Proofs of security for the Unix password hashing algorithm

David Wagner Ian Goldberg
University of California, Berkeley
{daw,iang}@cs.berkeley.edu

Abstract. We give the first proof of security for the full Unix password hashing algorithm (rather than of a simplified variant). Our results show that it is very good at extracting almost all of the available strength from the underlying cryptographic primitive and provide good reason for confidence in the Unix construction.

1 Introduction

This paper examines the security of the Unix password hashing algorithm, the core of the Unix password authentication protocol [14]. Although the algorithm has been conjectured cryptographically secure, after two decades and deployment in millions of systems worldwide it still has not been proven to resist attack. In this paper, we provide the first practical proof of security (under some reasonable cryptographic assumptions) for the Unix algorithm.

The hashing algorithm is a fairly simple application of DES, perhaps the best-known block cipher available to the public. Since DES has seen many man-years of analysis, in an ideal world we might hope for a proof (via some reduction) that the Unix password hash is secure if DES is. However, so far no such proof has appeared in the literature.

In earlier work, Luby and Rackoff presented strong theoretical evidence that the basic approach found in the Unix algorithm is likely to be sound, by presenting proofs of security for a simplified variant of the Unix hash [12, 13]. However, their proofs have three serious limitations: the abstract model they analyze omits some important features of the real algorithm (they analyze the variant $k \mapsto E_k(0)$ rather than the full iterated construction $k \mapsto E_k^{25}(0)$); their proofs of security are asymptotic, and so do not directly apply to real (necessarily finite) instantiations of the construction; and they assume a uniform distribution on passwords. Therefore, we feel that, from a practical point of view, the security of the real Unix password hash remains an open question.

In the first half of this paper, we take a further step towards justifying the design of the Unix password hash by removing the first two limitations mentioned above (we also make some progress towards removing the final limitation in the second half of this paper, as will be discussed below). Our primary contribution is that we show how to analyze the *full* Unix construction, removing the need to abstract away features of the algorithm. This removes the gap between what has been analyzed and what is currently in use. In particular, we extend Luby and Rackoff's proof techniques to handle the iterated construction $k \mapsto (E_k \circ E_k \circ \cdots \circ E_k)(0)$ found in the real Unix hash.

We also provide explicit, quantitative security measures for the Unix hash (instead of asymptotic estimates), and as a result, our proofs can be directly applied to the real (finite) Unix algorithm. We make no claims about the novelty of this calculation—it is a straightforward technical exercise—but the concrete bounds are important if we are to assess the practical security of the Unix hash construction in real life.

In practice the Unix password hash suffers from the same limitation as DES: both algorithms appear to be very well designed, but their short key size limits the attainable security level. Nonetheless, we show here that the construction used in the Unix password hash is cryptographically sound and does a very good job of extracting almost all of the available strength from the underlying cryptographic primitive.

We also show a result that may be of independent interest: every pseudo-random generator forms a one-way function, and this construction is simultaneously very efficient and strongly security-preserving. See Theorem 1 for a surprisingly tight reduction (our concrete security parameters are better than those obtained from the apparently-standard analysis of this construction [6, Proposition 3.3.6]). This theorem was effectively the main tool in the Luby-Rackoff proof, but it was never separated out explicitly.

The main practical shortcoming of the proof techniques discussed above is that, for best results, we must assume that the passwords are uniformly distributed. To remedy this shortcoming, we also present some initial progress towards handling the non-uniform case as well.

In general, the security issues associated with non-uniformly distributed keying material appear to be under-represented in the literature. A second contribution of this paper is that we make some initial progress on this problem, presenting a formal model that we hope may serve as a foundation for future exploration in this area. Using this model, we are able to show relatively good lower bounds on the security of the Unix algorithm when used with non-uniformly distributed passwords. These techniques provide practically useful results for the special case of the Unix hash function, but in general the results that can be obtained via these methods are not as strong as we would like, and so we leave this as an open question for further research. See Section 7.

This paper is structured as follows. We recall the definition of the Unix password hash in Section 2 and then summarize the results of our analysis in Section 3. The remainder of the paper is dedicated to the theoretical analysis: Section 4 outlines the main ideas from a high level, Section 5 gives some important definitions, and Section 6 dives into the details of the proofs. Finally, Section 7 gives a formal model for security with non-uniformly distributed passwords and presents some initial results in this area.

2 The Unix algorithm

We briefly recount the definition of the Unix password hashing function. The function—let us call it H—is built out of a 25-fold iterated version of DES, in the following way. Let $\mathrm{DES}_k(x)$ denote the DES encryption of plaintext x under key k and $\mathrm{DES}_k^n(x) = \mathrm{DES}_k(\mathrm{DES}_k^{n-1}(x))$ denote the n-fold iteration of DES_k. Then the hash $H(k)$ of the 8-character password k may be defined as

$$H(k) = \mathrm{DES}_k^{25}(0).$$

When a new user account is created, the hash $H(k)$ of the user's initial password is stored with the user's id in the (world-readable) system password file `/etc/passwd`. When the user attempts to log on with password k', the system computes $H(k')$ and compares the result to the value stored in the password file, allowing the user to log on only if $H(k') = H(k)$.

Our description of the Unix password scheme omits one important feature of the construction: the salt. In fact, when the user register his password k for the first time, a random 12-bit salt s is generated, and the system computes a salted hash $H_s(k)$ from k and s. We do not analyze the effect of the salt in this paper.

3 Results

The main consequence of our analysis is the following informal result:

> If DES is a $(t, 25, e)$-secure block cipher, then the Unix password hashing function is a (t', p)-secure password hashing function, where $t' \approx t$ and $p \approx (1 + 1/255)e$.

Some interpretation of this analytical result is clearly in order. Formal definitions of (t, q, e)-security for block ciphers and (t', p)-security for hash functions will be provided later in Section 5, but for now we just sketch the intuition. Roughly speaking, the theorem says that if DES is secure against all attacks using at most 25 chosen plaintexts, and if the password is chosen uniformly at random, then the Unix construction is secure against password guessing attacks.

Note that our security proofs require only very mild assumptions on the properties of DES. To break the Unix algorithm, the adversary must have some way to break DES with only 25 chosen plaintexts, which is likely to be a very difficult task. Furthermore, even the existence of such an attack on DES is no guarantee of success at breaking the Unix hash function, since it seems to be very difficult to control the internal values of the hash computation. Therefore, we expect that the Unix hash function is likely to be even stronger than our lower bounds would suggest.

An example. Let us try to estimate the resources needed to reverse the Unix hash function. We start by estimating the concrete security level afforded by DES. The best attack reported in the literature for breaking DES with 25 chosen plaintexts is exhaustive keysearch; differential and linear cryptanalysis do not help with such a small number of chosen texts. If we mount a partial exhaustive keysearch, searching over t keys, we obtain an attack with time complexity t and success probability[1] $e \leq t/2^{55}$. Therefore, if the cryptanalytic results reported in the literature are representative and this is indeed the best available attack, we may conclude that DES forms a $(t, 25, t/2^{55})$-secure block cipher. Theorem 1 then says that the Unix scheme is (t, p)-secure for $p \approx (1 + 1/255)t/2^{55}$, which is only larger than the corresponding success probability for attacking DES by the small multiplicative factor of $1+1/255$. To summarize:

> For an adversary with a given set of resources, the chances of breaking the Unix password hash are at most only slightly higher—less than 1% higher—than the chances of breaking DES with the same resources.

This illustrates that the reduction is nearly tight: our analysis requires only very weak assumptions of security for DES, and as a result, our results will still be relevant even if DES is found to have some small weakness. In other words, the Unix construction is robust: any small imperfections that might exist in DES are guaranteed not to be magnified by the Unix construction into a fatal flaw for a hashing function.

Limitations. There are several important technical limitations to our work. First, we do not analyze the salt, so we do not consider attacks on many passwords in parallel. Second, although we are for the first time able to show that the iteration in the Unix hash does not harm security, we were not able to prove that iteration actually improves security, as one would intuitively expect. Third, our results for the non-uniform distribution are not as strong as we would like, as is discussed in more detail elsewhere in this paper.

In practice probably the most significant vulnerabilities in the Unix password hash function are that real passwords often do not contain enough entropy to resist

[1] We assume the adversary exploits the DES complementation property, and thus $e = t/2^{55}$, not $t/2^{56}$ as one might naively expect.

dictionary attacks [3, 5, 9, 11, 15], that the 56-bit keysize of DES is too short to resist exhaustive keysearch attacks [4], and that cleartext passwords are inappropriate for use in a networked environment. However, our results show that the Unix password hashing construction attains about as much cryptographic strength as possible, given these unavoidable limits on its security.

4 An outline of the analysis

Our analysis of the Unix hash uses essentially only one new idea[2]: an observation about close ties between the Unix hash and the CBC-MAC construction. In the remainder of this section, we give a high-level sketch of these two fundamental observations.

RELEVANCE OF THE CBC-MAC. First, we show that the Unix password hashing algorithm is just a special case of the more general and better-studied DES-CBC-MAC construction [1]. Consequently, we can take advantage of well-known results on the security of DES-CBC-MAC.

Let f-CBC-MAC(x) denote the CBC-MAC of the message x under the function f. Recall that the f-CBC-MAC of a n-block message x under function f is defined as

$$f\text{-CBC-MAC}(x) = f(x_n \oplus \cdots f(x_2 \oplus f(x_1)) \cdots).$$

Then it is not hard to see that we get a close relation between n-fold iterated encryption and the CBC-MAC on a n-block message:

$$f^n(x) = f\text{-CBC-MAC}(\langle x, 0, 0, \dots, 0\rangle).$$

This observation may be of independent interest, because it gives a simple and powerful way to analyze iterated encryption.

Using this trick, we observe that the Unix password hashing algorithm can be related to the DES-CBC-MAC by

$$\text{Unix-hash}(k) = \text{DES-CBC-MAC}_k(\langle 0, \dots, 0\rangle).$$

This is the basis for our treatment of iteration in the Unix password hash.

5 Definitions

Definitions of concrete security are parametrized by a measure of the resources needed to break the cryptographic primitive. In general, we say that an attack R-breaks a crypto primitive if the algorithm succeeds in breaking the primitive with resources specified by R, and we say that a crypto primitive is R-secure if there is no algorithm[3] to R-break it. In the definitions to follow, we elaborate on the measure of an adversary's resources.

First, we formally define the concept of a pseudorandom function (PRF). Let $F : \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ be a keyed function. We say that the oracle algorithm A is an adversary which (t, q, e)-breaks the alleged-PRF F if A runs in time t, makes

[2] We also give what we believe to be a simpler presentation of Luby and Rackoff's proof that $k \mapsto E_k(0)$ is a good one-way function if E is a good block cipher, but the result itself is not new.

[3] We may assume without loss of generality that all adversarial algorithms behave deterministically, since any probabilistic adversary can be de-randomized using standard techniques.

at most q queries to its oracle, and has advantage $\mathrm{Adv}\, A = e$. The adversary's advantage $\mathrm{Adv}\, A$ is defined to be

$$\mathrm{Adv}\, A = |\Pr[A^{F_k} = 1] - \Pr[A^R = 1]\,|,$$

where the probability is taken over the choice of k and R, and where the random variable k is drawn from the uniform distribution over $\mathcal{K}$ and $R : \mathcal{X} \to \mathcal{Y}$ is a random function. We say that F is a (t, q, e)-secure PRF if there is no adversary which (t, q, e)-breaks F.

A (t, q, e)-secure "super" pseudorandom permutation (PRP) $E : \mathcal{K} \times \mathcal{Y} \to \mathcal{Y}$ is a family of permutations with the property that E_k is indistinguishable from a random permutation $\pi : \mathcal{Y} \to \mathcal{Y}$ chosen uniformly at random from the set of all permutations on $\mathcal{Y}$, when k is drawn uniformly at random from $\mathcal{K}$. The advantage of an adversary A is defined as $\mathrm{Adv}\, A = \left|\Pr[A^{E_k, E_k^{-1}} = 1] - \Pr[A^{\pi, \pi^{-1}} = 1]\right|$. Note that we typically omit the "super" prefix for brevity.

A pseudorandom generator (PRG) is a function $g : \mathcal{K} \to \mathcal{Y}$ which stretches a short seed (from $\mathcal{K}$) into a long, random-looking output. The advantage of an adversary for g is defined to be $\mathrm{Adv}\, A = |\Pr[A(g(k)) = 1] - \Pr[A(u_Y) = 1]|$, where the random variables k and u_Y are chosenly randomly according to the uniform distributions on $\mathcal{K}$ and $\mathcal{Y}$ (resp.). In the case of pseudorandom generators, normally one insists that the output be longer than the seed, i.e., that $|\mathcal{Y}| > |\mathcal{K}|$.

Also, it is useful to have the concept of a one-way function (OWF). Let $h : \mathcal{K} \to \mathcal{Y}$ be an unkeyed function. An adversary B attacking h is an algorithm with input $y \in \mathcal{Y}$ which outputs a symbol in $\mathcal{K} \cup \{\bot\}$, and which is *correct*; B is correct when $y \in \mathcal{Y}, B(y) \neq \bot$ implies $h(B(y)) = y$. We say that an adversary B (t, p)-breaks the alleged-OWF h if B runs in time t and succeeds with probability $p = \Pr[B(h(k)) \neq \bot]$, where the probability is taken over the choice of $k \in \mathcal{K}$, and the random variable k is drawn from the uniform distribution. Finally, we say that g is a (t, p)-secure OWF if there is no adversary which (t, p)-breaks it.

Note that the notion of a one-way function exactly captures the security properties we need from a password hashing function. In particular, if g is a (t, p)-secure OWF, then the success probability of any adversary running in time t is at most p.

6 Analysis

The main result is a proof that any pseudorandom generator is a good one-way function. This is an version of Luby and Rackoff's result [12, 13], adapted to the concrete security model.

Theorem 1. *Let $g : \mathcal{K} \to \mathcal{Y}$ be a (t, e)-secure pseudorandom generator, with $|\mathcal{Y}| > |\mathcal{K}|$. Then g is a (t', p)-secure one-way function, where $p = e/(1 - |\mathcal{K}|/|\mathcal{Y}|)$ and $t' \approx t$.*

Remark 1. To be more precise, we show that g is (t', p)-secure, where $t' = t - c$ and c is a universal constant which depends only on the machine model. However, in practice c is extremely small compared to t, so for simplicity of exposition in this paper we omit these tiny constants and summarize the situation by writing $t' \approx t$.

Proof. We prove the contrapositive. Let $h = g$ be our alleged one-way function. Suppose that there is an adversary B which (t, p)-breaks h (viewed as a one-way function). We construct an adversary A against g (viewed as a PRG), defined by

$$A(y) = \begin{cases} 1 & \text{if } B(y) \neq \bot \\ 0 & \text{otherwise.} \end{cases}$$

Our claim is that A (t, e)-breaks g (the pseudorandom generator), i.e., that Adv $A \geq (1 - |\mathcal{K}|/|\mathcal{Y}|) \cdot p$. A bit of notation: we let k stand for a random variable uniformly distributed over $\mathcal{K}$, and u_Y for a r.v. that is uniform over $\mathcal{Y}$. All probabilities are calculated with respect to k.

Let $V = \{y \in \mathcal{Y} : B(y) \neq \perp\}$ be the set of outputs of h where B succeeds. Also, let $W = \{k \in \mathcal{K} : h(k) \in V\}$ be the set of inputs to h which are not secure against B. We see that $p = \Pr[h(k) \in V] = \Pr[k \in W]$.

Next, we observe that $|V| \leq |W|$. The argument goes like this. We may view B as a deterministic function (by standard de-randomization results). We examine B', the restriction of B to the domain V. This restriction is well-defined, since when $v \in V$, $B(v)$ is a well-defined element of K. Moreover, using the correctness of B, we have $g(B(v)) = v \in V$ for all $v \in V$, so that $B(v) \in W$ for all $v \in V$. Thus we may consider B' as a function with signature $V \to W$. Also, if $v, v' \in V$ and $B(v) = B(v')$, we find that $v = g(B(v)) = g(B(v')) = v'$; therefore, B' is one-to-one. In summary, we have exhibited a one-to-one function from V to W, which demonstrates that $|V| \leq |W|$.

Finally, we are ready to calculate the advantage of the adversary A. First,

$$\Pr[A(g(k)) = 1] = \Pr[B(h(k)) \neq \perp] = \Pr[h(k) \in V] = p.$$

Also $|W| = |\mathcal{K}| \cdot \Pr[k \in W] = |\mathcal{K}| \cdot p$ and $|V| \leq |W|$, so

$$\Pr[A(u_Y) = 1] = \Pr[u_Y \in V] = |V|/|\mathcal{Y}| \leq |W|/|\mathcal{Y}| = |\mathcal{K}|/|\mathcal{Y}| \cdot p.$$

Plugging into the definition of Adv A gives

$$\text{Adv } A \geq |p - |\mathcal{K}|/|\mathcal{Y}| \cdot p| = (1 - |\mathcal{K}|/|\mathcal{Y}|) \cdot p = e.$$

To recap, under the assumption that there is an adversary B which (t, p)-breaks h, we obtain an adversary A which shows that g is not (t, e)-secure, and this is the desired result. □

Lemma 1. *A (t, q, e)-secure PRP $E : \mathcal{K} \times \mathcal{Y} \to \mathcal{Y}$ is a $(t, q, e + q^2/2|\mathcal{Y}|)$-secure PRF.*

Proof. This lemma is a well-known consequence of the birthday paradox. For a full proof, see, e.g., [1]. □

Lemma 2. *If $F : \mathcal{K} \times \mathcal{Y} \to \mathcal{Y}$ is a (t, q, e)-secure PRF, then F^n is a $(t', q/n, e')$-secure PRF, where $t' = t - q \log_2 |\mathcal{Y}|$ and $e' = e + 1.5q^2/|\mathcal{Y}|$.*

Proof. Recall that $F_k^n(x) = F_k(\cdots(F_k(x))\cdots)$ is a F_k-CBC-MAC on the message $\langle x, 0, 0, \ldots, 0\rangle$, as noted in Section 4. Now invoke [1, Lemma 4.1] to show that the F_k-CBC-MAC is a secure PRF. □

Lemma 3. *If F is a $(t, 1, e)$-secure PRF, then $g(k) = F_k(0)$ is a (t, e)-secure PRG.*

Proof. Immediate from the definitions. □

Theorem 2. *If DES is a $(t, 25, e)$-secure pseudorandom permutation, then the Unix construction is a (t', p)-secure password hashing function, where $p = (1 + 1/255)e + (1 + 1/255) \cdot 25^2/2^{63} \approx (1 + 1/255)e$ and $t' \approx t$.*

Proof. Applying Lemmas 1 and 2, we see that $x \mapsto \text{DES}_k^{25}(x)$ is a $(\tau, 1, \epsilon)$-secure PRF, where $\tau = t - 25 \times 64$ and $\epsilon = e + 2q^2/|\mathcal{Y}|$. Lemma 3 then shows that the Unix algorithm is a (τ, ϵ)-secure PRG. Finally, Theorem 1 assures us that the Unix construction is a (t', p)-secure one-way function, where $t' = \tau - c = t - 25 \times 64 - c \approx t$ and $p = \epsilon/(1 - 2^{-8}) = (1 + 1/255)\epsilon$. As discussed above, this is exactly the notion needed to show that the Unix password hashing algorithm is secure when used with uniformly-distributed passwords. □

7 Non-uniformly distributed secrets

So far our proofs of security have assumed that all passwords are uniformly distributed. In practice, though, such an assumption is often far from the mark [3, 5, 9, 11, 15]. This section tackles the issue of security for non-uniform distributions.

In this section, we introduce a new security model, *passphrase-based cryptography*, where keying material and other cryptographic secrets are derived from human-entered passphrases and thus are likely to have a highly non-uniform distribution. This is a significant departure from the standard model, where the very definitions of security assume a uniform distribution on the keys. A second important difference is that passphrases are typically relatively short, so the secret entropy in them is a scarce resource which we must not waste. We show that the standard approaches to smoothing non-uniform distributions are unsuitable for practical use because they waste too much entropy. Therefore, new techniques are needed.

Let us start by developing formal definitions of security for passphrase-based cryptography. We need a small amount of background. Let D be a distribution on $\mathcal{K}$ which assigns the probability $D(k)$ to each $k \in \mathcal{K}$, and let $D(S) = \sum_{k \in S} D(k)$ for all $S \subseteq \mathcal{K}$.

We define the notion of a *one-way function secure for* D as in Section 5, except that the success probability is now calculated when inputs are distributed according to D rather than the uniform distribution. (We assume that the distribution is fixed in advance, so that the attack algorithm may depend on D.) Let $f : \mathcal{K} \to \mathcal{Y}$ be an unkeyed function and let $B : \mathcal{Y} \to \mathcal{K} \cup \{\perp\}$ be an adversary against f that is *correct* (i.e., $B(y) \neq \perp$ implies $f(B(y)) = y$). We say that the algorithm B (t, p)-breaks f (for D) if B runs in time t and succeeds with probability $p = \Pr[B(f(k)) \neq \perp]$, where k is chosen from $\mathcal{K}$ according to the distribution D and the probability is taken with respect to the choice of k. Finally, the one-way function f is (t, p)-secure for D if there is no adversary which (t, p)-breaks f for D.

In this paper, we define

$$\chi_D(t) = \max\{D(S) : S \subseteq \mathcal{K}, |S| \leq t\}.$$

This definition is motivated by the following upper bound on the security of hashing inputs with distribution D:

Theorem 3. *For all one-way hash functions f, all distributions D, and all time bounds t, there is a generic attack, called the* dictionary attack*, which $(t, \chi_D(t))$-breaks f.*

Proof. The dictionary attack proceeds by trying the t elements of D with the t largest probabilities. (Each guess can be easily checked with a single computation of f.) If we write the D-probabilities in decreasing order, $d_1 \geq d_2 \geq d_3 \geq \cdots$, we can see that the success probability of the dictionary attack is $d_1 + d_2 + \cdots + d_t$; furthermore, this quantity is precisely $\chi_D(t)$. □

Therefore, χ_D describes the effectiveness of the optimal dictionary-search attack against D: no matter what we do, *every* one-way hash function with inputs chosen according to D can be broken with probability $\chi_D(t)$ and time t.

There is no way to avoid the dictionary attack. This motivates our definition of security for a one-way hash function that operates on inputs with a non-uniform distribution:

Definition 1. *We say that the one-way function f is* ideally-secure *for distribution D if f is $(t, \chi_D(t))$-secure (for all t) when its inputs are distributed according to D.*

Intuitively speaking, a one-way function is ideally-secure if the dictionary attack is the best attack.

We are able to show that any one-way function that is sufficiently strong for uniformly distributed inputs will also be relatively good for other distributions.

Theorem 4. *Let f be a one-way function that is (t,p)-secure for uniformly-distributed inputs. Then, for every distribution D on $\mathcal{K}$, f is a $(t, \chi_D(p|\mathcal{K}|))$-secure one-way hash function for D.*

Proof. Let A be an adversary which (t,p')-breaks f for D, where $p' \geq \chi_D(p|\mathcal{K}|)$. We will show that A also (t,p)-breaks f (for uniformly distributed inputs), and then taking the contrapositive will yield the desired result.

Let $S = \{k \in \mathcal{K} : A(f(k)) \neq \perp\}$ be the set of f-inputs which are not safe against A. Note that $p' = D(S)$, and moreover that $\chi_D(|S|) \geq D(S)$ (by the definition of χ_D), so we have

$$\chi_D(|S|) \geq D(S) = p' \geq \chi_D(p|\mathcal{K}|).$$

Since $\chi_D(t)$ is a monotonically increasing function of t, we may conclude that $|S| \geq p|\mathcal{K}|$.

Now we may prove that A indeed works well, not just for the distribution D, but also for the uniform distribution. Note that

$$\Pr[A(f(k)) \neq \perp] = \Pr[k \in S] = |S|/|\mathcal{K}| \geq p$$

when k is drawn from the uniform distribution on $\mathcal{K}$. Therefore, A is an adversary that (t,p)-breaks f (for the uniform distribution), as claimed, and the theorem follows. □

Corollary 1. *If the one-way function f is ideally-secure for the uniform distribution, then it is also ideally-secure for all other distributions as well.*

Proof. For the uniform distribution U on $\mathcal{K}$, we have $\chi_U(t) = t/|\mathcal{K}|$, so by assumption f is $(t, t/|\mathcal{K}|)$-secure for all t. Now Theorem 4 assures us that f is $(t, \chi_D(t))$-secure for all distributions D, since $\chi_D((t/|\mathcal{K}|) \cdot |\mathcal{K}|) = \chi_D(t)$. □

Applications to Unix password hashing. We can show that the Unix hash is good at hashing even non-uniformly distributed passwords, under assumptions on DES that appear to be reasonable (albeit stronger than one might like).

In Section 3, we argued that DES appears to be $(t, 25, t/2^{55})$-secure, if the cryptanalysis results reported in the literature do indeed represent the best attacks on DES (as many researchers believe). This assumption implies that the Unix hash H is a $(t', t/2^{56})$-secure one-way function when its inputs are uniformly distributed, where $t' = (1-2^{-8})(t/2-25^2/2^8) \approx (1+2^{-8})t/2$. Thus, Theorem 4 allows us to conclude that the Unix hash is $(t', \chi_D(t))$-secure—i.e., nearly $(t/2, \chi_D(t))$-secure—for every distribution D.

This lower bound only differs from Theorem 3's upper bound by a factor of about two[4]. Roughly speaking, this means that the Unix hash appears to be nearly ideally-secure for all distributions D: no shortcut attack can do much better than the dictionary attack.

Whether this result is useful in practice will depend on several factors. One disadvantage is that the approach requires relatively strong assumptions about DES—that there are no shortcut attacks on DES that reduce the workfactor of exhaustive

[4] If we consider that the Unix hash internally iterates DES 25 times and thus costs 25 times as much to compute as does a single DES trial encryption, the gap between the upper and lower bounds becomes a factor of about 50, which is still quite small.

keysearch by more than a small factor when the key is uniformly distributed—and as a result, the result is not as robust as we would like. For example, if small weaknesses are present in DES, our proof techniques cannot rule out the possibility that these weaknesses might be greatly magnified when one uses DES with patterned passwords, even though such a worst-case scenario is considered unlikely by practitioners.

However, it is interesting to point out that we obtain a proof of security for the Unix hash of patterned passwords starting only with the assumption that DES is secure for uniformly-distributed keys. In particular, we make no assumptions whatsoever about the behavior of DES when keyed from a non-uniform distribution. Consequently, we can take advantage of the decades of analysis on DES (which has all been premised on the assumption of uniformly-distributed keys) to gain confidence in the security of the Unix algorithm.

Applications to other crypto primitives. It is also worth noting that Theorem 4 can also be generalized to many other keyed cryptographic primitives, such as block ciphers, stream ciphers, and PRF's, using the same style of proof.

Tightness. One can show that our lower bound (given in Theorem 4) on the security of f for non-uniform distributions is essentially tight. In other words, it is unlikely that one can do much better without either making additional assumptions on f or finding a better construction.

The following simple example is due to David Zuckerman [16]. Let $g : \mathcal{K}_1 \to \mathcal{Y}_1$ be an ideally-secure one-way function with keyspace $\mathcal{K}_1$ and output space $\mathcal{Y}_1$. We construct $f : \mathcal{K} \to \mathcal{Y}$ as $f(\langle x, y\rangle) = \langle g(x), y\rangle$, where $\mathcal{K} = \mathcal{K}_1 \times \mathcal{K}_2$ and $\mathcal{Y} = \mathcal{Y}_1 \times \mathcal{K}_2$. Note that f is $(t, t/|\mathcal{K}_1|)$-secure (for all t) for the uniform distribution on $\mathcal{K}$.

Next consider the uniform distribution D on $S \times \mathcal{K}_2$ for some $S \subseteq \mathcal{K}_1$, i.e., $D(\langle x, y\rangle) = 1/(|S| \cdot |\mathcal{K}_2|)$ for all $\langle x, y\rangle \in S \times \mathcal{K}_2$ and $D(\langle x, y\rangle) = 0$ for $x \notin S$. Theorem 4 implies f is (t, p)-secure for D, where

$$p = \chi_D\left(\frac{t}{|\mathcal{K}_1|}|\mathcal{K}|\right) = \chi_D(t|\mathcal{K}_2|) = \frac{t \cdot |\mathcal{K}_2|}{|S| \cdot |\mathcal{K}_2|} = t/|S|.$$

At the same time, one may clearly $(t, t/|S|)$-break f using a dictionary attack when its inputs are distributed according to D (see Theorem 3). This shows that Theorem 4 is tight.

The power of stronger assumptions. One alternative approach is to start from the assumption that DES is secure (up to the possibility of dictionary attacks) no matter what distribution the key is drawn from. Then we may attempt to prove that the Unix hash is secure for passwords with distribution D if DES is secure for keys with distribution D.

The following theorem, which forms a nice example of this approach, is due to Bellare (and was stated as a homework problem in [7]):

Theorem 5. *If $g : \mathcal{K} \to \mathcal{Y}$ is a (t, e)-secure pseudorandom generator for seeds distributed according to D, then g is a (t, p)-secure one-way function for D, where $p = e + |\mathcal{K}|/|\mathcal{Y}|$.*

Remark 2. Of course, we may take D to be the uniform distribution in the above; however, this gives strictly weaker bounds than Theorem 1's dedicated analysis.

Proof. Use the same notation as in the proof of Theorem 1, and define the adversary A in the same way. Note that $\Pr[A(u_Y) = 1] \le |\{g(k) : k \in \mathcal{K}\}|/|\mathcal{Y}| \le |\mathcal{K}|/|\mathcal{Y}|$, and $\Pr[A(g(k)) = 1] = p$ as before, so we get

$$\mathrm{Adv}\, A = |\Pr[A(g(k)) = 1] - \Pr[A(u_Y) = 1]| \ge |p - |\mathcal{K}|/|\mathcal{Y}|| = e.$$

In other words, if there is an adversary B to (t, p)-break g as a one-way function, then there is another adversary A to (t, e)-break g as a pseudorandom generator, and the theorem follows. $\square$

While this result may be useful in some contexts, it doesn't give terribly useful lower bounds for the security of the Unix hash. For the Unix algorithm, we have $|\mathcal{K}|/|\mathcal{Y}| = 2^{-8}$, so we won't be able to rule out the possibility that there exists an algorithm that succeeds in breaking 1/256 of all passwords in constant time. Such a result is not very reassuring.

One could attempt to repair the flaw by defining a new hash construction, e.g., New-hash$(k) = \langle \mathrm{DES}_k(0\ldots00), \mathrm{DES}_k(0\ldots01)\rangle$. Such an approach would work—if one is willing to deploy an updated implementation of the password hashing algorithm on millions of machines around the world!—but it would still require strong assumptions about the security of DES when used with non-uniformly distributed keys. Since DES has not received as much scrutiny in this setting (where the key is non-uniformly distributed), it becomes harder to gain much confidence that the necessary assumptions are indeed satisfied.

Therefore, we conclude that this approach does not seem to yield security bounds that are as meaningful as those that can be achieved with Theorem 4.

Comparison to entropy smoothing. Another alternative approach to dealing with patterned passwords is to smooth out the non-uniformity in the distribution. A well-known result called the *leftover hash lemma* [8, 10] shows that universal hash functions are good at entropy smoothing: if h is selected uniformly at random from a family of universal hash functions with m-bit outputs, and if k is drawn from a distribution with at least $3m$ bits of Renyi entropy, the random variable $\langle h, h(k)\rangle$ will be approximately uniformly distributed.

The disadvantage with the leftover hash lemma is that it wastes at least two-thirds of the entropy of the password k: if we want to feed the smoothed bits into the Unix hash function (e.g., New-hash$(k) = \langle h, \text{Unix-hash}(h(k))\rangle$), we need a passphrase with at least $3 \times 56 = 168$ bits of entropy. This would require that passphrases consist of hundreds of characters, which is too difficult for most mere mortals to memorize. When we consider that, in the real world, one is lucky to find a password with more than 25–35 bits of entropy [3, 5, 9, 11, 15], it becomes clear that the leftover hash lemma is thoroughly unsuitable for practical use.

The problem is that universal hash functions (and their generalizations, e.g., extractors) are designed for use in de-randomization, where the scarce resource is uniformly-distributed randomness, and where non-uniformly distributed bits are very cheap. In contrast, for passphrase-based cryptography, *secret randomness* (e.g., passwords, passphrases, etc.) should be considered a very precious resource that must be conserved at all costs, whereas *public randomness* (even uniformly-distributed public randomness) is nearly free. This suggests that new approaches may be required, and we leave this as an interesting challenge for further study.

8 Acknowledgements

We would like to gratefully acknowledge the contributions of Mihir Bellare and Phil Rogaway (for their helpful observations and careful commentary), Dan Boneh (who pointed out the relevance of the leftover hash lemma), David Zuckerman (who contributed the simplified example used in showing the tightness of Theorem 4), Umesh Vazirani (for helpful discussions on extractors and entropy smoothing), and the anonymous referees for CRYPTO 2000 and ASIACRYPT 2000.

References

1. M. Bellare, J. Kilian, P. Rogaway, "The Security of the Cipher Block Chaining Message Authentication Code," *CRYPTO '94*, Springer-Verlag, 1994.
2. M. Bellare, R. Canetti, H. Krawczyk, "Keying hash functions for message authentication,", *CRYPTO '96*, Springer-Verlag, 1996.
3. M. Bishop, D.V. Klein, "Improving system security via proactive password checking," *Computers & Security*, vol.14, (no.3), 1995, pp.233–249.
4. Electronic Frontier Foundation, *Cracking DES: Secrets of Encryption Research, Wiretap Politics, and Chip Design*, O'Reilly, 1998.
5. D.C. Feldmeier and P.R. Karn, "UNIX password security—ten years later." *CRYPTO '89*, Springer-Verlag, 1990, pp.44–63.
6. O. Goldreich, "Foundations of Cryptography (Fragments of a Book)," Chapter 3.
7. S. Goldwasser, M. Bellare, "Lecture Notes on Cryptography," available online from `http://www-cse.ucsd.edu/users/mihir/papers/gb.html`.
8. J. Hastad, R. Impagliazzo, L.A. Levin, M. Luby, "A pseudorandom generator from any one-way function," *SIAM Journal on Computing*, vol.28 no.4, 1999.
9. J. Hietaniemi, "ipasswd—proactive password security," *Proc. 6th Systems Administration Conf. (LISA VI)*, USENIX Association, 1992, pp.105–114.
10. R. Impagliazzo and D. Zuckerman, "How to Recycle Random Bits," *FOCS '89*, IEEE Press, 1989.
11. D.V. Klein, "Foiling the cracker: a survey of, and improvements to, password security," *USENIX Workshop Proceedings: UNIX Security II*, USENIX Assoc., 1990.
12. M. Luby, C. Rackoff, "A study of password security," *CRYPTO '87*, Springer-Verlag, 1988.
13. M. Luby, C. Rackoff, "A study of password security," *J. Cryptology*, vol. 1 no. 3, 1989.
14. R. Morris, K. Thompson, "Password security: a case history," *Communications of the ACM*, vol. 22, no. 11, Nov. 1979.
15. T. Wu, "A real-world analysis of Kerberos password security," *Proc. 1999 Network and Distributed System Security Symp.*, Internet Soc., 1999, pp.13–22.
16. D. Zuckerman, personal communication, July 1999.