A Critical Look at Cryptographic Hash Function Literature

Scott Contini, Ron Steinfeld, Josef Pieprzyk, and Krystian Matusiewicz

Centre for Advanced Computing, Algorithms and Cryptography,
Department of Computing, Macquarie University

Abstract. The cryptographic hash function literature has numerous hash function definitions and hash function requirements, and many of them disagree. This survey talks about the various definitions, and takes steps towards cleaning up the literature by explaining how the field has evolved and accurately depicting the research aims people have today.

1 Introduction

The literature on cryptographic hash functions abounds with numerous different definitions and requirements. There is no universal agreement on what a cryptographic hash function is, or what it is supposed to achieve. The purpose of this terminology survey is to call the research community together to agree upon common definitions and requirements so that we can move forward with a clear set of goals.

As an illustration of the problems, we note that most researchers agree that a cryptographic hash function should compress data, in particular, it should map very large domains to fixed size outputs. However, Salsa20 [9] does not compress, even though it is labelled as a hash function and is used for cryptographic purposes. Furthermore, excluding such non-compressing hash functions, we note that the research community attempts to bifurcate hash functions into ordinary hash functions (meaning a single, fixed function) and hash function families, though this distinction is often blurred. Hash function families were introduced by Damgård [19] in order to make the security requirements very precise in the complexity theory model. He specifically looked at collision resistant[1] hash function families, where one can aim for an algorithm such that no polynomially bounded (in time and size) circuit can find collisions. Nowadays, some people define collision resistant hash functions to be collision resistant [41, §9.2] while others define them to be both collision resistant and preimage resistant [46, Def. 2.2]. Despite the fact that Damgård's definition [19] is usually cited, the current definitions do not seem to make a distinction between ordinary hash functions and hash function families. Apparently these definitions are supposed to cover both.

Within the literature of ordinary hash functions, we typically find security requirements such as the following:

[1] He actually used the term *collision free*, but this is no longer in use.

– Preimage resistance: Given y, it must be computationally infeasible to find x such that $h(x) = y$.
– Second preimage resistance: Given y and x_1 such that $h(x_1) = y$, it must be computationally infeasible to find $x_2 \neq x_1$ such that $h(x_2) = y$.
– Collision resistance: It must be computationally infeasible to find any x_1 and x_2 such that $h(x_1) = h(x_2)$.

These definitions are very informal, and attempts at formalizing them have had differing consequences. For instance, it is often stated that collision resistance implies second preimage resistance, but according to some interpretations of these requirements (such as in [41]), the claim is not true (details given later). Moreover, we note that these ordinary hash functions are used in cryptography in ways that require more complex properties than the three stated above. For example, the standardized HMAC [4] requires that the keyed compression function of the hash function is a pseudo random function family (PRF) [2], and the theory behind the standardized RSA-OAEP [6] and RSA-PSS [7] assumes the hash functions are random oracles. Since it is well known that random oracles do not exist [15], usually people settle for some type of emulation of random oracle goal. For instance, in NIST's recent draft call for a new hash standard, one of the security evaluation criterion is

> "The extent to which the algorithm output is indistinguishable from a random oracle."

What does this mean? We know that any ordinary hash function can be distinguished from a random oracle in a single hash function query since random oracles are secret objects whereas hash functions are entirely public. Since we have no definition of what random oracle emulation is, we have no objective way of telling how well candidate hash functions for NIST's new standard are behaving.

The main body of this paper elaborates on these and other problems with the cryptographic hash function literature. Although we do not have all the answers, we hope that it is a step forward in cleaning up the literature. However, we would like to emphasize that our criticisms of certain definitions and terminology should not be interpreted as a criticism of the people that said them. Cryptographic hash functions has been an evolving subject since its origin. Many people had good ideas as a step forward, but the ideas have now become obsolete as further research has been developed. One of the goals of this paper is to point out ideas which are obsolete so that new research can focus with less ambiguity on the current understanding of the way hash functions are intended to be used.

Before we begin, we remark that [48] deals with related issues of hash function terminology. In fact, we will reference this paper many times, since our goals are overlapping with theirs. However, our focus is more broad and less technical, and is more of a down-to-earth survey rather than original research.

2 A Brief History of Cryptographic Hashing

Here we outline a brief history on how we arrived to where we are today in hashing, according to the history we were able to puzzle together. We do not

attempt to cover all hashing issues, only the ones that are most relevant to our discussions. Note that we are mainly interested in unkeyed hash functions, but we will talk about some applications where these functions are keyed one way or another.

2.1 Definitions of Hash Functions

The use of one-way functions (which may be non-compressing) for authentication was noted in Diffie and Hellman's New Directions in Cryptography [26], although they did not use the terminology "hash function". They claimed to need preimage resistance (i.e. one-wayness) and second preimage resistance. The term "hash function" to imply a compressing one-way function seems to come from Merkle [42] or Rabin [47]. Both were specifically concerned with message authentication, though not necessarily through public key cryptosystems. At this time, neither considered collision resistance, but the importance of the concept appeared soon afterward [55]. As far as we are aware, all subsequent definitions of hash functions insisted that the functions were compressing[2] so for brevity, we will not mention it any more.

Around the same time, the RSA public key cryptosystem was invented and soon after attempts at attacking the system were published. Most notably, Davida [21] demonstrated how to break the RSA signature scheme by a chosen message attack, and improved attacks followed [25]. As a consequence, hash functions were suggested by a few researchers as a tool for preventing such attacks as well as speeding up signatures. In [22], Denning states that the hash function should destroy homomorphic structure in the underlying public key cryptosystem, it should be one-way, and it should be second preimage resistant. Note that collision resistance is not mentioned. In [54], it was specifically stated that collision resistance is a requirement for hashing in digital signature schemes, including RSA. However, ironically, he defines *one-way hash functions* to be collision-resistant only – the preimage resistance requirement is not stated. Merkle used similar definitions in 1989 [43] where he defined *weak one-way hash functions* (second preimage resistant) and *strong one-way hash functions* (collision resistant) without explicitly stating the one-way requirement (thus one could include that they are misnomers). One might assume that the researchers had a reason to believe that collision resistance implies preimage resistance, but the implication holds only under certain conditions [19, 49, 51]. Also in 1989, Damgård formally defined *families of collision free hash functions* [19], which are collision resistant in the complexity theory model. However, he remarks that collision resistance does not necessarily imply preimage resistance. Note that both Denning and Damgård explicitly accepted the *heuristic* use of hash functions to securing public key systems since at that time, there was a lack of systems with a *security proof*. Later, in Preneel's PhD Thesis, he notes that there are many different definitions of hash functions [46], and attempts to clean up the literature. Citing Merkle's work as well as Rabin's, Preneel defines a *one-way hash*

[2] Although [9] is claimed to be a hash function, the author does not seem to give a definition of hash function.

function to be preimage resistant and second preimage resistant. Additionally, citing Damgård's work and Merkle's work, he defines *collision resistant hash function* to be preimage resistant, second preimage resistant, and collision resistant. The Handbook of Applied cryptography [41] accepts Preneel's definition of one-way hash function (this may be the first two definitions that agree), yet they drop the preimage resistance requirement from the definition of collision resistant hash function. A cursory glance might lead one to think that this definition of collision resistant hash is the same as Damgård's families of collision free hash functions, but they are not. One of the major differences is that Damgård's definitions involve *families* of functions, which makes formalizing the definitions in the complexity theory model possible. The other definitions are informal, and attempts at formalizing them have given different results [49]. We will talk more about this later. A summary of this paragraph is in Table 1.

source	function name	pre. resist.	2 pre. resist.	coll. resist.	fam. funcs.
Winternitz [54]	o.w. hash			X	
Merkle [43]	weak o.w. hash		X		
Merkle [43]	strong o.w. hash			X	
Damgård [19]	coll. free hash			X	X
Preneel [46]	o.w. hash	X	X		
Preneel [46]	coll. resist. hash	X	X	X	
HAC [41]	o.w. hash	X	X		
HAC [41]	coll. resist. hash		X	X	

Table 1. Different definitions of hash functions and their requirements, according to various sources in cryptographic hash function literature.

Nowadays, when a new design is created, people typically refer to it as simply a "hash function" with the intended three properties of collision resistance, second preimage resistance, and preimage resistance. In some cases, people say informally that the outputs are also intended to look random. There are also some designs which are called "collision-resistant hash functions" which, in addition to collision resistance, may or may not intend to have preimage resistance and may or may not be a family of functions (depending upon the definition the inventors use). In general, the more theoretical community does not require preimage resistance as part of the requirement for collision resistant hashing (for example, the Merkle-Damgård construct turns a collision resistant compression function into a collision resistant hash function, but nothing is proved about preimage resistance). On the other hand, the cryptanalysis literature often looks at these designs from the viewpoint of an "ideal hash" (more details will be given below), which should have all of the above properties (except possibly the family of functions property) and more. In summary, we have practical designers, theoreticians, and cryptanalysts all speaking different languages, and even within a single group they do not necessarily speak the same.

2.2 Random Oracles

As the importance of hash functions became clear, the theoreticians tried to formalize concepts such as destroying the homomorphic structure in the underlying public key cryptosystem. The way forward seemed to be viewing the hash function as a randomly chosen function from the space of all functions with the same domain and range. From here came the closely related concepts of random oracles [28] and pseudo random function families (PRFs) [33]. The random oracle in particular was aimed at modelling the *ideal hash function* – one that behaves like a truly random function. The concept was used by Bellare and Rogaway [5] as a first step between bridging the theory and practice of public key cryptography. They developed the first practical versions of RSA encryption and signatures that had some notion of a proof of security – a concept that was long sought after, since heuristic defenses against attacks on RSA were very ad hoc. Systems which have security proofs (i.e. security reductions) involving random oracles are said to be *proven secure in the random oracle model*. Some interpret these proofs as an argument that the design is secure against any adversary who treats the hash function as a black box. The ideas of Bellare and Rogaway came at the right time since a few years later Bleichenbacher developed a clever attack on the RSA PKCS #1 encryption standard [10]. Consequently, the standard was able to be quickly updated to use the ideas of Bellare and Rogaway, namely RSA-OAEP and RSA-PSS, which have more of a theoretical foundation than the previous version.

Around the same time as Bleichenbacher's attack, a new way of performing public key encryption was developed by Cramer and Shoup [18] that is provably secure without the use of random oracles. The only thing they require is a collision resistant hash function, where in this case the exact requirement is collision resistance only (i.e. not preimage resistance). The Cramer and Shoup designs are practical, though not as efficient as RSA-OAEP or RSA-PSS. The theoreticians considered this a great breakthrough – a practical and provably secure design that relied only on plausible number theory assumptions. Adding to its significance is research casting doubt on the random oracle model, starting with the work of Canetti, Goldreich, and Halevi [15] who showed that there are protocols that are secure in the random oracle model but immediately become insecure as soon as any concrete hash function is substituted for the random oracle. Some have criticized this work since the examples are quite artificial, but newer related results are closer to the real world [3].

Today there is a research trend of avoiding random oracles, though a few researchers are resisting [38]. Regardless of whether one agrees or disagrees with random oracle proofs, there are widely used standards whose security foundation is based upon them. We do know that random oracles can never be realized in practice since these functions are private whereas hash functions are entirely public objects. Despite this, right now much of the hash function community is aiming for a function that emulates a random oracle. Unfortunately, nobody has yet said what this means. Although the topic has been looked at by theoreticians (example: [14]), it remains unresolved today. We remark that talking

about distinguishing a hash function from a random oracle seems to be the wrong concept – one can always distinguish in a single computation since hash functions are computable whereas random oracles are not. For instance, given an input/output pair (x, y), we can easily check whether $\text{SHA1}(x) = y$ just by running it through a computer program that implements SHA1. On the other hand, the probability that some random oracle $\mathcal{O}$ also has $\mathcal{O}(x) = y$ is 2^{-160}. So if the SHA1 computation matches, there is no reasonable doubt that it came from SHA1 instead of a random oracle.

2.3 Other Requirements for Hash Functions

Nowadays, hash functions are used in many different ways in practice. One of the most common ways is with Message Authentication Codes (MACs) which are a means that two users with a shared secret key can authenticate between each other. The widely used HMAC standard comes with a security proof [4]– it is secure provided that the underlying keyed compression function of the hash is a PRF. Hash functions are also widely used for pseudo random number generation. In one instance, security can be proved provided that the hash function behaves as a PRF where the secret involves adding a key to the message space [23]. See Table 2 later in our paper.

One of the reasons hash functions have taken on such a diverse role is because they have not been subject to export regulations, contrary to other cryptographic primitives. Additionally, they offered speed advantages and could be used without a license (unlike the IDEA block cipher). Today, export regulations are less strict and there are plenty of efficient alternative public domain cryptographic primitives available, so the mentioned benefits have disappeared. If we are going to continue to apply hashing to the various scenarios that we are using today, then we require a single function that satisfies all of these security requirements. An alternative is to cut back in the way we are using these functions.

2.4 Summary

Despite Preneel's efforts at cleaning up the hash literature, we still have multiple definitions and no clear vision of what exactly the requirements are. There seems to a few reasons for this. First, we need to carefully distinguish between ordinary hash functions and hash function families, and what they are aiming to achieve. Second, the names of the objects we are defining should accurately reflect the security properties they are intended to have. Third, we need precise definitions on exactly what properties we require.

3 Ordinary Hash Functions Versus Hash Function Families

Hash function families were introduced by Damgård [19] in order to make a formal definition of collision resistance in the complexity theory model. Sometimes

the word "ensembles" is used instead of "families." Damgård's functions had a single specific purpose rather than trying to solve all problems at once, contrary to the current approach used for ordinary hash functions.

Damgård required that there is no polynomially bounded circuit that could compute a collision with non-negligible probability, where the probability is evaluated over all members of the family. The problem with ordinary hash functions is that collision resistance cannot be defined in this way, since there is only a single member of the family. In other words, there is always a polynomially bounded circuit which will compute the collision for an ordinary hash function – the difficulty is finding that circuit, which depends upon human ignorance [48] rather than complexity theory. This point must be emphasized. When people design ordinary hash functions that are intended to be collision resistant, what they mean is that they have a design for which they believe it is tricky enough that nobody will be able to come up with a clever way of finding collisions – *not* that such algorithms *don't exist*.[3]

Thus, we see that there are different definitions used for ordinary cryptographic hash functions than there are for hash function families and that there are different research goals. With this is mind, it is a mistake to try to clump these together and use a single definition to represent both of them. We therefore shall treat the two topics separately. As we will justify below, ordinary hash functions meet engineering requirements but are lacking in a theoretical foundation, while hash function families have the theoretical foundation but are lacking in practical solutions. These gaps need to be closed.

3.1 Ordinary Hash Functions

By "ordinary hash functions," we mean the designs like MD5 and SHA-1 that are used today. We have not yet given a precise definition of what these are – we shall return to that later. But generally, we are speaking of single functions (not families) that compress and have additional goals such as collision resistance, second preimage resistance, and preimage resistance. We are essentially clumping together the so-called custom designed hash functions with hash functions based upon block ciphers.

Those who have done research in ordinary hash functions have been more concerned with engineering requirements than scientific definitions. They aim for a single solution that has numerous security properties, so that it can substituted anywhere without having to think about it. However, this view has come at a cost, since it is not clear that such goals are achieveable. Note that although the stated goals of such functions are usually only collision resistance, second preimage resistance, and preimage resistance, in fact the functions are being used in ways that require additional properties.

[3] Rogaway [48] also considers a hybrid variant, where unkeyed hash functions have a secuity parameter n representing the hash length. Most ordinary hash functions are currently not built in this way, but it may be advisable to do so in the future. Another option is custom designed families of functions having a security parameter.

Definitions and Implications of Security Properties Despite the fact that collision resistance is a central requirement of hash function research, nearly all analyses involving this concept are in the family of function setting. In practice we are not at all using that setting. Only recently has the model of human ignorance been proposed by Rogaway [48] in order to formalize collision resistance for ordinary hash functions. The next step is to develop results analagous to [49] in either the *code-constructive* or the *blackbox-constructive* form (see [48]). Note that the relation between aPre and aSec [49] carries directly over to this setting. Presumably, collision resistance carries over as well.

It is a convenient time to emphasize the importance of formalizing these definitions. The definition of second preimage resistance from the Handbook of Applied Cryptography [41] is:

> 2nd preimage resistance – it is computationally infeasible to find any second input which has the same output as any specified input, i.e., given x to find a 2nd preimage $x' \neq x$ such that $h(x) = h(x')$.

In [49], it is asked whether it is really meant that the specified x can be *any* domain point. In fact, this definition is better than most informal definitions in the sense that it does tell us how x is chosen. But, returning to Rogaway and Shrimpton's question, do we really mean *any* or is it sufficient to require *all except a negligible portion*? Or does it not matter at all?

In fact it does matter when one considers the relation between collision resistance and second preimage resistance. Informally, one would argue that the former trivially implies the latter, since if one can compute second preimages, then they have found a collision. But this argument is actually not valid for the Handbook of Applied Cryptography definition since it may be possible that there is a subset of negligible size where one can trivially compute second preimages, though in practice it may be impossible to find that subset [24]. For instance, consider the number theoretic hash of the form $h(x) = a^x \bmod N$, where N is a product of two large primes and a is a parameter having certain required properties – details in [45]. With proper padding (ignored here for simplicity) this function is provably collision resistant, with a security reduction from integer factorization. Actually, this is as a family of functions indexed by the modulus, N, but if we fix N then it becomes an ordinary hash function and we can use the code- or blackbox-constructive formalization. Despite having a security reduction from factoring N to collision finding, one can always trivially compute second preimages for any value of x that is a multiple of $\phi(N)$ since any other multiple of x is a second preimage. Of course, these pathological messages give away a break of that hash function, making it so that it is no longer collision resistant. Obviously, if factoring is really a hard problem, then it is safe to assume that we would never encounter such messages in practice, and therefore such counterexamples are of no interest. So we really want a definition that reflects that these negligible probability events should have no real security impact.

With proper definitions and formalizations, it appears that one can indeed show that collision resistance implies second preimage resistance for ordinary

hash functions, which seemingly would simplify our list of goals for hash functions. Unfortunately, the argument only shows that finding second preimages is at least as hard as finding collisions, so it only guarantees that finding a second preimage is at least order $2^{n/2}$ effort for a hash with an $n-$bit output (assuming the best collision attack is a generic one). Those who do research in ordinary hash functions often want that second preimages should take 2^n effort, though it usually isn't explicitly stated. In that case, the security relation between collision resistance and second preimage resistance does not matter to them. A further setback is [37], which showed that nearly all ordinary hash function constructions that are being seriously considered today cannot achieve 2^n second preimage resistance. They suggest that perhaps we should not expect more than $2^{n/2}$ security for any aspect of an $n-$bit hash. On the other hand, new constructions are being considered which *might* resist the Kelsey and Schneier attack [39]. In any event, the hash function community has to come to a common agreement on their security goals, and to be very precise about those requirements.

Furthermore, for some reason the hash community has left out the notions of PRF and random oracle emulation when considering the implications of various security notions, despite the fact that hash functions are being used for this purpose. In regard to PRF, the current folklore indicates that we could build one by taking a hash function and putting a secret element into one of the inputs (IV or message). But there is no theoretical justification for this idea. In regard to random oracle emulation, someone should formally define what random oracle emulation means, and that definition presumably would imply the other security requirements we have so far listed. Then researchers would have the single objective goal of designing a hash function that "looks like" a random oracle, rather than a list of many smaller goals that do not accurately depict all the necessary security requirements.

In the absence of a satisfactory definition for random oracle emulation, we face a dilemma for evaluating proposals for new hash function standards. On the one hand, we would like a new function to support existing random oracle applications. On the other hand, we do not have a simple well-defined security requirement on the hash function to support all such applications. In that case, we could deal with this problem by identifying the main important applications of a new hash function standard, and then, for each such application, specify in a well-defined way: (1) the precise details of the application and how it makes use of the hash function (with the hash function treated as a black box), and (2) the security requirement on the application incorporating the hash function. For example, consider the OAEP-RSA public-key encryption application [50], which makes use of two "Mask Generation Functions" (treated as random oracles in the security analysis [6, 32]). The precise details of the application and how it makes use of the hash functions (in particular how the "Mask Generation Functions" are constructed from a given hash function h) are specified in the PKCS standard [50] (see in particular the definition of MGF1 in Appendix B.2.1 in [50]), and the precise security requirement on the OAEP-RSA application is the well-known IND-CCA2 security requirement for public-key encryption

schemes [32]. The important advantage of this approach is that it allows a hash standard to support the most important random oracle applications of hash functions with *well-defined* security requirements for the hash function, allowing objective evaluation of proposals. The disadvantages of this approach are that these security requirements are complex (involving details of the applications), and that the approach introduces a new security requirement for each random oracle application that is to be supported by the hash standard. However, at present we do not see any better way available to support all the uses of random oracles with well-defined security requirements.

An open research problem is find simpler security requirements on the hash functions, which are sufficient at least for the most popular current applications of random oracles. Some theoretical progress has been made in this direction over the last few years [14–16, 3, 11, 27, 12, 13], but more research is needed for such results to be applicable in practice (e.g. to support the full security requirements of popular applications provable in the random-oracle model).

3.2 Hash Functions Families

In contrast to ordinary hash functions, the security properties and implications of hash function families are better understood [49]. Although not explicitly stated, hash function families seem to have traditionally been aimed at designing specific solutions for specific problems rather than a one solution for all problems approach. Where they are lacking is in practical solutions, both at the hash function design level and the protocol design level, which is entirely the reason they have so far not been used in practice today.

At the hash function design level, there has been some recent progress towards making this approach practical. Examples include VSH [17] and FFT Hash [40], which both achieve provable collision resistance and are approaching practicality. A preliminary implementation of VSH is reported to be within a factor of 25 of the speed of SHA-1. While both results are encouraging, more needs ot be done. Solutions need to be developed that have smaller output sizes (especially for FFT-hash, which requires very large parameters for the security reduction to hold) and faster speeds. Moreover, there is still no provable and practical hash solutions for other security needs, such as PRF.

At the protocol level, there have been many new designs that have eliminated random oracles. The down side is that they usually do not achieve the efficiency of random oracle based approaches.

4 What to Do about all the Cryptographic Hash Function Definitions?

We have so far eluded the question of how these hash functions should be defined. Given that so many people have cluttered the literature with different definitions, it is against our judgement to offer new definitions that are attempts at overriding others. However, we do recommend rejecting certain definitions.

Generally, we would like to avoid definitions that do not accurately reflect the security properties that they are intended to have. Examples include Merkle's weak and strong one-way hash functions, Winternitz's version of a one-way hash function, and Preneel's version of a collision-resistant hash function (since the name does not specify preimage resistance also). We would also like to avoid definitions that are not mainstream, such as definitions that do not involve compression.

Many hash function designers today simply call their design a "hash function" without adjectives such as "one-way" or "collision resistant." An interpretation of this vernacular is a compressing and easy to compute function that has additional security properties. We do not oppose such a definition (although it is informal), but we do strongly recommend that researchers say *exactly* what those additional properties are for their design. In particular, if researchers are proposing a single solution to be used everywhere like the way we are using SHA-1 today, then they should include PRF and random oracle emulation in their stated security goals, and at least include a reference to how to interpet such definitions formally. Moreover, when researchers develop a hash function family, do not omit the word "family."

5 Moving Forward

We recommend that standards bodies involved in selecting new standards for cryptographic hash functions, in collaboration with the cryptographic community, should aim to specify a set of *well-defined* security requirements for cryptographic hash functions. This set of security requirements would then allow an objective assessment of the security of candidate functions submitted for standardization. Such requirements are defined by specifying an interactive *computational game* between an adversary and a challenger, and defining a condition on the outcome of the game which defines *success* of the adversary in 'winning' the game, and a quantity called the *advantage* of the adversary (which is determined by its success probability). The security requirement can then be quantified by the maximal advantage of the adversary in the game given bounds on its computational resources (run-time/program, size, number of oracle queries, etc). In the standard complexity-theory model, the maximal adversary advantage is taken over *all* adversaries with the given resource bound. An alternative approach is the human ignorance model where details can be found in [48].

The security requirements needed from cryptographic hash functions are ultimately determined by their *applications*. Therefore, as a step towards the above stated goal, we present below a list of the main current practical applications of cryptographic hash functions. For each such application, we cite known well-defined security requirements on the underlying hash function which guarantee the security of the application (if such requirements are known). We also list other requirements from the hash function (whether a function *family* is needed, whether the family key is secret or public, the function input/output domain) and the relevant references to the literature where the security requirement was defined and shown sufficient for the application. To make this survey

self-contained we also provide a definition of the relevant security requirements (via the games and adversary success conditions) in the Appendix.

We note that this list is not intended to be exhaustive but is presented to indicate the variety of requirements needed from hash functions today. Other applications are expected to add yet other new requirements. We also note that it is debatable whether all the listed applications (and corresponding security requirements) should in fact be supported by a new hash standard. For example, applications which need a PRF are usually implementable using a block cipher as the underlying cryptograpic primitive, rather than a hash function. Indeed, block ciphers are usually designed to achieve a PRF-like design goal (more precisely the closely related PRP family goal). On the other hand, it is possible that hash functions may be preferred for such applications for various (not necessarily technical) reasons.

App	**Fam?**	**Key Sec?**	**In**	**Out**	**Security**	**Ref.**
HMAC [31] (a)	N	–	$B^b \times B^c$	B^c	$h(\cdot, K)$ and $h(K, \cdot)$ are PRF families	[2], 1
Deterministic	Y	N	B^*	B^c	$H_K(\cdot)$ is CR Family	[19], 2
Message	Y	N	$B^b \times B^c$	B^c	$h_K(\cdot)$ is CR Family	[19, 20], 2
Hashing for Secure Sig (b)						
Randomized	Y	N	B^*	B^c	$H_K(\cdot)$ is TCR Family	[44], 3
Message	Y	N	B^*	B^c	$H_K(\cdot)$ is eTCR Family	[35], 4
Hashing for	N	–	$B^b \times B^c$	B^c	$h(\cdot, \cdot)$ is e-SPR	[35], 5
Secure Sig (b),(c)						
RSA-PSS [50]	N	N	B^*	B^c	"random oracle"	[7], ???
(d)	Y	N	B^*	B^c	RSA-PSS (using $H_K(\cdot)$) is EF-CMA [34]	
RSA-OAEP [50]	N	N	B^*	B^c	"random oracle"	[6, 32], ???
(e)	Y	N	B^*	B^c	RSAES-OAEP (using $H_K(\cdot)$) is IND-CCA2 [32]	
FIPS 186 PRG [29, 30]	N	–	$B^b \times B^c$	B^c	$h((\cdot + K) \bmod 2^b, IV)$ is PRF family (fixed IV)	[23], 1
DSA [29, 30]	Y	N	B^*	B^c	DSA (using $H_K(\cdot)$)	
(f)					is EF-CMA [34]	
ECDSA [1]	Y	N	B^*	B^c	ECDSA (using $H_K(\cdot)$)	
(f)					is EF-CMA [34]	
Password Hashing (g)	N	–	B^b	B^c	preimage resistant	[53],6
Commitments	Y	N	$B^b \times B^c$	B^c	$h_K(\cdot, \cdot)$ is CR Family	[36], 2

Table 2. List of hash function applications and corresponding requirements. Refer to the text for more details. Letters in parenthesis (e.g. "(a)") in the leftmost column refer to the list of remarks listed below.

We use the following conventions in Table 2. Set B denotes binary set $\{0,1\}$ and B^* denotes the set of all finite binary strings. Column 'App' lists the applications. For each application, column 'Fam?' indicates whether a function family is required (Y) or not (N). If a function family is needed, column 'Key Sec?' indicates whether the function key is secret (Y) or public (N). Columns 'In' and 'Out' indicate the required hash function input and output domains respectively (with typical values). Column 'Security' indicates the sufficient security requirement required from the hash for the application. Column 'Ref' gives a reference in the literature for relevant security analysis, followed by a security requirement number, referring to our list of security requirement definitions in the appendix; if such a well-defined security requirement on the hash is not known, we write '???'. Note that for some applications there are several alternative hash function requirements, any one of which is sufficient for the application; in such cases, we list the alternatives on separate rows.

Remarks on table entries:

(a) The security requirement stated here are for the "2-key" variant of HMAC which uses two independent keys [2]. The security proof for the standard "1-key" variant of HMAC requires an additional 'related key' pseudorandomness assumption on h, see [2].

(b) In these applications the hash function is used to hash a long message prior to signing the hash value using a secure signature scheme which accepts short fixed-length input messages (of length at least equal to the output length c of the hash function). It is assumed that the given signature scheme is *fully* secure, i.e. is existentially unforgeable under adaptive chosen message attack (EF-CMA) [34]. In other words, we assume the hash function is only used to allow signing arbitrary length messages, and not for strengthening the security of the underlying signature scheme. This excludes many "classic" signature schemes such as plain RSA and ElGamal-type signatures (which are existentially forgeable and not secure against chosen message attacks). To strengthen such 'weak' signature schemes into secure ones, the hash function requires additional properties beyond collision-resistance. In many such cases, modelling the hash function as a "random oracle" is enough (see 'PSS-RSA' table entry for example). The hash security requirements listed here are also sufficient for other 'integrity checking' applications, where the message hash value is write-protected in some other way (rather than being signed), e.g. by storing it in a public read-only memory device, and is used to verify integrity of the message by hashing and comparing with the stored write-protected hash value.

(c) The alternative in the first row requires signing both the hash value and the hash randomizer, whereas in the alternatives in the last two rows, only the hash value needs to be signed.

(d) We refer here to the RSASSA-PSS signature scheme in the PKCS standard [50]. We denote by EF-CMA the standard security requirement for signature schemes, namely Existential Unforgeability under adaptive Chosen Message Attack [34].

(e) We refer here to the RSAES-OAEP public-key encryption scheme in the PKCS standard [50]. We denote by IND-CCA2 the standard security requirement for public-key encryption schemes, namely Indistinguishability under Adaptive Chosen Ciphertext Attack [32].
(f) Some security results for *variants* of DSA and ECDSA are known in the random oracle and generic group models, see [52] for a survey.
(g) For password hashing, applying the results of [53], if maximal adversary advantage against preimage resistance of $h(\cdot) : B^b \rightarrow B^c$ is $Adv(t)$ for runtime t then maximal adversary advantage against preimage resistance of $h(\cdot)$ when applied to a a uniformly random password from a subset $D \subset B^b$ of size $|D|$ is $Adv'(t) \leq \frac{Adv(t)2^b}{|D|}$, hence to guarantee $Adv'(t) \leq 1/2^s$ (password security level of 's-bits') we need a password set D of size at least $|D| \geq (Adv(t) \cdot 2^b) \cdot 2^s$. We note that we assume here (following [53]) that no 'salting' is used.

6 Conclusion

The field of cryptographic hash functions has been evolving since its origin approximately 30 years ago, and will continue to do so for quite some time. The informality of hash function terminology has resulted in cluttered literature, lacking a clean list of goals summarizing our security requirements. Consequently, there is no objective way of evaluating the security of new hash function proposals, except designs that are very obviously broken.

This survey has emphasized the importance of formal terminology and a clear set of objectives. The hope is that researchers will take our view into consideration as a first step of trying to clean up the cryptographic hashing literature.

References

1. ANSI X9.62. Public Key Cryptography for the Financial Services Industry: The Elliptic Curve Digital Signature Algorithm (ECDSA). American National Standards Institute. American Bankers Association, 1998.
2. M. Bellare. New proofs for NMAC and HMAC: Security without collision-resistance. In *Advances in Cryptology – CRYPTO'06*, volume 4117 of *LNCS*, pages 602–619. Springer, 2006.
3. M. Bellare, A. Boldyreva, and A. Palacio. An uninstantiable random-oracle-model scheme for a hybrid-encryption problem. In C. Cachin and J. Camenisch, editors, *Advances in Cryptology – EUROCRYPT '04*, volume 3027 of *LNCS*, pages 171–188. Springer, 2004.
4. M. Bellare, R. Canetti, and H. Krawczyk. Keying hash functions for message authentication. *Lecture Notes in Computer Science*, 1109, 1996.
5. M. Bellare and P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *First ACM Conference on Computer and Communications Security*, pages 62–73, Fairfax, 1993. ACM.

6. M. Bellare and P. Rogaway. Optimal asymmetric encryption. In *Advances in Cryptology – EUROCRYPT'94*, volume 950 of *LNCS*, pages 92–111. Springer, 1995.
7. M. Bellare and P. Rogaway. The exact security of digital signatures – how to sign with RSA and Rabin. In *Advances in Cryptology – EUROCRYPT'96*, volume 1070, pages 399–416. Springer, 1996.
8. M. Bellare and P. Rogaway. Collision-Resistant hashing: Towards making UOWHFs Practical. In *CRYPTO '97*, volume 1294 of *LNCS*, pages 470–484, Berlin, 1997. Springer-Verlag.
9. D. J. Bernstein. The Salsa20 hash function. Web page, `http://cr.yp.to/salsa20.html`.
10. D. Bleichenbacher. Chosen ciphertext attacks against protocols based on rsa encryption standard pkcs #1. In *Advances in Cryptology – CRYPTO'98*, volume 1462 of *LNCS*, pages 1–12. Springer-Verlag, 1998.
11. A. Boldyreva and M. Fischlin. Analysis of random-oracle instantiation scenarios for OAEP and other practical schemes. In *Advances in Cryptology – CRYPTO '05*, volume 3621 of *LNCS*, pages 412–429. Springer-Verlag, 2005.
12. A. Boldyreva and M. Fischlin. On the security of OAEP. In *Advances in Cryptology – ASIACRYPT '06*, volume 4284 of *LNCS*, pages 210–225. Springer-Verlag, 2006.
13. D. Brown. Unprovable security of RSA-OAEP in the standard model. Cryptology ePrint Archive, Report 2006/223, June 2006. `http://eprint.iacr.org/`.
14. R. Canetti. Towards realizing random oracles: Hash functions that hide all partial information. In *CRYPTO '97*, volume 1294, pages 455–469, London, UK, 1997. Springer-Verlag.
15. R. Canetti, O. Goldreich, and S. Halevi. The random oracle methodology, revisited. *J. ACM*, 51(4):557–594, 2004.
16. R. Canetti, D. Micciancio, and O. Reingold. Perfectly one-way probabilistic hash functions. In *STOC '98*, pages 131–140. ACM Press, 1998.
17. S. Contini, A. Lenstra, and R. Steinfeld. VSH, an efficient and provable collision-resistant hash function. In S. Vaudenay, editor, *EUROCRYPT '06*, volume 4004 of *Lecture Notes in Computer Science*. Springer, 2006.
18. R. Cramer and V. Shoup. A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In *CRYPTO '98*, volume 1462 of *LNCS*, pages 13–25. Springer, 1998.
19. I. B. Damgård. Collision free hash functions and public key signature schemes. In *Advances in Cryptology – EUROCRYPT'87*, volume 304 of *LNCS*, pages 203–216. Springer, 1987.
20. I. B. Damgård. A design principle for hash functions. In G. Brassard, editor, *Advances in Cryptology - CRYPTO '89*, volume 435 of *LNCS*, pages 416–427. Springer-Verlag, 1989.
21. G. Davida. Chosen signature cryptanalysis of the RSA (MIT) public key cryptosystem. *Tech. Rep.*, TR-CS-82-2, 1982.
22. D. E. Denning. Digital signatures with RSA and other public-key cryptosystems. *Commun. ACM*, 27(4):388–392, 1984.
23. A. Desai, A. Hevia, and Y. Yin. A practice-oriented treatment of pseudorandom number generators. In *EUROCRYPT 2002*, pages 368–383. Springer, 2002.
24. Y. G. Desmedt. Private email, 2005.
25. Y. G. Desmedt and A. M. Odlyzko. A chosen text attack on the RSA cryptosystem and some discrete logarithm schemes. In *CRYPTO '85*, pages 516–522. Springer, 1985.
26. W. Diffie and M. E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, IT-22(6):644–654, 1976.

27. Y. Dodis, R. Oliveira, and K. Pietrzak. On the generic insecurity of full-domain hash. In *CRYPTO '05*, volume 3621 of *LNCS*, pages 449–466. Springer, 2005.
28. A. Fiat and A. Shamir. How to prove yourself: practical solutions to identification and signature problems. In *Proceedings on Advances in cryptology—CRYPTO '86*, pages 186–194, London, UK, 1987. Springer-Verlag.
29. FIPS PUB 186-2. Digital signature standard. National Institute of Standards and Technology, 1994.
30. FIPS PUB 186-2 (Change Notice 1). Digital signature standard. National Institute of Standards and Technology, 2001.
31. FIPS PUB 198. The keyed-hash message authentication code (HMAC). National Institute of Standards and Technology, 2002.
32. E. Fujisaki, T. Okamoto, D. Pointcheval, and J. Stern. RSA-OAEP is secure under the RSA assumption. In *Advances in Cryptology– CRYPTO'01*, volume 2139, pages 260–274. Springer, 2001.
33. O. Goldreich, S. Goldwasser, and S. Micali. On the cryptographic applications of random functions. In G. R. Blakley and D. C. Chaum, editors, *CRYPTO '84*, pages 276–288. Springer, 1985. Lecture Notes in Computer Science No. 196.
34. S. Goldwasser, S. Micali, and R. Rivest. A digital signature scheme secure against adaptively chosen message attacks. *SIAM Journal on Computing*, 17(2):281–308, 1988.
35. S. Halevi and H. Krawczyk. Strengthening digital signatures via randomized hashing. In *Advances in Cryptology - CRYPTO'06*, volume 4117 of *LNCS*, pages 41–59. Springer, 2006.
36. S. Halevi and S. Micali. Practical and provably-secure commitment schemes from collision-free hashing. In *Advances in Cryptology - CRYPTO'96*, volume 1109, pages 201–215, 1996.
37. J. Kelsey and B. Schneier. Second preimages on n-bit hash functions for much less than 2^n work. In R. Cramer, editor, *Advances in Cryptology - EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 474–490. Springer-Verlag, 2005.
38. N. Koblitz and A. J. Menezes. Another look at "provable security". *Journal of Cryptology*, 20(1):3–37, January 2007.
39. S. Lucks. Failure-friendly design principle for hash functions. In B. K. Roy, editor, *Advances in Cryptology - ASIACRYPT '05*, volume 3788 of *LNCS*, pages 474–494. Springer, 2005.
40. V. Lyubashevsky, D. Micciancio, C. Peikert, and A. Rosen. Provably secure FFT hashing. Second NIST Workshop on Hash Functions, 2006.
41. A. J. Menezes, P. C. van Oorschot, and S. A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1996.
42. R. C. Merkle. *Secrecy, Authentication, and Public Key Systems.* PhD thesis, Stanford University, 1979.
43. R. C. Merkle. One way hash functions and DES. In G. Brassard, editor, *Advances in Cryptology - CRYPTO '89*, volume 435 of *LNCS*, pages 428 – 446. Springer-Verlag, 1989.
44. M. Naor and M. Yung. Universal one-way hash functions and their cryptographic applications. In *Proceedings of the Twenty First Annual ACM Symposium on Theory of Computing*. ACM Press, 1989.
45. D. Pointcheval. The composite discrete logarithm and secure authentication. In *PKC '00*, pages 113–128, London, UK, 2000. Springer-Verlag.
46. B. Preneel. *Analysis and design of cryptographic hash functions.* PhD thesis, Katholieke Universiteit Leuven, 1993.

47. M. Rabin. Digitalized signatures. In R. A. DeMillo, D. P. Dobkin, A. K. Jones, and R. J. Lipton, editors, *Foundations of Secure Computation*, pages 155–168. Academic Press, 1978.
48. P. Rogaway. Formalizing human ignorance. In *Progress in Cryptology - VIETCRYPT'06*, volume 4341, pages 211–228. Springer, 2006.
49. P. Rogaway and T. Shrimpton. Cryptographic hash-function basics: Definitions, implications, and separations for preimage resistance, second-preimage resistance, and collision resistance. In *Fast Software Encryption – FSE'04*, volume 3017 of *LNCS*, pages 371–388. Springer, 2004.
50. RSA Laboratories. PKCS ♯1 v2.1: RSA Cryptography Standard, June 2002.
51. D. Stinson. Some observations on the theory of cryptographic hash functions. *Des. Codes Cryptography*, 38(2):259–277, 2006.
52. S. Vaudenay. The security of DSA and ECDSA. In *PKC 2003*, volume 2567 of *LNCS*, pages 309–323. Springer-Verlag, 2003.
53. D. Wagner and I. Goldberg. Proofs of security for the unix password hashing algorithm. In D. Gollmann, editor, *ASIACRYPT 2000*, volume 1976 of *LNCS*, pages 560–572. Springer-Verlag, 2000.
54. R. S. Winternitz. Producing a one-way hash function from DES. In *CRYPTO'83*, pages 203–207, 1983.
55. G. Yuval. How to swindle Rabin. *Cryptologia*, 3:187–189, 1979.

A Security Requirement Definitions

To make this paper self-contained, we provide here a summary of hash function security requirement definitions referred to in Table 2 in the text. For each requirement listed, we provide: (a) Input and output domains of the hash function h , (b) the computational game between the adversary A and the challenger C, (c) the success condition for the adversary in the game, (d) the definition of adversary's *advantage* in the game, (e) an example of typical required upper bound for adversary advantage for typical adversary resources (here "run-time" t is typically measured as the sum of the run-time in machine instructions plus program length for some fixed computational machine model).

1. Security Requirement: $h(K, \cdot)$ is a PRF Family [2]
   (a) In/Out Domains: $h : B^b \times B^c \rightarrow B^c$
   (b) Game: (1) C chooses uniformly random bit $b \in B$. If $b = 0$, C chooses uniformly at random a function $h' : B^c \rightarrow B^c$ from the set of all functions from B^c to B^c. If $b = 1$, C chooses uniformly at random a key $K \in B^b$. (2) A runs with no input for time t and makes q queries to oracle $F : B^c \rightarrow B^c$ defined as follows: If $b = 0$, $F(x) \stackrel{\text{def}}{=} h'(x)$, and if $b = 1$, $F(x) \stackrel{\text{def}}{=} h(K, x)$. At the end of the game, A outputs a bit $b' \in B$.
   (c) A Success Condition: $b' = b$.
   (d) A Advantage: $Adv_{\mathsf{A}} \stackrel{\text{def}}{=} 2|Succ_{\mathsf{A}} - 1/2|$, where $Succ_{\mathsf{A}}$ is A's success probability.
   (e) Typical Quantitative Requirement: $Adv_{\mathsf{A}} \leq 2^{-80}$ for any A with resource bounds $t \leq 2^{80}$ and $q \leq 2^{40}$.

2. Security Requirement: $h_K(\cdot)$ is a CR Family [19]
   (a) In/Out Domains: $h_K : B^b \times B^c \rightarrow B^c$
   (b) Game: (1) C chooses uniformly random hash function key $K \in \mathsf{K}$ (K denotes hash family key space). (2) A runs with input K for time t and outputs $m, m' \in B^b \times B^c$.
   (c) A Success Condition: $m \neq m'$ and $h_K(m) = h_K(m')$.
   (d) A Advantage: $Adv_{\mathsf{A}} \stackrel{\text{def}}{=} Succ_{\mathsf{A}}$, where $Succ_{\mathsf{A}}$ is A's success probability.
   (e) Typical Quantitative Requirement: $Adv_{\mathsf{A}} \leq 2^{-80}$ for any A with resource bounds $t \leq 2^{80}$.
3. Security Requirement: $H_K(\cdot)$ is a TCR[4] Family [8, 44]
   (a) In/Out Domains: $H_K : B^* \rightarrow B^c$
   (b) Game: (1) A runs on no input and outputs $m \in B^*$, (2) C chooses uniformly random hash function key $K \in \mathsf{K}$ (K denotes hash family key space), (3) A continues to run on input K (for total time t) and outputs $m' \in B^*$.
   (c) A Success Condition: $m \neq m'$ and $H_K(m) = H_K(m')$.
   (d) A Advantage: $Adv_{\mathsf{A}} \stackrel{\text{def}}{=} Succ_{\mathsf{A}}$, where $Succ_{\mathsf{A}}$ is A's success probability.
   (e) Typical Quantitative Requirement: $Adv_{\mathsf{A}} \leq 2^{-80}$ for any A with resource bounds $t \leq 2^{80}$.
4. Security Requirement: $H_K(\cdot)$ is an eTCR Family [35]
   (a) In/Out Domains: $H_K : B^* \rightarrow B^c$
   (b) Game: (1) A runs on no input and outputs $m \in B^*$, (2) C chooses uniformly random hash function key $K \in \mathsf{K}$ (K denotes hash family key space), (3) A continues to run on input K (for total time t) and outputs $K' \in \mathsf{K}, m' \in B^*$.
   (c) A Success Condition: $(m, K) \neq (m', K')$ and $H_K(m) = H_{K'}(m')$.
   (d) A Advantage: $Adv_{\mathsf{A}} \stackrel{\text{def}}{=} Succ_{\mathsf{A}}$, where $Succ_{\mathsf{A}}$ is A's success probability.
   (e) Typical Quantitative Requirement: $Adv_{\mathsf{A}} \leq 2^{-80}$ for any A with resource bounds $t \leq 2^{80}$.
5. Security Requirement: $h(\cdot, \cdot)$ is e-SPR [35]
   (a) In/Out Domains: $h : B^b \times B^c \rightarrow B^c$
   (b) Game: (1) A runs on no input and outputs ℓ values $\Delta_1, \ldots, \Delta_\ell$, each in B^b, (2) C chooses uniformly random $r \in B^b$ and computes $m = r \oplus \Delta_\ell$ and $c = H^{c_0}(r \oplus \Delta_1, \ldots, r \oplus \Delta_{\ell-1})$ (here $c_0 \in B^c$ is a fixed IV and $H^{c_0}(x_1, \ldots, x_\ell)$ denotes the output of the ℓ-block Merkle-Damgård (MD) iteration function [19] with ℓ message blocks $(x_1, \ldots, x_\ell)$ and IV c_0), (3) A is given r, continues to run (for total time t) and outputs $c' \in B^c, m' \in B^b$.
   (c) A Success Condition: $(m, c) \neq (m', c')$ and $h(m, c) = h(m', c')$.
   (d) A Advantage: $Adv_{\mathsf{A}} \stackrel{\text{def}}{=} Succ_{\mathsf{A}}$, where $Succ_{\mathsf{A}}$ is A's success probability.

[4] The term TCR (Target Collision Resistant) was introduced by Bellare and Rogaway [8] as an alternative name for UOWHF (Universal One-Way Hash Function), introduced by Naor and Yung [44]. Both TCR and UOWHF terms are now commonly used in the cryptographic literature to refer to this notion.

(e) Typical Quantitative Requirement: $Adv_{\mathsf{A}} \leq 2^{-80}$ for any A with resource bounds $t \leq 2^{80}$ and $\ell \leq 2^{64}$.

6. Security Requirement: $h(\cdot)$ is preimage resistant (one-way) [53]
   (a) In/Out Domains: $h : B^b \rightarrow B^c$
   (b) Game: (1) C chooses uniformly random $x \in B^b$ and computes $y = f(x)$, (2) A runs on input y for time t and outputs $x' \in B^b$.
   (c) A Success Condition: $h(x) = h(x')$.
   (d) A Advantage: $Adv_{\mathsf{A}} \stackrel{\text{def}}{=} Succ_{\mathsf{A}}$, where $Succ_{\mathsf{A}}$ is A's success probability.
   (e) Typical Quantitative Requirement: $Adv_{\mathsf{A}} \leq 2^{-80}$ for any A with resource bounds $t \leq 2^{80}$.