Comparing Authentication Techniques

Matt Bishop[1]

Department of Mathematics and Computer Science
Dartmouth College
Hanover, NH 03755

ABSTRACT

In this paper we consider a set of metrics for comparing different authentication schema. We describe a model for an authentication system; unlike other models, this one is oriented towards implementation and not cryptographic properties. We compare several authentication systems within the framework of this model using as metrics the size and probability distribution of authenticators provided by the system, the ease of use, and the ease of abuse by attackers.

Introduction

Designing an authentication scheme requires the integration of very different aspects of computer science and allied sciences; depending on the technique chosen, it could require the use of cryptography, medicine, psychology, systems design and analysis, protocol design, and computer security. All authentication schemes are designed to ensure that a party's identity is proven to another party, and require the party to be authenticated to provide some information (be it a code word, proof of knowledge of an algorithm, or a measurement) to the authenticating party. Such information is called an "authenticator." For the UNIX operating system[2], for example, the authenticator is a password of no more than 8 characters. For privacy-enhanced electronic mail, the authenticator is a private key for the RSA cryptosystem [14]. For a scheme using a biometrical device, the authenticator is the set of measurements the device uses to determine the relevant characteristics about the person.

Four factors dictate the efficacy of an authentication scheme: how uniformly authenticators are distributed over the space of possible authenticators and how large that space is, how easily users can provide their authenticator, how difficult it is for another to provide an authorized user's authenticator, and how costly the authentication mechanism is. In this paper, we shall describe several common authentication schemes and compare and contrast them with respect to the first three criteria. We shall say a few words about the last two afterwards.

Background

An *authentication system* is the embodiment of that information required by both the user and a computer system to authenticate the user with a degree of assurance sufficient to the comput-

1. The support of grant NAG 2-628 from the National Aeronautics and Space Administration is gratefully acknowledged.
2. UNIX is a Registered Trademark of AT&T Bell Laboratories.

```
int isguessed(char *guess, char *hash)
{
     return(strcmp(hash, crypt(hash, guess)) == 0);
}
```

Figure 1. When given a guess `guess` for a user's password and the user's complementary data `hash`, this function returns 1 if the guess is correct and 0 if not. Note that it may be executed by anyone who has the complementary data, as `crypt` is a function in the standard library.

er system (and vice versa). It is composed of a set P of authenticators called *passwords*. Each element $p \in P$ has associated with it *complementary data* $\overline{p} \in \overline{P}$ which is generated by a *complementation function* $c: P \rightarrow \overline{P}$ and is stored on the computer system. The user selects, or is assigned, a password from P using a selection function s. When the user enters a password, the system retrieves the complementary data associated with that user and applies a *password function* $f: P \times \overline{P} \rightarrow \{0,1\}$ to both. If the complementary data is generated by applying the complementation function to the supplied password, the password function returns 1 to indicate the user has been authenticated; otherwise, it returns 0.

In what follows, when cryptographic-based schema are used, we will assume that the cryptosystem involved is neither invertible (in the sense that there are no trap doors) nor vulnerable to a chosen plaintext attack. We will further assume that the password complements are on-line and available to the authenticating user, system, or device, although they may not be directly accessible (for example, consider a dedicated satellite computer which is passed the supplied password and responds "authenticated" or "bogus")

Given our cryptanalytic assumptions, the most feasible attack on an authentication system is to guess a password, try it, and repeat until the correct password is found; such an attack is called a *dictionary attack*. The effectiveness of such an attack depends on the size of the set P as well as the distribution induced by S. If S selects passwords based on a uniform random distribution, and it takes time T to test a guess (by applying a member of F), then the expected time X for a dictionary attack is clearly $E(X) = \frac{1}{2}|P|T$. In fact, this is a simplification, for it assumes that passwords are selected using a uniform random distribution, which depends upon the members of S. This assumption is usually made for ease of analysis.

Some Authentication Systems

With all authentication systems, all members of F (the set of functions which takes as input elements of P and $\overline{P}$ and returns 1 or 0) are either external functions (such as the login program) or internal functions (such as the function in Figure 1 for UNIX systems). The former are executed when a user tries to access the system or an account; the latter may be executed at any time. Similarly, all members of S may be grouped into three categories, some of which may be empty for any given scheme: user selection, in which the user generates the password; system selection, in which the system selects the password; and combined selection, in which the user and system together select the password.

Traditional Password Systems

Among the most common authentication systems are the traditional password systems. These systems consider P to be a set of character strings, c to be a set of mathematical functions, $\overline{P}$ to be the values resulting from applying all members of C to all members of P, and S to be the set of functions implemented by the password assignment and/or changing tools. For example, the well-known UNIX password scheme [1] has:

P = { all character strings of length 8 or less and not including an ASCII NUL }

c = { 4096 different DES-based functions (each the DES with a different E table) }

$\overline{P}$ = { number of the used c prepended to 64 bit output of c and mapped to printable form }

The interesting questions, from a system standpoint, center on three issues. First, does the selection function ensure (a reasonable approximation to) uniformity of the distribution? Second, how are the password complements stored? Third, are members of C and internal members of F available to users?

Challenge-Response Protocols

Traditional password systems require that the password be validated by the user's terminal or transmitted to some other part of the system for validation. The former is generally not feasible (although they are available, few users use terminals with a powerful enough CPU to validate identity, and most system administrators would be leery of trusting terminals unless they were very well protected), and the latter exposes the system to compromise by a passive wiretapper. (Note it is not sufficient to encrypt the password before transmitting it.) A challenge-response system [3] seeks to overcome these problems. The user is sent a random string from the computer and must transform it according to an algorithm known to the user and the computer. Hence:

P = { possible algorithms }

$\overline{P}$ = P or { the inverses to elements of P }

C = { the identity mapping } or { functions to derive inverse transformations }

Note here that the members of F do not take the password and complement as input, but rather the output of the password and the complement when applied to a random string. Further, the members of S may select algorithms on a per-site basis [13] or a per-user basis [9], the latter being more common.

The Number and Distribution of Passwords

Given an authentication scheme, consider the set of possible passwords P. Clearly, if P is "too small" (the precise meaning of which depends upon the time needed to validate a password), the authentication scheme will be vulnerable to a dictionary attack. However, under other conditions, a successful dictionary attack can be launched regardless of how large P is. Each element in P has a certain probability of being chosen, and the function describing these probabilities is called

Figure 2a.

Figure 2b.

Figure 2. Here, the axes labelled "prob" reflect the probability of a password being chosen, and the axes labelled "index" reflect an ordering of the passwords in the set of possible passwords. The area under both bold lines is 1, but the distribution of 2a is skewed, whereas that of 2b is uniform.

the *probability distribution* of passwords in *P*. If this distribution is skewed, an attacker will have a high probability of guessing a correct password should he begin with the most probable passwords.

This metric suggests that the more constant the probability distribution, and the larger the size of *P*, the better the authentication scheme. Unfortunately, often a certain sophistication is needed to determine these two characteristics for a given authentication scheme. A wonderful story in [17] illustrates this point: at one installation, all passwords were 8 characters long, and characters were chosen randomly from the set of lower case letters and digits. The expected time to test all passwords on this system was 112 years. Unfortunately, the pseudorandom number generator that selected the characters had a period of 2^{15}, so only 32,768 different character sequences needed to be checked. The attacker got every password within 41 minutes. Here, the attacker took advantage of the skewed distribution (only 0.0000012% of all possible passwords could be selected) to cut his search time radically.

This situation is summarized in Figure 2a. This particular authentication schema tends to use passwords chosen from a rather small subset of possible passwords; hence, depending on the size of the set P, may be vulnerable to a dictionary attack. The authentication scheme in Figure 2b is much less vulnerable, as the degree of skewing is rather low.

The nature of the functions in *S* plays a key role in determining the distribution. If passwords are selected randomly, and the period of the generation function is the same as $|P|$, then the distribution induced by *S* will be the same as that of the generation function. However, if users are allowed to select the password, or select from a set that a generator provides, then the distribution becomes more problematic.

To make the above more concrete, consider the UNIX authentication system described earlier. In it, $|P| \approx 2^{56}$. If the computer chooses passwords for users using a uniform pseudorandom number generator with period 2^{56}, then all possible passwords are equally likely to be assigned. If the range of the generating function is restricted, for example to pronounceable passwords, then the generator must be carefully checked to ensure that the distribution is reasonably uniform and the size of the range of the generating function is sufficiently large. As an example of such an analysis, consider the password generator described in [10];[3] even though it restricts passwords to those which are pronounceable, a statistical analysis shows it chooses uniformly from among those pass-

3. This generator was written for MULTICS. However, the statistical analysis does not depend on this.

words which are possible, and provides a "large enough" set of possible passwords. An alternate approach is to have each phoneme represent *n* bits; for example, if there are 1024 possible syllables, each would represent 10 bits. This way, the use of a pronounceable password generator does not affect the set of passwords selected by members of *S*.

But if users are allowed to select their own password, a wealth of experience [15][17] indicates that many will select passwords from a small subset of *P*, thereby leading to a skewed distribution and a decrease in the time needed to mount a successful dictionary attack. Hence such a selection function effectively imposes a skewed distribution upon *P*.

Given that many sites prefer users selecting passwords, key crunching (taking something not in the set *P* and mapping it into the set of valid elements of *P*) [11] can smooth the distribution substantially. A strong cryptographic hash function, such as MAC [12], generates a much smaller set of bits from a larger set. Then the small set can be used as the appropriate member of *P*.[4]

The technique of preventing users from picking common passwords [2] is inferior to key crunching, because it seeks to smooth the distribution by eliminating peaks rather than making all elements of *P* equally likely. But if the set of common passwords is sufficiently small, it is effective.

A simpler version of this involves the use of pass-phrases, in which phrases are used to provide associations for passwords. If one's favorite poet is William Blake, choosing the first character of each word (except "the") from the lines "Tyger! Tyger! Burning bright/In the forests of the night" gives "TTBbifon", which would be considered to have a very low probability of being selected as a password (unless one's poetic tastes were known). In essence, the human is performing a form of "key crunching" on the longer pass-phrase to derive the appropriate password.

As an alternate example, consider the use of challenge-response algorithms. The number of potential algorithms is infinite, but some case must be taken to ensure selection is made from a fairly large set, if only because the algorithms must be programmed into the computer. For example, requiring the user to use a hand-held calculator with an encryption function to encrypt a string using the DES and a secret key stored on the calculator restricts the number of algorithms possible to 2^{56}, and also prevents the attacker from obtaining the user's algorithm by passive wiretapping and cryptanalysis (as obtaining the algorithm is the same as obtaining the user's key). In general, this scheme compares quite favorably to the more traditional password schema; the problems it suffers from do not relate to the probability distribution of algorithms in *P*, but rather to algorithm distribution and concealment, and sometimes to issues of physical security.

4. If the typed password is a member of *P*, this adds nothing to the probability distribution induced by *S* since the output of *S* will be the password to be typed. But if the typed password is not a member of *P*, then the attacker must try either to find it, or try elements of *P* which are not likely to have been selected if chosen by the user. Either way, the attacker will need to try more potential passwords. One can view this as increasing the size of *P* and altering the members of *C*, or "evening out" the distribution induced by *S*. This paper takes the latter view.

Ease of Authentication

The second issue, how easily users can provide their password, relates to human and technological capabilities. All authentication schemes are based on one (or more) of three characteristics: what users remember, what they possess, and/or what they are. In some sense, these form a hierarchy; possession is the simplest to subvert, since the device can be taken (or, in some cases, simply looked at); followed by memory, since memory depends upon a person's mental associations (which can often be guessed); and physical characteristics are the hardest, since one can hardly loan another a fingerprint. We consider only the first two, leaving the discussion for biometrics to another place.

Schemes based on memory usually require that passwords not be written down; therefore, results from the study of human memory in the field of psychology have provided a number of interesting and often misinterpreted models. Miller's summary of experiments [16] indicate that about 8 "chunks," or meaningful items (such as digits, letters, or words) can be repeated with perfect accuracy; this means that at most one random password of 8 "chunks" (or two of 4 "chunks," and so forth) can be remembered correctly. On the basis of Miller's summary, it has been suggested ([5], p. 342) that if the password were somehow made "meaningful," that is, if some association could be found to aid the user's memory, then the user could memorize considerably more, because the password would itself become a "chunk." One common technique is the use of a pronounceable password, which contains two or three chunks only; indeed, such passwords are recommended in many guidelines [19]. The problem with such schemes is that the phonemes – the smallest unit of pronunciation, and hence the "chunk" – are themselves essentially random, and therefore while this scheme is suited for one or two passwords, users will still write the passwords down when faced with remembering many different pronounceable passwords for many machines.

We should note another problem if each phoneme represents a set of n bits, as suggested in the previous section. In that case, the passwords would simply be using "chunks" instead of letters, and therefore suffer from the problems of non-pronounceable passwords.

But some memory-based schemes assume users will write down passwords, and have the users memorize an algorithm instead. At the Numerical Aerodynamic Simulation facility at NASA Ames Research Center, where many people must know the operator passwords to 40 (or so) computers, each user is given a list of passwords altered using an invertible password algorithm. For example, if the algorithm were "delete the fifth letter and append the number 32," and the operator's password for computer *athene* were listed as "gleork" then the real password would be "gleok32" [6]. This technique is akin to more general "pass-algorithm" techniques mentioned earlier, but does not involve challenge-response protocols. It is also an example of an authentication scheme depending on two characteristics: what the user knows (the algorithm) and what the user has (the list of transformed passwords).

For schemes based on possession, the physical characteristics of the devices in use and their susceptibility to theft control the efficacy of the authentication scheme. It may not be necessary to take the device; if someone writes down their password, simply looking at the paper will give an

attacker enough information to masquerade as the real user. However, in many cases it is necessary for an attacker to acquire the device, even for only a limited amount of time. The likelihood of this depends upon the determination of the attacker, the care of the user, and the value of the information on the system to the attacker.

We should note that many such devices depend upon some identification code imprinted or embedded upon the device [4]. If this code becomes known, an attacker can either make a device (eminently feasible for simple ones like smart cards) or simulate one (again, quite feasible for a calculator that is used in a challenge-response system); in either case, subversion of the system under attack will soon follow.

Perhaps for this reason, many manufacturers simply combine memory and possession by requiring the user to enter an identification code before authenticating the user, much as an automated teller machine requires the user to enter a Personal Identification Number after inserting a plastic card. This is an application of the principle of separation of privilege [20], which says that access to objects (such as a computer) should depend on more than one condition being satisfied. The technique the NAS facility uses to protect its operator passwords is another example of this.

Ease of Masquerade

The third issue, how difficult an attacker would find providing the password, asks how difficult protecting the plaintext passwords and their complements is. If an attacker cannot gain access to those complements, he must guess by undergoing the authentication procedure; by taking appropriate countermeasures such as exponential backoff or lockout, these attacks can be rendered too expensive and too easily detected. Similarly, allowing access to the complements but concealing the complementation function would also prevent dictionary attacks.

We should note that the latter (protecting the complementation function rather than the complements themselves) is the poorer of the two, because it violates the principle of open design [20]. This principle dictates that system design considerations be documented and the documentation available to the user community, and is sound for two reasons. First, often the information is available from other sources, and so hiding it simply gives the managers a false sense of security. Secondly, in many cases the information can be deduced by clever users, especially if they are sophisticated in techniques of cryptanalysis or system analysis.

The principle does not require the revealing of all information: it is perfectly reasonable to hold information which depends on an individual, such as a cryptographic key or a password, since the specific nature of that item is not central to the design of the system. Hence hiding the complements is acceptable, and in fact wise, for if the attacker can gain access to the complements, it may be possible to launch an undetectable dictionary attack.

A good example of this has been raised by discussions on how to improve UNIX password security. One school of thought states that protecting the complements (using shadow password files) is sufficient. A second argues that, rather than protect the complement, protect the algorithm by iterating the DES a variable number of times. This simply makes dictionary attacks more incon-

venient; if the number of iterations is system dependent, a clever attacker could simply disassemble the relevant files to determine the number, and if it is user-dependent, the number of iterations must be as available as the complement. In either case, the effect is the same as adding more bits to the password; it will affect the expected time for a dictionary attack to succeed, but will not force the use of the standard authentication procedures. Shadow password files will.

With challenge-response protocols, two issues arise: first, ensuring that the challenge is random (because if not, an attacker could simply record challenges and responses, then when attempting to masquerade as the victim, simply send the appropriate reply when challenged with a challenge to which he has the right response); second, ensure the algorithm (and user code, if needed) are adequately protected. Many vendors use a satellite computer placed between the point of connection and the computer itself; this machine is a dedicated authentication server, and can only be accessed either through the authentication procedure (where its role is simply to generate challenges and check responses) or through physical access (which enables the operators to change user keys if needed). Assuming adequate physical protection, this effectively eliminates any way to get the user code from the computer.

A second aspect of this metric is the degree of access to the passwords themselves. No matter how well the complements are protected, if the passwords are visible to an attacker, the attacker can spoof the authorized user. For example, many remote login programs transmit the password in the clear, making it available to any wiretapper. Standard cryptographic protocols are available to handle these situations [7][8][18][21]. If for some reason these protocols are considered infeasible in the site's working environment, a trusted path of some sort is necessary to prevent compromise.

Infeasibility may occur due to the need to interact with other sites not using the protocols. This emphasizes that the nature of the computing environment dictates what can be done. For example, many networks use the TELNET virtual terminal protocol to provide a remote virtual terminal service from which the user can then log in. As TELNET supports neither session keys nor any of the above-mentioned protocols, passwords are sent in the clear[5] and hence may be recorded by a wiretapper for later use.

Conclusion

The forth metric, the cost of the authentication mechanism, ties issues of risk analysis into the selection of an authentication scheme. It includes the financial cost of additional hardware, the cost of installing (and implementing, if appropriate) the scheme, and the daily cost of its use. The first two affect directly the financial impact and must be balanced against the possible losses suffered through not installing the scheme. The third is relevant because of the design principle of psychological acceptability [20]: mechanisms must not be so onerous that it is easier to bypass them

5. Note that encrypting passwords will only achieve the desired purpose (prevention of masquerade) only if each TELNET session uses a different session key, and that key is not available to the attacker. Otherwise, the attacker could simply record the initial exchange of datagrams, which would correspond to the login procedure, and spoof the user. From there, the desired parts of the relevant session could be replayed, or the TELNET server could be tricked into changing session keys (depending upon how the session key was implemented).

then work with them. Other ancillary costs (legal costs if the mechanism fails, etc.) also fall into this category, and are sufficiently complex that we shall not discuss them further here.

With respect to system issues, the three axes we have discussed at length provide a characterization of authentication systems sufficient to enable a comparison. The precise nature of comparison depends in large part upon the environment in which the authentication system is to be used, as was discussed earlier, and further work is needed to refine issues raised by such a statement. The issues raise questions of risk analysis, just as the fourth metric does, and should be considered along with that metric, as they relate to various forms of cost. In some sense, the fourth axis "warps around" the other three, to include elements from them, and so is not really an independent axis. This indicates that some more refinement of certain aspects of the model is necessary, and so falls into the category of "future work."

References

[1] M. Bishop, "An Application of a Fast Data Encryption Standard Implementation," *Computing Systems* **1**(3) pp. 221-254 (Summer 1988).

[2] M. Bishop, "A Proactive Password Checker," Technical Report PCS-TR90-152, Department of Mathematics and Computer Science, Dartmouth College, Hanover, NH 03755 (June 1990).

[3] G. Brassard, *Modern Cryptography: A Tutorial*, Springer-Verlag, New York City, NY (1988).

[4] S. Carlton, J. Taylor, and J. Wyszynski, "Alternate Authentication Mechanisms," *Proceedings of the Eleventh National Computer Security Conference*, pp. 333-338 (Oct. 1988).

[5] C. Coombs, R. Dawes, and A. Tversky, *Mathematical Psychology: An Elementary Introduction*, Mathesis Press, Ann Arbor, MI (1981).

[6] M. Crabb, "Password Security in a Large Distributed Environment," *Proceedings of the UNIX Security Workshop II* pp. 17-30 (Aug. 1990).

[7] D. Denning and G. Sacco, "Timestamps in Key Distribution Protocols," *CACM* **24**(8) pp. 533-536 (Aug. 1981).

[8] H. Feistel, W. Notz, and J. Smith, "Some Cryptographic Techniques for Machine to Machine Data Communications," *Proceedings of the IEEE* **63**(11) pp. 1545-1554 (Nov. 1975).

[9] "Final Evaluation Report of Sytek PFX A2000 and PFX A2100," CSC-EPL-86/006 (Nov. 1986), *cited in* [4].

[10] M. Gasser, "A Random Word Generator for Pronounceable Passwords," Technical Report ESD-TR-75-97, Electronic Systems Division, Hanscom Air Force Base, Bedford, MA (Nov. 1975)

[11] L. Grant, "DES Key Crunching for Safer Cipher Keys," *SIG Security Audit and Control Review* **5**(3), pp. 9-16 (Summer 1987).

[12] “Guidelines for Implementing and Using the NBS Data Encryption Standard,” FIPS PUB 74 (Apr. 1981).

[13] J. Haskett, “Pass-Algorithms: A User Validation Scheme Based on Knowledge of Secret Algorithms,” *CACM* **27**(8) pp. 777-784 (Aug. 1984).

[14] S. Kent and J. Linn, *Privacy Enhancement for Electronic Mail: Part II – Certificate-Based Key Management*, RFC-1114 (Aug. 1989).

[15] D. Klein, “Foiling the Cracker: A Survey of, and Improvements to, Password Security,” *Proceedings of the UNIX Security Workshop II*, pp. 5-14 (Aug. 1990).

[16] G. Miller, “The Magical Number Seven Plus or Minus Two: Some Limits on Our Capacity for Processing Information,” *Psychological Review* **63** (1956) pp. 81-97, *cited in* [5]

[17] R. Morris and K. Thompson, “Password Security: A Case History,” *CACM* **22**(11) pp. 594-597 (Nov. 1979).

[18] R. Needham and M. Schroeder, “Using Encryption for Authentication in Large Networks of Computers,” *CACM* **21**(12) pp. 993-999 (Dec. 1978).

[19] *Password Management Guideline*, Report CSC-STD-002-85, Department of Defense Computer Security Center, Fort George G. Meade, MD (Apr. 1985).

[20] J. Saltzer and M. Schroeder, “The Protection of Information in Computer Systems,” *Proceedings of the IEEE* **63**(9) pp. 1278-1308 (Sep. 1975).

[21] J. Steiner, C. Neuman, and J. Schiller, “Kerberos: An Authentication System for Open System Networks,” *USENIX Conference Proceedings*, pp. 191-202 (Winter 1988).