CyberSecurity – HW#3

Sahand Khoshdel – 810196607

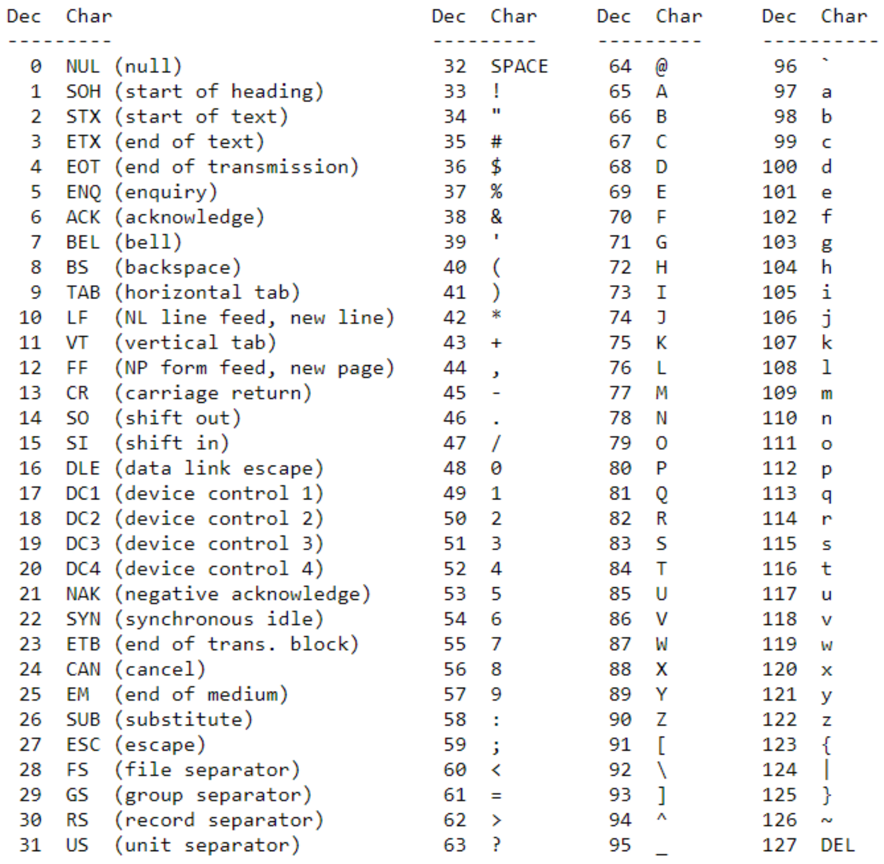
Analytical Assignment #1

Table of Contents

Q1) Radix 64 Encoding

The word “security’ contains 8 letters. Each one of them are mapped to an 8-bit ASCII code. So, a total of $8 \times 8 = 64$ bits are going to be converted in to 6-bit blocks which result in 10 complete blocks and one padded block (2 remainders). So, **we expect the final Radix-64 output to be 11 characters long**.

First, we should encode the word “security” in ASCII format according to the ASCII table shown below.



Dec	Char	Dec	Char	Dec	Char	Dec	Char
0	NUL (null)	32	SPACE	64	@	96	`
1	SOH (start of heading)	33	!	65	A	97	a
2	STX (start of text)	34	"	66	B	98	b
3	ETX (end of text)	35	#	67	C	99	c
4	EOT (end of transmission)	36	$	68	D	100	d
5	ENQ (enquiry)	37	%	69	E	101	e
6	ACK (acknowledge)	38	&	70	F	102	f
7	BEL (bell)	39	'	71	G	103	g
8	BS (backspace)	40	(	72	H	104	h
9	TAB (horizontal tab)	41	)	73	I	105	i
10	LF (NL line feed, new line)	42	*	74	J	106	j
11	VT (vertical tab)	43	+	75	K	107	k
12	FF (NP form feed, new page)	44	,	76	L	108	l
13	CR (carriage return)	45	-	77	M	109	m
14	SO (shift out)	46	.	78	N	110	n
15	SI (shift in)	47	/	79	O	111	o
16	DLE (data link escape)	48	0	80	P	112	p
17	DC1 (device control 1)	49	1	81	Q	113	q
18	DC2 (device control 2)	50	2	82	R	114	r
19	DC3 (device control 3)	51	3	83	S	115	s
20	DC4 (device control 4)	52	4	84	T	116	t
21	NAK (negative acknowledge)	53	5	85	U	117	u
22	SYN (synchronous idle)	54	6	86	V	118	v
23	ETB (end of trans. block)	55	7	87	W	119	w
24	CAN (cancel)	56	8	88	X	120	x
25	EM (end of medium)	57	9	89	Y	121	y
26	SUB (substitute)	58	:	90	Z	122	z
27	ESC (escape)	59	;	91	[	123	{
28	FS (file separator)	60	<	92	\	124	|
29	GS (group separator)	61	=	93	]	125	}
30	RS (record separator)	62	>	94	^	126	~
31	US (unit separator)	63	?	95	_	127	DEL

Each character in the word ‘security’ is mapped to a 8-bit binary value according to the table below:

Character	Decimal ASCII value	Binary ASCII value
s	115	01110011
e	101	01100101
c	99	01100011
u	117	01110101
r	114	01110010
i	105	01101001
t	116	01110100
y	121	01111001

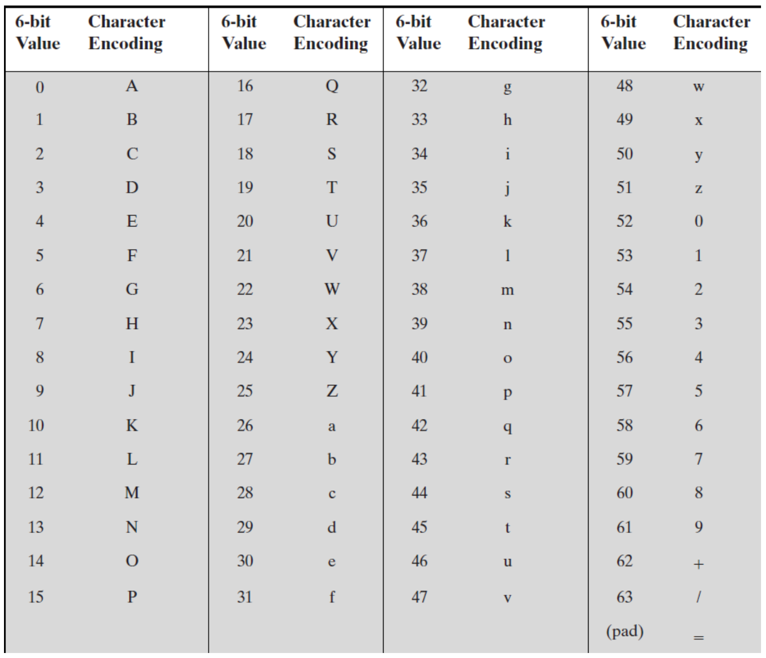
security = 01110011 01100101 01100011 01110101 01110010 01101001 01110100 01111001

For Radix-64 encoding the bitstream we've obtained should **be converted to groups of 6-bit blocks**, to represent 64 different output characters.

security = 011100 110110 010101 100011 011101 010111 001001 101001 011101 000111 1001(00)

the last 2 bits are zero paddings. We should show padding by the special character "=" it has in the radix-64 format.

Table 7.9 Radix-64 Encoding



6-bit Value	Character Encoding	6-bit Value	Character Encoding	6-bit Value	Character Encoding	6-bit Value	Character Encoding
0	A	16	Q	32	g	48	w
1	B	17	R	33	h	49	x
2	C	18	S	34	i	50	y
3	D	19	T	35	j	51	z
4	E	20	U	36	k	52	0
5	F	21	V	37	l	53	1
6	G	22	W	38	m	54	2
7	H	23	X	39	n	55	3
8	I	24	Y	40	o	56	4
9	J	25	Z	41	p	57	5
10	K	26	a	42	q	58	6
11	L	27	b	43	r	59	7
12	M	28	c	44	s	60	8
13	N	29	d	45	t	61	9
14	O	30	e	46	u	62	+
15	P	31	f	47	v	63	/
						(pad)	=

6-bit block	Decimal value	Radix-64 value
011100	28	c
110110	52	2
010101	21	V
100011	35	j
011101	29	d
010111	23	X
001001	9	J
101001	41	p
011101	29	d
000111	7	H
100100	36	k
-	"Padding"	=

The corresponding Radix-64 representation for "security" is "**c2VjdXJpdHk=**"

Q2) Client Authentication in SSL

Yes and No. :)

SSL will be **insecure** for the server cause impersonation can be **done but according to application layer authentication and the concept of "layer security transparency**", **Accurate Application layer authentication** for the client by means of the server, **can eliminate the need of a lower layer authentication.**

What happens in SSL is: the server can request the client to show a certificate but due to a lot of reasons (such as cracked Windows), it may not have a certificate to offer. If the client doesn't have a certificate **an attacker can impersonate himself as the client and even sign the previous messages in the finish phase**.

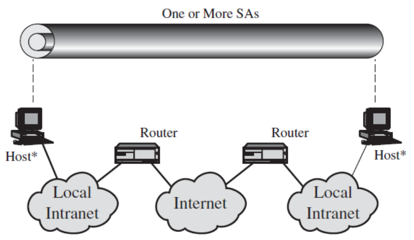
a) If the server supports DH as well as RSA:
   If both RSA and DH are supported by the server **the attacker can impersonate himself as the client and choose DH so SSL security is ruined.** But it doesn't matter if we are authenticating the client with a password in the application layer which we assume the attacker doesn't have it.

b) If the server doesn't support DH:
   SSL provides complete security cause the attacker can't impersonate due to the secure channel against MITM attacks, the authentication can be done in the finish phase while signing previous messages.

Q3) Implementing IP-Sec

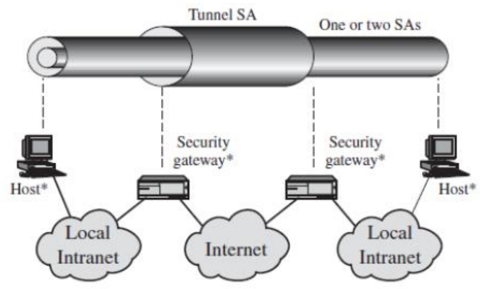
A) Both entities and their gateway routers support IP-Sec:

In this case we have complete security end to end. Both tunnel and transport modes can be used. **AH and ESP** can be used **separately** in **transport** mode.



(a) Case 1

We can also combine them. In this case, an **ESP SA can be inside an AH SA in transport mode** and **either of them can be inside one another in the tunnel mode.** But ESP inside AH is recommended as mentioned in the next problem.



B) Both gateway routers but only one of the entities support IP-Sec:

Security isn't provided within the LAN of the host that doesn't support IP-Sec, but it is provided elsewhere. The tunnel mode is used between the routers and the it can support both ESP and AH

C) Both gateway routers but neither the entities support IP-Sec:

Security is provided on the Internet, but not on either of LAN's. This is a representative of a VPN connection where only a tunnel SA between the routers is required. AS mentioned in the previous part it can support both ESP and AH.

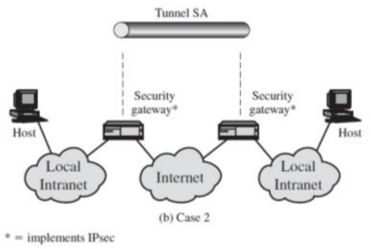


Figure 8.10 Basic Combinations of Security Associations

D) Does IP-Sec implementation mode depend on whether we're going to use AH or ESP?

No. Either SA's can be used for each mode. The only thing we need to note is that if the IP-Sec implementation is host to host only tunnel mode can be used. If not, we're free to use both.

Q4) Combined Implementation of AH & ESP

For providing both confidentiality and integrity we can use ESP and AH beside each other, but the reason that it's better to use ESP first is that the **authentication is applied on both the original IP header and the IP payload,** in the other scenario where AH is applied first it will only be applied on the payload so the header can be vulnerable to impersonation despite being encrypted by ESP.

Q5) Downsides if using Web Proxies for Web Traffic Analysis

The problem rises due to the fact that not all **what is considered as "web traffic" is sent on HTTP and HTTPS ports.** So, devices such as web proxies which are **port sensitive can't actually identify the whole web traffic** easily. Thus, we can't rely on them to block or access all intended web servers. Setting proxies on a mode to listen to various number of ports can result in complexity for traffic analysis and is time inefficient as well.

For better web traffic analysis some concept is introduced known as **Traffic Classification** which is the **process of separating traffic according to the other various features instead of only the port number**.

The features contain the **application payload, temporal characteristics, packet size, etc.** ML techniques such as SVM's and Bayesian analysis are usually used to train a classifier for separating different traffics such as web traffic, for further investigation.

Q6) Firewall Planning

a) **Organization webpage should be accessed** from **both Internet and LAN users**. But these users should be separated from one another.

   So, the **external firewall** should allow **all input traffic to the LAN**, with the **destination IP of the webpage and the destination port of 80/443** representing HTTP/HTTPS. They should **also allow responses form the web server.**

   The **internal firewall** should allow **all output traffic form the LAN** with the previous conditions (**destination port and IP**)

   In addition to the mentioned rules, **DNS servers between the firewalls should also be accessed** in order to load the webpage for the internal and external users. **DNS replies sent from the DNS servers** should also be allowed by both firewalls.

b) **LAN users should be able to send emails via SMTP but only be able to receive them via outlook (POP3)**

   SMTP traffic should be allowed to exit if sent by LAN users and **Acknowledgements from the webserver** should be allowed to enter by the Internal firewall. Emails form the email server should only be allowed if the port is POP3.

Internal Firewall:

Src. Address	Dst. Adress	Src. Port	Dst. Port	Comment	Action
Any	Web server		80/443		A
Web server	Any	80/443			A
Any	DNS server	53	53		A
DNS server	Any	53	53		A
Any	Email server		587		
Email server	Any				A
Email server	Any		POP3		A
Any	Any				D

External Firewall:

Src. Address	Dst. Adress	Src. Port	Dst. Port	Comment	Action
Any	Web server		80/443		A
Any	DNS server	53	53		A
Web server	Any	80/443			A
DNS server	Any	53	53		A
Email server	Any		587		
Any	Email server				A|ack flag
Any	Email server		587		A
Any	Any				D

Q7) WEP

a)

CRC is a linear operation because it's **obtained by dividing the data** which is high degree polynomial with binary coefficients (showing presence of 0 and 1 bits respectively) **by a fixed degree polynomial** (CRC polynomial)

Now if a **data changes,** in some sense, it means **we have actually added a polynomial with binary coefficients to the old data**, and as addition of a binary stream is **equivalent to an XOR operation**, we can find the CRC for the new data according to the CRC of the old data, **which shows linearity.**

b)

In case we receive three packets with the same IV, the keystreams generated using those IVs would be identical. Therefore:

$$c_1 = (m_1|CRC(m_1)) \oplus keystream$$
$$c_2 = (m_2|CRC(m_2)) \oplus keystream$$
$$c_3 = (m_3|CRC(m_3)) \oplus keystream$$

If we add up the encrypted packets:

$$c_1 \oplus c_2 \oplus c_3 = (m_1|CRC(m_1)) \oplus keystream \oplus (m_2|CRC(m_2)) \oplus keystream \oplus (m_3|CRC(m_3)) \oplus keystream$$

Adding the keystream 3 times is equivalent to adding it just one time, and due to the fact that XOR doesn't have expansion or permutation effects:

$$c_1 \oplus c_2 \oplus c_3 = (\,(m_1 \oplus m_2 \oplus m_3)\ |\ [CRC(m_1) \oplus CRC(m_2) \oplus CRC(m_3)]) \oplus keystream$$

And finally due to the unfortunate linearity of CRC:

$$c_1 \oplus c_2 \oplus c_3 = (\,(m_1 \oplus m_2 \oplus m_3)\ |\ [CRC(m_1 \oplus m_2 \oplus m_3)]) \oplus keystream$$

Now by creating a message such as $c_4 = c_1 \oplus c_2 \oplus c_3$, it is completely valid and thus we can add it to other messages and degrade integrity.

Q8) Open System Authentication Downsides

a) **Security problem:**

This system isn't tolerant against MAC spoofing. Access points don't authenticate the users correctly due to the fact that messages are sent plaintext and MAC addresses can easily be spoofed and turned into a valid address that isn't in the black list. Therefore, a user that is on the black list can impersonate himself as a valid user in the LAN. This attack can be classified as a sort of MITM attack on the open network.

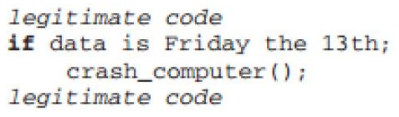
b) **Practical problem:**

Updating Blacklists are a problem especially where a lot of access points have to collaborate and share a uniform black list across the network, and that itself is done on plaintext!

The first problem is for LANs with a lot of users such as airports. Updating the list via broadcast messages among AP's are time inefficient.

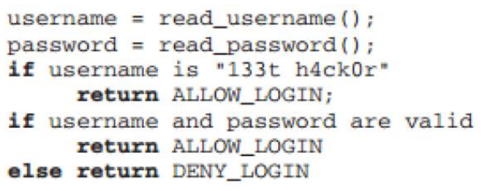
Moreover, an attacker can easily spoof an AP MAC address and broadcast a fake blacklist to other AP's.

Q9) Malware Detection



```
legitimate code
if data is Friday the 13th;
    crash_computer();
legitimate code
```

This code shows a malware called "**Logic Bomb**". Logic bombs are **inserted in software** and are activated if certain conditions are met. The conditions here are "date = **Friday 13'th**" which show the **activation date** of the logic bomb. A **payload or function** is executed when the condition are met. The payload is **crashing the host's computer** in this example.



```
username = read_username();
password = read_password();
if username is "133t h4ck0r"
     return ALLOW_LOGIN;
if username and password are valid
     return ALLOW_LOGIN
else return DENY_LOGIN
```

This code belongs to **a backdoor or back-trap**. The second if statement is the default user login procedure. But the first if statement allows a certain username which is the attacker to proceed without authentication and gives access to data.