Secure Human Identification Protocols

Nicholas J. Hopper and Manuel Blum

Computer Science Department, Carnegie Mellon University, 5000 Forbes Ave.
Pittsburgh PA 15213, USA
{hopper,mblum}@cs.cmu.edu

Abstract. One interesting and important challenge for the cryptologic community is that of providing secure authentication and identification for unassisted humans. There are a range of protocols for secure identification which require various forms of trusted hardware or software, aimed at protecting privacy and financial assets. But how do we verify our identity, securely, when we don't have or don't trust our smart card, palmtop, or laptop?

In this paper, we provide definitions of what we believe to be reasonable goals for secure human identification. We demonstrate that existing solutions do not meet these reasonable definitions. Finally, we provide solutions which demonstrate the feasibility of the security conditions attached to our definitions, but which are impractical for use by humans.

1 Introduction

Consider the problem of human identification. A human H wishes to prove his identity to a computational device C. The channel over which H and C will communicate is insecure and possibly controlled by an adversary. The protocol which accomplishes this task must satisfy the property that no adversary, even one who has witnessed past identifications, may successfully impersonate H except with negligible probability. Complicating matters further, H and C would like to reuse the secret they share for many identifications.

This problem arises on a daily basis in our society, yet the solutions to date are inadequate for several reasons. The traditional password approach is unacceptable, since a network snoop can record the password and will then be able to falsely authenticate as the user at will. Schemes which build a cryptographically strong key from some initial weak secret, such as SRP and EKE, require trusted hardware and software, since the computations involved are far beyond the abilities of most humans. Zero-knowledge schemes such as Fiat-Shamir [1] require trusted hardware which can be stolen or compromised. One-time passwords [2] are just that – good for only a single authentication; pads of such passwords are vulnerable to theft and still require a large ratio of "key material" to authentications.

These schemes all require the human to have some computational or memory aid to securely authenticate himself. In this paper we seek a solution that is viable for the traveler who lost his luggage, or the purchaser who forgot his wallet. We believe that practical scenarios such as these justify the need for such a solution.

An alternative to the above schemes (SRP, EKE, Fiat-Shamir, one-time passwords) is a challenge-response protocol:

- The user and computer share a secret.
- The computer randomly challenges the user
- The user responds in such a way that an adversary cannot easily learn the secret.

Papers by Matsumoto and Imai [3], Wang *et al* [4], and Matsumoto [5] provide schemes which are sufficient for a small number of authentications. In their case, the secret can be recovered in polynomial time once a linear (in the size of the secret) number of authentications have been witnessed by an eavesdropper. (In our case, the number of authentications that must be witnessed to recover a secret in polynomial time is quadratic in the size of the challenge, which in turn is superpolynomial in the size of the secret.) Naor and Pinkas [6] give an identification protocol which is secure for a number of identifications which is linear in the size of the challenge and which requires a low-tech hardware item: a transparency. If stolen, the transparency can be copied and used to masquerade successfully as the legitimate user.

It is the goal of this paper to suggest that protocols which allow unaided humans to identify themselves securely and repeatedly may be feasible and should be a goal of the cryptographic community. In Section 2 we provide security definitions which we contend should be the goal of human identification protocols. In Section 3, we give examples of some cryptographic primitives which humans can execute without assistance. Section 4 gives a protocol which is provably secure against eavesdropping adversaries, based on these primitives; Section 5 outlines a protocol which is heuristically secure against arbitrary adversaries. This protocol is composed of a small number of steps that are individually feasible for humans. As a whole, however, the protocol requires too much computation (and possibly too much memory) to be practical for most humans.

2 Definitions

We begin by formally defining the notion of an identification protocol, and what we will mean for a protocol to be human executable. We then define two notions of security, in terms of passive and active adversaries. Finally we show how some traditional solutions to this problem either fail to satisfy the conditions of human execution or security.

2.1 Human Identification Protocols

We follow [7] in defining a protocol as a pair of (public, probabilistic) interacting programs (H, C) with auxiliary inputs; we denote the result of interaction between H and C with inputs x and y as $\langle H(x), C(y)\rangle$ and we denote the transcript of bits exchanged during their interaction by $T(H(x), C(y))$. A protocol yields some form of identification if H and C accept with high probability when run with the same auxiliary input and reject with high probability when run with different auxiliary input.

Definition 1. *An* identification protocol *is a pair of probabilistic interactive programs (H, C) with shared auxiliary input z, such that the following conditions hold:*

- *For all auxiliary inputs z, $Pr[\langle H(z), C(z)\rangle = \texttt{accept}] > 0.9$*
- *For each pair $x \neq y$, $Pr[\langle H(x), C(y)\rangle = \texttt{accept}] < 0.1$*

When $\langle H, C\rangle = \texttt{accept}$, we say that H verifies his identity to *C, C* authenticates *H, or H* authenticates to *C.*

In this paper we are interested in the case where H can be executed by a human. For the reasons outlined in Section 1, we rule out any form of computational aid. Additionally, we allow for occasional human error and varying abilities of the human population:

Definition 2. *An identification protocol (H, C) is said to be (α, β, t) -* human executable *if at least a $(1-\alpha)$ portion of the human population can perform the computations H unaided and without errors in at most t seconds, with probability greater than $1-\beta$.*

An ultimate goal might be to design a $(.1, .1, 10)$-human executable identification protocol that also meets the security definitions defined subsequently; the protocols we give here are on the order of $(.9, .2, 300)$-human executable, which is clearly not practical as a replacement for traditional solutions to the problem. Still, since they meet our security conditions, we believe they provide evidence that such a protocol is feasible.

A practical issue concerns whether the claim "(H, C) is (α, β, t)-human executable" can be demonstrated. Since we lack a well-defined model of human computation, establishing the claim rigorously seems infeasible in most cases. However, we believe that for the present, in many cases such claims can be evaluated intuitively. In cases where they cannot, empirical evidence should suffice.

2.2 Security Definitions

We give both a weak characterization of security, in terms of *passive* adversaries, and a strong characterization of security, in terms of *active* adversaries. Both characterizations are parameterized by a pair (p, k) where p gives the probability that a computationally bounded attacker can successfully simulate H to C after k interactions with H and/or C.

Definition 3. *An identification protocol* (H, C) *is* (p, k)-secure against passive adversaries *if for all computationally bounded adversaries* $\mathcal{A}$*,*

$$Pr[\langle \mathcal{A}(T^k(H(z), C(z))), C(z)\rangle = \texttt{accept}] \leq p \ ,$$

where $T^k(H(z), C(z))$ *is a random variable sampled from* k *independent transcripts* $T(H(z), C(z))$*.*

That is, even after a passive adversary has witnessed k identification sessions between H and C, he still cannot successfully masquerade as H with probability greater than p. A passive adversary models the eavesdropper or "shoulder-surfer" who is willing to watch H identify himself but does not control the communication channel between H and C. On the other hand, an active adversary is permitted to control the channel between H and C, which leads to a much stronger definition of security.

Definition 4. *An identification protocol* (H, C) *is* (p, k)-secure against active adversaries *if for all computationally bounded adversaries* $\mathcal{A}$*,*

$$Pr[\langle \mathcal{A}(T^k(\mathcal{A}, H(z), C(z))), C(z)\rangle = \texttt{accept}] < p \ ,$$

where $T^k(\mathcal{A}, H(z), C(z))$ *denotes a random variable sampled from* k *sessions where* $\mathcal{A}$ *is allowed to observe and make arbitrary changes to the communications between* H *and* C*.*

This last definition is a theoretical goal which in practice is not achieved by any existing solution to this problem, except for the case $k = 1$. For example, most password-based protocols may be compromised in one authentication by a trojan horse which records the user's password before performing (or failing to perform) the computational steps involved. Therefore, we will relax this condition as follows. We will allow a third outcome for the interaction of H and C (and any third parties): we will allow H to reject C. This will be denoted by $\langle H(\cdot), C(\cdot)\rangle = \perp$. Our relaxed security requirement is that after eavesdropping on k identification sessions, $\mathcal{A}$ still has probability at most q of interacting with H and C without being detected:

Definition 5. *An identification protocol* (H, C) *is* (p, q, k)-detecting against active adversaries *if for all computationally bounded adversaries* $\mathcal{A}$*,*

- $Pr[\langle H(z), \mathcal{A}(T^k(H(z), C(z)))\rangle \neq \perp] < q$
- $Pr[\langle \mathcal{A}(T^k(H(z), C(z))), C(z)\rangle = \texttt{accept}] < p$.

In this setting, we deprive the adversary $\mathcal{A}$ of the opportunity to interfere with communication between H and C. For a protocol satisfying this security condition, H should consider his communications with C to be compromised once H rejects C, and should not respond to any further authentication requests until the parties may securely exchange a new secret z'.

We note that in the human-executable setting some parameters may be relaxed when compared with computationally intensive protocols for identification. For example, a "standard" cryptographic goal for an identification protocol might be a protocol which is $(2^{-m}, 2^{-m}, 2^{100})$-detecting against active adversaries. But when a human is providing the transcripts for $T^k(C(z), H(z))$ it is quite reasonable to expect that security for 10^6 authentications will be sufficient, since a human would take decades to provide so many. Further, many applications which require human authentication are apparently more tolerant to false positives; for example, most automated teller machines have a confidence level of only 10^{-4}. Thus a $(10^{-6}, 10^{-6}, 10^7)$-detecting protocol may be acceptable for humans.

3 Plausible Hard Problems

In this section we introduce two computational problems as candidates for constructing secure human executable authentication protocols, along with some evidence that these computational problems are hard. Both problems can be characterized as loosely based on the sparse subset sum problem, taken over vectors of digits, with some twists intended to allow more authentications.

3.1 Learning Parity in the Presence of Noise

Suppose the secret shared between the human and the computer is a vector $\mathbf{x}$ of length n over $GF(2)$. Authentication proceeds as follows: The computer, C, generates a random n-vector $\mathbf{c}$ over $GF(2)$ and sends it to the human, H, as a challenge. H responds with the bit $r = \mathbf{c} \cdot \mathbf{x}$, the inner product over $GF(2)$. C accepts if $r = \mathbf{c} \cdot \mathbf{x}$. Clearly on a single authentication, C accepts a legitimate user H with probability 1, and an impostor with probability $\frac{1}{2}$; iteration k times results in accepting an impostor with probability 2^{-k}. Unfortunately, after observing $O(n)$ challenge-response pairs between C and H, the adversary M can use Gaussian elimination to discover the secret $\mathbf{x}$ and masquerade as H.

Suppose we introduce a parameter $\eta \in (0, \frac{1}{2})$ and allow H to respond incorrectly with probability η; in that case the adversary can no longer simply use Gaussian elimination to learn the secret $\mathbf{x}$. This is an instance of the problem of *learning parity with noise* (LPN). In fact the problem of learning $\mathbf{x}$ becomes NP-Hard in the presence of errors; it is NP-Hard to even find an $\mathbf{x}$ satisfying more than half of the challenge-response pairs collected by M [8]. Of course, the hardness results of Håstad [8] simply imply that there exist instances of this problem which cannot be solved in polynomial time unless P=NP; it is still possible that the problem is tractable in the random case. However, Kearns[9] has shown that in the random case, parity is not efficiently learnable in the statistical query model; and all known efficient learning algorithms for noisy concepts can be cast in this model. Additionally, Blum *et al* [10] show that for the case of uniformly distributed challenges, weak prediction is equivalent to strong prediction – that is, any algorithm to predict the next response bit with probability

$\frac{1}{2} + \frac{1}{n^c}$ can be used to recover the underlying parity function; and any algorithm which can learn LPN when the parity function is chosen uniformly can be used to learn arbitrary parity functions. Further, the best known algorithm for the general random problem, due to Blum, Kalai and Wasserman, requires $2^{\Omega(n/\log n)}$ challenge-response pairs and works in time $2^{\Omega(n/\log n)}$; here we will give some evidence that this problem is, in fact, uniformly hard and cannot be solved in time and sample size $poly(n, 1/(\frac{1}{2} - \eta))$.

In the following, we will refer to an instance of LPN as a $m \times n$ matrix $\mathbf{A}$ (where $m = poly(n)$); a m-vector $\mathbf{b}$, and a noise parameter η; the problem is to find a n-vector $\mathbf{x}$ such that $|\mathbf{A}\mathbf{x} - \mathbf{b}| \leq \eta m$, where $|\mathbf{x}|$ denotes the Hamming weight of the vector $\mathbf{x}$.

Lemma 1. *(Pseudo-randomizability)*
Any instance of LPN can be transformed in polynomial time into an instance chosen uniformly at random from a space of 2^{n^2} *possibilities.*

Proof: Choose the $n \times n$ matrix $\mathbf{R} \in_U \{0,1\}^{n^2}$; Then if there is a solution to the instance $(\mathbf{AR}, \mathbf{b}, \eta)$, say $\mathbf{y}$, then we have:

$$|(\mathbf{AR})\mathbf{y} - \mathbf{b}| \leq \eta m \ ,$$

and if we let $\mathbf{x} := \mathbf{R}\mathbf{y}$ we find that $\mathbf{A}\mathbf{x} = \mathbf{A}(\mathbf{R}\mathbf{y}) = (\mathbf{AR})\mathbf{y}$, which yields the desired $\mathbf{x}$, since:

$$|\mathbf{A}\mathbf{x} - \mathbf{b}| = |(\mathbf{AR})\mathbf{y} - \mathbf{b}| \leq \eta m \ .$$

Thus there is a polynomial-time transformation between adversarial instances and and a large class of random instances, such that a solution to the randomly chosen instance can be transformed into a solution to the adversarial instance. Phrased differently, each instance of LPN belongs to a space of $O(2^{n^2})$ instances such that either all of the instances are easy or only a negligible fraction are easy. This is similar to the situation with discrete logarithms, where either all of the instances modulo a given prime are easy, or only a negligible fraction are easy.

Lemma 2. *(Log-Uniformity)*
If there exists an algorithm $\mathcal{A}$ capable of solving a $1/poly(n)$ fraction of the instances $(\mathbf{A}, \mathbf{b}, \eta)$ of LPN in time $poly(n, \log(1/(\frac{1}{2} - \eta)))$, then with high probability, any instance can be solved in time $poly(n, \log(1/(\frac{1}{2} - \eta)))$.

Proof: Let $\epsilon(\eta) = \frac{1}{2} - \eta$, and let $\mathcal{A}$ be an algorithm which solves random instances in time $poly(n, \log(1/\epsilon(\eta)))$. Let $(\mathbf{A}, \mathbf{b}, \eta)$ be an adversarial instance of LPN. Create the new instance $(\mathbf{A}', \mathbf{b}', \eta')$ as follows:

- For each row of $\mathbf{A}$, randomly choose n other rows of A and use the sum of these rows as the corresponding entry in $\mathbf{A}'$
- Fill in the corresponding entry in $\mathbf{b}'$ by adding the corresponding rows of $\mathbf{b}$.
- Set $\eta' := \frac{1}{2} - \frac{1}{2}(1 - 2\eta)^{n+1}$

Given the error rate η in the initial instance, the error rate η' is correct, by the following lemma (due to Blum, Kalai, and Wasserman):

Lemma 3. *Let $(a_1, b_1), \ldots, (a_s, b_s)$ be samples from $(\mathbf{A}, \mathbf{b}, \eta)$; then $b_1 + \ldots + b_s$ is the correct label for $a_1 + \ldots + a_s$ with probability $\frac{1}{2} + \frac{1}{2}(1 - 2\eta)^s$.*

The proof follows by induction on s [11]. The resulting instance is distributed uniformly; so with probability $1/\mathit{poly}(n)$, $\mathcal{A}$ solves it in time $\mathit{poly}(n, \log(1/\epsilon(\eta')))$. But note that:

$$\begin{aligned}\epsilon(\eta') &= \frac{1}{2}(1 - 2\eta')^{n+1} \\ &= \frac{1}{2}(1 - 2(\frac{1}{2} - \epsilon(\eta)))^{n+1} \\ &= \frac{1}{2}(2\epsilon(\eta))^{n+1}\end{aligned}$$

so that $\mathit{poly}(n, \log(1/\epsilon(\eta'))) = \mathit{poly}(n, \log(1/\epsilon(\eta)))$; since the expected number of attempts to find an instance soluble by $\mathcal{A}$ is $\mathit{poly}(n)$, $\mathcal{A}$ solves adversarial instances in time $\mathit{poly}(n, \log(1/\epsilon(\eta)))$.

Conjecture 1. (Hardness of LPN)
LPN is uniformly hard in n and η: there is no algorithm to solve a uniformly chosen instance $(\mathbf{A}, \mathbf{b}, \eta)$ in time $\mathit{poly}(n, 1/(\frac{1}{2} - \eta))$ with non-negligible probability.

Evidence:

- (LPN) is not efficiently learnable in the statistical query model; combined with the uniformity results of Blum *et al* this suggests that uniformly chosen inputs are hard.
- The best known algorithm for the random case, given by Blum, Kalai, and Wasserman, has superpolynomial complexity.
- Lemmas 1 and 2.

This assumption is not unprecedented: the McEliece public-key cryptosystem [12] relies on a related assumption, and the pseudo-random generator proposed by Blum, Furst, Kearns and Lipton [10] is secure under a very similar assumption.

In adapting this problem to use by humans, we restrict the hamming weight of the secret vector x to be k, where k is roughly logarithmic in n, the length of the challenge. Rather than taking the inner product of vectors over $GF(2)$, challenges are vectors of decimal digits, and responses are the sum without carries (i.e., modulo 10) of the digits in the positions corresponding to the non-zero entries of x. Our best algorithm for solving instances of this related problem has complexity $\binom{n}{k/2}$. The algorithm proceeds by evaluating all possible hamming-weight $k/2$ vectors on the challenges, and applying hashing to find pairs of vectors which sum to the correct response on roughly a fraction $1 - \eta$ of challenges. Note that while this attack is better than the brute force approach of guessing all weight-k vectors – which has complexity $\binom{n}{k}$ – the complexity is still superpolynomial when k is logarithmic in n.

3.2 Sum of k Mins

Let $z = \langle (x_1, y_1), (x_2, y_2), \ldots, (x_k, y_k) \rangle$ be a set of pairs (x_i, y_i) of integers mod n. Let $\mathbf{v} \in \{0, \ldots, 9\}^n$, and define $f(\mathbf{v}, z)$ by:

$$f(\mathbf{v}, z) = \sum_{i=1}^{k} \min\{\mathbf{v}[x_i], \mathbf{v}[y_i]\} \bmod 10 \ .$$

Then the sum of k mins problem is: given m pairs $(\mathbf{v_1}, u_1), \ldots (\mathbf{v_m}, u_m)$, where $\mathbf{v_i} \in \{0, \ldots, 9\}^n$, $u_i \in \{0, \ldots, 9\}$, and $k \log_{10} n \leq m \leq \binom{n}{2}$, find a set z such that $u_i = f(\mathbf{v_i}, z)$ for all $i = 1, \ldots, m$.

An algebraic approach to this problem is to form the system of equations given by:

$$\begin{bmatrix} v_{1,1,2} & v_{1,1,3} \cdots & & & v_{1,n-1,n} \\ v_{2,1,2} & \cdots & v_{2,i,j} & \cdots & \\ \vdots & & & \ddots & \\ v_{m,1,2} & \cdots & v_{m,i,j} & \cdots & v_{m,n-1,n} \end{bmatrix} \begin{bmatrix} z_{1,2} \\ \vdots \\ z\vdots \\ z_{i,j} \\ \vdots \\ z_{n-1,n} \end{bmatrix} = \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_m \end{bmatrix} \pmod{10} \ ,$$

where $v_{k,i,j} = \min\{\mathbf{v_k}[i], \mathbf{v_k}[j]\}$, $z_{i,j} = 1$ if $(i,j) \in z$, and $1 \leq i < j \leq n$. If $m \geq \binom{n}{2}$ we expect to solve this system uniquely by Gaussian elimination. When $m < \binom{n}{2}$, on the other hand, this approach leads to a sparse subset sum problem. The best known algorithms for these instances have complexity roughly $\binom{n(n-1)/2}{k/2}$ (which is greater than $\binom{n}{k}$ when $k > 3$).

Another approach to the problem is a form of maximum-likelihood estimation (MLE): for some subset of the locations in z, try all possible values, while modeling the remaining inputs to $f(\cdot, z)$ as uniform random variables (an accurate model when the $\mathbf{v_i}$ are chosen at random). Choose the subset of locations which gives the best chance of observing the output values u_i. If the subset of z we are guessing has l locations, this algorithm has complexity $\binom{n}{l}$. However, to succeed in selecting correct locations, the algorithm may require many samples (perhaps more than $\binom{n}{2}$).

For any distribution $\mathcal{D}$, the maximum probability of distinguishing between $\mathcal{D}$ and the uniform distribution on the same range U is $\Delta(\mathcal{D}, U)$, where

$$\Delta(A, B) = \frac{1}{2} \sum_{e \in E} |Pr_A[e] - Pr_B[e]| \ ,$$

i.e., the statistical distance between A and B. Thus the expected minimum number of samples required to distinguish between $\mathcal{D}$ and U is $1/\Delta(\mathcal{D}, U)$. Therefore calculating this distance for the distribution of the modulo 10 sum of k mins will help us develop lower bounds on the required sample complexity for MLE.

To calculate the statistical distance between k mins and uniformly random digits, we derive an expression which will allow us to calculate the probability of obtaining a digit as the sum of k mins. Let P_d^k denote the probability of obtaining the digit d as a sum of k mins. Then the P_d^1 are easily obtainable by enumerating all pairs of digits. For $k > 1$, we note that for each d, there are 10 ways to obtain d as the sum of k mins: for each digit d', we obtain d' from one min and $d - d'$ mod 10 from the other $k - 1$ mins. In other words, we can write the recurrence $P_d^k = \sum_{0 \leq d' \leq 9} P_{d-d'}^{k-1} P_{d'}^1$, which leads to the observation that dynamic programming is sufficient for obtaining the distribution over k mins. Table 1 yields the result of applying this procedure to calculate the expected minimum sample complexity to distinguish between the uniform distribution on $\{0, \ldots, 9\}$ and a sum of k mins, for $k \leq 12$.

Table 1. Distribution of sum of k mins, and expected minimum number of samples required to distinguish from uniform

k	1	2	3	4	5	6	7	8	9	10	11	12
#S	4	14	44	140	532	1346	4154	12848	39696	122682	379100	1171498

Note that the essential meaning of this table is that without guessing more than 12 locations from a challenge, an adversary cannot expect to use statistical procedures to learn a sum of 12 mins password with fewer than 1,171,498 challenge-response pairs. In general, we can protect against this attack by choosing k such that the number of required samples is greater than $\binom{n}{2}$, since $\binom{n}{2}$ samples are sufficient for Gaussian elimination.

4 Security Against Passive Adversaries

In this section, we will give a protocol which is (p, k)-secure against passive adversaries but not against arbitrary adversaries. We also give some empirical evidence that it is $(0.9, 0.25, 160)$-human executable. Intuitively, C generates the coefficient matrix of some LPN instance while H generates the output vector and some errors. Thus after a number of repetitions C can be reasonably sure that H knows the shared secret vector $\mathbf{x}$.

Protocol 1

Shared Secrets: H and C share a secret 0-1 vector $\mathbf{x}$ with $|\mathbf{x}| = k$.

Authentication:

```
(C1) $C$ sets $i := 0$
 – Repeat $m$ times:
     (C2) $C$ selects a random challenge $\mathbf{c} \in_R \{0,1\}^n$ and sends it to $H$
     (H1) With probability $1-\eta$, $H$ responds with $r := \mathbf{c} \cdot \mathbf{x}$, otherwise $H$ responds
          with $r := 1 - \mathbf{c} \cdot \mathbf{x}$.
     (C3) if $r = \mathbf{c} \cdot \mathbf{x}$, $C$ increments $i$.
(C4) if $i \geq (1-\eta)m$, $C$ accepts $H$.
```

Theorem 1. *If H guesses random responses r, C will accept H with probability at most*

$$\left(\frac{1}{2}\right)^m \sum_{i=(1-\eta)m}^{m} \binom{m}{i} \leq e^{-c_0 m} \ ,$$

where $c_0 \geq \frac{2}{3}$ is a constant depending only on η.

Proof: Let X be the random variable denoting the number of times H guesses correctly; since this probability is at most $\frac{1}{2}$, the probability of guessing correctly exactly i times out of m is $\binom{m}{i}\left(\frac{1}{2}\right)^m$; the first result follows from summing the probabilities of guessing correctly $(1-\eta)m$ or more times; the second result follows by a Chernoff bound with $c_0 = (3-2\eta)^2/6 \geq (3-1)^2/6 = \frac{2}{3}$.

Theorem 2. *If LPN is hard, then Protocol 1 is $(e^{-\frac{2}{3}m}, poly(n))$ - secure against a passive adversary.*

Proof: Obvious. Since a passive adversary can only observe challenge-response pairs $(\mathbf{c}, r)$, obtaining the secret $\mathbf{x}$ can only be accomplished via solving the LPN problem.

Unfortunately, as previously mentioned, this protocol is not secure against an active adversary: suppose M can insert arbitrary challenges into the interaction; then M can record $n/m(1-\eta)^2$ successful authentications and replay them back to H, discarding $(\mathbf{c}, r)$ pairs which do not match; the remaining pairs will have no errors and can be solved by Gaussian elimination. Additionally, this protocol must be iterated many times in order to achieve any sort of security.

As an additional consideration for the human user, the challenges $\mathbf{c}$ could be selected from $\{0, \ldots, 9\}^n$ and the arithmetic done modulo 10, a natural base for many humans. This reduces the number of iterations necessary for a given security level by a constant factor. It also requires modifying the method of making an error: in cases when an error is to be made, the response should be chosen uniformly from $\{0, \ldots, 9\}$.

Note that assuming the best known attack complexity is optimal, we can choose parameters which will provide ample security in this setting. For example, when $n = 1000$ and $k = 19$, the best known attack's complexity of $\binom{n}{\lceil k/2 \rceil}$ is roughly 2^{78}. This compares favorably with common minimum strength guidelines for choosing cryptographic parameters.

To assess the property of human executability, we conducted the following experiment. A computer implementing this authentication system with $m = 7$, $\eta = \frac{1}{7}$, $n = 200$ and $k = 15$ was attached to a Coke machine in our department's lounge. The system was also implemented as a web page, which provided a tutorial in its use. Students and faculty were permitted to access the web page as often as they wished, and a free Coke was given to anyone who could successfully authenticate himself to the computer attached to the coke machine. In a one week period, 54 users attempted 195 authentications and successfully completed 155. The average time per successful authentication was 166 seconds, and the average time per unsuccessful authentication was 171 seconds. Thus it is empirically clear that there is some value α for which this is a $(\alpha, .25, 160)$-human executable identification protocol which is secure against computationally bounded eavesdropping adversaries.

5 Security Against Arbitrary Adversaries

The protocol of the previous section is quite insecure against an adversary who is capable of modifying the communications between H and C. For example, by simply replaying the same challenge $\mathbf{r}$ back to H several times, $\mathcal{A}$ can compute the true value of $\mathbf{r} \cdot \mathbf{x}$ and thus after collecting n such error-free values, can learn the secret $\mathbf{x}$ by Gaussian elimination. Even if we simply replace weight-k LPN by sum of $\lceil k/2 \rceil$ mins, the problem persists. That is, while simply replaying the same challenge to H will no longer allow $\mathcal{A}$ to learn the secret z, replaying the same challenge with a slight change — for example, changing a single '9' to a '0' — will still allow $\mathcal{A}$ to learn the secret z with $O(n)$ well-chosen challenges.

Thus we seek to make it difficult for $\mathcal{A}$ to submit arbitrary challenges to H in place of those sent by C. To do so, we will introduce two mechanisms. First, *Error-Correcting Challenges* have the property that it with very high probability a challenge cannot be modified in a small number of locations. Second, we require the challenges to satisfy some concept which is hard to learn without membership queries, such as satisfying $f(\mathbf{r}, z) = 0 \bmod 10$ for an independent k-mins password z.

5.1 Error-Correcting Challenges

Blum *et al.* [13] show how a function which is linear with probability $1 - \delta$ can be self-corrected to a linear function which matches the given function with probability $1 - 2\delta$. Self-correction of this form is used in many Probabilistically Checkable Proof (PCP) arguments. We propose that a similar error-detecting/self-correcting approach can be applied to the challenges in our system, resulting in a system which has the property that with high probability an adversary cannot make local changes to a challenge.

The protocol proposed in this document will use the self-correction algorithm of [13] to achieve this goal. A legitimate challenge will consist of $w \times h$ 10×10 squares of digits, or $n = 100wh$ digits. Each square will be generated by choosing 3 digits (a, b, c) uniformly at random; then the digit at location x, y will have the value $L(x, y) = ax + by + c \bmod 10$. Linearity can be tested by choosing a random point $\boldsymbol{x}$ (mod 10) and random offset mod 10, $\boldsymbol{r}$, and testing whether $L(\boldsymbol{x}) = L(\boldsymbol{x}+\boldsymbol{r}) - L(\boldsymbol{r}) + L(0)$. If a challenge square passes this test several times then we say that it is close to linear, and in the subsequent phase we will access the value of a location $\boldsymbol{x}$ by accessing its "self-corrected" value at the randomly chosen offset $\boldsymbol{r}$, which is given by $L(\boldsymbol{x} + \boldsymbol{r}) - L(\boldsymbol{r}) + L(0)$. Thus if we reject a challenge which contains a highly non-linear square and self-correct otherwise, with high probability an adversary will be unable to effect a local change to a challenge.

5.2 The Protocol

Coupled with a deterministic response protocol to prevent replay attacks, we obtain the protocol outlined below.

Protocol 2

Shared Secrets: H and C share two sum of k mins secrets p_1 and p_2, and a secret digit d. As in Section 3, we denote by $f(c, p_i)$ the result of taking the sum of the self-corrected min of each pair in p_i for the challenge c.

Authentication: Repeat m times for confidence 10^{-m}:

(C1) Uniformly pick wh sets of parameters (a_i, b_i, c_i) and form the error-correcting challenge for these parameters, $c = ECC(\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c})$. If $f(c, p_1) \neq d$, repeat until the condition holds. Send the resulting challenge to H:

$$C \to H \ : \ c = ECC(\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}) \ .$$

(H1) Test each square for linearity. Reject if any square is not close to linear. (Report a network infiltration to system administrator and choose a new password)

(H2) Check that $f(c, p_1) = d$. If not, reject and report a network infiltration to system administrator.

(H3) Respond with the self-corrected sum of mins for the password p_2:

$$H \to C \ : \ r = f(c, p_2) \ .$$

(C2) Reject if $r \neq f(c, p_2)$.

C accepts H if it has not rejected after m rounds.

Intuitively, we use self-correction on error-correcting challenges to make it infeasible for an adversary to make local changes to a challenge. Thus, to make a membership query, the adversary must make global changes to the challenge, yet since $f(\cdot, p_1)$ is distributed essentially uniformly any global change will be caught with probability at least 0.9.

Thus heuristically, we have a protocol which is $(0.1, 0.1, \binom{n}{2})$-detecting against computationally bounded adversaries. With the challenge size $n = 900$, $k = 12$ and $m = 6$, the best known attack on sum of k mins given fewer than $\binom{n}{2}$ samples has complexity greater than $\binom{900}{12}$, which is roughly 2^{89}. Thus the security of the system appears to be quite high.

It seems reasonable that a human can learn to do linearity testing on sight, since error-correcting challenges form distinctive patterns of digits; thus the human computational load in this protocol may be as low as 96 base 10 sums and 24 mins to compute the response to a single challenge. For confidence 10^{-6}, this translates to a protocol which requires a minimum of 576 base 10 sums plus considerable search effort. Therefore, while this protocol offers a great deal of security against arbitrary computationally bounded adversaries, it seems unlikely to be of practical significance on its own.

6 Some Inherent Limitations

The approach to human-executable primitives taken here has some inherent limitations which may, unfortunately, make it difficult to improve on these protocols without a new approach. We now consider a large class of "similar" protocols in which the shared secret is a set of k of relevant locations in a n-digit challenge. We will show that, assuming the "meet in the middle" attack of time complexity $O(\binom{n}{k/2})$ is optimal, there is no significantly harder function in this class, computationally speaking, than parity with noise.

We model the human as a finite automaton which sequentially processes k inputs by transitions between states in the range $\{1, \ldots, Q\}$ and which gives an output in the range $\{1, \ldots, d\}$. Since humans have highly bounded memory, this model seems fitting for human computation in this application. We assume that the transition table for this automaton may change between inputs but is publicly known.

Now consider an attack which uses m challenge-response pairs and processes all sequences of locations of length $k/2$. For each location sequence, the automaton is run forward $k/2$ steps, producing a string in the range $\{1, \ldots, d\}^m$. This string and its location are inserted in a hash table with Q^m spaces. Also, for each sequence of $k/2$ locations, for each challenge, the automaton is started from each intermediate state $\{1, \ldots, Q\}$ and the list of intermediate states which produced the correct response is retained. For each challenge, the expected number of intermediate states retained will be Q/d. Thus we will expect approximately $(Q/d)^m$ sequences of intermediate states to match the correct responses for each sequence of $k/2$ locations; each of these intermediate state sequences can be inserted into the same hash table of size Q^m. Any match in the hash table between a "first-half" sequence and a "second-half" sequence suggests a length k sequence of locations which matches on the m challenge-response pairs under consideration; such a sequence can be tested against the $O(k \log_d n)$ challenge-response pairs required to uniquely determine the secret k locations.

Now we assess the total computational work factor for this attack. First, for each sequence of $k/2$ locations, $(Q/d)^m$ length-m sequences must be inserted into the hash table, for a total of $O((Q/d)^m n^{k/2})$ work. Also, each collision between a "first-half" sequence and a "second-half" sequence will require some work to check against the full set of challenge-response pairs. For an appropriate family of universal hash functions, the expected number of collisions will be

$$\frac{n^{k/2} \times (Q/d)^m n^{k/2}}{Q^m} = \frac{n^k}{d^m} \ .$$

Choosing m to minimize the sum $(Q/d)^m n^{k/2} + \frac{n^k}{d^m}$ results in the choice

$$m = \frac{\log \frac{n^{k/2} \log d}{\log(Q/d)}}{\log Q} \ ,$$

and gives the total work factor

$$O(n^{k(1 - \frac{\log d}{2 \log Q})}) \ .$$

Thus if d and Q are close, or equal as in our protocols, an attacker can always break such a protocol by guessing only about half of the shared secret. On the other hand, decreasing d relative to Q increases the number of challenges a user must respond to for a given confidence level, while increasing Q adds to the cognitive load on the human. Thus while some incremental improvement in the computational security of our protocols may be possible, overall our choice of primitives represent a close to optimal tradeoff between computational difficulty and human cognitive load for this class of protocols.

7 Conclusions

We believe that the search for protocols providing secure, reusable authentication to unaided humans is an interesting and important pursuit for the cryptographic community. In this paper, we have shown that no current solutions to this problem exist. We have provided definitions that we believe are reasonable goals for such protocols, and we have given protocols which achieve the security conditions attached to these goals. While we do not argue that the protocols we present are practical solutions to this problem – executing the protocols and remembering the secrets seem too hard – we believe that they are surprisingly close to practical while offering a good deal of security. Thus we believe that they suggest that more practical solutions may exist, which can match or even exceed their security conditions. We invite the reader to surpass them.

Acknowledgements

The authors wish to thank Avrim Blum for several discussions concerning the difficulty of parity with noise, Moni Naor for pointing out that the subset sum meet-in-the-middle attack can still be applied in the presence of noise, and Preston Tollinger for conducting the Coke Machine experiment. This material is based upon work supported under a National Science Foundation Graduate Research Fellowship.

References

1. Fiat, A., Shamir, A.: How to prove yourself: Practical solutions to identification and signature problems. In Odlyzko, A.M., ed.: Advances in Cryptology—CRYPTO '86. Volume 263 of Lecture Notes in Computer Science., Springer-Verlag, 1987 (1986) 186–194
2. Lamport, L.: Password authentication with insecure communication. Communications of the ACM **24** (1981)
3. Matsumoto, T., Imai, H.: Human identification through insecure channel. In Davies, D.W., ed.: Advances in Cryptology—EUROCRYPT 91. Volume 547 of Lecture Notes in Computer Science., Springer-Verlag (1991) 409–421

4. Wang, C.H., Hwang, T., Tsai, J.J.: On the Matsumoto and Imai's human identification scheme. In Guillou, L.C., Quisquater, J.J., eds.: Advances in Cryptology—EUROCRYPT 95. Volume 921 of Lecture Notes in Computer Science., Springer-Verlag (1995) 382–392
5. Matsumoto, T.: Human-computer cryptography: An attempt. In Neuman, C., ed.: 3rd ACM Conference on Computer and Communications Security, New Delhi, India, ACM Press (1996) 68–75
6. Naor, M., Pinkas, B.: Visual authentication and identification. In Kaliski Jr., B.S., ed.: Advances in Cryptology—CRYPTO '97. Volume 1294 of Lecture Notes in Computer Science., Springer-Verlag (1997) 322–336
7. Goldreich, O.: Foundations of cryptography (fragments of a book). Available electronically at `http://theory.lcs.mit.edu/ oded/frag.html` (1998)
8. Håstad, J.: Some optimal inapproximability results. In: Proceedings of the Twenty-Ninth Annual ACM Symposium on Theory of Computing, El Paso, Texas (1997) 1–10
9. Kearns, M.: Efficient noise-tolerant learning from statistical queries. In: Proceedings of the Twenty-Fifth Annual ACM Symposium on the Theory of Computing, San Diego, California (1993) 392–401
10. Blum, A., Furst, M., Kearns, M., Lipton, R.J.: Cryptographic primitives based on hard learning problems. In Stinson, D.R., ed.: Advances in Cryptology—CRYPTO '93. Volume 773 of Lecture Notes in Computer Science., Springer-Verlag (1993) 278–291
11. Blum, A., Kalai, A., Wasserman, H.: Noise-tolerant learning, the parity problem, and the statistical query model. In: Proceedings of the Thirty-Second Annual ACM Symposium on Theory of Computing, Portland, Oregon (2000)
12. McEliece, R.J.: A public-key cryptosystem based on algebraic coding theory. Technical report, Jet Propulsion Laboratory (1978) Deep Space Network Progress Report.
13. Blum, M., Luby, M., Rubinfeld, R.: Self-testing/correcting with applications to numerical problems. In: Proceedings of the Twenty Second Annual ACM Symposium on Theory of Computing, Baltimore, Maryland (1990) 73–83