FIT3031: Tutorial 7 Sample Solutions

EMAIL SECURITY

Q1 What are the five principal services provided by PGP?

Ans: Authentication, confidentiality, compression, e-mail compatibility, and segmentation

Q2 What is the utility of a detached signature?

Ans: A detached signature is useful in several contexts. A user may wish to maintain a separate signature log of all messages sent or received. A detached signature of an executable program can detect subsequent virus infection. Finally, detached signatures can be used when more than one party must sign a document, such as a legal contract. Each person's signature is independent and therefore is applied only to the document. Otherwise, signatures would have to be nested, with the second signer signing both the document and the first signature, and so on.

Q3 Why does PGP generate a signature before applying compression?

Ans: **a.** It is preferable to sign an uncompressed message so that one can store only the uncompressed message together with the signature for future verification. If one signed a compressed document, then it would be necessary either to store a compressed version of the message for later verification or to recompress the message when verification is required.

b. Even if one were willing to generate dynamically a recompressed message for verification, PGP's compression algorithm presents a difficulty. The algorithm is not deterministic; various implementations of the algorithm achieve different tradeoffs in running speed versus compression ratio and, as a result, produce different compressed forms. However, these different compression algorithms are interoperable because any version of the algorithm can correctly decompress the output of any other version. Applying the hash function and signature after compression would constrain all PGP implementations to the same version of the compression algorithm.

Q4 What is R64 conversion?

Ans: R64 converts a raw 8-bit binary stream to a stream of printable ASCII characters. Each group of three octets of binary data is mapped into four ASCII characters.

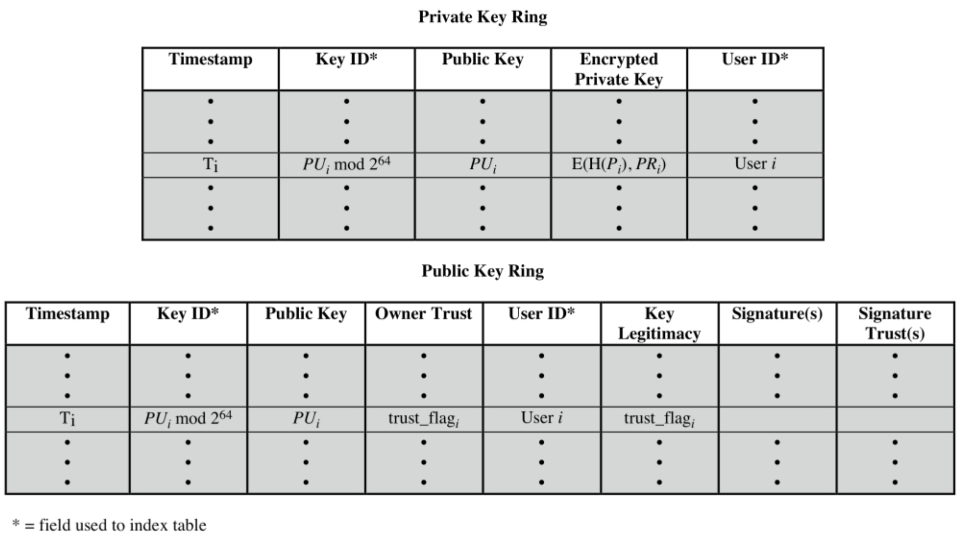
Q5 Why is R64 conversion useful for an e-mail application?

Ans: When PGP is used, at least part of the block to be transmitted is encrypted. If only the signature service is used, then the message digest is encrypted (with the sender's private key). If the confidentiality service is used, the message plus signature (if present) are encrypted (with a one-time symmetric key). Thus, part or all of the resulting block consists of a stream of arbitrary 8-bit octets. However, many electronic mail systems only permit the use of blocks consisting of ASCII text.

Q6 Why is the segmentation and reassembly function in PGP needed?

Ans: E-mail facilities often are restricted to a maximum message length.

Q7 How does PGP use the concept of trust?



Private Key Ring

Timestamp	Key ID*	Public Key	Encrypted Private Key	User ID*
⋮	⋮	⋮	⋮	⋮
T_i	$PU_i \bmod 2^{64}$	PU_i	$E(H(P_i), PR_i)$	User i
⋮	⋮	⋮	⋮	⋮

Public Key Ring

Timestamp	Key ID*	Public Key	Owner Trust	User ID*	Key Legitimacy	Signature(s)	Signature Trust(s)
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
T_i	$PU_i \bmod 2^{64}$	PU_i	$trust\_flag_i$	User i	$trust\_flag_i$		
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

* = field used to index table

Figure 7.4 General Structure of Private and Public Key Rings

Ans: PGP includes a facility for assigning a level of trust to individual signers and to keys. PGP computes a "key legitimacy field" for each public key certificate in the key ring. The higher the trust level, the higher the confidence the certificate is authentic. PGP uses two trust fields to maintain key legitimacy:

- signature trust field: indicates the degree to which the PGP user trusts the signer to certify public keys
- owner trust field: indicates the degree to which this public key is trusted to sign other public key certificates

Q8 What is MIME?

Ans: MIME is an extension to the RFC 822 framework that is intended to address some of the problems and limitations of the use of SMTP (Simple Mail Transfer Protocol) or some other mail transfer protocol and RFC 822 for electronic mail.

Q9 What is S/MIME? What are the cryptographic functions used in S/MIME?

Ans: S/MIME (Secure/Multipurpose Internet Mail Extension) is a security enhancement to the MIME Internet e-mail format standard, based on technology from RSA Data Security. S/MIME incorporates many algorithms for the various cryptographic functions it provides:

- The Digital Signature (DSS) for digital signature.
- S/MIME lists Diffie-Hellman (ElGamal) as the preferred algorithm for encrypting one-time session keys or RSA alternatively for encrypting messages.
- It uses 160 bit SHA-1 for creating Hash of the message for the Digital signature.
- For message encryption it supports triple DES, 40-bit RC2 using the one-time session key
- It creates a message authentication code using HMAC with SHA-1.

Q10 What is DKIM? How is the DKIM e-mail authentication service different when compared to S/MIME or PGP?

Ans: DomainKeys Identified Mail (DKIM) is a specification for cryptographically signing e-mail messages, permitting a signing domain to claim responsibility for a message in the mail stream.

DKIM e-mail authentication service is different when compared to S/MIME or PGP as indicated below:

1. S/MIME needs both sender and receiver to employ S/MIME. Most of the S/MIME mail users, bulk of the incoming mail does not use S/MIME.
2. S/MIME signs only the message contents. Header information maybe compromised.
3. DKIM is not implemented in client programs (MUAs) and is therefore transparent to the user; the user need not take any action.
4. DKIM applies to all mail from cooperating domains.
5. DKIM allows good senders to prove that they did send a particular message and prevent forgers from masquerading as good senders.

PROBLEMS

1. In Figure 7.4 given below, each entry in the public-key ring contains an Owner Trust field that indicates the degree of trust associated with this public-key owner. Why is that not enough? That is, if this owner is trusted and this is supposed to be the owner's public key, why is that trust not enough to permit PGP to use this public key?

Private Key Ring

Timestamp	Key ID*	Public Key	Encrypted Private Key	User ID*
• • •	• • •	• • •	• • •	• • •
Ti	$PU_i \bmod 2^{64}$	PU_i	$E(H(P_i), PR_i)$	User *i*
• • •	• • •	• • •	• • •	• • •

Public Key Ring

Timestamp	Key ID*	Public Key	Owner Trust	User ID*	Key Legitimacy	Signature(s)	Signature Trust(s)
• • •	• • •	• • •	• • •	• • •	• • •	• • •	• • •
Ti	$PU_i \bmod 2^{64}$	PU_i	trust_flag_i	User *i*	trust_flag_i		
• • •	• • •	• • •	• • •	• • •	• • •	• • •	• • •

* = field used to index table

Figure 7.4 General Structure of Private and Public Key Rings

Ans: We trust this owner, but that does not necessarily mean that we can trust that we are in possession of that owner's public key.

2. What is the basic difference between X.509 and PGP in terms of key hierarchies and key trust?

Ans: In X.509 there is a hierarchy of Certificate Authorities. Another difference is that in X.509 users will only trust Certificate Authorities while in PGP users can trust other users.

3. Phil Zimmermann chose IDEA, three-key triple DES, and CAST-128 as symmetric encryption algorithms for PGP. Give reasons why each of the following symmetric encryption algorithms described in this book is suitable or unsuitable for PGP:

a. DES,

Ans: DES is unsuitable because of its short key size.

b. two-key triple DES,

Ans: Two-key triple DES, which has a key length of 112 bits, is suitable.

and

c. AES.

Ans: AES is also suitable.

4. Consider radix-64 conversion, shown in Table 7.9, as a form of encryption. In this case, there is no key. But suppose that an opponent knew only that some form of substitution algorithm was being used to encrypt English text and did not guess that it was R64. How effective would this algorithm be against cryptanalysis?

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	/

Ans: It certainly provides more security than a monoalphabetic substitution. Because we are treating the plaintext as a string of bits and encrypting 6 bits at a time, we are not encrypting individual characters.

Therefore, the frequency information is lost, or at least significantly obscured.

5. Encode the text "plaintext" using the following techniques. Assume characters are stored in 8-bit ASCII with zero parity.

a. Radix-64

Ans: The first step is to convert the characters into 8-bit ASCII with zero parity.
Consulting the table on ASCII , we have the following correspondence:

p	01110000
l	01101100
a	01100001
i	01101001
n	01101110
t	01110100
e	01100101
x	01111000
t	01110100

Next, we block these off into groups of 6 bits, show the 6-bit decimal value, and
do the encoding.

011100	000110	110001	100001	011010	010110	111001	110100
28	6	49	33	26	22	57	52
c	G	x	h	a	W	5	0

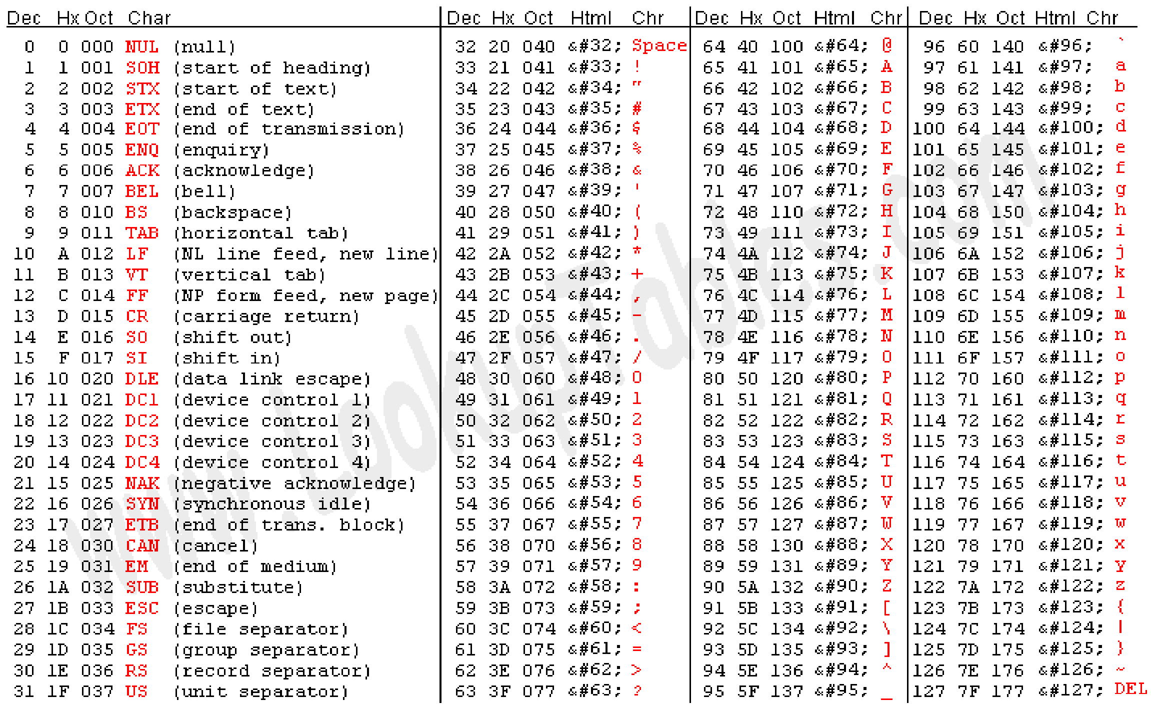
011001	010111	100001	110100
25	23	33	52
Z	X	h	0

So the radix-64 encoding is cGxhaW50ZXh0

b. Quoted-printable

Ans: All of the characters are "safe", so the quoted-printable encoding is simply plaintext

ASCII Table is given below for your reference.



Dec	Hx	Oct	Char		Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL	(null)	32	20	040	&#32;	Space	64	40	100	&#64;	@	96	60	140	&#96;	`
1	1	001	SOH	(start of heading)	33	21	041	&#33;	!	65	41	101	&#65;	A	97	61	141	&#97;	a
2	2	002	STX	(start of text)	34	22	042	&#34;	"	66	42	102	&#66;	B	98	62	142	&#98;	b
3	3	003	ETX	(end of text)	35	23	043	&#35;	#	67	43	103	&#67;	C	99	63	143	&#99;	c
4	4	004	EOT	(end of transmission)	36	24	044	&#36;	$	68	44	104	&#68;	D	100	64	144	&#100;	d
5	5	005	ENQ	(enquiry)	37	25	045	&#37;	%	69	45	105	&#69;	E	101	65	145	&#101;	e
6	6	006	ACK	(acknowledge)	38	26	046	&#38;	&	70	46	106	&#70;	F	102	66	146	&#102;	f
7	7	007	BEL	(bell)	39	27	047	&#39;	'	71	47	107	&#71;	G	103	67	147	&#103;	g
8	8	010	BS	(backspace)	40	28	050	&#40;	(	72	48	110	&#72;	H	104	68	150	&#104;	h
9	9	011	TAB	(horizontal tab)	41	29	051	&#41;	)	73	49	111	&#73;	I	105	69	151	&#105;	i
10	A	012	LF	(NL line feed, new line)	42	2A	052	&#42;	*	74	4A	112	&#74;	J	106	6A	152	&#106;	j
11	B	013	VT	(vertical tab)	43	2B	053	&#43;	+	75	4B	113	&#75;	K	107	6B	153	&#107;	k
12	C	014	FF	(NP form feed, new page)	44	2C	054	&#44;	,	76	4C	114	&#76;	L	108	6C	154	&#108;	l
13	D	015	CR	(carriage return)	45	2D	055	&#45;	-	77	4D	115	&#77;	M	109	6D	155	&#109;	m
14	E	016	SO	(shift out)	46	2E	056	&#46;	.	78	4E	116	&#78;	N	110	6E	156	&#110;	n
15	F	017	SI	(shift in)	47	2F	057	&#47;	/	79	4F	117	&#79;	O	111	6F	157	&#111;	o
16	10	020	DLE	(data link escape)	48	30	060	&#48;	0	80	50	120	&#80;	P	112	70	160	&#112;	p
17	11	021	DC1	(device control 1)	49	31	061	&#49;	1	81	51	121	&#81;	Q	113	71	161	&#113;	q
18	12	022	DC2	(device control 2)	50	32	062	&#50;	2	82	52	122	&#82;	R	114	72	162	&#114;	r
19	13	023	DC3	(device control 3)	51	33	063	&#51;	3	83	53	123	&#83;	S	115	73	163	&#115;	s
20	14	024	DC4	(device control 4)	52	34	064	&#52;	4	84	54	124	&#84;	T	116	74	164	&#116;	t
21	15	025	NAK	(negative acknowledge)	53	35	065	&#53;	5	85	55	125	&#85;	U	117	75	165	&#117;	u
22	16	026	SYN	(synchronous idle)	54	36	066	&#54;	6	86	56	126	&#86;	V	118	76	166	&#118;	v
23	17	027	ETB	(end of trans. block)	55	37	067	&#55;	7	87	57	127	&#87;	W	119	77	167	&#119;	w
24	18	030	CAN	(cancel)	56	38	070	&#56;	8	88	58	130	&#88;	X	120	78	170	&#120;	x
25	19	031	EM	(end of medium)	57	39	071	&#57;	9	89	59	131	&#89;	Y	121	79	171	&#121;	y
26	1A	032	SUB	(substitute)	58	3A	072	&#58;	:	90	5A	132	&#90;	Z	122	7A	172	&#122;	z
27	1B	033	ESC	(escape)	59	3B	073	&#59;	;	91	5B	133	&#91;	[	123	7B	173	&#123;	{
28	1C	034	FS	(file separator)	60	3C	074	&#60;	<	92	5C	134	&#92;	\	124	7C	174	&#124;	|
29	1D	035	GS	(group separator)	61	3D	075	&#61;	=	93	5D	135	&#93;	]	125	7D	175	&#125;	}
30	1E	036	RS	(record separator)	62	3E	076	&#62;	>	94	5E	136	&#94;	^	126	7E	176	&#126;	~
31	1F	037	US	(unit separator)	63	3F	077	&#63;	?	95	5F	137	&#95;	_	127	7F	177	&#127;	DEL

Source: www.LookupTables.com