Dynamic Cryptographic Hash Functions

William R. Speirs II and Samuel S. Wagstaff, Jr.

Center for Education and Research in Information Assurance and Security (CERIAS)
Department of Computer Sciences, Purdue University

Abstract. We present the *dynamic cryptographic hash function*, a new type of hash function which takes two parameters instead of one. The additional parameter, the security parameter, specifies the internal workings and size of the digest produced. We provide a formal definitions for a dynamic cryptographic hash function and for the traditional security properties, modified for dynamic hash functions. Two additional properties, *security parameter collision resistance* and *digest resistance*, are also defined. The additional properties are motivated by scenarios where a dynamic hash functions more cleanly provides a solution to a typical cryptographic problem.

Keywords: Hash function, dynamic hash function, security parameter.

1 Introduction

In this paper we introduce a new type of cryptographic hash function that is a natural extension of traditional hash functions. We call this new type of hash function a *dynamic hash function*. Dynamic hash functions take a second input, besides the message, that specifies the level of security required from the function. By changing the security parameter the function dynamically changes the way a digest is computed, hence the reason for the name. The change might be as simple as changing the number of rounds used for processing a message block or the size of the output of the function.

Requiring a hash function to have a security parameter is advantageous for a number of reasons. First, it allows designers to more easily test functions by scaling down the number of rounds to a manageable number and then launching attacks against this reduced version. If attacks can be launched against a reduced version then there is the possibility of extending the attack to the full version of the function. This technique has been used numerous times in the past: [1], [4], [3], etc.

Second, if the security parameter relates to the size of the digest, as suggested in this paper, then a single dynamic hash function can be used in a number of different applications without modification. For example, the hash-then-sign paradigm that is commonly used in a number of protocols. If the signing algorithm has the ability to vary the number of bits used for the key (like RSA, ElGamal, or Pohlig-Hellman), then why should the hash function used be restricted to a fixed size digest? If a user requires increased security by selecting a

larger key size, the size of the digest should also increase to provide potentially increased security.[1]

Finally, using a dynamic hash function in a protocol makes implementation much easier when it is mandated that the size of the digest be increased. Instead of being forced to rewrite software to include the option for a new hash function, the protocol can be designed from the start with a field to specify the size of the digest. Several protocols need to be reimplemented now that attacks on MD5 [2, 7] have became prevalent. The same type of reimplemention is currently needed as systems migrate from SHA-1 to SHA-256 or SHA-512. Using a dynamic hash function and specifying in the design that the digest can change size makes reimplemention easier when attacks are discovered; the security parameter used only needs to be changed.

1.1 The Contributions of This Paper

In this paper we formally define a dynamic cryptographic hash function and the security requirements for such a function. This definition generalizes the definition for a traditional cryptographic hash function. The security requirements of a traditional cryptographic hash function are extended to a dynamic cryptographic hash function and new security requirements are provided that do not have analogs in the traditional setting. These new requirements allow dynamic hash functions to be used in new and interesting ways.

1.2 Overview of this Paper

This paper begins by presenting the dynamic cryptographic hash function and formally defining the security properties a dynamic hash function needs to be considered cryptographically secure. We also define additional security properties unique to dynamic cryptographic hash functions. Scenarios where these new properties are useful are provided.

2 Definition of a Dynamic Cryptographic Hash Function

A dynamic cryptographic hash function is the same as a traditional cryptographic hash function except that a security parameter is provided which can affect how the function works internally and the size of the output. The definition of a traditional hash function is modified to define a dynamic hash function as follows. A dynamic hash function is then defined as follows.

Definition 1 (Dynamic Hash Function). *Let d, λ and υ be monotone increasing functions from $\mathbb{N} \to \mathbb{N}$ such that $d(s) > 0$ and $0 < \lambda(l) \leq \upsilon(l)$. A dynamic hash function is a function*

$$H : \Sigma^* \times \mathbb{N} \to \Sigma^* \cup \{\text{“undefined”}\},$$

[1] Increasing the size of a key or digest of a hash function does not always relate to increased security.

such that when $|M| = l$ *and* $\lambda(l) \leq s \leq \upsilon(l)$, $|H(M,s)| = d(s)$. *If it is not true that* $\lambda(l) \leq s \leq \upsilon(l)$, *then* $H(M,s)$ *is undefined. The functions* λ, υ *and* d *all run in polynomial time.*

One should note that a dynamic cryptographic hash function creates a family of traditional cryptographic hash functions each meeting the requirements of a traditional cryptographic hash function. Since a dynamic cryptographic hash function takes a second parameter that can potentially vary the digest's size and how the function is computed, the traditional properties must be modified appropriately.

Before defining preimage and collision resistance for a dynamic hash function, a dynamic hash function family must be defined. The reason for this is that for collision resistance one could imagine a probabilistic polynomial time algorithm which has a table of (message, security parameter, digest) triples encoded in the algorithm for a specific dynamic hash function. When asked to compute either the preimage of a given digest, or a collision for the hash function, the algorithm would search the table, in polynomial time, for either a preimage or a collision and output accordingly [5]. However, defining a function family as an infinite family of finite sets of hash functions prevents such an algorithm from succeeding for all hash functions in the family, because there are infinitely many.

Definition 2 (Dynamic Hash Function Family). *A dynamic hash function family* $\mathcal{H}$ *is an infinite set of functions where each function in the family is indexed by a key* K, *and each function is of the form*

$$H_K : \Sigma^l \times \mathbb{N} \rightarrow \Sigma^* \cup \{\text{“undefined”}\},$$

so that $\forall s \in [\lambda(l), \upsilon(l)]$, $H_K(\cdot, s) : \Sigma^l \rightarrow \Sigma^{d(s)}$.

The same three requirements imposed on a traditional hash function family are also imposed on the dynamic hash function family $\mathcal{H}$.

1. $\mathcal{H}$ is accessible, that is, there is a probabilistic polynomial time algorithm, that on input K outputs an instance H_K.
2. $D = \Sigma^l$ is samplable, that is, there is a probabilistic polynomial time algorithm, that selects an element uniformly from D.
3. H_K is polynomial time computable, that is, there is a polynomial time algorithm (polynomial in s and in $|M|$) that computes $H_K(M,s)$.

2.1 Traditional Properties for Dynamic Hash Functions

One should note that a dynamic cryptographic hash function creates a family of traditional cryptographic hash functions, each possessing the required security properties of a traditional cryptographic hash function.

Because a dynamic cryptographic hash function takes a second parameter that determines the digest's size and how the function is computed, the definitions for the traditional properties must be modified appropriately. Preimage

resistance, second preimage resistance, and collision resistance are defined for dynamic hash functions in Definition 3. The experiments are the same as for traditional hash functions except a dynamic hash function is used instead. Figures 1, 2, and 3 define the experiments for these properties for dynamic hash functions.

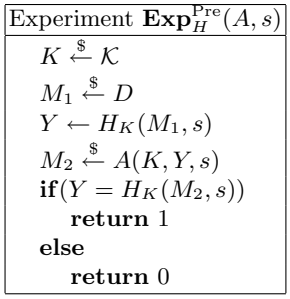

Fig. 1. Preimage resistance experiment for dynamic hash functions.

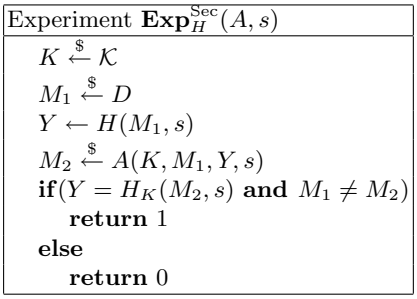

Fig. 2. Second preimage resistance experiment for dynamic hash functions.

Definition 3. *A dynamic hash function family $\mathcal{H}$ is (t, ϵ)-preimage resistant if there exists an s so that $\boldsymbol{Adv}_H^{Pre}(A, s) < \epsilon$ for all adversaries A with a running time less than t, where*

$$\boldsymbol{Adv}_H^{Pre}(A, s) = Pr\big[\boldsymbol{Exp}_H^{Pre}(A, s) = 1\big].$$

A dynamic hash function family $\mathcal{H}$ is (t, ϵ)-second preimage resistant if there exists an s so that $\boldsymbol{Adv}_H^{Sec}(A, s) < \epsilon$ for all adversaries A with a running time less than t, where

$$\boldsymbol{Adv}_H^{Sec}(A, s) = Pr\big[\boldsymbol{Exp}_H^{Sec}(A, s) = 1\big].$$

Experiment $\mathbf{Exp}_H^{\mathrm{Col}}(A, s)$

$K \xleftarrow{\$} \mathcal{K}$
$(M_1, M_2) \xleftarrow{\$} A(K, s)$
if $(H_K(M_1, s) = H_K(M_2, s)$ **and** $M_1 \neq M_2)$
  return 1
else
  return 0

Fig. 3. Collision resistance experiment.

A dynamic hash function family $\mathcal{H}$ is (t, ϵ)-collision resistant if there exists an s so that $\boldsymbol{Adv}_H^{Col}(A, s) < \epsilon$ for all adversaries A with a running time less than t, where

$$\boldsymbol{Adv}_H^{Col}(A, s) = Pr\big[\boldsymbol{Exp}_H^{Col}(A, s) = 1\big].$$

For each experiment the probability is taken over all $K \in \mathcal{K}$, $M \in \mathcal{M}$, and the random choices of A.

3 Dynamic Versions of the Traditional Properties

The traditional properties defined in Section 2.1 require that the same security parameter is used for the digests being compared. Because the security parameter dictates the way the digest is computed, properties must be defined for digests created using different security parameters. These properties are defined by allowing the adversary to choose the value of a second security parameter. While the security parameter also dictates the size of the digest, it does not make sense to compare digests of different sizes. This restricts the adversary to those security parameters that keep the digests the same size.

Definition 4 (Dynamic Preimage Resistance). *A dynamic hash function family $\mathcal{H}$ is (t, ϵ)-dynamic preimage resistant if $\boldsymbol{Adv}_H^{dPre}(A) < \epsilon$ for all adversaries A with a running time less than t, where*

$$\boldsymbol{Adv}_H^{dPre}(A) = Pr\big[\boldsymbol{Exp}_H^{dPre}(A) = 1\big],$$

and the probability is taken over all $K \in \mathcal{K}$, $M \in \mathcal{M}$, $s \in [\lambda(l), \upsilon(l)]$, and the random choices of A.

Definition 4 states that the probability of an adversary successfully finding a message and a security parameter that will hash to the given digest is negligible.

Dynamic preimage resistance is needed in the following scenario where a dynamic hash function is used for the Linux password system rather than a traditional hash function. In Linux systems there are two files that dictate user

Experiment $\mathbf{Exp}_H^{\mathrm{dPre}}(A)$
$K \xleftarrow{\$} \mathcal{K}$
$M_1 \xleftarrow{\$} \mathcal{M}$
$s_1 \xleftarrow{\$} [\lambda(l), \upsilon(l)]$
$Y \leftarrow H_K(M_1, s_1)$
$(M_2, s_2) \xleftarrow{\$} A(K, Y, s_1)$
if$(Y = H_K(M_2, s_2)$ **and** $d(s_1) = d(s_2))$
return 1
else
return 0

Fig. 4. Dynamic preimage resistance experiment.

authentication: `passwd` and `shadow`. Instead of a traditional hash function, a dynamic hash function can be used with only slight modifications to the authentication system. The `passwd` file stores user name and security parameter pairs. The `shadow` file stores user name and digest pairs, where the digest is computed using the security parameter in the `passwd` file and the user's password. Everyone has read access to the `passwd` file and only the administrator has access to the `shadow` file.

Assume Alice is a user on the system with a password of P. The digest of her password is stored in the `shadow` file using security parameter s. Through a vulnerability in the system, Mallory is able to gain write access to the `passwd` file and read access to the `shadow` file. Mallory is now able to read Alice's digest from the `shadow` file and the security parameter from the `passwd` file.

If the dynamic hash function used on the system does not have dynamic preimage resistance, then Mallory can compute a password P' and security parameter s' such that $H(P, s) = H(P', s')$.[2] Because Mallory has write access to the `passwd` file, she can change Alice's security parameter from s to s'. Mallory can now use the new password P' to authenticate with the system as if she were Alice. When Mallory uses the user name "Alice" and the password P', the system reads the security parameter s' from the `passwd` file and computes the digest $H(P', s')$. The digest is then compared with the one stored in the `shadow` file, and Mallory is granted access to the system as Alice.

Definition 5 (Dynamic Second Preimage Restance). *A dynamic hash function family $\mathcal{H}$ is (t, ϵ)-dynamic second preimage resistant if $\boldsymbol{Adv}_H^{dSec}(A) < \epsilon$ for all adversaries A with a running time less than t, where*

$$\boldsymbol{Adv}_H^{dSec}(A) = Pr\big[\boldsymbol{Exp}_H^{dSec}(A) = 1\big],$$

and the probability is taken over all $K \in \mathcal{K}$, $M \in \mathcal{M}$, $s \in [\lambda(l), \upsilon(l)]$, and the random choices of A.

[2] Note that P may or may not be the same as P' and s may or may not be the same as s'.

Experiment $\mathbf{Exp}_H^{\mathrm{dSec}}(A)$

$K \xleftarrow{\$} \mathcal{K}$
$M_1 \xleftarrow{\$} \mathcal{M}$
$s_1 \xleftarrow{\$} [\lambda(l), \upsilon(l)]$
$Y \leftarrow H_K(M_1, s_1)$
$(M_2, s_2) \xleftarrow{\$} A(K, M_1, Y, s_1)$
if$(Y = H_K(M_2, s_2)$ **and** $M_1 \neq M_2$ **and** $d(s_1) = d(s_2))$
  return 1
else
  return 0

Fig. 5. Dynamic second preimage resistance experiment.

Definition 5 states that the probability of an adversary successfully finding a different message and a security parameter that will hash to the given digest is negligible.

Dynamic second preimage resistance is required in the following scenario where digests are compared manually and assumptions are made about the security parameter that is used. Assume a key server with a web interface and a protocol for downloading keys, such as a PGP key server. Users can search, by e-mail address, for another user's public key. The web page displays e-mail addresses and the corresponding digest of that user's public key when hashed with a dynamic hash function using security parameter s. When a user clicks on an e-mail address a file is downloaded that contains the public key and the security parameter used to compute the digest. The security parameter is included in the file to provide algorithm agility, one of the main reasons for using a dynamic hash function. If certain security parameters are found to be insecure in the future, the security parameter is change in the files and the website is updated to reflect the new digests. These changes can all be done without the user updating the encryption software on his or her computer.

Assume Mallory is able to control the files that are downloaded from the website, but has no control over the web pages that are generated when a search is performed. Alice visits the website in search of Bob's public key, K_{Bob}. She searches for Bob's e-mail address, finds it and clicks on the link to download his key. Instead of downloading the proper file, Mallory is able to intercept it and change it to a file she has precomputed. The new files contains her public key $K_{Mallory}$ and a different security parameter, s'. Alice's encryption program computes $H(K_{Mallory}, s')$ and displays this digest on the screen for Alice to check against the digest associated with Bob's e-mail address on the website.

If the dynamic hash function that was used in the following scenario does not have dynamic second preimage resistance, then the digest of Mallory's key and new security parameter will be the same as the one on the web-page: $H(K_{Mallory}, s') = H(K_{Bob}, s)$. Alice will think that she has Bob's public key after verifying the digest computed by her encryption program is the same as

the one on the website. Actually, the key that Alice has is Mallory's. Mallory can now read any message sent from Alice to Bob. Mallory can also re-encrypt the message with Bob's actual public key and send it to Bob as if nothing has happened. The same scenario can occur in reverse so that Mallory can intercept any message sent from Bob to Alice.

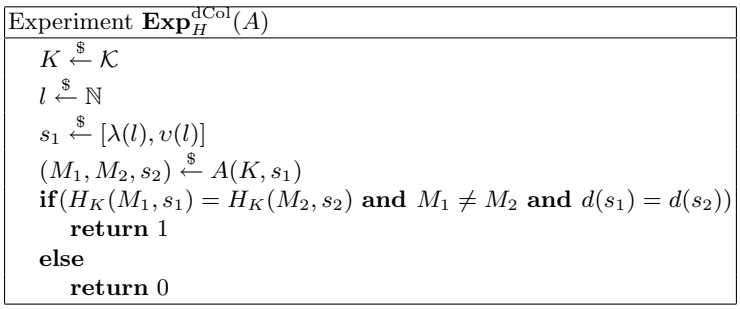

Fig. 6. Dynamic collision resistance experiment.

Definition 6 (Dynamic Collision Resistance). *A dynamic hash function family $\mathcal{H}$ is (t, ϵ)-dynamic collision resistant if $\boldsymbol{Adv}_H^{dCol}(A) < \epsilon$ for all adversaries A with a running time less than t, where*

$$\boldsymbol{Adv}_H^{dCol}(A) = Pr\big[\boldsymbol{Exp}_H^{dCol}(A) = 1\big],$$

and the probability is taken over all $K \in \mathcal{K}$, $M \in \mathcal{M}$, $s \in [\lambda(l), \upsilon(l)]$, and the random choices of A.

Definition 6 states that the probability of an adversary successfully finding two messages and a security parameter that will hash to the same digest using different security parameters is negligible.

Dynamic collision resistance is needed in the following contract signing scenario between Mallory and Bob. Mallory agrees to buy Bob's car for $1,000. Bob sends Mallory a security parameter s to use for computing the digest of the contract in which Mallory agrees to buy his car for $1,000. Mallory creates a contract C stating the price she will pay for the car and sends it to Bob. Bob computes the digest $H(C, s)$ and sends a digitally signed copy of the digest back to Mallory: $\textsc{Sign}(H(C, s))$. Before Mallory pays Bob for his car she contests Bob's version of the contract, C, with a trusted mediator Trent.

If the dynamic hash function used in this scenario does not have dynamic collision resistance, then Mallory is able produce a second contract C', which states that she will only pay Bob $10 for his car, and a second security parameter s' such that $H(C', s') = H(C, s)$. Because the two digests are the same, the two signatures will be the same: $\textsc{Sign}(H(C, s)) = \textsc{Sign}(H(C', s'))$. The trusted

mediator Trent will verify Bob's signature and compute the digest of the second contract C' with the second security parameter s'. The two digests will be the same, and Mallory will be awarded Bob's car by Trent for only $10.

The fact that only small changes are needed for these new security properties to be defined is not surprising. The dynamic versions of the traditional security properties are the generalized versions of the traditional properties. In fact, there is a natural implication between the dynamic versions of the traditional security properties and the traditional security properties.

4 Properties Without Traditional Analogs

While the properties in Section 3 are analogous to the traditional properties, there are two new properties, *security parameter collision resistance* and *digest resistance*, that are not. These properties are only applicable to dynamic hash functions because they directly relate to the function's ability to dynamically generate digests for different security parameters.

4.1 Security Parameter Collision Resistance

Security parameter collision resistance ensures that one cannot find a message that will result in the same digest for two different security parameters. It is the property of security parameter collision resistance that dictates dynamic hash functions are different from hash functions where the security parameter only affects the size of the digest. This property avoids having a weak hash function where if given $H(M, s_1) = H(M', s_2)$, it is probably the case that $M = M'$.

Experiment $\mathbf{Exp}_H^{\mathrm{PCol}}(A)$

$K \xleftarrow{\$} \mathcal{K}$
$l \xleftarrow{\$} \mathbb{N}$
$s_1 \xleftarrow{\$} [\lambda(l), \upsilon(l)]$
$(M, s_2) \xleftarrow{\$} A(K, s_1)$
if $(H_K(M, s_1) = H_K(M, s_2)$ **and** $s_1 \neq s_2$ **and** $d(s_1) = d(s_2))$
  return 1
else
  return 0

Fig. 7. Security parameter collision resistance experiment.

Definition 7 (Security Parameter Collision Resistance). *A dynamic hash function family $\mathcal{H}$ is (t, ϵ)-dynamic security parameter collision resistant if $\boldsymbol{Adv}_H^{PCol}(A) < \epsilon$ for all adversaries A with a running time less than t, where*

$$\boldsymbol{Adv}_H^{PCol}(A) = Pr\big[\boldsymbol{Exp}_H^{PCol}(A) = 1\big],$$

and the probability is taken over all $K \in \mathcal{K}$, $M \in \mathcal{M}$, $s \in [\lambda(l), \upsilon(l)]$, *and the random choices of* A.

Definition 7 states that the probability of an adversary successfully finding a message and a security parameter that will hash to the same digest as the same message and the given security parameter is negligible.

Dynamic hash functions construction without security parameter collision resistance are vulnerable when used in the following type of scenario. Assume a computer system with $r = |\mathcal{S}|$ levels of access. Each level has a security parameter associated with it. For simplicity the security parameters are the integers $\{1, 2, \ldots, r\}$. Let P be a password, l be one of the r levels, and SIGN be the signature of the computer system. When an account is created on the system, the administrator assigns the access level and the user picks a password. The system computes and sends to the user $\text{SIGN}(H(P, l))$. A user authenticates with the system by sending P, l, and $\text{SIGN}(H(P, l))$ to the system. The server checks that the signature and the digest are both valid. Assume Alice is given an account on the computer system at level one. Alice chooses the password P for her account and the system computes and sends to Alice $\text{SIGN}(H(P, 1))$.

If the dynamic hash function used in the computer system does not have security parameter collision resistance, then Alice can choose a higher level of access l' and a password P such that $H(P, 1) = H(P, l')$. Because the two digests are the same, the signatures will be the same: $\text{SIGN}(H(P, 1)) = \text{SIGN}(H(P, l'))$. Alice can now authenticate with the system by sending P, l' and $\text{SIGN}(H(P, l'))$. The system will check the signatures and the digest, and authenticate Alice at the new level. Alice is now able to operate at an increased level of access than the level intended by the administrator.

4.2 Digest Resistance

Digest resistance ensures that it is not easy to create one digest from another. This property is motivated by the following situation. Suppose Alice has created a secret document. She posts the digest and the security parameter used to compute the digest of her document on the Internet, staking her claim that she created the document. Bob argues that he is the original creator of the document. He has a digest using a different security parameter on his website which he claims proves he is the creator of the document. Carol, acting as a trusted mediator, has both Alice and Bob recompute the digest of their documents using a different security parameter that Carol chooses. Because the security parameter will change how the digest is computed, the procedure will allow Carol to determine if Alice and Bob actually have the same document without revealing to Carol what the two documents contain.

Definition 8 (Digest Resistance). *A dynamic hash function family* $\mathcal{H}$ *is* (t, ϵ)*-dynamic digest resistant if* $\boldsymbol{Adv}_H^{Dig}(A) < \epsilon$ *for all adversaries* A *with a running time less than* t, *where*

$$\boldsymbol{Adv}_H^{Dig}(A) = Pr\big[\boldsymbol{Exp}_H^{Dig}(A) = 1\big],$$

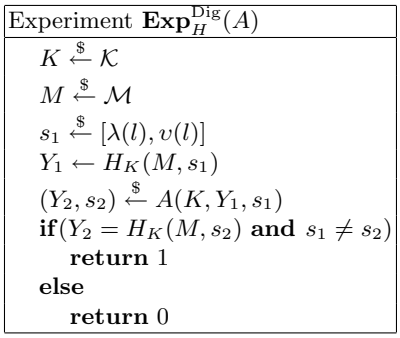

Fig. 8. Digest resistance experiment.

and the probability is taken over all $K \in \mathcal{K}$, $M \in \mathcal{M}$, $s \in [\lambda(l), \upsilon(l)]$, and the random choices of A.

Definition 8 states that the probability of an adversary successfully finding the digest of an unknown message and a security parameter, given a digest of the same message with a different security parameter, is negligible.

This property ensures, among other things, that a dynamic cryptographic hash function is not constructed by concatenating or truncating a standard cryptographic hash function. Another motivation for this property is to protect against attacks that would leverage the ability to reduce the digest space to a manageable size and then launch another attack against the hash function. For example, it is insecure for the 50-bit digest of a message to be constructed from the 160-bit digest of the same message. If this property is not ensured the following attack could be launched against a password authentication scheme.

Assume that a two computer system stores the hash of users' passwords in a central database. The security parameter for system A produces a 160-bit digest and the security parameter for system B produces a 100-bit digest. The only terminals to log into either system are connected to the central database via a secure communication link. The authentication is done by having the terminal compute the digest of the password and send it to the database for comparison. Suppose that Alice has accounts on both systems and that Bob is able to discover the hash of Alice's password for system A.

If the dynamic hash function used to compute the digest a of user's passwords does not have digest resistance, then Bob can compute Alice's 100-bit digest from her 160-bit digest. Because the digest and user name are sent from the secure terminal to the server, Bob can send the 100-bit digest and the user name "Alice" to authenticate with the system B without ever knowing Alice's password.

5 Conclusion

This paper introduced a new type of cryptographic hash function, the dynamic cryptographic hash function. Dynamic cryptographic hash functions are different from traditional cryptographic hash functions in that a security parameter dictates how the digest is computed. The goal of a dynamic cryptographic hash function is essentially the same as a traditional hash function: provide a cryptographically secure hash function with respect to the properties of preimage resistance, second preimage resistance, and collision resistance. However, because an adversary potentially has control of the way in which the digest is computed, additional security properties are required to ensure a dynamic cryptographic hash function is secure.

Security parameter collision resistance and digest resistance, were introduced and formally defined in Section 4.2. These new properties prevent an adversary from gaining an advantage in attacking the dynamic hash function by controlling the security parameter. For example, digest resistance prevents an attacker from creating a smaller version of the digest for a message, and then searching for a collision for the larger digest. Security parameter collision resistance and digest resistance are what differentiate dynamic cryptographic hash functions from previous cryptographic hash functions that simply create variable size digests.

References

1. Bert den Boer and Antoon Bosselaers. An attack on the last two rounds of MD4. In Joan Feigenbaum, editor, *Advances in Cryptology - CRYPTO '91, 11th Annual International Cryptology Conference*, Lecture Notes in Computer Science, pages 194–203, Santa Barbara, California, USA, August 1991. Springer.
2. Bert den Boer and Antoon Bosselaers. Collisions for the compressin function of MD5. In Tor Helleseth, editor, *Advances in Cryptology - EUROCRYPT '93, Workshop on the Theory and Application of Cryptographic Techniques*, Lecture Notes in Computer Science, pages 293–304, Lofthus, Norway, May 1993. Springer.
3. Hans Dobbertin. RIPEMD with two-round compress function is not collision-free. *Journal of Cryptology*, 10(1):51–70, 1997.
4. Hans Dobbertin. The first two rounds of MD4 are not one-way. In Serge Vaudenay, editor, *Fast Software Encryption, 5th International Workshop, FSE '98*, Lecture Notes in Computer Science, pages 284–292, Paris, France, March 1998. Springer.
5. Bart Preneel. E-Mail correspondence with Bart Preneel, December 2005.
6. William R. Speirs. *Dynamic Cryptographic Hash Functions*. PhD thesis, Purdue University, 2007.
7. Xiaoyun Wang and Hongbo Yu. How to break MD5 and other hash functions. In Ronald Cramer, editor, *Advances in Cryptology - EUROCRYPT 2005, 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques*, Lecture Notes in Computer Science, pages 19–35, Aarhus, Denmark, May 2005. Springer.

Appendix A Additional Experiments for Dynamic Hash Function Security Properties

Stronger Versions of the Traditional Properties

Experiment $\mathbf{Exp}_H^{\mathrm{sSec}}(A)$	Experiment $\mathbf{Exp}_H^{\mathrm{sCol}}(A)$

$K \xleftarrow{\$} \mathcal{K}$
$M_1 \xleftarrow{\$} D$
$(M_2, s) \xleftarrow{\$} A(K, M_1)$
if$(H_K(M_1, s) = H_K(M_2, s)$ **and** $M_1 \neq M_2)$
  return 1
else
  return 0
	$K \xleftarrow{\$} \mathcal{K}$
	$(M_1, M_2, s) \xleftarrow{\$} A(K)$
	if$(H_K(M_1, s) = H_K(M_2, s)$ **and** $M_1 \neq M_2)$
	  return 1
	else
	  return 0

Stronger Versions of the Dynamic Properties

Experiment $\mathbf{Exp}_H^{\mathrm{sdSec}}(A)$	Experiment $\mathbf{Exp}_H^{\mathrm{sdCol}}(A)$

$K \xleftarrow{\$} \mathcal{K}$
$M_1 \xleftarrow{\$} D$
$(M_2, s_1, s_2) \xleftarrow{\$} A(K, M_1)$
if$(H_K(M_1, s_1) = H_K(M_2, s_2)$
and $M_1 \neq M_2$ **and** $d(s_1) = d(s_2))$
  return 1
else
  return 0
	$K \xleftarrow{\$} \mathcal{K}$
	$(M_1, M_2, s_1, s_2) \xleftarrow{\$} A(K)$
	if$(H_K(M_1, s_1) = H_K(M_2, s_2)$
	and $M_1 \neq M_2$ **and** $d(s_1) = d(s_2))$
	  return 1
	else
	  return 0

Weaker Versions of the Dynamic Properties

Experiment $\mathbf{Exp}_H^{\mathrm{wdPre}}(A)$	Experiment $\mathbf{Exp}_H^{\mathrm{wdSec}}(A)$

$K \xleftarrow{\$} \mathcal{K}$
$M_1 \xleftarrow{\$} D$
$s_1 \xleftarrow{\$} [\lambda(l), \upsilon(l)]$
$s_2 \xleftarrow{\$} [\lambda(l), \upsilon(l)]$
$Y \leftarrow H_K(M_1, s_1)$
$(M_2) \xleftarrow{\$} A(K, Y, s_1, s_2)$
if$(Y = H_K(M_2, s_2)$ **and** $d(s_1) = d(s_2))$
  return 1
else
  return 0
	$K \xleftarrow{\$} \mathcal{K}$
	$M_1 \xleftarrow{\$} D$
	$s_1 \xleftarrow{\$} [\lambda(l), \upsilon(l)]$
	$s_2 \xleftarrow{\$} [\lambda(l), \upsilon(l)]$
	$(M_2) \xleftarrow{\$} A(K, M_1, s_1, s_2)$
	if$(H_K(M_1, s_1) = H_K(M_2, s_2)$
	and $M_1 \neq M_2$ **and** $d(s_1) = d(s_2))$
	  return 1
	else
	  return 0

Experiment $\mathbf{Exp}_H^{\text{wdCol}}(A)$
$K \xleftarrow{\$} \mathcal{K}$
$l \xleftarrow{\$} \mathbb{N}$
$s_1 \xleftarrow{\$} [\lambda(l), \upsilon(l)]$
$s_2 \xleftarrow{\$} [\lambda(l), \upsilon(l)]$
$(M_1, M_2) \xleftarrow{\$} A(K, s_1, s_2)$
if $(H_K(M_1, s_1) = H_K(M_2, s_2)$
and $M_1 \neq M_2$ **and** $d(s_1) = d(s_2))$
return 1
else
return 0

Weak and Strong Versions of Security Parameter Collision Resistance

Experiment $\mathbf{Exp}_H^{\text{wPCol}}(A)$	Experiment $\mathbf{Exp}_H^{\text{sPCol}}(A)$
$K \xleftarrow{\$} \mathcal{K}$	$K \xleftarrow{\$} \mathcal{K}$
$l \xleftarrow{\$} \mathbb{N}$	
$s_1 \xleftarrow{\$} [\lambda(l), \upsilon(l)]$	$(M, s_1, s_2) \xleftarrow{\$} A(K)$
$s_2 \xleftarrow{\$} [\lambda(l), \upsilon(l)]$	**if** $(H_K(M, s_1) = H_K(M, s_2)$
$(M) \xleftarrow{\$} A(K, s_1, s_2)$	**and** $d(s_1) = d(s_2))$
if $(H_K(M, s_1) = H_K(M, s_2)$	**return** 1
and $d(s_1) = d(s_2))$	**else**
return 1	**return** 0
else	
return 0	

Weak Version of Digest Resistance

Experiment $\mathbf{Exp}_H^{\text{wDig}}(A)$
$K \xleftarrow{\$} \mathcal{K}$
$M \xleftarrow{\$} \mathcal{M}$
$s_1 \xleftarrow{\$} [\lambda(l), \upsilon(l)]$
$s_2 \xleftarrow{\$} [\lambda(l), \upsilon(l)]$
$Y_1 \leftarrow H_K(M, s_1)$
$(Y_2) \xleftarrow{\$} A(K, Y_1, s_1, s_2)$
if $(Y_2 = H_K(M, s_2)$ **and** $s_1 \neq s_2)$
return 1
else
return 0