A Secure Human-Computer Authentication Scheme

Nicholas J. Hopper Manuel Blum

May 2000
CMU-CS-00-139

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Abstract

We introduce a protocol for authentication between a human and a computer, where the human is able to use no special hardware other than a dumb terminal. Authentication is based on a shared secret which can be reused polynomially often with no danger of exposure, assuming the conjectured uniform hardness of learning parity functions in the presence of noise. Under this conjecture, the protocol is secure against a polynomially-bounded passive adversary and also some forms of active adversary, although it is not secure against arbitrary active adversaries.

Keywords: Authentication, Human-Computer Cryptography, passwords, Parity with Noise

1 Introduction

Consider the common scenario of a human user attempting to authenticate to a (possibly) remote computer over an insecure connection. 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. Zero-knowledge schemes such as Fiat-Shamir [3] require trusted hardware which can be stolen or compromised. One-time passwords [5] 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 authentication.

An alternative is a challenge-response protocol in which the user and computer have a shared secret that the user can use to respond to the computer's challenges in such a way that an adversary cannot easily learn the secret. Papers by Matsumoto and Imai [7], Wang *et al* [9], and Matsumoto [6] provide schemes which are sufficient for a small number of authentications but are generally vulnerable to an active adversary or require the user to remember a large secret or perform many calculations to achieve a large number of secure authentications.

Ideally, we would like to have a scheme which allows the user to remember a moderately sized secret (of size, say, $poly(\log n)$) and remains secure against an adversary even after many authentications (say, any $poly(n)$ authentications). In this paper, we describe a system based on the problem of learning parity functions in the presence of noise, which achieves these objectives, if the underlying problem is uniformly hard.

2 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 $GF(2)$, where $|\mathbf{x}| = \log n$, that is, $\mathbf{x}$ has $\log n$ positions set to 1, and the rest set to 0. 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 dot-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 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 in the presence of noise* (LPN). In fact the problem of learning $\mathbf{x}$ becomes NP-Hard in the presence of errors; in fact it is NP-Hard to even find an $\mathbf{x}$ satisfying more than half of the challenge-response pairs collected by M [4]. Of course, the hardness results of Hastad [4] 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, the best known algorithm for the general random problem, due to Blum, Kalai and Wasserman, requires $2^{O(n/\log n)}$ challenge-response pairs and works in time $2^{O(n/\log n)}$; here we will give some evidence that this problem is, in fact, uniformly hard and cannot be learned 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 $|\mathbf{A}\mathbf{x} - \mathbf{b}| \leq \eta m$.

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 $\mathbf{x} := \mathbf{Ry}$ we find that $\mathbf{Ax} = \mathbf{A}(\mathbf{Ry}) = (\mathbf{AR})\mathbf{y}$, which yields the desired $\mathbf{x}$, since:

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

Thus there is a polynomial-time transformation between adversarial instances and pseudo-random instances, such that a solution to the pseudo-random instance can be transformed into a solution to the adversarial instance.

Lemma 2 *(Log-Uniformity)*
If a random instance $(\mathbf{A}, \mathbf{b}, \eta)$ of LPN can be solved in time poly$(n, \log(1/(\frac{1}{2} - \eta)))$, than 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 [2]. The resulting instance is distributed uniformly; thus $\mathcal{A}$ solves it in time *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 *poly*$(n, \log(1/\epsilon(\eta'))) =$ *poly*$(n, n\log(1/\epsilon(\eta))) =$ *poly*$(n, \log(1/\epsilon(\eta)))$; thus $\mathcal{A}$ solves adversarial instances in time *poly*$(n, \log(1/\epsilon(\eta)))$.

Conjecture 1 *(Hardness of LPN)*
LPN is uniformly hard in n and η: if there is an algorithm to solve a random instance in time poly$(n, 1/(\frac{1}{2} - \eta))$, then any instance can be solved in time poly$(n, 1/(\frac{1}{2} - \eta))$.

EVIDENCE:

- The best known algorithm for the random case, given by Blum, Kalai, and Wasserman, has exponential complexity.
- Lemmas 1 and 2

This assumption is not unprecedented; the McEliece public-key cryptosystem [8] relies on a related assumption, and the pseudo-random generator proposed by Blum, Furst, Kearns and Lipton [1] is secure under a very similar assumption.

3 The Simple Protocol

This discussion gives rise to the Protocol 1 below; 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

1. *C sets $i := 0$*
2. *Repeat k times:*
   (a) *C selects a random challenge $\mathbf{c} \in_R \{0,1\}^n$ and sends it to H*
   (b) *With probability $1-\eta$, H responds with $r := \mathbf{c}\cdot\mathbf{x}$, otherwise H responds with $r := 1-\mathbf{c}\cdot\mathbf{x}$.*
   (c) *if $r = \mathbf{c}\cdot\mathbf{x}$, C increments i.*
3. *if $i \geq (1-\eta)k$, C accepts H.*

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

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

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 k is $\binom{k}{i}\left(\frac{1}{2}\right)^k$; the first result follows from summing the probabilities of guessing correctly $(1-\eta)k$ or more times; the second result follows by a Chernoff bound with $c_0 = (3-2\eta)^2/6$.

Theorem 2 *If LPN is hard, then Protocol 1 is secure against a passive adversary, for any polynomial number of authentications.*

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, this protocol is not secure against an active adversary: suppose M can insert arbitrary challenges into the interaction; then M can record $n/k(1-\eta)^2$ 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.

There is also one other potential problem in this protocol: can the human H be counted on to make "random enough" errors, that is, to make errors in an unpredictable pattern? Since no hardware is allowed, not even using dice will be satisfactory. We are currently engaged in empirical research to determine whether a human can (or will) produce unpredictable errors.

4 A More Secure Protocol

Protocol 2 describes a more complicated protocol which retains the security features of Protocol 1 but makes fewer computational requirements on H. Intuitively, C and H now share *two* secrets: One secret allows H to compute whether or not an error should be made in using the second secret, and also how to make that error. Thus the requirement of unpredictability is placed on the computer rather than the human, and the number of rounds necessary for a given security level is reduced.

The protocol is given below. As before, H and C will share secret binary vectors of weight $\log n$; however now H and C will share two binary vectors $\mathbf{x}$ and $\mathbf{y}$, and a binary vector $\mathbf{z}$ of weight 1. The protocol follows.

Protocol 2

1. *C sets $i := 0$*
2. *Repeat k times:*
   - (a) *C generates a random challenge $c \in_R \{0,1\}^n$.*
   - (b) *With probability $1-\eta$, C modifies $\mathbf{c}$ so that $\mathbf{c}\cdot\mathbf{y} = \mathbf{c}\cdot\mathbf{z}$, and sets $a := \mathbf{c}\cdot\mathbf{x}$*
   - (c) *Otherwise C modifies $\mathbf{c}$ so that $\mathbf{c}\cdot\mathbf{y} \neq \mathbf{c}\cdot\mathbf{z}$, and sets $a := 1 - \mathbf{c}\cdot\mathbf{x}$.*
   - (d) *C sends $\mathbf{c}$ to H.*
   - (e) *H checks if $\mathbf{c}\cdot\mathbf{y} = \mathbf{c}\cdot\mathbf{z}$; if so, H responds with $r := \mathbf{c}\cdot\mathbf{x}$, otherwise H responds with $r := 1 - \mathbf{c}\cdot\mathbf{x}$.*
   - (f) *C checks if $a = r$ and increments i if true.*
3. *C accepts H if $i = k$.*

Theorem 3 *If H guesses random responses r, C accepts H with probability at most $\left(\frac{1}{2}\right)^k$.*

Proof: H's guesses can be correct with probability at most $\frac{1}{2}$. The result is obvious.

Theorem 4 *(Passive adversary)*
If Conjecture 1 is true, Protocol 2 is secure against any polynomially-bounded passive adversary for any polynomial number of authentications.

Proof: Assume that LPN is hard; then we wish to show that any polynomial-time algorithm $\mathcal{A}$ recovering the secrets $\mathbf{x}$, or $\mathbf{y}$ is also an algorithm to solve LPN. There are three cases:

1. $\mathcal{A}$ ignores $\mathbf{y}$ and attempts to recover $\mathbf{x}$; in that case using the rows of the LPN instance as challenges and the corresponding parity bits as responses produces a transcript with the secret the same as the parity vector sought.

2. $\mathcal{A}$ ignores $\mathbf{x}$ and attempts to recover $\mathbf{y}$; in that case we can produce a transcript by appending the result column of an LPN instance to the matrix; the recovered $\mathbf{y}$ will be the $\mathbf{x}$ from the original LPN instance.

3. $\mathcal{A}$ attempts to recover $\mathbf{x}$ and $\mathbf{y}$ simultaneously. In this case, we create a transcript from an LPN instance $(\mathbf{A}, \mathbf{b}, \eta)$ by appending $\mathbf{b}$ to $\mathbf{A}$ and also using $\mathbf{b}$ as the sequence of responses. The recovered vectors will both be the $\mathbf{x}$ for the LPN instance.

Since LPN is NP-Hard and Conjecture 1 states that random instances are equivalent to adversarial instances, no passive adversary can recover the secrets $\mathbf{x}$ and $\mathbf{y}$ in polynomial time unless P = NP, given the assumption that LPN is uniformly hard.

There are several advantages of Protocol 2 over Protocol 1. First, for a given security parameter and noise rate the second protocol requires fewer iterations. Protocol 2 is also secure against the replay attack mentioned previously, and does not rely on H to produce cryptographically strong randomness. On the other hand, the protocol is still not secure against an active adversary who can submit any challenge; it is an open question whether any protocol exists which a human can execute that is secure against a polynomially bounded active adversary for polynomially many authentications.

5 A Concrete Protocol

A human executing this protocol would most likely conceptualize the vectors $\mathbf{x}$, $\mathbf{y}$ and $\mathbf{z}$ as sets of $\log n$, $\log n$, and 1 challenge locations, respectively; a reasonable question is: to what extent can $\mathbf{x}$ and $\mathbf{y}$ overlap (reducing the memory requirement on the human) without loss of security? We conjecture that the size of the intersection may be as much as $\frac{1}{2}\log n$ with no significant loss of security, but that for every element in the intersection above that threshold, the search complexity for either $\mathbf{x}$ or $\mathbf{y}$ is reduced by a factor of n. Thus this protocol has a secret size of $\frac{3}{2}\log n + 1$ locations; since describing each location requires $\log n$ bits, the secret is $O(\log^2 n)$ bits in length.

Alternatively, note that knowledge of one of $\mathbf{x}$ and $\mathbf{y}$ immediately gives knowledge of the other; thus while $\mathbf{x}$ and $\mathbf{y}$ should not be the same set of locations, it would appear that the security of the scheme would not be threatened by defining one as a simple transformation of the other, say $y_i = x_{i+1 \bmod n}$. Thus H need only remember $\log n + 1$ locations to achieve the same level 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 mod10, 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: when $\mathbf{c} \cdot \mathbf{y} \not\equiv \mathbf{c} \cdot \mathbf{z} \pmod{10}$, set the correct answer $a := \mathbf{c} \cdot \mathbf{y} \bmod 10$ rather than $1 - \mathbf{c} \cdot \mathbf{x}$; this construction apparently preserves the security of Protocol 2 while decreasing the computational load on the human.

Finally, we select concrete parameters to this protocol which we believe will give adequate security. We recommend using the above modifications and Protocol 2 with $n \geq 128$, and using 12 locations each for $\mathbf{x}$ and $\mathbf{y}$. Thus the user will need to remember 19 locations in a 128-element challenge; this will correspond to a security factor of $2^{7*12} = 2^{84}$. For $n = 1000$, the same secret size provides a security factor of $2^{10*12} = 2^{120}$ but the conceptual difficulty of searching a 1000-element challenge may be too difficult for many humans. We also recommend using $k \geq 6$, making an adversary's probability of randomly guessing a correct sequence of responses at most 10^{-6}. Finally, the error rate η should be between 0.2 and 0.8; it can be tuned to optimize the expected number of mod10 additions H must perform.

6 Conclusions

We have given a protocol which we believe a human can execute, having relatively small secret size and an adequate security margin against a passive adversary and some forms of active adversaries. Unfortunately, this protocol is not secure against an active adversary in the sense that it can be compromised in less than $poly(n)$ steps for some polynomial by inserting cleverly chosen challenges. One important question is whether a protocol exists that is secure against such an adversary and can be executed by a human. Another question is whether the presented protocols can be generalized in a way not requiring extensive numerical computation on the part of the human user.

References

[1] Avrim Blum, Merrick Furst, Michael Kearns, and Richard J. Lipton. Cryptographic primitives based on hard learning problems. In Douglas R. Stinson, editor, *Advances in Cryptology—CRYPTO '93*, volume 773 of *Lecture Notes in Computer Science*, pages 278–291. Springer-Verlag, 22–26 August 1993.

[2] Avrim Blum, Adam Kalai, and Hal Wasserman. 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, 21–23 May 2000.

[3] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In A. M. Odlyzko, editor, *Advances in Cryptology—CRYPTO '86*, volume 263 of *Lecture Notes in Computer Science*, pages 186–194. Springer-Verlag, 1987, 11–15 August 1986.

[4] Johan Håstad. Some optimal inapproximability results. In *Proceedings of the Twenty-Ninth Annual ACM Symposium on Theory of Computing*, pages 1–10, El Paso, Texas, 4–6 May 1997.

[5] Leslie Lamport. Password authentication with insecure communication. *Communications of the ACM*, 24(11), November 1981.

[6] Tsutomu Matsumoto. Human-computer cryptography: An attempt. In Clifford Neuman, editor, *3rd ACM Conference on Computer and Communications Security*, pages 68–75, New Delhi, India, March 1996. ACM Press.

[7] Tsutomu Matsumoto and Hideki Imai. Human identification through insecure channel. In D. W. Davies, editor, *Advances in Cryptology—EUROCRYPT 91*, volume 547 of *Lecture Notes in Computer Science*, pages 409–421. Springer-Verlag, 8–11 April 1991.

[8] R. J. McEliece. A public-key cryptosystem based on algebraic coding theory. Technical report, Jet Propulsion Laboratory, 1978. Deep Space Network Progress Report.

[9] Chih-Hung Wang, Tzonelih Hwang, and Jiun-Jang Tsai. On the Matsumoto and Imai's human identification scheme. In Louis C. Guillou and Jean-Jacques Quisquater, editors, *Advances in Cryptology—EUROCRYPT 95*, volume 921 of *Lecture Notes in Computer Science*, pages 382–392. Springer-Verlag, 21–25 May 1995.