Alice Who?

Authentication Protocols

Andreas Zeller/Stephan Neuhaus
Lehrstuhl für Softwaretechnik
Universität des Saarlandes, Saarbrücken

SS 2004/CVS Revision: 1.6

The Menu

> Oh, I don't know why she's leaving,
> Or where she's gonna go
> I guess she's got her reasons
> But I just don't wanna know
> 'Cos for 24 years
> I've been livin' next door to Alice —Bob

- Simple Authentication Protocols
- Common Pitfalls
- Ways to Analyze Protocols
- Login-only protocols
- Mutual authentication with Key Distribution Center
- Needham-Schroeder

Basics (1)

Authentication happens between two or more parties and is the process of convincing another party that one party has indeed the identity it claims to have.

Meet Alice and Bob:

Alice and Bob want to communicate, but can't really be sure that the other is really who he/she says he/she is. So they exchange a series of messages ⇒ a protocol.

Basics (2)

- May be one-sided: Alice may be a computer and Bob may be a user. Bob logs in to Alice; Alice then knows it's Bob, but Bob doesn't (in general) know it's Alice.
- May be mutual: Bob logs in to Alice so that both of them are convinced of the other's identity afterwards.
- May use trusted third parties, online (Bob asks the trusted party—Trent—to establish a conversation with Alice) or offline (Alice could present a certificate signed by Trent).
- There might be an eavesdropper—Eve—that can listen to and/or modify messages as they are exchanged between Alice and Bob.
- There might be an intruder—Trudy—that can listen to and inject messages.

Basics (3): Protocol Notation

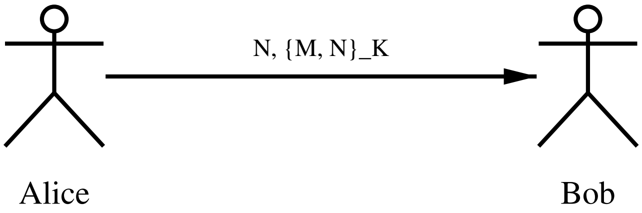
$$\text{Alice} \longrightarrow \text{Bob} : N, \{M, N\}_K$$

This notation means that the principal Alice transmits to the principal Bob a message containing a nonce N, and the plaintext M concatenated with N, encrypted under the key K.

A *nonce* is anything that guarantees the freshness of a message, such as a random number, a serial number, or a challenge received from a third party.

We'll usually distinguish between a principal "Bob" and the identifying information that he sends over the wire, "Bob".

Basics (4)



We won't use this often, because it's often easier to see what happens when using the formula notation, especially when there are more than two parties involved.

The Simplest Protocol

The simplest authentication protocol has no name.

$$\text{Alice} \longrightarrow \text{Bob} \quad : \quad \text{“Hi, I'm } \textit{Alice}\text{.”}$$

It can be extended into a mutual protocol:

$$\begin{aligned} \text{Alice} \longrightarrow \text{Bob} \quad &: \quad \text{“Hi, I'm } \textit{Alice}\text{.”} \\ \text{Bob} \longrightarrow \text{Alice} \quad &: \quad \text{“Hi, I'm } \textit{Bob}\text{.”} \end{aligned}$$

The problem is of course that Eve can successfully pretend to be Alice:

$$\text{Eve} \longrightarrow \text{Bob} \quad : \quad \text{“Hi, I'm } \textit{Alice}\text{.”}$$

Usage of this Protocol

This protocol is actually in widespread use:

- TCP connections are generally not authenticated. This is a problem with mitigating factors, because if you spoof the sender address, you usually won't get the return packets; also, if you are on the same Ethernet, you have to do something about the other party's ARP daemon. But it's possible.
- Telephone calls are usually not (properly) authenticated; otherwise Kevin Mitnlick couldn't have been as successful as he was. (Remember the very first lecture in this course?)

Threats Against Authentication Protocols

The basic threat is always that it is possible for Trudy or Eve eventually to impersonate Alice or Bob. They can accomplish this for example by:

- *Replaying* all or part of a previously recorded conversation;
- *Eavesdropping* on a conversation and learning secrets;
- *Modifying messages* en route to their destination;
- Modifying the *message flow* by inserting or deleting messages in the network.
- *Assuming another's identity* (e.g., using the other's network address).
- *Stealing another's databases*, to steal keys.

As you can see, we'll encounter pretty powerful adversaries.

But we'll not defend against all threats. For example, we'll usually not defend against deleted messages (for the practical reason that there's not much that we can do about it).

Improvements

How can this protocol be improved?

- Alice and Bob could *share a secret*. Alice could present that secret to show that she really is Alice. (Who you are is what you know.)
- Variation: Alice claims that she knows a secret that is unique to her. Instead of presenting the secret, Alice could prove that she knows the secret without divulging it (*zero-knowledge-proof*).
- Alice could be in the posession of a unique token that she presents to Bob. (Who you are is what you have.)
- Alice could agree on submitting to a biometric scan, e.g., a fingerprint scan or face scan. (Who you are is what you are.)

...What You Know (aka Passwords)

The protocol goes like this: Bob maintains a database of secret passwords. Alice then authenticates herself to Bob like this:

$$\text{Alice} \longrightarrow \text{Bob} : \text{“Hi, I'm } \textit{Alice}\text{, and my password is ‘}\textit{x\&8e;pqA}\text{’.”}$$

Eve can break this protocol if we assume that she can listen to the conversation between Alice and Bob. She simply captures the password and replays it:

$$\text{Eve} \longrightarrow \text{Bob} : \text{“Hi, I'm } \textit{Alice}\text{, and my password is ‘}\textit{x\&8e;pqA}\text{’.”}$$

Note that this is independent of the guessablity of the password.

This attack is not always feasible, but it's feasible enough in so many environments that you *must* abstain from using this protocol.

Encrypting the Exchange

Assume Alice and Bob share a secret K that can be used as a cryptographic key.

$$\text{Alice} \longrightarrow \text{Bob} : \{\text{“Hi, I'm } \textit{Alice}\text{, and my password is ‘}\textit{x\&8e;pqA}\text{’.”}\}_K$$

That's much better. An eavesdropper couldn't decrypt the message and therefore wouldn't be able to recover the password.

But is this really necessary?

No, because Eve can still just capture the entire encrypted message and replay it to Bob.

Challenge-Response

$$\begin{array}{lcl} \text{Alice} \longrightarrow \text{Bob} & : & \text{“Hi, I'm } \textit{Alice}\text{.”} \\ \text{Bob} \longrightarrow \text{Alice} & : & \text{“Hi } \textit{Alice}\text{, please encrypt } \texttt{0x67f810a762df5e}\text{.”} \\ \text{Alice} \longrightarrow \text{Bob} & : & \{\texttt{0x67f810a762df5e}\}_K \end{array}$$

Or, more formally,

$$\begin{array}{lcl} \text{Alice} \longrightarrow \text{Bob} & : & \textit{Alice} \\ \text{Bob} \longrightarrow \text{Alice} & : & R \\ \text{Alice} \longrightarrow \text{Bob} & : & \{R\}_K, \end{array}$$

where R is a random challenge.

Problems with C-R

- It's one-sided: Bob knows about Alice, but not vice versa.
- Somehow Bob needs to maintain a database of secrets *and keep it secure*. In practice, that's bloody difficult.
- Trudy could hijack the connection after the initial exchange.
- If K is derived from a password (that only Alice needs to know), then Eve could mount an offline password-guessing attack.

Variation 1

$$\begin{aligned} \text{Alice} \longrightarrow \text{Bob} &\quad : \quad \mathit{Alice} \\ \text{Bob} \longrightarrow \text{Alice} &\quad : \quad \{R\}_K \\ \text{Alice} \longrightarrow \text{Bob} &\quad : \quad R, \end{aligned}$$

where R is a random challenge.

- Requires reversible cryptography.
- If K is derived from password, and if R is distinguishable from random bits, Eve can mount a password-guessing attack without snooping, by initiating the protocol as *Alice*.
- Authentication is mutual *if* R is a recognizable quantity with a limited lifetime.

Variation 2

$$\text{Alice} \longrightarrow \text{Bob} \quad : \quad \mathit{Alice}, \{t\}_K,$$

where t is a timestamp.

- One-sided (Bob authenticates Alice, not vice versa).
- Requires clocks to be reasonably synchronized.
- When using the same secret K for multiple servers, Eve can impersonate Alice at the other servers (if she's fast enough).
- Replay possible if Eve can cause Bob's clock to be turned back.
- Time setting and login are now coupled.

Mutual Authentication

$$\begin{array}{rcl} \text{Alice} \longrightarrow \text{Bob} & : & \mathit{Alice} \\ \text{Bob} \longrightarrow \text{Alice} & : & R_1 \\ \text{Alice} \longrightarrow \text{Bob} & : & \{R_1\}_K, R2 \\ \text{Bob} \longrightarrow \text{Alice} & : & \{R_2\}_K \end{array}$$

Mutual Authentication "Optimized"

We attempt to optimize this protocol:

$$\begin{array}{rcl} \text{Alice} \longrightarrow \text{Bob} & : & \mathit{Alice}, R_2 \\ \text{Bob} \longrightarrow \text{Alice} & : & \{R_2\}_K, R_1 \\ \text{Alice} \longrightarrow \text{Bob} & : & \{R_1\}_K \end{array}$$

We eliminated 25% of all messages. Not bad!

What's wrong with this protocol?

Reflection Attack

This protocol suffers from a *reflection attack*:

$$\begin{array}{rcl} \text{Trudy} \longrightarrow \text{Bob} & : & \mathit{Alice}, R_2 \\ \text{Bob} \longrightarrow \text{Trudy} & : & \{R_2\}_K, R_1 \\ \text{Trudy} \longrightarrow \text{Bob} & : & \mathit{Alice}, R_1 \\ \text{Bob} \longrightarrow \text{Trudy} & : & \{R_1\}_K, R_3 \\ \text{Trudy} \longrightarrow \text{Bob} & : & \{R_1\}_K \end{array}$$

Rules

- Don't use the same key K for Alice and Bob. Instead, use $K + 1$, $K \oplus \mathrm{0x0F0F0F0F}$, $\neg K$, or something like this.
- Different challenges. Either remember past challenges and decline to encrypt known challenges, or insist that the challenges must be different for Alice and Bob (see exercises).
- Let the initiator of a protocol be the first to prove his identity.

Authentication With Public Key

$$\begin{aligned}
\text{Alice} \longrightarrow \text{Bob} &: \ \mathit{Alice} \\
\text{Bob} \longrightarrow \text{Alice} &: \ R \\
\text{Alice} \longrightarrow \text{Bob} &: \ [R]_{\text{Alice}}
\end{aligned}$$

- Bob's database doesn't contain secrets anymore ⇒ need not be protected against theft.
- Database must still be protected against *modification* (*much* easier).

Variation and Criticism (1)

$$\begin{aligned}
\text{Alice} \longrightarrow \text{Bob} &: \ \mathit{Alice} \\
\text{Bob} \longrightarrow \text{Alice} &: \ \{R\}_{\text{Alice}} \\
\text{Alice} \longrightarrow \text{Bob} &: \ R
\end{aligned}$$

- Needs encryption in addition to signature.
- Both protocols have the flaw that if Eve can impersonate Bob, she can get arbitrary values signed (or encrypted).
- This is a *serious* flaw if the Alice's key pair is used for things other than authentication (e.g., for signing bank transfers).

Criticism (2)

This problem can be solved if we stipulate that

- keys are never reused for different applications; or
- the system is coordinated that it's not possible to use one protocol to break another (for example by formatting the R values differently for different applications).

Also note what this means:

By combining two protocols that are secure in themselves, you get a system that is not secure at all; and you can design protocols whose deployment threatens the security of a system that is already in place!

For people who like to sound clever, we can also say that security isn't closed under composition.

Mutual Authentication With Public Key

$$\begin{array}{lcl} \text{Alice} \longrightarrow \text{Bob} & : & \mathit{Alice}, \{R_2\}_{\text{Bob}} \\ \text{Bob} \longrightarrow \text{Alice} & : & R_2, \{R_1\}_{\text{Alice}} \\ \text{Alice} \longrightarrow \text{Bob} & : & R_1 \end{array}$$

In an obvious variation, Alice could send R_2 and Bob could return $[R_2]_{\text{Bob}}$; Bob would then send R_1 and Alice would return $[R_1]_{\text{Alice}}$.

Here the obvious problem is, how do Alice and Bob obtain the other's public key?

- With a Key Distribution Center (KDC);
- With Public Key Infrastructure (PKI)

How Does Alice Obtain Her Private Key?

Assume Alice is sitting at her workstation. Can we really make her type in a 512-bit RSA private key?

- She can carry her key with her on a USB stick or other portable device.
- She can obtain an encrypted version of her key from a KDC (or even from Bob) and decrypt it using a password.

At the same place, one can store information that would enable Alice to learn Bob's public key:

- Encrypted with a key derived from Alice's password;
- Signed with Alice's private key.

Mediated Authentication

Mediated authentication happend when Alice first asks a trusted intermediary, Trent, to introduce her to Bob.

Because Trent is trusted by both Alice and Bob, authentication is mutual.

Does *not* need public key cryptography!

$$\begin{aligned}
\text{Alice} \longrightarrow \text{Trent} &: \textit{Alice} \text{ wants } \textit{Bob} \\
\text{Trent} &: \text{Invents } K_{AB} \\
\text{Trent} \longrightarrow \text{Alice} &: \{\text{Use } K_{AB} \text{ for Bob}\}_{\text{Alice}} \\
\text{Trent} \longrightarrow \text{Bob} &: \{\text{Use } K_{AB} \text{ for Alice}\}_{\text{Bob}}
\end{aligned}$$

After this exchange, Alice and Bob can (must) authenticate themselves.

Mediated Authentication in Practice

In practice, it's impractical to use the protocol like this:

- Alice's first message to Bob (encrypted with K_{AB}) might arrive at Bob before Trent's message that contains K_{AB}.
- It's impractical for Trent to open a connection to Bob.

Therefore, Trent will in general return to Alice not only $\{\text{Use } K_{AB} \text{ for Bob}\}_{\text{Alice}}$, but also $t = \{\text{Use } K_{AB} \text{ for Alice}\}_{\text{Bob}}$, which is called a *ticket*.

Alice will then present t when she initiates a connection to Bob.

Both will then have to complete a mutual authentication.

Needham-Schroeder (1)

- It's a classic mediated authentication protocol with mutual authentication.
- It's been a model for many other protocols.
- It's used in Kerberos and Kerberos is used in Active Directory $\Longrightarrow$ huge installed base.
- We'll analyze this protocol in some detail in order to understand its strengths and weaknesses.

Needham-Schroeder (2)

$$
\begin{array}{rcl}
\text{Alice} \longrightarrow \text{Trent} & : & N_1, \mathit{Alice} \text{ wants } \mathit{Bob} \\
\text{Trent} & : & \text{Invents } K_{AB} \\
\text{Trent} \longrightarrow \text{Alice} & : & \{N_1, \mathit{Bob}, K_{AB}, \{K_{AB}, \mathit{Alice}\}_{\text{Bob}}\}_{\text{Alice}} \\
\text{Alice} & : & \text{Verifies } N_1 \text{, extracts } K_{AB} \text{ and ticket} \\
\text{Alice} \longrightarrow \text{Bob} & : & \{K_{AB}, \mathit{Alice}\}_{\text{Bob}}, \{N_2\}_{AB} \\
\text{Bob} & : & \text{Extracts } K_{AB} \text{ from ticket} \\
\text{Bob} \longrightarrow \text{Alice} & : & \{N_2 - 1, N_3\}_{AB} \\
\text{Alice} \longrightarrow \text{Bob} & : & \{N_3 - 1\}_{AB}
\end{array}
$$

where $\{K_{AB}, \mathit{Alice}\}_{\text{Bob}}$ is Trent's ticket for Alice's conversation with Bob and the N_i are *nonces*, i.e., quantities used only once.

Analysis of Needham-Schroeder (1)

Why the Nonce in the first message?

Otherwise, the protocol could be susceptibe to a replay attack. Assume that Eve has captured a previous exchange of this modified Needham-Schroeder protocol and has, by some effort, broken K_{AB}:

$$
\begin{array}{rcl}
\text{Alice} \longrightarrow \text{Eve} & : & \mathit{Alice} \text{ wants } \mathit{Bob} \\
\text{Eve} \longrightarrow \text{Alice} & : & \{\mathit{Bob}, K_{AB}, \{K_{AB}, \mathit{Alice}\}_{\text{Bob}}\}_{\text{Alice}}
\end{array}
$$

and Eve will now be able to decrypt the conversation between Alice and Bob. This can't happen with N_1 used in the first step, because Eve can't encrypt N_1.

Analysis of Needham-Schroeder (2)

Why is *Bob* in the message from the KDC to Alice?

To make it impossible for Trudy to substitute her own name for Bob's:

$$\begin{array}{rcl}
\text{Alice} \longrightarrow \text{Trudy} & : & \textit{Alice} \text{ wants } \textit{Bob} \\
\text{Trudy} & : & \text{Intercepts and changes the message} \\
\text{Trudy} \longrightarrow \text{Trent} & : & \textit{Alice} \text{ wants } \textit{Trudy} \\
\text{Trent} \longrightarrow \text{Trudy} & : & \{K_{AB}, \{K_{AB}\}_{\text{Trudy}}\}_{\text{Alice}} \\
\text{Trudy} \longrightarrow \text{Alice} & : & \{K_{AB}, \{K_{AB}\}_{\text{Trudy}}\}_{\text{Alice}} \\
\text{Trudy} & : & \text{Impersonates Bob}
\end{array}$$

Nonces

As we have said, a *nonce* is a number used only once.

It is possible to introduce weaknesses into protocols if the nonces have the wrong properties.

Nonce types are:

- a timestamp;
- a sequence number; and
- a large random number.

Large Random Numbers as Nonces (1)

Why can we use a random number as a nonce when there is a chance that it would be reused?

Back-of-envelope-calculation: Assume n-bit random numbers; there are $N = 2^n$ of them. The probability that k independent draws out of N numbers yield all different numbers is $N(N-1)\cdots(n-k+1)/N^k$.

The relative difference between N and $N-k+1$ is $\delta = (k-1)/N$. (I.e., $N-k+1 = (1-\delta)N$.) Let's assume we generate a 128-bit nonce every millisecond for 1000 years. That will be $1000 \cdot 366 \cdot 24 \cdot 3600 \cdot 1000 = 31622400000000$ or about 2^{45} nonces. With $N = 2^{128}$ and $k = 2^{45}$, we have $\delta \approx 2^{45}/2^{128} = 2^{-83}$.

Large Random Numbers as Nonces (2)

$N - k + 1 \approx (1 - 2^{-83})N$; therefore

$$\begin{aligned} N(N-1)\cdots(N-k+1)/N^k &\geq (N-k+1)^k/N^k \\ &\approx (1-2^{-83})^k N^k/N^k \\ &\approx (1-2^{-83})^k \\ &\approx 1 - k\cdot 2^{-83} \\ &\approx 1 - 2^{45}\cdot 2^{-83} \\ &= 1 - 2^{-38}. \end{aligned}$$

Therefore, it is practically certain that all nonces are different.
($2^{-38} \approx 3.6 \cdot 10^{-12}$.)

Timestamps and Sequence Numbers

- Timestamps require synchronized clocks.
- A sequence number requires that at least one party remembers the last sequence number it has handed out.

$$\begin{aligned} \text{Alice} \longrightarrow \text{Bob} &: \textit{Alice} \\ \text{Bob} \longrightarrow \text{Alice} &: \{R\}_{AB} \\ \text{Alice} \longrightarrow \text{Bob} &: R \end{aligned}$$

Breaking The Protocol

If Bob used sequence numbers, Eve could listen in to only one exchange between Alice and Bob. Then she would know the current value of R and could impersonate Alice:

$$\begin{aligned} \text{Eve} \longrightarrow \text{Bob} &: \textit{Alice} \\ \text{Bob} \longrightarrow \text{Eve} &: \{R+1\}_{AB} \\ \text{Eve} \longrightarrow \text{Bob} &: R+1 \end{aligned}$$

Eve can answer “$R + 1$” in step 3, even though she can’t decrypt $\{R + 1\}_{AB}$, because she can *predict* what the challenge will be.

Random Numbers

If you use random numbers for nonces, be sure to pick good ones. We've had two lectures on how to do that, so we won't talk about that any further.

Performance

In order to evaluate a protocol's performance, the following factors must be checked:

- Number of signatures
- Number of public-key encryptions
- Number of bytes encrypted with a secret key
- Number of bytes to be hashed
- Number and size of messages transmitted
- Number of connection buildups and teardowns

Checklist

A checklist can be found in Charlie Kaufman, Radia Perlman, Mike Speciner, *Network Security*, Prentice-Hall. (The second edition has the list on p. 285f.)

Summary

- Simple Authentication Protocols
- Common Pitfalls
- Ways to Analyze Protocols
- Login-only protocols
- Mutual authentication with Key Distribution Center
- Needham-Schroeder

Resources

- Ross Anderson, *Security Engineering*, John Wiley & Sons
- Bruce Schneier, *Applied Cryptography*, John Wiley & Sons
- Charlie Kaufman, Radia Perlman, Mike Speciner, *Network Security*, Prentice-Hall