Computer Security – Practical Exercise I

Course Lecturer: Mike Just

Posted: 31 January 2010

Abstract

The is the first practical exercise for the Computer Security course. Please record your solutions, and bring them to your Tutorial I session during the week of February 15th, 2010, at which time we will discuss the answers, as well as concepts related to the topics in this practical.

If you have any problems or questions with any part of this practical, please use the newsgroup `eduni.in.course.cs` to ask for assistance. As well, since this practical will not be formally assessed, you are encouraged to work on the answers with your classmates.

As a guide, you will likely spend approximately 8-10 hours completing the answers, including time for additional reading. And please remember that I make ask questions on the final exam that are related to concepts learned on this practical, and on material from the corresponding tutorial session.

Part A - Hashes, MACs and Integrity

The following questions are intended to help you further understand some of the concepts learned in the lectures regarding the use of hash functions and Message Authentication Codes (MACs) for integrity.

Question 1

A recent submission to a security conference proposed the following Message Authentication Code (MAC) algorithm in order to improve the efficiency of MAC computation. Instead of inputting the entire message directly into the MAC function computation, the message m is first divided into 128-bit pieces (with zeroes padded to the end of the message to obtain a multiple of 128 bits) and all of the pieces are XORed together to obtain a 128-bit result R. The MAC function is then applied to the shorter result R, as opposed to the original message m.

1. Describe at least one way to find two (meaningful) messages $m \neq m'$, such that the messages convey a different meaning, yet the resulting MAC values are the same.
2. Confirm your proposal from above by finding and downloading code to a MAC algorithm of your choice, implement the proposed changes to produce R as described above, find two such messages using your solution above, and compute the MAC for each using your code.

Question 2

Recall the encryption of a message m using Output Feedback (OFB) mode as discussed in class. Suppose that in addition to encrypting, message integrity is provided to the message m by computing $c = E_K(m||h(m))$ using a hash function h.[1] Show how an attacker can, knowing only the plaintext-ciphertext pair (m, c) produced with the key K, encrypt and provide message integrity to another message m', i.e., compute c' such that $c' = E_K(m'||h(m'))$.

Question 3

Consider using a symmetric block cipher E to encrypt a message $M = m_1, m_2, \ldots, m_n$ into a set of ciphertext blocks $C = c_1, c_2, \ldots, c_n$ using a mode of operation whereby blocks are encrypted as $c_t = m_t \oplus w_t$, and the internal variable is defined as $w_0 = IV$ for some IV and $w_t = E_K(w_{t-1})$ for the remaining variables.

Let the resultant ciphertext blocks be stored in some file. Suppose that block c_i is deleted from the file and that blocks $c_{i+1}, \ldots, c_n$ are decrypted, re-encrypted and re-filed using the same variables described above so that $c'_i = m_{i+1} \oplus w_i$, $c'_{i+1} = m_{i+2} \oplus w_{i+1}$, and so on to $c'_{n-1} = m_n \oplus w_{n-1}$.

[1] For simplicity, assume that the message m has length equal to a multiple of 128 bits, that the hash function h produces a 128-bit output, and that no padding of the message is performed.

1. Show that if m_i is known, as well as the contents of the original file of ciphertext, and the updated file of ciphertext, then for all $t \geq i$, all keystream variables w_t and all plaintext blocks m_t can be deduced.

2. Based upon the results above, suggest some guidance to users of this mode of operation, regarding conditions under which they should, or should not, use this mode.

Part B - PRNGs and Key Entropy

The following questions are intended to help you further understand some of the concepts learned in the lectures regarding the use of pseudo-random number generators, and key entropy.

Question 1

A bit source S produces a statistically biased random sequence of bits $b_1 b_2 b_3 \ldots$ and it is known that for each bit b_i, $prob(b_i = 0) = p$ and $prob(b_i = 1) = (1 - p)$, for some $0 < p < 1$ where $p \neq 1/2$. Devise a simple algorithm to extract from S an unbiased random bit sequence $a_1 a_2 a_3 \ldots$ (i.e., produce a bit sequence such that $prob(a_i = 0) = prob(a_i = 1) = 1/2$). Justify that your algorithm works as required.

Question 2

A colleague is preparing a presentation in which he wants to demonstrate the superiority of the public key cryptosystem RSA over the conventional cryptosystem AES. One of his arguments is that because RSA uses a key size of 2048 bits, while AES uses a key size of 128 bits, RSA offers a more secure alternative (because 2048 is a much larger number than 128). Do you agree with this argument? Why or why not?

Question 3

A vendor is attempting to sell cryptographic software to your company and suggests that their software uses a key length of 160 bits, and hence requires $O(2^{160})$ operations to attack the cipher. The vendor explains that these 160 bits are derived as follows: the software repeatedly selects 20 random characters from the set $\{a, \ldots, z\}$ and that since 8 bits are used to encode each character, the key length is $8 * 20 = 160$ bits long. Do you believe his claim that this cipher offers 160 bits of security through its key? Why or why not?

Part C - Fun with Protocols

The following question is intended to help you further understand some of the concepts learned in the lectures regarding the use and potential pitfalls of (seemingly simple) protocols.

Question 1

Each user U in a communication network has a unique public encryption algorithm PE_U, and private decryption algorithm PD_U. User A will always send a message to user B with the following protocol:

- A sends to B the message $(PE_B(m), A)$, where 'A' identifies A as the source of the message.
- B decrypts to recover m and acknowledges A's message by returning $(PE_A(m), B)$ back to A.

Unfortunately, this protocol is not secure against an *active attacker*, as you will now demonstrate.

1. Let X be an opponent capable of intercepting and inserting their own messages. Show how X can recover a copy of the message m as an active attacker between the above exchange of A and B.

2. Being aware of the above attack by X, A and B modify the protocol such that they now send and respond with, respectively, the messages $PE_B(PE_B(m), A)$ and $PE_A(PE_A(m), B)$. Show that this modified protocol still does not guarantee the confidentiality of the message m.