Hash Functions and Digital Signatures

Ivan Damgrd

PhD Crypto Course, 1999

1 Collision Intractable Hash Functions

A family of *collision intractable* hash functions $\mathcal{H}$ (CIHF) is a family of functions which for each natural number k contains a set of functions mapping from $\{0,1\}^{p(k)}$ to $\{0,1\}^k$, where $p(k) > k$ is some fixed polynomial. The following must be satisfied:

1. There exists a PPT H which on input 1^k outputs (a decsription of) $h \in \mathcal{H}$ such that $h : \{0,1\}^{p(k)} \rightarrow \{0,1\}^k$
2. There exists PPT B computing $h(x)$ from x and a description of h.
3. For any PPT A, The probability
$$Pr[x \neq y, h(x) = h(y) : h \leftarrow H(1^k); x, y \leftarrow A(h)]$$
is negligible in k.

Thus we require that even though collisions of course exist, they must be hard to find. It is easy to see that

Exercise

Any family $\mathcal{H}$ as above is a family of one-way functions.

But there is no relation known between existence of these functions and for instance one-way permutations. The most obvious use of such functions is for authentication: to compute short fingerprints of large pieces of data. The rationale is that the collision intractability property should enable us to authenticate the fingerprint in stead of the data itself, e.g. using a digital signature scheme. We shall see later that this indeed is indeed secure in some cases.

But a natural first question is whether we are constrained by the polynomial $p()$ specifying the input length. Fortunately, this is not the case:

1.0.1 Theorem 1

If CIHF $\mathcal{H}$ with any input length $p(k) > k$ exists, then for any polynomial $p'(k)$, a CIHF with input length $p'(k)$ exists.

The proof is by noting that clearly the assumption implies existence of CI functions f with input length $k+1$ (and output length k). Then a simple iterative construction allows us to build from this a CI function with any input length: Choose any k-bit x, let the input be the bit string $B = b_1, ..., b_{p'(k)}$, and define a new function

$$h'(B) = f(...f(f(x||b_1)||b_2)...)||b_{p'(k)})$$

One then shows that finding collisions for h' implies you can find collisions for f. This will work for the definition as stated. In practice, one would often want a function that does not take only inputs of length precisely $p'(k)$, but any input of length at most $p'(k)$. The iterated construction can handle this too, with extra technicalities added, to mark the start or end of a message.

Existence of CIHF's follows from several standard intractability assumptions. For instance if n is an RSA modulus and $g \in Z_n^*$ is fixed, but chosen randomly initially, then the function $x \rightarrow g^x \bmod n$ is CI if n is hard to factor: a collison on inputs x, y implies that $x - y$ is a multiple of the order of g. This may not mean that $\phi(n)$ divides $x - y$, but they will have a large factor in common with high probability. This can be used to factor n uisng standard methods similar to the ones showing that n can be factored given both the public and secret RSA exponent. Also constructions of CIHF's from discrete log problems exist.

For practical purposes, one often prefers more efficient ad hoc constructions, many of which are standardized. Examples are the SHA-1, MD5 and RIPEMD-160 hash functions. All of these are fixed functions, not families, outputting typically 160 bit results. The conjecture is that finding collisions cannot be done essentially better than by brute force search, i.e. using about $2^{160/2} = 2^{80}$ evaluations of the function.

Let Σ be a signature sheme. Let $\mathcal{H}(\Sigma)$ be the signature scheme where we generate keys by first generating keys for Σ (security parameter k), and then choose a function $h := H(1^k)$. We sign messages m by signing $h(m)$ in Σ. Then

Theorem 2

If $\mathcal{H}$ is a CIHF, and Σ is not existentially forgeable under chosen message attacks, then neither is $\mathcal{H}(\Sigma)$.

The proof is straightforward, and simply shows that a forger successful against $\mathcal{H}(\Sigma)$ must be able to break either $\mathcal{H}$ or Σ.

2 Universal One-Way Hash Functions

There exists a slightly weaker concept, which is nevertheless sufficient for many purposes where one would otherwise use CHIF's:

A family of Universal One-Way hash Functions (UOWHF) $\mathcal{U}$ is a family of functions which for each natural number k contains a set of functions mapping from $\{0,1\}^{p(k)}$ to $\{0,1\}^k$, where $p(k) > k$ is some fixed polynomial. The following must be satisfied:

1. There exists a PPT H which on input 1^k outputs (a decsription of) $h \in \mathcal{H}$ such that $h : \{0,1\}^{p(k)} \rightarrow \{0,1\}^k$
2. There exists PPT B computing $h(x)$ from x and a description of h.
3. For any PPT A, The probability

$$Pr[x \neq y, h(x) = h(y) : x \rightarrow A(1^k); h \leftarrow H(1^k); y \leftarrow A(h)]$$

is negligible in k.

Thus the only slight twist is that the adversary must tell us one of the inputs in a collision *before* seeing the function chosen. Thus, random collisions for h do not count, the adversary must find a collision involving x to win.

One can show, just like for CHIF's, that the polynomial $p()$ is not essential: UOWHF's can be composed without loosing the essential property, so again a family that just hashes $k+1$ to k bits is sufficient to get any polynomial input length.

The main advantage of UOWHF is that they seem to exist under much weaker assumptions:

Theorem 3

UOWHF's exist if one-way functions exist

We sketch the proof of a somewhat weaker result, namely that one-way permutations are sufficient: assume we have a fixed one-way permutation f. Let $\mathcal{G}$ be a class of (strongly) universal hash functions mapping $\{0,1\}^k$ to $\{0,1\}^{k-1}$. As a concrete example, we will think of the following example: a function in the class is described by two elements a, b in the field with 2^k elements. Then function $g_{a,b}(x) := ax + b|_{k-1}$, meaning that we compute $ax + b$ in the field and extract the least significant $k-1$ bits.

Then our UOWHF family members for output length $k-1$ are described by a function $g \in \mathcal{G}$, and our function maps $x \in \{0,1\}^k$ to $g(f(x))$.

Suppose PTT A breaks this construction, and we want to find the preimage under f of random value z. Then A first outputs x. We then choose a universal hash function g randomly such that $g(f(x)) = g(z)$. This is not necessarily possible efficiently for any universal class of hash functions, but is certainly easy with the particular example mentioned above. Because f is a permutation, z is a uniform k-bit string. This and universality of $\mathcal{G}$ implies that g will be random in $\mathcal{G}$. Therefore the success probability of the adversary is exactly the same in this reduction as in the real attack. Finally observe that since a random function from our example $\mathcal{G}$ is 2 to 1 except with negligible probability (only failure is if $a = 0$), A's only chance of finding a collision is to output a preimage of z.

3 Digital Signatures from Hash Functions

A rather stupid looking signature scheme is easy to build based on any one-way function $f : X \rightarrow Y$. Choose x_0, x_1 at random in X, let this be the secret key, compute $(y_0, y_1) = (f(x_0), f(x_1))$, and let this be the public key.

Now we can sign one bit once: the signature of bit b is x_b, which can be verified by just evaluating the function and comparing to y_b. This of course only works once: if the next bit to sign is different from the first, signing it in the same way implies no secrets are left in the system. On the other hand it does in fact satisfy the definition of secure signatures, provided f is one-way and the adversary is constrained to asking for the signature of one one-bit message.

We can expand this trivially to a one-time signature system where one message of many bits can be signed: we just make as many pairs of values as we have bits in the message. Let $X(t, R), Y(t, R)$ denote a set of t pairs of x's, resp. y's constructed as above from random input R, where R is just a long enough random bit string.

A crucial observation now is that if we have a CIHF, then we can authenticate *more* than t bits this way, by hashing them down to t bits first, this will be secure by Theorem 2. So this means that we can use a set $X(t, R), Y(t, R)$ of data, where $Y(t, R)$ is the public key to authenticate a message with $t/2$ bits *and* a new set $Y(t, R')$ by hashing it down to $t/2$ bits (where the signer also generates and saves the corresponding $X(t, R')$. This new set be used to authenticate the next message: if we include the authenticator for $Y(t, R')$ in the signature, the the verifier can first check the message against part of $Y(t, R')$, and then $Y(t, R')$ against part of $Y(t, R)$.

This can go on "polynomially for ever", and it is easy to show that an adversary successful against this system will need to either invert the one-way function or find collisions for the hashing.

In this system, the length of a signature grows linearly with the number of messages signed. However, if we use a set $Y(t, R)$ to authenticate *two* new sets of y's and the next message, then we can let our set of data grow in a tree like fashion, and it will follow immediately that the signature length now grows only logarithmically.

Essentially the same construction works with UOWHF's in stead, provided that a new function from the family is used in each node of the tree, and that the description of the function is signed along with the y-values in the node. This, together with Theorem 3, implies

Theorem 4

If one-way functions exist, then secure digital signature schemes exist (with public key length independent of the number and size of messages signed and signature size logarithmic in the number of signatures made)

In the way of practical constructions of signatures with provable security, it is known that one can build such schemes based on the RSA assumption:

Theorem 5

Under the RSA assumption, for any integers l, d, there exists a secure signature scheme with public key size depending only on the security parameter k, where l^d signatures can be made, of length $O(kd)$ bits each.

The trick is to build a very flat authentication tree, i.e. with outdegree l and depth d. By using the particularities of the RSA assumption, such a tree can built without increasing the size of the nodes in the tree (unlike the case of general one-way functions).

4 Notes on References

The definition of CIHF's and Theorems 1 and 2 originate from

- Damgård: *Collision free hash functions and public key signature schemes*, Proc. of EuroCrypt 87, Springer Verlag LNCS.

The definition of UOWHF's, the proof that they follow from one-way permutations, and the construction of signatures based on UOWHF's is from

- Naor and Yung: *Universal one-way hash functins and their cryptographic applications*, Proc. of STOC 89.

Theorem 3 (and hence Theorem 4) is from

- Rompel: *One-way functions are necessary and sufficient for digital signatures* Proc. of STOC 90.

Finally, Theorem 5 is from

- Cramer and Damgård: *New generation of secure and practical RSA-based signatures*, Proc. of Crypto 96, Spinger Verlag LNCS.