

## HW #6: Access Control/Security Design Principles

CS 392/681: Computer Security  
Fall 2006

**[100pts] DUE 11/05/2006 (midnight)**

### **Problem 1 [10+15pts]**

I mentioned in the class that the “textbook” RSA that we studied is not secure. The primary reason is that the “textbook” RSA is deterministic, i.e., when same message is encrypted twice, we get the same ciphertext (see some other attacks on “textbook” RSA in the textbook handbook of applied cryptography). In order to make it secure, a padding mechanism (which also involves hash functions) is employed. This padding mechanism is called OAEP (optimal asymmetric encryption padding). Download the paper explaining the details regarding RSA OAEP at <http://www.cs.ucsd.edu/~mihir/papers/oaep.pdf>, read through it and answer the following questions:

1. Explain how RSA OAEP encryption and decryption works (you can cut/paste the “picture” in the paper when you write your answer)
2. Refer back to the HW #3 Problem 3, where you experimented with the timing of RSA encryption/decryption using the code at <http://www.funet.fi/pub/crypt/cryptography/asymmetric/rsa/rsaref2.tar.gz>

First check if this code uses RSA OAEP (let me know if it does not). Look at the source code that implements OAEP padding. Then, time the cost of performing OAEP padding on a message of length 160-bit. Also, time the cost of “inverse padding”.

### **Problem 2 [20+6=26pts]**

1. In an organization, there are a total of  $s$  subjects and  $o$  objects. The subjects are also divided among groups consisting of a total of  $r$  roles. While incorporating access control for this organization, the security architect of the organization has the choice of either using the approach of access control matrix or the role-based access control. If you were the security architect, which approach would you recommend using, under what conditions and why?
2. In an operating system, access control is implemented using 18 rings. A data segment  $D$  has an access bracket  $(3,15)$ . What access permissions a process  $P1$

executing in ring 16 has on the data segment D? What access permissions a process P2 executing in ring 6 has on the data segment D? Assume that a process P3 executing in ring 2 is infected with a virus. Can this virus have a potential adverse effect when process P3 refers to the data segment D?

### **Problem 3 [8\*3=24pts]**

Give an “original” example (thought of by yourself, different than the one discussed in class or in the textbook) of a real-world security mechanism each that violates

1. Principle of least privilege
2. Principle of fail-safe defaults
3. Principle of economy of mechanism
4. Principle of complete mediation
5. Principle of open design
6. Principle of separation of privilege
7. Principle of least common mechanism
8. Principle of psychological acceptability

Explain your answer in each case.

### **Problem 2 [25pts]**

In the class, we discussed a way to implement principle of separation of privilege using polynomial secret sharing. Here is a reminder how a  $(t+1)$ -out-of- $n$  secret sharing scheme works. A trusted authority TA picks up a secret  $x$  in  $Z_q$  ( $q$  is a 160-bit prime). The TA picks  $(a_1, a_2, \dots, a_t)$  all in  $Z_q$  and sets the polynomial

$$f(z) = x + a_1 * z + a_2 * z^2 + \dots + a_t * z^t \pmod{q}$$

TA then issues an entity with identity  $ID_i$  a secret share  $x_i$ , such that  $x_i = f(id_i)$  (this is done over a secure channel between the TA and each entity)

Now, since  $f(z)$  is a  $t$  degree polynomial, any set of  $(t+1)$  entities can collaborate together and recover the secret  $x$  (or open up a lock that is encrypted with  $x$ ). Moreover, if at most  $t$  entities are malicious (or corrupted), they can not recover the secret  $x$ .

The above scheme is not robust against malicious entities in the system. In other words, when a set of entities want to recover the secret  $x$ , the malicious (or corrupted) entities might not send their correct secret shares. This way the secret can not be recovered (e.g., the lock can not be opened). This is an example of denial-of-service.

Your task in this problem is to modify the above scheme in such a manner that it becomes robust to at most  $t$  malicious (or corrupted) entities. Basically, you need to come up with a mechanism using which the entities can verify the correctness of each others' secret shares during the recovery of secret  $x$ .

*[Hint: you can use a discrete logarithm setting to achieve this. That is use  $(p, q, g)$  parameters, and the idea that if a secret value is  $x$  in  $\mathbb{Z}_q$ , one can safely publish the  $y = g^x \pmod{p}$ ]*