CSCD27 – Network Security

Shih-Chin Liang 10/10/2014

Question 1: Course Feedback

Send it to you via email

Question 2: PGP/GPG: secure email in practice

——BEGIN PGP PUBLIC KEY BLOCK——
Version: GnuPG v1.4.11 (GNU/Linux)
mI0EVDhPOwEEAMRY4s4+rqnwt2MwrC74Ab9bgGTMzPQksBCEExGXmQpdaEcPUu7X
gFjDCWDtRCeMxcIPsDwOwClNtc4dv0bM5v5/JmEfMZcDtbU1hKqxrrKwfwu5nk/q
UnBLDKyQkwtavGf2tXLR+Pi7DJu6ZVcQqVVfMImK+i26QsTQP+0SR21tABEBAAG0
Q1NoaWhDaGluIExpYW5nIChDU0NEMjcgRmFsbCAyMDE0KSA8c2hpaGNoaW4ubGlh
bmdAbWFpbC51dG9yb250by5jYT6IvgQTAQIAKAUCVDhPOwIbAwUJAIEzAAYLCQgH
AwIGFQgCCQoLBBYCAwECHgECF4AACgkQ2CNWXVW12wbOJAQAimkPov/w7bQwn90A
SY5u2iYWbOfzfWj69D2PXjugm0CHHkTrLHKP4tMqVMChQD34ql+O9vE9VfsCb7Iv
uO6AEfZHNKi8GmYCOHxsbK/ECnJLLlOLSeTejditn2rdfLxN9obnn++IyNQrlHMQ
zu29uUuWBQSVvMlfNiStHuGH+364jQRUOE87AQQAtUeNvwPckWu4CAhgjgN0mTkz
qQ/1Qr7F0gNKEYmvbvOvLPVTTaRWsOugfbiIUC/1TPdVAiOEe/ZGI+7Zpd4j0XYq
GgcrAtDjKAXZZZzUTvUR07VoWXax/9oGe/H3SjfcYUI86IWu0uFFmDmalvesmWmX
5jmOMGrY+NhH9SsIYUEAEQEAAYilBBgBAgAPBQJUOE87AhsMBQkAgTMAAAoJENgj
Vl1VtdsGUyAD/jLTRNDAaNx3QG0/gJYhDaQfb8zSueOcnohXUPj2vRQwoPU8OpjH
+mPkfJEp7u65UzpMFgSDU7bHTiX5K5krLi3SMCIG9aWh+ipj+pV37Mb5C4HwJ9l5
LpQ8Mcu4BA/JvPUt4okTKcDQjnIKz8Z0W/P8nQBI0Ixhb3q6HiEEzGGL
=LNrB
——END PGP PUBLIC KEY BLOCK——

Question 3: GPG: Revoking a Key

-----BEGIN PGP PUBLIC KEY BLOCK-----
Version: GnuPG v1.4.11 (GNU/Linux)
Comment: A revocation certificate should follow
iMMEIAECAC0FAlQ4YYAmHQJKdXN0IGluIGNhc2UgSSBmb3JnZXQgdGhlIHBhc3Nw
aHJhc2UACgkQ2CNWXVW12wbM4gP/SzIIPk923F8O3vpb6SMq7Y/ltLMas05POZiw
s7YagWW6ZZOoY4Lu0mHaVYCOt4+Yy0oe2vgzNrMi+GKAwp7PWmFyRagYK72xho+k
MNRyMT65AMdEIx8sDEQOh35b9CwFBc+SDuaDXK3eVpNancBNwaWeKivDK7+2Cmls
onOSBPU=
=8A3b
-----END PGP PUBLIC KEY BLOCK-----

Question 4: Insider Threats

Use XL to compile gcc and use the original source code to compile gcc.By doing this, the XL compiled version of gcc denoted as XL_gcc and the original source compiled version of gcc denoted as OS_gcc can be obtained. Then, use XL_gcc and OS_gcc to compile gcc to get A and B respectively. Since XL_gcc and OS_gcc is functionally the same, therefore if the system is not tampered, then A and B will be the same. If $A \neq B$, then this implies the system has been tampered. The system can be rebuild by first fixing the binary of the original gcc and rebuild the system by the bug free gcc.

Reference:http://www.acsa-admin.org/2005/abstracts/47.html

Question 5: Brute-Force Attacks

Core: 16
Clock speed: 3.4GHz = 3,400,000,000 cycle per second
Since $2^{10} = 1024 \approx 10^3$
Also, according to the ietf (http://www.ietf.org/rfc/rfc4772.txt). The average case is the worse case divide by 2

a)**Des:** key length = 56 bits

$$\begin{aligned}\text{Average time} &= 80 \times 2^{55}/(3.4 \times 10^9 \times 16)\\ &\approx 80 \times 10^{16}/(3.4 \times 10^9 \times 16)\\ &= 80 \times 10^7/(3.4 \times 16)\\ &= 1.47059 \times 10^7 \text{ seconds}\\ &= 170 \text{ days}\end{aligned}$$

b)**3Des:** key length = 168 bits

$$\begin{aligned}
\text{Average time} &= 80 \times 2^{167}/(3.4 \times 10^9 \times 16) \\
&\approx 80 \times 10^{49}/(3.4 \times 10^9 \times 16) \\
&= 80 \times 10^{40}/(3.4 \times 16) \\
&= 1.47059 \times 10^{40} \quad \text{seconds} \\
&= 4.66321 \times 10^{32} \quad \text{years}
\end{aligned}$$

c)**AES-128:** key length = 128 bits

$$\begin{aligned}
\text{Average time} &= 80 \times 2^{127}/(3.4 \times 10^9 \times 16) \\
&\approx 80 \times 10^{36}/(3.4 \times 10^9 \times 16) \\
&= 80 \times 10^{27}/(3.4 \times 16) \\
&= 1.47059 \times 10^{27} \;\; \text{seconds} \\
&= 4.66321 \times 10^{19} \;\; \text{years}
\end{aligned}$$

d)**AES-256:** key length = 256 bits

$$\begin{aligned}
\text{Average time} &= 80 \times 2^{255}/(3.4 \times 10^9 \times 16) \\
&\approx 80 \times 10^{78}/(3.4 \times 10^9 \times 16) \\
&= 80 \times 10^{69}/(3.4 \times 16) \\
&= 1.47059 \times 10^{69} \;\; \text{seconds} \\
&= 4.66321 \times 10^{61} \;\; \text{years}
\end{aligned}$$

Question 6: Perfect Ciphers

a):

0 = ’00’, 1= ’01’, 2 = ’10’

It’s not perfect cypher because by formal definition "requires of a system that after a cryptogram is intercepted by the enemy the a posteriori probabilities of this cryptogram representing various messages be identically the same as the a priori probabilities of the same messages before the interception" (from textbook) In other word, the result has to be evenly distributed such that if the attacker obtains the cipher text it won’t help the attacker decrypt it.

The table below shows all the possibilities

$M \oplus K$	Result
$00 \oplus 00$	00
$00 \oplus 01$	01
$00 \oplus 10$	10
$01 \oplus 00$	01
$01 \oplus 01$	00
$01 \oplus 10$	11
$10 \oplus 00$	10
$10 \oplus 01$	11
$10 \oplus 10$	00

In total, ’00’ appears three times, ’01’ appears twice, ’10’ appears twice, ’11’ appears twice implies this is not perfect cypher.

b):

Proposed algorithm:

key + message + 1 mod(3). This will give you a perfect cypher because the result are evenlly distributed and the reason to add one is to avoid matching the key ’00’ and message ’00’ to ciphered text ’00’.

Key + Message + 1	result
00 + 00 + 01	01
00 + 01 + 01	10
00 + 10 + 01	00
01 + 00 + 01	10
01 + 01 + 01	00
01 + 10 + 01	01
10 + 00 + 01	00
10 + 01 + 01	01
10 + 10 + 01	10

In addition, this guarantees that the rows and the columns in the look-up table are all unique as shown below.

Key / Message	00	01	10
00	01	10	00
01	10	00	01
10	00	01	10

Question 7: Feistel Ciphers

The encryption gives you
$L_{i+1} = R_i$
$R_{i+1} = L_i \oplus F(R_i, K_i)$
The ciphertext is (R_{i+1}, L_{i+1})
Now, set $L_i = R_{i+1}, R_i = L_{i+1}$ and plug into the encryption
Since, $R_i = L_{i+1}$ gives you $R_i = L_{i+1}$
$R_{i+1} = L_i \oplus F(R_i, K_i)$ equals $L_i = R_{i+1} \oplus F(R_i, K_i)$
Also, $R_i = L_{i+1}$ which gives us $L_i = R_{i+1} \oplus F(L_{i+1}, K_i)$
This gives you the decryption algorithm. So, even the function F isn't invertible. The decryption can still be performed.

Question 8: DES for Password Authentication

Assuming salting happens before encryption. Thus the input for two cases is E(passowrd + salt, 0) and E(0 + salt, password)

The answer is No, using a key-value of all 0-bits + salt to protect the password is dangerous because once the attacker has your salt and salted password in your password file, it will be really easy for them to figured out the key and hence have all the passwords due to the fact that there aren't many variations when salting with all 0-bits. On the other hand, (password + salt) creates lots of variations because password is hardly the same, even one key is compromised won't effect the safety of the rest. Moreover, if all passwords are encrypted by one key obtaining a bunch of salted passwords can greatly increase the chance of figuring out the key just like the adobe case mentioned in lecture. An extremely dangerous case occurs if the salted value produces 0 as well. Thus, you will encrypt your password with a key of all 0s.

Question 9: Brute Force Attack on DES

Answer is approximately $\frac{255}{256}$
Consider the probability that $k \neq k'$ which is
$Pr(k \neq k' : e(k', p) = e(k, p))$ = the probability the collision will happens= $\frac{2^{56}-1}{2^{64}} \approx \frac{1}{2^8}$
Thus, the probability $k = k'$ such that $e(k', p) = e(k, p)$ is approximately$1 - \frac{1}{2^8} = \frac{255}{256}$

Question 10: Block-Cipher Modes of Operation

i) C_0 is the initialize vector(IV). This is the block that will be XOR with the first block of the plain text before encryption. The subsequent blocks of plain text will then xor with the ciphered text block before encryption.

ii)

It's not secure because the IV is not encrypted. Once, the IV is known, every AES block can be figured out which is equivalent to AES ECB. AES ECB is insecure because by using the same key, identical plain text will encrypt to identical cipher text. On top of that, this is a PoS terminal system, therefore, the special formats for different credit cards are known. This decreases the security even more. A typical attack can be done simply by making a transaction at the target PoS and gather the transmitted data, then figured the key out.

iii)

Since, the IV now is encrypted along with AES-128 which is proved to be secure and fixed the block-chain problem by encrypting the xor result of C and plain text.

Question 11: CBC Integrity Vulnerability

The technique is as followed. Find a modified IV such that modified IV xor the desired text is equivalent as the cipher key. Then, replace the original IV to the modified IV. This works because the decryption algorithm uses IV as a "translator" to start decrypting the cypher text.

Assuming the first block(b1710117cfe1cc5549bbb45f0bad1c8c) is the IV
Let orgtext = "4769766520506174726f6e2024313030"(100)
Let orgiv = "b1710117cfe1cc5549bbb45f0bad1c8c"
Let modtext = "4769766520506174726f6e2024393837"(987)
result = orgtext $\oplus$ orgiv
= f6187772efb1ad213bd4da7f2f9c2cbc
answer = result $\oplus$ modtext
= "b1710117cfe1cc5549bbb45f0ba5148b"
The final solution is b1710117cfe1cc5549bbb45f0ba5148b ae11fb7f135d0ac6591bf66facf651b7