Problems with the Normal Use of Cryptography for Providing Security on Unclassified Networks

Russell L. Brand
Lawrence Livermore National Labs
Livermore, California

Abstract

The normal use of cryptography in unclassified computing systems often fails to provide the level of protection that the system designers and users would expect. This is partially caused by confusion of cryptographic keys and user passwords, and by underestimations of the power of known plaintext attacks. The situation is worsenned by performance constraints and occasionally by the system builder's gross misunderstandings of the cryptographic algorithm and protocol.

1 Introduction

For the past five years, my colleagues and I have been studying unauthorized intrusions (attacks) on unclassified computers owned by government agencies and universities. While some of these attacks were made possible by problem in the computer operating system or the software utilities present on the machine, most of the attack we aided by poor password practices. Some of these practices are caused by users and system designers thinking of a user password as if it were a cryptographic key at some points in their analysis and not treating it as cryptographic key at other points.

Often the use of encryption is incomplete. This leads to "partially signed" message, easily forgable signatures, and simplified ease dropping. In most cases, the design errors are at the level of cryptographic protocol rather encipherment algorithm.

After discussing the typical problems that we have found in practice, we will describe what seems to be needed by the unclassified community by cryptography.

2 A Password is Not a Key

While passwords bear some superficial similarities to cryptographic keys, they are not really keys. They are not quite treated as keys either.

2.1 Passwords are often shorter than the look

Where cryptographic keys tend to be "long" and "random", passwords tend to short and not random. A cryptographic key that is N bits in length is expected to have N bits of randomness (or under certain circumstances where the key is an N bit prime number, there may only be $\log N$ bits of randomness) and N is measured in the hundreds or thousands or bits. Passwords, in contrast tend to be no larger than 64 bits. Of these about 36 of the bits are significant (the others being just padding of one form or another) and these 36 bits

G. Brassard (Ed.): Advances in Cryptology - CRYPTO '89, LNCS 435, pp. 30-34, 1990.

tend to contain at most 12 bits of randomness since the password is often a collection of names or proper words. [2]

2.2 Re-used Passwords Lead to Difficulties

While an individual would have a different cryptographic key for each distinct (mutually suspsicious) entity he would interact with, it is common practice for a user to have a single password that he uses on computers run by different computer centers in different (potentially competing) firms. While a cryptanalyst would demand proof that none of these agencies can exploit the use of a "key in common", common practice with passwords is to do this with no precautions taken and obvious means of exploitation possible.

2.3 Broadcast of Clear Text "Keys" is Poor Practice

Another difference between a cryptographic key and a password is in their respective handling. It is common practice to broadcast passwords over known insecure channels. This would of course never be done with a cryptographic key. Even after many documented cases of "wire tapping" (which is both effortless and undetectable in a traditionally configured network) the transmission of passwords in clear text remains standard operating procedure. Only recently have serious alternatives begun to be considered [4,6]

There in fact ways to make this even worse. In one system passwords were "usually" sent encrypted. Each password was always encrypted in the same way and hence one could gain full access by just having seen the encrypted password which was both broadcast over insecure channels and stored in public areas. Further, the method for updating a password entailed sending the old password enciphered and the new password in clear text. In this manner the user could pay the extra computational cost of encryption on his slow local machine on each access where a naive attacker would not have to pay any encryption or decryption costs.

3 Known Plaintext Attacks are not Foiled by Salt

The most common use of encryption in unclassified computing is for authentication. In particular many systems keep and encrypted, hashed or "trapped doored" version of the users password in a table. For each access, the user provides his password to the machine which encrypts, hashes, or applies a trap door function as appropriate and then compares the result with the value stored in the table.

There are two good features to this method. First if the encipherment of of the password in the table is effectively uninvertable then compromise of this table will not allow an attacker to impersonate the real user. Second a variety of standard mistakes that can be made in comparing clear text passwords are not possible here.

In common versions of "unix" operating system, the encrypted password table is made publically readable. In many cases this table is stored in such a fashion as to allow copies of it to be read even by people who have no other access to the system. It has become increasingly popular for copies of these tables to be "stolen" by an attacker who then breaks the passwords and later impersonates legitimate users.

While inverting the encipherment is very hard, it is very easy to make a guess at password and then check to see if it is correct. On a small personal computer hundreds of guesses can be made per second and with only 12 bits of randomness, it doesn't take long to

guess successfully. Since this testing is done on a separate computer than the one that the password tables were stolen from, it is undetectable.

In the unix system a technique called "salting" is used to make it more difficult to precompute guesses. It does not prevent this "guess and test strategy" and is more fulled discussed in [3]

4 Unauthenticated Authentication Servers lead to Problems

Even when the details of what should be sent encrypted, what should be sent in place plaintext and who should be trusted to make the comparisons are correctly handled, errors in the handling of encrypted passwords can still be made. In some cases there is no authentication of the password servers themselves; any machine can announce that it is itself the password server that all other machines should listen to. Similarly the authentications servers will often provide the crypted passwords to any machine that ask make requests. The authentications server trusts that these machine will prevent unauthorized users from making such requests.

5 Tampering of Signed Packets is often Possible

To prevent the problems of machines claiming to be authentication servers for each other or successfully impersonating each other in other transactions, a variety of crypto-graphically based "secure" transaction mechanism exist. In some cases each request (or packet) contains a the encrypted version of a sequence number. The system designers had confidence in this since an attempt to repeat use of a sequence number could be detected.

Unfortunately it is easy to make sure that a packet never arrives. One can "collide" with the request packet to assure that it is not delivered, steal the encrypted sequence number and put that "signature" into a different packet hence impersonating the legitimate machine. The computational costs of using cryptographic checksums or other more complete digital signature methods were deemed unacceptably high and hence this vulnerability was never fixed.

6 Difficult Factoring Effect the Security of Discrete Logs

One of the secure transaction protocols was based on a discrete log key system. For performance reasons the composite base number that was chosen was much too small. This is not surprising or unusual. Nor is it surprising that the number was factored and the hence the system broken [5]. What is counter-intuitive was the part of team charged maintaining the system seemed to not understand the the underlying cryptosystem and hence improperly choose a non-generator number and did not know that advances in factoring techniques had any impact on a discrete log based system. Further the system designers had posted a "challenge" number and felt secure since noone had solved their challenge. The details of announcing the challenge were such that the cryptographers working on related problems were unaware that a challenge had been posted.

7 Bad Information Leads to Bad Decisions

Designing the security aspects of a computer system is very difficult. It is made harder by a both lack of good information and the prevalence of disinformation. Computations requirements and performance estimates made for the cryptographic sections of system I have been involved with been off by more than 4 orders of magnitude in each direction. There is a general confusion (at the very least) about the legalities, strengths, and speeds of DES and of RSA. Beyond this, there seems to be little consensus on the feasibility of any of the non-DES non factoring based cryptographic methods.

8 User Errors are Compromise otherwise Good Systems

In practice, we often find cryptographic keys and passwords left in publically readable cleartext on otherwise safe machines. Often programs have options for reading passwords and keying information from file encouraging these errors.

Clear text version of encrypted files are often left in system buffers, temporary files or ordinary files. In several cases the unencrypted password has is written to publically readable areas of memory or disk before the encryption itself is done or program break leaving the passwords, keying information and/or clear text data available with other debugging information.

These are traditionally not the problems of the cryptographer but rather the problem of the system designer. Very often these issues are not successfully solved. As a result the attackers we have dealt with in the unsensitive unclassified world haven't had to learn to exploit any weaknesses in the cryptographic system or protocol.

In a similar vein, the unix crypt facility, a modified (one rotor) version of Enigma is still widely used despite the publication of an automated facility for breaking this cypher [1]. I do not claim to understand all of the social factors that encourage this type of inertia. Perhaps misinformation about the security and speed of DES play a major roll in it.

9 Authentication for the Academic World

To make it easier for computer systems designers to create systems with some degree of security, perhaps rather than trying to piece together system from what we find in the literature, we should be making an explicit request. In the hopes that such a request will encourage some of you in the Cryptographic Community to invent, adapt or simply more visibly announce tools that we can use, these requests follow.

1. A method of authentication whereby a user types a short password (about 12 bits) onto a computer and in doing so proves to another using only an insecure broadcast between the two computers that he is indeed that person. It should be the case the watching a large number of these broadcast transaction an attacker can't appreciably improve his chances of guessing the password or otherwise impersonating the user even with $1,000,000$ the computing power of the normal authentication.

2. A method where one computer can prove to another computer on a broadcast network of $5,000,000$ machines that it a given computer such that

   - Adding a new node is easy;

- Disabling the password of a compromised node is easy;
- Watching milions of transactions doesn't allow inpersonation;
- Keys do need to be changed more often then once per billion authentications per node;
- Private channels are very expensive and require human intervention;
- Each authentication can be done in a few million 32 bit instructions.

3. A fast method to tell that small file (less than 1000 bytes) has been signed by a given entity. Preferably using only a few hundred thousand 32 bit operations.

4. A true zero knowledge authentication analogous to the graph homomorphism method than can give a confidence level of one in 2^{20} with a few hundred thousand bytes of traffic and a few million 32 bit operations.

Anyone with practical answers to these requests, is encouraged to contact the author.

10 Conclusion

While social attacks of stolen keys and bribery may continue to offer the greatest threat to the high security environments, academic and other unclassified computing could be greatly simplified by better implementations of security systems. Better use and understanding of cryptosystems can form an important part of these computer systems.

It is hoped that by reviewing the current problems with the cryptographic parts of security systems that future designers can avoid these mistakes and that cryptosystem designer can better warn potential users of their system about the potential hazards and misuses.

References

[1] Robert Baldwin *Rule Based Analysis of Computer Security.* LCS Technical Report 401, Massachusetts Institute of Technology, 1987.

[2] Russell L. Brand *A Computer Security Tutorial.* 1987.

[3] David C. Feldmeier and Philip R. Karn *UNIX Password Security - Ten Years Later* (In this Volume) 1989.

[4] S. P. Miller, B. C. Neuman, J. I. Schiller, and J. H. Slatzer, *Kerberos Authentication and Authorization System.* Project Athena Technical Plan. Project Athena, Massachusetts Institute of Technology. 1987.

[5] Andrew Odlyzko (In preperation).

[6] J. G. Steiner, B. C. Neuman, and J. I. Schiller, *Kerberos: An Authentication Service for Open Network Systems.* IUSENIX Conference Proceedings, Winter 1988, pp. 191-202, USENIX Association, February 1988.