The AuthA Protocol for Password-Based Authenticated Key Exchange

Contribution to IEEE P1363, and its study group looking at new projects

Mihir Bellare* Phillip Rogaway†

March 14, 2000

Abstract

We suggest a simple protocol, AuthA, for the problem of password-based authenticated key exchange (AKE). We assume the asymmetric trust model: the client A has a password pwa and the server B has a particular one-way function of this, pwb. Two flows of the protocol comprise a Diffie-Hellman key exchange, using a group on which the Diffie-Hellman problem is hard. At least one of these two flows is encrypted using the key pwb. Then an authentication tag, $AuthA$, is flowed from the client to the server. This tag is just the hash of some values easily computable by both parties. The server checks the received tag prior to accepting the session key.

The protocol just sketched provides security against dictionary attack, and it ensures forward secrecy and client-to-server authentication. Server-to-client authentication can be added cheaply, by flowing a second authentication tag, $AuthB$, from server to client.

Like most work in this area, our protocol springs from ideas of Bellovin and Merritt [BM92, BM93]. There has been a large body of other follow-on to this, including protocol suggestions by [STW95, Ja96, Ja97, Lu97, MS99, Wu98, RCW98, BESW00, BMP00]. But AuthA would seem to be somewhat simpler and more efficient than prior suggestions.

Rigorous proofs and definitions in this domain turn out to be extremely complex, and a proof of security (in the random-oracle model or the ideal-cipher model, under the Diffie-Hellman assumption) is the subject of ongoing work by the authors. Definitions appear in [BPR00], as does a proof for the symmetric protocol at the core of what is described here.

* Dept. of Computer Science & Engineering, University of California at San Diego, 9500 Gilman Drive, La Jolla, CA 92093, USA. E-Mail: mihir@cs.ucsd.edu. URL: http://www-cse.ucsd.edu/users/mihir.

† Dept. of Computer Science, Engineering II Bldg., University of California at Davis, Davis, CA 95616, USA. E-mail: rogaway@cs.ucdavis.edu. URL: http://www.cs.ucdavis.edu/~rogaway.

1 Introduction

SETTING. Consider the scenario in which there are two entities—a client A and a server B—where A holds a password *Password* and B holds a function of this password, $f(\mathit{Password})$. The parties would like to engage in a conversation at the end of which each holds a session key, sk, which is known to nobody but the two of them. There is present an active adversary whose capabilities include enumerating, off-line, the words in a dictionary D, this dictionary being rather likely to include *Password*. In a protocol we deem "good" the adversary's chance to defeat protocol goals will depend on how much she interacts with protocol participants—it won't significantly depend on her expenditure of computation.

This lovely problem, password-based authenticated key exchange (AKE), comes out of two papers of Bellovin and Merritt [BM92, BM93]. These authors brought forth the problem and gave the first proposed solutions. There have been a great many subsequent suggestions for password-based AKE protocols, including the work of Steiner, Tsudik and Waidner [STW95], Jablon [Ja96, Ja97], Lucks [Lu97], Wu [Wu98], Roe, Christianson and Wheeler [RCW98], MacKenzie and Swaminathan [MS99], Boyko, MacKenzie and Patel [BMP00], and Bellare, Pointcheval and Rogaway [BPR00]. Further related work includes Shoup [Sh99] and Halevi and Krawczyk [HK99]. Gong, Lomas, Needham, and Saltzer [GLNS93] were also involved early in this topic, though they focus on a richer trust model.

The purpose of this note is to suggest yet another protocol for this problem. A separate note, currently being written, encourages standardization in this domain and discusses what one should look for in a protocol for password-based AKE.

PROTOCOL AUTHA. The protocol we describe, which we call AuthA, seems to provide the same security properties as the best prior suggestions, but it does so more simply, at lower cost in communications, and with greater versatility. The method is based on the encrypted Diffie-Hellman key exchange of Bellovin and Merritt. There are many choices possible for the underlying group. When it is an appropriate elliptic curve group, the communications cost for AuthA is as little as two flows of (say) 160 bits and one flow of (say) 64 bits. The client and server each perform three exponentiations (multiplications in the language for elliptic curve groups), with trivial computational overhead beyond that. For both client and server, one of these operations may be done off-line. See Section 2 for a description of the protocol.

FURTHER CHARACTERISTICS OF AUTHA. Let us single out some further characteristics of the protocol AuthA.

1. The protocol is in the asymmetric trust model (the client key is different from the server key, with the former being hard to compute from the latter.) If desired, the symmetric model can also be supported by a trivial protocol change.

2. This adversary won't be able to obtain any information about the distributed session key more effectively than by interactively trying the most likely passwords, in order (security against "dictionary attacks").

3. Security is provided against an active adversary who can direct multiple sessions (the "arbitrary interleaving model").

4. Learning already distributed session keys won't help the adversary (security against "Denning-Sacco attack").

5. The distributed session key is not used within the protocol. (This is desirable because premature use of the session key destroys any chance for composibility and provable-security results.)

6. If the adversary learns *pwa* or *pwb* (the client's and server's password-derived key, respectively), still the adversary won't be able to ascertain anything about already distributed session keys ("forward secrecy").

7. Even after learning *pwa* or *pwb*, the adversary won't be able to ascertain anything about session keys if the adversary only eavesdrops.

8. If the adversary learns *pwb* for server B, the adversary will still have to perform a dictionary attack in order to impersonate A or a server $B' \neq B$ (the point of the asymmetric model).

9. The protocol is very simple, and stems from well-known techniques.

10. The protocol supports a variety of "flow architectures" (that is, who speaks to whom when).

11. A variety of groups can be used, including both modular exponentiation and elliptic-curve methods.

12. The protocol always provides client-to-server authentication. It optionally provides server-to-client authentication.

We believe that protocol AuthA meets the definitions of [BPR00], with reasonable bounds, and under reasonable assumptions. However, these definitions are too complicated to explain here.

Let us move on now to describe the protocol.

2 Description of AuthA

2.1 Preliminaries

Protocol AuthA involves two entities, A and B. We refer to A as the *client* and we refer to B as the *server*. Each is named by a string and, for notational simplicity, we will not distinguish in notation between the entity and the string which names it.

There is an underlying client-password *Password* of unknown quality. Client A has a secret *pwa* which is derived from *Password*. Server B has a secret *pwb* which is derived from *pwa*. How *pwa* and *pwb* are determined from *Password* is defined in Section 2.2. It is not our concern how A and B came to hold *pwa* and *pwb*, but likely *Password* was typed in by a human user, client A is executing on behalf of that user, and *pwb* was formerly installed at the server B with which the client will communicate. It is allowed that there be multiple clients or servers who hold keys derived from the same underlying user password.

Operations will be performed in a cyclic group G. We will denote the group operation multiplicatively, so that applying the group operation to an element X a total of $i-1$ times is denoted by exponentiation: X^i. The group is assumed to be given by a generator $\langle g \rangle$. We let $q = |G|$. We assume that G, g and q are well-known, and that there is a fixed representation of group elements as binary strings. There must be a way to efficiently go from group elements to binary strings, and from binary strings to group elements. We will interchangeably write group elements and the strings which represent them. The group G should be a group on which the Diffie-Hellman problem is hard. One possibility is $G = \mathbb{Z}_p^*$, where p is a large prime number. A second possibility is that

G is an appropriate subgroup of $\mathbb{Z}_p^*$. A third possibility is that G is an appropriate elliptic curve group, with well-known parameters. This last case can have efficiency advantages.

Protocol AuthA uses two types of primitives beyond the group operation. The first is a mask-generation function: H and H'. These map strings of effectively arbitrary length to strings of whatever length we need. The second primitive we need is an encryption function: $\mathcal{E}^1$ and $\mathcal{E}^2$. These map group elements into strings under the control of a key which is again a group element. Beware that the properties that the encryption function must possess are different from the customary ones for an encryption scheme. See Section 1 for a description of some possible instantiations of $\mathcal{E}^1$ and $\mathcal{E}^2$.

We summarize the notation introduced so far:

A	The client, or the name of the client.
B	The server, or the name of the server.
G	The underlying group.
q	The size of this group.
Password	The client's password.
pwa	The *Password*-derived key known by the client. This is an element of G.
pwb	The *pwa*-derived key known by B (and A). This is an element of G.
H, H'	Mask-generation functions. Like a cryptographic hash function, but the output-length is whatever is convenient.
$\mathcal{E}^1$, $\mathcal{E}^2$	Encryption functions for use by A and B, respectively.

We will describe two "versions" of protocol AuthA. The "UA version" (unilateral authentication) is slightly more efficient than the "MA version" (mutual authentication). The UA version provides client-to-server but not server-to-client authentication. The MA version provides both.

Section 2.3 describes what messages need to be exchanged for our protocol, but it is open-ended about who speaks first. This is because the application domain may involve special constraints or considerations. For example, [BESW00] explains that in a password-based authenticated key exchange for TLS (the standard corresponding to SSL) it may be desirable for the server to send the first relevant flow, and that this flow should not depend on the client's identity. Separating what messages are exchanged from when they are exchanged, is an approach used to achieve this versatility. We are also open-ended about who encrypts, and how. This is done by speaking in terms of two encryption functions, $\mathcal{E}^1$ and $\mathcal{E}^2$, one of which may be instantiated by the identity function.

Our protocol description is not intended as a bit-level definition. For an area like password-based AKE, where implementation considerations vary a lot, starting with a higher-level framework would seem to be best.

2.2 Deriving keys *pwa* and *pwb*

Convert client-password *Password*, which is a string, into group elements *pwa* and *pwb* as follows:

$$\begin{aligned} pwa &= \text{the group element represented by } H'(A \parallel B \parallel Password) \\ pwb &= g^{pwa} \end{aligned}$$

To carry out our protocol client A will use *pwa* and *pwb*, while server B will use *pwb*.

2.3 Message exchanges

THE ENCRYPTED DH KEY EXCHANGE. The following two steps can be performed in any order.

- Client A chooses a random number $x \in \{1, \ldots, q\}$, computes $X = g^x$, and then computes $X^* = \mathcal{E}^1_{pwb}(X)$. Client A sends X^* to the server.
- Server B chooses a random number $y \in \{1, \ldots, q\}$, computes $Y = g^y$, and then computes $Y^* = \mathcal{E}^2_{pwb}(Y)$. Server B sends Y^* to the server.

Other information (such as the sender's name) may accompany the flows. The parties then continue as follows:

- Client A receives $\underline{Y^*}$, computes $\underline{Y} = \mathcal{D}^2_{pwb}(\underline{Y^*})$, and then computes $\mathit{DiffieHellmanKeyA} = \underline{Y}^x$.
- Server B receives $\underline{X^*}$, computes $\underline{X} = \mathcal{D}^1_{pwb}(\underline{X^*})$, and then computes $\mathit{DiffieHellmanKeyB} = \underline{X}^y$.

The value of $\underline{X^*}$ might differ from X^*, and the value of $\underline{Y^*}$ might differ from Y^*, due to the behavior of an adversary.

AUTHENTICATING AT LEAST A TO B, AND DERIVING THE SESSION KEY. Client A computes the following:

$$\begin{array}{rcl}
\mathit{MasterKeyA} & = & H(A \parallel B \parallel X \parallel \underline{Y} \parallel \mathit{DiffieHellmanKeyA}) \\
\mathit{SessionKeyA} & = & H(\mathit{MasterKeyA} \parallel 0) \\
\mathit{AuthA} & = & H(\mathit{MasterKeyA} \parallel \underline{Y}^{pwa})) \\
[\ \mathit{AuthBcheck} & = & H(\mathit{MasterKeyA} \parallel 2)\ \ ]
\end{array}$$

The value $\mathit{AuthBcheck}$ need only be calculated for the MA version of the protocol.

Server B likewise computes the following:

$$\begin{array}{rcl}
\mathit{MasterKeyB} & = & H(A \parallel B \parallel \underline{X} \parallel Y \parallel \mathit{DiffieHellmanKeyB}) \\
\mathit{SessionKeyB} & = & H(\mathit{MasterKeyB} \parallel 0) \\
\mathit{AuthAcheck} & = & H(\mathit{MasterKeyB} \parallel pwb^y) \\
[\ \mathit{AuthB} & = & H(\mathit{MasterKeyB} \parallel 2)\ \ ]
\end{array}$$

The value AuthB need only be calculated for the MA version of the protocol.

Client A flows AuthA to B. Server B receives flow $\underline{\mathit{AuthA}}$. Server B accepts session key $\mathit{SessionKeyB}$ if and only if $\underline{\mathit{AuthA}} = \mathit{AuthAcheck}$.

In the MA version of the protocol the server B flows AuthB to A. Client A receives $\underline{\mathit{AuthB}}$ and accepts session key $\mathit{SessionKeyA}$ if and only if $\underline{\mathit{AuthB}} = \mathit{AuthBcheck}$.

In the UA version of the protocol the server B does not flow AuthB. In this version of the protocol A accepts session key $\mathit{SessionKeyA}$ as soon as it is calculated.

It is not clear to the authors that the symmetric model should be supported. If that is desired, allow that $\mathit{AuthA} = H(\mathit{MasterKeyA} \parallel 1)$ and $\mathit{AuthAcheck} = H(\mathit{MasterKeyB} \parallel 1)$.

2.4 Mapping the message exchanges into protocol flows

The sequences of flows depicted in Figure 1 capture various possibilities for mapping the message exchanges we have described into protocol flows. We comment that the name of the sender likely accompanies the first flow, but we regard this as an element of the implementation and note that, in general, further information may accompany each flow.

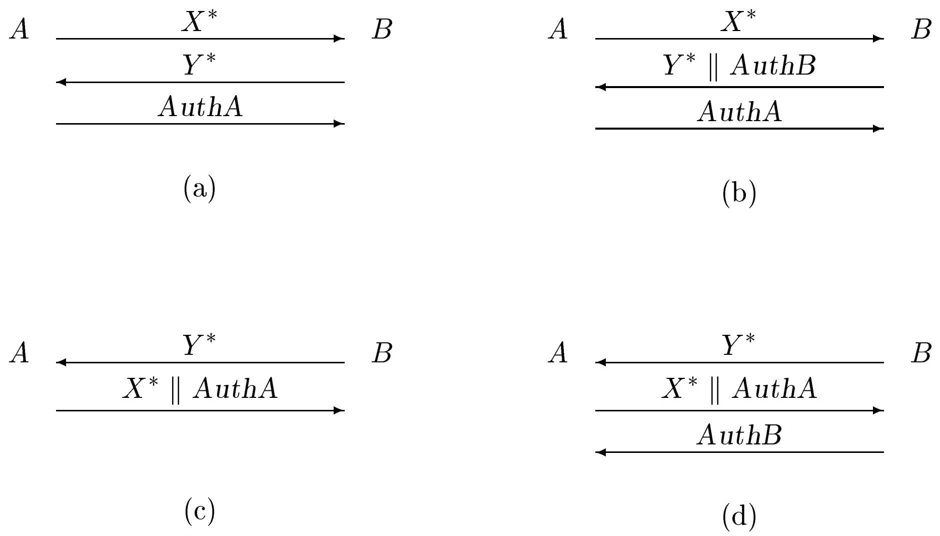

Figure 1: *Possible flow sequences in protocol* AuthA. *Scenarios (a) and (b) are client-initiated. Scenarios (c) and (d) are server-initiated. All four scenarios provide client-to-server authentication. Scenarios (b) and (d) provide server-to-client authentication as well.*

2.5 Instantiating the primitives

Instantiating H and H'. These functions are easily instantiated, using a cryptographic hash function, by applying techniques well-known in P1363. We comment that there is reason to arrange H' such that its computation is intentionally slow. In particular, doing this slows down a dictionary attack by a dishonest server or by an adversary who has obtained the server's database.

Instantiating $\mathcal{E}^1$ and $\mathcal{E}^2$. We believe that security can be proven when $\mathcal{E} \in \{\mathcal{E}^1, \mathcal{E}^2\}$ is realized in one of the following ways, and the underlying group is appropriately chosen:

- By an ideal cipher, $\pi_{pwb}(x)$.
- By $\mathcal{E}_{pwb}(x) = x \cdot H(pwb)$ where H is a random oracle.
- By $\mathcal{E}_{pwb}(x) = (r,\ x \cdot H(r \parallel pwb))$, where H is a random oracle and r is a random string of some appropriate length.

The second of these possibilities is the simplest to concretely instantiate: you apply the mask generation function to pwb, interpret the result as a group element, and multiply by the plaintext. Instantiations which more directly imitate an ideal cipher are also possible. We will give more feedback on desirable instantiations of $\mathcal{E}^1$, $\mathcal{E}^2$ in the future. We warn that incorrect instantiation of the encryption primitive, including instantiates which are quite acceptable in other contexts, can easily destroy the protocol's security.

IDENTITY-INSTANTIATION OF ONE ENCRYPTION FUNCTION. Referring to Figure 1, scenario (a): it is acceptable for $X^* = X$, assuming that $\mathcal{E}^2$ is a proper encryption function. In scenario (b): it is acceptable for $Y^* = Y$, assuming that $\mathcal{E}^1$ is a proper encryption function. In scenario (c): it is acceptable for $X^* = X$, assuming that $\mathcal{E}^2$ is a proper encryption function. In scenario (d): it is acceptable for $X^* = X$, assuming that $\mathcal{E}^2$ is a proper encryption function.

2.6 Comments on known limitations

We'd like to be clear about the following limitations on AuthA:

1. If the server is compromised a dictionary attack is unavoidable.
2. We have not yet worked out a proof for security. This is extremely complex. We are working on it.
3. We do not yet fully understand the assumptions required of the encryption scheme in order to get a proof of security. We are hoping that multiplication by $H(pwb)$ works out.
4. Any valid security proof in this domain is likely to be so complicated as to make verification rather difficult. The protocol is simple, but the definitions and analysis are not.

References

[BCK98] M. Bellare, R. Canetti, and H. Krawczyk. A Modular Approach to the Design and Analysis of Authentication and Key Exchange Protocols. *Proc. of the 30th STOC.* ACM Press, New York, 1998.

[BPR00] M. Bellare, D. Pointcheval and P. Rogaway. Authenticated Key Exchange Secure Against Dictionary Attack. To appear in Eurocrypt 2000.

[BR94] M. Bellare and P. Rogaway. Entity Authentication and Key Distribution. *CRYPTO '93*, LNCS 773, pages 232–249. Springer-Verlag, Berlin, 1994.

[BR95] M. Bellare and P. Rogaway. Provably Secure Session Key Distribution: the Three Party Case. *Proc. of the 27th STOC.* ACM Press, New York, 1995.

[BM92] S. Bellovin and M. Merritt. Encrypted Key Exchange: Password-Based Protocols Secure against Dictionary Attacks. *Proc. of the Symposium on Security and Privacy*, pages 72–84. IEEE, 1992.

[BM93] S. Bellovin and M. Merritt. Augmented Encrypted Key Exchange: A Password-Based Protocol Secure against Dictionary Attacks and Password File Compromise. *Proceedings of the 1st Annual Conference on Computer and Communications Security*, ACM, 1993.

[Bo99] M. Boyarsky. Public-Key Cryptography and Password Protocols: The Multi-User Case. *Proceedings of the 6th Annual Conference on Computer and Communications Security*, ACM, 1999.

[BMP00] V. Boyko, P. MacKenzie and S. Patel. Provably Secure Password Authenticated Key Exchange Using Diffie-Hellman. To appear in Eurocrypt 2000.

[BESW00] P. Buhler, T. Eirich, M. Steiner, M. Waidner. Secure Password-Based Cipher Suite for TLS. Proceedings of Network and Distributed Systems Security Symposium. February 2000.

[GLNS93] L. Gong, M. Lomas, R. Needham, and J. Saltzer. Protecting Poorly Chosen Secrets from Guessing Attacks. *IEEE Journal on Selected Areas in Communications*, 11(5):648–656, June 1993.

[HK99] S. Halevi and H. Krawczyk. Public-Key Cryptography and Password Protocols. February 1999. Earlier version in *Proc. of the 5th CCS.* ACM Press, New York, 1998.

[Ja96] D. Jablon. Strong Password-Only Authenticated Key Exchange. *ACM Computer Communications Review*, October 1996.

[Ja97] D. Jablon. Extended Password Key Exchange Protocols Immune to Dictionary Attacks. *Proc. of WET-ICE '97*, pages 248–255. IEEE Computer Society, June 1997.

[Lu97] S. Lucks. Open Key Exchange: How to Defeat Dictionary Attacks Without Encrypting Public Keys. *Proc. of the Security Protocols Workshop*, LNCS 1361. Springer-Verlag, Berlin, 1997.

[MS99] P. MacKenzie and R. Swaminathan. Secure Authentication with a Short Secret. Manuscript. November 2, 1999. Earlier version as Secure Network Authentication with Password Identification. Submission to IEEE P1363a. August 1999.
Available from `http://grouper.ieee.org/groups/1363/addendum.html`

[MVO96] A. Menezes, P. van Oorschot and S. Vanstone, *Handbook of Applied Cryptography.* CRC Press, 1996.

[RCW98] M. Roe, B. Christianson and D.Wheeler. Secure Sessions from Weak Secrets. Technical report from University of Cambridge and University of Hertfordshire, 1998. Submitted to *Operating Systems Review.*

[Sh99] V. Shoup. On Formal Models for Secure Key Exchange (version 4). Manuscript, November 15, 1999. Proceedings version in *ACM Computer and Communications Security*, 1999.

[STW95] M. Steiner, G. Tsudik and M. Waidner. Refinement and Extension of *Encrypted Key Exchange.* Operating Systems Review, vol. 29, Iss. 3, pp. 22-30 (July 1995).

[Wu98] T. Wu. The Secure Remote Password Protocol. *Proceedings of the Internet Society Symposium on Network and Distributed System Security*, pages 97–111, 1998.