SRC Technical Note
1998 - 007

March 18, 1998

Two Facets of Authentication

Martín Abadi

Systems Research Center
130 Lytton Avenue
Palo Alto, California 94301

http://www.research.digital.com/SRC/

Two Facets of Authentication

Martín Abadi
Digital Equipment Corporation
Systems Research Center
ma@pa.dec.com

March 18, 1998

Abstract

Authentication can serve both for assigning responsibility and for giving credit. Some authentication protocols are adequate for one purpose but not the other. This paper explains the distinction between responsibility and credit, through several examples, and discusses the role of this distinction in the design and analysis of protocols.

Contents

1 Responsibility and Credit

Authentication can serve both for assigning responsibility and for giving credit. An "authenticated" message M from a principal A to a principal B may be used in at least two distinct ways:

- B may believe that the message M is being supported by A's authority. For example, suppose that B is a file server, A a client, and M a request to delete a file f. Then B may use A's identity as an argument to the access control decision of whether to delete f.

- B may attribute credit for the message M to A. For example, suppose that B is running a contest, offering a prize to the principal that mails the factors for a large number. When B receives the message M as an entry, B may give credit for the entry to A.

These two uses are sharply different, and require different concepts of authentication. Furthermore, some natural protocols are adequate for assigning responsibility but not for giving credit, and vice versa. We consider some of those protocols in this paper.

When a new authentication protocol is designed, it is therefore prudent to state whether the protocol is intended to establish responsibility, credit, or both. Similarly, when an authentication protocol is analyzed, it is prudent to clarify whether its properties suffice for establishing responsibility or credit. However, a quick look at the literature (discussed below) suggests that this clarification is not often present.

This paper explains the distinction between responsibility and credit, through several examples, and discusses the role of this distinction in the design and analysis of protocols. Papers by Gollman, Lowe, and others have shed light on several possible definitions of authentication [Gol96, Low97]. This paper does not attempt to review those studies, but aims to complement them.

The two facets of authentication are most clearly separate in protocols that rely on asymmetric cryptosystems, such as the RSA cryptosystem [RSA78]. We therefore take public-key protocols as examples. Although our list of examples is not meant to be comprehensive, we illustrate the distinction between responsibility and credit through several protocols of different styles. Our emphasis is not on classifying cryptographic techniques. (See [MvOV96, Chapter 12] for a helpful classification.) Rather, we focus on the higher-level guarantees that protocols provide to the applications that may rely on them.

The direct author of a message need not always be held responsible for the message or be given credit for it. In particular, a principal may delegate some part of its authority to another principal, taking responsibility for messages sent by the delegate (much like one can let a friend withdraw from one's bank account). Similarly, a principal may legitimately divert credit for its messages to another principal (much like one can deposit into a friend's bank account). Therefore, even when it is proved beyond a reasonable doubt that a principal sent a message, responsibility and credit may not follow. Responsibility and credit are thus distinct from non-repudiation of origin and other related concepts [ZG96, Roe].

2 Four Examples

This section discusses four examples that illustrate the concepts of responsibility and credit, and their applications. The examples concern techniques that are common in published, useful protocols. In all four examples, responsibility and credit for the messages between two parties is assigned by the parties themselves. For simplicity, we do not consider scenarios where a third party (for example, a judge) may assign responsibility or credit, but such scenarios could be interesting too.

2.1 Signing a Public Key

As a first example, we consider a simple protocol where a principal A creates a short-term key pair, sends the short-term public key to a principal B, signing it with its long-term secret key, then uses the short-term secret key for signing further messages:

$$\begin{array}{lll} \text{Message 1} & A \rightarrow B: & A,\, B,\, \{K,\, A,\, B,\, T\}_{K_A^{-1}} \\ \text{Message 2} & A \rightarrow B: & A,\, B,\, \{\{M\}_{K^{-1}}\}_{K_B} \end{array}$$

Here M is an arbitrary message, T is a timestamp, K_A is A's long-term public key, K_A^{-1} is the corresponding secret key, K is A's short-term public key, and K^{-1} is the corresponding secret key. We use braces for signatures, as in $\{M\}_{K^{-1}}$, and for encryptions, as in $\{\{M\}_{K^{-1}}\}_{K_B}$. Message 1 is the core of the protocol; Message 2 appears only as an example of the use of K.

In one interpretation of this protocol, Message 1 conveys that A takes responsibility for the key K, so B can blame A for any message signed with K^{-1}. Thus, the protocol fits applications that require responsibility for a message (or for a connection). For example, the protocol seems adequate for the situation where B is a file server, A a client, and M a request to delete a particular file. It seems adequate even if A gives K^{-1} to a delegate, allowing the delegate to make requests on A's behalf (although delegation mechanisms where B knows the identity of the delegate may be preferable [LABW92]).

In a second interpretation of this protocol, B would assign credit to A for every message that is signed with K^{-1}. This interpretation is not justified, as the following would constitute an attack:

Message 1	$A \rightarrow B:$	$A,\ B,\ \{K,\ A,\ B,\ T\}_{K_A^{-1}}$	(intercepted by C)
Message 1'	$C \rightarrow B:$	$C,\ B,\ \{K,\ C,\ B,\ T\}_{K_C^{-1}}$	
Message 2	$A \rightarrow B:$	$A,\ B,\ \{\{M\}_{K^{-1}}\}_{K_B}$	(intercepted by C)
Message 2'	$C \rightarrow B:$	$C,\ B,\ \{\{M\}_{K^{-1}}\}_{K_B}$	

Here C is an attacker who signs the public key generated by A. On receipt of M, we would have B give credit for M to C rather than to A. This sequence of messages constitutes an attack with the second interpretation, since it may result in erroneous credit to C. On the other hand, with the first interpretation, this sequence of messages may result in erroneous blame to C; it may also result in denial of service to A and B, but has no other direct negative impact on them.

The protocol can be strengthened in a variety of ways. In particular, A may sign its own name with the secret key K^{-1}, for example as in:

$$\begin{array}{lll} \text{Message 1} & A \rightarrow B: & A,\ B,\ \{K,\ A,\ B,\ T\}_{K_A^{-1}},\ \{A\}_{K^{-1}} \\ \text{Message 2} & A \rightarrow B: & A,\ B,\ \{\{M\}_{K^{-1}}\}_{K_B} \end{array}$$

or

$$\begin{array}{lll} \text{Message 1} & A \rightarrow B: & A,\ B,\ \{K,\ A,\ B,\ T\}_{K_A^{-1}} \\ \text{Message 2} & A \rightarrow B: & A,\ B,\ \{\{A,\ M\}_{K^{-1}}\}_{K_B} \end{array}$$

These modification do not prevent an attacker C from signing the key K with its key K_C^{-1}; however, C cannot sign its own name with K^{-1}, since C does not have K^{-1}. Therefore, the protocol becomes adequate for giving credit for M to A.

We may say that, with these modifications, A proves possession of the secret key K^{-1} [MvOV96, Definition 12.7]. However, this statement should not be taken literally. For example, suppose that A is a user with a smart-card and that B is a server. The smart-card may sign a short-term key K generated by the workstation to which the smart-card is connected, and the workstation may sign the name A, while the user and the smart-card may never have access to the secret key K^{-1}.

Protocols similar to this one may be used for registering public keys. In this application, B is a certification authority and A is a client that enters a new public key K into B's registry. It has been argued that, whenever a client is associated with a public key K, the client should prove possession of the corresponding secret key K^{-1} [MvOV96, Remark 13.23]. However, this view is not universal—it is not shared, in particular, in some recent public-key-infrastructure designs [Ell97].

2.2 Encrypting a Session Key

While the first example shows that a signature may lead to responsibility, the second example shows that an encryption may lead to credit, and that a decryption may lead to responsibility.

We consider the situation where a principal A transmits a session key K (say, a DES key [DES77]) to another principal B, encrypting K under B's public key K_B, and including the name A along with K:

$$\begin{array}{lll} \text{Message 1} & A \rightarrow B: & \{A, K\}_{K_B} \\ \text{Message 2} & A \rightarrow B: & \{M\}_K \\ \text{Message 3} & B \rightarrow A: & \{M'\}_K \end{array}$$

Messages 2 and 3 simply illustrate the use of the session key K for sending some messages, M and M'. We assume that both principals have means for recognizing and ignoring their own messages, so for example A will not mistake a replay of $\{M\}_K$ for a message from B.

This protocol is adequate for applications that require responsibility for B, and perhaps for applications that require credit for A:

- Only B can extract K from Message 1. Therefore, B can take responsibility for the use of K. Whenever A receives a message encrypted under K, if A did not generate the message itself, then A can be certain that B generated the message, or that some principal to which B gave K generated the message. So A can hold B responsible for the message.

- Since Message 1 yields no proof of A's identity, B does not know who else has K. Therefore, B should not send any secrets under K. We do not consider the issue of credit for B for lack of a compelling example where B would claim credit for public data. (Perhaps only secrets are worth claiming credit for.)

- Since any principal could have produced Message 1, B cannot hold anyone responsible for messages from A encrypted under K.

- Nevertheless, by including its name in Message 1, A claims credit for messages encrypted under K. Of course, if A chooses K incompetently or maliciously, then another principal C might have K as well, and might also produce messages encrypted under K. Still, since A has K, and assuming that C sends those messages to A or that A is capable of intercepting them, A can read and produce those same messages. Therefore, (some) credit to A is justified.

The last of these points indicates that this protocol may not be a solid basis for unqualified credit to A. The following alternative protocol provides a clearer case for credit to A and preserves the responsibility of B:

$$\begin{array}{lll} \text{Message 1} & A \rightarrow B: & \{J_A\}_{K_B} \\ \text{Message 2} & B \rightarrow A: & J_B \\ \text{Message 3} & A \rightarrow B: & \{M\}_K \\ \text{Message 4} & B \rightarrow A: & \{M'\}_K \\ & (K = H(J_A, J_B, A, B)) & \end{array}$$

where J_A and J_B are random quantities invented by A and B, respectively, and K is computed by applying a one-way hash function H to the concatenation of J_A, J_B, and the names A and B. With this alternative protocol, A can still forward a message M received from another principal, of course, but A cannot pick a session key K in use by other principals.

2.3 Making a Session Key from Encrypted Shares

As a third example, we consider a protocol for obtaining a shared key from encrypted shares. The protocol has been applied in several contexts. It appears in the work of Lampson et al. [LABW92], and it also appears as a component of Krawczyk's SKEME protocol [Kra96]. We adopt Krawczyk's name for this protocol: SHARE.

SHARE enables two principals A and B to obtain a shared key, assuming that initially each knows the public key of the other (K_A or K_B). Each of the principals invents a random quantity (J_A or J_B), as a "half key". Then the following exchange takes place:

$$\begin{array}{lll} \text{Message 1} & A \rightarrow B: & \{J_A\}_{K_B} \\ \text{Message 2} & B \rightarrow A: & \{J_B\}_{K_A} \end{array}$$

After this exchange, the two principals A and B compute a shared key K by applying a one-way hash function to the concatenation of the half keys J_A and J_B.

In his explanation of SHARE, Krawczyk says:

> If A follows the protocol then she is assured that the shared key [...] is not know to anyone except B (though A does not have the assurance that B knows the key). And analogously for B.

These properties should not be interpreted literally (and it is not Krawczyk intention that they be interpreted literally [Kra97]). In fact, in their literal interpretation, these properties do not hold. Consider, for example, the following message sequence:

$$
\begin{array}{lll}
\text{Message 1} & A \to B: & \{J_A\}_{K_B} \\
\text{Message 1'} & B \to C: & \{J_A\}_{K_C} \\
\text{Message 2} & C \to A: & \{J_C\}_{K_A}
\end{array}
$$

In this sequence, A sends a half key to B, who then sends the same half key to C. By whatever means, B makes C believe that Message 1' comes from A. Therefore, C replies to A with another half key. As a result of this exchange, both A and C compute a shared key; C believes that it is shared with A, while A believes that it is shared with B.

Whether this scenario constitutes an attack depends on the use of the protocol. It constitutes an attack in applications where it can result in harm to the principals A and C, which follow the protocol.

- In some situations, this scenario may cause some confusion, but no clear harm. As a result of B's behavior, A may attribute C's requests to B, and lend them the weight of B's authority. Moreover, A may inadvertently reveal some of B's secrets to C. Thus, B's behavior may harm B, but would not directly harm A or C.

  Note that B does not obtain the key that A and C compute. Therefore, B cannot learn C's secrets by decrypting the subsequent messages from C to A. (In this respect, this scenario is quite different from Lowe's attack on the Needham-Schroeder public-key protocol [Low96], although its structure is reminiscent of that attack.)

- More concretely, A may be a file server, and B and C two of its clients. In this case, the confusion may result in some immediate damage to B's files as a result of C's requests. Moreover, C may be able to read some of B's files.

  Indirectly, however, there can also be some harm to C if C's requests are not sufficiently explicit. Because of the confusion, A may write some of C's data into B's files. In a later session B can read these files, obtaining C's data.

- Finally, if A is running a contest, then it may think that a winning entry that arrives under K came from B when in fact it came from C.

In short, SHARE is adequate for some applications that require responsibility but not necessarily for applications that give credit.

With some additional precautions, SHARE can be used in applications that give credit. For example, each message that an application encrypts under the shared key could include (bound in the encryption) the name of its source, who expects credit for this message if any credit is to be had at all. The inclusion of this name is a prudent measure in any case; it agrees with Principle 3 of [AN96]:

> If the identity of a principal is essential to the meaning of a message, it is prudent to mention the principal's name explicitly in the message.

Thus, the issue of credit can be separated from the protocol for agreement on a key, and shifted to the use of the key.

Alternatively, we can easily strengthen SHARE, preventing the scenario described above. One possible modification is to let K be the result of applying a one-way hash function to the concatenation of J_A, J_B, and the names A and B, as in section 2.2. Another one is to add messages where the two parties A and B prove knowledge of K, as in SKEME.

Krawczyk does not view SHARE as a complete protocol, but only as a phase of SKEME:

> To be really meaningful this phase [SHARE] needs to be combined with the other phases of the protocol [SKEME].

One may instead argue that SHARE is meaningful on its own, although precautions are needed for its use, and elucidating its meaning requires separating credit from responsibility.

2.4 The Station-to-Station Protocol

As a last example, we consider the Station-to-Station protocol [DvOW92]. This protocol is fairly well known, so we do not reproduce it or discuss it in detail.

Basically, the Station-to-Station protocol consists of a signed Diffie-Hellman exchange [DH76], where both parties sign the Diffie-Hellman shares, and where they confirm possession of the resulting session key by using it to encrypt the signed Diffie-Hellman shares.

Without the key confirmation, the protocol suffices for establishing responsibility: each party may hold the other responsible for messages under the session key. With the key confirmation, the protocol is also suitable for establishing credit.

Although the Station-to-Station protocol may be rather attractive, it is not without alternatives. Other techniques for key confirmation could replace the encryption of the signed Diffie-Hellman shares. Furthermore, the key confirmation could be postponed or avoided altogether if, whenever credit is wanted, each message encrypted under the session key would contain the name of its source.

3 Discussion

While the examples of the previous section may be somewhat confusing, they are fairly typical of the authentication field. As this section discusses, there does not seem to be a general agreement on the goals of authentication, or a systematic analysis of the guarantees that authentication mechanisms provide to higher-level applications.

3.1 Design

In the literature on protocol design, there seems to be a consensus that an authentication protocol should at least establish responsibility, probably because responsibility is crucial for access control. In the context of access control, the primary property of a secure channel from a principal is that the channel "speaks for" the principal [LABW92, ABLP93]. In essence, a channel C speaks for a principal A if A's authority backs every message on C. This semantics justifies the following axioms:

- if A says that C speaks for A, then C does speak for A;
- if C speaks for A and C says s, then A says s.

These axioms are useful and sensible when "A says s" entails that A takes responsibility for s; but they would be unreasonable if "A says s" was meant to imply that A deserves credit for s. In access control, responsibility is largely separate from credit.

There does not seem to be a consensus that an authentication protocol should also establish credit. Although the literature does not seem to contain an explicit, articulate debate about this issue, we can propose likely arguments for both sides.

- We may argue that, once an authentication protocol has set up a channel that speaks for a principal, it is easy to use the channel for establishing credit whenever the need arises. (For example, the principal may send its name on the channel; see sections 2.1 and 2.3.) So it is not essential that an authentication protocol establish credit.

- On the other hand, we may say that stronger guarantees are better than weaker guarantees, particularly when protocols may be applied carelessly or in ways not fully anticipated by their designers. Establishing credit is a matter of prudence.

The latter argument may be prevailing. Popular Internet protocols (such as the SSL protocol [FKK96]) probably establish credit, although one may conjecture that applications rarely take advantage of this feature.

3.2 Analysis

Formal analyses seldom highlight the distinction between responsibility and credit, and the effect of a proof of possession of a secret key. There are some exceptions, for example in the work of van Oorschot and Syverson [vO93, SvO94]. Elsewhere, the distinction between responsibility and credit seems to be made through decisions in the modelling of protocol participants. (See, for example, work based on process algebras [Sch96, Low96, AG97].)

- Honest protocol participants are expected to follow the rules of the protocol faithfully, and not to try to obtain credit for messages that they did not generate themselves. A proof about honest protocol participants may show that a protocol establishes responsibility, but not credit.

- When an attacker is included as protocol participant, the attacker is not forced to follow the rules of the protocol, and may attempt to get undue credit. A proof that concerns such an attacker can show that a protocol establishes credit.

The protocols described in this paper should make good examples for future work on protocol analysis. While these protocols do not present any challenges of scale, they exhibit common subtleties.

There seem to be indications that responsibility and credit are dual notions. For example, through simple protocols, a principal may take responsibility for the statements of another principal, or may defer credit for its own statements to another principal. Moreover, responsibility sometimes comes with signatures, while credit sometimes comes with encryptions. A crisper understanding of this duality might lead to more regular protocol designs and to more systematic arguments about their correctness.

Acknowledgements

Discussions with Mike Burrows, Hugo Krawczyk, and Butler Lampson led to this paper. I believe that the arguments "for credit" and "against credit" of section 3.1 approximately reflect the views of Krawczyk and Lampson, respectively. (So they deserve some credit for these arguments but are not responsible for them.) Also helpful were discussions with Ross Anderson, Dieter Gollman, Gavin Lowe, Ron Rivest, Mike Roe, and Adi Shamir. Comments by anonymous referees suggested the mention of non-repudiation and the mention of third-party judgement.

References

[ABLP93] Martín Abadi, Michael Burrows, Butler Lampson, and Gordon Plotkin. A calculus for access control in distributed systems. *ACM Transactions on Programming Languages and Systems*, 15(4):706–734, October 1993.

[AG97] Martín Abadi and Andrew D. Gordon. A calculus for cryptographic protocols: The spi calculus. In *Proceedings of the Fourth ACM Conference on Computer and Communications Security*, pages 36–47, 1997.

[AN96] Martín Abadi and Roger Needham. Prudent engineering practice for cryptographic protocols. *IEEE Transactions on Software Engineering*, 22(1):6–15, January 1996.

[DES77] Data encryption standard. Fed. Inform. Processing Standards Pub. 46, National Bureau of Standards, Washington DC, January 1977.

[DH76] W. Diffie and M. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, IT-22(6):644–654, November 1976.

[DvOW92] Whitfield Diffie, Paul C. van Oorschot, and Michael J. Wiener. Authentication and authenticated key exchanges. *Designs, Codes and Cryptography*, 2:107–125, 1992.

[Ell97] Carl M. Ellison. SPKI certificate documentation. Web pages at `http://www.clark.net/pub/cme/html/spki.html`, 1997.

[FKK96] Alan O. Freier, Philip Karlton, and Paul C. Kocher. The SSL protocol: Version 3.0. Available at `http://home.netscape.com/newsref/std/SSL.html`, March 1996.

[Gol96] Dieter Gollman. What do we mean by entity authentication? In *Proceedings of the 1996 IEEE Symposium on Security and Privacy*, pages 46–54, May 1996.

[Kra96] Hugo Krawczyk. SKEME: A versatile secure key exchange mechanism for internet. In *Proceedings of the Internet Society Symposium on Network and Distributed Systems Security*, February 1996. Available at `http://bilbo.isu.edu/sndss/sndss96.html`.

[Kra97] Hugo Krawczyk. Private communication. 1997.

[LABW92] Butler Lampson, Martín Abadi, Michael Burrows, and Edward Wobber. Authentication in distributed systems: Theory and practice. *ACM Transactions on Computer Systems*, 10(4):265–310, November 1992.

[Low96] Gavin Lowe. Breaking and fixing the Needham-Schroeder public-key protocol using FDR. In *Tools and Algorithms for the Construction and Analysis of Systems*, volume 1055 of *Lecture Notes in Computer Science*, pages 147–166. Springer Verlag, 1996.

[Low97] Gavin Lowe. A hierarchy of authentication specifications. In *Proceedings of the 10th IEEE Computer Security Foundations Workshop*, pages 31–43, 1997.

[MvOV96] Alfred J. Menezes, Paul C. van Oorschot, and Scott A. Vanstone. *Handbook of Applied Cryptography.* CRC Press, 1996.

[Roe] Michael Roe. *Cryptography and Evidence.* PhD thesis, University of Cambridge Computer Laboratory.

[RSA78] R. L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, February 1978.

[Sch96] Steve Schneider. Security properties and CSP. In *IEEE Symposium on Security and Privacy*, pages 174–187, 1996.

[SvO94] Paul F. Syverson and Paul C. van Oorschot. On unifying some cryptographic protocol logics. In *IEEE Computer Society Symposium on Research in Security and Privacy*, pages 14–28, 1994.

[vO93] Paul C. van Oorschot. Extending cryptographic logics of belief to key agreement protocols. In *Proceedings of the first ACM Conference on Computer and Communications Security*, pages 232–243, 1993.

[ZG96] Jianying Zhou and Dieter Gollman. A fair non-repudiation protocol. In *Proceedings of the 1996 IEEE Symposium on Security and Privacy*, pages 55–61, May 1996.