Exercise 9 in Advanced Methods of Cryptography - Proposed Solution -

Prof. Dr. Rudolf Mathar, Henning Maier, Markus Rothe

2014-01-09

Solution of Problem 28

The paper is easily found online, e.g.: *http://tnlandforms.us/cns06/lamport.pdf*

Remarks on reading this paper:

- Familiarize yourself with the paper structure
- Formulate elementary questions about the content and answer them
- Note that the formal notation might differ from our lecture notes
- Look up unknown expressions
- Check the references
- Feel free to discuss further implications (are there any errors or loopholes?)

Solution of Problem 29

a) In order to break Lamport's protocol we need to compute the $(A, i+1, w_{i+1})$ given (A, i, w_i) from the previous transmission i. Since the computation of A and $i+1$ is trivial, we only need to compute the following inverse hash function:

$$w_{i+1} = H^{t-i-1}(w) = H^{-1}(H^{t-i}(w)) = H^{-1}(w_i).$$

If H is a *secret* one-way function, this step is clearly infeasible. However, even for a *public* one-way function, this step is also infeasible, since the computing w_{i+1} and H^{-1} is infeasible given H and w. Hence, using a secret function is not required.

b) Check if each of the four basic requirements on hash functions is necessary:

1. H is easy to compute:
   Recall: *Given $m \in \mathcal{M}$, $H(m)$ is easy to compute.*
   This not required, but still a very useful property to provide an efficient protocol.
2. H is preimage resistant: (required ✓)
   Recall: *Given $y \in \mathcal{Y}$, it is infeasible to find m such that $H(m) = y$.*
   Otherwise, $w_i = H(w_{i+1})$ could be broken, see a).

3. H is second preimage resistant: (required ✓)
   Recall: *Given $m \in \mathcal{M}$, it is infeasible to find $m' \neq m$, such that $H(m) = H(m')$.*
   Otherwise, the attacker would be able to find a w' such that $H(w') = H(w_{i+1})$.
4. H is collision-free:
   Recall: *It is infeasible to find $m \neq m' \in \mathcal{M}$ with $H(m) = H(m')$.*
   Although finding an arbitrary collision would indeed break the system, it will affect a random chain of passwords in this scheme with negligible probability.

c) The discrete logarithm problem is hard to solve in $\mathbb{Z}_p^*$:
It is hard to determine x in $a^x \equiv y \mod p$ for given values of the primitive element a modulo p and y.
Lamport's protocol in terms of the discrete logarithm problem is described by:

- Functions and Parameters:
  Use the one-way hash-function $H : \{2, ..., p-2\} \to \mathbb{Z}_p^*$ with $w \to a^w \mod p$.
  Choose a secret value $w \in \{2, ..., p-2\}$ and a primitive element $a \mod p$.
  Choose t, the maximal number of identifications.
  Select the initial value $w_0 = H^t(w)$.
- Protocol steps:
  Compute next session key $H^{t-i}(w) = w_i$.
  Session authentication $A \to B : (A, i, w_i)$.
  B checks if $i = i_A$ and $w_{i-1} \equiv a^{w_i} \mod p$ is true.
  If correct, B accepts, sets $i_A \leftarrow i_A + 1$ and stores w_i for the next sesssion.

d) *Man-in-the-middle attack* on Lamport's protocol:
Let E intercept the current key w_i from A. E uses it for authentication as A at B. Furthermore, if E gains access to the initial value w and knows the current session number i, the protocol is completely broken.

Solution of Problem 30

a) Claimant Alice (A) wants to prove her identity to verifier Bob (B). This identification is done for a fixed password by comparing its hash value to a stored hash value. The password is sent without protection: $A \overset{pwd}{\to} B$. B calculates $h(pwd)$ and compares it with the stored hash value, to verify the identity of A.

In a *replay attack*, eavesdropper Eve (E) intercepts the password and impersonates A by reusing the password in a later session:

$A \overset{pwd}{\to} B$ (plain password transmission)

$A \overset{pwd}{\to} E$ (by intercepting/eavesdropping)

$E \overset{pwd}{\to} B$ (impersonating A)

Improvement: Instead of revealing the password itself, a time stamp is encrypted with a symmetric (secret) key. By comparing the time stamp with its internal clock, B can verify that the claimant A knows the shared secret key. After authentication, the response is expired and cannot be reused.

Authentication protocol:

$B \to A : t_A$ (time stamp implicit in internal clock, no challenge necessary)

$A \to B : E_K(t_A)$ (response)

Alternatively, the challenge can be made explicit, by taking a random value r_B:

$B \to A : r_B$ (explicit challenge)

$A \to B : E_K(r_B)$ (response)

b) Consider the following authentication protocol:

$A \to B : r_A$ (A challenges B)

$B \to A : E_K(r_A, r_B)$ (B responds to A and challenges A)

$A \to B : r_B$ (A responds to B)

In the *reflection attack*, E uses A to reveal the correct responds:

$A \to E : r_A$ (challenge)

$E \to A : r_A$ (the same challenge back)

$A \to E : E_K(r_A, r_{A'})$ (response)

$E \to A : E_K(r_A, r_{A'})$ (the same response back)

$A \to E : r_{A'}$ (second response)

$E \to A : r_{A'}$ (the same second response back)

Remark: No user B is involved here, only the 'reflection' of A.

c) Consider the following mutual authentication protocol:

1. $A \to B : r_A$ (challenge)
2. $B \to A : S_B(r_B, r_A, A)$ (response and 2nd challenge)
3. $A \to B : r'_A, S_A(r'_A, r_B, B)$ (2nd response)

The *interleaving attack* uses the information of simultaneous sessions:

$E \to B : r_A$ (1st session 1.)

$B \to E : r_B, S_B(r_B, r_A, A)$ (1st session 2.)

$E \to A : r_A$ (2nd session 1.)

$A \to E : r'_A, S_A(r'_A, r_B, B)$ (2nd session 2.)

$E \to B : r'_A, S_A(r'_A, r_B, B)$ (1st session 3.)

Now E can impersonate as A to B. Remark: In this case the sessions of two protocols are interleaved (overlapped) like in a man-in-the-middle attack.