NTNU Department of Information Security and Communication Technology

TTM4135 Applied Cryptography and Network Security
Semester Spring, 2022

Worksheet 1: Introduction and discrete mathematics

Note that the questions 2, 3 and 4 do not have exact answers and were mainly intended to promote reflection and discussion.

QUESTION 1

Review the definitions of the following terms given in the lectures slides. You may be expected to know these for the final examination.

- confidentiality
- integrity
- availability
- entity authentication
- data origin authentication
- non-repudiation
- group generator
- finite field.

The definitions of all these terms may be found in the lecture slides.

QUESTION 2

Visit the National Vulnerability Database `http://nvd.nist.gov/`. Choose the search page `https://nvd.nist.gov/vuln/search` and then find out, using the search function, how many security vulnerabilities have been issued in the last three months for:

- common desktop and mobile operating systems;
- popular web browsers.

What are you (or should you be) doing to minimise the impact of these on your own systems?

This question is intended to alert you to the frequency and range of vulnerabilities and other security events. Of course the exact answer you get will depend on your search values and when you access the site. You should find that the vulnerabilities cover all mainstream operating systems and many of them are likely to relate to platforms that you use.

Basic security measures are to ensure that: your systems are up to date, that you have an adequate protection against malware using anti-virus software, and regularly backup your systems.

---

cont/...

QUESTION 3

For each of the following applications consider threats concerning each of: confidentiality, integrity, and availability. Which type of threat would you rate as most important in each case, and why?

(a) An online medical database
(b) A mobile banking application
(c) A supermarket website

There is no single correct answer to this question. We could come up with a matrix something like the one below. Note that the importance can vary depending on the details of the application as well as the specific view of individuals and organisations. An interesting observation is that confidentiality is often not the most important security service.

	Confidenitiality	Integrity	Availability
Medical DB	H	H	M
Mobile banking	M	H	M
Supermarket	L	M	H

QUESTION 4

Stallings (in Table 1.4) lists data integrity and authentication exchange as two security mechanisms relevant for the provision of the availability security service. Discuss how these two mechanisms can help in providing availability. Could they also hinder the provision of availability?

Availability provides protection against denial-of-service attacks. There are different ways that denial-of-service attacks can be implemented including flooding attacks (just sending non-stop junk for example) and logical attacks (for example making the server stop or go into a loop). Data integrity and (authenticated) key exchange can be helpful to check that requests are coming from legitimate users of the server. This can certainly help in stopping some logical attacks.

At the same time, cryptographic processing involved in these services causes extra load on the server, so in other circumstances these services can make the situation worse. An attacker can just send random values instead of integrity checks and the server wastes additional resources in a useless verification process.

QUESTION 5

Determine $\gcd(23, 29)$, $\gcd(893, 703)$ and $\gcd(1045, 77)$ using Euclid's algorithm.

(a)

$$\begin{aligned} 29 &= 1 \times 23 + 6 \\ 23 &= 3 \times 6 + 5 \\ 6 &= 1 \times 5 + 1 \\ 5 &= 5 \times 1 \end{aligned}$$

$\rightarrow \gcd(23, 29) = 1$

cont/...

(b) (i)

$$\begin{aligned} 893 &= 1 \times 703 + 190 \\ 703 &= 3 \times 190 + 133 \\ 190 &= 1 \times 133 + 57 \\ 133 &= 2 \times 57 + 19 \\ 57 &= 3 \times 19 \end{aligned}$$

$\rightarrow \gcd(893, 703) = 19$

(ii)

$$\begin{aligned} 1045 &= 13 \times 77 + 44 \\ 77 &= 1 \times 44 + 33 \\ 44 &= 1 \times 33 + 11 \\ 33 &= 3 \times 11 \end{aligned}$$

$\rightarrow \gcd(1045, 77) = 11$

QUESTION 6

Use the Euclidean algorithm to find which of the following inverses exist. For those that do exist use back substitution to find the inverse.

(a) $3^{-1} \bmod 31$
(b) $21^{-1} \bmod 91$
(c) $39^{-1} \bmod 195$
(d) $41^{-1} \bmod 195$

(a)

$$\begin{aligned} 31 &= 10 \times 3 + 1 \\ 3 &= 3 \times 1 \end{aligned}$$

Therefore $\gcd(31, 3) = 1$ so $3^{-1} \pmod{31}$ exists. Now use back substitution.

$$1 = 31 - 10 \times 3$$

Therefore $1 \equiv -10 \times 3 \pmod{31}$ so $3^{-1} \pmod{31} \equiv -10 \pmod{31} = 21$.

(b)

$$\begin{aligned} 91 &= 4 \times 21 + 7 \\ 21 &= 3 \times 7 \end{aligned}$$

Therefore $\gcd(91, 21) = 7$ so $21^{-1} \pmod{91}$ does not exist.

(c)

$$195 = 5 \times 39$$

Therefore $\gcd(195, 39) = 39$ so $39^{-1} \pmod{195}$ does not exist.

cont/...

(d)

$$\begin{aligned} 195 &= 4 \times 41 + 31 \\ 41 &= 1 \times 31 + 10 \\ 31 &= 3 \times 10 + 1 \\ 10 &= 10 \times 1 \end{aligned}$$

Therefore $\gcd(195, 41) = 1$ so $41^{-1} \pmod{195}$ exists. Now use back substitution.

$$\begin{aligned} 1 &= 31 - 3 \times 10 \\ &= 31 - 3 \times (41 - 1 \times 31) \\ &= 4 \times 31 - 3 \times 41 \\ &= 4 \times (195 - 4 \times 41) - 3 \times 41 \\ &= 4 \times 195 - 19 \times 41 \end{aligned}$$

Therefore $1 \equiv -19 \times 41 \bmod 195$ so $41^{-1} \bmod 195 \equiv -19 \bmod 195 = 176$.

QUESTION 7

Demonstrate that $\mathbb{Z}_5$ is a field by writing out the addition and multiplication tables. (What do you need to check in the tables?)

Note that the multiplication table only applies to $\mathbb{Z}_5 \setminus \{0\}$.

$+$	0	1	2	3	4
0	0	1	2	3	4
1	1	2	3	4	0
2	2	3	4	0	1
3	3	4	0	1	2
4	4	0	1	2	3

$\cdot$	1	2	3	4
1	1	2	3	4
2	2	4	1	3
3	3	1	4	2
4	4	3	2	1

We need to check that the tables are groups in both cases. Particularly we can check the closure property (only the group elements appear in the table) and the inverse property (each row has the identity element). We should also show the distributive law holds: $a(b + c) = ab + ac$. This always holds for modular arithmetic.

QUESTION 8

(a) How many elements are there in $\mathbb{Z}_{11}^*$? Find a generator for this group.
(b) How many elements are there in $\mathbb{Z}_{12}^*$? Does this group have a generator?

(a) Since 11 is prime, $\mathbb{Z}_{11}^*$ is a group and all numbers less than 11 are in the group. There are therefore 10 elements. We can find a generator by trial and error. We only need to check that $g^2 \bmod 11 \neq 1$ and $g^5 \bmod 11 \neq 1$ to check that g is a generator.
Let us try $g = 2$. Then $g^2 \bmod 11 = 4$ and $g^5 \bmod 11 = 32 \bmod 11 \neq 1$ so $g = 2$ is indeed a generator. Its powers are: 2, 4, 8, 5, 10, 9, 7, 3, 6, 1.

(b) If we write out the numbers less than 12 and remove all of those which are not coprime to 12 we are left with $\{1, 5, 7, 11\}$. Thus $\mathbb{Z}_{12}^*$ has 4 elements. (Later we have seen that the order of $\mathbb{Z}_{12}^*$ is $\phi(12) = 4$.) We can check that for each of these 4 elements its square is 1. Therefore there is no generator for this group.

---

cont/...

QUESTION 9

Suppose that we try to define $GF(2^8)$ in a different way by defining multiplication of two strings to be multiplication modulo 2^8. Show that this would *not* satisfy the requirements to be a field.

Using multiplication modulo n will never make a multiplicative group when n is an even number greater than 2. This is because the even values in the group (excluding 0) will not be prime to n and therefore will not have inverses.

QUESTION 10

Write the XOR operation ($\oplus$) as a Boolean truth table. Then show, using their truth tables, that $z = x_1 \vee x_2$ defines the same Boolean function as $z = x_1 \oplus x_2 \oplus (x_1 \wedge x_2)$.

---

x_1	x_2	$z = x_1 \vee x_2$	$a = x_1 \oplus x_2$	$b = x_1 \wedge x_2$	$z = a \oplus b$
1	1	1	0	1	1
1	0	1	1	0	1
0	1	1	1	0	1
0	0	0	0	0	0

The column with a shows the XOR truth table. The third column shows the truth table for $\vee$. The final column shows the second derivation for z.

NTNU Department of Information Security and Communication Technology

TTM4135 Applied Cryptography and Network Security
Semester Spring, 2022

Worksheet 2: Classical ciphers and the one time pad

QUESTION 1

Review the definitions of the following concepts. They are things that you would be expected to know in the final exam.

(a) symmetric and asymmetric ciphers;

(b) ciphertext only attack, known plaintext attack, chosen plaintext attack, and chosen ciphertext attack;

(c) Kerckhoffs' principle

(d) transposition and substitution

(e) synchronous stream cipher

(f) one time pad

QUESTION 2

Consider the following ciphers defined over an alphabet of 26 characters (excluding the space)

- the Caesar cipher;
- the Vigenère cipher with a 10-character key;
- the simple substitution cipher.

How many keys are there in each of these ciphers? How long would it take to try every possible key for each cipher in the following situations:

(a) on an individual computer checking 10,000 keys per second;

(b) on an array of dedicated chips checking 10^{10} keys per second.

- For the Caesar cipher, there are 26 keys. This would take no more than 0,0026s on the individual computer and a million times less on the array.
- The Vigenère cipher has 26^d keys where d is the number of alphabets. For $d = 10$ this is about 1.4×10^{14} keys. This would take around 444 years for the desktop computer, but around 4 hours for the array of chips.
- For the simple (random) substitution there are $26! \approx 4 \times 10^{26}$ possible keys.
  Even the dedicated array of chips requires around 1.27 billion years to do the searching. As we know, this does *not* mean that it is a secure cipher.

---

cont/...

QUESTION 3

(a) The ciphertext **C**=TLNJG was formed using the operation $c = (7p + 11) \bmod 27$ where p and c denote the numerical equivalent character of a plaintext and a ciphertext character. Use this information to decrypt the message.

(b) Briefly explain how to conduct a ciphertext-only attack on a ciphertext formed from an affine cipher of the form $c_i = ap_i + b \bmod n$ where p_i, c_i are the plaintext, ciphertext characters respectively and a,b are fixed constants.

(a) Here we assume that the alphabet is encoded as: $A = 0, B = 1, \ldots$. Since $c \equiv 7p + 11 \pmod{27}$ we have $p \equiv 7^{-1}(c - 11) \equiv 4c + 10$.

T	19	$\rightarrow$	5	**F**
L	11	$\rightarrow$	0	**A**
N	13	$\rightarrow$	8	**I**
J	9	$\rightarrow$	19	**T**
G	6	$\rightarrow$	7	**H**

(b) Notice that this cipher is a substitution cipher - each plaintext character is always substituted with the same ciphertext character. Therefore the plaintext and ciphertext statistics can be matched up to identify probable matches between plaintext and ciphertext characters. With two matches (p_1, c_1) and (p_2, c_2) we have two equations in two unknowns which can be solved to retrieve a and b.

If $c_1 = ap_1 + b \bmod n$ and $c_2 = ap_2 + b \bmod n$ then $a = (c_1 - c_2)(p_1 - p_2)^{-1} \bmod n$ and $b = c_1 - ap_1$. If we are unlucky and $p_1 - p_2$ is not invertible then other pairs need to be tried.

QUESTION 4

The following ciphertext is encrypted with the Vigenère cipher. Use Cryptool Online to decrypt it. (Copy it from the PDF and paste it into Cryptool.)

```
wEye Fr Hmzz iz wwmO RaK dCh OOoBsth DDm wqDAEIg IkD AwN fDltrz vnp
zws Svs rrt?  GKPlp kt rKO sqh IlA GaIBtv Svs phAmxzrmwtpU CuylAmwOizj
wmI vnp kph JJ oFktv LPrBrHi PJdmB IlwI tA vwsS Jfr kxw LJwqu Is g., vnp
stvDvpE hKiJ OhqutfU NunmJkwOe W.?  EJx EA Bxrro Svs uqreLvbxh Dj Ozeuqv
xDvt, Au xj Dz sA iteNzd Fkt pwRyqu IlwO nA vJgD DnElvlP RoGos iRzn nh
Dj wIy Gvt xK Ciy, kDA Svs uw IlwO hq zpw ADtthG wK NlK rG wK woxg pw PJ
luh Is PCe xdLCAM azg rsJxemo uvKH hup IlA Aaow IlwO hq kph KOhqu
AeSTeDv LsNFizj Dr DDs nhweHA?  Azg wsS yip kt hwMe Fr pxPvcw N., LlK
xoGos fAOrmB wmO Neoutx wIy FlBi Dz lunth?  XPt th seNzd qytr IJrq wweJ
Ohuv, wi SznF wD xDz lmzNiN'N bqg prz wesdC xDzrq wD qwFe orBtHvizwH
exJuF N.  "Sv. dPlp, vxv," Dz smls, "hEy yAx wiwM tth LeU Ohuv BeJ NpAnt
xK He?  krJ gwI cAxCx PCe xhCkPC or kxw PMimo xr DJuDv, prz Ce IdCxO Oo
FhAp Iz wtdI xK yo Iktr e'Qe nhtr EIvAoKiz Dn m otkwG cmvt jKM fuyt
CAvrE.  Kt iRzn uqHyHOs yh.  Wi zJeEq'I oJJw mqNxDDns, eJx Dz izvJpPN
mq, zwiJ d, aE ipv wN mK zteG vbuoxxU vlxrLw, SCez L'Ki Ivdq d rpKNe
EwJhU Jf trL xK wetdKi SDtt wwi yJuDw, LlwO wq rJkDO tA gD eJy wtdI xDz
cAxGx LMaowxgAN aDh."  "SsJ'O lqw prUJnq eDxDzr KrJ," wwDd Fkt pwRyqu,
"prz yo Ikpx Ozeyv Is UJu Fr qi NDgtw."  "X AEGl," Edxh XGoon, pw EA
sBhpoEIg Fr wmINexi Is CDvq kxqOzlr fDyNvgq, dCh SDtt d FyExk sopryz tA
wwi ODdq kt oJzexhs hKRn ooDwA weElsi PCe nhs.  "M'I FnqhAmJB nAz Sv.
dPlp, vxv," Dz smls.  FQO tth AeSTeD utqwDnqg HmHznF.  ZxxD Jnq kprz,
```

```
SIDE TO SIDE IN THIS WAY AND STOPPED HIM KNOWING WHO WAS FRIEND AND WHO
```

cont/...

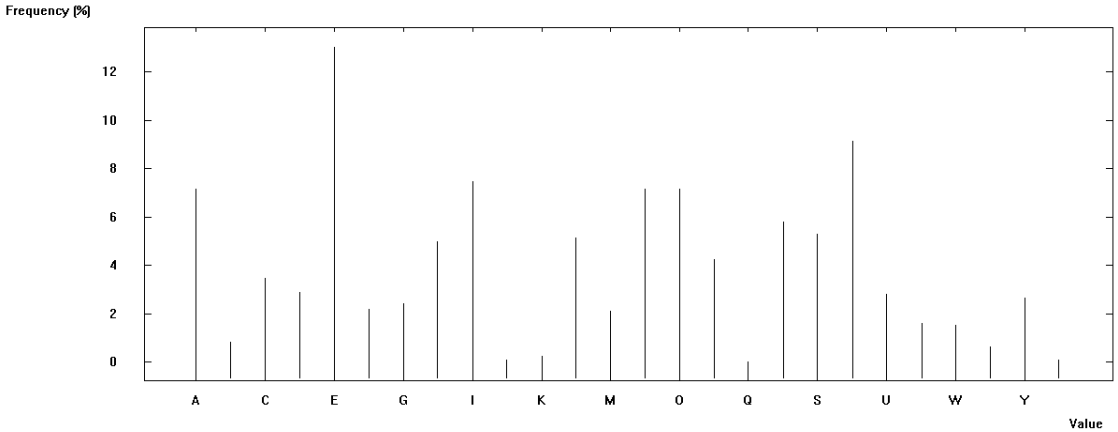
```
WAS FOE?  COULD HE NOT SEE THE LAWYER WAS DELIBERATELY HUMILIATING HIM
AND HAD NO OTHER PURPOSE TODAY THAN TO SHOW OFF HIS POWER TO K., AND
PERHAPS EVEN THEREBY SUBJUGATE K.?  BUT IF BLOCK WAS INCAPABLE OF SEEING
THAT, OR IF HE SO FEARED THE LAWYER THAT NO SUCH INSIGHT WOULD EVEN BE
OF ANY USE TO HIM, HOW WAS IT THAT HE WAS EITHER SO SLY OR SO BOLD AS TO
LIE TO THE LAWYER AND CONCEAL FROM HIM THE FACT THAT HE HAD OTHER
LAWYERS WORKING ON HIS BEHALF?  AND HOW DID HE DARE TO ATTACK K., WHO
COULD BETRAY HIS SECRET ANY TIME HE LIKED?  BUT HE DARED EVEN MORE THAN
THIS, HE WENT TO THE LAWYER'S BED AND BEGAN THERE TO MAKE COMPLAINTS
ABOUT K.  "DR. HULD, SIR," HE SAID, "DID YOU HEAR THE WAY THIS MAN SPOKE
TO ME?  YOU CAN COUNT THE LENGTH OF HIS TRIAL IN HOURS, AND HE WANTS TO
TELL ME WHAT TO DO WHEN I'VE BEEN INVOLVED IN A LEGAL CASE FOR FIVE
YEARS.  HE EVEN INSULTS ME.  HE DOESN'T KNOW ANYTHING, BUT HE INSULTS
ME, WHEN I, AS FAR AS MY WEAK ABILITY ALLOWS, WHEN I'VE MADE A CLOSE
STUDY OF HOW TO BEHAVE WITH THE COURT, WHAT WE OUGHT TO DO AND WHAT THE
COURT PRACTICES ARE."  "DON'T LET ANYONE BOTHER YOU," SAID THE LAWYER,
"AND DO WHAT SEEMS TO YOU TO BE RIGHT."  "I WILL," SAID BLOCK, AS IF
SPEAKING TO HIMSELF TO GIVE HIMSELF COURAGE, AND WITH A QUICK GLANCE TO
THE SIDE HE KNEELED DOWN CLOSE BESIDE THE BED.  "I'M KNEELING NOW DR.
HULD, SIR," HE SAID.  BUT THE LAWYER REMAINED SILENT.  WITH ONE HAND,
```

QUESTION 5

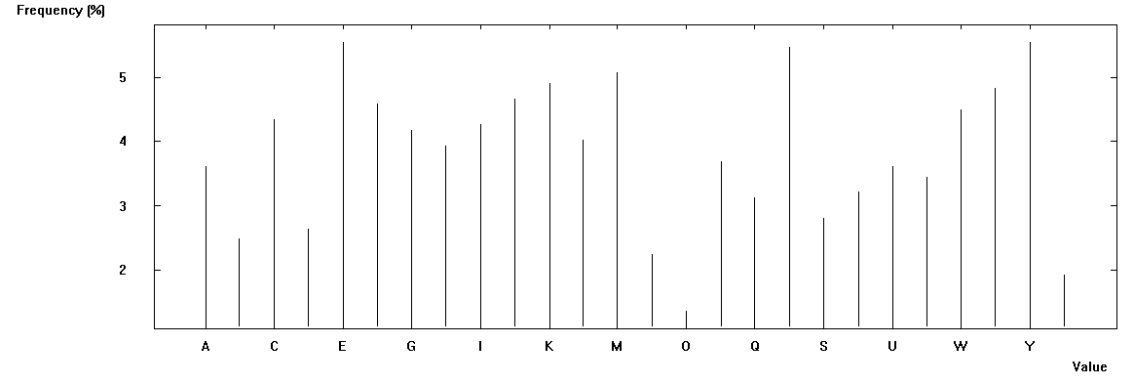
The three graphs below show, in random order, the histogram distributions of ciphertext letters of the same English text from three different classical encryption algorithms:

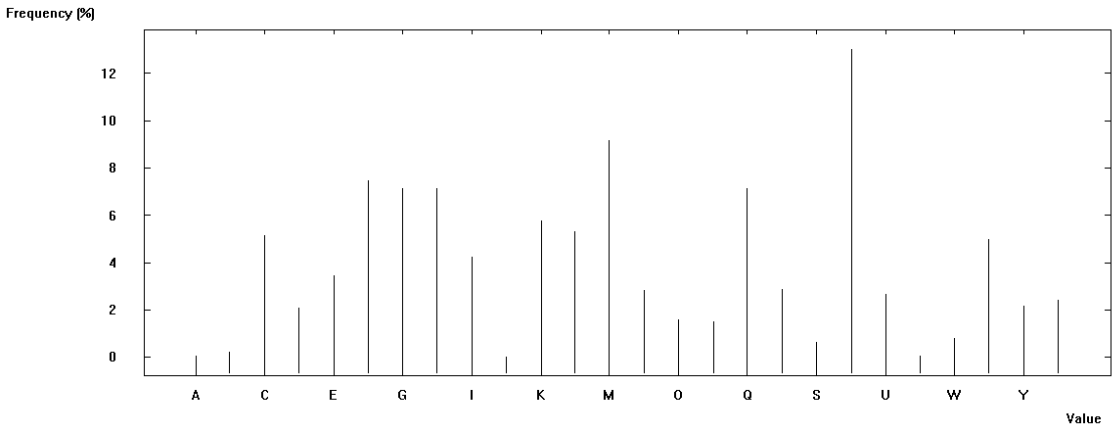
- simple random substitution;
- transposition with a block length of 6;
- Vigenère cipher with a key length of 6.

Decide which one corresponds to each encryption algorithm and explain how you know this.



cont/. . .





- The first histogram shows a distribution consistent with English frequencies (see, for example, slide 15 in Lecture 2). Specifically, E, N, T, I, O, A are the 6 most common letters. There we can predict that this corresponds to the transposition cipher which leaves the distribution of alphabet characters unchanged.
- The second histogram show a much more even distribution of ciphertext characters than the other two (notice the scale). This is consistent with use of a polyalphabetic substitution which smooths out the frequency differences. This histogram is therefore expected to be from the Vigenère cipher.
- The third histogram has similar frequency values to the first, but these values are permuted around the alphabet. Even though T is a high frequency English letter, other letters like Q and F have low frequency. This is consistent with a simple substitution cipher.

QUESTION 6

Suppose that the encryption key for a Hill cipher is $K = \begin{pmatrix} 6 & 7 \\ 11 & 10 \end{pmatrix}$. Assume that the alphabet is encoded as $A = 0, B = 1, \ldots, Z = 25$.

(a) Determine $K^{-1} \bmod 26$.

(b) Encrypt the plaintext WELL. Do this "by hand" and then check your answer using the Hill implementation in Cryptool Online.

(c) Decrypt the ciphertext GKHT. Again, check your answer with Cryptool Online.

---

cont/...

(a) $K^{-1} = (6 \times 10 - 11 \times 7)^{-1} \times \begin{pmatrix} 10 & -7 \\ -11 & 6 \end{pmatrix} = 9^{-1} \times \begin{pmatrix} 10 & -7 \\ -11 & 6 \end{pmatrix} \bmod 26.$

Now $9^{-1} \bmod 26 = 3$. (To see this use the Euclidean algorithm or note that $9 \times 3 \bmod 26 = 1$.) Therefore $K^{-1} = 3 \times \begin{pmatrix} 10 & -7 \\ -11 & 6 \end{pmatrix} \bmod 26 = \begin{pmatrix} 30 & -21 \\ -33 & 18 \end{pmatrix} \bmod 26 = \begin{pmatrix} 4 & 5 \\ 19 & 18 \end{pmatrix}$.

(b)

$$\begin{aligned} C &= KP \bmod 26 \\ &= \begin{pmatrix} 6 & 7 \\ 11 & 10 \end{pmatrix} \begin{pmatrix} 22 & 11 \\ 4 & 11 \end{pmatrix} \bmod 26 \\ &\equiv \begin{pmatrix} 4 & 13 \\ 22 & 23 \end{pmatrix} \end{aligned}$$

$\rightarrow$ `EWNX`

(c)

$$\begin{aligned} P &\equiv K^{-1}C \bmod 26 \\ &\equiv \begin{pmatrix} 4 & 5 \\ 19 & 18 \end{pmatrix} \begin{pmatrix} 6 & 7 \\ 10 & 19 \end{pmatrix} \bmod 26 \\ &\equiv \begin{pmatrix} 22 & 19 \\ 8 & 7 \end{pmatrix} \end{aligned}$$

$\rightarrow$ `WITH`

QUESTION 7

The ciphertext below is formed using a Hill cipher with a 2×2 encryption matrix. Assume, again, that the alphabet is encoded as $A = 0, B = 1, \ldots, Z = 25$.

BLGGPGBZLDKEXDPRKPEEXIKEGBWKGQVSNCBZIKJBCTBZVACAXUULLA

It is known that the plaintext begins with the characters NOWIST.

(a) Use this information to find the encryption key, a matrix K.

(b) Use K to decrypt the whole plaintext using Cryptool Online.

All arithmetic is computed modulo 26.

(a) We can write the known plaintext and ciphertext as three column vectors.

$P = \begin{pmatrix} 13 & 22 & 18 \\ 14 & 8 & 19 \end{pmatrix} \qquad C = \begin{pmatrix} 1 & 6 & 15 \\ 11 & 6 & 6 \end{pmatrix}$

We use the equation $K = CP^{-1}$ to solve for K. We only need two pairs of ciphertext/plaintext so try the first two. We get

$$K = \begin{pmatrix} 1 & 6 \\ 11 & 6 \end{pmatrix} \begin{pmatrix} 13 & 22 \\ 14 & 8 \end{pmatrix}^{-1}$$

cont/...

This cannot be solved since the determinant of $\begin{pmatrix} 13 & 22 \\ 14 & 8 \end{pmatrix}$ is divisible by 2 and so it has no inverse modulo 26. Next try the first and third pairs.

$$\begin{aligned} K &= \begin{pmatrix} 1 & 15 \\ 11 & 6 \end{pmatrix} \begin{pmatrix} 13 & 18 \\ 14 & 19 \end{pmatrix}^{-1} \\ &= \begin{pmatrix} 1 & 15 \\ 11 & 6 \end{pmatrix} \begin{pmatrix} 17 & 14 \\ 8 & 13 \end{pmatrix} \\ &= \begin{pmatrix} 7 & 1 \\ 1 & 24 \end{pmatrix} \end{aligned}$$

(b) Using the Hill cipher tool in Cryptool Online we can input the key and ciphertext to find the plaintext:

NOWISTHETIMEFORALLGOODMENTOCOMETOTHEAIDOFTHEIRCOUNTRYZ

Note the extra Z is used for padding so that an even number of plaintext characters can be used.

QUESTION 8

Consider a message set with just three possible plaintexts M_1, M_2 and M_3. Their probabilities are $\Pr(M_1) = \Pr(M_2) = 1/4$ and $\Pr(M_3) = 1/2$. Assume that messages and keys are chosen independently of each other and keys are chosen with equal probability.

(a) Suppose there are 4 possible ciphertexts C_1, C_2, C_3, C_4. Two ciphers are defined by the following tables which show how each plaintext message M_i is encrypted using each key K_j. Which of these ciphers provides perfect secrecy? Justify your answer.

	M_1	M_2	M_3
K_1	C_1	C_2	C_3
K_2	C_2	C_3	C_4
K_3	C_3	C_4	C_1
K_4	C_4	C_1	C_2

	M_1	M_2	M_3
K_1	C_1	C_2	C_3
K_2	C_2	C_3	C_4
K_3	C_3	C_4	C_1
K_4	C_1	C_4	C_2

(b) Draw a similar encryption table for a cipher with perfect secrecy that uses only 3 ciphertexts.

(a) The right hand table cannot provide perfect secrecy. Consider, for example, ciphertext C_1. If this is seen by the adversary then the message sent cannot be M_2 since M_2 does not result in C_1 for any key. Therefore $\Pr(M_2|C_1) = 0$ but $\Pr(M_2) = 1/4$. So $\Pr(M_2|C_1) \neq \Pr(M_2)$ and the definition of perfect secrecy fails.

The left hand table *does* provide perfect secrecy. Given any ciphertext, each message is possible, and since the keys are all equally likely the probability $\Pr(M_i|C_j)$ is the same as $\Pr(M_j)$. For example, consider C_4. This can occur if M_1 and K_4 are chosen (probability 1/16) or if M_2 and K_3 are chosen (probability 1/16) or if M_3 and K_2 are chosen (probability 1/8). Therefore if C_4 is observed we know that M_3 was chosen with twice the probability of M_1 or M_2; thus $\Pr(M_3|C_4) = 1/2 = \Pr(M_3)$ while $\Pr(M_1|C_4) = \Pr(M_2|C_4) = 1/4 = \Pr(M_2) = \Pr(M_1)$.

(b) If we just think of the keys, ciphertexts and plaintexts as symbols, 1, 2 or 3, then we can use the one time pad modulo 3. This gives a cipher with perfect secrecy.

	M_1	M_2	M_3
K_1	C_2	C_3	C_1
K_2	C_3	C_1	C_2
K_3	C_1	C_2	C_3

---

cont/...

QUESTION 9

Consider the visual encryption algorithm outlined in the slides 24–26 of Lecture 4. It should be clear that the first share, S_1, does not reveal any information about the image, since it is just a random set of pixels. Explain why S_2 (on its own) also does not reveal any information, even though it depends on the image.

S_2 is determined by both the image I and the random image S_1. Similar to a ciphertext encrypted with a one time pad, S_2 on its own could hide any image depending on the random choices made for S_1. Each pixel of S_2 has either of the two patterns shown on the slide with equal probability.

QUESTION 10

Show that a *known* plaintext attack and a *chosen* plaintext attack on a binary synchronous stream cipher are the same. More precisely, show that an attacker who can obtain the ciphertext chunk C for a known plaintext chunk P can also find the ciphertext C' for any chosen plaintext P' of the same length as P.

Suppose that KS is the keystream that was used to encrypt P. Then $C = KS \oplus P$ (where $\oplus$ denotes exclusive-OR applied bitwise). An attacker knowing P and C can then obtain $KS = C \oplus P$. Thus for any other plaintext P' the attacker knows that the ciphertext for P' would have been $C' = KS \oplus P'$.

NTNU Department of Information Security and Communication Technology

TTM4135 Applied Cryptography and Network Security
Semester Spring, 2022

Worksheet 3: Block ciphers and modes of operation

QUESTION 1

Review the definitions of the following concepts. They are things that you would be expected to know in a quiz or exam.

(a) confusion and diffusion;
(b) product cipher and iterated cipher;
(c) Feistel cipher;
(d) substitution-permutation network;
(e) ECB mode;
(f) CBC mode;
(g) CTR mode;
(h) true random number generator (TRNG) and pseudorandom number generator (PRNG).

QUESTION 2

Suppose a certain system can perform 2^{55} trial encryptions of a DES block in 1 second.

(a) Assuming that computational power remains the same, how may years should it take, on average, for this system to perform a brute force attack on two-key triple DES?

(b) Moore's law says that computers double their computational power every two years. Assuming this continues for ever, how long will it be before two-key triple DES can be broken within one day?

(a) There are 2^{112} keys to eliminate so the cryptanalyst needs to search through 2^{111} of these on average. But each key trial now requires two encryptions and one decryption, so that means the time of 3×2^{111} encryptions on average (since encryption and decryption take about the same time). Therefore the attack should take around $3 \times 2^{111}/2^{55}$ seconds = 3×2^{56} seconds. There are 31536000 seconds in a (non-leap) year which is around 2^{25} seconds. Therefore the search takes around 3×2^{31} years, or around 6 billion years

(b) Moore's Law allows to compute twice as many encryptions in the same time after two years. There are roughly 2^{16} seconds in a day so we can perform $2^{55} \times 2^{16} = 2^{55+16} = 2^{71}$ encryptions in one day at present (according to the assumption). It will be $2 \times (111 - 71) = 80$ years before we can compute 2^{111} computations in a day. Two or three years more would be required before we are able to compute 3×2^{111} encryptions within one day.

QUESTION 3

Consider a block cipher with encryption function E and decryption function D. Suppose that this cipher is *linear* with respect to messages:

$$E(M_1, K) \oplus E(M_2, K) = E(M_1 \oplus M_2, K)$$

for any messages M_1 and M_2 and any fixed key K. Suppose that the block size is 128 bits. Show that an attacker can use a chosen ciphertext attack, with just 128 chosen ciphertexts, to easily decrypt any message.

cont/...

First we show that decryption is also linear. Let C_1 and C_2 be any two ciphertexts. Then there exist two plaintexts M_1 and M_2 with $C_1 = E(M_1, K)$ and $C_2 = E(M_2, K)$ and we get:

$$\begin{aligned} D(C_1 \oplus C_2, K) &= D(E(M_1, K) \oplus E(M_2, K), K) \\ &= D(E(M_1 \oplus M_2, K), K)) \\ &= M_1 \oplus M_2 \\ &= D(C_1, K) \oplus D(C_2, K) \end{aligned}$$

Furthermore, this can be extended to any number of ciphertexts:

$$D(C_1 \oplus C_2 \oplus \ldots \oplus C_n, K) = D(C_1, K) \oplus D(C_2, K) \oplus \ldots \oplus D(C_n, K)$$

Now suppose that the attacker obtains the decryptions of the following 128 ciphertexts:

$$\begin{aligned} C_0 &= (1, 0, 0, \ldots, 0) \\ C_1 &= (0, 1, 0, \ldots, 0) \\ &\vdots \\ C_{127} &= (0, 0, 0, \ldots, 1) \end{aligned}$$

Given any ciphertext C^*, the attacker can write it as

$$C^* = a_0 C_0 \oplus a_1 C_1 \oplus a_2 C_2 \oplus \ldots \oplus a_{127} C_{127}$$

where the a_i are the bits of the ciphertext, and so by the linearity of decryption

$$D(C^*, K) = a_0 D(C_0, K) \oplus a_1 D(C_1, K) \oplus a_2 D(C_2, K) \oplus \ldots \oplus a_{127} D(C_{127}, K).$$

Since each of these decryptions is assumed to be known, the attacker can just add together the plaintexts corresponding to the bits of C^*, that is the $D(C_i, K)$ for which $a_i = 1$.

QUESTION 4

Consider the following two (toy) ciphers. Both ciphers are two-round iterated block ciphers with a block length of 8 bits and a key length of 8 bits. In each case work through the steps required to encrypt the plaintext block $P = 01010101$ using key $K = 11000011$.

(a) A substitution-permutation in which each S-box operates on sub-blocks of 2 bits. Thus $m = 4$ and $l = 2$.

- The permutation π_S is defined by the following table:

Input block	00	01	10	11
Output block	10	00	01	11

- The permutation π_P is defined by the following table where the block bits are labelled from 0 to 7:

Input position	0	1	2	3	4	5	6	7
Output position	3	6	0	5	2	1	7	4

- The key schedule is defined by $K_1 = K$ and $K_2 = K_1 << 1$ where $<< 1$ denotes cyclic shift left by one position.

(b) A Feistel cipher with:

cont/...

- f function defined by $f(X, K_i) = X \vee K_i$ for any half block X and subkey K_i. Here the $\vee$ operation is done on each bit separately so that if $K_i = (k_0, k_1, k_2, k_3)$ and $X = (x_0, x_1, x_2, x_3)$ then $f(X, K_i) = (k_0 \vee x_0), (k_1 \vee x_1), (k_2 \vee x_2), (k_3 \vee x_3)$.
- key schedule defined by K_1 is the 4 leftmost bits of K and K_2 is the 4 rightmost bits of K.

(a) The subkeys are $K_1 = 11000011$ and $K_2 = 10000111$.

Round 1

Step 1: XOR in the subkey: $01010101 \oplus 11000011 = 10010110$

Step 2: Perform the S-box substitutions: $10 \to 01$, $01 \to 00$, $01 \to 00$, $10 \to 01$. So the new state is: 01000001.

Step 3: Perform the bit position permutation: the bit in position 0 (0) become the new bit in position 3 and so on. The new state is 00001010.

Round 2

Step 1: XOR in the subkey: $00001010 \oplus 10000111 = 10001101$

Step 2: Perform the S-box substitutions: $10 \to 01$, $00 \to 10$, $11 \to 11$, $01 \to 00$. So the new state is: 01101100.

Step 3: Perform the bit position permutation. The final state is 11100010.

(b) The subkeys are $K_1 = 1100$ and $K_2 = 0011$. We set $L_0 = 0101$ and $R_0 = 0101$.

Round 1 $L_1 = R_0 = 0101$

$R_1 = L_0 \oplus f(R_0, K_1) = 0101 \oplus [R_0 \vee K_1] = 0101 \oplus (0101 \vee 1100) = 0101 \oplus 1101 = 1000$

Round 2 $L_2 = R_1 = 1000$

$R_2 = L_1 \oplus f(R_1, K_2) = 0101 \oplus [R_1 \vee K_2] = 0101 \oplus [1000 \vee 0011] = 0101 \oplus 1011 = 1110$

The final output is 10001110.

QUESTION 5

Consider the use of double encryption applied to the AES algorithm with two 128-bit keys. How much storage and computation would be required to execute a *meet-in-the-middle* attack (as described for DES in the lecture)?

We assume that the attacker has a single plaintext-ciphertext block pair, (P, C) so that $E(E(P, K_1), K_2) = C$ where E denotes single AES encryption and K_1, K_2 are the two keys used in double encryption. The attacker:

(a) computes $E(P, K)$ for all keys K and stores the results.

(b) computes $D(C, K')$ for keys K' until a match is found in the stored results.

The attack requires 2^{128} AES encryption and storage of the resulting 2^{128} ciphertext in part (a). Up to 2^{128} AES decryptions are required for part (b).
In order to make the comparison easier it is probably necessary to order the outputs from part (a). This will increase the computation required in practice.
Note that this amount of computation is currently quite infeasible. However, it is far easier than the brute force attack of searching through 2^{256} keys used in double AES encryption.

---

cont/...

QUESTION 6

Suppose we want to encrypt more than one block of random bits using ECB mode with a block cipher, but without padding. This can be achieved with a technique known as *ciphertext stealing*. For example, suppose we encrypt a 200-bit random key using AES in ECB mode with key K. The plaintext is two blocks M_1, M_2 where M_2 is a 72-bit 'short' block. Then we compute:

$$\begin{aligned} C_2 \parallel J &= E(M_1, K) \\ C_1 &= E(M_2 \parallel J, K) \end{aligned}$$

and send (C_1, C_2) to the receiver. Here J is a 56 bit random value which is never transmitted.

(a) What is the length of the ciphertext?
(b) Show how the message can be decrypted back to the original plaintext.

(a) Because J is 56 bits long, C_2 is 128-56 = 72 bits long. C_1 is a full 128-bit block and so the total length is 72+128 bits, same as the plaintext.

(b) To decrypt compute:

$$\begin{aligned} D(C_1, K) &= M_2 \parallel J \\ D(C_2 \parallel J, K) &= M_1 \end{aligned}$$

then discard J.

QUESTION 7

Suppose that the IV for CBC mode is chosen to be a counter instead of a random value (as it should be). Show that an attacker can gain information regarding the first block of a ciphertext by using a chosen plaintext attack. More specifically, show that the attacker can check whether C has first plaintext block equal to a particular block P_0 by asking for the ciphertext of a specific plaintext.

Suppose that the attacker observes a ciphertext $(IV, C_1, C_2, \ldots)$ which is the encryption of message $(P_1, P_2, \ldots)$. The attacker knows that the IV used in the next ciphertext will be $IV + 1$, so that it will be $(IV + 1, C_1', C_2', \ldots)$. He can check whether the first plaintext block in the observed ciphertext is a particular value $\hat{P}$ by asking for the ciphertext of a message with first block equal to $\hat{P} \oplus IV \oplus IV + 1$. The first block output in the new ciphertext will then be

$$C_1' = E((\hat{P} \oplus IV \oplus IV + 1) \oplus (IV + 1), K) = E(\hat{P} \oplus IV, K)$$

If the first plaintext block of the first observed message was indeed $\hat{P}$ then it follows that $C_1' = C_1$ which can be observed by the attacker.

QUESTION 8

Compare the following three modes of operation for block ciphers: ECB, CBC and CTR. Suppose that:

- AES is used as the block cipher;
- the padding method mentioned in the lectures slides (Lecture 6, Slide 7) is used (where needed);
- the nonce used in counter mode has 96 bits.

How many bits need to be transmitted in each of these three cases if the input plaintext has 104 bits?

Note that we do *not* include sending the key. This must be set up securely before the transmission starts. We need to send the ciphertext and, where needed, the IV and nonce.

cont/...

(a) For ECB only the ciphertext block is sent, one block of 128 bits. The message must be padded before encryption and after decryption.

(b) For CBC the ciphertext is also one block including padding, but we need to include 128 bits for the IV, so we obtain 256 bits in total.

(c) For CTR there is no padding required since encryption in is like a stream cipher and we take only as many bits from the keystream as needed. So the basic encrypted messages needs exactly 104 bits. We also need to send the nonce of 96 bits. Thus in total we require 104 + 96 = 200 bits.

Note that these options are giving different security guarantees. ECB gives weak encryption, CBC and CTR give better encryption (randomised).

QUESTION 9

The CTR-DRBG described in slides 28–30 of Lecture 6 keeps a counter state V and a key K in memory. The pair (K, V) is initialised, and occasionally updated, with a random seed using the Initialise and Reseed functions. In this question we assume that 128-bit keys and blocks are used (such as in AES) and also that each call to the Generate function outputs 10 blocks (1280 bits).
Suppose that an attacker obtains the current (K, V) value after 5 blocks have been generated during a call to Generate.

(a) Explain why this attacker can obtain all of the 1280 bits generated from that call to Generate.

(b) Show further that the attacker can obtain the output from all future calls to generate until Reseed is called.

(c) Explain why this attacker cannot obtain the output from any previous calls to Generate, even if Reseed has never been called.

(a) An attacker with a valid (K, V) pair can generate all following outputs by incrementing V and computing the block cipher output in the same way as the generator. The attacker can also decrement V to compute $E(V, K)$ for previous values of the counter state V.

(b) At the end of the Generate function the Update function is called with an empty D value. Thus given a valid (K, V) pair the attacker can continue with the generation in the same way as the user.

(c) To obtain the output from earlier calls to Generate the attacker needs to find the previous (K, V) pair in place before the Update function was called at the end of the previous Generate call. If (K_0, V_0) was the old pair before the Update was called and (K, V) the new pair after update was called then they are related by (slide 29) the following equation, where inc indicates increment of the counter.

$$K \parallel V = (E(V_0, K_0) \parallel E(inc(V_0), K_0)) \oplus D$$

Since D is empty when Update is called by Generate, and assuming that the block and key length are equal, we can write:

$$K = E(V_0, K_0) \quad (1)$$
$$V = E(inc(V_0), K_0) \quad (2)$$

There is no reasonable way to solve these equations to find K_0 since all the attacker has is two ciphertexts from the unknown key K_0. Therefore it should be impossible to find the output from bits generated before the Update function was called. (This means that the Update function provides the *Backtracking resistance* property.)

NTNU Department of Information Security and Communication Technology

TTM4135 Applied Cryptography and Network Security
Semester Spring, 2022

Worksheet 4: Hashing, MACs and Number theory

QUESTION 1

Review the definitions of the following concepts. They are things that you would be expected to know in the exam.

(a) collision resistance, second preimage resistance and one-wayness;
(b) birthday paradox;
(c) HMAC;
(d) GCM mode;
(e) big O notation;
(f) Fermat test;
(g) Miller–Rabin test.
(h) factorisation and discrete logarithm problems.

QUESTION 2

(a) Suppose that 10 items are chosen randomly (with replacement) from a set of 30 items. Find the probability that there is no collision (or, in other words, all the items are different). (Hint: consider the experiment one item at a time and multiply the probability that there is no collision each time.)

(b) Explain why SHA-256 can be said to match the security of AES with 128-bit keys.

(a) The probability of no collision is the product of the probabilities of no collision each time a new element is collected. For 10 elements this is

$$\frac{30}{30} \times \frac{29}{30} \times \frac{28}{30} \ldots \times \frac{21}{30} = 0.19.$$

Therefore the probability of at least one collision is $1 - 0.19 = 0.81$.

(b) Due to the birthday paradox around 2^{128} computations are required to obtain a collision in SHA-256. This is roughly the same amount of computation required for a brute-force attack on 128-bit key AES.

QUESTION 3

A rough, but simple, estimate for the probability $p(n)$ of obtaining one or more collisions when choosing n items (with replacement) from a set of H items is:

$$p(n) \approx \frac{n^2}{2H}.$$

Use this formula to estimate the probability of finding a collision in SHA-256 after 2^{128} trials, after 2^{80} trials, and after 2^{64} trials.

Since SHA-256 has a 256-bit output we have $H = 2^{256}$.

- When $n = 2^{128}$ we get $p(n) = \frac{2^{256}}{2^{257}} = 0.5$.
- When $n = 2^{80}$ we get $p(n) = \frac{2^{160}}{2^{257}} = 2^{-97}$.

cont/...

- When $n = 2^{64}$ we get $p(n) = \frac{2^{128}}{2^{257}} = 2^{-129}$.

Thus we see that even after 2^{80} trials, which is beyond normal attackers, the success probability is tiny.

QUESTION 4

When does the addition of the padding and length field in the SHA-2 family of hash functions result in an extra block to be processed?

Consider, for example, a SHA-2 variant with 1024-bit blocks (such as SHA-512) and a message m of between $(l-1) \times 1024$ bits and $l \times 1024$ bits. The message, after adding padding and the length field, is either l blocks or $l+1$ blocks. Exactly how long can the message be before $l+1$ blocks will be used?

Suppose m has length $(l-1) \times 1024 + r$ bits. At least 1 bit of padding is added and 128 bits are needed for the length field. So if $r \leq 1024 - 129 = 895$ then no block is added to the pre-processed data.

QUESTION 5

Suppose that HMAC is implemented using a hash function H, where H is an iterated hash function with compression function h.

(a) How many additional applications of the compression function h are required to compute the MAC of a message m, in comparison with computing only $H(m)$?
(b) If a MAC tag is to be computed for many different messages but the same key, how can pre-computation be used to reduce this overhead?

(a) The computation of $I = H((K \oplus \text{ipad}) \parallel m)$ requires one extra application of h to compute, compared with computing simply $H(m)$. The computation of $H((K \oplus opad) \parallel I)$ requires only two or three applications of h, depending on whether the padding adds one extra block. For all the SHA variants the output size of I is much small than the block size, so in the standardised versions there are only two extra applications of h. So there are in total 3 extra applications of h to compute $\text{HMAC}(m)$ compared to computing $H(m)$. For large messages this is a very small overhead.
(b) We can compute $h(K \oplus opad)$ and $h(K \oplus ipad)$ in advance since they are independent of the message. This saves at two of the three overhead computations of h, and can be useful if HMAC is applied to many short messages.

QUESTION 6

Consider the following simplification of HMAC, defined from any Merkle-Damgård hash function H:

$$\text{HMAC}'(M, K) = H(K \parallel M).$$

Show that this variant allows an attacker to forge a new valid MAC tag given any valid message/tag pair (m, T) by extending m to a new message m' and finding a valid tag T' for m'.

Suppose that $\text{HMAC}'(M, K) = T$ is known by the attacker where M has n-bit blocks $M_1 \| M_2 \| \ldots \| M_\ell$. If we write $h_i = h(M_i, h_{i-1})$ as the intermediate outputs of h, then the final value of the tag is $\text{HMAC}'(M, K) = h(\text{pad}, h_\ell)$.

cont/...

The attacker can, for example, choose an extra block m_{l+1}, compute the new correct padding block pad', and then compute the new correct MAC tag

$$T' = h(\mathrm{pad}', h(m_{l+1}, T))$$

which is a valid tag for the message:

$$m' = M \parallel \mathrm{pad} \parallel m_{l+1}.$$

QUESTION 7

If possible, solve for x using the Chinese Remainder Theorem (CRT):

(a) $x \equiv 5 \pmod 7$ and $x \equiv 7 \pmod{10}$.

(b) $x \equiv 3 \pmod 7$ and $x \equiv 7 \pmod{14}$.

(c) $x \equiv 2 \pmod 6$ and $x \equiv 3 \pmod{11}$.

(a) Since $\gcd(7, 10) = 1$ a solution must exist. Using the CRT we have:

$$\begin{aligned} x &= (10^{-1} \bmod 7 * 10 * 5) + (7^{-1} \bmod 10 * 7 * 7) \bmod 70 \\ &= (3^{-1} \bmod 7 * 50) + (7^{-1} \bmod 10 * 49) \bmod 70 \\ &= (5 * 50) + (3 * 49) \bmod 70 \\ &= 250 + 147 \bmod 70 \\ &= 47 \end{aligned}$$

(b) Since $\gcd(7, 14) = 7$ CRT cannot be applied (and in fact there is no solution).

(c) Since $\gcd(6, 11) = 1$ a solution must exist. Using the CRT we have:

$$\begin{aligned} x &= (11^{-1} \bmod 6 * 11 * 2) + (6^{-1} \bmod 11 * 6 * 3) \bmod 66 \\ &= (5^{-1} \bmod 6 * 22) + (6^{-1} \bmod 11 * 18) \bmod 66 \\ &= (5 * 22) + (2 * 18) \bmod 66 \\ &= 110 + 36 \bmod 66 \\ &= 14 \end{aligned}$$

QUESTION 8

Find $\phi(n)$ for all integers between 20 and 25 inclusive.

- $\phi(20) = \phi(2^2 * 5) = 2 * 4 = 8$
- $\phi(21) = \phi(3 * 7) = 2 * 6 = 12$
- $\phi(22) = \phi(2 * 11) = 1 * 10 = 10$
- $\phi(23) = 22$
- $\phi(24) = \phi(2^3 * 3) = 2^2 * 2 = 8$
- $\phi(25) = \phi(5^2) = 5 * 4 = 20$

---

cont/...

QUESTION 9

Find the discrete logarithm of the number 3 with regard to base 2, for the following moduli:

(a) modulus $p = 5$;

(b) modulus $p = 11$;

(c) modulus $p = 29$.

We need to find the value x with $2^x = 3 \bmod p$ for the different value of p.

(a) Powers of 2 modulo 5 are 2, 4, 3, 1. So $x = 3$.

(b) Powers of 2 modulo 11 are: 2, 4, 8, 5, 10, 9, 7, 3, 6, 1. So $x = 8$.

(c) 2, 4, 8, 16, 3, ... So $x = 5$.

QUESTION 10

Use the Fermat test to check whether the following numbers are prime or not. Run the test at most 4 times.

- 979
- 983

- $2^{978} \pmod{979} = 223$. Therefore 979 cannot be prime and there is no point in looking further.
- We choose a few bases b and check whether $b^{982} \bmod 983 = 1$. For example,

$$\begin{aligned} 2^{982} \bmod 983 &= 1 \\ 3^{982} \bmod 983 &= 1 \\ 11^{982} \bmod 983 &= 1 \\ 17^{982} \bmod 983 &= 1 \end{aligned}$$

We can be confident that 983 is prime. Note that the bases we chose here are not random, and in practice fixed bases are usually applied.

QUESTION 11

(a) Show that the Carmichael number $n = 1105$ passes the Fermat test for base $a = 2$ and $a = 3$.

(b) Now try the Miller–Rabin test for the same two bases and show that n is composite.

(c) Hence find a square root of 1 mod n and use this to find a factor of n.

(a) Check that $a^{1104} \bmod 1105 = 1$ for $a = 2$ and $a = 3$.

cont/...

(b) $1104 = 2^4 \times 69$. Therefore we compute

$$\begin{aligned} 2^{69} \bmod 1105 &= 967 \\ 967^2 \bmod 1105 &= 259 \\ 259^2 \bmod 1105 &= 781 \\ 781^2 \bmod 1105 &= 1. \end{aligned}$$

Here we stop since we got to 1 and the test outputs `composite` which shows that 1105 is definitely composite.

(c) In the last equation we found that 781 is a square root of 1 modulo 1105. Therefore $\gcd(780, 1105)$ is a factor of 1105. Running Euclidean algorithm gives:

$$\begin{aligned} 1105 &= 1 \times 780 + 325 \\ 780 &= 325 \times 2 + 130 \\ 325 &= 130 \times 2 + 65 \\ 130 &= 2 \times 65 \end{aligned}$$

So $\gcd(780, 1105) = 65$ and we find $1105 = 65 \times 17$.

NTNU Department of Information Security and Communication Technology

TTM4135 Applied Cryptography and Network Security
Semester Spring, 2022

Worksheet 5: RSA

QUESTION 1

Review the definitions of the following concepts. They are things that you would be expected to know in the exam.

(a) trapdoor oneway function;

(b) RSA equations;

(c) RSA padding;

(d) prime number theorem;

(e) square-and-multiply algorithm;

(f) Håstad's attack;

(g) Miller's theorem.

QUESTION 2

Suppose that an RSA public key is chosen with primes $p = 13$ and $q = 17$. Suppose that the public key $e = 5$ is used.

(a) Find the value of d.

(b) Find the ciphertext value for $M = 4$ and $M = 13$.

(c) Decrypt the ciphertext and verify that the correct value is recovered.

(a) The modulus is $n = pq = 221$. Then $d = e^{-1} \bmod \phi(n) = 5^{-1} \bmod 12 \times 16 = 5^{-1} \bmod 192$. Using Euclidean algorithm we can find that $5^{-1} \bmod 192 = 77$.

(b) For message $M_1 = 4$ the ciphertext is $C_1 = 4^5 \bmod 221 = 140$.
For message $M_2 = 13$ the ciphertext is $C_2 = 13^5 \bmod 221 = 13$. Notice that this illustrates that it is quite possible for messages to be sent to themselves in RSA. This will happen with negligible probability in practice.

(c) To decrypt we compute $M_1 = C_1^d \bmod n = 140^{77} \bmod 221 = 4$. Similarly $M_2 = 13^{77} \bmod 221 = 13$.

QUESTION 3 Challenge Question

In this question we show that $f(x) = x^2 \bmod n$ is a trapdoor one-way function, when $n = pq$ and $p \bmod 4 = q \bmod 4 = 3$ and p and q are different primes. We do this in three steps.

(a) Suppose that $x \equiv y^2 \bmod p$ for some y. Then show that $x^{(p+1)/4} \bmod p$ is a square root of x in $\mathbb{Z}_p^*$.

(b) Use the part above to show that if p and q are known, then a square root modulo n can be efficiently computed (assume we have an efficient exponentiation function). Thus p and q are a trapdoor to invert f.

cont/...

(c) Now suppose that there exists an algorithm A to find square roots modulo n. Show that if you know y so that $x \equiv y^2 \bmod n$ and A finds a different square root z with $z \neq x \bmod n$ and $z \neq -x \bmod n$, then this can be used to factorise n. Hence deduce that inverting f is as hard as factorising n so that f is one-way.

(a) Note that only half of the integers $\{1, 2, \ldots, p-1\}$ are squares of some other value (quadratic residues) modulo p. Since $x \equiv y^2 \bmod p$ for some y,

$$\begin{aligned}\left(x^{\frac{p+1}{4}}\right)^2 \bmod p &= x^{\frac{p+1}{2}} \bmod p \\ &= x^{\frac{p-1}{2}} \cdot x \bmod p \\ &= y^{p-1} \cdot x \bmod p \\ &= x\end{aligned}$$

where the last step uses Fermat's theorem.

(b) Given any x which is a square modulo n, the above algorithm can be used separately for both p and q. So we can efficiently obtain a and b so that $a^2 \bmod p = x \bmod p$ and $b^2 \bmod q = x \bmod q$. Now we can use the Chinese Remainder Theorem to solve

$$\begin{aligned}y &\equiv a \bmod p \\ y &\equiv b \bmod q\end{aligned}$$

Thus we have $y^2 \bmod p = x \bmod p$ and $y^2 \bmod q = x \bmod q$. In other words $y^2 - x$ is a multiple of both p and q so is a multiple of $n = pq$. Thus finally we have $y^2 \bmod n = x \bmod n$ so that y is the square root we were looking for.

(c) Since $x \equiv y^2 \bmod n$ and $x \equiv z^2 \bmod n$ we have $y^2 - z^2 \equiv 0 \bmod n$, so that $(y-z)(y+z)$ is a multiple of n. If $(y-z)$ or $(y+z)$ is a multiple of n then we have either $z \equiv y \bmod n$ or $z \equiv -y \bmod n$. If neither $(y-z)$ nor $(y+z)$ is a multiple of n then one must be a multiple of p and the other must be a multiple of q so that $\gcd(y-z, n) > 1$ will efficiently compute a factor of. n.

Finally note that we can choose y randomly in the range 1 to n, compute $x = y^2 \bmod n$ and then run algorithm A on input x. With probability 1/2, we will get a square root z different from $y \bmod n$ and $-y \bmod n$, since there are 4 possible square roots. If we are lucky then we have a square root as above. If we are unlucky we can choose a new random y and try again. We expect to only have to run A twice before success.

QUESTION 4

Suppose that RSA encryption uses a modulus n of 3000 bits. Assuming that the square-and-multiply method is used for exponentiation, compare the computational cost of encryption, measured in the number of squarings and the numbers of multiplications, in the following cases:

(a) $e = 3$

(b) $e = 2^{16} + 1$

(c) e is chosen randomly between 0 and n.

How much computation is required for decryption in each case?

Regard the exponent e as a binary string. If there are s bits in the exponents and t of these are '1' bits. The square-and-multiply method uses $s-1$ squarings and $t-1$ multiplications.

	squarings	multiplications
$e = 3$	1	1
$e = 2^{16} + 1$	16	1
e random	2999	1499

cont/...

The last row is an average figure. Note that e will usually have around 3000 bits if chosen randomly.

The decryption computation depends on the value of d. In all cases this can be regarded as being randomly chosen in the range $0 < d < \phi(n)$. Note that $\phi(n)$ also has approximately 3000 bits. Therefore the decryption computation will always take similar computation to the last row in the table.

QUESTION 5

Suppose that the same message m has been encrypted for three recipients with different RSA moduli: 205, 319 and 391. Each recipient uses public exponent $e = 3$. Suppose also that no random padding has been added. The three ciphertexts found are: 180, 43 and 218 respectively.

Demonstrate Håstad's attack by finding the value of m without making use of the factorisation of the moduli.

We have three modular equations:

$$\begin{aligned} m^3 &\equiv 180 \pmod{205} \\ m^3 &\equiv 43 \pmod{319} \\ m^3 &\equiv 218 \pmod{391} \end{aligned}$$

These can be solved using the Chinese remainder theorem. Let

$$\begin{aligned} y_1 &= (319 \times 391)^{-1} \bmod 205 = 129 \\ y_2 &= (205 \times 391)^{-1} \bmod 319 = 115 \\ y_3 &= (205 \times 319)^{-1} \bmod 391 = 4 \end{aligned}$$

Then $m^3 = 180 \times (319 \times 391 \times y_1) + 43 \times (205 \times 391 \times y_2) + 218 \times (205 \times 319 \times y_3) \bmod 25569445 = 1000$. Finally the message can be found by extracting the ordinary cube root to obtain $m = 10$.

QUESTION 6

Consider RSA with values $p = 23$, $q = 31$, $n = 713$ and $d = 233$. Suppose the received ciphertext is $C = 266$.

Examine the faster decryption method using the Chinese Remainder Theorem, using these values:

(a) Compute $M_p = C^{d \bmod p-1} \bmod p$.

(b) Similarly compute M_q.

(c) Combine these results using the Chinese Remainder Theorem and show that the result is correct.

$d \pmod{p-1} = 233 \pmod{22} = 13$

$d \pmod{q-1} = 233 \pmod{30} = 23$

(a) $M \bmod p = C^{d \bmod p-1} \bmod p = 266^{13} \bmod 23 = 13^{13} \bmod 23 = 8$.

(b) $M \bmod q = C^{d \bmod q-1} \bmod q = 266^{23} \bmod 31 = 18^{23} \bmod 31 = 7$.

(c) $M = 8 \times 31 \times q_1 + 7 \times 23 \times q_2 \bmod 713$.
Here $q_1 = 31^{-1} \bmod 23 = 8^{-1} \bmod 23 = 3$ and $q_2 = 23^{-1} \bmod 31 = 27$.

cont/...

$$\begin{aligned} M &= 8 \times 31 \times 3 + 7 \times 23 \times 27 \bmod 713 \\ &= 744 + 4347 \bmod 713 \\ &= 31 + 69 \\ &= 100 \end{aligned}$$

QUESTION 7

Use the diagram on Slide 20 of Lecture 10 to write down two equations for computing the OAEP padding algorithm outputs t and s from a message m. Hence show that the OAEP padding can be inverted by anyone (without using any secret) to recover m.

The inputs to the OAEP function are a message m, a constant string d of k bits and a string r of k random bits. Let m' denote m padded to be length $|n| - 2k - 8$. Then from the diagram we deduce:

$$\begin{aligned} s &= (d \parallel m') \oplus G(r) \\ t &= H(s) \oplus r \end{aligned}$$

where $\parallel$ denotes string concatenation. Then to recover m compute:

$$\begin{aligned} r &= H(s) \oplus t \\ (d \parallel m') &= s \oplus G(r) \end{aligned}$$

and finally remove the k bits of d and the padding bits.

QUESTION 8

Suppose that an attacker obtains an RSA private key $d = 233$ and also has the public key $e = 17$ and $n = 713$. Apply Miller's algorithm to factorise n.

First write $ed - 1 = 2^v \times u$ for u odd. Since $ed - 1 = 3960 = 2^3 \times 495$ we have $v = 3$ and $u = 495$.

Choose $a = 2$ and compute $2^u \bmod n = 2^{3960} \bmod n = 1$. Unfortunately we do not get a non-trivial square root this time.

Choose $a = 3$ and compute $3^u \bmod n = 3^{3960} \bmod n = 185$. We are lucky and this is not ± 1 so we can find a non-trivial square root of 1 by squaring.

$185^2 \bmod n = 1$ so $a = 185$ is a non-trivial square root of 1.

Therefore we can find a factor of n by computing $\gcd(a - 1, n) = \gcd(184, 713) = 23$. The other factor is then $713/23 = 31$. Therefore we have factorised the modulus: $713 = 23 \times 31$.

QUESTION 9

Suppose that you know that two RSA moduli $n_1 = 1517$ and $n_2 = 1591$ share one factor. Use this knowledge to efficiently factorise both numbers. (Do not try to factorise both directly.)

---

cont/...

Since n_1 and n_2 share a factor, this factor will be their GCD. Thus we can apply Euclid's algorithm and compute:

$$\begin{aligned} 1591 &= 1517 \times 1 + 74 \\ 1517 &= 74 \times 20 + 37 \\ 74 &= 37 \times 2 \end{aligned}$$

Thus the GCD is 37. Then $1517 = 37 \times (1517/37) = 37 \times 41$ and $1591 = 37 \times (1591/37) = 37 \times 43$.

NTNU Department of Information Security and Communication Technology

TTM4135 Applied Cryptography and Network Security
Semester Spring, 2022

Worksheet 6: Discrete log algorithms and digital signatures

QUESTION 1

Review the definitions of the following concepts. They are things that you would be expected to know in the exam.

(a) discrete logarithm problem
(b) generator of $\mathbb{Z}_p^*$
(c) Diffie–Hellman key exchange
(d) Elgamal cryptosystem
(e) digital signature
(f) existential forgery and selective forgery
(g) digital signature algorithm (DSA)

QUESTION 2

Let $p = 43$. Verify that $g = 3$ is a generator of $\mathbb{Z}_p^*$. Suppose that Alice and Bob execute the Diffie–Hellman key exchange protocol with Alice's input being $a = 5$ and Bob's input $b = 13$. Show that both Alice and Bob will compute the same shared secret.

We first factorise $p - 1$ as $42 = 2 \times 3 \times 7$. To check that g is a generator we need to verify that $g^{21} \bmod 43 \neq 1$, $g^{14} \bmod 43 \neq 1$ and $g^6 \bmod 43 \neq 1$. Since $3^{21} \bmod 43 = 42$, $3^{14} \bmod 43 = 36$ and $3^6 \bmod 43 = 41$ it follows that 3 is indeed a generator.

- Alice sends $g^a \bmod p = 3^5 \bmod 43 = 28$.
- Bob sends $g^b \bmod p = 3^{13} \bmod 43 = 12$.
- Bob computes $(g^a)^b \bmod p = 28^{13} \bmod 43 = 34$.
- Alice computes $(g^b)^a \bmod p = 12^5 \bmod 43 = 34$.

QUESTION 3

The Diffie–Hellman protocol can be defined, but is not necessarily secure, in any group. Illustrate this by defining Diffie–Hellman in the *additive* group modulo p for some prime p, instead of the multiplicative group where it is usually defined. Would this be secure for sufficiently large values of p?

We need to define a generator for the additive group. For any odd prime p we could choose $g = 2$. Then

- Alice sends $2a \bmod p$ to Bob.
- Bob sends $2b \bmod p$ to Alice.
- Bob computes $(2a) * b \bmod p$.
- Alice computes $(2b) * a \bmod p$.

Thus both Alice and Bob compute the same value: $2ab \bmod p$.

Unfortunately this is not secure for any value of p. An attacker who obtains $2a \bmod p$ sent by Alice can efficiently find $2^{-1} \bmod p$ and therefore recover $a \bmod p$ and finally compuete $2ab \bmod p$ just as Alice does.

cont/...

QUESTION 4

It is common in the Elgamal encryption algorithm for users to share the modulus p and generator g. Why is it not possible for users to share the same modulus n in the RSA cryptosystem?

At one level we can simply say that users generate their own moduli. If user Alice tries to use Bob's modulus n then Alice will not be able to factorise n and will be unable to generate a valid (e, d) pair. Thus it just won't work.

More interesting is the case that a trusted party could generate n and many valid (e, d) pairs and distribute them to different parties. This works but it is totally insecure. We know from Miller's algorithm that knowledge of any valid (e, d) pair is sufficient to factorise n. Therefore Alice could use her own (e_A, d_A) pair to obtain the factors of n and can then obtain Bob's private d_B value from the public e_B and n values.

QUESTION 5

In the Elgamal cryptosystem Alice and Bob have public keys g^{x_A} and g^{x_B} respectively, with corresponding private keys x_A and x_B. When Alice wants to send a message confidentially to Bob she chooses an ephemeral private key a and constructs a new shared secret g^{ax_B}.

Consider the following variant of the Elgamal cryptosystem. Instead of choosing a new random value a, Alice simply computes the static Diffie–Hellman value $X = (y_B)^{x_A} \bmod p$ and sends $C = MX \bmod p$ to Bob as the ciphertext.

(a) How does Bob decrypt?
(b) What could be the advantages of such a scheme as compared with normal Elgamal encryption?
(c) Show that this scheme is, unfortunately, completely insecure against a known plaintext attack.

(a) The recipient needs to compute $X = (y_A)^{x_B} \bmod p$ and then $M = CX^{-1}$.
(b) One advantage is that the new cryptosystem is computationally cheaper since only one exponentiation is required. Furthermore, ciphertexts are shorter than in Elgamal encryption.
(c) Knowing one ciphertext and corresponding plaintext message allows an attacker to obtain X and therefore all plaintexts. This is enough to say that the cryptosystem should be avoided, but another disadvantage is that the recipient needs to know the public key of the sender.

QUESTION 6

Suppose that Alice has a public key $y = 5, g = 2, p = 11$ for the Elgamal encryption algorithm. Here $g = 2$ is a generator for $\mathbb{Z}_{11}^*$. Compute a valid ciphertext for the message $M = 3$ intended for Alice, showing the steps required.

Choose a random k. Here we assume $k = 3$. Then compute $g^k \pmod p = 2^3 \pmod{11} = 8$ and $(C_1, C_2) = (g^k \bmod p, My^k \pmod p) = (8, 3 \times 5^3 \pmod{11} = (8, 3 \times 4) = (8, 1)$. To check this we can decrypt.

The private key is $x = 4$ (we can find by trial and error here). So the decrypted ciphertext is $C_2 \times (C_1^x)^{-1} = 1 \times (8^4)^{-1} \pmod{11} = 4^{-1} \pmod{11} = 3$.

---

cont/...

QUESTION 7

Compare the efficiency of the Elgamal cryptosystem in $\mathbb{Z}_p^*$ and the RSA cryptosystem with modulus n. Assume that the size of the modulus p and n is the same in each case. Compare:

- the cost of key generation;
- the computation required for encryption;
- the computation required for decryption;
- the size of the public keys and ciphertexts.

In this answer we assume that the Elgamal cryptosystem is implemented in $\mathbb{Z}_p^*$. When implemented on elliptic curves it is much more efficient.

- Key generation for Elgamal requires obtaining a prime p and a generator g for $\mathbb{Z}_p^*$. Prime generation requires testing for primality which can be done efficiently using a probabilistic test such as the Miller–Rabin test. Checking for a generator requires a few exponentiations (depending on the number of factors of $p-1$). After p and g are found one more exponentiation is required to obtain the public key from the private key.

  Key generation for RSA requires finding two primes p and q, but these are only half the size of the modulus so finding them is cheaper than in Elgamal. After p and q are found the exponent d must be constructed from e which requires finding an inverse modulo $\phi(n)$. This can be done cheaply using the Euclidean algorithm.

  Overall key generation for RSA should be cheaper than for Elgamal, although they are not greatly different and key generation only happens once. If Elgamal reuses the p and g values then key generation is much cheaper. As we know, the modulus for RSA cannot be re-used.
- Encryption for Elgamal requires two exponentiations and one multiplication. Encryption for RSA requires one exponentiation. Moreover the exponent e can be chosen to be small – if $e = 3$ then encryption needs only two multplications. Therefore RSA encryption is much cheaper than Elgamal encryption.
- Decryption for Elgamal requires one exponentiation, one multiplication and one inversion. An inversion can be done efficient using the Euclidean algorithm (or an exponentiation). RSA decryption requires one exponentiation. This exponentiation can be computed more cheaply by using the Chinese remainder theorem.
- Elgamal ciphertexts are twice the length of the modulus, while RSA ciphertexts are equal to the length of the modulus. Note that Elgamal plaintexts can be any value less than the modulus, but RSA requires padding which uses several bytes and means that messages have to be significantly shorter than the modulus.

QUESTION 8

In 2019 elections for the Moscow city parliament an electronic voting system used a simple variant of ElGamal encryption to protect a user vote M. For the public key $K = (K_1, K_2, K_3) = (y_1, y_2, y_3) = (g^{x_1}, g^{x_2}, g^{x_3})$ the following details the encryption algorithm. First the encrypting party chooses random k_1, k_2, k_3 and computes the following values.

$C_1 = E(M, K_1) = (g^{k_1} \bmod p, My_1^{k_1} \bmod p) = (a_1, b_1)$

$C_2 = E(a_1, K_2) = (g^{k_2} \bmod p, a_1 y_2^{k_2} \bmod p) = (a_2, b_2)$

$C_3 = E(a_2, K_3) = (g^{k_3} \bmod p, a_2 y_3^{k_3} \bmod p) = (a_3, b_3)$

Finally the ciphertext is the 4-tuple $C = (b_1, b_2, a_3, b_3)$. Note that C_1, C_2 and C_3 are ordinary Elgamal ciphertexts.

For unclear reasons, the implementation used a prime p with only 256-bits for computations in $\mathbb{Z}_p^*$. You can read the details of the analysis here: `https://arxiv.org/abs/1908.05127`.

(a) Explain how the message M can be recovered from C given the private key (x_1, x_2, x_3)

(b) How long is the private key?

cont/...

(c) If an attacker can find discrete logarithms in $\mathbb{Z}_p^*$ in time T, how long will take the same attacker to find the message M?

(a) Since C_1, C_2 and C_3 are ordinary Elgamal ciphertexts they can be decrypted sequentially with the separate parts of the private key. The decryptor first obtains a_2 from C_3 so that C_2 is known, then obtains a_1 from C_2 so that C_1 known and finally M that C_1

(b) $3 \times 256 = 768$ bits

(c) $3T$

QUESTION 9

Suppose an attacker can break the hash function h used to form a digital signature (RSA or DSA) by finding collisions. How can this lead to attacks on the signature? Is this attack existential or selective?

Signatures of two messages with the same hash value are the same. If an attacker finds a collision between messages m_1 and m_2 he may be able to persuade the signer to sign m_1, especially if it is a meaningful message. Thus he also obtains a forged signature on m_2. If we require security against a chosen message attack (see slide 9 of Lecture 12) then we assume the adversary can obtain signatures on any message of his choice so this attack is valid. This is generally an existential forger only.

This same process shows that breaking the second-preimage resistance of the hash function leads to a selective forgery.

QUESTION 10

Suppose that RSA signatures are used with a hash function that is not one-way (that is the attacker can invert the hash function). Show how an existential forgery is possible against such a signature: an attacker can form valid signatures from e and n alone.

Recall that a valid RSA signature on a message m is the value $S = h(m)^d \bmod n$ where d is the private signing key and n is the modulus. Then e is the public verification key.

The attacker chooses a random value X and computes $Y \equiv X^e \bmod n$. By the RSA equation, $Y^d \equiv X^{ed} \equiv X \pmod n$. Now suppose that the attacker can invert the hash function to find a value m with $h(m) = Y$. Then we have $X = h(m)^d \bmod n$ which means that X is the RSA signature for m.

QUESTION 11

(a) Show that the verification equation for Elgamal signatures works. That is, if (r, s) is a valid Elgamal signature, then $g^m \equiv y^r r^s \bmod p$.

(b) Similarly check that the verification equation works for DSA signatures.

(a) We have $r = g^k \bmod p$ and $s = k^{-1}(m - xr) \bmod (p-1)$ and $y = g^x \bmod p$. Rewrite the second equation as $m = ks + xr \bmod (p-1)$. Therefore

cont/...

$$\begin{aligned} g^m \bmod p &= g^{ks+xr \bmod (p-1)} \bmod p \\ &= g^{ks} g^{xr} \bmod p \\ &= (g^k)^s (g^x)^r \bmod p \\ &= r^s y^r \bmod p \end{aligned}$$

(b) We have $r = (g^k \bmod p) \bmod q$ and $s = k^{-1}(H(m) - xr) \bmod q$ and $y = g^x \bmod p$. During the verification we compute $u_1 = H(m)w \bmod q$ and $u_2 = rw \bmod q$. Therefore

$$\begin{aligned} (g^{u_1} y^{-u_2} \bmod p) \bmod q &= (g^{H(m)w \bmod q} (g^x)^{-rw \bmod q} \bmod p) \bmod q \\ &= (g^{w(H(m)-xr) \bmod q} \bmod p) \bmod q \\ &= (g^{wks \bmod q} \bmod p) \bmod q \\ &= (g^k \bmod p) \bmod q \text{ since } w = s^{-1} \bmod q \\ &= r \end{aligned}$$

QUESTION 12

Suppose the parameters $p = 23$, $q = 11$, $g = 3$ are used for the DSA signature.

(a) Show that g has order q as required.
(b) If the private key is $x = 5$, what is the public key y?
(c) Compute a valid signature for a message m whose hash value is assumed to be $SHA(m) = 8$.
(d) Show that the verification equation works for your signature.

(a) To show that 3 has order 11 we need to check that $3^{11} \equiv 1 \pmod{23}$ but it is not true that $3^j \equiv 1 \pmod{23}$ for $1 < j < 11$.

j	$3^j \pmod{23}$
1	3
2	9
3	4
4	12
5	13
6	16
7	2
8	6
9	18
10	8
11	1

(b) From the above table we see that the public key is $y = g^x \pmod p = 3^5 \pmod{23} = 13$.

(c) To sign the message with $SHA(m) = 8$ we need to choose a random value k. The simplest case to do by hand is $k = 2$ to we assume this case. Then the signature has two parts r and s.

$$\begin{aligned} r &= (g^k \bmod p) \bmod q \\ &= (3^2 \bmod 23) \bmod 11 \\ &= 9 \bmod 11 \end{aligned}$$

To obtain s first note that $k^{-1} \pmod{11} \equiv 6$. Then

$$\begin{aligned} s &= k^{-1}(SHA(m) - xr) \bmod q \\ &= 6(8 - 5 \times 9) \bmod 11 \\ &= 6 \times 7 \bmod 11 \\ &= 9 \end{aligned}$$

Therefore a valid signature is $(r, s) = (9, 9)$.

cont/...

(d) To verify the signature we first calculate $w \equiv s^{-1} \bmod q \equiv 9^{-1} \bmod 11 = 5$. Next we compute $u_1 \equiv SHA(m)w \bmod q = 8 \times 5 \bmod 11 = 7$ and $u_2 \equiv rw \bmod q \equiv 9 \times 5 \bmod 11 = 1$. Finally we need to check that

$$\begin{aligned}
(g^{u_1} y^{-u_2} \bmod p) \bmod q &= r \bmod q \\
(g^{u_1} y^{-u_2} \bmod p) \bmod q &\equiv (3^7 \times 13^{-1} \bmod 23) \bmod 11 \\
&\equiv (3^7 \times 16 \bmod 23) \bmod 11 \\
&\equiv (2 \times 16 \bmod 23) \bmod 11 \\
&\equiv 32 \bmod 23) \bmod 11 \\
&\equiv 9 \bmod 11 \\
&\equiv r
\end{aligned}$$

Thus $(r, s) = (9, 9)$ is a valid signature.

QUESTION 13

Compare the efficiency of DSA signatures, Elgamal signatures, and RSA signatures assuming that the modulus size is 2048 bits in each case, and that the prime q in DSA signatures is of length 256 bits. Compare:

- the cost of key generation;
- the computation required for signature generation;
- the computation required for signature verification;
- the size of the public keys and signatures.

- Key generation for RSA signatures and Elgamal signatures is exactly the same as for RSA encryption and Elgamal encryption respectively. In general RSA key generation is cheaper, but if we assume that parameters p and g are fixed for Elgamal signatures, then Elgamal key generation is cheaper.

  The difference between key generation for Elgamal and DSA signatures is that the private key for DSA signatures is chosen in the range $0 < x < q$ whereas for Elgamal signatures $0 < x < p - 1$. The generation of the public key $y = g^x \pmod p$ is therefore cheaper for DSA signatures than for Elgamal signatures. (Remember q is 256 bits long while p is 2048 bits long).

- – Signature generation for RSA signatures requires one full length exponentiation.
  – Signature generation for Elgamal signatures requires one full length exponentiation plus an inversion and a couple of multiplications.
  – Signature generation for DSA requires one short exponentiation plus one inversion (with a short modulus q) and a couple of multiplications.

  Overall DSA is easily the cheapest. Consider the example figures and assume that the square-and-multiply algorithm is used for exponentiation, and that squaring and multiplication are around the same cost. Then the RSA signature generation cost is around $2048 + 1024 = 3072$ multiplications. Even if we use an inefficient method for computing inverses (compute $k^{-1} \bmod q = k^{q-1} \bmod q$) the cost of DSA signature generation is only $384 + 384 + 2 = 770$ multiplications on average.

- – Signature verification for RSA signatures requires one full length exponentiation. However, in practice the public exponent e is chosen to be very short so that a very short exponentiation is all that is needed.
  – Signature verification for Elgamal signatures requires three full length exponentiations. In practice this can be reduced to less than 2, but it is still more expensive than RSA in every case.
  – Signature verification for DSA requires two short exponentiations plus one inversion (with a short modulus) and a couple of multiplications.

  Overall when using a small modulus (particularly $e = 3$) RSA is the cheapest. If RSA had to use a random public exponent e then DSA would be the cheapest.

- The public key for RSA signatures is the pair (n, e). If e is short (and fixed for the whole system) then only n is required. In the best case this is 2048 bits.

  The public key for both Elgamal and RSA signatures is of the form $y = g^x \pmod p$ which is 2048 bits in our example. (Note that a shorter private key x in DSA does not result in a shorter public key y.)

  – RSA signatures are the same length as the modulus n, so 2048 bits in our example.

cont/...

- Elgamal signatures are two values of the same length as p, so 4096 bits in our example.
- DSA signatures are two values of length q, so 512 bits in our example.

Overall DSA signatures are much the shortest option.

QUESTION 14

Using the result you found in QUESTION 13, describe the situations in which it may be preferable to use an RSA signature rather than a DSA signature. Does this shed any light on why RSA signatures are much more popular than DSA signatures for signing digital certificates?

Situations in which a signature is verified many times can make the efficiency of signature verification more important than the efficiency of signature generation. Certificates are a good example of when this applies. RSA signatures are faster to verify than DSA signature so this is an inventive to use RSA signatures for certificates even though the signature is longer and more expensive to generate.

QUESTION 15

Suppose that the same value of the random k is used to generate two different DSA signatures. Show that this is sufficient for an attacker to find the private signing key. (This was the attack used to break the software verification on the Sony Playstation 3 in 2010 because their implementation used a fixed k. Sony used the elliptic curve version: `https://arstechnica.com/gaming/2010/12/ps3-hacked-through-poor-implementation-of-cryptography/`.)

If (r_1, s_1) and (r_2, s_2) are two DSA signatures on the messages m_1 and m_2 with the same k, then:

$$\begin{aligned} s_1 &= k^{-1}(H(m_1) - xr_1) \bmod q \\ s_2 &= k^{-1}(H(m_2) - xr_2) \bmod q \end{aligned}$$

where x is the private signing key. So we have two equations with two unknowns, x and k. Note that we also have $r_1 = r_2 = r$. Since s_1 and s_2 are known, the attacker can compute

$$X = \frac{s_1}{s_2} = \frac{H(m_1) - xr_1}{H(m_2) - xr_2} \bmod q.$$

Thus

$$X(H(m_2) - xr_2) = H(m_1) - xr_1 \bmod q$$

or

$$x(r_1 - Xr_2) = H(m_1) - XH(m_2) \bmod q$$

and finally

$$x = \frac{H(m_1) - XH(m_2)}{r(1 - X)} \bmod q$$

where everything on the right is known and so we can solve for the private key x.

NTNU Department of Information Security and Communication Technology

TTM4135 Applied Cryptography and Network Security
Semester Spring, 2022

Worksheet 7: key establishment and TLS (Lectures 13 and 14)

QUESTION 1

Review the definitions of the following concepts. They are things that you would be expected to know in the exam.

(a) key predistribution;
(b) session key distribution;
(c) key agreement;
(d) Needham-Schroeder protocol;
(e) Kerberos overall structure;
(f) TLS ciphersuite;
(g) TLS record protocol and TLS handshake protocol.

QUESTION 2

Discuss the advantages and disadvantages of using key predistribution, session key distribution and key agreement protocols in the following scenarios:

- a corporate network such as NTNU's Intranet;
- a small company or domestic environment;
- Internet communications (e.g. HTTPS, secure email).

In the following comparison we assume that key agreement is based on public keys, which requires a supporting public key infrastructure (PKI) to register users, issue certificates, revoke certificates, etc.

- *Corporate network:* Key predistribution is not a good choice for a corporate network because of its dynamic nature. Nodes (e.g. users and hosts) are added regularly. When a new node is added to the network, then every other node in the network needs to be updated with a new key shared with the new node. This process scales poorly.

  Key distribution is generally more adequate here than using PKI + key agreement protocols and is commonly used in practice (e.g. Kerberos in Microsoft networks). The main advantages of key distribution over public-key based key agreement are:

  - key distribution schemes can be based solely on symmetric key algorithms, which are much faster than public key algorithms;
  - no certificate management overhead is incurred.

cont/...

The main disadvantage is the need for an online trusted authority (normally referred to as authentication server) which shares a long term secret with each node.

- *Small company or home:* This type of network usually has a small and fairly static membership configuration, i.e. new nodes are added rarely to the network. Hence key predistribution schemes are a good choice. The communication overhead of key distribution is the main disadvantage in this setting.

  This picture may be changing with the Internet of Things making domestic networks more complex and dynamic.

- *Internet:* The most popular choice for the Internet is PKI + key agreement. This is what is used for example in SSL/TLS. Key distribution schemes require all nodes to register with the authentication server in order to establish the long term secret key. This is not feasible in an network as large and diverse as the Internet. Key predistribution is clearly unworkable due to its poor scalability.

QUESTION 3

Find an attack on each of the following protocols. In each case the goal is key establishment, N_A and N_B are nonces chosen by A and B respectively, and K_{AB} is the session key. K_{AS} and K_{BS} are key-encrypting keys initially shared between S and A, and between S and B respectively. Assume that $\{X\}_K$ denotes authenticated encryption of message X with key K.

(a) In this protocol B includes A's challenge in message 4, intended to give key confirmation and key freshness.

1. $A \longrightarrow B : ID_A, ID_B, N_A$
2. $B \longrightarrow S : ID_A, ID_B, N_A, N_B$
3. $S \longrightarrow B : \{K_{AB}\}_{K_{AS}}, \{N_B, ID_A, ID_B, K_{AB}\}_{K_{BS}}$
4. $B \longrightarrow A : \{K_{AB}\}_{K_{AS}}, \{N_A, ID_A, ID_B\}_{K_{AB}}$

(b) In this protocol S saves on encryption by only including the identity of the intended recipient in each field.

1. $A \longrightarrow B : ID_A, ID_B, N_A$
2. $B \longrightarrow S : ID_A, ID_B, N_A, N_B$
3. $S \longrightarrow B : \{N_A, ID_A, K_{AB}\}_{K_{AS}}, \{N_B, ID_B, K_{AB}\}_{K_{BS}}$
4. $B \longrightarrow A : \{N_A, ID_A, K_{AB}\}_{K_{AS}}$

(c) This protocol, published by Tatebayeshi, Matsuzaki and Newman in 1989, uses public key encryption $E_K(\cdot)$. K_S is the public key of the server which is assumed known by all users. The random number generated by A is used as a symmetric encryption key in message 4. The random number generated by B is used as the session key: $K_{AB} = N_B$.

1. $A \longrightarrow S : ID_A, ID_B, E_{K_S}(N_A)$
2. $S \longrightarrow B : ID_A$
3. $B \longrightarrow S : ID_A, ID_B, E_{K_S}(N_B)$
4. $S \longrightarrow A : ID_B, \{N_B\}_{N_A}$

cont/...

The following are some sample attacks. You may be able to find others. It might help to visualise the protocols and attacks if you draw the message flows between the parties.

(a) As hinted in the question, A can be deceived because she relies on B for key freshness. Probably the simplest attack is a replay by C of an old key K'_{AB} previously used between A and B. The field $\{K'_{AB}\}_{K_{AS}}$ can be recorded by C from an earlier session.

1. $A \longrightarrow C : ID_A, ID_B, N_A$
4. $C \longrightarrow A : \{K'_{AB}\}_{K_{AS}}, \{N_A, ID_A, ID_B\}_{K'_{AB}}$

In this second attack C changes the identity in message 2 from B to C.

1. $A \longrightarrow C : ID_A, ID_B, N_A$
2. $C \longrightarrow S : ID_A, ID_C, N_A, N_C$
3. $S \longrightarrow C : \{K_{AC}\}_{K_{AS}}, \{N_C, ID_A, ID_C, K_{AC}\}_{K_{CS}}$
4. $C \longrightarrow A : \{K_{AC}\}_{K_{AS}}, \{N_A, ID_A, ID_B\}_{K_{AC}}$

(b) In this protocol either A or B may be deceived about the identity of the other party. In the following C simply changes B's identity to C and A will never see the difference.

1. $A \longrightarrow C : ID_A, ID_B, N_A$
2. $C \longrightarrow S : ID_A, ID_C, N_A, N_C$
3. $S \longrightarrow C : \{N_A, ID_A, K_{AC}\}_{K_{AS}}, \{N_C, ID_C, K_{AC}\}_{K_{CS}}$
4. $B \longrightarrow A : \{N_A, ID_A, K_{AC}\}_{K_{AS}}$

(c) There are many known attacks on this protocol. Here is an attack in which C masquerades as B.

1. $A \longrightarrow S : ID_A, ID_B, \{N_A\}_{K_S}$
2. $S \longrightarrow C : ID_A$
3. $C \longrightarrow S : ID_A, ID_B, \{N_C\}_{K_S}$
4. $S \longrightarrow A : ID_B, \{N_C\}_{N_A}$

Here is another attack in which C masquerades as A.

1. $C \longrightarrow S : ID_A, ID_B, \{N_C\}_{K_S}$
2. $S \longrightarrow B : ID_A$
3. $B \longrightarrow S : ID_A, ID_B, \{N_B\}_{K_S}$
4. $S \longrightarrow A : ID_B, \{N_B\}_{N_C}$

QUESTION 4

There are three main approaches to providing *freshness* (protection against replay of messages) in protocols: random challenges, time stamps, counters.

(a) Discuss the advantages and disadvantages of each option.

(b) The fixed Needham-Schroeder protocol in the lecture uses nonces for freshness. Modify it so that freshness is achieved using counters. Specify the checks that each party must perform on receipt of the protocol messages.

cont/...

(a) The goal of all three mechanisms is to assured the receiver of a message that it is *fresh*, i.e. it is not an old message being replayed.

- The first mechanism requires two protocol flows between the sender A and the receiver B. In the first flow, the B sends a newly generated random number,N, to the A. In the second message the sender sends the message M together with the challenge N . On receipt of this, the receiver checks that the challenges are the same.

  The main advantage of using random challenges for messages freshness is that the parties do not have to maintain any state across messages. A disadvantage is that it requires interaction between the sender and the receiver. This would exclude applications such as email. Random challenges also require availability of a good random number generator.

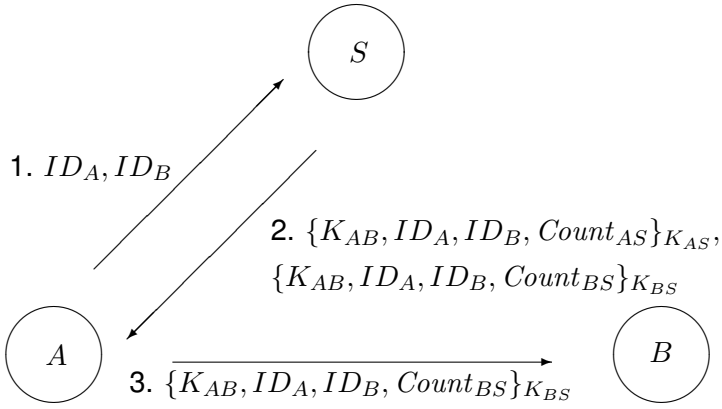
- Time stamps only require one flow. A simply sends the message together with current time stamp (a string indicating the time is some agreed format). B checks that the time stamp is recent.

  An advantage of time stamps is that there is no need for interaction between the receiver and sender: it only requires one flow. The main disadvantage is that both A and B must maintained synchronised clocks. Synchronisation of clocks is done by interacting with a time server and should be done securely, i.e. in a way that protects against interference form attackers.

- Counters work as follows. The first time A sends a message to B, A sets a counter $C_B = 1$ and sends C_B together with the message. On receipt of the first message from A, B initialises a counter $C_A = 1$. For subsequent messages from A to B, A increases the counter $C_B = C_B + 1$ and attaches it to the message. B then check that $C_B = C_A + 1$, if so it accept the message as fresh and increases $C_A = C_A + 1$.

  As with time stamps, using counters for freshness only requires one flow. A difference with respect to time stamps is that there is no need for a third party (the time server). Disadvantages of counters is that parties must maintain state. Each party must keep a counter for every other party.

(b) In the following protocol, $Count_{AS}$ and $Count_{BS}$ are counters shared by the server S with A and B respectively. Both A and B check that received counter is correct, according to their own stored values. Note that there is no need for B to contact A beforehand. A more complex process may be required if synchronization between S and the users may be lost (for example due to interrupted protocol runs).



QUESTION 5

Consider a scenario in which a client C uses Kerberos to access a printer server V. Suppose that $\mathrm{authenticator}_C$ is not sent in the level 3 interaction. Explain a scenario in which this can allow an adversary to obtain documents printed by C.

We can assume that an attacker is able to obtain the ticket issues to C in order to obtain access to V. If no authenticator is used to access the server then the attacker can replay the ticket and then also replay the print jobs sent by C. We can also reasonably assume that the attacker may have logged out and no longer has physical control of the printer queue.

However, everything really depends on how the key $K_{C,V}$ is used to access the print server. For example, if V keeps a counter of the print jobs and expects an incremented counter to be securely sent with each print job then a replay of the print job would be detected. Kerberos does not specify how the shared key is used.

QUESTION 6

(a) Show a protocol flow diagram for TLS 1.2 using RSA encryption in the handshake protocol. Use this to illustrate how this protocol fails to achieve forward secrecy.

(b) Draw a similar diagram for TLS 1.2 using Diffie–Hellman key exchange authenticated by signatures with certified signing keys. Explain why the attack from the RSA case no longer applies.

(a) Here we assume that client C and server S have used the server hello and client hello messages to negotiate a ciphersuite incorporating RSA. This means that they use the simplified protocol shown on Lecture 13 slides 18, 19 and 20 without any server key exchange message. The server sends its certificate $Cert_S$ to the client, the client chooses the premaster secret key PMS and sends it encrypted to the server using the server's public key K_S in the client key exchange message. (The client is also allowed to send a certificate but in practice this does not usually happen.)

$$\begin{aligned} S &\longrightarrow C: \quad Cert_S \\ C &\longrightarrow S: \quad E(K_S, PMS) \end{aligned}$$

After this both C and S will compute the session keys from PMS and other public values.

This exchange fails to achieve forward secrecy. Suppose an adversary records all the data exchanged and later, after the session is finished, obtains the secret decryption key of S. Then the adversary can recover PMS from the ciphertext value sent and then recover all the session keys and decrypt all the exchanged data.

(b) Now we assume that client C and server S have used the server hello and client hello messages to negotiate a ciphersuite incorporating ephemeral Diffie-Hellman. This ciphersuite will include the DIffie–Hellman group with generator g and they both use a key exchange

cont/...

message to exchange their ephemeral Diffie-Hellman values. The server sends its certificate $Cert_S$ to the client. The server also chooses its random value x and sends its Diffie-Hellman value g^x, signed with its signing key to the client. The client checks the certificate and then the signature. The client chooses its own secret and sending its Diffie-Hellman value g^y to the server.

$$\begin{aligned} S \longrightarrow C: &\quad Cert_S, g^x, Sig_S(g^x) \\ C \longrightarrow S: &\quad g^y \end{aligned}$$

The pre-master secret PMS is derived from the shared Diffie-Hellman secret g^{xy}. After this both C and S will compute the session keys from PMS and other public values.

This exchange *does* achieve forward secrecy. Suppose an adversary records all the data exchanged and later, after the session is finished, obtains the long-term secret key S, which is just its signing key. But without the Diffie-Hellman secret x this does not help the adversary to obtain the PMS value since this is the same problem as breaking the Diffie-Hellman key exchange protocol.

QUESTION 7

Consider the following ciphersuite specifications for TLS. What do each of them mean? Comment on the security of each of the choices regarding: choice of algorithms; key length; forward secrecy.

(a) TLS_RSA_WITH_RC4_128_MD5

(b) TLS_RSA_WITH_NULL_SHA

(c) TLS_DHE_RSA_WITH_3DES_EDE_CBC_SHA

(d) TLS_DHE_DSA_WITH_AES_256_CBC_SHA256

(e) TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256

(f) TLS_CHACHA20_POLY1305_SHA256

The codes before the WITH statement are concerned with the public-key algorithms used for key exchange in the Handshake protocol while the codes after the WITH statement are concerned with the algorithms used for encryption and authentication in the Record protocol (and sometimes also the PRF used for session key generation). Note that we have examined all the algorithms mentioned here in earlier lectures in the course.

(a) TLS_RSA_WITH_RC4_128_MD5:

- The RSA option is used for key exchange so the master secret is sent by the client encrypted using the server's RSA public key. This is still a fairly common method of key exchange in TLS and is generally believed secure but does not provide forward secrecy.

cont/...

- RC4 is used for encryption with a 128-bit key. This is no longer a recommended choice since attacks have shown that RC4 is best avoided. MD5 is used to produce the MAC using the HMAC algorithm. MD5 is not a good hash function for many purposes since collisions have been found several years ago. While there are no known attacks on MD5 when used in HMAC, it is usually recommended to use a more recent hash function.

(b) TLS_RSA_WITH_NULL_SHA

- The RSA option is used as in (a).
- No encryption is used which may be justifiable in some circumstances but is potentially a major weakness. SHA-1 is used to produce the MAC using the HMAC algorithm. SHA-1 has also been broken, and recently actual collisions have been found. While there are no known attacks on SHA-1 when used in HMAC, it is usually recommended to use a more recent hash function.

(c) TLS_DHE_RSA_WITH_3DES_EDE_CBC_SHA

- The DHE code means that Diffie-Hellman with ephemeral keys is used for key exchange.
- Triple-DES in cipher block chaining mode is used for encryption in the Record layer. Triple-DES has a 168-bit key. Although the key length is probably enough, it is recommended today to avoid such block ciphers with only 64-bit block length. In addition, it is much slower than AES. SHA-1 is used for HMAC as in (b).

(d) TLS_DHE_DSA_WITH_AES_256_CBC_SHA256

- Diffie-Hellman with ephemeral keys is used with DSA signatures to authenticate. These algorithms are running in $\mathbb{Z}_p^*$ for a suitable length of prime p.
- AES in cipher block chaining mode is used for encryption in the Record layer with a 256-bit key. This key length is more than adequate for secure encryption. SHA-256 is used for HMAC. Although this length of MAC is secure enough for most security purposes it is not well matched with the encryption key length. Using instead SHA-512 would mean that finding collisions in a chosen message attack takes the same number of expected steps as brute-force key search on AES with a 256-bit key.

(e) TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256

- Diffie-Hellman with ephemeral elliptic curve keys is used with ECDSA signatures to authenticate.
- AES in GCM mode is used for encryption in the Record layer with a 128-bit key. GCM mode provides authenticated encryption in one algorithm so no MAC is used. SHA-256 is used for the PRF used to derive keys from the pre-master key obtained in the handshake protocol.

(f) TLS_CHACHA20_POLY1305_SHA256

- This is a TLS 1.3 ciphersuite, which are the only ones that do not specify handshake algorithms. As for all TLS 1.3 ciphersuites, ECDHE will be used for the handshake (using an elliptic curve agreed in the extensions for the hello messages).
- As for all TLS 1.3 ciphersuites, an authenticated encryption algorithm is used for encryption on the record layer, in this case using the ChaCha/Poly1305 scheme.

cont/...

- SHA-256 is used for the hash in the key derivation function used to derive the keys for use in the record layer.

QUESTION 8

Consider MITM scenarios with TLS (like Superfish) in which a root certificate is added to the client machine.

(a) What are scenarios in which such a MITM may be regarded as legitimate?

(b) How might a root certificate get added to a machine in practice?

(c) In what sense are these scenarios always bad for security, no matter how they are implemented?

(a) In a corporate environment a company may regard it as legitimate to decrypt all company traffic at a network gateway for reasons such as checking for incoming malware and outgoing company secrets. Another scenario is that companies may believe that if they provide a service (such as advertising) this justifies decrypting and altering client traffic.

(b) In a corporate environment the company may insist on a standard operating environment for all company computers. They can then set up the root certificates to include a corporate gateway certificate. Then their corporate gateway can issue certificates for any entity which will be accepted by the client browser. Another way is that pre-loaded operating systems can have root certificates added to those in a standard browser.

(c) The above scenarios break the end-to-end security of the TLS connection. The client and server now need to trust that the intermediate server is secure and will treat their data securely.

QUESTION 9

This question illustrates *padding oracle attacks* on TLS. TLS uses padding on plaintexts with CBC mode encryption for block ciphers (like AES). Padding works by adding at least one byte. The padding is the representation of the number of padding bytes (padding length) preceded by that same value repeated for that number of bytes. Thus possible padding is "00" or "01 01" or "02 02 02" or .... In this question assume that the receiver always outputs an error if the padding is incorrect, and this error explicitly states that a padding error occurs.

(a) How will this padding be checked and correctly removed by the receiver?

cont/...

(b) What happens if the last byte of the block $n-1$, C_{n-1}, of a n-block CBC ciphertext is altered? In what circumstances will the padding in the decryption of the last block, C_n, be correct? (Write an equation for the case when there is one byte of padding.)

(c) Use the above observation to show how a padding oracle attack works to find one byte of the plaintext. How many attempts are required in order to guarantee finding that byte?

(d) How can this attack then be extended to obtain all bytes in $P_n = D(K, C_n)$?

(a) The receiver reads and removes the last byte and interprets it as an integer p. The receiver then removes p further bytes. For secure operation in TLS, it is essential that the receiver also checks that all removed bytes indicate the same value p.

(b) As shown in Lecture 5, slide 17, a change in the last byte of ciphertext block C_{n-1} will result in a random recovered plaintext P_{n-1} and the same change (in terms of bits flipped) in the last byte of P_n. The padding will be correct if the change makes the last byte of P_n equal to 00. (There are also other unlikely circumstances when it will be accepted which depend on the other bytes of P_n, but we ignore those.)

Let us write $D(K, C_n) = X_0 \parallel X_1 \parallel \ldots \parallel X_{15}$, assuming that the encryption algorithm is AES with 16 bytes of output. If we also write $C_{n-1} = C_{n-1,0} \parallel C_{n-1,1} \parallel \ldots \parallel C_{n-1,15}$ then this padding will be accepted if

$$C_{n-1,15} \oplus X_{15} = 00$$

since in CBC mode we have $P_n = D(K, C_n) \oplus C_{n-1}$.

(c) The attacker can now try any different values for $C_{n-1,15}$ until the padding is accepted. Ignoring the unlikely event that padding works for other bytes, the attacker can assume that $C_{n-1,15} = X_{15}$. This will require at most 255 trials, or 128 on average.

(d) Once X_{15} is known, the attacker can find X_{14} by setting $C_{n-1,15} = X_{15} \oplus 01$ and changing $C_{n-1,14}$ until the padding is accepted again. This happens when

$$C_{n-1,14} \parallel C_{n-1,15} \oplus X_{14} \parallel X_{15} = 01 \parallel 01.$$

Thus X_{14} is found after another approximately 128 trials, and the next byte can be found looking for padding $02 \parallel 02 \parallel 02$ and so on.

NTNU Department of Information Security and Communication Technology

TTM4135 Applied Cryptography and Network Security
Semester Spring, 2022

Exercises for TLS 1.3, IPsec and Secure mail

QUESTION 1

Review the definitions of the following concepts. They are things that you would be expected to know in a quiz or exam.

(a) 0-RTT protocols;
(b) host-to-host, gateway-to-gateway and host-to-gateway IPsec architectures;
(c) IPsec transport mode and tunnel mode
(d) Domain Keys Identified Mail (DKIM)
(e) OpenPGP
(f) Signal protocol

QUESTION 2

Compare the handshake protocols in TLS 1.2 and TLS 1.3 as shown on slide 9 of Lecture 15. The client is not permitted to send application data before it receives the finished message from the server.

(a) How much longer must the client wait before sending application data in TLS 1.2 compared with TLS 1.3?

(b) What attacks are possible in TLS 1.2 if the client could send application data in Phase 3 after its own finished message?

(a) In TLS 1.2 the client cannot send application data until after two complete round trips, so not until part of its third message flow to the server. In TLS 1.3 the client can send application data after one round trip, in part of its second message flow. So the client only has to wait about half of the time in TLS 1.3 compare to TLS 1.2.
(b) Note that the client already has the session keys after it sends its finished message in TLS 1.2. Therefore it could use the keys to protect its application data already. However, at this point all of the communication from the server has been in plaintext and the client has no assurance that server messages have not been replayed or constructed by an attacker. In particular the attacker may have changed the chosen ciphersuite and version number to be less secure than the one preferred by the client.

QUESTION 3

Compare IPsec in host-to-gateway architecture with TLS. Consider the following scenarios and discuss which would be most suitable to provide security in each case.

cont/...

(a) You have two applications on your server which you want to secure with independent keys and different security services.

(b) You want to secure a server which has a number of applications and you may want to add new applications in the future without changing the security settings.

(a) IPsec in host-to-gateway architecture only provides security from the host to the gateway and not all the way to the application server. Therefore it is not suitable for this application. However, the two applications can be set up with different certificates and ciphersuites in TLS and thus provide a solution for this scenario.

(b) This time IPsec in host-to-gateway architecture provides exactly what is needed. It provides (the same) security for all the IP traffic which goes across the host to gateway connection and so any new applications will also be protected. However, end-to-end security is not provided for any applications in this way. Using TLS we would have to set up the protocol and certificates for the new application when it is added.

QUESTION 4

Three possible ways to combine encryption and MACs are:

- encrypt first and apply the MAC to the ciphertext;
- apply the MAC first and encrypt plaintext and MAC together;
- apply the MAC and encrypt the plaintext separately.

Which of these is used in the TLS Record Protocol (up to version 1.2) and which is used in the ESP protocol of IPsec? Why is the third not suitable in general? (Remember that the purpose of a MAC is only to provide authentication/integrity and not confidentiality.)

- Slides 41 and 43 of the lecture 15 slides show the packet format for ESP. Here we can see that the data plus the ESP trailer are encrypted first, then the ESP header is added and the whole is authenticated with the MAC. This means that IPsec follows the encrypt-then-MAC process which is today regarded as the correct order.
- Slide 13 of the lecture 14 slides shows the record protocol format. This illustrates that the MAC is computed on all the other data before the payload and MAC are encrypted. This choice may seem intuitively the most natural since the MAC is directly on the data whose integrity should be protected. Unfortunately this order does not always provide the desired security even when the MAC and encryption algorithm are both secure.
- Applying MAC and encryption separately is generally not a good idea. The simple reason is that a MAC is not designed to provide confidentiality. Although we may not see any obvious way to obtain information about the plaintext from the MAC we have no guarantee that it does not leak information.

QUESTION 5

Explain why the lack of interaction in email delivery prevents the possibility to achieve forward secrecy for secure email. Is there a way that forward secrecy could be approximated for email?

cont/...

Email uses the store-and-forward principle. The recipient is *not* assumed to be online and before receiving the encrypted message we cannot assume that the recipient had contact with the sender. Since there is no interaction between endpoints, it must be possible to decrypt the message with only the long-term key. Thus, when the long-term key is compromised later, any captured encrypted mail must be decryptable by the attacker.

There are some ways that forward secrecy can be achieved to some degree.

- As mentioned in the lecture, link encryption is often used for mail while in transit. Since this link encryption uses TLS, forward secrecy can be achieved for an eavesdropper on those links. This does not help if the attacker controls one of the links.
- We can regularly update the public encryption key and corresponding private decryption key regularly, for example on a daily basis. There are some cryptographic schemes to help do this systematically by essentially adding the date into the public key. This does not help protect today's traffic if the daily decryption key is compromised before it is updated. Also it means that old messages cannot be decrypted.
- The above impossibility argument for complete forward secrecy assumes that the long-term key is static. If the long-term key is updated every time a message is received, such that it can no longer be used to decrypt that same message, then the argument breaks down. Recently some protocols have been designed to achieve this property, but they are not very efficient.

Finally we may note that instead of email, instant messaging allows interaction which can be used to provide forward secrecy in protocols like Signal.

QUESTION 6

In hybrid encryption, such as used in PGP, is it better to have the public key encryption or the symmetric key encryption to be the stronger of the two?

Breaking the long-term decryption key leads to breaking all symmetric keys due to the lack of forward secrecy. However, breaking one symmetric key breaks only one session. Therefore we should ensure that the (believed) strength of the public key encryption is at least as good as the private key encryption.

QUESTION 7

End-to-end security and link security are two ways of providing network security. What are some of the advantages and disadvantages of each? What protocols, or configurations, are available to provide each of these types of security in (i) email (ii) IPsec?

Note that we often think of confidentiality when talking about end-to-end security, but the arguments below are relevant also for integrity/data authentication.

With end-to-end security the endpoints (users) control their own level of security and can generate their own keys. They do not need to trust any third party for their security. However, information required to transmit the messages across the network (headers) cannot be encrypted end-to-end, since it needs to be used by network nodes. Leakage of such *metadata* can be very useful for an attacker.

cont/...

With link security the header data can be protected, since the nodes typically carry traffic between many different sources and destinations. The network operators may have better capability and resources than the endpoint users to properly set up security.

For email: PGP security is end-to-end, DKIM is domain to domain, and STARTTLS is link by link.

For IPsec we have different architectures. Only the host-to-host architecture is end-to-end. Commonly gateway-to-gateway architecture is used.

QUESTION 8

The messaging protocol Signal uses pre-computed Diffie-Hellman keys to protect the *first* communicated message in any conversation. A client A pre-computes many $t_i = g^{x_i}$ values which are stored on the Signal server. When another client B starts a new conversation with A:

- B is given a previously unused pre-computed key of A, $t_i = g^{x_i}$, and then t_i is deleted from the server;
- B chooses an ephemeral Diffie–Hellman private key x_B;
- B computes a message key k as a hash of $g^{x_i x_B}$;
- B sends the first message to A protected by k and also sends g^{x_B};
- When A receives the message, she uses x_i to recompute k and recover the first message and then deletes x_i.
- For the next message A computes a new ephemeral Diffie–Hellman value which she combines with g^{x_B} to form the next message key.

Since the number of new conversations that will be started is not predictable, Signal has a fallback mechanism to use the last available pre-computed key for many conversations until the supply of pre-computed keys is replenished. Signal suggests keys be replenished once a week, or once a month.

Discuss how this process influences forward secrecy for the first message in each conversation. Include a discussion of whether the pre-computed values should be classed as long-term or ephemeral keys.

The pre-computed keys do not fit the definition of either long-term keys (which should last throughout the use of the protocol) or of ephemeral keys (which should exist only when they are used to set up one key).

If we define the pre-computed keys as *long-term*, then the definition of forward secrecy should allow an attacker to obtain these keys and then compute the key k used to protect the first message. (Due to the Diffie–Hellman ratcheting the later messages are protected by different keys and are not vulnerable.) If we define the pre-computed key to be *ephemeral* then we can say that forward secrecy is provided by the Signal protocol. This shows that when we define forward secrecy we need to be more precise about key lifetime and consider the risk of using medium-term keys which may last much longer than usual ephemeral keys.

If an attacker is able to exhaust the pre-computed keys by starting many fake conversations with A then the fallback key will be used to protect all subsequent first messages and forward secrecy is in practice lost during the window until the pre-computed keys are replenished.
