COS595D February 29, 2008

Lecture Notes 4: UOWHF from any One-Way Function

Professor: Boaz Barak *Scribe:Mohammad Mahmoody-Ghidary*

1 Introduction

Digital signatures are one of the main primitives in cryptography with many applications. Unfortunately the existence of digital signatures implies $\mathbf{P} \neq \mathbf{NP}$ and more strongly the existence of one-way functions (OWF). So with our current knowledge in complexity theory the best we can do is to construct schemes which are secure based on believable assumptions, and at best we would like to base the existence of signature schemes on the sole assumption of the existence of one-way function.

The first suggestion on how to construct signature schemes was in the paper of Diffie and Hellman [DH76]. The assumption they used for getting signatures was the existence of *trapdoor one-way functions*. A family of trapdoor functions has a generation algorithm which generates (the description of) a function f together with a trapdoor t. Then f can be computed efficiently, but without knowing t it is hard to invert f on a random image with more than negligible probability. In order to sign a message using trapdoor permutations, (roughly speaking) the signature for a message m will be $f^{-1}(m)$ using t (which is the signing key), and the correctness of the signature can be verified by only knowing how to compute f (which is the verification key).

Later there were constructions for trapdoor functions based on algebraic assumptions that RSA trapdoor permutation [RSA78] among them is quite well known. Then Naor and Yung [NY89] introduced the notion of universal one-way hash functions (UOWHF) and showed that its existence is enough for having signature schemes, and showed how to construct UOWHF from any one-way permutation. Finally Rompel [Rom90] showed how to construct UOWHF from any one-way function. Katz and Koo recently wrote the full proof of Rompel's scheme [KK08].

In the next section we see the formal definition of UOWHF and see a simple construction using RSA assumption (which gives us collision resistence —something stronger than just UOWHF). Then in Section 3 we will see the construction from one-way permutations and then finally in Section 4 we see the construction based on any OWF.

2 Universal one-way hash functions

Before defining UOWHF, let us have a look at its twin: collision resistent hash function (CRHF). A CRHF is a family of compressing functions which is efficiently computable, but it is hard to find a collision for it.

Definition 2.1. A *collision resistant hash function* (CRHF) is (a sequence of) families of functions H_n (for $n \in \mathbb{N}$) such that each $h \in H_n$ is a compressing function $f\colon \{0,1\}^n \to \{0,1\}^{n-1}$ and we have:

- **Sampling** There is a randomized poly-time algorithm S which $S(1^n)$ outputs some $h \in H_n$.
- **Evaluation** There is a polynomial time algorithm A such that given (the description of) h and $x \in \{0,1\}^n$ computes $h(x)$ in poly(n) time.
- **Collision Resistance (CR)** For any polynomial time algorithm B we have $\Pr[h(x) = h(y) \mid h = S(1^n), B(h) = (x,y)] \leq n^{-\omega(1)}$.

Simon [Sim98] showed that black-box reductions can not construct CRHF from OWF. But perhaps surprisingly if we relax the requirements a bit then suddenly everything changes. A UOWHF has a similar but weaker definition compared to that of CRHF (in the sense that a CRHF is a UOWHF as well). This time the adversary is given one specific point and she needs to find another point that collides to that point. So, a UOWHF has the same definition as a CRHF with the difference that the third condition is different:

- **Target Collision Resistance (TCR):** For any polynomial time algorithm B we have $\Pr[h(x) = h(y) \mid h = S(1^n), B(x,h) = y] \leq n^{-\omega(1)}$ for every $x \in \{0,1\}^n$.

It is easy to see that once we can compress one bit (for both CRHF and UOWHF), then we can hash $\{0,1\}^{\text{poly}(n)}$ to $\{0,1\}^n$ with composing the function many times, (or even hash $\{0,1\}^*$ to $\{0,1\}^n$ with a bit more careful composition).

Sometimes people use another condition for the collision resistance in the definition of UOWFHF:

- **TCR':** For any polynomial time algorithm B we have $\Pr_{x \leftarrow_R \{0,1\}^n}[h(x) = h(y) \mid h = S(1^n), B(x,h) = y] \leq n^{-\omega(1)}$.

Note that TCR property implies TCR', but it is easy to see that if H has property TCR', then the following UOWHF has property TCR:

- **Sampling:** Get $h = S_H(1^n)$ and $z \leftarrow_{\text{R}} \{0,1\}^n$ and output (h,z).
- **Evaluation:** Given the function (h,z) and input x, output $h(x \oplus z)$ where $\oplus$ is the bitwise exclusive or.

So in the following we might as well use the TCR' property instead of TCR whenever it is more convenient.

3 Constructing CRHF based on RSA assuption

In this section we show how to get CRHF assuming the RSA trapdoor permutation is one-way. The RSA trapdoor permutation is described by a number $N = pq$ where $p, q \in \mathbb{N}$ are prime numbers and also e, d where the greatest common divisor of e and $\varphi(N)$ is $(d, \varphi(N)) = 1$ and $ed = 1 \mod \varphi(N)$. For $x \in Z_N$ the function is $f(x) = x^e \mod N$ and $f^{-1}(x) = x^d \mod N$. The assumption is that if we generate (N, e, d) in some random way, then it is hard to invert $f(x)$ for a random x (where the security parameter is $n \approx \log(N)$). Before going over the construction and its security proof note that the RSA function has the properties that $f(uv) = f(u)f(v)$ and $f(1/w) = 1/f(w)$ (when $w \neq 0$). The hash function H_{RSA} based on the RSA assumption is the following:

- **Sampling:** Generate (N, e, d) for $N \approx 2^n$ for the RSA system which describes a permutation $f\colon Z_N \to Z_N$, and choose $a \leftarrow_{\mathrm{R}} Z_N$. Let $f_0(x) = f(x)$ and $f_1(x) = f(ax) = f(a)f(x)$ for $x \in Z_n$. The hash function is described by (f_0, f_1). Note that we did not use e and we only needed $f(a)$ rather than a itself. Note also that if $(a, N) = 1$ then f_0 and f_1 are both permutations over Z_N.
- **Evaluation:** Given $M = m_1 \dots m_{2n} \in \{0,1\}^{2n}$, the hashed value is: $h(M) = f_{m_{2n}}(\dots f_{m_1}(1)\dots)$.

Now we prove the TCR property.

Theorem 3.1. *Assuming RSA permutations are one-way, the family of hash functions H_{RSA} described above is collision resistant (and therefore target collision resistant as well).*

Proof. Suppose for the sake of contradiction that the adversary B breaks the CR property of H_{RSA}. Now we show how to get an adversary E who breaks the security of RSA. Given (N, e) and $f(a) = a^e \mod N$ for a random $a \in Z_N$, E wants to find a. Note that $f(a) = 0$ iff $a = 0$ and $(a, N) \neq 1$ iff $(f(a), N) \neq 1$. So if $f(a) = 0$, E outputs 0 and if $f(a) \neq 0, (f(a), N) \neq 1$ she can decompose N by taking $(f(a), N)$ and find d and compute a. So we might as well assume that $(a, N) = 1$.

At this point E samples $h \in H_{RSA}$ by using (N, e) and $f(a)$, and feeds h to B. Let $M \neq M'$ be the collision found by B where $M = m_1 \dots m_{2n}, M' = m'_1 \dots m'_{2n}$. Let i be the largest i such that $m_i \neq m'_i$ and let $x = f_{m_{i-1}}(\dots(f_{m_1}(1)\dots)$ and $x' = f_{m'_{i-1}}(\dots(f_{m'_1}(1)\dots)$ (we do not have necessarily $x \neq x'$). Because f_0 and f_1 are permutations, so is $f_{m_{2n}}(\dots f_{m_{i+1}}(\cdot)\dots) = f_{m'_{2n}}(\dots f_{m'_{i+1}}(\cdot)\dots)$ and because M, M' make a collision we have $f_{m_i}(x) = f_{m'_i}(x')$ which because $m_i \neq m'_i$ means that $f_0(x) = f_1(x')$ and we get $f(x) = f(ax')$. So we have $f(x) = f(a)f(x')$ and so $f(a) = f(x)/f(x') = f(x/x')$ and therefore $a = x/x'$. □

4 Construction based on one-way permutation

In this section we see the construction of UOWHF based on one-way permutations given in [NY89]. The construction uses (a variant of) a powerful tool proved to be very useful in cryptography: pairwise independent hash functions PIHF.

Definition 4.1. A PIHF $\mathcal{F}$ is a set of functions $\mathcal{F} = \{f \mid f\colon A \to B\}$ for sets A, B with the property that: For every $b_1, b_2 \in B$ where $a_1, a_2 \in A$ where $a_1 \neq a_2$ we have $\Pr_{f \leftarrow_{\mathrm{R}} \mathcal{F}}[f(a_1) = b_2, f(a_2) = b_2] = \frac{1}{|B|^2}$.

The definition generalizes naturally to 3-wise independet and n-wise independent hash functions.

It is easy to see that if we remove some elements from the set A or set B and look at the restriction of $\mathcal{F}$ on the remaining sets, it is still a PIHF. Now we see how to construct a PIHF $\mathcal{F}$ from $A = \{0,1\}^n$ to $B = \{0,1\}^m$ where $|\mathcal{F}| = 2^{2\max(m,n)}$ (and so a member of $\mathcal{F}$ can be described with $2\max(m, n)$ bits). We only show how to construct $\mathcal{F}$ for the case $m = n$ and for the general case we can first extend the smaller set between A and B to the size of $2^{\max(m,n)}$ and then remove the extra elements at the end.

We can look at A and B as $GF(2^n)$. Now we claim that $\mathcal{F} = \{f(x) = ux + v \mid u, v \in GF(2^n)\}$ is a PIHF. The reason is that for any fixed $a_1, a_2, b_1, b_2 \in GF(2^n)$ where $a_1 \neq a_2$, the system of equations $b_1 = ua_1 + v, b_2 = ua_2 + v$ over variables u, v is nonsingular (i.e. $1a_1 - 1a_2 \neq 0$) and therefore has a unique solution.

In order to get $h\colon \{0,1\}^n \to \{0,1\}^{n-1}$ we can remove the last bit of the image. More precisely, $GF(2^n)$ can be represented with binary vectors of length n such that the addition is just componentwise exclusive or, and in this representation we remove the last bit to get the smaller domain $\{0,1\}^{n-1}$. Let $\mathcal{F}_n^{n-1}$ be this specific PIHF mapping $\{0,1\}^n$ to $\{0,1\}^{n-1}$. It can be seen that if we restrict $u \neq 0$ in the constructions $\mathcal{F}$ and $\mathcal{F}_n^{n-1}$ above the result is not a PIHF (because $f \in \mathcal{F}$ is now always a permutation), but if $\mathcal{I}$ be the $\mathcal{F}_n^{n-1}$ restricted to $u \neq 0$ it can be seen that we still have the "weakly-pairwise" independence which says $\Pr_{f \leftarrow \mathcal{I}}[f(a_1) = f(a_2)]$ is independent of the pair $a_1 \neq a_2$ (more precisely it is $\frac{1}{2^n - 1}$ because removing the last bit is a 2-to1 function). This property holds for any PIHF as well. But $\mathcal{I}$ has the extra property (used below) that for any a_1 and $f \in \mathcal{I}$, there is a unique $a_2 \neq a_1$ such that $f(a_1) = f(a_2)$.

Now we see how to construct a UOWHF H mapping $\{0,1\}^n$ to $\{0,1\}^{n-1}$ from the one-way permutation $p\colon \{0,1\}^n \to \{0,1\}^n$ and almost-PIHF $\mathcal{I}$ mapping $\{0,1\}^n$ to $\{0,1\}^{n-1}$ described above.

Theorem 4.2. *Let $p\colon \{0,1\}^n \to \{0,1\}^n$ be a one-way permutation and $\mathcal{I}$ be the APIHF described above, then $H = \{h \mid h = f \circ p, f \in \mathcal{F}\}$ is a UOWHF mapping $\{0,1\}^n$ to $\{0,1\}^{n-1}$.*

Proof. The intuition is that given a fixed x_0, for a random $f \in \mathcal{I}$, the unique x for which $f(p(x)) = f(p(x'))$ is distributed randomly, and therefore $p(x)$ is also distributed at random, and finding x is like inverting the random point of $p(x)$ which is impossible.

More formally, suppose B is the adversary who finds a collision for fixed x_0 for a random $h \leftarrow_{\mathrm{R}} H$ with probability at least ϵ. Then we show how to construct an adversary A who inverts $p(x)$ for a random x with probability at least ϵ. The adversary A knows x_0 (and $y_0 = p(x_0)$) and has access to B.

Note that $\mathcal{I}$ has the weakly-pairwise property and it is also 2-to-1. So if we first choose a random $y \neq y_0$ and then choose a random $f \in \mathcal{I}$ conditioned on $f(y) = f(y_0)$ we would sample a uniform $f \in \mathcal{I}$.

Given $p(x) = y$ for a random $x \in \{0,1\}^n$, if $y = y_0$, A can simply output x_0. So we can assume that $x \neq x_0$. Now we will show that given a random $y \neq y'$ how to sample $f \in \mathcal{I}$ conditioned on $f(y) = f(y_0)$, and by the above discussion it is as if we are choosing a random $f \in \mathcal{I}$ and then choosing y to be the unique point where $f(y) = f(y_0)$. So, if we feed $x_0, h = f \circ g$ to B it finds the unique x where $h(x) = h(x_0)$ with probability at least ϵ which is what A wants: $f(x) = y$.

Now we only need to show how to sample a random $f \in \mathcal{I}$ conditioned on $f(y) = f(y_0)$ for $y \neq y_0$. We have to choose $u \neq 0$ and v such that $uy + v$ and $uy_0 + v$ differ only in their last bit (in the representation explained above) because they can not be equal on all the bits (otherwise $uy = uy_0$ and so $u = 0$). So, if $z \in GF(2^n)$ be the element which has representation $00 \cdots 001$, we need to have $(uy + v) - (uy_0 + v) = z$ (note that $+$ and $-$ are the same in $GF(2^n)$). So it means $uy - uy_0 = z$, or equivalently: $u = z/(y - y_0)$. So the set of $f \in \mathcal{I}$ conditioned on $f(y) = f(y_0)$ can be described by $u = z/(y - y_0)$

and an arbitrary v, and we can choose v at random.

□

5 Construction based on one-way function

Let f be a function operating on the domain D_f. For every $x \in D_f$ we define $\mathsf{Sib}_f(x) = \{y \mid f(y) = f(x)\}$. The construction from OWF has the following steps:

1. Getting weak UOWHF: Starting from OWF f we get a family of weak UOWHF H and will define $\mathsf{Hard}_h(x) \subset \mathsf{Sib}(x)$ such that:
   - For every PPT algorithm A and $x \in D_H$ we have $\Pr_{h \leftarrow_R H}[A(h,x) \in \mathsf{Hard}_h(x)] = \mathsf{neg}$.
   - $\mathsf{E}_{x,h}[\frac{|\mathsf{Hard}_h(x)|}{|\mathsf{Sib}_h(x)|}] \geq \frac{1}{\mathrm{poly}}$.
2. Amplifying hard sets: Using UOWHF from the previous step, we get H such that for every $x \in D_H$, with probability $1 - \mathsf{neg}$ over $h \leftarrow_R H$ we have that the set $\mathsf{Easy}_h(x) = \mathsf{Sib}_h(x) \backslash \mathsf{Hard}_h(x)$ has negligible relative size: $\frac{|\mathsf{Easy}_h(x)|}{|\mathsf{Hard}_h(x)|} = \mathsf{neg}$.
3. Making all siblings hard: Then we get UOWHF h such that for every $x \in D_H$, with probability $1 - \mathsf{neg}$ over $h \leftarrow_R H$ we have $\mathsf{Hard}_h(x) = \mathsf{Sib}_h(x)$.
4. Shrinking the length: We change H such that its output has length less than its input.

The main step is the first one and the others are rather straightforward.

5.1 Getting weak UOWHF

Let $f\colon \{0,1\}^n \to \{0,1\}^n$ be a OWF. We can divide its domain into sets on which f is almost regular: For $0 \leq i \leq n$ let $C_i = \{x \mid 2^i \leq |\mathsf{Sib}_f(x)| < 2^{i+1}\}$. By pigeonhole principle, (since if $C_n \neq \emptyset$ the function is not one-way) there is k such that $|C_k| \geq 2^n/n$. By throwing away some elements from C_k we can get a set C which the restriction of f on C is 2^k regular and $|C| \geq \frac{2^n}{2n}$. We assume that n is a power of two, and so if $|f(C)| = q$ then $2^k q \geq \frac{2^n}{2n}$ and we can throw away more elements from C to keep 2^k regularity of $f(C)$ and have $|C| = \frac{2^n}{2n} = 2^{n-\log(2n)}$ and $|f(C)| = 2^{n-k-\log(2n)}$ exactly. C will be such set in the following.

For now, we assume that we know the value of k (the regularity of f over C), and we will take care of this issue later. Our weak UOWHF of this step uses two PIHF's: $P_i = \{h_i \mid h_i\colon \{0,1\}^{n-k-\log(2n)} \to \{0,1\}^n\}$, $P_o = \{h_o \mid h_o\colon \{0,1\}^n \to \{0,1\}^{n-k-\Delta}\}$ where $\Delta = 10\log(n)$.[1] The weak UOWHF family H is defined as $H = \{h = h_o \circ f \circ h_i \mid h_o \in P_o, h_i \in P_i\}$.

For any $x \in D_H = \{0,1\}^{n-k-\log(2n)}$ and $h = h_o \circ f \circ h_i \in H$, if $h_i(x) \imath C$ then we define $\mathsf{Hard}_h(x) = \{y \mid h(x) = h(y), h_i(y) \in C, f(h_i(y)) \neq f(h_i(x))\}$, and if $h_i(x) \notin C$ we define $\mathsf{Hard}_h(x) = \emptyset$. We first prove that it is in fact hard for every $x \in D_H$ to find a member of $\mathsf{Hard}_h(x)$ for a random h and then show that on average there are noticeable fraction of them among $\mathsf{Sib}_h(x)$.

[1] Note that we can assume w.l.o.g that $|f(C)| = 2^{n-k-\log(2n)} = n^{\omega(1)}$, because otherwise f cannot be one-way: Given $y = f(x)$ for $x \leftarrow_R C$ if we output a random $x' \leftarrow C$ we succeed with probability $1/|f(C)|$.

Lemma 5.1. *For every $x \in D_H$ and every PPT algorithm A we have* $\Pr_{h \leftarrow_R H}[A(x,h) \in \mathsf{Hard}_h(x)] = \mathsf{neg}$.

Proof. The proof will be by reducing finding hard siblings to inverting f. Let x_0 and A be such that $\Pr_{h \leftarrow_R H}[A(x_0,h) \in \mathsf{Hard}_h(x)] \geq \epsilon = 1/\text{poly}$. We assume that we have oracle access to A and it is deterministic. That is w.l.o.g., because with probability $1/\text{poly}$ we can fix a "good" randomness for A such that it still has the property above (with a bit weaker ϵ).

We show that the algorithm B below inverts f on a random $y \leftarrow_R f(C)$ with probability $1/\text{poly}$, and since $\frac{|C|}{2^n} = \frac{1}{2n} \geq \frac{1}{\text{poly}}$ it is a contradiction.

Algorithm B: Given $y \leftarrow_R C$ choose $h_i \leftarrow_R P_i$ at random. Then let $y_0 = f(h_i(x_0))$ and if $y_0 = y$ output $h_i(x_0)$. Otherwise choose $h_o \leftarrow_R P_o \mid h_o(y) = h_o(y_0)$ at random.[2] Let $h = h_o \circ f \circ h_i$ and run A to get $x = A(x_0, h)$, and output $h_i(x)$.

Analysis of B: We call $h_i \in P_i$ good if $\Pr_{h_o \leftarrow_R P_o}[A(x_0, h_o \circ f \circ h_i) \in \mathsf{Hard}_h(x)] \geq \epsilon/2$. By an average argument we have $\Pr_{h_i \leftarrow_R P_i}[h_i \text{ is good}] \geq \epsilon/2$. So in the following we assume that h_i is good (which in particular implies $h_i(x_0) \in C$). We hope to get back $x = A(x_0, h)$ such that $f(h_i(x)) = y$.

In order to analyze B, it is helpful to look at the following bipartite graph $G = (V_L, V_R, E)$ where $V_L = f(C) \backslash \{y_0\}$, and $V_R = P_o$. We put an edge between $z \in V_L$ and $h_o \in V_L$ if $h_o(z) = h_o(y_0)$. What the algorithm B does is first choosing a random $y \in f(C), y \neq y_0$, and then choosing a random neighbor of y: $h_o \leftarrow_R N(y)$. Then suppose $x \in \mathsf{Hard}(x_0)$ is what A returns. If $f(h_i(x)) = y$ we color the edge (y, h_o) green (which means we are happy). We know that at least $\epsilon/2$ fraction of vertices in V_R have a green edge. Our hope is to choose a green edge when we sample a random neighbor of y (for a random y), because then we will get x such that $f(h_i(x)) = y$ and so $h_i(x) \in f^{-1}(y)$. So all we have to do is to show that by choosing a random $y \in V_L$ and a random neighbor of y we choose a green edge with probability $1/\text{poly}$. But the graph G has the following properties:

- G is left regular. That is because of pairwise independence of P_o, for every $z \in V_L$ we have $d = deg(z) = \frac{|V_R|}{2^{n-k-10\log(n)}}$. So by choosing $y \leftarrow_R V_L$ and choosing $h_o \leftarrow_R N(y)$ we are choosing a random edge $(y, h_o) \leftarrow_R E$.
- At least $\frac{\epsilon}{n^9}$ fraction of E are green. That is because $|E| = d|V_L| = \frac{|V_R|}{2^{n-k-10\log(n)}}(|f(C)|-1) < |V_R|\frac{n^{10}}{2n} = \frac{|V_R|n^9}{2}$, and there are at least $\frac{\epsilon}{2}|V_R|$ green edges. So, at least $\frac{\epsilon}{n^9}$ fraction of the edges are green.

Note that if A gives x such that the edge $(f(h_i(x)), h_o)$ is green, then B inverts f successfully. Therefore the algorithm B inverts f on $y \leftarrow_R f(C)$ with probability at least $(\frac{\epsilon}{2})(\frac{\epsilon}{n^9}) \geq \frac{1}{\text{poly}}$. □

The following lemma shows that the hard siblings have density at least $1/\text{poly}$ on average. In the proof of the next lemma we assume that the families of hash functions P_i and P_o are three-wise independent, but a more careful analysis gives weaker yet sufficient bounds using pairwise independence.

[2] The construction for PIHF given in Section 4 enables us to sample h_o with this condition.

Lemma 5.2. *For every $x \in D_H$ with probability at least $\frac{1}{3n}$ over $h \leftarrow_R H$ we have $\frac{|\mathsf{Hard}_h(x)|}{|\mathsf{Sib}_h(x)|} \geq \Omega(\frac{1}{n^2})$. Therefore for every $x \in D_H$ we have $\mathsf{E}_h[\frac{|\mathsf{Hard}_h(x)|}{|\mathsf{Sib}_h(x)|}] \geq \Omega(\frac{1}{n^3}) > \frac{1}{n^4}$ (for large enough n).*

Proof. With probability $\frac{1}{2n}$ we have $h_i(x) \in C$. Let fix the value of $h_i(x) \in C$ and even $h_o(f(h_i(x))$ in the following and all probabilities and expectations will be conditioned on the fixed values of $h_i(x), h(x)$. Note that h_i and h_o are still pairwise independent for other (non-fixed) values. We first prove $\mathsf{E}[|\mathsf{Sib}_h(x)|] \leq n^9$. For every $u \in D_H$ define the boolean random variable $Y_u = 1$ iff $h(u) = h(x)$. For $u \neq x$ we have $\mathsf{E}[Y_u] = \Pr[Y_u = 1] = \Pr[f(h_i(u)) = f(h_i(x))] + \Pr[h(u) = h(x) \mid f(h_i(u)) \neq f(h_i(x))]$. Because for every $z \in f(C)$ (including $f(h_i(x))$), we have $|f^{-1}(x)| \leq 2^{k+1}$. Therefore by pairwise independence of P_i we get $\Pr[f(h_i(u)) = f(h_i(x))] \leq \frac{1}{2^{n-k-1}}$, and by pairwise independence of P_o, we get $\Pr[h(u) = h(x) \mid f(h_i(u)) \neq f(h_i(x))] = \frac{1}{2^{n-k-10\log n}}$. Note that although $Y_x = 1$ is fixed, Y_u's for $u \neq x$ are still pairwise independent and so by Lemma A.1 we have $\mathsf{E}_h[|\mathsf{Sib}_h(x)|] = 1 + \sum_{u \neq x} \mathsf{E}[Y_u] < 1 + 2^{n-k-\log(2n)}(\frac{1}{2^{n-k-1}} + \frac{1}{2^{n-k-10\log n}}) = 1 + \frac{1}{n} + \frac{n^{10}}{2n} < n^9$ (for large enough n). So using Markov's inequality, we have $\Pr[|\mathsf{Sib}_h(x)| \leq n^{10}] \geq 1 - \frac{1}{n}$.

Now we show that $\Pr[|\mathsf{Hard}_h(x)| \geq \frac{n^8}{16} - 1] \geq 1 - \frac{1}{n}$, and then by using union bound we get that (over all randomness of h_i and h_o) with probability at least $(\frac{1}{2n})(1 - \frac{1}{n} - \frac{1}{n}) > \frac{1}{3n}$ (for large n) we have $\frac{|\mathsf{Hard}_h(x)|}{|\mathsf{Sib}_h(x)|} \geq \Omega(\frac{1}{n^2})$ which proves the lemma.

Let $X = f^{-1}(f(h_i(x))) \cap C$ which has size $|X| = 2^k < \frac{|C|}{2}$ and so we have $\frac{|C \backslash X|}{2^n} \geq \frac{1}{4n}$. For any $u \neq x$ we have $u \in \mathsf{Hard}(x)$ iff $h_i(u) \in C \backslash X$ and $h(u) = h(x)$. Define the boolean random variable $Z_u = 1$ iff $u \in \mathsf{Hard}(x)$. For $u \neq x$ we have $\mathsf{E}[Z_u] = \Pr_{h_i}[h_i(u) \in C \backslash X] \Pr_{h_o}[h_o(f(h_i(u))) = h(x) \mid h_i] \geq (\frac{1}{4n})(\frac{1}{2^{n-k-10\log n}})$. Note that although $Z_x = 0$ is fixed, Z_u's for $u \neq x$ are still pairwise independent and $\sum_{u \neq x} \mathsf{E}[Z_u] \geq (2^{n-k-\log(2n)} - 1)(\frac{1}{(4n)(2^{n-k-10\log n})})$. If we pretend that Z_x is also random (in a similar way) we have $\sum_u \mathsf{E}[Z_u] \geq \frac{n^{10}}{(4n)(2n)}$. We do this artificially and then subtract one from the total amount. So by Lemma A.1 we have $\Pr[|\mathsf{Hard}_h(x)| \geq \frac{n^8}{16} - 1] \geq 1 - \frac{32}{n^8} > 1 - \frac{1}{n}$ (for large enough n). □

5.2 Amplifying the hard sets

By $H^{\times \ell}$ we mean the family of hash functions which is made by ℓ independent copies of H of previous section acting on ℓ chunks. So $H^{\times \ell} = \{(h_1, \ldots, h_\ell) \mid h_i \in H\}$ where H is the hash function of the previous section. Let $x = x_1 \ldots x_\ell, y = y_1 \ldots y_\ell \in D_{H^{\times \ell}}$. We define $y \in \mathsf{Hard}_h(x)$ if $h(x) = h(y)$ and $\exists i \; y_i \in \mathsf{Hard}(x_i)$ where the latter is based on the definition of hardness for a single h. The important point is that it is still computationally hard to find a member of $\mathsf{Hard}_h(x)$ given x (for $\ell = \text{poly}(n)$). That is because if the adversary Eve can compute a member of $\mathsf{Hard}_h(x)$ with probability at least ϵ, then for some $1 \leq i \leq \ell$ she can compute a member of $\mathsf{Hard}(x_i)$ with probability at least ϵ/ℓ (or if we want to stick to the uniform setting, one can guess some i and will succeed to find a hard sibling for $h \in H$ with probability ϵ/ℓ). Let $\alpha = \mathsf{E}_{h \in H^{\times \ell}}[\log |\mathsf{Sib}_h(x)|]$. Note that α is not dependent on the choice of x and we have $\log(n) \leq \alpha \leq n$.[3] The following lemma shows that we can amplify

[3]The first inequality is because $\alpha = \mathsf{E}[\log |\mathsf{Sib}(x)|] \geq \log(\mathsf{E}[|\mathsf{Sib}(x)|]) > \log(\mathsf{E}[|\mathsf{Hard}(x)|]) > \log(\Omega(n^7)) > \log(n)$.

the hard sets exponentially by taking $\ell = \text{poly}(n)$ large enough.

Lemma 5.3. *Let $\ell = n^{20}$, $\epsilon = \ell^{-1/3}$, and $\alpha\ell = m$, then for every $x \in D_{H^{\times\ell}}$ with probability at least $1 - O(2^{-\ell^{1/3}})$ over $h \leftarrow_R H^{\times\ell}$ we have $|\mathsf{Sib}(x)| = 2^{m(1\pm\epsilon)} \leq 2^{m+n\ell\epsilon} < 2^{m+n^{15}}$ and $\frac{|\mathsf{Easy}_h(x)|}{|\mathsf{Sib}_h(x)|} \leq 2^{-\Omega(\frac{\ell}{n^4})} < 2^{-\Omega(n^{16})}$. Therefore $|\mathsf{Easy}_h(x)| \leq 2^{m+n^{15}-\Omega(n^{16})} < 2^{m-100\ell^{3/4}}$.*

Proof. Note that $x = x_1 \dots x_\ell$ is a sibling of $y = y_1 \dots y_\ell$ iff for all i, x_i is a sibling of y_i, and in this case, y is a hard sibling of x iff for at least one i, y_i is a hard sibling of x_i.

If $\alpha_i = \log|\mathsf{Sib}(x_i)|$, we have $\mathsf{E}[\alpha_i] = \alpha$, so by Chernoff bound $\mathsf{E}[\log|\mathsf{Sib}_h(x)|] \in \ell\alpha(1\pm\epsilon)$ with probability at least $1 - 2e^{-2\ell\epsilon^2}$.

For $x_i \in D_H$, let $\theta_i = \frac{|\mathsf{Hard}_h(x_i)|}{|\mathsf{Sib}_h(x_i)|}$. Note that $\mathsf{E}_{h\leftarrow_R H}[\theta_i] = \theta$ is not dependent on x_i and we have $\theta > \frac{1}{n^4}$. By Chernoff bound we have $\sum_i \theta_i = \ell\theta(1\pm\epsilon) > \Omega(\frac{\ell}{n^4})$ with probability at least $1 - 2e^{-2\ell\epsilon^2}$. In this case we have $\frac{|\mathsf{Easy}_h(x)|}{|\mathsf{Sib}_h(x)|} = \prod_i(1-\theta_i) \leq \prod_i e^{-\theta_i} = e^{-\sum\theta_i} < 2^{\Omega(\frac{\ell}{n^4})}$.

So by union bound, with probability at least $1 - O(e^{-2\ell\epsilon^2}) > 1 - O(2^{-\ell^{1/3}})$ we get both of the inequalities above. □

5.3 Making all siblings hard

Note that our current hash function H maps domain D_H of size 2^w for $w = \ell(n-k-\log(2n))$ to range R_H of size $2^{w-10\log n+\log(2n)}$. Also note that by the definition of hardness, it is computationally hard to find a hard sibling of x. Now we show how to make all the siblings hard by increasing the output length. We will shrink the output in the next step.

Let P be a family of 3-wise independent hash functions mapping length w to $m - 50\ell^{3/4}$ (in this section we only use its pairwise independence). If H is the hash family of the previous section and $\bar{H} = \{h(x) = h_1(x)h_2(x) \mid h_1 \in H, h_2 \in P\}$ be a new hash family (which increases the length). For $x, y \in D_{\bar{H}}$ we call y a hard sibling for x under $h = (h_1, h_2) \in \bar{H}$ if $h(x) = h(y)$ and y is a hard sibling for x under h_1. With probability at least $1 - 2^{-n}$ over $h_1 \leftarrow_R H$ there are at most $2^{m-100\ell^{3/4}}$ easy siblings for x under h_1. Now if this is the case, each of the easy siblings will map to the same point as x does under $h_2 \leftarrow_R P$ with probability at most $2^{-m+50\ell^{-3/4}}$. Therefore using union bound, with probability at least $1 - 2^{-50\ell^{3/4}}$ over $h_2 \leftarrow_R P$ none of them will map to the same point as x. So we get that with probability at least $(1 - 2^{-n})(1 - 2^{50\ell^{3/4}}) > (1 - O(2^{-n}))$ over $h \leftarrow_R \bar{H}$, all of the siblings of x are hard.

5.4 Shrinking the length

Note that our hash function, before making all the siblings hard, had this property that for every x, with probability at least $1 - O(2^{-\ell^{1/3}})$ we had $|\mathsf{Sib}(x)| > 2^{m(1-\epsilon)} > 2^{m-\ell^{3/4}}$. If this is the case for x, by changing H to $\bar{H}$, the expected number of siblings remains at least $2^{49\ell^{3/4}}$. Again in the following, we use the name H for $\bar{H}$. By Lemma A.1, with probability at least $1 - 2^{-40\ell^{3/4}}$ over $h \leftarrow_R \bar{H}$, we have $|\mathsf{Sib}(x)| > 2^{40\ell^{3/4}}$. We call such x a good one. So, the expected fraction of good x's among all $x \in D_H$ is at least $1 - 2^{-40\ell^{3/4}}$, and by an average argument, with probability at least $1 - 2^{-6\ell 3/4}$, we have that at least $1 - 2^{-6\ell^{3/4}}$ fraction of x's are good. When this is the case, the size of the image of the hash function

$|h(D_H)|$ is at most $2\frac{|D_H|}{2^{6\ell^{3/4}}}$, because the good x' can contribute at most $\frac{|D_H|}{2^{40\ell^{3/4}}}$, and the remaining x's are at most $\frac{|D_H|}{2^{6\ell^{3/4}}}$ many.

Therefore with probability at least $1-O(2^{-\ell^{1/3}}) > 1-2^{-n}$ we have $|h(D_H)| < \frac{|D_H|}{2^{2n}}$. Note that the input length $n' = \log |D_H|$ is bigger than n now. If we take P to be a new family of pairwise independent hash functions mapping the output of H to $\{0,1\}^{n'-n}$, and take $\bar{H} = \{f \circ h \mid f \in P, h \in H\}$, then $\bar{H}$ is shrinking n' bits to $n'-n$ bits. But more importantly, for every x, with high probability $\bar{H}$ does not add any sibling to $\mathsf{Sib}(x)$ and all of them remain hard. Namely, with probability at least $1-2^{-n}$, we have $|h(D_H)| < \frac{|D_H|}{2^{2n}} = 2^{n'-2n}$, and by union bound, with probability at least $1-2^{-n}$, there is no new point in the image of H colliding with the image of x, under $f \leftarrow_{\mathrm{R}} P$. Therefore, for any single x, with probability at least $1-O(2^{-n})$, all of its siblings are hard.

5.5 Concatenation

Note that we did not know neither the value of k nor the value of m. Assuming that we knew k, and we knew m (up to $n^{1/100}$ multiplitcative approximation) we showed how to get a shrinking hash function family. If we try all the constructions by using all $0 \le k \le n$ and all values of $m = n^{i/100}$ for $1 \le i \le 10000$, all of them are shrinking (so the concatenation is also shrinking) and one of them is universal one-way. It is easy to see that the the latter property makes the concatenation also universal one-way.

A Some useful facts

Lemma A.1. *Let $X_1, \ldots, X_N$ be pairwise independent boolean random variables such that $\mathsf{E}[X_i] = \Pr[X_i = 1] = p$, and $X = \sum_i X_i$, and so $\mu = \mathsf{E}[X] = Np$. Then $\Pr[X \ge \mu/2] \ge 1 - 4/\mu$.*

Proof. Using Chebyshev's inequality we have:

$$\Pr[X < \frac{\mu}{2}] \le \Pr[|X-\mu| > \frac{\mu}{2}] < \frac{\mathsf{Var}[X]}{(\frac{\mu}{2})^2} = \frac{\sum_i \mathsf{Var}[X_i]}{\mu^2} = \frac{4Np(1-p)}{\mu^2} < \frac{4Np}{\mu^2} = \frac{4\mu}{\mu^2} = \frac{4}{\mu}$$

□

References

[DH76] W. Diffie and M. Hellman. New Directions in Cryptography. *IEEE Transactions on Information Theory*, IT-22(6):644–654, Nov. 1976.

[KK08] J. Katz and C.-Y. Koo. On Constructing Universal One-Way Hash Functions from Arbitrary One-Way Functions. *Journal of Cryptology*, 2008. To appear. Preliminary version available on `http://www.cs.umd.edu/~jkatz/`.

[NY89] M. Naor and M. Yung. Universal One-Way Hash Functions and their Cryptographic Applications. In *Proc. 21st STOC*, pages 33–43. ACM, 1989.

[RSA78] R. L. Rivest, A. Shamir, and L. M. Adleman. A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. *Communications of the ACM*, 21(2):120–126, Feb 1978.

[Rom90] J. Rompel. One-way functions are necessary and sufficient for secure signatures. In *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing, STOC'90 (Baltimore, Maryland, May 14–16, 1990)*, pages 387–394, New York, 1990. ACM SIGACT, ACM Press.

[Sim98] D. R. Simon. Finding Collisions on a One-Way Street: Can Secure Hash Functions Be Based on General Assumptions? In *EUROCRYPT*, pages 334–345, 1998.