

Approved 48hr Extension Request

We've processed your Assignment extension request FORM-AEX-260413



no-reply@qut.edu.au

To Ash Phillips

[Reply](#) [Reply All](#) [Forward](#)

Sun 16/10/2022 9:34 PM

Click here to download pictures. To help protect your privacy, Outlook prevented automatic download of some pictures in this message.



Hi Ashley,

Thank you for your assignment extension request (**FORM-AEX-260413**).

We have approved your request and the due date for your assignment **Assignment 2 : Symmetric-key Cryptography**, for unit CAB340 has been extended by 48 hours from the original due date. If your unit outline does not specify that your assignment is eligible for an extension, this confirmation email is not valid and unless you submit by the original due date, the late assessment policy will apply.

You are responsible for ensuring that this assignment is eligible for extension before submitting it after the original due date. Check your [unit outline](#) for eligibility.

Be aware that a copy of this email is kept on file. You should not alter this email in any way. Email notifications that have been altered or differ in any way from the original may result in an allegation of student misconduct as set out in the [Student Code of Conduct](#).

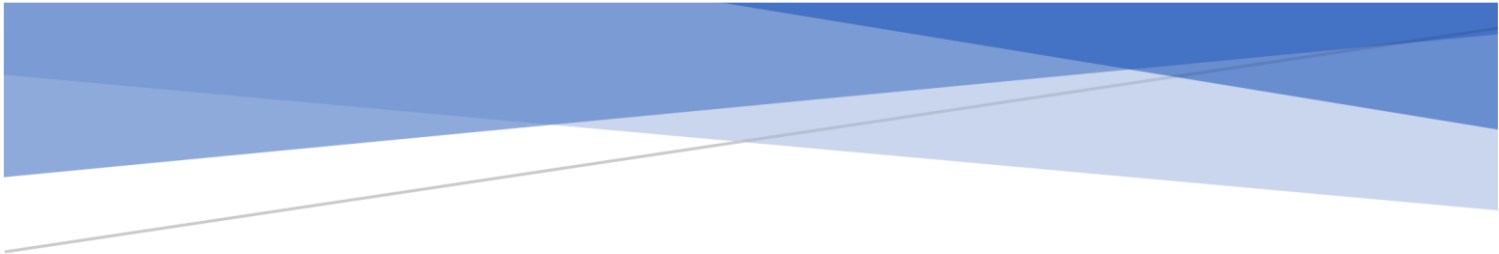
Need extra support? You can access free, confidential [counselling with qualified professionals](#). We also offer [planning and support if you have a disability, injury or health condition](#) that affects your ability to study. If you are struggling to meet your university commitments, [academic support](#) is also available.

Have a question? You can contact us by email or phone. We're also available to help you on campus or via online chat. Visit the [HIQ website](#) for our contact details and opening hours.



You have received this email because you have submitted an assignment extension request. View our [Privacy and Security statement](#).

Ref ID: 10477659 FORM-AEX-260413



ASSESSMENT 2: STREAM CIPHERS, BLOCK CIPHERS, MODES OF OPERATION, APPLICATIONS

CAB340: Cryptography

Ash Phillips, n10477659

Extended Due Date: 19/10/22, 11:59 pm

1.0. LFSRs Found in Real Systems

(a)

The brute force complexity is $2^{40} \approx 1.0995 \times 10^{12}$, rounded to 4 decimal places.

(b)

The remaining bit in each register is set to 1 to initialise the LFSR with a seed value to prevent the possibility of a null cycle (a stream of all zeros).

(c)

The maximum possible period for each LFSR is:

$$\text{For } L_1 \text{ with length 17, } 2^{17} - 1 = 131,071$$

$$\text{and for } L_2 \text{ with length 25, } 2^{25} - 1 = 33,554,431$$

(d)

The maximum possible period for the chosen cipher is:

$$(2^{17} - 1) \times (2^{25} - 1) = \mathbf{4.3980 \times 10^{12}}$$
 rounded to 4 decimal places

For the alternatives i. and ii., the maximum possible period for each LFSR and overall ciphers are:

- i. For L_1 and L_2 with length 21, $2^{21} - 1 = 2,097,151$

For this cipher the max period is:

$$2(2,097,151) = \mathbf{4.1943 \times 10^6}$$
 rounded to 4 decimal places

- ii. For L_1 with length 7, $2^7 - 1 = 127$

and for L_2 with length 35, $2^{35} - 1 = 34,359,738,367$

For this cipher the max period is:

$$127 \times 34,359,738,367 = \mathbf{4.3637 \times 10^{12}}$$
 rounded to 4 decimal places

As calculated, cipher i. has a much shorter maximum period compared to the other options, meaning it will cover less data. Also, by using these matching LFSR lengths there is the risk that each LFSR may produce the same data streams and cancel out, creating a weak-key. Cipher ii. has a slightly smaller maximum period compared to the chosen cipher, and would not be much different in coverable data, however, as the LFSR with length 7 has a small maximum period it would be easiest to break, making this less useful. From this, we can see why the designers chose the cipher with register lengths 17 and 25.

(e)

To ensure security of the cipher for large video encryption it needs to be able to encrypt between 2^{35} and 2^{36} bits of data (<4.7GB), roughly $<4.0372 \times 10^{10}$ bits, within its period so the key does not repeat. As can be seen from the above results, cipher i. has the period 4.1943×10^6 , which is less than the required size, and therefore could not be used to encrypt securely as it would take more than one period to do so. The periods of the other two ciphers proposed exceed the required data size and therefore can be used securely for the intended purpose.

2.0. Do Block Modes Protect Against Cipher Linearity?

(a)

Let:

- $(c_1, c_2, \text{ and } c_3)$ = ciphertext blocks
- $(p_1, p_2, \text{ and } p_3)$ = plaintext blocks
- K = key

Then:

$$c_1 = p_1 \oplus \binom{n}{t}(K)$$

$$c_2 = p_2 \oplus \binom{n}{t}(K)$$

$$c_3 = p_3 \oplus \binom{n}{t}(K)$$

(b)

To reduce the attack on Hill-CTR to Hill-ECB, assuming the counter is known, we can XOR the plaintexts and counters that are presents in the triplets of plaintext, counter, and ciphertext to then create pairs of the new XORed plaintexts and ciphertexts. From there, it has been reduced to Hill-ECB and can be broken in a KPA attack.

(c)

Let:

- $(c_1, c_2, \text{ and } c_3)$ = ciphertext blocks
- $(p_1, p_2, \text{ and } p_3)$ = plaintext blocks
- K = key
- IV = initialised vector

Then:

$$c_1 = K(p_1 \oplus IV) \bmod 26$$

$$c_2 = K(p_2 \oplus c_1) \bmod 26$$

$$c_3 = K(p_3 \oplus c_2) \bmod 26$$

(d)

To reduce the problem of a KPA against Hill-CBC to Hill-ECB a similar approach to (b) can be taken. Here, there is no counter so we instead XOR the plaintext with the ciphertext to created pairs or new XORed plaintext and ciphertext. This has now been reduced to Hill-ECB and can be broken.

3.0. Message Authentication Codes

(a)

i.

The hash formula for a single-block message m is:

$$H(m) = h$$

If we replace $H(m)$ with CBC instead, which takes different arguments (h, K), we have this (assuming message block 1):

$$CBC(h, K) = m \oplus IV$$

Now as h , IV , and K are given, we can rearrange to solve for m :

$$m = CBC(h, K) \oplus IV$$

ii.

To extend the attack past just a single block, IV needs to be replaced for the hash of the previous block, h_{n-1} :

$$m_n = CBC(h_n, K) \oplus h_{n-1} \quad \text{where } n \geq 1$$

(b)

i.

In this form of padding for CBC-MAC, adversaries would simply be able to construct another message with additional '0' bits that then has the same pad as the original message and then will be given the same tag for the key, a length extension attack.

ii.

For this instance, the same concept can be applied in that additional '0' bits should be added, but in the form of a new "dummy" block. If this does not work, the adversary should instead use unambiguous padding by appending a single '1' bit followed by as few '0' bits required to complete the block.

iii.

A different, and more secure, padding method would be to use CMAC, which uses three keys (k_1 , and randomly generated k_2 and k_3) to encrypt the message. If the message is not a multiple of the block length, normal padding will be applied and then this block will be XORed with k_2 . If the message is a multiple of the block length, nothing is padded but it is XORed with k_3 . Due to this, if the adversary wished to construct a message with the same tag, they would need to know k_2 or k_3 , stopping the ability to conduct a length extension attack.

(c)

i.

The attack present in tutorial 6 is based on the ability to append to the message and still return the same valid MAC tag as the original message. For XCBC this length extension attack no longer works as the prefix IV is not used and multiple keys, k_1 ,

k_2 , and k_3 , are used for message encryption. For example, where k_i can be either k_2 or k_3 , m_1 gets padded to $m_1 \oplus k_i$ and another message $m_1 || m_2$ gets padded to $m_1(m_2 \oplus k_i)$, we can see $m_1 \oplus k_i$ is not a prefix of $m_1(m_2 \oplus k_i)$. For an attacker to construct a prefix they would need to find $m_1 \oplus k_i$. As keys k_2 and k_3 are derived from k_1 using a pseudo-random number generator, the attacker can only guess at what these keys could be, essentially becoming a brute force attack, removing the ability to perform a length extension attack.

ii.

If k_2 and k_3 were equal, then m_n would be encrypted the same way regardless of padding in the final XOR encryption. This gives an attacker the chance of generating a valid message, MAC pair. The attacker could request the MAC of a message m and get a valid tag, T , so the pair $(\text{Pad}(m), T)$ could also be valid.

4.0. Using Block Ciphers for Hash Functions

(a)

To use AES-128 as a compression function using the Merkle-Damgård construction the following can be done:

Let:

- $E_k(m)$ = a block cipher, AES
- k = key
- m = message block

Then:

$$f(W_1) = E_{m_1}(IV) \oplus IV$$

$$f(W_2) = E_{m_2}(W_1) \oplus W_1$$

:

$$f(W_n) = E_{m_n}(W_{n-1}) \oplus W_{n-1} \quad \text{where } n \geq 1$$

As shown, this hash function occurs for each stage of the AES with the message used and the previous hash used as the two inputs (except for W_1 , which instead uses IV instead of the previous).

(b)

Yes, the one-way requirement is met as the input for each stage of the function is the output of the previous stage, making this function one-way and secure.

(c)

While the modern hash functions have 256-bit outputs, the AES-128 would have 128-bit outputs instead, so the collision resistance would be lower than that of a 256-bit value, making it less secure, but still viable.

5.0. Proofs of Work

(a)

The expected amount of work measured in the number of individual evaluations of H required to find a qualifying Nonce is 2^δ .

(b)

The size of the output function for such PoW depends on the time taken, t , such that:

$$\delta \times \frac{2^{32}}{t}$$

Where the probability that a block is found is:

$$\frac{1}{2^{32}}$$

(c)

No, such a H could not work for the PoW application as it is a fundamental requirement that H is collision resistant and, as such, is virtually impossible to find two values with the same hash. It is infeasible that alterations can be made to an old block without invalidating all following blocks.

(d)

Yes. CRCs were designed to detect any accidental changes and transmission errors to the data (i.e., detect damaged files), ensuring message authenticity, though, they cannot protect files from modification and are therefore not secure. The linearity of the function however allows bits to be flipped (or XORed) in an encrypted frame and then alter the CRC so that integrity checks succeed for the modified frame(s). Therefore, using this linear function as H will still work for the PoW application.

6.0. References

Boneh, D. (2017). *MAC Padding*. CourseRa.

<https://www.coursera.org/lecture/crypto/mac-padding-vTbf0>

CryptoGrinders. (2018). *Consensus in the Blockchain Part 2 – Proof of Work*.

Medium. <https://medium.com/@cryptogrinders/consensus-in-the-blockchain-part-2-proof-of-work-d2baba9816a5>

Miner Daily. (2022). *How are Bitcoin's Difficulty and Hash Rate Calculated?*.

MinerDaily. <https://minerdaily.com/2021/12/31/how-are-bitcoins-difficulty-and-hash-rate-calculated/>

Paar, C. (2021). *Introduction to Cryptography II*. RUB Faculty of Computer Science.

<https://informatik.rub.de/studium/lehrveranstaltungen/Krypto2/>

Straub, A. (2017). *Sketch of Lecture 34*. ArminStraub.com.

<https://arminstraub.com/downloads/teaching/cryptography-spring17/lecture34.pdf>

Tutorials Point. (2022). *How is Hill Cipher Vulnerable to Chosen Plaintext Attack?*.

Tutorials Point. <https://www.tutorialspoint.com/how-is-a-hill-cipher-vulnerable-to-chosen-plaintext-attack>

Will, D. (2021). *AES CBC Mode – Chosen Plaintext Attack*. DerekWill.com.

<https://derekwill.com/2021/01/01/aes-cbc-mode-chosen-plaintext-attack/>