One-Way Functions are Necessary and Sufficient for Secure Signatures

John Rompel*

Laboratory for Computer Science
Massachusetts Institute of Technology
Cambridge, MA 02139

1 Introduction

Much research in theoretical cryptography has been centered around finding the weakest possible cryptographic assumptions required to implement major primitives. Ever since Diffie and Hellman first suggested that modern cryptography be based on one-way functions (which are easy to compute, but hard to invert) and trapdoor functions (one-way functions which are, however, easy to invert given an associated secret), researchers have been busy trying to construct schemes that only require one of these general assumptions. For example, pseudo-random generators at first could only be constructed from a specific hard problem, such as discrete log [BM2]. Later it was shown how to construct pseudo-random generators given any one-way permutation [Y], and from other weak forms of one-way functions [Le, GKL]. Finally [ILL] proved that the existence of any one-way function was a necessary and sufficient condition for the existence of pseudo-random generators. Similarly, the existence of trapdoor permutations can be shown to be necessary and sufficient for secure encryption schemes.

However, progress on characterizing the requirements for secure digital signatures has been slower in coming. We will be interested in signature schemes which are secure against existential forgery under adaptive chosen message attacks. This notion of security, as well as the first construction of digital signatures secure in this sense was provided by [GMR]. Their scheme was based on factoring, or more generally, the existence of claw-free pairs. More recently, signatures based on any trapdoor permutation [BM1] and any one-way permutation [NY] have been constructed. In this paper, we present a method for constructing secure digital signatures given any one-way function. This is the best possible result, since a one-way function can be constructed from any secure signature scheme.

Our method follows [NY] in basing signatures on one-way hash functions: functions which compress their input, but have the property that even given one preimage, it is hard to find a different one. This in itself provides a weak form of signature; they show how to build secure signatures from this primitive. To complete their construction, they provide a simple method for constructing a one-way hash function from any one-way permutation.

However, arbitrary one-way functions must be managed much more carefully. The bulk of this paper is concerned with building a one-way hash function given any one-way function. First we show how to build a function which has the property that given one preimage, although most other preimages may be trivial to find, a small fraction must be hard. Then we show how to amplify this into a full one-way hash function. Our proof makes heavy usage of universal hash functions [CW].

2 Preliminaries

2.1 Signature Schemes

A signature scheme has the following components:

- A security parameter k, determining the security, running time, and lengths of messages.
- A message space, which we will assume to be strings in Σ^k, where $\Sigma = \{0,1\}$.
- A polynomial S_B called the signature bound

*supported in part by a National Science Foundation Graduate Fellowship, DARPA contract N00014-80-C-0622, and Air Force Grant AFSOR-86-0078

- A probabilistic poly-time key generation algorithm KG, which on input 1^k, outputs a public key PK, and a matching secret key SK.
- A probabilistic poly-time signing algorithm SP, which given a message m and a matching pair of keys <PK,SK>, outputs a signature of m with respect to PK.
- A poly-time verification algorithm V, which given S, m, and PK, tests whether or not S is a valid signature of m with respect to PK.

The notion of security we are interested in is security against existential forgery under adaptive chosen message attack. This means we will allow our adversary, the forger, to adaptively choose messages, and be supplied with signatures. We will consider him successful if, on his own, he is able to produce a valid signature for any message that we did not sign for him. We will be interested in signature schemes for which no polynomial-sized forger has a $1/p(k)$ chance of producing a forged message, for any polynomial p and for sufficiently large k. This is the strongest natural notion of security.

Lemma 1 *The existence of a secure signature scheme implies the existence of a one-way function.*

Proof: Let $f(1^k, x)$ run the KG algorithm on input 1^k and using random tape x, and output PK. Then f is a one-way function. Why? Assume we could invert f. Then given the public key PK, we would be able to obtain a secret key SK$'$ with the property that SK$'$ could generate signatures valid for PK. But this implies that the signature scheme is insecure, which is a contradiction. □

The [BM1] and [NY] signature schemes both use the notion of a window, due to Lamport [La]. In this limited signature scheme, the public file contains some one-way function f and some $\alpha_1^0, \alpha_1^1, \ldots, \alpha_m^0, \alpha_m^1$. To sign a bit b_i, the signer reveals $f^{-1}(\alpha_i^{b_i})$. The limitation is that only m bits can be signed. The scheme used by [BM1] is to have f be a trapdoor permutation, and at each stage sign a message and a new f. In this way, the same α's could be used over and over. The [NY] scheme modifies this slightly by, at each stage, signing the hash value of a new set of α's. They define a uniform family of one-way hash functions to to be such that no poly-time circuit exists which, first outputs an x, then, for a randomly selected function f from the family, can output a sibling of x—an $x' \neq x$ such that $f(x') = f(x)$. Using this type of hash function, they are assured that signing the hash value of the α's is sufficient for security.

Their construction of a one-way hash function is as follows: Given $f : \Sigma^n \rightarrow \Sigma^n$ a one-way permutation, pick a random $h : \Sigma^n \rightarrow \Sigma^{n-1}$ from a 2-universal family of hash functions. Their one-way hash function is $h \circ f$.

2.2 Universal Hash Functions

A k-universal family of hash functions [CW] from m-bit strings to l-bit strings is a collection $\{h_i\}$ of poly-time computable functions from Σ^m to Σ^l such that

1. $Pr[h_i(x_1) = y_1 \wedge \cdots \wedge h_i(x_k) = y_k] = 2^{-kl}$ for any $x_1, \ldots, x_k \in \Sigma^m$ and $y_1, \ldots, y_k \in \Sigma^l$, and
2. Given $j \leq k$ and any $x_1, \ldots, x_j \in \Sigma^m$ and $y_1, \ldots, y_j \in \Sigma^l$, it is possible in probabilistic polynomial-time to uniformly sample from the h_i such that $h_i(x_1) = y_1 \wedge \cdots \wedge h_i(x_j) = y_j$.

Note that the first condition is equivalent to saying that the random variables $\{h_i(x) | x \in \Sigma^m\}$ are uniform and k-wise independent. In light of this equivalence, the following lemma will prove useful in calculations involving hash functions:

Lemma 2 *Let random variables $X_1, \ldots, X_N$ be $2k$-wise independent and have $\Pr[X_i = 1] = p$ and $\Pr[X_i = 0] = 1 - p$. Furthermore, let $X = \sum_{i=1}^N X_i$. If $k = o(pN)$, then*

$$Pr[|X - pN| > a] < O((\frac{2kpN}{a^2})^k).$$

Proof Sketch: Let $Z = (X - pN)^{2k}$. Then,

$$\begin{aligned} \Pr[|X - pN| > a] &= \Pr[Z > a^{2k}] \\ &< E[Z]/a^{2k}. \end{aligned}$$

Note that, since Z is a degree $2k$ polynomial in the X_i's, E[Z] has the same value in any distribution with at least $2k$-wise independence. In particular, assuming full independence of the X_i's and using that $k = o(pN)$, we can compute a bound of $(2kpN)^k$ on $E[Z]$. The lemma follows from plugging in this bound for $E[Z]$. □

Finally, we give the standard construction of a k-universal family of hash functions from m-bit strings to m-bit strings. Let $F = \mathrm{GF}(2^m)$. For $a_0, \ldots, a_{k-1} \in F$, we let

$$h_{a_0,\ldots,a_{k-1}}(x) = a_0 + a_1 x + \cdots + a_{k-1} x^{k-1}.$$

The fact that this is a k-universal hash function follows from a simple interpolation argument. Finally note that the case $m < l$ can be handled by padding out the argument to the hash function; the case $m > l$ can be handled by stripping bits from the result of the hash function.

2.3 Notation

Function $f : \Sigma^k \rightarrow \Sigma^m$ is one-way in the non-uniform (uniform) model if f is poly-time computable, but there is not poly-sized family of circuits A (probabilistic poly-time algorithm A) such that

$$\Pr[f(A(f(x), 1^k)) = f(x)] > k^{-c}$$

for infinitely many k, where the probability is as x is chosen uniformly from Σ^k. In this paper, we will use the non-uniform model of security unless stated otherwise.

For a function f, we define the following:

The siblings of x under f, $S_f(x) = \{x' : f(x') = f(x)\}$.
$D_i(f) = \{x : 2^i \leq |S_f(x)| < 2^{i+1}\}$.
$R_i(f) = f(D_i(f)) = \{y : 2^i \leq |f^{-1}(y)| < 2^{i+1}\}$.

We will denote the concatenation of strings x and y by $x \cdot y$. We will denote the composition of functions f and g by $f \circ g$, i.e. $(f \circ g)(x) = f(g(x))$.

3 Constructing a One-way Hash Function

3.1 Overview

In this section we will show how to, given any one-way function, create a family of one-way hash functions. This will imply that we can construct, given any one-way function, a signature scheme secure against existential forgery under adaptive chosen message attacks.

Starting with our original one-way function f, we construct a series of functions, each one closer to our goal of a one-way hash function. In Section 3.2, we construct a function f_1 with the property that, although most siblings under f_1 may be easy to find, a non-negligible fraction are provably hard to find. Next, in Section 3.3, we construct a function f_2 such that most siblings under f_2 are provably hard to find. Then, in Section 3.4, we construct a function f_3 with the property that almost all the time, it is hard to find any sibling. This function is length-increasing, so in Section 3.5 we construct from it a function f_4 which is sibling-hard and length-decreasing. Finally, in Section 3.6, we give a family of one-way hash functions.

3.2 Making Some Siblings Hard

The big cryptographic property of a one-way hash function is that it is hard to find a sibling of any domain element. Naor and Yung start with a one-way permutation, which trivially has this property, and thus only had to show how to make the function compress as well.

We are not so fortunate. Consider, for example, the function $f(< x, y >) =< g(x), 1^k >$, where x and y are k-bit strings, and g is some one-way permutation. It it easy to show that f is one-way, but that each element of the domain has an exponential number of siblings, all of which are trivial to find. Nevertheless, in this section, we will show how to take *any* one-way function and convert it into one for which we can prove that at least some (a non-negligible fraction) of the siblings are hard to find.

We begin by putting our one-way function into a normal form. Assume we are given a function $f : \Sigma^m \rightarrow \Sigma^m$ which is one way. Let $l(m) = \lceil \log m \rceil$ and $k(m) = 2^{l(m)}$. Let $n = m + 4k(m) + l(m) + 2$. We construct the function $f_0 : \Sigma^n \rightarrow \Sigma^n$ such that for $x \in \Sigma^m$, $y \in \Sigma^{4k(m)}$, and $i \in \Sigma^{\log 4k(m)}$, $f_0(x \cdot y \cdot i) = f(x) \cdot (y \wedge (1^i \cdot 0^{4k(m)-i}) \cdot i$, where the $\wedge$ is bitwise AND. The reason for constructing f_0 is that its preimage structure is largely known.

Lemma 3 *$|D_j(f_0)|$, as a function of j, is increasing in the range 0 to m, flat and equal to $2^n/4k(m)$ in the range m to $4k(m)$, and decreasing in the range $4k(m)$ to $m + 4k(m)$.*

Proof:

$$\begin{aligned} |D_j(f_0)| &= \sum_{i=0}^{4k(m)-1} 2^{4k(m)} D_{j-i}(f) \\ &= 2^{4k(m)} \sum_{i=\max(0, j-4k(m)+1)}^{\min(m,j)} D_i(f). \end{aligned}$$

From this form, one can easily observe the monotone nature of $|D_j(f_0)|$ for small and large j. For $m \leq j < 4k(m)$, we note that

$$\begin{aligned} |D_j(f_0)| &= 2^{4k(m)} \sum_{i=0}^{m} |D_i(f)| \\ &= 2^{4k(m)+m} \\ &= 2^n/4k(m). \square \end{aligned}$$

Now we can construct a function f_1 such that a non-negligible fraction of the siblings are hard to find. Let h_1 be randomly chosen from an n-universal family of hash functions mapping $(n/2 + \log n)$-bit strings to n-bit strings. Let h_2 be randomly chosen from an n-universal family of hash functions mapping n-bit strings to $(n/2 - 2\log n)$-bit strings. Then we let $f_1 = h_2 \circ f_0 \circ h_1$ (f_1 maps $\Sigma^{n/2+\log n}$ to $\Sigma^{n/2-2\log n}$).

Next we must define what we mean by a hard sibling. For any h_1, h_2, and for any $x \in \Sigma^{n/2-2\log n}$, we define the hard sibling set $H_{h_1,h_2}(x)$ to be the set of y's in $\Sigma^{n/2-2\log n}$ such that

1. $f_1(y) = f_1(x)$
2. $f_0(h_1(y)) \neq f_0(h_1(x))$
3. $h_1(y) \in D_{n/2}(f_0)$.

The first condition states that y is a sibling of x. We will not be able to prove any hardness results about elements which collide with x on h_1 or f_0, so we will simply declare them to be easy siblings. The second condition states that y is a sibling of x due to a collision on h_2, not on f_0 (or h_1). Finally, we give for technical reasons which will be apparent below the third condition on a sibling being hard. For convenience, we also define the set of easy siblings, $E_{h_1,h_2}(x) = S_{f_1}(x) - H_{h_1,h_2}(x)$.

Now we can give two lemmas which make precise the hardness of f_1. The first is that hard siblings are actually hard.

Lemma 4 *If there exists a poly-sized family of circuits $\{A_n\}$ such that for an infinite number of n of the form $m+4k(m)+l(m)+2$, there exists an x of length $n/2+\log n$, such that the probability over h_1 and h_2 that A outputs an element of $H_{h_1,h_2}(x)$ is at least n^{-c}, then there is a poly-sized family of circuits which invert f_0 with non-negligible probability.*

Proof: Let $n = m+4k(m)+l(m)+2$ and $x \in \Sigma^{n/2+\log n}$ be such that A_n outputs a hard sibling of x with probability at least n^{-c}. We will show that there is a circuit B such that $\Pr[f_0(B(f_0(w))) = f_0(w)] \geq n^{-c-2}$ where the probability is taken over w picked uniformly from Σ^n. Algorithm B, given $z = f_0(w)$, picks h_1 at random, picks $r \in \Sigma^{n/2-2\log n}$ at random, then picks h_2 at random subject to the constraint that $h_2(f_0(h_1(x))) = h_2(z) = r$. It then lets y be the output of A_n given h_1, h_2, and x. Finally, if $f_0(h_1(y)) = z$, it outputs $h_1(y)$, otherwise it outputs "failure."

We will, in fact, only try to invert strings in $R_{n/2}(f_0)$. So, for now, let us assume that w is picked uniformly from $D_{n/2}(f_0)$. Under this assumption, let us first consider the probability that A_n outputs an element of $H_{h_1,h_2}(x)$. This probability can be written as

$$\sum_{h_1,h_2} \Pr[A_n \text{ outputs } y \in H_{h_1,h_2}(x) | h_1,h_2] \Pr[B \text{ picks } h_1,h_2].$$

The above summation would be at least n^{-c} if the probability that B selected h_1 and h_2 was exactly equal to the probability of picking h_1 and h_2 uniformly at random. What we will in fact show is that, for almost all h_1 and h_2, these two probabilities differ by at most a $(1 + 2^{-n/8})$ factor. This implies that the summation above is at least $n^{-c}(1 - 2^{-n/8})$.

Fix any h_1 and r. Let $G(h_2)$ contain the elements of $D_{n/2}(f_0)$ which $h_2 \circ f_0$ map to r, i.e.

$$G(h_2) = f_0^{-1}(h_2^{-1}(r)) \cap D_{n/2}(f_0).$$

The expected size of $G(h_2)$ is $n^2 2^{n/2}/4k(m)$. Using a slight variant of Lemma 2 we can show that for all but an exponentially vanishing fraction of the h_2's,

$$\frac{n^2 2^{n/2}}{4k(m)}(1 - 2^{-n/8}) \leq |G(h_2)| \leq \frac{n^2 2^{n/2}}{4k(m)}(1 + 2^{-n/8}).$$

This holds even if we first restrict to those h_2 which map $f_0(h_1(x))$ to r. Since for every w, the fraction of the h_2's for which $h_2(f_0(w)) = h_2(f_0(h_1(x))) = r$ is exactly $2^{-n-4\log n}$, we can think of B as uniformly picking a pair $< w, h_2 >$ from the set $\{< w, h_2 > | h_2(f_0(w)) = h_2(f_0(h_1(x))) = r\}$. So clearly, the probability of picking h_2 is exactly proportional to the size of $|G(h_2)|$, and is thus within a $(1+2^{-n/8})$ factor of uniform for almost all h_2.

Now let us assume that A_n does in fact output a hard sibling y. Then in particular, $h_1(y) \in G(h_2) - S_{f_0}(h_1(x))$. But given the information A_n sees (i.e. h_1, h_2, r, and x), w is just a random element of $G(h_2) - S_{f_0}(h_1(x))$. So

$$\begin{aligned} \Pr[f_0(h_1(y)) = f_0(w) = z] &\geq \frac{|S_{f_0}(w)|}{|G(h_2)|} \\ &\geq \frac{4k(m)}{n^2}(1 - 2^{-n/8}) \\ &\geq 1/n. \end{aligned}$$

So if w is picked uniformly from $D_{n/2}(f_0)$ then the probability B outputs "failure" is at most $1-1/n$. Since $D_{n/2}(f_0)$ accounts for more than a $1/n$ fraction of Σ^n, this implies that the probability the B outputs failure is at most $1 - 1/n^2$. □

Lemma 4 says that we cannot find hard siblings in polynomial time. The next lemma says that these hard siblings form a non-negligible fraction of the possible siblings.

Lemma 5 *Fix $x \in \Sigma^{n/2+\log n}$ and let X be a random variable (dependent on the random choice of h_1 and h_2) such that*

$$X = \log \frac{|S_{f_1}(x)|}{|E_{h_1,h_2}(x)|},$$

i.e. X is the logarithm of the ratio between the number of siblings and the number of easy siblings. Then the X takes on values in the range 0 to n and has expected value at least $\Omega(1/n)$.

Proof: Consider some $w \in \Sigma^{n/2+\log n}$. We will show that with constant probability, w has at most n^3 siblings, and at least n^2 of them are hard. Pick $x \in \Sigma^n$ and $z \in \Sigma^{n/2-2\log n}$ at random, and let $y = f_0(x)$. We will restrict our attention to hash functions h_1 which map

w to x and h_2 which map y to z. Note that picking x at random and then h_1 at random subject to $h_1(w) = x$ is the same as just picking h_1 at random, and we are similarly picking h_2 at random.

Consider any sibling w' and let $x' = h_1(w')$ and $y' = f_0(x')$. To show that the number of siblings is small, we consider 4 cases: First we count the number of w' such that $x' \in \cup_{i \leq n/2} D_i(f_0)$. Using standard hash function arguments, we can show that with overwhelming probability, the number of such x' mapped to z is at most $\frac{2}{3}n^2 2^{n/2}$, and thus with overwhelming probability, the number of w' mapped to such x' is at most $3n^3/4$. The second case is $x' \in D_i(f_0)$ for some i in the range $n/2 < i \leq n/2 + \frac{3}{2}\log n$. For each such i, a similar argument shows that, with overwhelming probability, there are at most $n^{11/4}$ w' mapped to such x'. The third case is $x' \in D_i(f_0)$ for some $i > n/2 + \frac{3}{2}\log n$. In this case, with probability at least $1 - 1/\sqrt{n}$, there are no such w' at all mapping to z, other than ones for which $y' = y$. Finally, we consider the case when $y' = y$. With probability at least 3/8, $|f_0^{-1}(y)| < 2^{n/2}$, in which case it is very unlikely that more than n such w' exist. Putting this all together, with probability at least 1/3, the number of siblings is at most $4n^3/5$.

Now consider the number of hard siblings. With overwhelming probability, at least $1.1n2^{n/2}$ elements of $D_{n/2}(f_0) - S_{f_0}(y)$ map to z, and thus with overwhelming probability, at least n^2 siblings of w map through $D_{n/2}(f_0) - S_{f_0}(y)$. These are exactly the hard siblings of w.

Therefore, with probability at least 1/3, X is at least $1/n$, so $E[X] \geq 1/3n$. □

More details of the hash function arguments in the above lemmas will appear in the full version of this paper.

3.3 Making Most Siblings Hard

In the previous section, we showed how to, given any one-way function, construct one for which some of the siblings are hard to find. In this section we will construct a function for which almost all siblings are hard to find.

Let $f_2 : \Sigma^{n^6+2n^5\log n} \rightarrow \Sigma^{n^6-4n^5\log n}$ be constructed be the function which runs $2n^5$ copies of f_1 in parallel *each with independently chosen* h_1 *and* h_2. More precisely, we pick $h_1^1, \ldots, h_1^{2n^5}$ and $h_2^1, \ldots, h_2^{2n^5}$ at random, and let

$$f_2(x^1 \cdot \ldots \cdot x^{2n^5}) = f_1^{h_1^1,h_2^1} \cdot \ldots \cdot f_1^{h_1^{2n^5},h_2^{2n^5}},$$

where $f_1^{h_1^i,h_2^i}$ is f_1 using hash functions h_1^i and h_2^i. To define our set of hard siblings, we let $E_{\vec{h_1},\vec{h_2}}(x^1 \cdot \ldots \cdot x^{2n^5})$ be

$$\{y^1 \cdot \ldots \cdot y^{2n^5} | y^i \in E_{h_1^i,h_2^i} \text{ for } 1 \leq i \leq 2n^5\},$$

and let $H_{\vec{h_1},\vec{h_2}}(x) = S_{f_2}(x) - E_{\vec{h_1},\vec{h_2}}(x)$, i.e. a sibling is easy if and only if it is easy in each component.

Again, we must show two facts, first that our notion of hardness is correct, and second that most siblings are in fact hard.

Lemma 6 *If there exists a poly-sized family of circuits A such that for an infinite number of n, there exists an x of length $n^6 + 2n^5 \log n$, such that the probability over $\vec{h_1}$ and $\vec{h_2}$ that A outputs an element of $H_{\vec{h_1},\vec{h_2}}(x)$ is at least n^{-c}, then there is a poly-sized family of circuits which invert f_0 with non-negligible probability.*

Proof: Given an algorithm for finding a hard sibling under f_2, we obtain an algorithm for finding a hard sibling of f_1 as follows. If $x = x_1 \cdot \ldots \cdot x_{2n^5}$, we will select an i at random between 1 and $2n^5$ and attempt to find a hard sibling for x_i. Given h_1^i and h_2^i picked at random, we pick the rest of $\vec{h_1}$ and $\vec{h_2}$ at random, run A to get $y = y_1 \cdot \ldots \cdot y_{2n^5}$. If y is a hard sibling of x under f_2, then some y_j must be a hard sibling of x_j under f_1. Since i was chosen at random, there is a $1/2n^5$ chance that $i = j$. Thus with probability at least $1/2n^{c+5}$, we output a hard sibling for x_i. By Lemma 4, this implies the existence of an inversion algorithm for f_0. □

Lemma 7 *Fix $x \in \Sigma^{n^6+2n^5\log n}$ and let X be a random variable (dependent on the random choice of $\vec{h_1}$ and $\vec{h_2}$) such that*

$$X = \log \frac{|S_{f_2}(x)|}{|E_{\vec{h_1},\vec{h_2}}(x)|}.$$

Then the probability the $X \leq \Theta(n^4)$ is at most $e^{-\Theta(n)}$. In other words, with all but an exponentially small probability, there is an exponential gap between the number of siblings and the number of easy siblings.

Proof: Because both easy siblings and all siblings of f_2 are just cross products of the respective sets from f_1, we get that X is the sum of $X_1, \ldots, X_{2n^5}$, chosen independently according to the distribution defined in Lemma 5. But by Lemma 5, each X_i is between 0 and n and has expected value at least $1/3n$. The lemma follows from applying Chernoff's bounds. □

3.4 Making All Siblings Hard

From the previous section, we now have a function where almost all the siblings are hard to find. Unfortunately, there still may be an exponential number of easy siblings for any element of the domain. In this section, we show how to use the exponential gap between easy

and hard siblings to construct a function where, except with an exponentially small probability, *all* siblings are hard.

Let us look more carefully at Lemma 7. We rewrite the random variable X as $Y - Z$, where random variable $Y = \log|S_{f_2}(x)|$ and random variable $Z = \log|E_{\vec{h_1},\vec{h_2}}(x)|$. Assume for now that we are told the values of $E[Y]$ and $E[Z]$ (we will remove this assumption in Section 3.6). Let $l = (E[Y] + E[Z])/2$. Then let h_3 be randomly chosen from an 2-universal family of hash functions mapping $(n^6 + 2n^5 \log n - l)$-bit strings to $(n^6 + 2n^5 \log n)$-bit strings. Let $f_3(x) = f_2(h_3(x))$. The intuition is that h_3 selects a very small subset of the domain of f_2. This subspace is so small that it will not contain any easy siblings, although it will contain many hard siblings. This intuition is captured more formally in the following lemma.

Lemma 8 *Fix $x \in \Sigma^{n^6+2n^5\log n-l}$. Then with probability at least $1 - e^{-\Theta(n)}$ (over the choice of the hash functions), $h_3(S_{f_3}(x) - \{x\}) \subseteq H_{\vec{h_1},\vec{h_2}}(x)$.*

Proof: Z is the sum of $Z_1, \ldots, Z_{2n^5}$, where each Z_i is the logarithm of the number of easy siblings of x_i and is between 0 and n. Then, by Chernoff bounds,

$$\Pr[|Z - E[Z]| > a] < e^{-a^2/4n^7}.$$

In particular, plugging in $a = (l - E[Z])/2 > n^4/6$, we get that the probability $Z > (E[Z] + l)/2$ is at most $e^{-n/144}$. Assuming that $Z \leq (E[Z] + l)/2$, The probability that $h_3^{-1}(E_{\vec{h_1},\vec{h_2}}(x)) \neq x$ is at most $|E_{\vec{h_1},\vec{h_2}}(x)|/2^l$, which assuming that $Z \leq (E[Z] + l)/2$, is at most $2^{-n^4/6}$. □

Given Lemma 8, it is trivial to show that any algorithm that outputs a sibling for f_3 some polynomial fraction of the time can be converted into one which outputs a sibling for f_2 with virtually the same probability (which in turn implies the existence of an inverter for f_0).

3.5 Compressing

We have finally achieved a function with the hard-sibling property that we want. However, there are still a couple of problems left to be solved. The most obvious is that, in our quest to get the hard-sibling property, we have created a length-increasing function. In particular, the function f_3 constructed in the previous section maps $(n^6 + 2n^5 \log n - l)$-bit strings to $(n^6 - 4n^5 \log n)$-bit strings. Since one can show that $l = \Theta(n^6)$, it is clear that f_3 expands its input, and by quite a bit. In fact, simply applying a randomly selected hash function h_4 mapping $(n^6 - 4n^5 \log n)$-bit strings to $(n^6 + 2n^5 \log n - l - n/50)$-bit strings to the result of f_3 will solve the problem. So let $f_4(x)$ be $h_4(f_3(x))$. Then we have the following lemma.

Lemma 9 *Fix $x \in \Sigma^{n^6+2n^5\log n-l}$. Then with probability at least $1 - e^{-\Theta(n)}$, h_4 induces no collisions with x, i.e. $S_{f_4}(x) = S_{f_3}(x)$.*

Proof: We will bound the size of the range of f_3 by $2^{n^6+2n^5\log n-l-n/40}$. Once we have established this fact, the lemma follows trivially. In principle, we would like to bound the range of f_3 by

$$\sum_{i=0}^{n^6+2n^5-l} 2^{-i}|D_i(f_3)|.$$

In fact, it will be more convenient to work with f_2. We will consider 2 cases.

First, consider $z \in \bigcup_{i>l} R_i(f_2)$. We will conservatively assume that all such z are in the range of f_3, i.e. if we expect to have z in the range, then just assume it is. We can bound the number of such z by

$$\sum_{i=l}^{n^6+2n^5} 2^{-i}|D_i(f_2)|.$$

Now we must bound $|D_i(f_2)|$. For all y, for random $\vec{h_1}$ and $\vec{h_2}$, $\log|S_{f_2}(y)|$ is just the random variable Y we have already considered. In particular, by Chernoff bounds, we can show

$$\Pr[|Y - E[Y]| > a] < e^{-a^2/4n^7}.$$

However, what we are interested in is closer to the case where the hash functions are fixed and y is chosen at random. To get the bound we desire, consider pairs $<y, <\vec{h_1}\vec{h_2}>>$. Using the bound above, we can show, for any ϵ, for all but an ϵ fraction of the hash functions, the fraction of y's with $|S_{f_2}(y)| < 2^{E[Y]-a}$ is at most $e^{-a^2/4n^7}/\epsilon$. In particular, fix $\epsilon = 2^{-n/100}$. This gives that

$$\begin{aligned}
&\sum_{i=l}^{n^6+2n^5} 2^{-i}|D_i(f_2)| \\
&\leq \sum_{i=l}^{n^6+2n^5} 2^{n^6+2n^5-i+n/100} e^{-(E[Y]-i)^2/4n^7} \\
&\leq \sum_{j=E[Y]-n^6-2n^5}^{E[Y]-l} 2^{n^6+2n^5-E[Y]+j+n/100} e^{-j^2/4n^7} \\
&= 2^{n^6+2n^5-E[Y]+n/100} \sum_{j=E[Y]-n^6-2n^5}^{E[Y]-l} 2^{j-j^2/3n^7}
\end{aligned}$$

$$\leq 2^{n^6+2n^5-E[Y]+n/100+1}2^{E[Y]-l-(E[Y]-l)^2/3n^7}$$
$$\leq 2^{n^6+2n^5-l+n/100+1}2^{(n^4/3)^2/3n^7}$$
$$\leq 2^{n^6+2n^5-l+n/40-1}.$$

The second case we consider is $z \in \bigcup_{i<l} R_i(f_2)$. Here, we will bound the number of such z in the range of f_3 by the number of x in the domain of f_3 which map to them. To do this, we first note that

$$\left|\bigcup_{i<l} D_i(f_2)\right| \leq \sum_{i=0}^{l} 2^{n^6+2n^5+n/100}e^{-(E[Y]-i)^2/4n^7}$$
$$\leq 2^{n^6+2n^5+n/100+1}e^{-(E[Y]-l)^2/4n^7}$$
$$\leq 2^{n^6+2n^5+n/100}e^{-(n^4/3)^2/4n^7}$$
$$\leq 2^{n^6+2n^5-n/35}.$$

Now we can apply Lemma 2 to bound the number of x in the domain of f_3 which map to $z \in \bigcup_{i<l} R_i(f_2)$ by $2^{n/6+2n^5-l-n/40-1}$ for almost all h_3.

Adding these two parts together, we get the desired bound on the range of f_3 for almost all hash functions, so h_4 is very unlikely to induce a collision. □

As an immediate corollary, we can show that any sibling finder for f_4 is a sibling finder for f_3.

3.6 Putting Things Together

Now that we have achieved some compression, we can run this scheme in parallel and series to achieve arbitrary amounts of compression. More specifically, we achieve a compression of at least a $(1 - 1/50n^5)$ factor at each stage, so if we start out with $O(n^8)$ copies in parallel, we can in $O(n^5 \log n)$ stages compress by a factor of $O(n^3)$. This is important, because there is still one last problem to remove. When constructing f_3, we assumed we knew the correct value of l. In fact, we know little about it. However, we need only know its value to within an additive $\Theta(n^4)$. So we can build a new function f_5 which hashes its input using $\Theta(n^2)$ different values for l and outputting all the hash values. By the above, we know that f_5 compresses by a factor of n. Since any sibling under f_5 is a sibling under each component hash function, a sibling finder for f_5 is automatically a sibling finder for whichever of the components has a good value for l. Thus f_5 is a one-way hash function. Summing up we get the following theorems.

Theorem 1 *Under the assumption that one-way functions exist, one-way hash functions exist.*

Theorem 2 *Under the assumption that one-way functions exist, there exists a signature scheme which is secure against existential forgery under adaptive chosen message attacks.*

Finally, we note that, although this paper has been mostly phrased in terms of the non-uniform model of security, our construction works equally well in the uniform model. Thus we get the following theorem.

Theorem 3 *Under the assumption that one-way functions in the uniform model exist, there exists a signature scheme which is secure against existential forgery under adaptive chosen message attacks by polynomial-time algorithms.*

4 Acknowledgments

I would like to thank Mihir Bellare for many helpful discussions on the subject of signature schemes and pseudorandomness. I would also like to thank him for patiently listening to early versions of this work and giving helpful comments.

References

[BM1] Bellare, M., and Micali, S., "How to Sign Given any Trapdoor Function", Proc. 20th STOC, 1988, pp. 32-42.

[BM2] Blum, M., and Micali, S., "How to Generate Cryptograpically Strong Sequences of Pseudo-Random Bits", SIAM J. Comp., **13:4** (Nov. 84), pp. 850-864.

[CW] Carter, J.L., and Wegman, M.N., "Universal Classes of Hash Functions", JCSS, **18** (1979), pp. 143-154.

[DH] Diffie, W., and Hellman, M., "New Directions in Cryptography", IEEE Trans. on Info. Theory, vol. IT-22, **6** (1976), pp 644-654.

[GKL] Goldreich, O., Krawczyk, H., and Luby, M., "On the Existence of Pseudorandom Generators", Proc. 29th FOCS, 1988, pp. 12-24.

[GMR] Goldwasser, S., Micali, S., and Rivest, R., "A Secure Digital Signature Scheme", SIAM J. Comp., **17:2**, 1988, pp. 281-308.

[ILL] Impagliazzo, R., Levin, L., and Luby, M., "Pseudorandom Generation from One-Way Functions", Proc. 21st STOC, 1989, pp. 12-24.

[La] Lamport, L., "Constructing Digital Signatures from One-Way Functions", SRI intl. CSL-98, Oct. 1979.

[Le] Levin, L., "One-Way Functions and Pseudorandom Generators", Proc. 17th STOC, 1985, pp. 363-365.

[NY] Naor, M., and Yung, M., "Universal One-Way Hash Functions and their Cryptographic Applications", Proc. 21st STOC, 1989, pp. 33-43.

[Y] Yao, A.C., "Theory and Applications of Trapdoor Functions", Proc. 23rd FOCS, 1982, pp. 80-91.