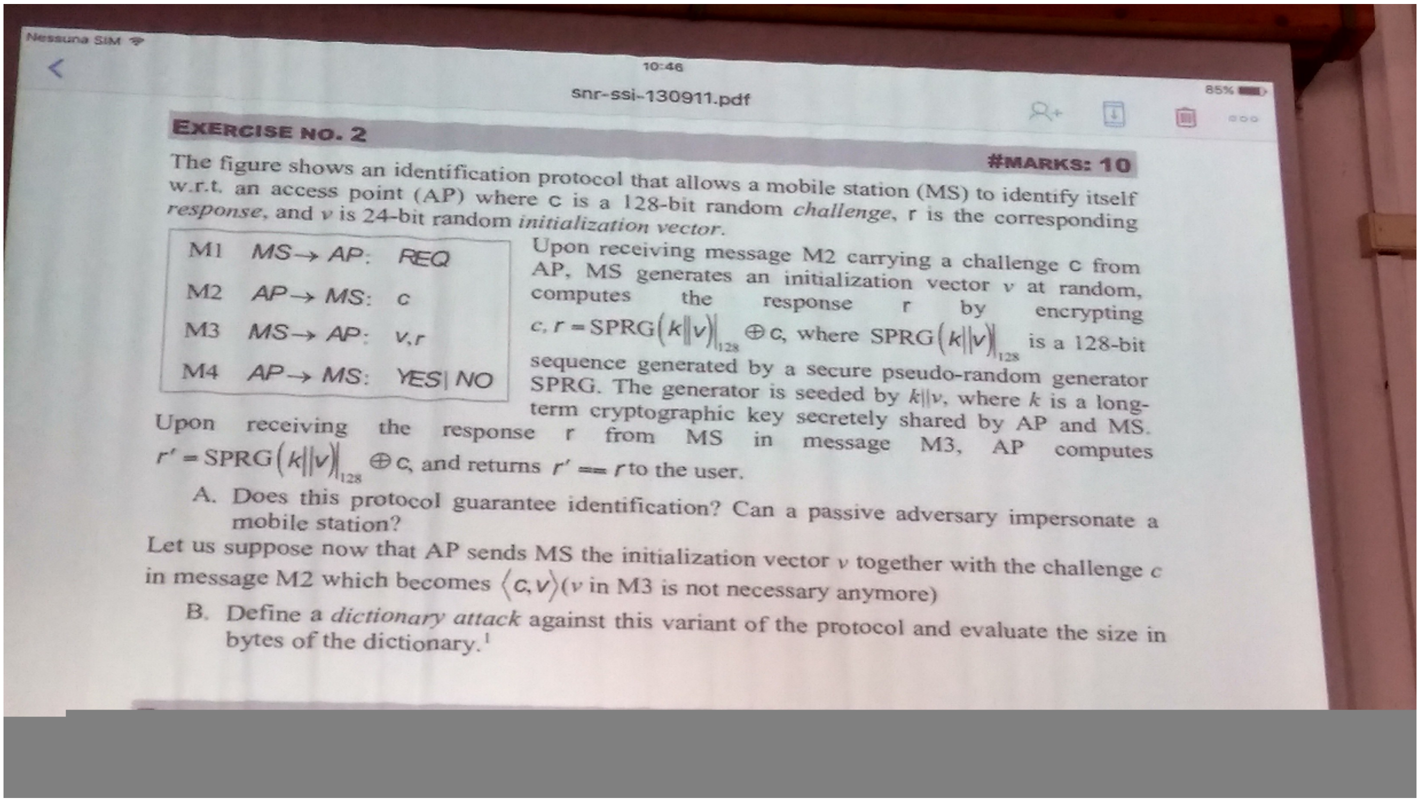
September 14th, 2013

giovedì 25 maggio 2017 10:45

Exercise 2



Nessuna SIM 10:46 85%

snr-ssi-130911.pdf

EXERCISE NO. 2 **#MARKS: 10**

The figure shows an identification protocol that allows a mobile station (MS) to identify itself w.r.t. an access point (AP) where c is a 128-bit random *challenge*, r is the corresponding *response*, and v is 24-bit random *initialization vector*.

M1	$MS \rightarrow AP$:	REQ
M2	$AP \rightarrow MS$:	c
M3	$MS \rightarrow AP$:	v, r
M4	$AP \rightarrow MS$:	$YES \mid NO$

Upon receiving message M2 carrying a challenge c from AP, MS generates an initialization vector v at random, computes the response r by encrypting c, $r = \text{SPRG}(k \| v)|_{128} \oplus c$, where $\text{SPRG}(k \| v)|_{128}$ is a 128-bit sequence generated by a secure pseudo-random generator SPRG. The generator is seeded by $k \| v$, where k is a long-term cryptographic key secretely shared by AP and MS. Upon receiving the response r from MS in message M3, AP computes $r' = \text{SPRG}(k \| v)|_{128} \oplus c$, and returns $r' == r$ to the user.

A. Does this protocol guarantee identification? Can a passive adversary impersonate a mobile station?

Let us suppose now that AP sends MS the initialization vector v together with the challenge c in message M2 which becomes $\langle c, v \rangle$ (v in M3 is not necessary anymore)

B. Define a *dictionary attack* against this variant of the protocol and evaluate the size in bytes of the dictionary.[1]

September 14th, 2013

A. This protocol guarantee identification because the challenge is encrypted by XORing it with a random number seeded with *k*, that is a long-term shared secret key between the AP and the MS. Only the MS knows *k*, so only that MS can generate such a random number.
   If an adversary wants to impersonificate the MS, it has to guess the *k* shared key in order to reply to the challenge correctly.
   The adversary cannot reply M3 because the AP changes it challenge *c* at every protocol iteration (the probability of re-using the same *c* key value is very low).
   v though is relatively slow.
   The only weakness is on the shared secret key *k*.
   z = SPRG(k||v) is called **keystream**.
   $r = z \oplus c$
   a. Eavesdropping
      i. All of them are public quantities, not related to each other.
      ii. z can be calculated by doing $z = r \oplus c$
   b. The adversary can start a new protocol instance
      This can be done because *v* is always the same: it doesn't need to discover *k*, it just uses *z*.
      i. It receives c' as a challenge
      ii. It can send v and $r' = z \oplus c'$, with a previously spoofed *z*.
   This protocol was used by WEP.
B. The adversary can still calculate *z*, but the AP could use another initialization vector *v*, so the previous attack could not be performed anymore.
   The adversary could build a dictionary of (v, z) pairs.
   If the AP re-uses a *v*, the adversary would have a corresponding z.
   v are on 24-bits, so they're not so much: the dictionary would be $2^{24}\ entries \cdot \frac{128\ bits\ for\ each\ keystream}{8 \frac{bits}{Byte}} = 2^{24} \cdot 2^{4} = 2^{28} = 256\ MB$
   If *v* is generated by means of a counter, everytime the AP is rebooted, the *v* is generated from 0, and so on.
   Otherwise, 2^{24} IVs is not a large number of them.