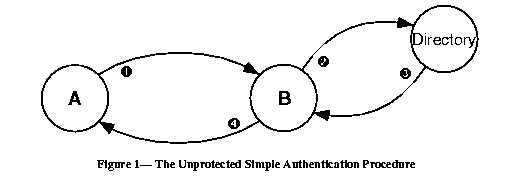
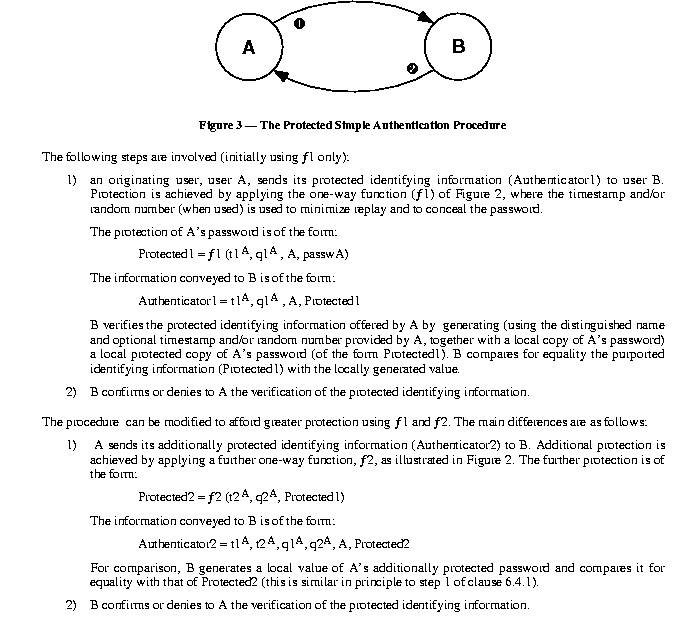
# Authentication Procedures

X.509 also includes three alternative authentication procedures that are intended for use across a variety of applications. All these procedures make use of public-key signatures. It is assumed that the two parties know each other’s public key, either by obtaining each other’s certificates from the directory or because the certificate is included in the initial message from each side. These procedures are treated as strong contrary to simple authentication procedures in which a client is authenticated to a server by sending him its identifier and password in clear or hashed together with a timestamp and nonce:





# Authentication Procedures (Cont 1)



# One-Way Authentication

One way authentication involves a single transfer of information from one user (A) to another (B), and establishes the following:

1. The identity of A and that the message was generated by A
2. That the message was intended for B
3. The integrity and originality (it has not been sent multiple times) of the message

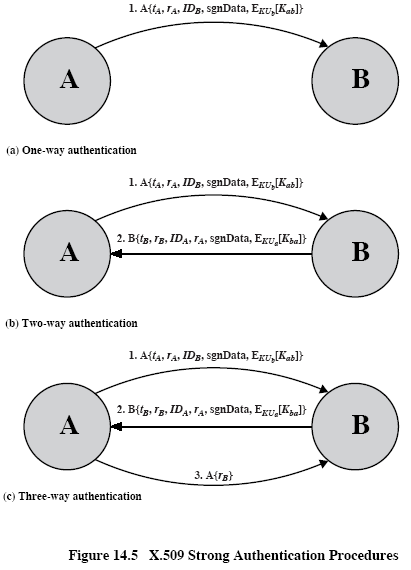
Note that only the identity of the initiating entity is verified in this process, not that of the responding entity.

At minimum, the message includes a timestamp, , a nonce , and the identity of B and is signed with A’s private key. The timestamp consists of an optional generation time and an expiration time. This prevents delayed delivery of messages. The nonce can be used to detect replay attacks. The nonce value must be unique within the expiration time of the message. Thus, B can store the nonce

# One-Way Authentication (Cont 1)

until it expires and reject any new messages with the same nonce.

For pure authentication, the message is used simply to present credentials to B. The message may also include information to be conveyed. This information, sgnData, is included within the scope of the signature, guaranteeing its authenticity and integrity. The message may also be used to convey a session key to B, encrypted with B’s public key.



# Two-Way Authentication

In addition to the three elements just listed, two-way authentication establishes the following elements:

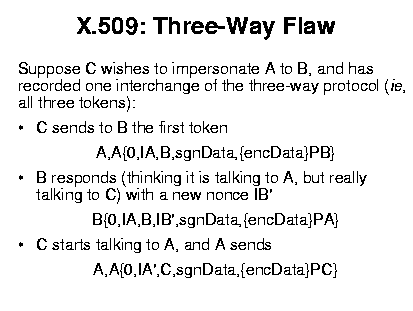
1. The identity of B and that the reply message was generated by B
2. That the message was intended for A
3. The integrity and originality of the reply

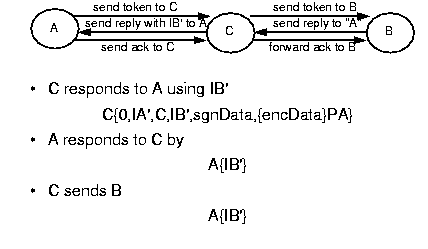
The reply message includes the nonce from A, to validate the reply. It also includes a timestamp and nonce generated by B. As before, the message may include signed additional information and a session key encrypted with A’s public key.

# Three-Way Authentication

Here, a final message from A to B is included, which contains a signed copy of the nonce . The intent of this design is that timestamps need not be checked: Because both nonces are echoed back by the other side, each side can check the returned nonce to detect replay attack. This approach is needed when synchronized clocks are not available. It is not shown in Figure 14.5, but the response from A also includes B to counter meet-in-the middle attack when opponent C intercepts messages between A and B

( <http://nob.cs.ucdavis.edu/classes/ecs153-1997-01/Postscript/kerbiso.ps> ).





Timestamps in the protocol above are zeros because in the three-way authentication clocks are not used.

# One-Time Password (OTP)

We follow [**L. Lamport, Password authentication with insecure communication. – Communications of ACM, 1981, v. 24, No 11, 770-772,** [**http://cmpe.emu.edu.tr/chefranov/cmpe552-06/Lecture%20Notes/Lamport81.pdf**](http://cmpe.emu.edu.tr/chefranov/cmpe552-06/Lecture%20Notes/Lamport81.pdf)].

Passwords can be compromised when they cross a network or from a server’s database. In previous considerations, we assumed that there is a trusted third party and/or password database on the server side is safe. But this may not be true, and OTP schemas give solution for such cases.

Solution is strongly based on one-way hash functions.

Lamport’s schema allows having some finite number,, of authentications of a user to a server before the initialization procedure will be required. In initialization procedure, the user and server exchange securely secret information (by means of some special channel, personally, by ordered mail, courier, or in some other secure way). Schema assumes that a password never crosses insecure network, and the server’s password database might be compromised, but can’t be changed by an intruder. The server and user use one and the same hash function, , in the authentication procedure. The server authenticates the user applying the hash function to a current “password” value in its database. Actually, this value is derived from the passport, and is used as a current password, which changes from one authentication to another. That’s why the schema is called “one-time password”. Current password depends on the authentication number and can’t exceed . OTP schemas represent “challenge-response” schemas.

# Initialization Procedure

The client selects a password, , a number, , calculates

,

where

.

The client securely delivers to the server (, and the servers saves it into () tuple.

# Authentication Procedure

When the client, C, requests authentication by the server, S, the following proceeds:

1. C -> S: C\_ID //client sends his ID

2. S -> C: Counter(C\_ID) //server responds by respective Counter value

3. C -> S: C\_ID, 

4. S: If  then {

S authenticates C, and sets ()=()

}

Else C is not authenticated

After authentications,  will become equal to one, and on the next authentication, C should pass his password in clear. Hence, maximal number of authentications without sending of the secret client’s password is . The trick with the schema is that due to one-wayness of the hash function, it is not feasible to find a value such that its hash will be equal to the stored by S value. As far as this value changes from one authentication to another, knowledge of the current password does not allow using it because rate of cracking may be less than that of password update. The schema has a problem that if is large (to avoid often initialization) then C should perform on average large amount of computations (Step 3 of authentication procedure)