Security on the Internet, winter 2008

MICHAEL NÜSKEN, DANIEL LOEBENBERGER

10. Exercise sheet
Hand in solutions until
Monday, 19 January 2009, 11^{59}am (deadline!).

As usual: Any claim needs a proof or an argument.

Exercise 10.1 (Keyed MAC). (5+2 points)

Along with authenticity (the request that the two peers authenticated reciprocally by using a public key signature mechanism) and privacy (secret key encrypted communication), an important issue in secure communication protocols is the *message integrity*. Assume that Alice and Bob decide that they have no need for encrypted communication (which might also slightly slow down the communication). They definitively wish to maintain the integrity of their communication and ascertain that no third party can interfere and modify the messages they send to each other. Hashing may be helpful in this respect: if Alice adds the hash value $h(M)$ to a message M sent by her, then Bob can compute the hash of the received message and compare the to the received hash. Due to the collision resistance of hashes, M cannot be *partially* altered. An eavesdropper may however send a totally different message $M'|h(M')$ together with its valid hash: hashing is not sufficient for defending against this kind of attack. However, if Alice and Bob have established a common shared secret S at session initialization, having a hash method which uses this secret would protect against message manipulation. The term of MAC (*message authentication code*) is standard for this idea of combining hashing with a shared secret value.

In this exercise you will discover the current standard HMAC which can be used in combination with various hash methods. You can download the paper *Keying Hash Functions for Message Authentication*, in which a universal MAC function is described, which can be used together with various hash functions.

(i) Read the paper and give an algorithmic description of the procedure for creating a HMAC on an input message M (of, say, 1 MB) using the SHA1 hash function and the secret key *secret*. 5

(ii) In `cryptool`, using the shared secret defined in the previous exercise, generate the HMAC of some text of own choice, by following your own algorithmic description given in (i). +2

Exercise 10.2 (AtE and died: confidentially poisoned). (12+2 points)

Horton's principle says that one should always prove the integrity of the *plain text*. One solution to ensure the integrity is to first authenticate and then encrypt (AtE). Though this paradigm is clearly correct and the conclusion grants integrity as desired, we overlooked a different issue here. This exercise shall prove it.

Suppose we use some encryption function ENC_{K_e} and any message authentication function MAC_{K_a}. For a message m we compute $a := \mathrm{MAC}_{K_a}(m)$ and send $c := \mathrm{ENC}_{K_e}(m|a)$. (Here, the vertical line '|' denotes concatenation.)

Assume both are as secure as you like. In particular, the encryption function shall guarantee that even to a *c*hosen *p*laintext *a*ttacker the encryptions of two known plain texts are *ind*istinguishable. In other words, there is no (ie. no probabilistic polynomial time) so-called IND-CPA attacker: the attacker may ask for encryptions of chosen plain texts and he fixes two further plain texts m_0, m_1 for which he never inquired the encryption. Finally, the attacker is given the encryption of m_0 or of m_1 and shall tell which of the two plain texts was used. One possible encryption function under these constraints is the one-time pad (assuming that the encryption procedure keeps track of the already used parts of the key).

Now, suppose additionally that the encryption XORs something on the cipher text (like AES-CTR), and define a variant $\mathrm{ENC}^*_{K_e}$ of this encryption function as follows: first replace every 0-bit by two bits 00 and every 1-bit by two bits 01 or 10, choose randomly each time, next encrypt with ENC_{K_e}. For the decryption we translate 00 back to 0, 01 and 10 to 1, and 11 is considered as a transmission error. So we send $\mathrm{ENC}^*_{K_e}(m|\,\mathrm{MAC}_{K_a}(m))$.

2+2 (i) Prove (at least, argue) that $\mathrm{ENC}^*_{K_e}$ is still secure in the previous sense.

4 (ii) Suppose that malicious Michael (or hoeing Hugo) has overheard the messages of your login to some server which was done by sending the password. Of course, your password was authenticated and encrypted, as all messages. Now, malicious Michael takes the transmission of your password and resends it with a bit pair in the cipher text inverted.

(a) How does the recipient react if the original bit was 0?

(b) How does the recipient react if the original bit was 1?

Conclude that Michael learns the bit from the reaction of the server (and thus your passwords after enough trials).

2 (iii) Estimate the effect of this observation.

2 (iv) In SSH we transmit $\mathrm{ENC}_{K_e}(m)|\,\mathrm{MAC}_{K_a}(m)$, so we authenticate and encrypt (rather than first authenticating and second encrypting). Is that better? [Try to use $\mathrm{ENC}^*_{K_e}$ here.]

2 (v) In IPsec we transmit $\mathrm{ENC}_{K_e}(m)|\,\mathrm{MAC}_{K_a}(\mathrm{ENC}_{K_e}(m))$. Is the previous attack here also successful? What about the paradigm?