The following paper was originally published in the
Proceedings of the Sixth USENIX UNIX Security Symposium
San Jose, California, July 1996.

Dual-workfactor Encrypted Key Exchange: Efficiently Preventing Password Chaining and Dictionary Attacks

Barry Jaspan

Dual-workfactor Encrypted Key Exchange: Efficiently Preventing Password Chaining and Dictionary Attacks

Barry Jaspan*

Abstract

Password-based key-server protocols are susceptible to *password chaining* attacks, in which an enemy uses knowledge of a user's current password to learn all future passwords. As a result, the exposure of a single password effectively compromises all future communications by that user. The same protocols also tend to be vulnerable to dictionary attacks against user passwords.

Bellovin and Merrit[1] presented a hybrid of symmetric- and public-key cryptography called Encrypted Key Exchange (EKE) that cleanly solves the dictionary attack problem. This paper presents an extension of their ideas called *dual-workfactor encrypted key exchange* that preserves EKE's strength against dictionary attacks but also efficiently prevents passive password-chaining attacks.

1 Introduction

Virtually all password-based key server protocols are vulnerable to *password chaining* attacks. In these systems, a user shares only a single secret, its password, with the trusted key server. In addition to protecting the normal authentication and key-distribution messages, the current password is also used to protect the protocol exchanges used to select a new password. An enemy that already knows the user's current password can "daisy-chain" the passwords together by decrypting the password-changing session and reading out the new password.

Many password-based protocols are also vulnerable to a dictionary attack against user passwords. In the Kerberos authentication system[20], for example, the initial protocol exchange is encrypted in a key derived from the user's password. A malicious user can obtain these encrypted messages and use them to attempt to guess the user's password. Since the format of the messages is well known, a correct guess is easily verifiable. Once the guessing attack succeeds, the attacker can use the password to impersonate the user until the password is changed. In practice, password guessing attacks have a high degree of success[12, 24].

The combination of these two weaknesses is particularly worrisome. The password guessing vulnerability makes it likely that at least one of a user's passwords will eventually be disclosed; password chaining means that once one password is disclosed, all future passwords are compromised. Since it is easy for an attacker to record all of a user's traffic while simultaneously attacking old messages, both past and future communications are at risk.

Bellovin and Merrit[1] presented a fundamentally novel hybrid of symmetric- and public-key cryptography called Encrypted Key Exchange (EKE) that cleanly solves the dictionary attack problem. This paper presents an extension of their ideas called *dual-workfactor encrypted key exchange* that preserves EKE's strength against dictionary attacks but also prevents passive password-chaining attacks, without incurring unnecessarily high overhead. The basic idea is to use an extremely strong EKE exchange for password-changing messages, thus preventing chaining, while using a much weaker EKE exchange with normal authentication for efficiency. Dual-workfactor encrypted key exchange is motivated in the context of a new EKE-based pre-authentication type for Kerberos 5 called PA-ENC-DH that can be implemented without changing the current Kerberos 5 protocol or requiring additional network messages.

Section 2 discusses previous work on password chaining and dictionary attacks. Section 3 provides background on the Kerberos and Diffie-Hellman exponential key exchange protocols, and motivates dual-workfactor encrypted key exchange. Section 4 discusses implementation issues for this protocol specific to Kerberos 5, including public-key parameter selection. Table 1 summarizes the notation used throughout this paper.

*Affiliation: Independent consultant, 4 Merrill Ave, Belmont, MA 02178, *bjaspan@mit.edu*. This work was also funded in part by BBN Planet Corporation and the Internet Software Consortium.

Notation	Meaning
C	client
S	server
KDC	Key Distribution Center, a.k.a. the Kerberos server
K_x	symmetric secret key for user x
K_{xy}	symmetric session key for users x and y
$\{X\}^{K_x}$	message X encrypted in symmetric key K_x
p_x	private Diffie-Hellman key for user x
P_x	public Diffie-Hellman key for user x, $P_x = q^{p_x}$
$\{X\}^R$	message X encrypted in symmetric key R derived from Diffie-Hellman key $q^{p_c p_s}$
T_{cs}	Kerberos ticket for client c to use service s
$\mathrm{PA} = D$	Kerberos pre-authentication with data D

Table 1: Notation used throughout this paper.

2 Previous work

There does not appear to be any previous work on strengthening password-based protocols against an attacker that already knows the password.

The issue of dictionary attacks has been widely discussed. Kerberos version 5[16] partially addresses the problem with *pre-authentication*, requiring the initial request to the KDC to prove the user's identity so an attacker cannot fraudulently request tickets for another user; see section 3.1. A protocol can be designed in which off-line guessing attacks are impractical and so every password guess must involve a central authority which can then detect the attack[18]. A password changing mechanism can be designed to enforce password quality, insisting on passwords with a minimum length, character mix, or other properties. It is also possible to design a password selection process that simplifies the selection of quality passwords that are both easy to remember and hard to guess[4].

3 Protocol description

This section provides background on the Kerberos and Diffie-Hellman protocols, explains how Encrypted Key Exchange is applied to Kerberos, and motivates dual-workfactor encrypted key exchange. The new PA-ENC-DH Kerberos pre-authentication type is presented.

3.1 Kerberos AS_REQ protocol

The most basic exchange of the Kerberos protocol is an Application Service Request, or AS_REQ. Client software uses an AS_REQ to request a Kerberos ticket for a user. The response is sent encrypted in a key derived from the user's password. Typically, an AS_REQ is used exactly once per login session to obtain a ticket-granting ticket. A simplified version is as follows:

$$\begin{aligned} C \rightarrow KDC &: C, S \\ KDC \rightarrow C &: C, T_{cs}, \{S, K_{cs}\}^{K_c} \end{aligned}$$

In the AS_REQ, the client C sends its own name and the name of a service S to the KDC. The KDC responds with an AS_REP (Application Service Reply) containing the client's name, a ticket T_{cs} for the client to use with the service, and a block of data *encrypted in the user's password-derived key* K_c containing the service name and a copy of the session key K_{cs} for the client and service to use. A dictionary attack is possible against the block encrypted in K_c in the AS_REP because the attacker can validate password guesses—if an attacker guesses K_c', performs the decryption, and a meaningful server name S appears in the result, the guess is correct.

Kerberos 5 introduced pre-authentication into the AS_REQ exchange as a partial defense against dictionary attacks. With pre-authentication, the user's initial request is required to contain a block of data that proves the identity of the requesting user. The Kerberos server refuses to respond to the request unless the pre-authentication data is correct, thus preventing an attacker from making a fraudulent request at any convenient time. The most common pre-authentication method uses an encrypted timestamp and is called PA-ENC-TIMESTAMP:

$$\begin{aligned} C \rightarrow KDC &: C, S, \mathrm{PA} = \{\mathrm{time}\}^{K_c} \\ KDC \rightarrow C &: C, T_{cs}, \{S, K_{cs}\}^{K_c} \end{aligned}$$

When the KDC receives the AS_REQ, it decrypts the timestamp using the client's secret key, and only honors the request if the timestamp is within a predefined window of the KDC's current time. An attacker cannot forge the pre-authentication data without knowing K_c. However, the attacker can still read either the request or the reply off the network and perform a dictionary attack against it. Other forms of pre-authentication exist[21] in which the request does not allow for dictionary attacks, but so far all of them leave the reply vulnerable. Thus, while existing pre-authentication schemes make a dictionary attack somewhat harder to perform, they are not a fully satisfactory solution.

3.2 Kerberos password-changing protocol

Kerberos users change their passwords by interacting with the Password Changing Service, typically implemented as part of the Kerberos administration system. The password-changing program performs an AS_REQ to obtain a ticket $T_{c,pw}$ for the administration principal S_{pw} which it then uses to perform encrypted communications with the Password Changing Server:

$$\begin{aligned} C \rightarrow KDC &: C, S_{pw} \\ KDC \rightarrow C &: C, T_{c,pw}, \{S_{pw}, K_{c,pw}\}^{K_c} \\ C \rightarrow S_{pw} &: T_{c,pw}, \{\text{new pw}\}^{K_{c,pw}} \end{aligned}$$

An attacker that already knows the user's password can decrypt $\{S_{pw}, K_{c,pw}\}^{K_c}$ to obtain $K_{c,pw}$ and then decrypt $\{\text{new pw}\}^{K_{c,pw}}$ to obtain the new password. This is called *password chaining* and it can be performed indefinitely to learn all of a user's future passwords.

3.3 Exponential key exchange

Diffie-Hellman exponential key exchange[8] is a well-known method for two parties to exchange a secret key across an open network. A good explanation appears in [23]. Briefly, the Diffie-Hellman protocol requires a prime modulus m and a primitive root of that modulus q; both values are public knowledge. Suppose that two parties C and S want to exchange a key. C picks a random number p_c, computes from it $P_c \equiv q^{p_c} \bmod m$, and sends that value to S:

$$C \rightarrow S : P_c$$

Similarly, S picks a random number p_s, computes $P_s \equiv q^{p_s} \bmod m$, and sends the value to C:

$$S \rightarrow C : P_s$$

At this point, both C and S can compute $R \equiv {P_c}^{p_s} \equiv {P_s}^{p_c} \equiv q^{p_c p_s}$. An attacker knows both P_c and P_s but cannot compute R without knowing one of p_c or p_s, neither of which is available. C and S can use R to encrypt further communications.

This protocol provides key exchange, but it does not provide authentication[9] because it is subject to a man-in-the-middle attack. An attacker can intercept all communications between C and S and substitute its own values for P_c and P_s, and neither C nor S will notice. Furthermore, the choice of m is critically important from a practical point of view[17, 22] if the protocol is to be resistant to cryptanalytic attacks.

3.4 Merging Diffie-Hellman and Kerberos

Encrypted Key Exchange (EKE) can be used to merge Diffie-Hellman exponential key exchange and Kerberos. The protocol presented here is a proposed pre-authentication type for Kerberos called PA-ENC-DH:

$$\begin{aligned} C \rightarrow KDC &: C, S, \text{PA} = (\{P_c\}^{K_c}, m) \\ KDC \rightarrow C &: C, T_{cs}, \{S, K_{cs}\}^{R}, \text{PA} = \{P_s\}^{K_c} \end{aligned}$$

The client chooses a random p_c, computes P_c, and sends P_c *encrypted with its secret key* K_c along with the modulus m as pre-authentication data. The KDC receives the request, decrypts P_c with its local copy of K_c, generates a fresh p_s and P_s of its own, and computes $R \equiv {P_c}^{p_s}$. Finally, and this is the critical part, the KDC encrypts the block containing the server name and session key in R instead of the user's key K_c. The KDC then sends the AS_REP to the client along with its own public exponential P_s encrypted in K_c. The client can decrypt $\{P_s\}^{K_c}$ to obtain P_s, use P_s and its own p_c to compute R, and use R to decrypt the KDC response and extract the session key K_{cs}.

3.5 Analysis of PA-ENC-DH

Using R to protect the AS_REP has two significant consequences. The new AS_REP is not vulnerable to a dictionary attack because R is derived entirely from random values unconnected to the user's key K_c. The new AS_REP is also not vulnerable to a passive password chaining attack, because knowledge of K_c only yields P_c and P_s which are not sufficient to compute R.

A dictionary attack against $\{P_c\}^{K_c}$ is not possible because there is no way for an attacker to verify a correct guess. P_c has no number-theoretical properties other than being less than m (this affects the choice of m; see section 4.1) and is generated from a random number p_c; thus, when an attacker guesses K'_c and gets P'_c, there is no way to tell if the guess is right.[1] The same argument applies to P_s. Even an attacker that computes the potential P'_c for all K_c in the dictionary has still accomplished nothing because P_c alone will not decrypt the AS_REP. R cannot be computed without p_c or p_s, neither of which is available without computing the discrete log. Since both of the random exponentials are encrypted for transmission, it is extraordinarily difficult to compute the discrete log even for small values of the modulus m. If P_s were not encrypted, for example, an attacker could calculate its discrete log p_s and then validate guesses at K_c by comparing each potentially correct $P_c'^{p_s}$ with R. P_s is not available, so the enemy has to perform a dictionary attack against $\{P_s\}^{K_c}$ and then compute the discrete log of *every* result.

A man-in-the-middle attack is thwarted, as well. An attacker that intercepts $\{P_c\}^{K_c}$ cannot predict how a replacement will be decrypted by the KDC. If $\{P'_c\}^{K_{fake}}$ is inserted, the KDC will decrypt to get $P_{unknown}$ and compute $R' \equiv P_{unknown}^{p_s}$; the response will then be encrypted in R' but will include the legitimate P_s. Since the attacker cannot replace P_s without knowing K_c, the client will compute $R \equiv {P_s}^{p_c} \not\equiv R'$, and will notice the attack when it is unable to decrypt the response.

An attacker cannot successfully forge a response from the KDC because he cannot generate a correct pair $(\{P_s\}^{K_c}, R)$ matching the client's p_c. Furthermore, since the exchanged key R in the reply is derived from the value P_c just provided to the KDC, the client is assured that the KDC's response is fresh and not replayed. This guaranteed freshness has the additional benefit that the client can set its clock by the ticket's timestamp, using the KDC itself as a secure time service and eliminating Kerberos' dependence on time synchronization[6]. This is a simple variation on the basic idea presented in [7].

Finally, even if the attacker already knows K_c, he still cannot determine R without performing a discrete logarithm computation. Without R, the attacker cannot obtain the session key K_{cs} so as to eavesdrop on the conversation. It is this property that prevents passive password chaining attacks—if the modulus m used to compute R is long enough, knowledge of the current password cannot be used to discover future passwords.

3.6 Dual-workfactor encrypted key exchange

The previous analysis of PA-ENC-DH presents a conflict in choosing the modulus length. For efficiency, the modulus should be as short as possible because the KDC's throughput will generally be limited by the time required to perform exponentiations modulo m. The nature of PA-ENC-DH allows the modulus to be considerably shorter than that which would normally be required for long-term Diffie-Hellman security. However, a short modulus limits the effectiveness of one of PA-ENC-DH's primary benefits, that knowledge of the password alone cannot be used to learn the session key K_{cs} without also computing a discrete logarithm modulo m. If a short modulus is used and the user's password is ever compromised, any individual recorded session protected by that password can be disclosed merely by computing a single discrete logarithm under the small m. This is a substantial improvement over the current Kerberos protocol, but does not fully take advantage of PA-ENC-DH's potential.

The solution is to enhance EKE by using *dual-workfactor* encrypted key exchange, employing a harder public-key problem for the more critical password-changing messages than for the day-to-day authentication messages. In particular, the password-changing Kerberos exchange should be protected with a modulus that provides sufficiently long-term protection for the password-changing message all by itself, perhaps a value 1,024 or 2,048 bits in length. An attacker with the user's password will be able to read previously recorded ordinary sessions protected with the shorter modulus of perhaps a few hundred bits with only moderate difficulty, but when the attacker comes to the password-changing session the discrete logarithm will not be computationally feasible. The exposure of a disclosed password will thus be limited to the sessions actually protected by that password. Meanwhile, the majority of AS_REQs used to acquire normal ticket-granting tickets can still use a smaller modulus for efficiency.

4 Implementation issues

This section discusses some implementation requirements for the PA-ENC-DH extension to the Kerberos 5 protocol, including a discussion of Diffie-Hellman parameter selection. Familiarity with the discrete logarithm problem and Kerberos is assumed.

4.1 Properties of m and q

Any modulus selected must conform to all the normal requirements for a good prime for the discrete logarithm problem. The current literature[15, 26] recommends using *safe primes* for which both m and $(m-1)/2$ are prime. These primes are time-

[1] Note that this is only true if P_c is not encoded with redundant information (e.g. with ASN.1[5]) before encryption.

consuming to find but, as described below, can be computed off-line.

PA-ENC-DH imposes the additional requirement on m that it not provide an attacker any information that can be used to validate a guess at a user's password; this is not normally a consideration with Diffie-Hellman. Such information can be leaked because, with an N-bit m, the random exponentials P_c and P_s by definition satisfy

$$P_x < m < 2^N$$

but the decryption of $\{P_c\}^{K_c}$ with an incorrect guess of K_c can have any value less than 2^N with equal probability. A guess K_c' that yields a value

$$m \leq P_c' < 2^N$$

can be immediately discarded as incorrect, reducing the number of entries in the dictionary for which an attacker must compute a separate discrete logarithm. Viewed another way, a guess of K_c' that yields a valid P_c' for a large number of messages has an improved probability of being correct; in effect, this is a simple dictionary attack applied over multiple messages instead of a single message.

This probabilistic attack can be prevented by choosing m to be close enough to its maximum value 2^N. [2] How close is enough? The probability that a guess K_c' will yield a P_c' smaller than m, and thus be possibly correct, is $\frac{m}{2^N}$. Thus each message will allow an attacker to reduce the size of the working dictionary for that user by a factor of $\frac{m}{2^N}$; the fraction $1 - \frac{m}{2^N}$ of passwords in the dictionary will be disqualified. The effect is cumulative, so if an attacker steals M messages, the working dictionary will be reduced by a factor of $(\frac{m}{2^N})^M$[6].

If the attacker's dictionary has D entries and it takes T time to compute a single discrete logarithm in the field, m must be chosen so that $D * T * \frac{m}{2^N}^M$ is still large enough to make the attack impractical. It is reasonable to assume that an attacker could collect on the order of 1,000 messages encrypted in K_c—two per day during login, assuming the password is changed only annually. If m differs from 2^N by no more than one part in 10,000, the attacker's work load will be reduced by a factor of $(0.9999)^{1000} \approx 0.90$, or by about 10%, which seems reasonable. Consequently, m should be chosen so that its first 14 bits are all 1.[3]

If the modulus m is a safe prime, the generator q can be 2; the provides an additional speed advantage[26].

4.2 Generation of exponents

The most recent work on how long Diffie-Hellman exponents need to be in order to provide sufficient security[26] confirms the widely held opinion that the exponents should be twice as long as the symmetric key that will be derived from the exponentially exchanged key. For Kerberos, the longest keys currently in use are 168 bits, for Triple-DES. Thus, the exponents need be at least 336 bits long. Note that if the exponentials are not a multiple of the symmetric cryptosystem's block size in length, the extra high-order bits of the last encrypted block of $\{P_x\}^{K_c}$ must be filled with random data[1]; alternatively, the exponentials' length can simply be rounded up.

The exponents p_c and p_s selected by the client and KDC must be "random" in order for PA-ENC-DH to be secure. The question of how much randomness is needed and how it can be generated on a computer without a hardware random number source is still unresolved. [11] and [14] address this issue and provide various implementation suggestions. The implementation must be careful not to use the client's password K_c to generate random exponents in a way that will expose it to an indirect dictionary attack. For example, a guessable seed for a PRNG whose output is encrypted with K_c will provide at least some verifiable bits of plaintext to an attacker.

4.3 Modulus length recommendations

To perform a dictionary attack against PA-ENC-DH, an enemy must compute a discrete logarithm for every password guess. The cost of a dictionary attack is therefore the product of the time to compute the logarithm and the size of the attacker's dictionary. The best algorithms for computing discrete logarithms are executed in two phases, a pre-computation phase and a per-exponent phase. The pre-computation phase involves building a large, modulus-specific database and is the time consuming portion of the attack. The per-exponent phase is relatively fast. In fact, [17] estimates that discrete logarithms for a 192 bit modulus can be computed in only several minutes after pre-computation is complete. If "several minutes" for a

[2] Bellovin and Merritt suggest an alternative, and perhaps simpler, solution. Instead of transmitting $\{P_c\}^{K_c}$, transmit $\{P_c + rm\}^{K_c}$ where rm is added to P_c using non-modular addition such that the value encrypted can have any value $0 < P_c + rm < 2^N$ with equal probability; the legal values of r are discussed in [1]. The legitimate user can be sure the decrypted value is right and thus can factor out rm after decryption; the attacker doesn't know whether the decrypted value is greater than m because of the addition or because the guessed password is wrong and so gains no information.

[3] The number of most-significant bits of $2^N - 1$ and $p(2^N - 1)$ that are equal is $\lceil N - 1 - \log_2(2^N - 1 - p(2^N - 1)) \rceil \approx \lceil N - 1 - \log_2((2^N - 1)(1 - p)) \rceil \approx -\lceil \log_2(1 - p) \rceil$. For $p = 0.9999$, $-\lceil \log_2 0.0001 \rceil = 14$ bits.

192 bit modulus actually means five minutes, it would take just less than a year to try 100,000 passwords, after pre-computation. Depending on site password policies and the importance of past sessions remaining secret, that may not be long enough. As described in section 4.2, the modulus must be at least 336 bits anyway. This might be sufficient, but it seems prudent to use a 512 bit modulus for now. [15] states that pre-computation on a 512 bit modulus currently takes about a year. Assuming per-exponent computation time grows at least linearly with length, an attacker would then have to spend at least several more years guessing passwords. This is probably strong enough protection for a user that might choose a guessable password.

The situation is more clear for the password-changing modulus. The modulus must be chosen with the assumption that an attacker already knows the user's password and, therefore, the public exponentials. Only one discrete logarithm computation will be required. Therefore, the modulus must be long enough to prevent the pre-computation phase entirely. [15] suggests that 1,024 bits is enough to protect a single modulus for a decade, and a 2,048 bit modulus is currently believed to prevent discrete logarithm computation forever. Since passwords are typically changed at most monthly and often much less frequently than that, a reasonably high delay can be tolerated, so there is no good reason not to use 2,048 values.

4.4 Configuration of moduli

The specific modulus to use for each transaction is not a fixed constant for the protocol but rather is specified by the client as part of the protocol message itself. This is necessary to allow dual-workfactor encrypted key exchange. It also frees the protocol from being tied to a specific modulus which may be found to be too short or too weak for other reasons.

Moduli can be site-specific and changed as often as necessary to keep up with computational speed increases and theoretic developments. The various moduli a client is allowed to use should be specified in a configuration file (e.g. `/etc/krb5.conf`) that is available on all client hosts.[4] Two classes of moduli must be specified, one for normal authentication and one for use with the password-changing protocol. The client can simply choose one of the listed values, and then include it in the PA-ENC-DH message; alternatively, the protocol could be designed to specify an index into a list of moduli for transmission efficiency. The KDC should have a copy of the same list of moduli and accept a message based on any of them, but refuse a message based on any other value on the grounds that it may not be secure.

[4]Clients must verify that the moduli meet the requirements of section 4.1 to prevent trojan horse values.

5 Consequences and limitations of PA-ENC-DH

PA-ENC-DH only prevents *passive* password chaining attacks against recorded sessions. An attacker that knows the user's password and fully controls the network (rather than merely being able to listen on the network) can perform an *active* man-in-the-middle attack by replacing the encrypted exponentials in the client's and server's message with its own value. The attacker can then allow the client and server to complete the password change without interruption while at the same time learning the user's new password. Interestingly, it seems *almost* possible for a client to detect a man-in-the-middle attack by timing the password-changing protocol, since an attacker is forced to compute several exponentiations himself; however, a clever attacker can avoid such detection[6].

PA-ENC-DH does not actually provide pre-authentication of the client's request to the Kerberos server. Furthermore, it makes using other forms of pre-authentication questionable since some of them (e.g. PA-ENC-TIMESTAMP) will re-expose the client's password to a dictionary attack. This implies that it is once again not possible to have any control whatsoever on who requests tickets for a given user. As PA-ENC-DH solves the problem that pre-authentication was designed to solve better than pre-authentication does, this is not really a drawback.

The lack of pre-authentication does expose PA-ENC-DH to an undetectable on-line password guessing attack[10]. Since the KDC cannot authenticate the entity performing an AS_REQ or detect the freshness of a request, an attacker is free to make password guesses by performing the AS_REQ protocol with PA-ENC-DH pre-authentication at any time. This is not a serious threat, however. The KDC can be modified to allow only a certain number of requests per day and refuse (or simply log to a high-priority channel) any requests over the limit. An upper limit of perhaps 20 requests per day will be more than enough for users but still low enough to prevent such on-line guessing attacks.

The KDC is sometimes used to maintain state about user logins (e.g. invalid login attempts, last login time) that provide useful administrative information. Pre-authentication allowed this information to be considered "trustworthy" because a request with bad pre-authentication data would be rejected without updating user state. Without pre-authentication, the KDC cannot tell a valid request from an invalid one. However, most of this information can be tracked just as usefully without pre-authentication—there is no loss in security, for example, of considering the last login time of a user to be the time of the last AS_REQ, authenticated or not.

6 Conclusion

With only a single modulus, EKE can either prevent password chaining but make the protocol slow, or prevent dictionary attacks without protecting against password chaining. Dual-workfactor encrypted key exchange allows EKE to be used efficiently to prevent both of these attacks.

PA-ENC-DH simultaneously solves three long-standing limitations of Kerberos: vulnerability to dictionary attacks, dependence on insecure time synchronization, and vulnerability to password chaining. There are other solutions for at least the first two of these problems, notably password quality and secure time protocols, but none of them solve all three problems simultaneously and with such simplicity. Although other public-key modifications of the Kerberos protocol have been proposed that accomplish some or all of these goals, only PA-ENC-DH does so without introducing extra protocol exchanges or requiring substantial protocol modification. PA-ENC-DH also preserves the primarily symmetric-key nature of Kerberos, giving it both higher performance and better long-term security prospects than purely public-key based systems.

The PA-ENC-DH protocol can be varied and generalized in many ways. It applies to essentially any password-based key server protocol, not just Kerberos. Any number of different workfactors can be employed to provide an appropriate level protection for multiple categories of transactions. Diffie-Hellman exponential key exchange can be replaced with many other public-key cryptographic algorithms, notably including elliptic curve cryptography, which may provide substantial practical benefits such as greater resistance to cryptanalysis.

Naturally, PA-ENC-DH should not be used as an excuse to lessen the emphasis on choosing quality passwords nor on protecting those passwords carefully. A password-based protocol will still be more secure with PA-ENC-DH and a good password than with PA-ENC-DH and a bad password.

7 Acknowledgements, patents

Don Davis and Brian LaMacchia reviewed early drafts of this paper and provided substantial helpful discussion and analysis of PA-ENC-DH. Don also introduced me to Bellovin and Merrit's previous work in this area after reviewing an initial sketch of this paper.

Diffie-Hellman exponential key exchange is covered by a U.S. patent[13] which expires in March, 1997. Bellovin and Merrit's Encrypted Key Exchange is covered by a U.S. patent[3] which expires in August, 2010.

References

[1] Steven M. Bellovin and Michael Merrit. Encrypted key exchange: Password-based protocols secure against dictionary attacks. In *Proceedings of the IEEE Symposium on Research in Security and Privacy*, May 1992.

[2] Steven M. Bellovin and Michael Merritt. Limitations of the Kerberos authentication system. In *USENIX Conference Proceedings*, pages 253–267, Dallas, TX, Winter 1991. USENIX. A version of this paper was published in the October, 1990 issue of Computer Communications Review.

[3] Steven M. Bellovin and Michael Merritt. Cryptographic protocol for secure communications. U.S. Patent 5,241,599, August 1993.

[4] Matt Bishop. Anatomy of a proactive password checker. In *Proceedings of the 3rd USENIX UNIX Security Symposium*, Baltimore, MD, September 1992.

[5] CCITT. Recommendation X.509: The Directory Authentication Framkework, December 1988.

[6] Don Davis. Personal communication.

[7] Don Davis, Daniel Geer, and Theodore Ts'o. Kerberos with clocks adrift: History, protocols, and implementation. In *Proceedings of the 5th USENIX UNIX Security Symposium*, Salt Lake City, June 1995.

[8] W. Diffie and M. E. Hellman. New directions in cryptography. *IEEE Trans. Inform. Theory*, IT-22:644–654, November 1976.

[9] Whitfield Diffie, Paul C. Van Oorschot, and Michael J. Wiener. Authentication and authenticated key exchanges. *Designs, Codes, and Cryptography*, 2(2):107–125, June 1992.

[10] Yun Ding and Patrick Horster. Undetectable on-line password guessing attacks. *Operating Systems Review*, 29(4):77–86, October 1995.

[11] D. Eastlake, S. Crocker, and J. Schiller. RFC 1750: Randomness Recommendations for Security, December 1994.

[12] David C. Feldmeirer and Philip R. Karn. UNIX password security - ten years later. In G. Brassard, editor, *Proc. CRYPTO 89*, pages 44–63. Springer-Verlag, 1990. Lecture Notes in Computer Science No. 435.

[13] M. Hellman and R. C. Merkle. Public key cryptographic apparatus and method. U.S. Patent 4,218,582, August 1980.

[14] IEEE P1363 working group. Standard for RSA, Diffie-Hellman, and Related Public Key Cryptography: Appendix E, November 1995. This is a draft document currently available at http://stdsbbs.ieee.org/groups/1363.

[15] P. Karn and W. A. Simpson. The Photuris session key management protocol, November 1995. This is a working draft document of the IETF.

[16] John Kohl and Clifford Neuman. RFC 1510: The Kerberos Network Authentication Service (V5), September 1993.

[17] B.A. LaMacchia and A.M. Odlyzko. Computation of discrete logarithms in prime fields. In A.J. Menezes and S. A. Vanstone, editors, *Proc. CRYPTO 90*, pages 616–618. Springer-Verlag, 1991. Lecture Notes in Computer Science No. 537.

[18] T. Mark A. Lomas, Li Gong, Jerome H. Saltzer, and Roger M. Needham. Reducing risks from poorly chosen keys. In *Proceedings of the 12th ACM Symposium on Operating System Principles*, pages 14–18, December 1989.

[19] R. M. Needham and M. D. Schroeder. Using encryption for authentication in large networks of computers. *Communications of the ACM*, 21(12):993–999, December 1978.

[20] B. Clifford Neuman and Theodore Ts'o. Kerberos: An authentication service for computer networks. *IEEE Communications Magazine*, 32(9):33–38, September 1994.

[21] Clifford Neuman and Glen Zorn. Integrating single-use authentication mechanisms with Kerberos, November 1995. This is a working draft document of the IETF.

[22] S. C. Pohlig and M. E. Hellman. An improved algorithm for computing logarithms over $GF(p)$ and its cryptographic significance. *IEEE Trans. Inform. Theory*, IT-24:106–110, January 1978.

[23] B. Schneier. *Applied Cryptography: Protocols, Algorithms, and Source Code in C.* John Wiley & Sons, New York, 1993.

[24] Eugene Spafford. Observing reusable password choices. In *Proceedings of the 3rd USENIX UNIX Security Symposium*, Baltimore, MD, September 1992.

[25] J.G. Steiner, B.C. Neuman, and J.I. Schiller. Kerberos: an authentication service for open network systems. In *Usenix Conference Proceedings*, pages 191–202, Dallas, Texas, February 1988.

[26] P. C. van Oorschot and M. J. Wiener. On Diffie-Hellman key agreement with short exponents. In *Proceedings of Eurocrypt '96* (to appear). Springer-Verlag, 1996.