H_b}(k, A(i-1))$.

Analysis of $\mathsf{Comb}^{H_0,H_1}_{\mathsf{TLS-prf}}$. We show that the TLS-combiner for key derivation is a pseudorandom function if at least one of the two hash functions H_0, H_1 is based on a pseudorandom compression function. To this end, we first show that T_{H_b} inherits the pseudorandomness of the underlying hash function.

Note that the T_{H_b} construction uses the HMAC transform to obtain a PRF, which gets keyed via the input data, out of a standard hash function H_b with fixed IV. It was shown in [Bel06] that HMAC is a pseudorandom function, when the underlying compression-function is a *dual PRF*, i.e., it has to be a secure PRF when keyed by either the data input or the chaining value. Thus, while functional-wise HMAC uses the cryptographic hash function only as a black-box, the security guarantee is still based on the underlying compression function h_b. We therefore consider each hash function $H_b : \{0,1\}^* \to \{0,1\}^n$ as the Merkle-Damgård iteration of a compression function $h_b : \{0,1\}^n \times \{0,1\}^\ell \to \{0,1\}^n$. By applying Lemma 5.1 we can conclude that HMAC_{H_b} is a pseudorandom function, when h_b is a dual PRF.

Next, we show that the design of the T_H construction preserves the pseudorandomness of HMAC_H. For a modular analysis – and for the sake of readability – we simplify the description of T_H by replacing HMAC and the hash function H by the same function H, and prove that the modified function T'_{H} is a pseudorandom function if H is. Furthermore, we make a rather syntactical change of T to obtain a function that is efficiently computable on its own: According to the TLS specification, the T construction produces output of arbitrary length from which the combiner takes as much bytes as required, e.g., the first 48 bytes in case of the derivation of the master secret. In the

following we slightly deviate from that notation and assume that T gets also parametrized by an integer c which indicates that an output of length $c \cdot n$ is requested. Overall, we analyze the following function T':

$$\mathsf{T}'_{\mathsf{H}}(k, M, c) = \\ \mathsf{H}(k, A(0)||M) \;||\; \mathsf{H}(k, A(1)||M) \;||\; \ldots \;||\; \mathsf{H}(k, A(c-1)||M) \tag{5.4}$$

where $A(0) = M, \quad A(i) = \mathsf{H}(k, A(i-1))$.

Lemma 5.2 *Let $\mathsf{H} : \{0,1\}^n \times \{0,1\}^* \to \{0,1\}^n$ be a pseudorandom function with key space $\{0,1\}^n$, and let $\mathsf{T}'_{\mathsf{H}} : \{0,1\}^n \times \{0,1\}^* \to \{0,1\}^{c \cdot n}$ be defined by (5.4) above. For all adversaries $\mathcal{A}$ running in time t, making q queries of length at most l and with $c \leq c_{\max}$, there exist an adversary $\mathcal{B}$ such that*

$$\mathbf{Adv}^{prf}_{\mathsf{T}'}(\mathcal{A}) \;\leq\; \mathbf{Adv}^{prf}_{\mathsf{H}}(\mathcal{B}) + q \cdot \binom{c_{max}}{2} \cdot 2^{-n}$$

where $\mathcal{B}$ makes at most $2c_{max} \cdot q$ queries, each of length at most $l + n$ and runs in time at most $t + \mathcal{O}(c_{max})$.

Proof. Assume that there is an adversary $\mathcal{A}$ that can distinguish the function $\mathsf{T}'_{\mathsf{H}}(k, \cdot)$ from a random function $F : \{0,1\}^* \to \{0,1\}^n$ with advantage $\mathbf{Adv}^{\mathrm{prf}}_{\mathsf{T}'}(\mathcal{A})$. Given $\mathcal{A}$ we show how to obtain an adversary $\mathcal{B}$ against the underlying hash function $\mathsf{H}(k, \cdot)$. Recall that $\mathcal{A}$ has black-box access to an oracle that is either the keyed construction $\mathsf{T}'_{\mathsf{H}}(k, \cdot, \cdot)$ or a random function $F : \{0,1\}^* \to \{0,1\}^*$ (where, formally, F also takes the parameter c as additional input and outputs strings of length cn). The distinguisher $\mathcal{B}$ has to simulate this oracle with the help of its own oracle, which is either the keyed hash-function $\mathsf{H}(k, \cdot)$ or a random function $f : \{0,1\}^* \to \{0,1\}^n$. To this end, for any query (M, c) of $\mathcal{A}$, the adversary $\mathcal{B}$ mimics the construction T' but replaces each evaluation of the underlying hash function H by the response of its oracle on the corresponding query. If $\mathcal{A}$ stops outputting its guess d, algorithm $\mathcal{B}$ stops with output d too.

If the oracle of $\mathcal{B}$ was the hash function H, then $\mathcal{B}$ perfectly simulates the construction T'. Thus, the output distribution of $\mathcal{B}$ equals the one of $\mathcal{A}$ with access to T':

$$\mathrm{Prob}\left[\mathcal{B}^{\mathsf{H}(k,\cdot)} = 1\right] = \mathrm{Prob}\left[\mathcal{A}^{\mathsf{T}'_{\mathsf{H}}(k,\cdot,\cdot)} = 1\right].$$

In the case that the oracle of $\mathcal{B}$ was the truly random function f, we have to show that processing its random answers in the T' construction yields random values again. Recall that for each query (M, c) the adversary $\mathcal{B}$ now computes the sequence $f(A(0)||M) \;||\; f(A(1)||M) \;||\ldots||\; f((A(c-1)||M)$ where $A(i) = f(A(i-1))$ starting with $A(0) = M$. As long as $A(i) \neq A(j)$ for all $i \neq j \in \{0, 1, \ldots c-1\}$ holds for each query, the function f gets evaluated in the outer iterations on distinct and unique values, such that the corresponding outputs

from f are independently and uniformly distributed. Thus, it remains to show that the probability for collisions on the $A(i)$ values, which are derived using f in a cascade, is small. Assume that for a query (M, c) a collision occurred, i.e., there exist (unique) indices $i^* \in \{0, \ldots, c-1\}$ and $j^* \in \{0, \ldots, i^*-1\}$ such that $f(A(i^*-1)) = A(j^*)$ but $A(i^*-1) \neq A(j)$ for all $j = 0, 1, \ldots, i^*-2$. That is, f has never been invoked on the value $A(i^*-1)$ but maps to an value $A(j^*)$ which is an previous answer of (the cascade of) f. Since f is a truly random function, such a collision can only occur with probability $q \cdot \binom{c_{\max}}{2} \cdot 2^{-n}$ where q denotes the number of $\mathcal{A}$ queries and $c_{\max}$ is the largest value for c that appeared in the simulation.

Overall, the output distribution of $\mathcal{B}^f$ results from

$$\begin{aligned} \text{Prob}\Big[\mathcal{B}^f = 1\Big] &\leq \text{Prob}\Big[\mathcal{B}^f = 1 \,\Big|\, \text{no Collision}\Big] + \text{Prob}\,[\,\text{Collision}] \\ &= \text{Prob}\,\big[\,\mathcal{A}^F = 1\big] + q \cdot \binom{c_{\max}}{2} \cdot 2^{-n}. \end{aligned}$$

Thus, $\mathcal{B}$ distinguishes H from f with probability:

$$\begin{aligned} &\text{Prob}\Big[\mathcal{B}^{\mathsf{H}(k,\cdot)} = 1\Big] - \text{Prob}\Big[\mathcal{B}^f = 1\Big] \\ &\quad \geq \text{Prob}\Big[\mathcal{A}^{\mathsf{T}_{\mathsf{H}}(k,\cdot,\cdot)} = 1\Big] - \text{Prob}\,\big[\,\mathcal{A}^F = 1\big] - q \cdot \binom{c_{\max}}{2} \cdot 2^{-n}. \end{aligned}$$

This proves the claim. □

Putting Lemma 5.1 and Lemma 5.2 together, we now obtain that the pseudorandomness of h_b is preserved by the corresponding construction HMAC_{H_b} and, in turn, by $\mathsf{T}'_{\mathsf{HMAC}_{H_b}}$ which equals T_{H_b}. Furthermore, XOR is a robust combiner for pseudorandom functions, and thus, if least one of $\mathsf{T}_{H_0}, \mathsf{T}_{H_1}$ is a PRF, also $\mathsf{Comb}^{H_0,H_1}_{\mathsf{TLS-prf}}$ provides outputs that are indistinguishable from random. This, together with the fact that the key is divided into independent halves, implies the following theorem:

Theorem 5.3 *Let $H_b : \{0,1\}^n \times \{0,1\}^* \to \{0,1\}^n$ for $b \in \{0,1\}$ be a hash function with underlying compression function $h_b : \{0,1\}^n \times \{0,1\}^\ell \to \{0,1\}^n$. Let $\mathsf{Comb}^{H_0,H_1}_{\mathsf{TLS-prf}}$ be defined as in (5.2). For all adversaries $\mathcal{A}$ running in time t, making q queries of length at most l and such that $c \leq c_{\max}$, there exist adversaries $\mathcal{A}_0, \mathcal{A}_1$ such that*

$$\begin{aligned} &\mathbf{Adv}^{prf}_{\mathsf{Comb}_{\mathsf{TLS-prf}}}(\mathcal{A}) \\ &\quad \leq \min\Big\{\mathbf{Adv}^{prf}_{\mathsf{HMAC}_{h_0}}(\mathcal{A}_0), \mathbf{Adv}^{prf}_{\mathsf{HMAC}_{h_1}}(\mathcal{A}_1)\Big\} + q \cdot \binom{c_{max}}{2} \cdot 2^{-n} \end{aligned}$$

where each of $\mathcal{A}_0, \mathcal{A}_1$ makes at most $2c_{max} \cdot q$ queries of length at most $l + n$ and runs in time at most $t + \mathcal{O}(c_{max}(1 + 2q \cdot T_{\bar{b}}))$ where $T_{\bar{b}}$ denotes the time required for one evaluation of $\mathsf{T}_{\bar{b}}$ (as defined in (5.3)).

Improvements. When the combiner $\mathsf{Comb}^{H_0,H_1}_{\mathsf{TLS-prf}}$ is used for key derivation, the underlying construction T ensures that sufficiently many output bytes are produced. However for the purpose of range extension of a PRF, the construction T is neither optimal in terms of efficiency nor security. Namely, if one assumes HMAC_H to be a secure PRF, one could simply augment the input M by a fixed-length encoded counter $\langle i \rangle$, which ensures distinct inputs for each PRF evaluation:

$$\mathsf{T}^*_{H_b}(k, M) = \mathsf{HMAC}_{H_b}(k, M||\langle 1 \rangle) \;||\; \mathsf{HMAC}_{H_b}(k, M||\langle 2 \rangle)||\ldots$$

Replacing T with T^* would result in better security bounds, as one gets rid of the probability $q \cdot \binom{c_{\max}}{2} \cdot 2^{-n}$ of a collision on the $A(i)$ values. In terms of efficiency, the above construction only requires half of the PRF evaluations as needed in the original T function.

Another solution is to use solely the outputs of $A(\cdot)$, i.e., without feeding them into HMAC again:

$$\mathsf{T}^*_{H_b}(k, M) = A(1) \;||\; A(2) \;||\; A(3) \;||\; \ldots$$

with $A(i)$ being the i-th cascade of $\mathsf{HMAC}(k, M)$ as defined in (5.3). With this construction one inherits the same security bound as in the original solution, but invokes HMAC after the first evaluation only one shorter inputs, e.g., 128 bits in the case of $\mathsf{MD5}$ and 160 bits for $\mathsf{SHA1}$, which decreases the computational costs.

5.3.2 The PRF-Combiner used in SSL

In the SSL protocol the following construction gets repeated until sufficient key material for the master secret is generated:

$$\begin{aligned} ms \;=\; & \mathsf{MD5}(pms||(\mathsf{SHA1}(\text{“A”}||pms||\text{ClientRandom}||\text{ServerRandom}))|| \\ & \mathsf{MD5}(pms||(\mathsf{SHA1}(\text{“BB”}||pms||\text{ClientRandom}||\text{ServerRandom}))|| \\ & \mathsf{MD5}(pms||(\mathsf{SHA1}(\text{“CCC”}||pms||\text{ClientRandom}||\text{ServerRandom}))|| \\ & \ldots \end{aligned}$$

Both functions get keyed by the input data, where in the case of the outer hash function the key is prepended to the message, and for the inner hash the key is somewhat embedded in the message. Due to length-extension attacks, key-prepending approaches must be accompanied by prefix-free encoding, otherwise the hash function cannot serve as a pseudorandom function, as shown in [BCK96b]. For the analysis we assume that the hash function takes care of that issue, and thus that a hash function $H_b : \{0,1\}^* \to \{0,1\}^n$ is a secure PRF when keyed via the first n bits of the data input.

On a more abstract level, each repetition of the SSL-combiner above for prefixes “A”, “BB”, “CCC” etc. can be represented as the following construction:

$$\mathsf{Comb}^{H_0,H_1}_{\mathsf{SSL-prf}}(k, M) = H_0(k \;||\; H_1(k||M)), \tag{5.5}$$

e.g., where $H_1(k||M)$ implements $\mathsf{SHA1}(\text{“CCC”}||k||M)$ for the fixed value “CCC”. To be a robust combiner for pseudorandom functions, the SSL-combiner needs to be robust for H_0 and each such function H_1. From now on we fix an arbitrary H_1.

Analysis of $\mathsf{Comb}^{H_0,H_1}_{\mathsf{SSL-prf}}$. The cascade $\mathsf{Comb}^{H_0,H_1}_{\mathsf{SSL-prf}}$ of two hash functions is not a robust design for pseudorandomness, because as soon as the outer function becomes insecure the combiner, too, can be easily distinguished from a random function: Consider as an example the constant function $H_0(x) = 0^n$ that maps any input to zeros, which is obviously distinguishable from random. Then, also the combiner $\mathsf{Comb}^{H_0,H_1}_{\mathsf{SSL-prf}}(k, M) = H_0(k||H_1(k||M))$ becomes a constant function, independently of the strength of the inner hash function H_1. Hence,

Proposition 5.4 *The combiner* $\mathsf{Comb}^{H_0,H_1}_{\mathsf{SSL-prf}}$ *is* not PRF*-robust.*

Actually, $\mathsf{Comb}^{H_0,H_1}_{\mathsf{SSL-prf}}$ is not even PRF-preserving, i.e., there exist two functions H_0, H_1 that are both secure pseudorandom functions, but become easily distinguishable when used in the SSL-combiner. The problem arises from the fact that the same secret key is used for both functions, which contradicts the general design paradigm of provably robust combiners.

For the counter example let $H_1 : K \times \{0,1\}^* \to \{0,1\}^n$ be a pseudorandom function. Define $H_0(k, x)$ now as follows: if $x = H_1(k, 0^n)$ then return 0^n, else output $H_1(k||1||x)$. Then H_0 basically inherits the pseudorandomness of H_1 because any distinguisher with access to $H_0(k, \cdot)$ only retrieves replies $H_1(k||1||x)$ to queries $x \in \{0,1\}^*$, unless it is able to predict the value $H_1(k||0^n)$. The latter would contradict the pseudorandomness of H_1, though. But when both functions are combined into $H_0(k \mid\mid H_1(k||M))$, the combiner returns 0^n for input 0^n and is obviously therefore not a pseudorandom function.

In order to allow any reasonable statement about the security of the construction $\mathsf{Comb}^{H_0,H_1}_{\mathsf{SSL-prf}}$, we assume in the following that the combiner splits the key into two independent halves, and invokes the hash functions on distinct shares:

$$\mathsf{Comb}^{H_0,H_1}_{\mathsf{SSL-prf}^*}(k_0||k_1, M) = H_0(k_0 \mid\mid H_1(k_1||M))$$

Note that the first discussed counter example is still valid, as it did not require any dependencies of the individual keys. Thus, even $\mathsf{Comb}^{H_0,H_1}_{\mathsf{SSL-prf}^*}$ is not a robust combiner in general. However, the security can be considered to be somewhat above property-preservation, since we can relax the assumption on one hash function while the combiner still preserves the pseudorandomness property of the stronger function:

PRF + weakCR = PRF. In the case that the outer hash function H_0 is a secure pseudorandom function, the inner hash function only needs to

ensure that for distinct queries $M \neq M'$ of an adversary to the combiner, the function H_0 gets evaluated on different values too, i.e., $H_1(k, M) \neq H_1(k, M')$ holds for $M \neq M'$. Thus, it suffices for H_1 to be *weakly collision-resistant*, which is defined similarly to collision-resistance, except that here the function is keyed with a secret key and the adversary only gets black-box access to the function. Even though weakCR is a weaker assumption than standard CR, and it is for fixed input-length functions implied by the MAC security, it might suffer for variable length-inputs from the strong attacks against CR [Hir04]. In particular, $\mathsf{MD5}$ and $\mathsf{SHA1}$ are still assumed to be good pseudorandom functions but lack security against weakCR attacks, which was also the reason to restate the proof of HMAC in [Bel06].

weakPRF + PRF = PRF. If the inner hash function H_1 is a pseudorandom function, an adversary that queries the combiner gets to see images of H_0 only for random domain points. Thus, it is not necessary that the outer function is a full-fledged PRF as well. In this case, already the assumption of H_0 being a *weak pseudorandom function* is sufficient. This notion weakens the regular concept of PRFs in the sense that the adversary is only allowed to query the function an random inputs instead of values of his choice. Note that a weakPRF is significantly weaker than a PRF, as e.g., they can exhibit weak input points or be commutative.

insecure+insecure = PRF? One might ask if one can even start with two functions that both are not full PRFs itself, but add up to a secure PRF when used in the combiner construction. It turns out that the SSL combiner allows for two almost entirely insecure hash functions to yield a secure PRF. However, this only holds for very artificial and tailored hash functions, hence, the impact on the security statements for practical considerations is quite limited. We also stress that this is not a particular benefit of the SSL design, as we can obtain similar examples for the TLS combiner as well. For both examples we start from secure PRFs $\mathsf{H}_0, \mathsf{H}_1$ which we modify into functions $\mathsf{H}_0^*, \mathsf{H}_1^*$ that lose their security when used in a stand-alone fashion:

Example for the SSL combiner $\mathsf{Comb}_{\mathsf{SSL-prf}}^{H_0,H_1}$: Our example for SSL holds for the weaker version of the combiner where both functions get keyed with the same master secret, and which we showed to be neither PRF-robust nor preserving in general. Consider the function $\mathsf{H}_1^*(k, M) = k||M$ that simply outputs its secret key and the message it was invoked on. This function is clearly not pseudorandom. The second function $\mathsf{H}_0^*(k, M)$ parses each input as $M = k'||M'$ and checks whether $k = k'$ holds. If so it outputs $\mathsf{H}_0(k, M)$, else 0^n. When H_0^* is used alone, the probability of hitting an input among q queries, whose prefix matches the secret key $k \leftarrow \{0,1\}^n$, is $q \cdot 2^{-n}$ and thus negligible. Hence, with

overwhelming probability one merely gets replies 0^n and can thus easily distinguish this function from random. However, in the combination $\mathsf{H}_0^*(k, \mathsf{H}_1^*(k, M))$, the outer function will always run in the "good" exceptional state where it behaves like H_0 and provides random values. Thus, we exploit the same weakness as above where we showed that the SSL combiner is not even PRF-preserving, but now use that peculiarity to obtain a secure PRF out of insecure functions.

Example for the TLS combiner $\mathsf{Comb}_{\mathsf{TLS-prf}}^{H_0,H_1}$: For TLS we can obtain a similar result by constructing the following two functions $\mathsf{H}_0^*(k, M) = 0^{n/2}||H_0(k, M)|_{n/2}$ and $\mathsf{H}_1^*(k, M) = H_1(k, M)|_{n/2}||0^{n/2}$, where $x|_{n/2}$ denotes the leading $n/2$ bits of string x. Both functions output strings that consist of a constant half and a random half, and are obviously easily distinguishable from a truly random function. By merging them into the TLS combiner, we obtain $\mathsf{H}_0^*(k_0, M) \oplus \mathsf{H}_0^*(k_1, M)$ which nullifies the constant parts and yields random strings again.

5.3.3 Application Key Derivation in TLS and SSL

Both combiners $\mathsf{Comb}_{\mathsf{TLS-prf}}^{H_0,H_1}$ and $\mathsf{Comb}_{\mathsf{SSL-prf}}^{H_0,H_1}$ are used to obtain a shared master secret from a pre-shared key. However, subsequently, the same functions are deployed to derive further keys, e.g., for encryption or message authentication. To this end, the freshly computed master secret is used instead of the pre-master secret that was assumed to be a random value. For TLS we have shown that the combiner $\mathsf{Comb}_{\mathsf{TLS-prf}}^{H_0,H_1}$ provides a master secret that is indistinguishable from random when at least one hash function is a PRF. Thus, our result carries over to the application key derivation, that uses the combiner with the derived master secret. The same holds for SSL, but under stronger assumptions on the underlying hash functions.

5.4 Finished-Message

In this section we investigate the TLS/SSL combiners that are used to compute the so-called *finished*-message of the handshake protocols. The finished message is the last part of the key exchange and is realized by a message authentication code which is computed over the transcript of the previous communication. Thus, the combiners that are used for this application should optimally be robust for MAC, i.e., only rely on the unforgeability property instead of the stronger PRF-assumption.

We note that the finished message itself is already secured through the negotiated application keys. This complicates the holistic security analysis of this step. But since we are at foremost interested in the design of the combiners and their designated purpose, we only touch this issue briefly at the end of

Section 5.4.1 (where we address this issue for TLS; the same discussion holds for SSL).

5.4.1 The MAC-Combiner used in TLS

To compute the finished MAC, the TLS protocol applies the same combiner as for the derivation of the master secret, but already uses the new master key. As the key is known only at the very end of the protocol, the MAC cannot be computed iteratively during the communication. To circumvent the need of storing the entire transcript until the master secret is available, TLS hashes the transcript iteratively and then computes the MAC over the short hash value only:

$$\tau_{\text{finished}} = \mathsf{Comb}^{\mathsf{MD5,SHA1}}_{\mathsf{TLS-prf}}\big(ms,\ \text{FinishedLabel},\ \mathsf{MD5}(\mathsf{transcript})||\mathsf{SHA1}(\mathsf{transcript})\big)[0..11]$$

A further input to the combiner is the FinishedLabel which is either the ASCII string "client" or "server", which ensures that the MAC values of both parties are different, otherwise an adversary could simply return a finished tag back to its sender. The appendix [0..11] indicates again that the first 12 bytes of the combiner output are used as the MAC.

Recall that the combiner $\mathsf{Comb}^{H_0,H_1}_{\mathsf{TLS-prf}}$ is based on the construction T which produces arbitrary length output by invoking the underlying hash function in an iterative and nested manner. However, this range extension is only necessary when the combiner is used for key derivation. To compute the finished message, only the first 12 byte of the combiners output are used, which is shorter than the digests of both applied hash functions (16 bytes for $\mathsf{MD5}$ and 20 bytes for $\mathsf{SHA1}$). Thus, we can omit the T part from the construction and simplify the combiner as follows:

$$\mathsf{Comb}^{H_0,H_1}_{\mathsf{TLS-mac}}(k_0||k_1, M) = \mathsf{HMAC}_{H_0}(k_0, H_0(M)||H_1(M)) \ \oplus\ \mathsf{HMAC}_{H_1}(k_1, H_0(M)||H_1(M)) \tag{5.6}$$

Verification for the above MAC-combiner is done by recomputing the tag and comparing it to the given tag.

Analysis of $\mathsf{Comb}^{H_0,H_1}_{\mathsf{TLS-mac}}$. We have already shown that the combiner construction $\mathsf{Comb}^{H_0,H_1}_{\mathsf{TLS-prf}}$, which can be seen as the more complex version of $\mathsf{Comb}^{H_0,H_1}_{\mathsf{TLS-mac}}$, is robust for pseudorandom functions. Thus, if one is willing to assume that at least one hash function behaves like a random function, the combiner can be used directly as a MAC, as well.

However, ideally, the combiner $\mathsf{Comb}^{H_0,H_1}_{\mathsf{TLS-mac}}$ should be a secure MAC on the sole assumption that at least one of the underlying hash functions H_0, H_1 is unforgeable rather than being a pseudorandom function. Unfortunately, hashing the transcript before the MAC gets computed, imposes another assumption

on the hash functions (even when starting from the PRF assumption), namely at least one hash function needs to be collision-resistant. Otherwise an adversary could try to induce a collision on the input to the HMAC functions, which immediately gives a valid forgery for the entire MAC function. Under the assumption that such a collision is unlikely, we show that the combiner $\mathsf{Comb}^{H_0,H_1}_{\mathsf{TLS-mac}}$ is MAC-robust.

To this end, we first prove that the xor of two deterministic MACs (like HMAC_{H_b}) invoked directly with the message yields a robust combiner:

$$\mathsf{Comb}^{\mathsf{H}_0,\mathsf{H}_1}_{\oplus}(k_0||k_1, M) = \mathsf{H}_0(k_0, M) \oplus \mathsf{H}_1(k_1, M) \tag{5.7}$$

In the context of aggregate authentication, Katz and Lindell [KL08] gave a similar result by showing that multiple MAC tags, computed by (possibly) different senders on multiple (possibly different) messages, can be securely aggregated into a shorter tag by simply xoring them.

Lemma 5.5 *Let* $\mathsf{H}_0, \mathsf{H}_1 : \{0,1\}^n \times \{0,1\}^* \to \{0,1\}^n$ *be deterministic message authenticated codes, and let* $\mathsf{Comb}^{\mathsf{H}_0,\mathsf{H}_1}_{\oplus}$ *be defined by (5.7). For any adversary* $\mathcal{A}$ *against* $\mathsf{Comb}^{\mathsf{H}_0,\mathsf{H}_1}_{\oplus}$ *making at most* q *queries and running in time at most* t*, there exist adversaries* $\mathcal{A}_0, \mathcal{A}_1$ *such that*

$$\mathbf{Adv}^{mac}_{\mathsf{Comb}_{\oplus}}(\mathcal{A}) \leq \min\left\{\mathbf{Adv}^{mac}_{\mathsf{H}_0}(\mathcal{A}_0), \mathbf{Adv}^{mac}_{\mathsf{H}_1}(\mathcal{A}_1)\right\}$$

where $\mathcal{A}_b$ *for* $b = 0, 1$ *makes at most* q *queries and runs in time at most* $t + \mathcal{O}(qT_{\bar{b}})$ *where* $T_{\bar{b}}$ *denotes the time for one evaluation of* $\mathsf{H}_{\bar{b}}$.

Proof. We show that any adversary $\mathcal{A}$ against the combiner implies adversaries $\mathcal{A}_0, \mathcal{A}_1$ against both underlying MACs. Assume towards contradiction that an adversary $\mathcal{A}_{\mathsf{Comb}}$ after making q queries $M_1, \ldots, M_q$ to the $\mathsf{Comb}^{\mathsf{H}_0,\mathsf{H}_1}_{\oplus}(K, \cdot)$ oracle, outputs with some probability a tuple (M^*, τ^*) such that $\tau^* = \mathsf{Comb}^{\mathsf{H}_0,\mathsf{H}_1}_{\oplus}(K, M^*)$ but M^* was never submitted to the combiner oracle. Given $\mathcal{A}_{\mathsf{Comb}}$ we construct a MAC adversary $\mathcal{A}_b$ against the MAC H_b for $b \in \{0,1\}$. This adversary has oracle access to the function $\mathsf{H}_b(k_b, \cdot)$ and uses $\mathcal{A}_{\mathsf{Comb}}$ in a black-box way to produce its forgery. To this end, $\mathcal{A}_b$ first chooses a random key $k_{\bar{b}}$ for $\mathsf{H}_{\bar{b}}$ and then answers each query to the combiner with the help of its oracle access and the knowledge of $k_{\bar{b}}$. When $\mathcal{A}_{\mathsf{Comb}}$ holds, outputting a pair (M^*, τ^*), the adversary $\mathcal{A}_b$ computes its forgery (M^*, τ^*_b) with $\tau^*_b = \tau^* \oplus \mathsf{H}_{\bar{b}}(k_{\bar{b}}, M^*)$.

As M^* was not previously queried by $\mathcal{A}_{\mathsf{Comb}}$, the same holds for $\mathcal{A}_b$. Furthermore, as both MACs are deterministic, the value $\tau^* = \mathsf{H}_0(k_0, M^*) \oplus \mathsf{H}_1(k_1, M^*)$ fixes two well-defined tags for $\mathsf{H}_0, \mathsf{H}_1$. Thus, $\mathcal{A}_b$'s output is equal to the value $\mathsf{H}_b(k_b, M^*)$ for the unknown key k_b and thereby constitutes a valid forgery. Since the adversary $\mathcal{A}$ yields forgers $\mathcal{A}_0, \mathcal{A}_1$ for both MACs, it follows that the advantage cannot exceed the advantage of the smaller of the two security bounds for the MACs. □

Complementing the above Lemma 5.5 with the probability of finding collisions on the concatenated combiner $H_0(M)||H_1(M)$ yields Theorem 5.6.

Theorem 5.6 *Let $H_0, H_1 : \{0,1\}^n \times \{0,1\}^* \to \{0,1\}^n$ be hash functions, and let $\mathsf{Comb}^{\mathsf{H}_0,\mathsf{H}_1}_{\mathsf{TLS-mac}}$ be defined by (5.6). For any adversary $\mathcal{A}$ against $\mathsf{Comb}^{\mathsf{H}_0,\mathsf{H}_1}_{\mathsf{TLS-mac}}$ making at most q queries and running in time at most t, there exist adversaries $\mathcal{A}_0, \mathcal{A}_1, \mathcal{B}_0, \mathcal{B}_1$ such that*

$$\begin{aligned}\mathbf{Adv}^{mac}_{\mathsf{Comb}_{\mathsf{TLS-mac}}}(\mathcal{A}) \\ \leq\ & \min\left\{\mathbf{Adv}^{mac}_{\mathsf{HMAC}_{H_0}}(\mathcal{A}_0), \mathbf{Adv}^{mac}_{\mathsf{HMAC}_{H_1}}(\mathcal{A}_1)\right\} \\ +\ & \min\left\{\mathbf{Adv}^{cr}_{H_0}(\mathcal{B}_0), \mathbf{Adv}^{cr}_{H_1}(\mathcal{B}_1)\right\}\end{aligned}$$

where $\mathcal{A}_b$ for $b = 0, 1$ makes at most q queries and runs in time at most $t + \mathcal{O}(qT_{\bar{b}})$ where $T_{\bar{b}}$ denotes the time for one evaluation of $\mathsf{HMAC}_{H_{\bar{b}}}$, and $\mathcal{B}_b$ runs in time $t + \mathcal{O}(qT_b)$.

Note that for both properties, unforgeability and collision-resistance, it suffices that either one of the hash functions has this property (instead of one hash function with obeying both property simultaneously). This is similar to the difference between weak and strong combiners that we discussed in Section 4.2.

So far, we have reduced the security of the combiner $\mathsf{Comb}^{\mathsf{H}_0,\mathsf{H}_1}_{\mathsf{TLS-mac}}$ of H_0, H_1 to the collision-resistance of the hash functions and the unforgeability of the HMAC transforms HMAC_{H_0} and HMAC_{H_1}. Preferably, the security of HMAC_{H_b} should in turn only rely on the unforgeability of the underlying hash resp. compression function. However, such a reduction for the plain HMAC transform is still unknown. The previous results for this issue either require stronger assumptions than MAC (yet, weaker than PRF), or additional keying-techniques for the compression function. In the following, we briefly recall the two most relevant approaches for our scenario.

An and Bellare [AB99] observed that HMAC can be used to build a VIL-MAC from a FIL-MAC (i.e., from an unforgeable compression function) when the secret key enters each compression function evaluation. As this result was shown for compression functions in the dedicated-key setting, one needs to transform compression functions without a dedicated-key input into keyed ones. This can be done as follows: Let $h : \{0,1\}^n \times \{0,1\}^\ell \to \{0,1\}^n$ be an unkeyed compression function. Then the keyed analogue $kh : \{0,1\}^n \times \{0,1\}^{n+\ell'} \to \{0,1\}^n$ is defined as $kh(k, y, x) = h(y, k||x)$ where $k \in \{0,1\}^n$ is the secret key, $y \in \{0,1\}^n$ the chaining value and $x \in \{0,1\}^{\ell'}$ with $\ell' = \ell - n$ the (shortened) message block[2]. This approach reduces the number of bits

[2] Actually, An and Bellare proposed a transformation that keys the compression function via the chaining value, which would not allow black-box usage of the underlying hash function. We therefore swap the roles of chaining and key value, and assume that kh is a secure MAC when the key occupies the first n bits of each input.

that can be processed in each iteration but allows to use the SHA1 and MD5 compression functions, which do not possess a dedicated key-input. Starting from such a keyed FIL-MAC, the HMAC variant that prepends k_{in} (resp. k_{out} at the final evaluation) to each message block, is proven to be a secure VIL-MAC [AB99]. When HMAC is used only with a single-key k, unforgeability of kh must hold against related key attacks for $kh(k \oplus \Delta_{\mathsf{opad}}, \cdot)$ and $kh(k \oplus \Delta_{\mathsf{ipad}}, \cdot)$. Overall, the first approach relies solely on the unforgeability assumption, but comes with a reduced throughput, e.g., when using SHA1, the HMAC variant that is keyed in each iteration, would require ≈ 1.5 times the compression function evaluations of the standard $\mathsf{HMAC}_{\mathsf{SHA1}}$.

The second approach does not require any modification of HMAC, i.e., it has the same efficiency, but makes stronger assumptions on the compression function: In [Bel06] Bellare proved that NMAC is a secure MAC if the underlying compression function h is a *privacy-preserving MAC* (PP-MAC) and the iteration of h is *computationally almost universal* (cAU). The former resembles the indistinguishability notion for encryption and requires that an adversary given a tag $\mathsf{Mac}(k, M_b)$ for chosen M_0, M_1 cannot determine b. It was shown that PP-MAC is a weaker assumption than PRF and cAU is a milder assumption than weakCR. Fischlin [Fis08] has shown that, alternatively, non-malleability and unpredictability of the compression function suffices, too. In both cases, however, in order to lift the security of NMAC to the single-key version of HMAC, one additionally needs that the dual compression function $\bar{h}$ used to derive $k_{\text{in}}, k_{\text{out}}$ somehow preserves these conditions.

The Problem of Chopping. Theorem 5.6 states that the TLS-combiner for the finished message is robust for message authentication codes even when starting from the unforgeability assumption which is significantly weaker than assuming a PRF. However, according to the TLS specification, not the entire output of the combiner is used as tag, but only the first 12 bytes. Since the unforgeability notion is not closed under chopping transformations, a shortened output of a MAC loses any security guarantees. To allow usage of a chopped fraction of the combiners output, on either has to assume that one of the underlying MACs is secure for truncation, or one needs to make the stronger assumption that at least one of the two hash functions is a secure PRF.

Is Unforgeability Enough? When using MACs in a stand-alone fashion, unforgeability clearly gives sufficient security guarantees. However, in TLS (and SSL) the tag for the finished message is computed under the master secret, from which further application keys for encryption and authentication are derived. The tag itself is now encrypted and authenticated with these derived keys. On one hand, this may help to prevent the tag in the finished message from leaking some information about the master secret. On the other

hand, this causes critical circular dependencies between these values, possibly even enabling leakage of entire keys. This problem has already been noticed in other works (e.g., in [MSW08] where the analysis of the handshake protocol assumes that the tag is sent without securing it with the application keys; or more explicitly in the context of delayed-key authentication in [FL10]). It is beyond the scope of this work about combiners, though.

5.4.2 The MAC-Combiner used in SSL

The SSL-construction for the finished message resembles the HMAC construction, but appends the inner key to the message instead of prepending it. This stems from the same problem as in TLS, namely that the MAC should be computed iteratively as soon as the communication starts, although the necessary key is negotiated only at the end. To obtain a robust design, SSL uses the concatenation of the HMAC-like construction based on the MD5 and SHA1 functions:

$$\tau_{\text{finished}} = \mathsf{HMAC}^*_{\mathsf{MD5}}(ms, \text{Label}||\mathsf{transcript}) \,||\, \mathsf{HMAC}^*_{\mathsf{SHA1}}(ms, \text{Label}||\mathsf{transcript})$$

where HMAC^*_H is defined as:

$$\mathsf{HMAC}^*_H(k, M) = H(k||\mathsf{opad}||\;\; H(M||k||\mathsf{ipad})) \tag{5.8}$$

with $\mathsf{opad}, \mathsf{ipad}$ being the same fixed patterns as in HMAC. The structure of HMAC^* then allows to accomplish the bulk of the computation without knowing the key k.

Overall, the MAC combiner of SSL can be described as follows:

$$\mathsf{Comb}^{H_0,H_1}_{\mathsf{SSL-mac}}(k, M) = \mathsf{HMAC}^*_{H_0}(k, M) \,||\, \mathsf{HMAC}^*_{H_1}(k, M) \tag{5.9}$$

Analysis of $\mathsf{Comb}^{H_0,H_1}_{\mathsf{SSL-mac}}$. In contrast to the TLS-combiner, SSL uses the entire master secret as key for both hash functions. This approach results in a construction $\mathsf{Comb}^{H_0,H_1}_{\mathsf{SSL-mac}}$ that is not even MAC-preserving, although concatenation is MAC-robust when used with distinct keys for each hash function [Her05].

Proposition 5.7 *The combiner* $\mathsf{Comb}^{H_0,H_1}_{\mathsf{SSL-mac}}$ *is* not *MAC-preserving (and thus* not *MAC-robust either).*

Consider two secure MACs $\mathsf{H}_0, \mathsf{H}_1$, that on input of a secret key k and a message M outputs a tag τ_b. Assume furthermore that both MACs ignore parts of their key, i.e., H_0 ignores the left half of its input key and H_1 ignores the right part. We now derive functions H^*_b that can still be unforgeable when used alone, but become totally insecure when being plugged into the combiner. The first MAC H^*_0 behaves like H_0 but also leaks the left half k_l of

the secret key, i.e., $\mathsf{H}_0^*(k, M) = k_l||\mathsf{H}_0(k, M)$. The second function is defined analogously, but outputs the right half of the key: $\mathsf{H}_1^*(k, M) = k_r||\mathsf{H}_1(k, M)$. Even though each tag is now accompanied with a part of the key, it remains hard to create a forgery. When we use now both functions $\mathsf{H}_0^*, \mathsf{H}_1^*$ as in the SSL-combiner[3] we obtain: $\mathsf{H}_0^*(k, M) \mid\mid \mathsf{H}_1^*(k, M) = k_l||\tau_0||k_r||\tau_1$ which allows to easily reconstruct the entire secret key and subsequently forge tags for any message.

Improvements. In order to change the SSL-construction such that it becomes MAC-robust, the key should be split among both underlying hash functions. The concatenation of $\mathsf{HMAC}^*_{H_0}, \mathsf{HMAC}^*_{H_1}$ invoked with independent keys then clearly gives a secure MAC, if at least one of the underlying functions is unforgeable. As SSL deviates from the standard HMAC approach to build its MAC algorithms, the results from Section 5.4.1 do not apply for HMAC^*. However, Dodis and Puniya scrutinized in [DP08] the minimal assumptions of a compression function such that the corresponding iterated hash function (with various keying approaches) yields a secure MAC. They showed that a MAC based on the Merkle-Damgård construction with the key appended to the message, requires the compression function to be collision-resistant and unforgeable when it gets keyed by the input data. It is also claimed that HMAC with an appended key requires the same assumptions as the plain MD approach. Thus, the outer hash application in $\mathsf{HMAC}^*_{H_b}$ does not contribute to the security of the MAC, in the sense that it relaxes the underlying assumption and therefore can be omitted. Applying both discussed modifications we obtain the following construction:

$$\mathsf{Comb}^{H_0,H_1}_{\mathsf{SSL-mac}^*}(k_0||k_1, M) = H_0(M||k_0) \mid\mid H_1(M||k_1)$$

If it is desirable to save on communication, one can use the xor of the H_0, H_1 outputs instead of the concatenation, while retaining the same security guarantees: In both cases the proposed combiner is a robust MAC if at least one compression function is *simultaneously* collision-resistant and unforgeable. Note that this is a stronger assumption than for the TLS combiner, where both properties can be possessed by possibly different functions.

[3] Invoking the combiner directly on H_b^* instead of $\mathsf{HMAC}^*_{\mathsf{H}_b^*}$ still proves our statement as the HMAC transform can inherit the behavior H^*. We omit the additional level for the sake of simplicity.

Bibliography

[AB81] C. A. Asmuth and G. R. Blakley. *An Efficient Algorithm for Constructing a Cryptosystem which is Harder to Break than Two Other Cryptosystems. Computers and Mathematics with Applications*, 7:447–450, 1981.

[AB99] Jee Hea An and Mihir Bellare. *Constructing VIL-MACsfrom FIL-MACs: Message Authentication under Weakened Assumptions.* Advances in Cryptology — Crypto 1999, Volume 1666 of LNCS, pages 252–269. Springer-Verlag, 1999.

[ANPS07] Elena Andreeva, Gregory Neven, Bart Preneel, and Thomas Shrimpton. *Seven-Property-Preserving Iterated Hashing: ROX.* Advances in Cryptology — Asiacrypt 2007, Volume 4833 of LNCS, pages 130–146. Springer-Verlag, 2007.

[BB06] Dan Boneh and Xavier Boyen. *On the Impossibility of Efficiently Combining Collision Resistant Hash Functions.* Advances in Cryptology — Crypto 2006, Volume 4117 of LNCS, pages 570–583. Springer-Verlag, 2006.

[BCK96a] Mihir Bellare, Ran Canetti, and Hugo Krawczyk. *Keying Hash Functions for Message Authentication.* Advances in Cryptology — Crypto 1996, Volume 96 of LNCS, pages 1–15. Springer-Verlag, 1996.

[BCK96b] Mihir Bellare, Ran Canetti, and Hugo Krawczyk. *Pseudorandom Functions Revisited: The Cascade Construction and Its Concrete Security.* Proceedings of the Annual Symposium on Foundations of Computer Science (FOCS) 1996, pages 514–523. IEEE Computer Society Press, 1996.

[Bel06] Mihir Bellare. *New Proofs for NMAC and HMAC: Security without Collision-Resistance.* Advances in Cryptology — Crypto 2006, Volume 4117 of LNCS, pages 602–619. Springer-Verlag, 2006.

[BF05] Alexandra Boldyreva and Marc Fischlin. *Analysis of Random Oracle Instantiation Scenarios for OAEP and Other Practical Schemes.* Advances in Cryptology — Crypto 2005, Volume 3621 of LNCS, pages 412–429. Springer-Verlag, 2005.

[BF06] Alexandra Boldyreva and Marc Fischlin. *On the Security of OAEP.* Advances in Cryptology — Asiacrypt 2006, Volume 4284 of LNCS, pages 210–225. Springer-Verlag, 2006.

[BR93] Mihir Bellare and Phillip Rogaway. *Random Oracles are Practical: A Paradigm for Designing Efficient Protocols.* Proceedings of the Annual Conference on Computer and Communications Security (CCS). ACM Press, 1993.

[BR94] Mihir Bellare and Phillip Rogaway. *Optimal Asymmetric Encryption — How to Encrypt with RSA.* Advances in Cryptology — Eurocrypt 1994, Volume 950 of LNCS, pages 92–111. Springer-Verlag, 1994.

[BR96] Mihir Bellare and Phillip Rogaway. *The exact security of digital signatures — How to sign with RSA and Rabin.* Advances in Cryptology — Eurocrypt 1996, Volume 1070 of LNCS, pages 399–416. Springer-Verlag, 1996.

[BR97] Mihir Bellare and Phillip Rogaway. *Collision-Resistant Hashing: Towards Making UOWHFs Practical.* Advances in Cryptology — Crypto 1997, Volume 1294 of LNCS, pages 470–484. Springer-Verlag, 1997.

[BR06a] Mihir Bellare and Thomas Ristenpart. *Multi-Property Preserving Hash Domain Extensions and the EMD Transform.* Advances in Cryptology — Asiacrypt 2006, Volume 4284 of LNCS, pages 299–314. Springer-Verlag, 2006.

[BR06b] Mihir Bellare and Phillip Rogaway. *The Security of Triple Encryption and a Framework for Code-Based Game-Playing Proofs.* Advances in Cryptology — Eurocrypt 2006, Volume 4004 of LNCS, pages 409–426. Springer-Verlag, 2006.

[BR07] Mihir Bellare and Thomas Ristenpart. *Hash Functions in the Dedicated-Key Setting: Design Choices and MPP Transforms.* International Colloquium on Automata, Languages, and Progamming (ICALP) 2007, Volume 4596 of LNCS, pages 399–410. Springer-Verlag, 2007.

[CDMP05] Jean-Sebastien Coron, Yevgeniy Dodis, Cecile Malinaud, and Prashant Puniya. *Merkle-Damgård revisited: How to construct a*

hash function. Advances in Cryptology — Crypto 2005, Volume 3621 of LNCS, pages 430–448. Springer-Verlag, 2005.

[CR08] Christophe De Cannière and Christian Rechberger. *Preimages for Reduced SHA-0 and SHA-1*. Advances in Cryptology — Crypto 2008, Volume 5157 of LNCS, pages 179–202. Springer-Verlag, 2008.

[CRS+07] Ran Canetti, Ronald L. Rivest, Madhu Sudan, Luca Trevisan, Salil P. Vadhan, and Hoeteck Wee. *Amplifying Collision Resistance: A Complexity-Theoretic Treatment*. Advances in Cryptology — Crypto 2007, Volume 4622 of LNCS, pages 264–283. Springer-Verlag, 2007.

[Dam89] Ivan Damgård. *A Design Principle for Hash Functions*. Advances in Cryptology — Crypto 1989, Volume 435 of LNCS, pages 416–427. Springer-Verlag, 1989.

[DK05] Yevgeniy Dodis and Jonathan Katz. *Chosen-Ciphertext Security of Multiple Encryption*. Theory of Cryptography Conference (TCC) 2005, Volume 3378 of LNCS, pages 188–209. Springer-Verlag, 2005.

[DP08] Yevgeniy Dodis and Prashant Puniya. *Getting the Best Out of Existing Hash Functions; or What if We Are Stuck with SHA?* International Conference on Applied Cryptography and Network Security (ACNS) 2008, Volume 5037 of LNCS, pages 156–173. Springer-Verlag, 2008.

[EG85] Shimon Even and Oded Goldreich. *On the Power of Cascade Ciphers*. *ACM Transactions on Computer Systems*, 3(2):108–116, 1985.

[Fis08] Marc Fischlin. *Security of NMAC and HMAC Based on Non-malleability*. Topics in Cryptology — Cryptographer's Track, RSA Conference (CT-RSA) 2008, Volume 4964 of LNCS, pages 138–154. Springer-Verlag, 2008.

[FL07] Marc Fischlin and Anja Lehmann. *Security-Amplifying Combiners for Hash Functions*. Advances in Cryptology — Crypto 2007, Volume 4622 of LNCS, pages 224–243. Springer-Verlag, 2007.

[FL08] Marc Fischlin and Anja Lehmann. *Multi-Property Preserving Combiners for Hash Functions*. Theory of Cryptography Conference (TCC) 2008, Volume 4948 of LNCS, pages 375–392. Springer-Verlag, 2008.

[FL10] Marc Fischlin and Anja Lehmann. *Delayed-Key Message Authentication for Streams.* Theory of Cryptography Conference (TCC) 2010, Volume 5978 of LNCS, pages 288–305. Springer-Verlag, 2010.

[FLP08] Marc Fischlin, Anja Lehmann, and Krzysztof Pietrzak. *Robust Multi-property Combiners for Hash Functions Revisited.* International Colloquium on Automata, Languages, and Progamming (ICALP) 2008, Volume 5126 of LNCS, pages 655–666. Springer-Verlag, 2008.

[FLS+09] Niels Ferguson, Stefan Lucks, Bruce Schneier, Doug Whiting, Mihir Bellare, Tadayoshi Kohno, Jon Callas, and Jesse Walker. *The Skein Hash Function Family.* Submission to NIST (Round 2), 2009.

[FLW10] Marc Fischlin, Anja Lehmann, and Daniel Wagner. *Hash Function Combiners in TLS and SSL.* Topics in Cryptology — Cryptographer's Track, RSA Conference (CT-RSA) 2010, Volume 5985 of LNCS, pages 268–283. Springer-Verlag, 2010.

[GIL+90] Oded Goldreich, Russell Impagliazzo, Leonid Levin, Ramarathnam Venkatesan, and David Zuckerman. *Security Preserving Amplification of Hardness.* Proceedings of the Annual Symposium on Foundations of Computer Science (FOCS) 1990, pages 318–326. IEEE Computer Society Press, 1990.

[GKK+09] Danilo Gligoroski, Vlastimil Klima, Svein Johan Knapskog, Mohamed El-Haded, Jørn Amundsen, and Stig Frode Mjølsnes. *Cryptographic Hash Function BLUE MIDNIGHT WISH.* Submission to NIST (Round 2), 2009.

[GKM+09] Praveen Gauravaram, Lars R. Knudsen, Krystian Matusiewicz, Florian Mendel, Christian Rechberger, Martin Schläffer, and Søren S. Thomsen. *Grøstl – a SHA-3 candidate.* Submission to NIST (Round 2), 2009.

[GMP+08] Sebastian Gajek, Mark Manulis, Olivier Pereira, Ahmad-Reza Sadeghi, and Jörg Schwenk. *Universally Composable Security Analysis of TLS.* Provable Security, Second International Conference (ProvSec) 2008, Volume 5324 of LNCS, pages 313–327. Springer-Verlag, 2008.

[Her05] Amir Herzberg. *On Tolerant Cryptographic Constructions.* Topics in Cryptology — Cryptographer's Track, RSA Conference (CT-RSA) 2005, Volume 3376 of LNCS, pages 172–190. Springer-Verlag, 2005.

[Her09] Amir Herzberg. *Folklore, Practice and Theory of Robust Combiners. Journal of Computer Security*, 17(2):159–189, 2009.

[HIKN08] Danny Harnik, Yuval Ishai, Eyal Kushilevitz, and Jesper Buus Nielsen. *OT-Combiners via Secure Computation.* Theory of Cryptography Conference (TCC) 2008, Volume 4948 of LNCS, pages 393–411. Springer-Verlag, 2008.

[Hir04] Shoichi Hirose. *A Note on the Strength of Weak Collision-Resistance. IEICE Transactions on fundamentals of electronics communications and computer sciences*, 87(5):1092–1097, 2004.

[HKN+05] Danny Harnik, Joe Kilian, Moni Naor, Omer Reingold, and Alon Rosen. *On Robust Combiners for Oblivious Transfer and other Primitives.* Advances in Cryptology — Eurocrypt 2005, Volume 3494 of LNCS, pages 96–113. Springer-Verlag, 2005.

[HS06] Jonathan Hoch and Adi Shamir. *Breaking the ICE — Finding Multicollisions in Iterated Concatenated and Expanded (ICE) Hash Functions.* Fast Software Encryption (FSE) 2006, Volume 4047 of LNCS, pages 179–194. Springer-Verlag, 2006.

[HS08] Jonathan Hoch and Adi Shamir. *On the Strength of the Concatenated Hash Combiner When All the Hash Functions Are Weak.* International Colloquium on Automata, Languages, and Progamming (ICALP) 2008, Volume 5126 of LNCS, pages 616–630. Springer-Verlag, 2008.

[Jou04] Antoine Joux. *Multicollisions in Iterated Hash Functions.* Advances in Cryptology — Crypto 2004, Volume 3152 of LNCS, pages 306–316. Springer-Verlag, 2004.

[KL08] Jonathan Katz and Andrew Y. Lindell. *Aggregate Message Authentication Codes.* Topics in Cryptology — Cryptographer's Track, RSA Conference (CT-RSA) 2008, LNCS, pages 155–169. Springer-Verlag, 2008.

[Kra08] Hugo Krawczyk. *On Extract-then-Expand Key Derivation Functions and an HMAC-based KDF.* `http://webee.technion.ac.il/ hugo/kdf/kdf.pdf`, 2008.

[KS05] Jonathan Katz and Ji Sun Shin. *Modeling Insider Attacks on Group Key-Exchange Protocols.* Proceedings of the Annual Conference on Computer and Communications Security (CCS). ACM Press, 2005.

[Lis06] Moses Liskov. *Constructing an Ideal Hash Function from Weak Ideal Compression Functions.* Selected Areas in Cryptography (SAC) 2006, Volume 4356 of LNCS, pages 358–375. Springer-Verlag, 2006.

[LMPR08] Vadim Lyubashevsky, Daniele Micciancio, Chris Peikert, and Alon Rosen. *SWIFFT: A Modest Proposal for FFT Hashing.* Fast Software Encryption (FSE) 2008, Volume 5086 of LNCS, pages 54–72. Springer-Verlag, 2008.

[LR88] Michael Luby and Charles Rackoff. *How to Construct Pseudorandom Permutations from Pseudorandom Functions. SIAM Journal on Computing*, 17(2):373–386, 1988.

[LT09] Anja Lehmann and Stefano Tessaro. *A Modular Design for Hash Functions: Towards Making the Mix-Compress-Mix Approach Practical.* Advances in Cryptology — Asiacrypt 2009, Volume 5912 of LNCS, pages 364–381. Springer-Verlag, 2009.

[LTW05] Henry Lin, Luca Trevisan, and Hoeteck Wee. *On Hardness Amplification of One-Way Functions.* Theory of Cryptography Conference (TCC) 2005, Volume 3378 of LNCS, pages 34–49. Springer-Verlag, 2005.

[Mer89] Ralph Merkle. *One Way Hash Functions and DES.* Advances in Cryptology — Crypto 1989, Volume 435 of LNCS, pages 428–446. Springer-Verlag, 1989.

[MM93] Ueli Maurer and James L. Massey. *Cascade Ciphers: The Importance of Being First. Journal of Cryptology*, 6(1):55–61, 1993.

[MP06] Remo Meier and Bartosz Przydatek. *On Robust Combiners for Private Information Retrieval and Other Primitives.* Advances in Cryptology — Crypto 2006, Volume 4117 of LNCS, pages 555–569. Springer-Verlag, 2006.

[MPW07] Remo Meier, Bartosz Przydatek, and Jürg Wullschleger. *Robuster Combiners for Oblivious Transfer.* Theory of Cryptography Conference (TCC) 2007, Volume 4392 of LNCS, pages 404–418. Springer-Verlag, 2007.

[MRH04] Ueli Maurer, Renato Renner, and Clemens Holenstein. *Indifferentiability, Impossibility Results on Reductions, and Applications to the Random Oracle Methodology.* Theory of Cryptography Conference (TCC) 2004, Volume 2951 of LNCS, pages 21–39. Springer-Verlag, 2004.

[MSW08] Paul Morrissey, Nigel Smart, and Bogdan Warinschi. *A Modular Security Analysis of the TLS Handshake Protocol*. Advances in Cryptology — Asiacrypt 2008, Volume 5350 of LNCS, pages 55–73. Springer-Verlag, 2008.

[NIS] NIST. *National Institute of Standards and Technology: SHA-3 Competition*. `http://csrc.nist.gov/groups/ST/hash/sha-3/`.

[NR98] Moni Naor and Omer Reingold. *From Unpredictability to Indistinguishability: A Simple Construction of Pseudo-Random Functions from MACs*. Advances in Cryptology — Crypto 1998, Volume 1462 of LNCS, pages 267–282. Springer-Verlag, 1998.

[NR99] Moni Naor and Omer Reingold. *On the Construction of Pseudo-random Permutations: Luby-Rackoff Revisited*. *Journal of Cryptology*, 12(1):29–66, 1999.

[NS04] Mridul Nandi and Douglas R. Stinson. *Multicollision Attacks on a Class of Hash Functions*. Number 2004/330 in Cryptology eprint archive. `eprint.iacr.org`, 2004.

[NY89] Moni Naor and Moti Yung. *Universal One-Way Hash Functions and Their Cryptographic Applications*. Proceedings of the Annual Symposium on the Theory of Computing (STOC) 1989, pages 33–43. ACM Press, 1989.

[Pie07] Krzysztof Pietrzak. *Non-Trivial Black-Box Combiners for Collision-Resistant Hash-Functions don't Exist*. Advances in Cryptology — Eurocrypt 2007, Volume 4515 of LNCS, pages 23–33. Springer-Verlag, 2007.

[Pie08] Krzysztof Pietrzak. *Compression from Collisions, or why CRHF Combiners have a Long Output*. Advances in Cryptology — Crypto 2008, Volume 5157 of LNCS, pages 413–432. Springer-Verlag, 2008.

[PW08] Bartosz Przydatek and Jürg Wullschleger. *Error-Tolerant Combiners for Oblivious Primitives*. International Colloquium on Automata, Languages, and Progamming (ICALP) 2008, Volume 5126 of LNCS, pages 461–472. Springer-Verlag, 2008.

[Res01] Eric Rescorla. *SSL and TLS - Designing and Building Secure Systems*. Addison Wesley, 2001.

[Rog06] Phillip Rogaway. *Formalizing Human Ignorance*. Vietcrypt 2006, Volume 4341 of LNCS, pages 211–228. Springer-Verlag, 2006.

[RS04] Phillip Rogaway and Thomas Shrimpton. *Cryptographic Hash-Function Basics: Definitions, Implications, and Separations for Preimage Resistance, Second-Preimage Resistance, and Collision Resistance.* Fast Software Encryption (FSE) 2004, Volume 3017 of LNCS, pages 371–388. Springer-Verlag, 2004.

[SSA+09] Marc Stevens, Alexander Sotirov, Jacob Appelbaum, Arjen Lenstra, David Molnar, Dag Arne Osvik, , and Benne de Weger. *Short Chosen-Prefix Collisions for MD5 and the Creation of a Rogue CA Certificate.* Advances in Cryptology — Crypto 2009, Volume 5677 of LNCS, pages 55–73. Springer-Verlag, 2009.

[SSL94] *The SSL Protocol (Internet Draft).* Technical report, K.E.B. Hickman, 1994.

[TLS99] *The TLS Protocol Version 1.0.* Technical Report RFC 2246, T. Dierks, and C. Allen, 1999.

[TLS06] *The TLS Protocol Version 1.1.* Technical Report (TLS 1.1) RFC 4346, T. Dierks, and C. Allen, 2006.

[WLF+05] Xiaoyun Wang, Xuejia Lai, Dengguo Feng, Hui Chen, and Xiuyuan Yu. *Cryptanalysis of the Hash Functions MD4 and RIPEMD.* Advances in Cryptology — Eurocrypt 2005, Volume 3494 of LNCS, pages 1–18. Springer-Verlag, 2005.

[WY05] Xiaoyun Wang and Hongbo Yu. *How to Break MD5 and other Hash Functions.* Advances in Cryptology — Eurocrypt 2005, Volume 3494 of LNCS, pages 19–35. Springer-Verlag, 2005.

[WYY05] Xiaoyun Wang, Yiqun Lisa Yin, and Hongbo Yu. *Finding Collisions in the Full SHA-1.* Advances in Cryptology — Crypto 2005, Volume 3621 of LNCS, pages 17–36. Springer-Verlag, 2005.

[Yao82] Andrew Yao. *Theory and Applications of Trapdoor Functions.* Proceedings of the Annual Symposium on Foundations of Computer Science (FOCS) 1982. IEEE Computer Society Press, 1982.

[YW07] Hongbo Yu and Xiaoyun Wang. *MultiCollision Attack on the Compression Functions of MD4 and 3-Pass HAVAL.* 10th International Conference on Information Security and Cryptology (ICISC) 2007, Volume 4817 of LNCS, pages 206–226. Springer-Verlag, 2007.