Prof. Dr. Rudolf Mathar, Jose Calvo, Markus Rothe

Tutorial 10
- Proposed Solution -

Friday, January 22, 2016

Solution of Problem 1

Useful sources to study the Kerberos protocol are, e.g.:

- *Trappe, Washington - Introduction to Cryptography with Coding theory (Chapter 13)*
- *http://en.wikipedia.org/wiki/Kerberos_(protocol)*

Unilateral authentication by the Kerberos protocol with a ticket granting server:

1. *User logon, A requests client authentication at T to use G:*
   $A \to T : A, G$
2. *T grants client authentication for A at G:*
   T generates session key k_{AG}.
   T generates a ticket granting ticket (TGT): $TGT = G, E_{k_{TG}}(A, t_1, l_1, k_{AG})$.
   $T \to A : E_{k_{AT}}(k_{AG}), TGT$
3. *A requests client authentication for service at G:*
   A recovers k_{AG} using the shared key k_{AT}.
   A generates an authenticator $a_{AG} = E_{k_{AG}}(A, t_2)$.
   $A \to G : a_{AG}, TGT$
4. *G grants service to A:*
   G recovers A, t_1, l_1, k_{AG} from the TGT using k_{TG}.
   G recovers A, t_2 from a_{AG} using k_{AG}.
   G checks if the time stamp is within the validity period $(t_2 - t_1) < l_1$.
   G verifies A if authenticator and the ticket are correct.
   G generates session key k_{AB} and service ticket ST using k_{BG}: $ST = E_{k_{BG}}(A, t_3, l_2, k_{AB})$.
   $G \to A : ST, E_{k_{AG}}(k_{AB})$
5. *A communicates with B with the authenticated service of G:*
   A recovers k_{AB} using k_{AG}.
   A generates authenticator $a_{AB} = E_{k_{AB}}(A, t_4)$.
   $A \to B : a_{AB}, ST$
   B recovers A, t_3, l_2, k_{AB} from ST using k_{BG}.
   B recovers A and t_4 from a_{AB} using k_{AB}.

B checks if the time stamp is within the validity period $(t_4 - t_3) < l_2$.
B verifies A if authenticator and service ticket are correct.
Then, A is successfully authenticated to B.

Solution of Problem 2

Parameters: $n = pq$ with $p, q \equiv 3 \mod 4$, and p, q secret primes.
Each user chooses an arbitrary sequence of seeds $s_1, ...s_K \in \{1, ..., n-1\}$,
with $\gcd(s_i, n) = 1$ and publishes: $v_i = (s_i^2)^{-1} \mod n$.

A public hash function is applied:

$$H : \{0,1\}^* \to \{(b_1, ..., b_K) \mid b_i \in \{0,1\}\}$$

Signature generation:

(i) A chooses an arbitrary value $r \in \{1, ..., n-1\}$ and calculates $x \equiv r^2 \mod n$. (witness)

(ii) A calculates: $h(m, x) = (b_1, ..., b_k)$ (challenge)
and afterwards $y \equiv r \prod_{j=1}^{K} s_j^{b_j} \mod n$ (response)

(iii) The signature of m is (x, y):
$A \to B : m, x, y$

Verification:

(i) B calculates $h(m, x) = (b_1, ..., b_K)$. (challenge)

(ii) B calculates $z \equiv y^2 \prod_{j=1}^{K} v_j^{b_j} \mod n$. (response)

(iii) B accepts the signature if $z = x$ holds.

Proof that this signature and verification scheme is correct:

$$z = y^2 \prod_{j=1}^{K} v_j^{b_j} \equiv \underbrace{r^2}_{\equiv x} \underbrace{\prod_{j=1}^{K} s_j^{2b_j} \prod_{j=1}^{K} v_j^{b_j}}_{\equiv 1} \equiv x \mod n. \blacksquare$$

Solution of Problem 3

a) The secret service (MI5) chooses an arbitrary seed $s \in \mathbb{Z}_n$ per iteration.

The MI5 calculates the quadratic residue $y \equiv s^2 \mod n$:

MI5 $\rightarrow$ JB: y

JB calculates the four square roots of y modulo n using the factors p, q of n.
JB chooses a square root x:

JB $\rightarrow$ MI5: x

The MI5 verifies that $x^2 \equiv y \mod n$.

Since JB has no information about s, he chooses the x with probability $\frac{1}{2}$, such that $x \not\equiv \pm s \mod n$.
If the MI5 receives such an x, n can be factorized:

$$\begin{aligned} y \equiv s^2 \equiv x^2 &\mod n \\ \Rightarrow s^2 - x^2 \equiv 0 &\mod n \\ \Rightarrow (s-x)(s+x) \equiv 0 &\mod n. \end{aligned}$$

The probability that JB always fails by sending $x \equiv \pm s \mod n$ in all 20 submissions is:

$$\tfrac{1}{2^{20}} = \tfrac{1}{1048576} \approx 10^{-6}.$$

b) *Zero-knowledge property:* No information about the secret may be revealed during the response.

However, in this protocol it is even possible, that the full secret s is revealed. Hence, this is not secure a zero-knowledge protocol!

c) A passive eavesdropper E can only obtain the values x and y. E only knows the square roots $\pm x$ of y modulo n, which is useless in the next iteration. This knowledge is not sufficient to factorize n.