Refinement and Extension of *Encrypted Key Exchange*

Michael Steiner Gene Tsudik Michael Waidner
Communications and Computer Science Department
IBM Zürich Research Laboratory
CH-8803 Rüschlikon, Switzerland
tel: +41.1.724-8308, fax: 41.1.710-3608
email: {sti,gts,wmi}@zurich.ibm.com

December 7, 1994

Abstract

In their recent paper, "Encrypted Key Exchange: Password-based Protocols Secure Against Dictionary Attacks," Bellovin and Merritt propose a novel and elegant method for safeguarding weak passwords. This paper discusses a possible weakness in the proposed protocol, develops some enhancements and simplifications, and provides a security analysis of the resultant minimal EKE *protocol. In addition, the basic 2-party EKE model is extended to the 3-party setting; this yields a protocol with some interesting properties. Most importantly, this paper illustrates, once again, the subtlety associated with designing password-based protocols.*

1 Introduction

The *Encrypted Key Exchange* paper [1] (hereafter referred to as simply *EKE*) presents a novel and elegant method of protecting weak secrets from dictionary attacks. It develops several protocol variants based on different underlying cryptosystems, e.g., RSA, El-Gamal, and Diffie-Hellman. The 'generic' version of EKE is illustrated in Figure 1.[1] (Our notation, although unconventional, is largely similar to Bellovin and Merritt's in [1]. We use $X(Y)$ to denote encryption of input Y under the key X.)

1. $A \Longrightarrow B$	$A, P(E_a)$
2. $B \Longrightarrow A$	$P(E_a(K))$
3. $A \Longrightarrow B$	$K(C_a)$
4. $B \Longrightarrow A$	$K(C_a, C_b)$
5. $A \Longrightarrow B$	$K(C_b)$

Figure 1: Generic EKE

Briefly, the protocol begins with A generating a random key-pair (E_a, D_a) of some public key encryption scheme. Then, A sends to B the encryption of E_a under the password P (a weak shared secret.) B generates a new session key, K, encrypts it with E_a, super-encrypts the result with P, and forwards it back to A. The

[1] Taken from Figure 7 in [1].

remainder of the protocol – flows 3, 4, and 5 – represent standard hand-shaking that follows key distribution. (C_a and C_b denote A's and B's challenge values.)

2 Possible Attack

As noted in the EKE paper, the generic EKE protocol is susceptible to the type of attack that, for lack of better term, we shall call *Denning-Sacco Attack* or DS for short. This attack was first illustrated by Denning and Sacco [5] in their critique of Needham and Schroeder's seminal paper [9].

The attack proceeds as follows:

The attacker manages to obtain one of the session keys used in one run of a key distribution protocol. Armed with that knowledge, the attacker is then able to impersonate one of the parties indefinitely often.

As stated in the EKE paper, this attack does not directly apply to generic EKE, However, a more subtle variation does:

Again, the attacker somehow obtains one of the session keys distributed in one (recorded) run of EKE. Armed with that knowledge, the attacker mounts a dictionary attack on the password and, upon breaking the password, is able to impersonate one of the parties indefinitely.

In more detail, the DS attack is as follows:

- The attacker records one run of generic EKE and somehow obtains the key K.
- Iterating upon all possible choices of P:
  1. Pick a candidate $\bar{P}$
  2. Compute $\bar{E}_a = \bar{P}^{-1}(\ P(E_a)\ )$ where $P(E_a)$ is taken from flow 1 of the recorded run.
  3. Compute $\bar{E}_a(K)$ (only if $\bar{E}_a$ is a valid key)
  4. Compute $P(\ \bar{E}_a(K)\ )$ and compare it to $P(E_a(K))$ from recorded flow 2.

A match in the last step indicates correct guess of the password and earns the attacker *carte blanche* with respect to impersonating A.[2]

Bellovin and Merritt considered the above attack in their original paper and proposed a strengthening extension to EKE. In it, flow 3 becomes $K(C_a, S_a)$ where S_a is another random quantity generated by A. Flow 4 then becomes: $K(C_a, C_b, S_b)$ and flow 5 is changed in a similar way. The true session key S is then computed by both parties as $f(S_a, S_b)$ where f is a suitable one-way function.

1. $A \Longrightarrow B$	$A, P(E_a)$
2. $B \Longrightarrow A$	$P(E_a(K))$
3. $A \Longrightarrow B$	$K(C_a, S_a)$
4. $B \Longrightarrow A$	$K(g(C_a), C_b, S_b)$
5. $A \Longrightarrow B$	$K(g(C_b))$

g - strong one way function

Figure 2: Strengthened EKE

[2] In a recent follow-on to the EKE paper [2], Bellovin and Merritt proposed an augmented EKE protocol which has an additional benefit of being secure against the password file compromise. The resultant protocol includes (as flow 5) a field: $K(F(P, K))$ where "F is a one-way function which depends on both the password and the previously negotiated session key." An intruder who discovers one session key K can mount a dictionary attack on P using just this field.

The extension makes the DS attack *hard* since obtaining a session key S does not bring the attacker much closer to breaking P. The attacker can still try to break K in order to get a shot at a dictionary attack on P. A brute-force attack on K is, of course, possible since the attacker can iterate on all possible choices of K and for each $\bar{K}$:

1. Decrypt flows 3 and 4 with $\bar{K}$ and extract $\bar{C_a}, g(\bar{C_a})$;
2. Compute $g(\bar{C_a})$ and compare to $g(\bar{C_a})$.

However, this attack is difficult because K can be a much stronger key than S.

The only drawback we find with the extended EKE is its relative complexity. It requires 5 messages and 5 encryption operations over the total of 8 data blocks. At the same time, the protocol is general enough so that making it more efficient is likely to introduce unexpected vulnerabilities when adapting it to specific cryptosystems. For this reason, we consider only the least complicated variant of EKE – with Exponential Key Exchange.

3 EKE with Diffie-Hellman Exponential Key Exchange

The Exponential Key Exchange EKE variant (referred to as EKE-DH from here on) is illustrated in Figure 3. (Readers unfamiliar with the details of Diffie-Hellman key exchange are referred to the original paper [6] or any of a number of recent texts, e.g., [12, 13].) EKE-DH appears to be the most practical EKE variant because of the relative simplicity of the Diffie-Hellman key exchange.

1. $A \Longrightarrow B$	$A, P(R_a)$
2. $B \Longrightarrow A$	$P(R_b), K(C_b)$
3. $A \Longrightarrow B$	$K(C_a, C_b)$
4. $B \Longrightarrow A$	$K(C_a)$

g	base exponent
β	modulus, all exponentiation is modulo β
N_a, N_b	A's and B's randomly-chosen exponents
R_a, R_b	A's and B's residues; $R_a = g^{N_a}$, $R_b = g^{N_b}$
$K()$	encryption with session key $K = g^{N_a * N_b} mod \beta$

Figure 3: EKE using Exponential Key Exchange

Unlike generic EKE, EKE-DH appears to be resistant to the DS attack. Its resistance is due largely to the fact that the key is never communicated in any way. Instead, only residues ($mod \beta$) are communicated in encrypted form. Even if the attacker obtains both residues, R_a and R_b, he does not come closer to discovering K. Conversely, if the attacker somehow discovers K, he cannot validate correct guesses of R_b and R_a thus making a dictionary attack impossible. Therefore, DH-EKE does not require strengthening of the kind shown in Figure 2.

One of the improvements suggested in [1] is to omit one of the two P-based encryptions (in flows 1 or 2.) The EKE authors *judiciously* chose as example the elimination of the encryption in flow 1. (The result is that A sends $[A, R_a]$.)

It is, perhaps, interesting to note that omitting the encryption of R_b in flow 2 (while keeping the one in flow 1) results in a protocol vulnerable to **dictionary attack**. This is because sending R_b in the clear gives the attacker an opportunity to select N_b and the corresponding R_b. All the attacker has to do is masquerade as B in flow 2 and, in flow 3, receive $K(C_a, C_b)$ from A. Then, for every password guess, the attacker:

1. Decrypts $P(R_a)$ from flow 1 and constructs a candidate key $\bar{K}$;
2. Decrypts $K(C_b)$ (which the attacker himself concocted in flow 2) with $\bar{K}$, obtains $\bar{K}^{-1}(K(C_b))$;
3. Decrypts $K(C_a, C_b)$ (received from A in flow 3), obtains $\bar{K}^{-1}(K(C_a, C_b))$;
4. If the second half of $K^{-1}(K(C_a, C_b))$ matches $\bar{K}^{-1}(K(C_b))$, the password guess is correct.

This vulnerability is easy to fix by slight re-arrangement of protocol messages as shown in Figure 4. As A can authenticate B already after flow 2 it can stop the protocol run in case the challenge C_a does not match and therefore prevents the above attack.

1. $A \Longrightarrow B$	$A, P(R_a), C_a$
2. $B \Longrightarrow A$	$R_b, K(C_b, C_a)$
3. $A \Longrightarrow B$	$K(C_b)$

Figure 4: Mutated DH-EKE

4 A Finishing Touch

One of the advantages of using Diffie-Hellman key exchange is its inherent "democracy" – both parties contribute equally to the resultant key. This and the relative simplicity of implementation make it quite attractive and practical to implement, especially in low-end environments such as smartcards and cryptographic calculators. In such environments, one is typically concerned with protocol efficiency which manifests itself in several forms:

- Number of protocol flows
- Number of rounds
- Number of cryptographic operations
- Total amount (length) of data communicated
- State retention in the course of running the protocol

Two-party challenge-based protocol requires a minimum of three flows and three rounds. DH-EKE (in Figure 3) has four flows and four rounds. At least two encryption operations using the newly-distributed key are required for mutual authentication. As stipulated in [1], at least **one** password-based encryption is clearly required. DH-EKE has two encryptions with P and three encryptions (over a total of four data blocks) with K. (The authors point out that one of the two P-based encryptions can be omitted.) Assuming that all data units are of the same length – names, residues, challenges and encryptions thereof – DH-EKE (in Figure 3) communicates seven data blocks. It can be shown that the absolute minimum is only five data blocks.

In light of these considerations, the minimal EKE (M-EKE) protocol in Figure 5 represents a more efficient version of DH-EKE. We will describe this protocol and analyze its security in more detail in the Appendix.

f can be any of a number of simple functions, e.g., a one-way hash function such as MD5. Alternatively, we can set $f(R_b) = K(R_b)$ which makes flow 3 simply $K(K(R_b))$.

In summary, M-EKE consists of three messages and three rounds. It is not possible to cut down on either. This can be shown following the ideas of [7]. M-EKE requires three encryption operations: one P-based and two K-based. All three are over a single data block. A total of five data blocks are exchanged in the

1. $A \Longrightarrow B$	$A, P(R_a)$
2. $B \Longrightarrow A$	$R_b, K(R_b)$
3. $A \Longrightarrow B$	$K(f(R_b))$

Figure 5: Minimal EKE

course of the protocol; it is easy to see that less than five is impossible (unless A in flow 1 is evident from the context) as already the challenge-response handshake requires two times two blocks. Both A and B generate only one random number each.

Another advantage of M-EKE is that it does not require additional strengthening against cryptoanalytic attacks (as in Figure 2) since knowledge of a session key K does not open the door for either dictionary attacks on P or attacks on other session keys. The only strengthening detail that the protocol needs has to do with the computation of the session key K. Instead of setting $K = g^{N_a * N_b} mod \beta$ as in traditional Diffie-Hellman key exchange, we set $K = H(g^{N_a * N_b} mod \beta)$ where H is a strong one-way hash function. This is necessary to avoid so-called narrowing attacks. (See Appendix.)

5 3-Party EKE

The minimal EKE protocol (of Figure 5) can be extended to the 3-party setting. As shown in Figure 6 this is achieved by letting the trusted third-party (S) act as a relay (as in the well-known *Wide-Mouthed-Frog* protocol [4].)

1. $A \Longrightarrow B$ $\quad P_A(R_a \oplus B)$
2. $B \Longrightarrow S$ $\quad A, P_A(R_a \oplus B),\ P_B(R_b \oplus A)$
3. $S \Longrightarrow B$ $\quad {R_a}^{N_s}, {R_b}^{N_s}$

B computes $({R_a}^{N_s})^{N_b} = g^{N_a * N_s * N_b}$

4. $B \Longrightarrow A$ $\quad {R_b}^{N_s}, C_{ba}$

A computes $({R_b}^{N_s})^{N_a} = g^{N_b * N_s * N_a}$

5. $A \Longrightarrow B$ $\quad C_{ab}$

R_a	g^{N_a}
R_b	g^{N_b}
N_s	strong nonce generated by S
K_{ab}	$g^{N_a * N_b * N_s}$
C_{ba}	authenticator $B \to A$, e.g, $K_{ab}(flow1)$
C_{ab}	authenticator $A \to B$, e.g, $K_{ab}(C_{ba})$

Figure 6: 3-Party EKE

The present protocol performs the following tasks:[3]

- Secure distribution of session key K_{ab} to A and B
- Mutual authentication of A and B
- (Indirect) authentication of S to A and S to B

[3] A protocol very similar to the one presented in this section has been independently and concurrently constructed by Wendo Mao and Colin Boyd in [14]

Although the protocol provides no authentication of either A or B to S, the resultant session key K_{ab} is available only to the *bona fide* principals A and B.

One curious feature of this protocol is that the resultant session key, although contributed to and strengthened by S, is not disclosed to S. (Of course, we have to bear in mind that a *dishonest* S can always subvert the protocol by pretending to be either A or B.) The key is constructed from three equally-weighted contributions – N_a, N_b and N_s – from A, B and S.

Just like its 2-party counterpart, 3-party EKE is resistant to dictionary and verifiable plaintext attacks by outsider adversaries. The protocol is also resistant to insider attacks by dishonest A or B. This is because no verifiable (based on password) ciphertext is revealed in the course of the protocol. However, S has to make an effort to check for trivial (e.g., identity) values of R_a and R_b in order to prevent either A or B from obtaining N_s.

One interesting detail of this protocol is that is seems to be *impossible* to provide any kind of authenticated key distribution in this context. In other words, any kind of a strong integrity check provided for the benefit of either A or B in flow 3 would invariably create an opportunity for either a known-plaintext or a verifiable-text attack.

As far as protocol minimality, key distribution without the authentication handshake takes four messages and four rounds assuming current connectivity scenario, i.e., $A \leftrightarrow B \leftrightarrow S$. If both A and B can contact S concurrently and independently, protocol flows 1 and 2 can be collapsed into a single round. The same holds for flows 3 and 4. Thus, the entire protocol can be done in two rounds.

A and B each perform two exponentiations: one to compute R_a/R_b and one to compute K_{ab}. In addition, A and B perform one cryptographic operation each to compute $P_A(R_a \oplus B)$ and $P_B(R_b \oplus A)$, respectively.

The 3-Party EKE protocol is particularly well-suited for environments where nothing but a secure pseudo-random number generator is available to every user. As an illustration, we consider an environment where users have at their disposal a pool of *impersonal*, *unregistered* hardware devices – token cards. The number of cards can be much smaller than the number of users as long as the cards are tamper-proof. The cards are not registered anywhere and their whereabouts and current ownership are not tracked. A user is free to pick any card from the pool, use it at will and select any other card at a later time. Each card implements nothing more than a strong pseudo-random number generator (which produces $N_{a/b}$ values.) It is not difficult to see how such token cards can be integrated into the 3-Party EKE protocol. (Provided, of course, that the user can enter his password/PIN into a keypad on the card.)

6 Conclusions

The original EKE protocol introduced a novel method of protection against off-line dictionary attacks but were not very efficient. In this paper we demonstrated that the EKE protocol based on Diffie-Hellman key exchange can be improved in the number of protocol flows, rounds and cryptographic operations. We also illustrated the subtlety associated with designing password-based protocols.

We then presented a protocol which reaches the theoretical lower bound for two-party challenge-based protocols of three rounds and flows as well as the minimum of five data blocks. This protocol achieves perfect forward security and is shown to withstand dictionary attacks.

Finally, we extended the Diffie-Hellman-based EKE to the 3-party settting. The resultant protocol inherits the security properties of the 2-party protocol and is also resistant to insider attacks.

References

[1] S. Bellovin and M. Merritt, *Encrypted Key Exchange: Password-based Protocols Secure Against Dictionary Attacks*, **IEEE Symposium on Research in Security and Privacy**, May 1992.

[2] S. Bellovin and M. Merritt, *Augmented Encrypted Key Exchange: Password-based Protocol Secure Against Dictionary Attacks and Password File Compromise*, **1st ACM Conference on Computer and Communications Security**, November 1993.

[3] M. Bellare and P. Rogaway, *Entity Authentication and Key Distribution*, Crypto '93 LNCS 773, Springer-Verlag, Berlin 1994, 232-249.

[4] M. Burrows, M. Abadi and R. Needham, *A Logic of Authentication* Technical Report 39, DEC System Research Center, February 1990.

[5] D. Denning and G. Sacco, *Timestamps in Key Distribution Systems*, **Communications of the ACM**, August 1981.

[6] W. Diffie and M. Hellman, *New Directions in Cryptography*, IEEE Transactions on Information Theory, November 1976.

[7] L. Gong, *Lower bounds on messages and rounds for network authentication protocols.* **1st ACM Conference on Computer and Communications Security**, November 1993.

[8] National Bureau of Standards, *Federal Information Processing Standards*, National Bureau of Standards, Publication 46, 1977.

[9] R. Needham and M. Schroeder, *Using Encryption for Authentication in Large Networks of Computers*, Communications of the ACM, December 1978.

[10] R. Needham and M. Schroeder, *Authentication Revisited*, ACM Operating Systems Review, Vol. 21, No. 7, January 1987.

[11] R. Rivest, *The MD5 Message Digest Algorithm*, Internet DRAFT, July 1991.

[12] B. Schneier, *Applied Cryptography (Section 12.2)*, New York, NY:Wiley, 1994.

[13] D. Davies and W. Price, *Security for Computer Networks,* 2nd Edition, New York, NY:Wiley, 1989.

[14] W. Mao and C. Boyd, *Cryptographic Key Establishment Secure Against Exhaustive Key Search* **DRAFT, submitted to 1995 IEEE Symposium on Research in Security and Privacy**, available from *wenbo@comms.ee.man.ac.uk*, Fall 1994.

A Security Considerations for M-EKE

We now argue that M-EKE – as shown in Figure 5 – is a **secure** protocol. We begin by describing the protocol in some more detail.

A.1 Extended Protocol Description

To initialize M-EKE, we have to exchange the passwords and we have to choose the group used for the Diffie-Hellman key exchange (e.g., the multiplicative group of a prime field GF(p) where all computations are done modulo p). The set of all passwords and the group are chosen by a probabilistic polynomial time algorithm $\mathcal{X}$ that receives as an input the security parameter 1^k (i.e., k in unary representation — as usual in cryptography). $\mathcal{X}(1^k)$ outputs a description of the password set, $\mathcal{P}$, a description of the group, G, a generator of G, g, and n, the order of G. (In reality, $\mathcal{P}$ is a "small" set whose size is independent of k. But for our security analysis, we assume first that passwords are strong secrets; thus, we need this generality.)

We assume that passwords are randomly chosen from $\mathcal{P}$, and shared as required. Especially, A and B are entities that share password $P \in_{\mathcal{R}} \mathcal{P}$.[4]

Furthermore, we need to make an assumption about the basic security of the Diffie-Hellman key exchange.

Diffie-Hellman Assumption: Let $\mathcal{A}$ be any algorithm that has runtime polynomial in the first input, and let (G, g, n) be the last 3 values produced by a run of $\mathcal{X}(1^k)$.

1. Let $R_a, R_b \in_{\mathcal{R}} G$ and $b \in_{\mathcal{R}} \{0,1\}$. If $b = 0$, then assign $K \in_{\mathcal{R}} G$, otherwise $K \leftarrow g^{log(R_a)log(R_b)}$. Then, $\mathcal{A}_1(1^k, G, g, n, R_a, R_b, K)$ cannot approximate b with probability significantly larger the 0.5.
2. Let $R_b, K \in_{\mathcal{R}} G$ and $b \in_{\mathcal{R}} \{0,1\}$. If $b = 0$, then assign $R_a \in_{\mathcal{R}} G$, otherwise $R_a \leftarrow g^{log(K)/log(R_b)}$. Then, $\mathcal{A}_2(1^k, G, g, n, R_a, R_b, K)$ cannot approximate b with probability significantly larger the 0.5.

Informally, this means that R_a, R_b (or R_b, K) do not reveal any single bit of K (or R_a). The first one is the usual assumption made for all Diffie-Hellman key exchange protocols. The second one is needed since the secrecy of R_a is fundamental for the prevention of dictionary attacks on P.

[4] $x \in_{\mathcal{R}} X$ means that x is randomly chosen or randomly distributed in Set X.

We assume that the security parameter k is fixed, and that all parties receive $(\mathcal{P}, G, g, n)$ as a global input, and their passwords, P, as required.

Additionally, we need two families of functions: $\mathcal{F}$ for the first message, where P is used as an index, and $\mathcal{H}$ for the second and third message, where K is used as an index. Both are also chosen during initialization by probabilistic polynomial time algorithms, on input $(1^k, \mathcal{P}, G, g, n)$.

We assume that $\mathcal{F}$ is a family of invertible random permutations on G whose index set is the set of all passwords. To make our analysis easier, we assume that $\mathcal{F}$ is chosen randomly from the set of all families of permutations on G that can be enumerated be $\mathcal{P}$. This is a random oracle assumption, i.e., it is not constructive (replacing it by a cryptographic assumption would be possible, but it would make things more complicated).

Similarly, we assume that $\mathcal{H}$ is a family of random functions with domain and range G, in the same sense as above.

A		B
$N_a \in_{\mathcal{R}} \{0, \ldots, n-1\}$; $R_a \leftarrow g^{N_a}$.	$\xrightarrow{A, \mathcal{F}_P(R_a)}$	receive (A, X_a) and retrieve P; $R'_a \leftarrow \mathcal{F}_P{}^{-1}(X_a)$; $N_b \in_{\mathcal{R}} \{0, \ldots, n-1\}$; $R_b \leftarrow g^{N_b}$; $K \leftarrow R'_a{}^{N_b}$.
receive (R'_b, X_b); $K' \leftarrow R'_b{}^{N_a}$; $X_b \stackrel{?}{=} \mathcal{H}_{K'}(R'_b)$; accept key K'.	$\xleftarrow{R_b, \mathcal{H}_K(R_b)}$	
	$\xrightarrow{\mathcal{H}_{K'}(gR'_b)}$	receive Y; $Y \stackrel{?}{=} \mathcal{H}_K(gR_b)$; accept key K.

Figure 7: Extended Description of the M-EKE

A.2 Security Analysis

We do not give a formal proof of a security, but an informal argument (a formal proof would follow the ideas of [3]):

First we will show that M-EKE as described in Figure 7 is a secure key exchange protocol, based on what we required above. This means that P is assumed to be a strong key, not just a password. Then we will relax our assumption about P, and show that, even as a weak password, P is not susceptible to exhaustive search attacks, e.g., dictionary attacks.

A.2.1 Security Based on Strong Passwords

We will show the following:

Correctness: Protocol M-EKE is correct, i.e., if A and B follow the protocol and all messages are relayed correctly, they both accept, and $K = K'$. Furthermore, K is randomly distributed in G.

Authentication of A: If A accepts, then B must have received the first message of the protocol, and B will accept if A sends the last message, i.e., each accepting conversation of A is matched by a conversation of B.

Authentication of B: If B accepts, then A must have sent the first and last message, and A has already accepted, i.e., each accepting conversation of B is matched by an accepting conversation of A.

Secrecy of K: If both, A and B accept, then no adversary can distinguish K from a random value in G with probability significantly greater than 0.5.

Correctness is almost obvious. The randomness of K follows as usual for DH Key Exchange: Since $N_a, N_b \in_{\mathcal{R}} \{0, \ldots, n-1\}$, key $K = g^{N_a N_b}$ is a random element of G, as required (and as usual for DH).

Authentication of A follows from the fact that $\mathcal{H}_K$ is assumed to be a random function:

If A accepts, A has received the second message, and this message has passed the test. Passing the test proves that the sender (presumably B) knows K.

This follows from the fact that $\mathcal{H}_K$ is a random function, and without knowledge of K, the event that the adversary has chosen X_b correctly has negligible probability, even after a chosen message attack on $\mathcal{H}_K$.

Since N_a is secret and the only information on N_a was revealed in the first message, encrypted by $\mathcal{F}_P$, the event that the sender knows $K = {R'_b}^{N_a}$ implies that he must have received the first message, almost certainly.

Furthermore, this sender of the second message must have been able to read R_a, which implies that he knew P. This is due to our assumption that $\mathcal{F}$ is a family of random permutations.

Together, this implies that there is a matching conversation with an entity knowing P — which is assumed to be B.

Authentication of B follows almost the same way as authentication for A.

If B accepts, it must have received matching first and third messages. The third message proves that its sender (presumably A) knows K (the same way as before; note that in our perfect function family there is no way to infer $\mathcal{H}_K(gR_b)$ from $\mathcal{H}_K(R_b)$).

The DH-Assumption ensures that knowledge of K implies more than knowledge of R_a and R_b, i.e., the sender of the third message must have been the sender of the first one. Further, this entity must have accepted (according to the protocol), and it must have received the second message sent by B.

Thus, the conversations match, and B knows that it is talking with an entity knowing P — which is assumed to be A.

Secrecy of the Key follows from the DH-Assumption:

In the worst case, an adversary knows R_b from the second message, and he might know R_a from the first message, by some miraculous inspiration[5]. The only additional information on K the adversary receives are two applications of $\mathcal{H}_K$, where K is the key that should be secret. Since we assumed that $\mathcal{H}$ a family of random functions, these two images do not reveal any information on K. This holds even after a chosen message attack on $\mathcal{H}_K$, since R_b and therefore gR_b are randomly values in G.

A.2.2 Weak shared secrets

The analysis of the preceding section was based on the assumption that $\mathcal{F}$ is a family of random permutations, enumerated by $\mathcal{P}$. This means that A and B do not simply share a weak password, but a random function. If we change our oracle assumptions into the more realistic cryptographic assumptions (pseudo-random permutations) this would require P to be of size polynomial in k.

In reality, P might be fixed at 64 bits or even less. Therefore, the best we can do is to show that P can not be revealed by exhaustive search within the protocol.

There are two forms of exhaustive search attacks: i) passive dictionary attacks, where the adversary records some conversations and tries to figure out the correct P, and ii) active attacks, where the adversary sends some messages to A or B and utilizes the elicited responses in order to break P.

Dictionary attacks are not possible in M-EKE: Even if the adversary discovers a session key K, he is not able to compute any information on R_a from K and the public R_b — this is why we need the second part of our DH-Assumption. Thus, for the adversary, R_a is just a random value in G, and the first message does not enable the adversary to attack P.

Active attacks inside the protocol are not possible since, even if the attacker forges the first message, B will always produce and send a second message. Forging the second message sent to A has no advantage over a dictionary attack.

[5] Note that this is really the worst case, since knowledge of R_a would enable a dictionary attack on P. But even then, the session key K could not be exposed afterwards.