NANYANG TECHNOLOGICAL UNIVERSITY

SEMESTER I EXAMINATION 2017-2018 SUGGESTED SOLUTION

MH4311 – Cryptography

December 2017 TIME ALLOWED: 2 HOURS

INSTRUCTIONS TO CANDIDATES

1. This examination paper contains **FOUR (4)** questions and comprises **FOUR (4)** printed pages.
2. Answer all questions. The marks for each question are indicated at the beginning of each question.
3. Answer each question beginning on a **FRESH** page of the answer book.
4. This is a **RESTRICTED OPEN BOOK** exam. You are allowed to bring into the examination hall **ONE (1)** piece of A4-size paper written or printed on both sides.
5. Candidates may use calculators. However, they should write down systematically the steps in the workings.

Solutions provided by: Brandon Goh – bgoh008@e.ntu.edu.sg

Question 1. Hash function and MAC (20 marks)

(a) SHA-256 is applied to hash a message with length of 3000 bits. How many compression function operations are needed in the hashing?

(5 marks)

(b) HMAC-SHA-256 is applied to compute the authentication tag of a message with length of 3000 bits. How many compression function operations are needed? (5 marks)

(c) At a website, each user's password P is hashed together with a salt S into a password image PI. The password images are stored at the website. Suppose that each salt is a 256-bit random number. The following algorithm is used to hash the password and the salt:

$$\begin{aligned}
t1 &= SHA\text{–}256(P) \oplus S; \\
t2 &= SHA\text{–}256(t1) \oplus P; \\
t3 &= SHA\text{–}256(t2) \oplus t1; \\
t4 &= SHA\text{–}256(t3) \oplus t2; \\
PI &= t3 || t4;
\end{aligned}$$

Is this password hashing algorithm secure? Please justify your answer.

(10 marks)

Answer

(a) SHA-256: 512-bit block size

$$3000 + 1 + X + 64 = \alpha \cdot 512$$

3000: Bits allocated for message
1: 1-bit allocation for start of padding
X: Number of '0's required for padding
64: Length of message stored in 64 bits.

Solving the equation for the smallest α and X will tell us that
$\alpha = 6$
$X = 7$

$\therefore$ 6 compression functions are required $\square$

(b) Note that HMAC uses the following compression function:

$$MAC_k(M) = Hash((k' \oplus outerpad)||Hash((k' \oplus innerpad)||M))$$

The values of innerpad and outerpad are not required for this question. Instead, you should know that k' is a key that has been padded with '0's to match the block size (i.e. SHA-256: 512-bit block size).
$\Rightarrow ((k' \oplus innerpad)||M)$ has size **3512** bits.

Hashing 3512 bits using SHA-256 requires us to solve the equation:

$$3512 + 1 + X + 64 = \alpha \cdot 512$$

3512: Bits allocated for message
1: 1-bit allocation for start of padding
X: Number of '0's required for padding
64: Length of message stored in 64 bits.

Solving the equation for the smallest α and X will tell us that
$\alpha = 7$
$X = 7$

$\therefore$ 7 compression functions are required.

*Note: The calculation is not complete at this stage.

The output of SHA-256 is 256-bits, as the number implies. For simplification, we shall denote it with β.
$\Rightarrow ((k' \oplus outerpad)||\beta)$ is $512 + 256 =$ **768** bits long.

Hashing 768 bits using SHA-256 requires us to solve the equation:

$$768 + 1 + X + 64 = \alpha \cdot 512$$

Solving the equation for the smallest α and X will tell us that
$\alpha = 2$
$X = 191$

$\therefore$ 2 compression functions are required.

In total, 9 compression functions are required to hash 3000 bits in HMAC-SHA-256. $\square$

(c) This is a very simple question if you take the effort to rewrite the equations given.

Now, to obtain P we can rewrite the equations into the following form.

$$\begin{aligned} P &= t2 \oplus \text{SHA–256}(t1) \\ &= t4 \oplus \text{SHA–256}(t3) \oplus \text{SHA–256}(t1) \end{aligned}$$

We are further told that the salt is a 256-bit random number, which means S is 256 bits long. From here, we analyse two equations. Specifically:

$$\begin{aligned} t1 &= SHA - 256(P) \oplus S \\ t3 &= SHA - 256(t2) \oplus t1 \end{aligned}$$

This tells us that both $t1$ and $t3$ are 256 bits long.

The website stores login credentials as a password image PI. Assume that PI is n bits long. Since $t3$ is already known to be 256 bits long, $t4$ must be $n - 256$ bits long. Note that PI is a mere concatenation of $t3$ and $t4$, which means we have instantly obtained these intermediate values.

We still lack the value of $t1$, crucial for us to compute the value of P. Looking at the equations given, we can rewrite the following:

$$\begin{aligned} t3 &= \text{SHA–256}(t2) \oplus t1 \\ \Rightarrow t1 &= \text{SHA–256}(t2) \oplus t3 \end{aligned}$$

We find that to compute $t1$, $t2$ needs to be obtained. Again, the following equation can be rewritten:

$$\begin{aligned} t4 &= \text{SHA–256}(t3) \oplus t2 \\ \Rightarrow t2 &= \text{SHA–256}(t3) \oplus t4 \end{aligned}$$

From this point, we can obtain the values of $t2$ and $t1$. Note that the SHA-256 values can easily be calculated by using a program (as the algorithm is standardised). Substituting the respective values will now allow us to obtain P in its plaintext form. This demonstration shows you that this algorithm is extremely insecure (and the little effort required from an attacker to obtain passwords). □

Question 2. AES (15 marks)

(a) In AES, the irreducible polynomial with binary coefficients, $x^8 + x^4 + x^3 + x + 1$, is used to define $GF(2^8)$. Find the inverse of 5 in this field. (10 marks)

(b) Suppose that you are required to implement AES to encrypt files on your computer. The encryption and decryption is provided by the user. When a user inputs the decryption key, your program should check whether the key is correct or not, and the decryption is performed only when the key is correct. Briefly explain how to implement it. (5 marks)

Answer

(a) **NEVER attempt AES calculation questions in decimal form. Always convert to polynomial form first!**

5 in the field $GF(2^8)$ is expressed as the polynomial $x^2 + 1$. Calculate the inverse as follows:

$$x^8 + x^4 + x^3 + x + 1 \mod x^2 + 1 = (x^6 + x^4 + x)(x^2 + 1) + 1$$

$\therefore (x^2 + 1)^{-1} = (x^6 + x^4 + x)$.
$x^6 + x^4 + x$ expressed in hex(adecimal) form is $\{52\}$. □

(b) Hash the key and store the digest together with the ciphertext. □

Question 3. RSA (20 marks)

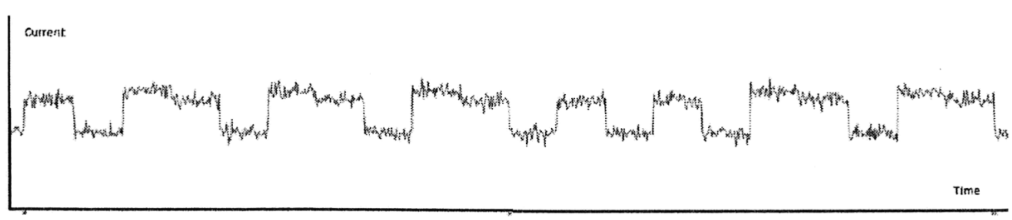
(a) In a toy RSA encryption scheme, the public key (n, e), the private key is d. It is given that $n = 3149 = 47 \times 67$. You are required to generate a pair (e, d). (10 marks)

(b) Factorise $n = 84923$ using the following relations: (10 marks)

$$\begin{aligned}
345^2 &\mod n = 2 \times 17^2 \times 59 \;; \\
513^2 &\mod n = 2^4 \times 3 \times 5 \times 7 \;; \\
519^2 &\mod n = 2^8 \times 3 \times 19 \;; \\
520^2 &\mod n = 7^2 \times 11 \times 29 \;; \\
527^2 &\mod n = 2^6 \times 3 \times 5^2 \times 7
\end{aligned}$$

(c) In an RSA implementation, the random number used in OAEP is generated by applying SHA-256 to hash the plaintext. Is this implementation secure? Please justify your answer. (10 marks)

(d) The decryption of a toy RSA is implemented on a computational device. The private key d is ~~10~~ **8**-bit. Right-to-left square-and-multiply algorithm is used in the implementation. Chinese remainder theorem and RSA blinding are not used in the implementation. During the decryption, the electrical current consumed by the device is measured and is shown in the following diagram (the horizontal axis is time, the vertical axis is the electrical current). What is the value of d? Briefly explain how you obtain the value of d. (10 marks)



Answer

(a)

$$\begin{aligned} n &= p \times q \\ &= 47 \times 67 \\ \varphi(n) &= (p-1)(q-1) \\ &= 46 \times 66 \\ &= 3036 \end{aligned}$$

We have obtained $\varphi(n)$ and so take e to be 5.

$$\begin{aligned} &\gcd(\varphi(n), e) = 1 \\ \text{(Check)} \quad &\gcd(3036, 5) = 1 \end{aligned}$$

Now calculate for the value of d.

$$\begin{aligned} ed &\equiv 1 \mod \varphi(n) \\ 3036 &= 607 \times 5 + 1 \\ \Rightarrow 1 &= 3036 - 607 \times 5 \\ \because d &= e^{-1} \\ \therefore d &= 5^{-1} \\ &= -607 \\ &= 2429 \mod 3036 \end{aligned}$$

For $n = 3149$, we can take a pair (e, d) as $(5, 2429)$. □

(b) This time the question on Dixon square algorithm is very easy. The key trick to this question is to make sure that for any number (> 2) of formulas that you multiply together, the **exponent** for each prime number on the RHS **must be even**.

- First equation, 59 does not appear elsewhere so this equation is not considered.
- Third equation, 19 does not appear elsewhere so this equation is not considered.
- Fourth equation, 29 does not appear elsewhere so this equation is not considered.

Two equations are left, so it is trivial that these two must be used to obtain the required intermediate value.

$$n = 84923$$

$$\begin{aligned}
513^2 \cdot 537^2 \mod n &= 2^{10} \times 3^2 \times 5^4 \times 7^2 \\
(513 \cdot 537)^2 \mod n &= (2^5 \times 3^1 \times 5^2 \times 7)^2 \\
(513 \cdot 537)^2 \mod n &= (275481)^2 \mod n \\
&= (3 \times 84923 + 20712)^2 \mod n \\
&= (3 \times 84923)^2 + 2 \times (3 \times 84923 \times 20712) + 20712^2 \mod n \\
&= 20712^2 \mod n \\
20712^2 \mod n &= (2^5 \times 3^1 \times 5^2 \times 7)^2 \\
20712 \mod n &= (2^5 \times 3^1 \times 5^2 \times 7) \\
&= 16800
\end{aligned}$$

Using 20712 and 16800, we can obtain both factors easily at this stage. (Steps omitted since you can use a calculator to obtain the answer)

$$\gcd(20712 + 16800, 84923) = 521$$
$$\gcd(20712 - 16800, 84923) = 163$$

Two factors, 521 and 163 are the values obtained from the algorithm. You can easily check that $521 \times 163 = 84923$. $\square$

(c) We know that k_0, k_1, n are publicly known and G, H are standard functions. There is also mention that the random number r in OAEP is changed to use the digest of SHA-256(P). If we write the equations to obtain m and r from X and Y, then we see that it can be written in the following form.

$$r = Y \oplus H(X) \tag{1}$$

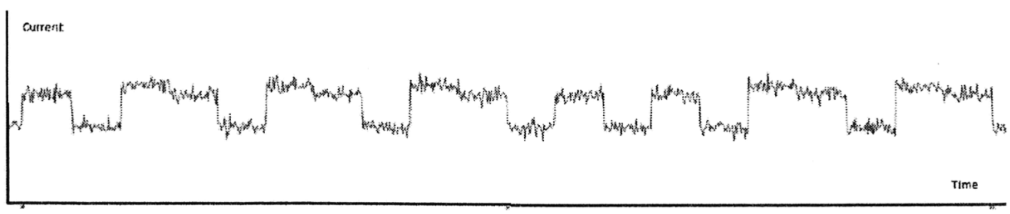
And the equation for m.

$$m||000 = X \oplus G(r) \tag{2}$$

When considering both equations (1) and (2), for any arbitrary plaintext P, m, r, X and Y are completely different. However, when duplicate plaintexts are sent, r is **NOT** random. Furthermore, X and Y will be the same. The attacker only needs to look out for duplicate X and Y for him to easily break the modified OAEP encryption (Reversal is simple enough). □

(d) At first sight, one might assume that the high current usage indicates a value of '1' while a low current usage indicates a value of '0'. However, we are given that it is a **8**-bit key. It is also crucial to note that during decryption, there are times where there are no computations taking place (executing code not relating to square-and-multiply). No device can operate at '0' current, so it can be safely assumed that the low current usage is the baseline current required for operation of the computational device.

With that, we focus on the peaks of the graph. In fact, there are 8 of it. With careful observation, you will notice that some peaks are longer than others, indicating a longer period where a higher current is drawn.



Going back to the square-and-multiply algorithm, when the value of the bit is '0', only 1 multiplication operation is performed. When the value of the bit is '1', 2 multiplication operations are performed. When operations are performed, power is required to perform the computations. Hence, if the bits were written in the order shown from the graph, it will give us this result: **01110011**.

This is not the final answer since the question states that the square-and-multiply operation is performed in a right-to-left order. Therefore, the bits must be reversed and so the private key d has the value **11001110** in binary or {CE} in hex. (Leaving your answer in decimal, binary or hex is <u>OK</u> but state the base where necessary). □

Question 4. Digital Signature Algorithm (25 marks)

(a) A 96-bit private key is used in the Digital Signature Algorithm. Please develop an efficient attack to break it. (10 marks)

(b) In the Digital Signature Algorithm, what is the risk if the same random number is reused to sign different messages? (5 marks)

(c) In the Digital Signature Algorithm, the random number used to sign message M is generated by applying SHA-256 to hash the key together with the message digest of M. Is this implementation of the Digital Signature Algorithm secure? Please justify your answer. (10 marks)

Answer

(a) A 96-bit private key is incredibly weak because it can be solved using discrete logarithm algorithms. Then there are 4 algorithms to choose from. Pollard Rho's algorithm is too complex in this scenario. In any case, Pohlig-Hellman has the lowest complexity class among the 4 algorithms (i.e. time efficient).

Pohlig-Hellman Algorithm:

Suppose that $n = p - 1 = p_1 p_2 \cdots p_i$
Find $x_i \mod p_i \ \ \forall i$
Solve for x using Chinese Remainder Theorem
Then let $x = u_i \cdot p_i + v_i$, where $v_i = x \mod p_i$

$$\begin{aligned} g^x &\equiv b \mod p \\ (g^x)^{\frac{n}{p_i}} &\equiv b^{\frac{n}{p_i}} \mod p \\ (g^{u_i \cdot p_i + v_i})^{\frac{n}{p_i}} &\equiv b^{\frac{n}{p_i}} \mod p \\ (g^{v_i})^{\frac{n}{p_i}} &\equiv b^{\frac{n}{p_i}} \mod p \\ (g^{\frac{n}{p_i}})^{v_i} &\equiv b^{\frac{n}{p_i}} \mod p \end{aligned}$$

Take $g' = g^{\frac{n}{p_i}}$, $b' = b^{\frac{n}{p_i}}$, then

$$(g')^{v_i} \equiv b' \mod p$$

From here, v_i can be obtained using Shank's baby-step giant-step algorithm. □

(b) If the same number k is reused to sign different messages, then we can set up multiple simultaneous equations to recover the private key x. $\square$

(c) When the random number k is changed to the value of SHA-256(M), it will still be secure. SHA-256(M) is a 256-bit output that is fixed, depending on the message. Typically, this means that the number of variables to recover will decrease. However, the nonlinear equations obtained are extremely complicated and solving it to obtain the private key is not as simple. $\square$

END OF PAPER