Protocol: BELLARE-ROGAWAY SKDS (1995)

1. Alice chooses a random number, r_A, and she sends $ID(Alice)$ and r_A to Bob.

2. Bob chooses a random number, r_B, and he sends $ID(Alice)$, $ID(Bob)$, r_A and r_B to the TA.

3. The TA chooses a random session key K. Then it computes

$$y_B = (e_{K_{Bob}}(K), MAC_{Bob}(ID(Alice) \parallel ID(Bob) \parallel r_B \parallel e_{K_{Bob}}(K))$$

and

$$y_A = (e_{K_{Alice}}(K), MAC_{Alice}(ID(Bob) \parallel ID(Alice) \parallel r_A \parallel e_{K_{Alice}}(K))$$

The TA sends y_B to Bob and y_A to Alice.

4. Alice decrypts y_A using her key K_{Alice}, obtaining K. Bob decrypts y_B using his key K_{Bob}, obtaining K.

Bellare-Rogaway SKDS (diagram)

$$TA \qquad\qquad B \qquad\qquad A$$

$$B \xleftarrow{\;A, r_A\;} A$$

$$TA \xleftarrow{\;A, B, r_A, r_B\;} B$$

$$TA \xrightarrow{\;y_A, y_B\;} B$$

$$B \xrightarrow{\;y_A\;} A$$

where

$$y_B = (e_{K_{Bob}}(K), MAC_{Bob}(ID(Alice) \parallel ID(Bob) \parallel r_B \parallel e_{K_{Bob}}(K))$$

$$y_A = (e_{K_{Alice}}(K), MAC_{Alice}(ID(Bob) \parallel ID(Alice) \parallel r_A \parallel e_{K_{Alice}}(K))$$

Security Properties of the Bellare-Rogaway Protocol

1. If Alice, Bob and the TA are honest and the adversary is passive, then Alice and Bob compute the same session key, K, and the adversary cannot compute any information about K.

2. If the (active) adversary changes messages originating from (or being sent to) an honest participant, then that participant will "reject" with high probability.

3. If Alice and the TA are honest, and the (active) adversary impersonates Bob, then then the adversary cannot compute the session key.

4. If Bob and the TA are honest, and the (active) adversary impersonates Alice, then then the adversary cannot compute the session key.

Analysis and Discussion

- properties **(1)**, **(3)** and **(4)** are guaranteed by using a secure encryption scheme
- proving property **(2)** requires analyzing the probability that the adversary can guess a correct new MAC or reuse a MAC from a previous session (similar to the analysis of the identification protocols considered earlier)
- the scheme is not secure against a known LL-key attack
- the *Bellare-Rogaway* protocol does “less” than *NS* or *Kerberos*, but its simplicity means that it can be analyzed rigourously

Consequences of the Security Properties

- if (honest) Alice accepts, then she has received a session key, K, and she believes Bob and the TA are the only parties that can determine K
- the adversary's goal is to cause an (honest) participant to accept in a situation where someone other than the (honest) intended partner of that participant knows the value of the session key K
- we argue that the adversary cannot achieve his goal, as follows:
  - if (honest) Alice accepts, then the adversary did not tamper with messages sent to or from Alice
  - hence, y_A was constructed by the TA, and K is encrypted only with the keys K_{Alice} and K_{Bob}
  - therefore, the adversary will not learn the value of K if Bob and the TA are honest

Protocol: Diffie-Hellman key agreement

1. U chooses a_U at random, where $0 \leq a_U \leq q-1$. Then U computes
$$b_U = \alpha^{a_U} \bmod p$$
and sends b_U to V.

2. V chooses a_V at random, where $0 \leq a_V \leq q-1$. Then V computes
$$b_V = \alpha^{a_V} \bmod p$$
and sends b_V to U.

3. U computes
$$K = (b_V)^{a_U} \bmod p$$
and V computes
$$K = (b_U)^{a_V} \bmod p.$$

Security wrt Passive and Active Attacks

- the key $K = \mathsf{CDH}(\alpha, b_U, b_V)$
- assuming that DDH is intractible, a passive adversary cannot compute any information about K
- the scheme is insecure against active attacks, however

Diffie-Hellman Key Agreement is supposed to look like this:

$$
\begin{array}{ccc}
U & & V \\
 & \xrightarrow{\qquad\qquad \alpha^{a_U} \qquad\qquad} & \\
 & \xleftarrow{\qquad\qquad \alpha^{a_V} \qquad\qquad} &
\end{array}
$$

Intruder-in-the-middle Attack

In an intruder-in-the-middle attack, W will intercept messages between U and V and substitute his own messages:

$$\begin{array}{ccccc} U & & W & & V \\ & \xrightarrow{\quad \alpha^{a_U} \quad} & & \xrightarrow{\quad \alpha^{a'_U} \quad} & \\ & \xleftarrow{\quad \alpha^{a'_V} \quad} & & \xleftarrow{\quad \alpha^{a_V} \quad} & \end{array}$$

At the end of the protocol, U has actually established the secret key $\alpha^{a_U a'_V}$ with W, and V has established a secret key $\alpha^{a'_U a_V}$ with W.

Designing Secure Authenticated Key Agreement Protocols

- start with a secure mutual identification protocol
- replace a random challenges by each participant, say r_U and r_V, by b_U and b_V respectively, where $b_U = \alpha^{a_U} \bmod p$ and $b_V = \alpha^{a_V} \bmod p$
- then each participant can compute the session key $K = \mathsf{CDH}(\alpha, b_U, b_V)$, but the adversary cannot do so

Protocol: Public-key Mutual Identification Protocol

1. U chooses a random challenge r_U and sends it to V.

2. V chooses a random challenge r_V. Then he computes

$$y_V = sig_V(r_V \parallel r_U).$$

and sends $(\mathbf{Cert}(V), r_V, y_V)$ to U.

3. U verifies y_V using ver_V (and she verifies $\mathbf{Cert}(V)$ using ver_{TA}). If the signature y_V is not valid, then she outputs "reject" and quits. Otherwise, she outputs "accept", computes

$$y_U = sig_U(r_U \parallel r_V),$$

and sends $(\mathbf{Cert}(U), y_U)$ to V.

4. V verifies y_U using ver_U (and he verifies $\mathbf{Cert}(U)$ using ver_{TA}). If the signature y_U is not valid, then U outputs "reject"; otherwise, he outputs "accept".

We now transform the previous protocol into a key agreement scheme:

Protocol: SIMPLIFIED STATION-TO-STATION PROTOCOL

1. U chooses a random number a_U, $0 \leq a_U \leq q-1$. Then she computes

$$b_U = \alpha^{a_U} \bmod p$$

and sends it to V.

2. V chooses a random number a_V, $0 \leq a_V \leq q-1$. Then he computes

$$\begin{aligned} b_V &= \alpha^{a_V} \bmod p \\ K &= (b_U)^{a_V} \bmod p, \quad \text{and} \\ y_V &= sig_V(\alpha^{a_V} \parallel \alpha^{a_U}). \end{aligned}$$

Then V sends $(\mathbf{Cert}(V), b_V, y_V)$ to U.

Protocol: Simplified station-to-station protocol (cont.)

3. U verifies y_V using ver_V (and she verifies **Cert**(V) using ver_{TA}). If the signature y_V is not valid, then she outputs "reject" and quits. Otherwise, she outputs "accept", computes

$$\begin{aligned} K &= (b_V)^{a_U} \bmod p, \quad \text{and} \\ y_U &= sig_U(\alpha^{a_U} \parallel \alpha^{a_V}), \end{aligned}$$

   and sends $(\mathbf{Cert}(U), y_U)$ to V.

4. V verifies y_U using ver_U (and he verifies **Cert**(U) using ver_{TA}). If the signature y_U is not valid, then U outputs "reject"; otherwise, he outputs "accept".