

Information Assurance: Homework 6 – Answer key

Due October 20, 2006

1. Consider attack #2 on RSA digital signatures discussed in section 10.6.2.1 of the text. If the CA enforced that all public key changes much change both the exponent and the modulus of the public key, would this attack be avoided? Why or why not.

In this attack Bob changes his public key exponent to make it appear that Alice has encrypted and signed message M instead of message m .

$$(m^{e_B} \bmod n_B)^{d_A} \bmod n_A.$$

Bob knows the factors of n_B , so he can compute r for the following

$$M^r \bmod n_B = m$$

Therefore, he can replace M^r for m in the original message

$$(M^{re_B} \bmod n_B)^{d_A} \bmod n_A$$

If he registers (re_B, n_B) as his new public key, then it will appear that M is the message that Alice encrypted and signed. But does the modulus have to stay the same? Say Bob was forced to pick a new modulus n_x which he also knows the factorization of.

He can still compute

$$M^x \bmod n_x = m$$

He can also compute

$$m^y \bmod n_x = m^{e_B} \bmod n_B$$

Again, he can compose

$$m^y \bmod n_x = M^{xy} \bmod n_x = m^{e_B} \bmod n_B$$

And register (xy, n_x) as his new public key. Bob is forced to do a bit more work but the attack is not computationally prohibitive.

2. Show that, under the Yaksha key escrow scheme that Alice can obtain the session key by computing $(C_{\text{Alice}})^{d_A} \bmod n_{\text{Alice}}$.

$$\begin{aligned} C_{\text{Alice}} &= (k_{\text{session}})^{d_{\text{Alice}}K^*e_{\text{Alice}}} \bmod n_{\text{Alice}} \\ (C_{\text{Alice}})^{d_{\text{Alice}}} \bmod n_{\text{Alice}} &= ((k_{\text{session}})^{d_{\text{Alice}}K^*e_{\text{Alice}}} \bmod n_{\text{Alice}})^{d_{\text{Alice}}A} \bmod n_{\text{Alice}} \\ &= (k_{\text{session}})^{d_{\text{Alice}}K^*d_{\text{Alice}}A^*e_{\text{Alice}}} \bmod n_{\text{Alice}} \end{aligned}$$

Remember that $d_{Alice}K * d_{Alice}A \bmod \Phi(n_{Alice}) = d_{Alice}$ the private exponent of the RSA key pair. Therefore $d_{Alice} * e_{Alice} \bmod \Phi(n_{Alice}) = d_{Alice}K * d_{Alice}A * e_{Alice} \bmod \Phi(n_{Alice}) = 1$.

$$(k_{session})^{(x\Phi(n_{Alice})+1)} \bmod n_{Alice} = k_{session} * k_{session}^{x\Phi(n_{Alice})} \bmod n_{Alice}$$

By Fermat's little theorem $k_{session}^{\Phi(n_{Alice})} = 1$. So

$$k_{session} * k_{session}^{x\Phi(n_{Alice})} \bmod n_{Alice} = k_{session} * (1)^x \bmod n_{Alice} = k_{session} \bmod n_{Alice}.$$

3. After the initial ESP proposal, reviewers complained that an encryption protocol without integrity verification was not useful. Why is this the case? What harm could a interceptor do in this case? Assume the interceptor has no knowledge of secrets between the IPSec peers.

At a minimum, the interceptor could cause a denial of service by randomly changing bytes in the IPSec packet. Depending on the higher level protocols, the change might be detected, or things might just behave very strangely. The receiver may or may not be able to tell that the decrypted data is invalid.

The header fields of the IP and IPSec header sent in the clear could be changed with out detection. Changing these headers could cause some problems if not detected.

Steven Bellovin goes through quick a number of more specific attacks in "Problem Areas for IP Security Protocols"

<http://www.usenix.org/publications/library/proceedings/sec96/bellovin.html>.

One of the attacks involve IPSec communication between two multi-user machines A and B. Assume that the SA's are setup to include all IP communication between the peer. Assume the bad guy can sniff the packet. Say it is a TCP packet. As a non-privileged user on the machines, he cannot dig into the stack and access the session keys directly. But he can send his own UDP packet to a peer process on machine B. With copies of both packets (encrypted by the same session key), he can compose a new packet from A to B with his packets header and a body of the good guy's packet. If the SA's are configured to use DES in CBC mode, the self healing property means that the body of the new packet will be mostly decrypted even if the initial IV is wrong.

4. A system allows the user to choose a password with a length of one to ten characters inclusive. Assume that 20,000 passwords can be tested per second. The system administrators want to expire passwords once they have a probability of 0.10 of having being guessed. Determine the expected time to meet this probability under each of the following conditions.

For all cases, let

c = number of characters in the alphabet.

R = the rate of guessing, 20^4 scaled to the appropriate time unit.

N = the total number of passwords

T = the time to reach the expected probability of having a password broken

Then

$(R \times T) / N = 0.10$, the total number of guesses over the total number of passwords is the probability $1/10$

$$T = N / (10R)$$

The total number of passwords is

$$N = \text{sum of } i=1 \text{ to } 10 = (1 - c^{11}) / (1 - c)$$

- a. Password characters must be digits (“0” through “9”).

$$c = 10, T = 55,555 \text{ seconds or } 15 \text{ hours}$$

$$T = 50,000 \text{ seconds if you consider only } 10 \text{ character long passwords}$$

- b. Password characters may be capital letters (“A” through “Z”) and numerics (“0” through “9”).

$$c = 36, T = 1.88 \times 10^{10} \text{ seconds or } 596 \text{ years}$$

$$T = 1.82 \times 10^{10} \text{ seconds or } 579 \text{ years if you only consider } 10 \text{ character long passwords}$$

- c. Passwords are entered using the Elbonian alphabet of 156 characters.

$$c = 156, T = 4.295 \times 10^{16} \text{ seconds or } 1.36 \times 10^9 \text{ years}$$

$$T = 4.267 \times 10^{16} \text{ seconds or } 1.35 \times 10^9 \text{ years if on only consider } 10 \text{ character passwords}$$

5. Try running the John the Ripper password cracking program

<http://www.openwall.com/john/>. You should be able to install it local to your environment for an unprivileged account. Obtain a password file from <http://www.cs.uiuc.edu/class/fa06/cs498sh/hw6/hw6-passwd>. This file contains nine accounts with passwords from a linux system. Four passwords should be cracked very easily. If you have access to a private system, try running the program for a while longer to see if you get more passwords cracked. I will also be posting word lists to the newsgroup that you can use to try to improve the cracking. Submit the account names and passwords that you crack. As long as you get four of the passwords, you will get full credit.

The accounts and passwords were

- *alice/alice*
- *bob/bobbob*
- *carol/carol77*
- *dave/12-17-2000*

- *ellen/EllenP*
- *grace/clarinet*
- *fred/lassie*
- *helga/flyaway*
- *ivan/lunch'*

6. A computer system uses biometrics to authenticate users. Discuss ways in which an attacker might try to spoof the system under each of the following conditions.
- a. The biometric hardware is directly connected to the system, and the authentication software is loaded onto the system.

The attacker could try to feed in authentication information for valid registered users. For example if the biometric hardware read fingerprints, he could try to get the fingerprint of a valid user and feed it in (ala the gelatin finger).

Otherwise, the attacker would have to somehow get access to the target computer to change the complementary information stored on the device. Though if the attacker can gain access to this sensitive information already it isn't clear why he would want to mess with the biometric interface.

- b. The biometric hardware is on a stand-alone computer connected to the system, and the authentication software on the stand-alone computer sends a “yes” or “no” to the system indicating whether or not the user has been authenticated.

The attacker could try the same attacks as in the first case, but the attacker try to simply bypass the biometric hardware. She could reverse engineer a piece of hardware that has the same interface as the biometric hardware. But the attacker's version only generates a “yes” response. This of course assumes that the attacker has sufficient physical access to replace the biometric hardware and no one notices her fiddling with the hardware.