IEICE TRANS. FUNDAMENTALS, VOL.E91–A, NO.1 JANUARY 2008

PAPER Special Section on Cryptography and Information Security

Classification of Hash Functions Suitable for Real-Life Systems

Yasumasa HIRAI[†a)], ***Member***, **Takashi KUROKAWA**[††], ***Nonmember***, **Shin'ichiro MATSUO**[†], **Hidema TANAKA**[††], ***Members***, ***and*** **Akihiro YAMAMURA**[††], ***Nonmember***

SUMMARY Cryptographic hash functions have been widely studied and are used in many current systems. Though much research has been done on the security of hash functions, system designers cannot determine which hash function is most suitable for a particular system. The main reason for this is that the current security classification does not correspond very well to the security requirements of practical systems. This paper describes a new classification which is more suitable for designing real-life systems. This classification is the result of a new qualitative classification and a new quantitative classification. We show a mapping between each class and standard protocols. In addition, we show new requirements for four types of hash function for a future standard.

key words: *security, hash function*

1. Introduction

A cryptographic hash function provides certain security properties and plays a crucial role in building security applications related to digital signatures, authentication, and message integrity. It is also used to construct pseudorandom number generators and practical instantiation of the random oracle. The most commonly used hash functions are MD5 and SHA-1, designed by Ronald Rivest and by the National Security Agency (NSA), respectively. These have become de facto standards and are widely used in many applications. Recently, successful attacks against hash functions have been demonstrated by numerous researchers. Among these, the attacks by Wang [54]–[56] have had a great impact on both theoretical and practical research on hash functions. Wang's attack finds a pair of two distinct messages with the same hash digest of SHA-0 and MD5. It takes only a few hours to find collisions of two distinct messages for MD5. On the other hand, according to Wang's latest announcement, the estimated cost to find a collision for SHA-1 is 2^{61}. These attacks unveil the vulnerability of some applications using existing hash functions, and the replacement of existing hash functions can be a severe burden for vendors, system designers, and users. In particular, the vulnerability of hash functions affects the public key infrastructure scheme and the replacement of weakened hash functions by new ones in PKI schemes or real-life protocols will be a major project. We note that Bellovin and Rescorla discuss how to deploy a new hash function without harming the existing real-life systems [7]. Hash functions are chiefly used for authentication and data integrity, and so the security provided can be measured by the lifespan of the data. Traditionally, the security notion of hash functions is discussed from a theoretical standpoint. Since hash functions are usually fixed techniques and the security parameter is not fed, the security is not measured asymptotically to the security parameter. On the other hand, it is sometimes necessary to consider the trade-off between security and performance because of some constraints. In this paper, we clarify the usage of cryptographic hash functions and the security requirement in practical security systems. In particular, we survey the usage in practical protocols specified in RFC and so on. Looking into practical applications, we observe that the security requirements are not properly evaluated. Therefore, a reexamination of the security concept of hash functions in protocols is required before replacement hash functions can be created. Hence, we discuss how to widen the scope of cryptographic hash functions and give a new classification for the future use of cryptographic hash functions in practical use. We also discuss changing the concepts of hash functions and their mode of operations.

In Sect. 2, we discuss the traditional classifications of hash functions and recall typical usages of hash functions in cryptographic schemes. In Sect. 3, we list several usages hash functions in real-life systems and describe hash functions from a system developer's viewpoint. We exemplify that hash functions are inevitable in numerous information communication systems even though the strong security properties of hash functions are not necessarily required in some cases. In Sect. 4, we argue the several reasons for traditional classification is not suitable for the current usages in real-life systems. Then we shall discuss how to classify hash-related techniques and how to evaluate its security properties in Sect. 5. We exclude message authentication codes (MAC) from our discussion because our goal is to discuss successor of the traditional collision resistant hash functions such as SHA-1. Then our proposal on classification will be given in Sect. 6 in which we emphasize four types of hash functions. The classification is made based on theoretical assurance, availability and performance. In Sect. 7, we consider additional requirements for hash functions.

Manuscript received March 27, 2007.
Manuscript revised July 3, 2007.

[†]The authors are with NTT DATA Corporation, Tokyo, 135-8671 Japan.
[††]The authors are with the National Institute of Information and Communications Technology (NICT), Koganei-shi, 184-8795 Japan.
a) E-mail: hiraiys@nttdata.co.jp

DOI: 10.1093/ietfec/e91–a.1.64

2. Existing Hash Function Classifications

Existing hash function classifications are based on only qualitative analysis. Hash functions are traditionally classified into two essentially distinct categories: non-cryptographic hash functions and cryptographic hash functions. Cryptographic hash functions are classified into message authenticated codes (MAC), one-way hash functions (OWHF), collision-resistant hash functions (CRHF), and universal one-way hash functions (UOWHF) (Fig. 1). We do not discuss MACs in this paper because our goal is to discuss successor of the traditional collision resistant hash functions such as SHA-1 and much of the research done on MACs is well known. Cryptographic hash functions are constructed using compression functions and composition methods. They are almost always required to be collision resistant, and most of the instantiations in practical security systems are restricted to a few hash algorithms of the SHA-family or the MD4-family. Non-cryptographic hash functions have been chiefly used to search database systems and have had nothing to do with security systems. Non-cryptographic hash functions may also require security to some extent. Of course, these need not be as secure as cryptographic hash functions. In this section, we review security properties of hash functions and explain why we need to re-examine hash functions.

2.1 Security of Cryptographic Hash Functions

Let $h : \{0,1\}^* \to \{0,1\}^n$ be a hash function. Cryptographic hash functions should have the following properties.

Collision resistant : The computational cost to find the input pair x and x' which holds $h(x) = h(x')$ must not be smaller than $2^{\frac{n}{2}}$, which is estimated with respect to the birthday paradox.

Pre-image resistant : The computational cost to find the input x, where $h(x) = y$ must not be smaller than 2^n.

Second pre-image resistant : The computational cost to find the input $x'(\neq x)$, where $h(x) = y$ and $h(x') = y$ must not be smaller than 2^n. It has been shown in [30] that the necessary number of algorithm step may be less than 2^n if immense memory is allowed, that is, a kind of time-memory trade-off methods is used.

Universal one-way : Choose an element $x \in D$. When a key K is randomly chosen, it is hard for an adversary to find $y \in D$ $(x \neq y)$ such that $h_K(x) = h_K(y)$.

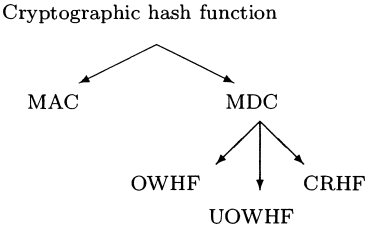


Fig. 1 Cryptographic hash functions.

2.2 Collision Resistant Hash Functions

Most collision-resistant hash functions $h : \{0,1\}^* \to \{0,1\}^n$ are constructed through the iterated use of compression function $f : \{0,1\}^{n+m} \to \{0,1\}^n$. Then the security of such hash functions depends on the design of compression functions and the iteration methods.

2.2.1 Construction

The Merkle-Damgård construction is a well known solution for the question of how to iterate a compression function [17], [35]. Tree construction is another way, but most practical hash functions are based on Merkle-Damgård or its extensions. Some research has examined the security relation between f and h, but has not led to a practical improvement in security. Recent attacks (e.g., Wang et al.) show that pseudo-collisions or near-collisions are a real threat to the Merkle-Damgård construction. A solution to these potential problems is needed.

2.2.2 Compression Function

Block cipher based: Block ciphers are used as compression functions because they are highly trusted functions that provide practical security and are easy to evaluate. PGV-style construction is widely used [40]. On the other hand, they must execute the key set-up function many times, so the performance is very slow.

Dedicated function: The MD4-family that has evolved from MD4 to SHA-224/-256/-384/-512 is a de facto standard. Many types of dedicated function have been chosen as international standards (e.g., SHA-1,2, RIPEMD-160, Whirlpool). Before 1998, no result was known for attacks against hash functions except for the attacks against the reduced MD4 and the near collision of [53]. In 1998, MD4 was completely broken by Dobbertin [20]. On the other hand, analysis had confirmed the security properties of the MD5 and SHA-1 compression functions and the collision search for reduced MD5, SHA-0, and SHA-1 until 2004 In 2004, MD5 was completely broken by Wang et al. [54] and SHA-1 is not considered secure enough [55]. Though practical performance is good, most dedicated functions are in a critical situation now.

Arithmetic function: MASH-1/-2, which are based on modular arithmetic, were among the functions chosen for ISO/IEC 10118-4. The functions based on the knapsack problem have unresolved problems regarding security. Some, such as VSH [13], have provable collision resistance, but most are not suitable for practical use.

2.2.3 Recent Works

In response to the SHA-1 crisis, some researches have

shown that security can be improved with minimal changes made to the algorithm. SHA-1 IME [51] has an improved form of message preparation compared to that of SHA-1, and randomized hash [26] is a technique which makes it difficult to find a collision pair through a randomizing message. In addition, new dedicated hash functions have been proposed (FORK-256, DHA-256, etc.). While most hash functions have a fixed output length, there have been some attempts to enable a variable length. For example, SHA-V is a variable length hash function based on SHA-1 [27].

2.3 Random Oracle Methodology

In practical systems, random oracles are instantiated by a hash function. Such instantiation does not necessarily provide a level of security equal to provable security. We should not depend too much on random oracle models: several studies have shown that various cryptographic systems are secure in the random oracle model, but insecure for any concrete instantiation of the random oracle [11], [16], [25]. At present, we should concentrate on techniques such as signature schemes and time stamps. There are several approaches that enable provable security with or without depending on random oracle models [9], [14].

2.3.1 Full Domain Hash Scheme

Here, the output size of a hash function is exactly the size of the (RSA) modulus. Implementing such a scheme requires a concrete method, such as MGF1, to extend the length of a hash function. In general, it is desirable to have consistency among the hash sizes and other security parameters such as the size of the modulus of the composite numbers used in factoring-based systems. It may be better to have hash functions that allow choice regarding the digest size.

2.3.2 Mask Generation Functions

In some provably secure cryptosystems, a mask generation function, such as MGF1 [62], [64], is used as a building block for instantiating random oracles. For example, RSASSA-PSS [64] was recently included in NIST FIPS 186-3 [60] and SP800-78 [61]. MGF1 is based on the ideas of Bellare and Rogaway [4], [5]. It is viewed as a custom method to extend the length of a hash function.

2.4 UOWHF (Universal One-Way Hash Function)

UOWHFs (introduced by Naor and Yung [36]) can be considered keyed hash functions that satisfy the following property: an adversary chooses a message x and then the function h_K is chosen randomly; it then becomes hard for the adversary to find y ($x \neq y$) such that $h_K(x) = h_K(y)$.

Signature schemes constructed based on the hash-and-sign method are no more secure than a hash function against a collision-finding attack. The recent attacks by Wang against SHA-1 call these methods into question. It is well known that a simple modification of the hash-and-sign paradigm may replace the collision-resistant hash with a UOWHF [6], [36]. On the other hand, the UOWHF has several disadvantages. So far, UOWHFs are inefficient compared to CRHFs; the key size grows logarithmically with the message size. In fact, a few practical techniques currently use UOWHFs; for example, ACE-Sign [49] is based on UOWHFs and has a not so tight reduction [33].

Defects of UOWHFs are that there is no international standard for UOWHFs and its poor performance and these demerits have slowed its development and deployment.

3. Hash Functions in Information Systems

Generally, a hash function is used to securely provide services through systems. However, many system developers and engineers use security products in which a hash function is applied, yet they are not conscious of each hash function's security requirements. They therefore choose hash functions such as MD5 and SHA-1 without being aware of how an attack on the hash function can affect a system. In this section, we describe hash function requirements from a system developer's viewpoint. First, we describe the requirements during the design of systems. After that, we describe the current usage of hash function in systems. We then categorize hash function usage with regard to the requirements.

3.1 Requirements in System Design

The system design requirements are as follows.

(1) Security requirements

Systems have to fulfill the following security requirements to provide services securely.

Confidentiality Systems which store and transmit data, such as individual information, have to protect it from eavesdropping by malicious users.
Authentication : Some systems have to identify users for payment, and confirm the validity of each operator to prevent internal crimes.
Certification : Some systems which issue certificates, such as a driver's license, have to prevent falsification of certificates and guarantee their validity.

(2) System development requirements

Performance : The design or algorithms used in a system are chosen based on requirements with respect to performance. For example, a light-weight algorithm is needed for low-power devices.
Development-cost : In almost all cases, the development cost is decided beforehand, and this includes the cost of security mechanisms. However, most customers are unaware of the additional costs of replacing a broken hash function with a new one.
Development period and system life cycle : Most information systems have a life cycle which includes develop-

ment, operation, and renewal phases. The basic design and algorithm are decided early in the development phase. Thus, when a hash function is broken during a later phase, replacement of the hash function will be planned immediately before the next renewal.

Interoperability : Some information systems such as financial system are related to other systems. For example, if a bank (A) replaces the hash function used in smart card and ATM network system, ATM systems of other banks are required to be able to accept the smart card issued by A [52].

(3) Requirements for each service

Long-term assurance : In financial systems and healthcare systems, many documents must be securely preserved for a long time. For example, in the United States, the Sarbanes-Oxley (SOX) Act obliges corporations to store fiscal reports, audit reports, and all business transaction and conference related documents for at least five years [65]. If, for example, we use time-stamp technology to ensure the integrity of documents, this sort of legislation means that a time-stamping system must provide long-term assurance, such as a guarantee of at least five years.

Enforcement of products : The Certificate Authority (CA) on PKI and Time Stamping Authority (TSA) systems, for example, must be high-level security systems. Therefore, customers typically request that these systems use a Hardware Security Module (HSM) that has been validated under the Cryptographic Module Validation Program (CMVP).

3.2 Usage of Hash Functions in Systems

Many information systems use security protocols with hash functions, such as SHA-1. In this section, we list the protocols used with a hash function, and the impacts of broken of hash functions.

3.2.1 Protocols Used with Hash Function

(1) Certification

This category consists of certification protocols which guarantee the validity of signers, documents, issuers, etc. For example, *digital signature schemes* use a hash function to ensure the integrity of messages. *PKIX* uses a hash function to ensure the integrity of the certificates and CRL profiles. Moreover, *Time stamping protocols* use hash functions to ensure the integrity of the time stamp token.

(2) Authentication

This category consists of authentication protocols which confirm clients, etc. For example, *Kerberos* uses a hash function to calculate the hash value of the entered client password and this becomes the secret key of the client. *IEEE802.1X-EAP* provides authentication, active key generation, and key delivery on wireless networks. The EAP-FAST protocols in *IEEE802.1X-EAP* are based on the *TLS* handshake protocol, which uses a hash function. Moreover, *APOP* uses a hash function (MD5) to prevent replay attacks and disclosure of a shared secret. *RADIUS* is also a protocol that uses a hash function (e.g., SHA-1) to ensure the integrity of data and applications in information systems.

(3) Secure communication

This category consists of the secure communication protocols which include key exchange, etc. For example, *Internet Key Exchange (IKE)* protocols in *IPsec* use hash functions as pseudo-random functions. Moreover, hash functions are used to ensure the integrity of all protocol messages in protocols. *Secure Socket Layer (SSL)* and *Transport Layer Security (TLS)* prevent eavesdropping, falsification, and message forgery. The handshake protocol in *SSL* uses a hash

Table 1 Requirements for hash function in systems.

Requirements		IETF RFC	Security			Development			Service
Protocols			Confidentiality	Authentication	Certification	Performance	Cost-Effectiveness	Renewal cost	
Certification	Digital Signature	-			○	smart card		High	DKIM: Short-term Document: Medium-term
	PKIX	X.509 Cert & CRL: RFC3280 Extensions: RFC4325							Long-term assurance
	Time Stamp	RFC3161							
Authentication	Kerberos	RFC4120		○	○			Medium	Short-term
	IEEE802.1X-EAP	draft-cam-winget-eap-fast03.txt							
	APOP	RFC2195 RFC2095	○					Low	Short-term
Secure Communication	IPsec	IKE: RFC4109 ESP: RFC2406		○	○			Medium	Short-term
	SSL/TLS	TLS Protocol: RFC2246 Extensions: RFC3546							
	SSH	RFC4253							
Secure E-mail	S/MIME	Ver2: RFC2312, RFC2311 Ver3.1: RFC3851, RFC3850		○	○			Medium	Long-term
	PGP	RFC3156							
Others	Packet Sampling Filtering	draft-ietf-psamp-sample-tech-07.txt						Medium	Short-term
	Database Retrieval	-						Medium	Medium-term
	Software Download	-			○			Low	Long-term

function to create a message authentication code. Moreover, *SSH* uses hash functions to produce message authentication codes that ensure the integrity of data exchanged between the two computers.

(4) Secure E-mail

This category consists of secure e-mail protocols which encrypt e-mail messages to protect privacy and verify the integrity of messages. For example, *S/MIME* and *PGP* uses a hash function in a digital signature scheme to ensure integrity of e-mail messages. *S/MIME* also uses it to ensure the authenticity of them.

(5) Other protocols

Protocols in this category do not have precise security requirements regarding the hash function. For example, *Packet sampling protocols* use hash functions as pseudo-random functions to select packets at random. *Packet filtering protocols* use hash functions to generate packet digests which are used to identify packets. In *Database retrieval* system, hash functions are applied to generate keys used to efficiently retrieve target data from large bodies of data stored in a database. Moreover, *Software download* systems use hash values as check sums to ensure the integrity of software.

We show the classification of these protocols in Table 1. Moreover, in Table 1, we describe about "Renewal cost" against protocols shown in Sect. 3.2.1. For example, when system developers update the information system used "Certification protocols," they need high renewal cost. Because information systems which use "Certification protocols" are required to collect and re-issue certifications. In the case of Secure communication protocols, etc., they need medium renewal cost.

4. Gaps between Existing Classification and Current Usage

In this section, we first discuss gaps between the area covered by the existing classification and the current usage of hash functions. After that, we state the classification requirements suitable for both worlds.

4.1 Lack of Quantitative Security

In Sect. 2, we described the existing hash function classifications. These classifications are based on qualitative analysis. However, the security of real-life systems is considered from the perspective of a risk analysis method such as ISMS, ISO/IEC 15408, ISO/IEC 27001-2005, and so on. The results of such an analysis are quantitative. For example, a digital signature used in time-stamping services must be valid for over seven years for SOX Act, and a digital signature for certificates must be valid for one to three years and remain resistant against possible adversaries.

On the other hand, a hash value used in only one session must be valid until the end of the session. If a hash value is used as a cookie in a Web-based service to prevent a man-in-the-middle attack, the system must prevent generation of a collision for the hash value before the end of the session. Typically, a session does not last long (i.e., one hour or less). Furthermore, some protocols which use hash functions do not need collision resistance and are used as compression functions. To protect such systems from being needlessly weakened, a weaker security class should be added and a guideline for such usage provided. The problem in this area is there are no security criteria for such usage.

We must consider the quantitative security aspect of hash functions and include this in the classification.

4.2 Reconsideration of Existing Classification

Current qualitative criteria for hash functions is collision resistance, second pre-image resistance and one-wayness as explained in Sect. 2. These three notions are all rigorously defined. However, there are no known provable secure constructions of them. Thus, they do not yield secure signature scheme.

On the other hand, in governmental cryptographic standard such as NESSIE and CRYPTREC, "provable security" is quite important nature for cryptographic algorithms. Government recommend provable secure digital signature schemes in e-government systems. And most of all applications of hash functions requires collision resistance. Here, we reconsider qualitative classification from applications point of view. We categorize secure hash functions into two class. One is collision for practical use. The other is a class for realizing secure digital signature. This class contains UOWHF. Hereafter we write former class as CRHF, and latter class as UOWHF.

4.3 Requirements for New Classification

For the reasons given above, we must define a new system of classification which covers everything from cryptographically strong classes to light-weight but practically secure classes.

The requirements that such a classification must meet are as follows.

- The classification must be redefined to cover the entire range from provable security to light and practical security.
- The classification must include a quantitative index, which covers from long-term security assurance to short-term security assurance.

Thus, a new classification system will be based on a two-dimensional matrix containing both qualitative index values and quantitative index values.

5. New Security Classification of Hash Functions

5.1 Qualitative Classification

At first, we add a qualitative classification. This classification covers theoretical security requirements. We describe these requirements as follows.

5.1.1 Requirement of CRHF

CRHFs should be developed by taking into account the following points.

Need to improve the compression function f

This will greatly affect the security and performance of a CRHF. Dedicated designs should be devised to achieve a qualitative security property. Arithmetical functions should improve performance.

Need to improve the construction h

The Merkle-Damgård construction has the property $h(x_1||x_2; IV) = h(x_2; h(x_1; IV))$. In an actual system, the demand for iterated hash functions is consequently large because of their good performance and ease of use. Therefore, the iterated construction should be used to find another construction which can solve the pseudo-collision problem or provide minimal security regarding collision resistance in a theoretical way.

Size of hash value

In addition to the demand for iterated hash functions, as stated above, from the viewpoint of hash size compatibility a decision on a standard 128-bit or 160-bit CRHF is desirable. For such use, security requirements are within a limited range. It is necessary to show that this limited security is enough for such usage.

5.1.2 Requirement for UOWHF

Naor and Yung's method [36] is not really practical because the length of keys for a UOWHF becomes longer in proportion to the message size. Bellare and Rogaway [6] provide a better solution to this problem. At present, the best methods provide signature techniques with a UOWHF where the key grows logarithmically with the message size. Among these, Shoup [50] gives the most efficient construction, which can be considered an extended Merkle-Damgård construction.

Theoretical assurance of the security of signature schemes has attracted increased attention since Wang's attacks. A UOWHF has some advantages compared to a CRHF. It is theoretically possible to construct a UOWHF using only a one-way function. This assumption is much weaker than the assumption needed for CRHFs. Most signature techniques depending on CRHFs are vulnerable to the recent attacks of Wang et al. However, signature techniques have been introduced that depend only on UOWHFs [36], [45] and Wang's attack is not a threat to signatures using UOWHFs. In addition, the security of UOWHFs is not compromised by the birthday paradox attack. Thus, the cost of a generic attack on a UOWHF is 2^n if the size of the hash digest is n. Note that the cost of a generic attack (a birthday paradox attack) on a CRHF is $2^{\frac{n}{2}}$. There is a trade-off between such generic security criteria and usability; UOWHFs need keys and the key length grows logarithmically with the message size, whereas a CRHF needs no key (Table 2).

Gennaro, Gertner, Katz and Trevisan [24] claim that the construction by [36] is optimal among a black box construction of UOWHFs using one-way functions. It follows that there is no effective way to construct a UOWHF in a universal manner. Therefore, it is necessary to construct an effective UOWHF directly from its theoretical primitives. In fact, we do not know whether an effective UOWHF actually exists. A worthy goal is to show whether an effective UOWHF construction exists and to standardize it if it does.

5.1.3 Mode of Operations

The mode of operation related to hash functions would be desirable in many ways; it provides additional properties for security systems based on hash functions. The ability to change the length of hash digests would be an attractive feature. It is sometimes critical to specify the object identifier (OID) of hash functions, which is known as the hash firewall [29]. It may be possible to adopt a mode of operation that takes care of this.

5.2 Quantitative Classification from Valid Period

Next, we add a quantitative classification. We categorize the quantitative security of hash functions into three groups.

(1) Long-term security

This category contains certificates assuring that data has not

Table 2 Comparison between CRHF and UOWHF.

	CRHF	UOWHF
Key	No	Key size grows logarithmically with the message size
Adversary Goal	Find $x, y \in D$ $(x \neq y)$ s.t. $h(x) = h(y)$	Choose $x \in D$, Given $h_K \in H$ Find $y \in D$ $(x \neq y)$ s.t. $h_K(x) = h_K(y)$
Compression functions	Dedicated functions Block cipher based Arithmetic	Strongly universal$_2$ functions
Construction methods	Merkle-Damgård Tree	XOR linear, XOR tree Shoup (extended Merkle-Damgård)
Standard	ISO 10118-3	No standard exists

been forged over long periods of time; e.g., over five years to comply with the SOX Act. A time-stamp token issued by a time-stamp authority is an example of this category. In such applications, the system must prevent forgery of time-stamp tokens over long time spans. Here, we assume that a hash function which will be secure over five years must be used for systems in this category. In addition, any desired security parameter (e.g., the search space) must exceed 2^{128} to account for the possibility of a birthday attack.

(2) Medium-term security

This category contains certificates which assure the data has not been forged and these certificates must remain reliable for two to three years. Public key certificates are a good example of this.

In this kind of application, the collision resistance of a hash function is important, but the required security is weaker than long-term security. Here, we assume that a hash function which will be secure anywhere from 1 month to five years should be used for systems in this category. In addition, the desired security parameter (e.g., the search space) must be 2^{80} to cope with birthday attacks.

(3) Short-term security

This category contains hash values and digital signatures used for key exchange, to prevent replay attacks and man-in-the-middle attacks, or for temporary authentication. The valid period of such values varies according to the services; however, this is limited to the period of service use. This is typically one month at the longest.

In this kind of application, the collision resistance of a hash function is still important, but the required security is weaker than medium-term security, and a lighter and minimally secure hash function is desirable. Some standard protocols require a short hash length, which is limited by the packet size and performance requests. Such usage also falls into this category. We assume that hash functions which will be secure for up to one month can be used for systems in this category. In addition, the desired security parameter (e.g., search space) should be 2^{64} to cope with birthday attacks. A shorter security parameter may be acceptable depending on the security policy of each system. These three categories are compared in Table 3.

In NIST Second Cryptographic Hash Workshop, several panelists discussed a new hash function. The consensus seemed to be that one function is optimal. In addition, some panelists stated the necessity of multiple modes. They state that a new hash function should have variable output length to achieve several different strengths. The above quantitative classification that we propose is corresponding to their opinions.

Table 3 Comparison among three categories.

Category	Period	Security parameter
Long-term	over 5 years	2^{128}
Medium-term	1 month - 5 years	2^{80}
Short-term	under 1 month	2^{64}

6. Our Proposal

In this section, we propose a new classification, then map the various types of usage described in Sect. 3.2 with respect to this classification.

6.1 New Classification

The new classification is based on a two-dimensional matrix with qualitative index values and quantitative index values. The qualitative index has two members as described in Sect. 5.1. These are the CRHF and the UOWHF. The quantitative index has three members, described in Sect. 5.2. These are long-term security, medium-term security, and short-term security. Consequently, the resulting matrix contains six cells.

Next, we map current forms of usage (described in Sect. 3.2) into the matrix. Table 4 shows the result of this mapping.

At first, when a system designer uses hash function in information systems, the system designer chooses guaranteed term, which is decided by requirements for service, from quantitative classification. For example, if system designer develops time-stamping system related to SOX Acts, he chooses long-term. Next, he chooses qualitative security property from qualitative classification. If high security such as provable secure digital signature is required in time stamping system, system designer chooses UOWHF. Thus system designer is able to decide a hash function type suitable for the system, uses a hash function classified in this type.

In Table 4, only four of the six cells are occupied. These are the four types of hash function that should be considered to establish an industrial standard.

6.2 Desired Hash Function Features

As shown in Table 4, four types of hash function are required for real-life information systems.

Type1. UOWHF with a long security parameter:
In this type, the hash function must have universal one-wayness and consider the provable security. The hash function is required to preserve universal one-wayness over a long term. Thus, the security parameter of this hash function must be over 2^{128} and be extendable.

Type2. CRHF with a long security parameter:
In this type, the hash function must be collision resistant and preserve its collision resistance over a long term. Thus, the security parameter of this hash function must exceed 2^{128} and the security parameter must be extendable.

Table 4 Proposed classification and mapping of current usage.

	CRHF	UOWHF
Long-term	Certification (Time-stamping by hash func.) Integrity check (Software Download)	Certification (Time-stamping by signature, Code Signing) Secure E-mai (S/MIME,PGP)
Medium-term	N/A	Certification (PKIX)
Short-term	Secure communication (IPSEC, SSL/TLS, SSH) Authentication (IEEE 802.1X-EAP, Kerberos, APOP, DKIM) Other usage (Packet sampling/filtering, Database Retrieval)	N/A

Type3. UOWHF with a medium security parameter:
In this type, the hash function must have universal one-wayness and consider the provable security. The hash function must preserve universal one-wayness over the medium term. Thus, the security parameter of this hash function must exceed 2^{80}.

Type4. CRHF with a short security parameter:
In this type, the hash function must be collision resistant, and preserve its collision resistance in the short term. Thus, the security parameter of this type of hash function may be about 2^{64}. A shorter security parameter may be chosen if security conditions permit.

The above four types can cover almost all the existing usage of hash functions. Thus, they meet the requirements for hash functions suitable for real-life systems.

7. Discussions

Hash functions have to satisfy other requirements from a system designer's point of view. Two additional points should be considered when designing hash functions.

(1) Compatibility with existing systems

When a system engineer plans to replace a weak hash function in a system with a stronger one, he must consider the replacement's compatibility with the existing system's design to minimize the replacement cost. The main issue is usually the hash value length. When designing a system, the designer decides which data structures to use for communication messages, databases, and so on. Changing the hash value length can necessitate system re-design, re-coding, and re-testing. The affected parts include not only hash-related processing, but also non-cryptographic processing. Thus, designing strong hash functions with the same output length is preferable for most system designers.

(2) Implementation for embedded hardware

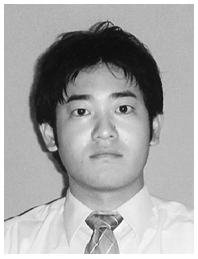
The key devices in current security systems include tamper-resistant devices such as smart cards. Smart cards are used for user authentication, key management, signing documents, and so on.

For example, in general PKI, a key pair of a public key cryptosystem is securely stored in the smart card of each user, and a digital signature and ciphertext is calculated within the smart card. Thus, a hash function must be implemented into the smart card. Smart cards are widely used in current systems, and there is no good alternative for them at present, so replacing the hash functions used in today's smart cards is a major concern regarding the hash function transition in information systems.

References

[1] B. Aboba and P. Calhoun, "RADIUS (Remote authentication dial in user service) support for extensible authentication protocol (EAP)," RFC3579, 2003.
[2] C. Adams, P. Cain, D. Ponkas, and R. Zuccherato, "Internet X.509 public key infrastructure time-stamp protocol (TSP)," RFC3161, 2001.
[3] J. Arkko, V. Torvinen, G. Camarillo, A. Niemi, and T. Haukka, "Security mechanism agreement for the session initiation protocol (SIP)," RFC3329, 2003.
[4] M. Bellare and P. Rogaway, "Optimal asymmetric encryption," Advances in Cryptology, EUROCRYPT'94, LNCS 950, pp.92–111, Springer-Verlag, 1995.
[5] M. Bellare and P. Rogaway, "The exact security of digital signatures-how to sign with RSA and Rabin," Advances in Cryptology, EUROCRYPT'96, LNCS 1070, pp.399–416, Springer-Verlag, 1996.
[6] M. Bellare and P. Rogaway, "Collision-resistant hashing: Towards making UOWHFs practical," Advances in Cryptology — CRYPTO'97, LNCS 1294, pp.470–484, Springer-Verlag, 1997.
[7] S. Bellovin and E. Rescorla, "Deploying a new hash algorithm," NIST Hash Function Workshop, pp.15–28, 2005.
[8] S. Blake-Wilson, M. Nystrom, D Hopwood, J. Mikkelsen, and T. Wright, "Transport layer security (TLS) extensions," RFC3546, 2003.
[9] A. Boldyreva and M. Fischlin, "Analysis of random oracle instantiation scenarios for OAEP and other practical schemes," Advances in Cryptology — CRYPTO 2005, LNCS 3621, pp.412–429, Springer-Verlag, 2005.
[10] N. Cam-Winget, D. McGrew, J. Salowey, and H. Zhou, "The flexible authentication via secure tunneling extensible authentication protocol method (EAP-FAST)," draft-cam-winget-eap-fast-03
[11] R. Canetti, O. Goldreich, and S. Halevi, "The random oracle methodology, revisited," STOC'98, pp.209–218, ACM, 1998.
[12] J.L. Carter and M.N. Wegman, "Universal classes of hash functions," JCSS, vol.18, pp.143–154, 1979.
[13] S. Contini, A.K. Lenstra, and R. Steinfeld, "VSH, an efficient and provable collision resistant hash function," http://eprint.iacr.org/2005/193
[14] J. Coron, Y. Dodis, C. Malinaud, and P. Puniya, "Merkele-Damgård revised: How to construct a hash function," Advances in Cryptology — CRYPTO 2005, LNCS 3621, pp.430–448, Springer-Verlag, 2005.
[15] B. den Boer and A. Bosselaers, "An attack on the last two rounds of MD4," Advances in Cryptology — CRYPTO'91, LNCS 576,

pp.194–203, Springer-Verlag, 1992.
[16] Y. Dodis, R. Oliveria, and K. Pietrzak, "On the generic insecurity of the full domain hash," Advances in Cryptology — CRYPTO 2005, LNCS 3621, pp.449–466, Springer-Verlag, 2005.
[17] I. Damgård, "A design principle for hash functions," Advances in Cryptology — CRYPTO'89, LNCS 435, pp.416–427, Springer-Verlag, 1990.
[18] T. Dierks and C. Allen, "The TLS protocol," RFC2246, 1999.
[19] S. Dusse, P. Hoffman, B. Rmsdell, L. Lundblade, and L. Repka, "S/MIME version 2 message specification," RFC2311, 1998.
[20] H. Dobbertin, "Cryptanalysis of MD4," J. Cryptology, vol.11, no.4, pp.253–271, 1998.
[21] S. Dusse, P. Hoffman, B. Rmsdell, and J. Weinstein, "S/MIME version 2 certificate handling," RFC2312, 1998.
[22] M. Elkins, D. Del Torto, R. Levien, and T. Rossler, "MIME security with OpenPGP," RFC3156, 2001.
[23] S. Farrell, T. Kause, and T. Mononen, "Internet X.509 public key infrastructure certificate management protocol (CMP)," RFC4210, 2005.
[24] R. Gennaro, Y. Gertner, J. Katz, and L. Trevisan, "Bounds on the efficiency of generic cryptographic constructions," SIAM J. Comput., vol.35, pp.217–246, 2005.
[25] S. Goldwasser and Y. Tauman, "On the (In)security of the Fiat-Shamir paradigm," Proc. 44th Annual IEEE Symposium of Foundations of Computer Science, pp.102–114, 2003.
[26] S. Halevi and H. Krawczyk, "Randomized hashing: Secure digital signatures without collision resistance," http://www.ee.technion.ac.il/~hugo/rhash.pdf
[27] Y.S. Her and K. Sakurai, "Analysis and design of SHA-V and RIPEMD-V with variable output-length," http://itslab.csce.kyushu-u.ac.jp/~ysher/14.pdf
[28] R. Housley, W. Polk, W. Ford, and D. Solo, "Internet X.509 public key infrastructure certificate and certificate revocation list (CRL) profile," RFC3280, 2002.
[29] B. Kaliski, "On hash function firewalls in signature schemes," Cryptographers' Track at the RSA Conference 2002, LNCS 2271, pp.1–16, Springer-Verlag, 2002.
[30] J. Kelsey and B. Schneier, "Second preimages on n-bit hash functions for much less than 2^n work," Advances in Cryptology—Eurocrypt 2005, LNCS 3494, pp.474–490, Springer-Verlag, 2005.
[31] S. Kent and R. Atkinson, "IP encapsulating security payload (ESP)," RFC2406, 1998.
[32] J. Klensin, R. Catoe, and P. Krumviede, "IMAP/POP AUTHorize extension for simple challenge/response," RFC2095, 1997.
[33] The NESSIE book (draft version April 19 2004). Available at http://www.cryptonessie.org
[34] P. Hoffman, "Algorithms for Internet key exchange, Version 1 (IKEv1)," RFC4109, 2005.
[35] R. Merkle, "One way hash functions and DES," Advances in Cryptology — CRYPTO'89, LNCS 435, pp.428–446, Springer-Verlag, 1990.
[36] M. Naor and M. Yung, "Universal one-way hash functions and their cryptographic applications," Proc. 21st STOC, pp.33–43, 1989.
[37] C. Neuman, S. Hartman, and K. Raeburn, "The Kerberos network authentication service (V5)," RFC4120, 2005.
[38] J. Peterson, "S/MIME advanced encryption standard (AES) requirement for the session initiation protocol (SIP)," RFC3853, 2004.
[39] W. Polk, R. Housley, and L. Bassham, "Algorithms and identifiers for the Internet X.509 public key infrastructure certificate and certificate revocation list (CRL) profile," RFC3279, 2002.
[40] B. Preneel, R. Govaerts, and J. Vandewalle, "Hash functions based on block ciphers: A synthetic approach," Advances in Cryptology — CRYPTO'93, LNCS 773, pp.368–378, Springer-Verlag, 1993.
[41] B. Ramsdell, "Secure/multipurpose Internet mail extensions (S/MIME) version 3.1 certificate handling," RFC3850, 2004.
[42] B. Ramsdell, "Secure/multipurpose Internet mail extensions (S/MIME) version 3.1 message specification," RFC3851, 2004.
[43] C. Rigney, W. Willats, and P. Calhoun, "RADIUS extensions," RFC2869, 2000.
[44] C. Rigney, S. Willens, A. Rubens, and W. Simpson, "Remote authentication dial in user service (RADIUS)," RFC2865, 2000.
[45] J. Rompel, "One-way functions are necessary and sufficient for secure signatures," Proc. 22nd STOC, pp.387–394, 2000.
[46] S. Santesson and R. Housley, "Internet X.509 public key infrastructure authority information access certificate revocation list (CRL) extension," RFC4325, 2005.
[47] J. Schaad, "Internet X.509 public key infrastructure certificate request message format (CRMF)," RFC4211, 2005.
[48] J. Schaad, B. Kaliski, and R. Housley, "Additional algorithms and identifiers for RSA cryptography for use in the Internet X.509 public key infrastructure certificate and certificate revocation list (CRL) profile," RFC4055, 2005.
[49] T. Schweinberger and V. Shoup, "ACE: The advanced cryptographic engine," Aug. 14, 2000. Available at http://www.shoup.net
[50] V. Shoup, "A composition theorem for universal one-way hash functions," Advances in Cryptology — EUROCRYPT 2000, LNCS 1807, pp.445–452, Springer-Verlag, 2000.
[51] M. Szydlo and Y.L. Yin, "Collision-resistant usage of MD5 and SHA-1 via message preprocessing," Topics in Cryptology — CT-RSA 2006, LNCS 3860, pp.99–114, Springer-Verlag, 2006.
[52] M. Une and M. Kanda, "Year 2010 issues on cryptographic algorithms, monetary and economic studies," Bank of Japan, vol.25, no.1, pp.129–164, 2007.
[53] S. Vaudenay, "On the need for multipermutations: Cryptanalysis of MD4 and SAFER," Fast Software Encryption — FSE'95, LNCS 1008, pp.286–297, Springer-Verlag, 1995.
[54] X. Wang, D. Feng, X. Lai, and H. Yu, "Collisions for hash functions MD4, MD5, HAVAL-128 and RIPEMD," IACR Eprint archive 2004/199, Aug. 2004.
[55] X. Wang, Y.L. Yin, and H. Yu, "Finding collisions in the full SHA-1," Advances in Cryptology — CRYPTO 2005, LNCS 3621, pp.17–36, Springer-Verlag, 2005.
[56] X. Wang, A. Yao, and F. Yao, "Cryptanalysis on SHA-1," Cryptographic Hash Workshop, Keynote Speech, 2005.
[57] T. Ylonen and C. Lonvick, "The secure shell (SSH) transport layer protocol," RFC4253, 2006.
[58] T. Zseby, M. Molina, N. Duffield, S. Niccolini, and F. Raspall, "Sampling and filtering techniques for IP packet selection," draft-ietf-psamp-sample-tech-07
[59] G. Zorn, D. Leifer, A. Rubens, J. Shriver, M. Holdrege, and I. Goyre, "RADIUS attributes for tunnel protocol support," RFC2868, 2000.
[60] National Institute of Standards and Technology, "Draft federal information processing standard (FIPS) 186-3 - Digital signature standard (DSS)," March 2006. Available at http://csrc.nist.gov/
[61] National Institute of Standards and Technology, "NIST special publication 800-78: Cryptographic algorithms and key sizes for personal identity verification," April 2005. Available at http://csrc.nist.gov/
[62] IEEE, "IEEE Std 1363-2000: Standard specifications for public-key cryptography," Jan. 2000.
[63] "Information technology — Security techniques — Time-stamping services," ISO/IEC18014-2.
[64] RSA Laboratories, "PKCS#1 v2.1: RSA cryptography standard," June 2002. available at http://www.rsasecurity.com/rsalabs/
[65] "Sarbanes-Oxley Act of 2002," SOX:www.sec.gov/about/laws/soa2002.pdf



Yasumasa Hirai received the B.E. and M.E. degrees in 2001 and 2003, respectively, from Ritsumeikan University. Since 2003, he has been a researcher at NTT DATA Corporation. He is presently engaged in research and development on cryptography and information security.

Takashi Kurokawa received the B.S. degree in mathematics from Waseda University, Tokyo, Japan, in 1988 and the M.S. in mathematics from the University of Tokyo in 1991. He has been engaged in research and development on cryptography and information security at Japan Systems Corporation, Tokyo, Japan, since 2000. He was a researcher of Information-Technology Promotion Agency, Japan, from 2001 to 2003 and a guest researcher of National Institute of Information and Communications Technology from 2004 to 2007.

Shin'ichiro Matsuo received the B.E., M.E. and D.E. degrees in 1994, 1996 and 2003, respectively, from Tokyo Institute of Technology. From 1996, he has been a research scientist at NTT DATA Corporation. He is presently engaged in researches on information security and cryptographic protocols.

Hidema Tanaka received the B.E., M.E., and Ph.D. degrees all in Electrical Engineering, from Science University of Tokyo, in 1995, 1997 and 2000, respectively. In 2000, he joined the faculty of Science and Technology, Science University of Tokyo. In 2002, he joined Communication Research Laboratory. Currently, he is a senior researcher of Security Fundamentals Group in National Institute of Information and Communications Technology. He was awarded one of the SCIS2003 paper prizes.

Akihiro Yamamura received his Ph.D. from the University of Nebraska-Lincoln in 1996. He is currently the Group Leader of Security Fundamentals Group in National Institute of Information and Communications Technology. He is presently engaged in research on information security and algorithmic problems on algeraic systems.