FIT3031
Information and Network Security
Assignment 1 - First Semester 2017

Submission Guidelines

- **Deadline:** Assignment 1 is due on Friday 7th April 2017, 11:55PM.
- **Submission File Format:** Only submitted PDF files are accepted. On various text editor software you can use "Save as PDF" option or use free converters to convert your file to PDF.
  Note: Do not submit a compressed version of the PDF file or a compression of multiple files. Such submissions may risk losing partial or complete assignment marks.
- **Submission Platform:** Caulfield - Electronic submission via Moodle.
- **Required Student Information:** Please include your name and student id within the main PDF file.
- **Filename Format:** Assign-1_FirstName_LastName_SUID.pdf
- **Late Submission Policy:** Submit a special consideration form (available on moodle) to formally request a late submission.
- **Late Submission Penalty:** A late submitted assignment without prior permission will receive a late penalty of a 5% deduction per day (including Saturday and Sunday) or part thereof, after the due date and time.
- **Plagiarism:** It is an academic requirement that your submitted work be original. Zero marks will be awarded for the whole submission if there is any evidence of copying, collaboration, pasting from websites, or copying from textbooks.
  Note: Plagiarism policy applies to all assessments.

Marks

- This assignment is worth **20%** of the total unit marks.
- The assignment is marked out of **100** nominal marks.
- For example if you obtain **60** marks for this assignment, it will contribute $\frac{60}{100} \times 20 = 12$ marks to your final unit grade.

Notes

- You can use the WolframAlpha web site to perform any calculation necessary for this assignment (`http://www.wolframalpha.com/`). A "Pro computation time" is **not** required to answer the questions and if you receive such message from the website it is likely that you are not performing the calculations right or the right calculations.
- In questions related to modular arithmetic do not write the answers in scientific notation, you need to provide all of the digits for any requested parameter (similar to provided values).
- Try small examples (from lecture notes or other resources) to make sure you are using the proper format for the web site and the correct equation before trying the given values.

1. We consider the security services:

   - confidentiality,
   - integrity,
   - authenticity, and
   - non-repudiation

   for a variety of simple protocols using symmetric or asymmetric cryptosystems, in a network environment where the two participants, Alice and Bob, are communicating in the presence of an adversary (attacker with malicious intent) Eve.
   In the following scenarios for each of the four security services, describe whether or not it is achieved, and in either case give reasons.

   (a) $A \to B : m||h(m)$
   (b) $A \to B : m||MAC(K_{AB}, m)$
   (c) $A \to B : E(PU_B, m)$
   (d) $A \to B : m||E(PR_A, h(m))$
   (e) $A \to B : E(K_{AB}, m||h(m))$

   Notation: m is the message, $h()$ a cryptographically strong hash function, $E()$ an encryption algorithm corresponding to the specified keys (symmetric for secret key, asymmetric for public/private key), $MAC()$ is a message authentication code algorithm, K_{XY} is a secret key shared between entities X and Y (symmetric algorithm), PU_X is the public key and PR_X the private key of entity X (asymmetric algorithm), $||$ is concatenation function, $X \to Y$ specifies X sends to Y.

   [20 Marks]

   (a) $A \to B : m||h(m)$
      i. Confidentiality: not achieved as message m is transmitted in clear text.
      ii. Integrity: not achieved as the adversary Eve can replace the message with m' and attach $h(m')$ hence $E \to B : m'||h(m')$
      iii. Authenticity: not achieved as there is no way for Bob to verify that the message is coming from Alice
      iv. Non-repudiation: not Achieved as Alice can deny sending a message (no way for Bob to prove the message was sent by Alice)

   (b) $A \to B : m||MAC(K_{AB}, m)$
      i. Confidentiality: not achieved as message m is transmitted in clear text.
      ii. Integrity: is achieved as only Alice and Bob know the value of K_{AB}, and Eve cannot modify or fabricate a message m' and attach a valid $MAC(K_{AB}, m')$ to that message
      iii. Authenticity: is achieved to some extent, that is either Alice or Bob is the creator of the message m
      iv. Non-repudiation: not achieved as the key K_{AB} is shared and Bob can fabricate a message m' and produce a valid $MAC(K_{AB}, m')$ for it

   (c) $A \to B : E(PU_B, m)$
      i. Confidentiality: is achieved as only Bob can decrypt the message with his private key PR_B
      ii. Integrity: not achieved as Eve can tamper with $E(PU_B, m)$ or replace the encrypted message $E(PU_B, m)$ with $E(PU_B, m')$ (as Bob's public key is public) and Bob cannot verify if the message is tampered with.
      iii. Authenticity: not achieved as there is no way for Bob to verify the message is sent by Alice
      iv. Non-repudiation: not achieved as there is no way for Bob to verify the message is sent by Alice hence Alice can deny sending the message

   (d) $A \to B : m||E(PR_A, h(m))$
      i. Confidentiality: not achieved as message m is transmitted in clear text.
      ii. Integrity: is achieved (digital signature) or $h(m)$ depends on m and encrypted by private key of Alice, as this is only known by Alice no one else can produce $E(PR_A, h(m))$ hence any tampering with the message m or fabrication will be detected.

iii. Authenticity: is achieved as only Alice can generate $E(PR_A, h(m))$ and Bob can verify this by generating $h(m)$ from the received datagram and decrypt $E(PR_A, h(m))$ using public key of Alice PU_A and compare the two values.

iv. Non-repudiation: Achieved, given $h(m)$ depends on m and PR_A is only known by one entity: Alice, only she can produce a valid signature for the message. She cannot deny sending the message since no one else can produce the signature for the message. This can be verified by a third party in case of a dispute between Alice and Bob.

(e) $A \rightarrow B : E(K_{AB}, m||h(m))$

i. Confidentiality: is achieved as the message is encrypted

ii. Integrity: is achieved, if Eve tampers with encrypted datagram, Bob can detect by generating $h(m)$ from the message part of the decrypted datagram and compare it with attached $h(m)$, if it matches the received datagram has not been tampered with.

iii. Authenticity: is achieved to some extent as Alice and Bob are the only entities who can generate $E(K_{AB}, m||h(m))$, however neither of the two can be uniquely identified as the origin of the message.

iv. Non-repudiation: not achieved as both Alice and Bob can generate $E(K_{AB}, m||h(m))$, (in case of a dispute any third party cannot distinguish between Alice and Bob as both have access to the shared key).

2. Alice and Bob agree to communicate privately via a protocol based on RC4, but they want to avoid using a new secret key for each transmission. Alice and Bob privately agree on a 128-bit key **k**. To encrypt a message **m** consisting of a string of bits, the following procedure is used:

(a) Choose a 16-bit seed (random value): **r**

(b) Generate the ciphertext $\mathbf{c} = \mathbf{RC4[h(r)||k]} \oplus \mathbf{m}$ for the message, where $h(r)$ is a cryptographically strong hash function that produces 128-bit message digest of the input

(c) Send the bit string $\mathbf{h(r)||c}$

Answer the following:

(a) Suppose Alice uses this procedure to send a message to Bob. Describe how Bob can recover the message **m**?

(b) After how many messages should Alice and Bob change the secret key **k** to avoid RC4 key being repeated?

(c) Does increasing the bit size of the seed **r** make any difference in the previous part of the question? Explain why or why not.

[20 Marks]

(a) Bob knows the protocol and reads the first 128-bit value as $h(r)$ and recovers the message $\mathbf{m} = \mathbf{RC4[h(r)||k]} \oplus \mathbf{c}$.

(b) The hash function is cryptographically strong (suitable for cryptographic purposes, is one-way and preimage and second preimage resistance) hence the change fo $h(r_1) = h(r_2)$ for 16-bit r_1, r_2 is negligible. Given the seed is only 16-bit and there are only 2^{16} distinct hash values, in the best case scenario (maximum) after exchanging 2^{16} messages r will be repeated (and hence $h(r)$ and $RC4[h(r)||k]$).
More advanced argument: The value of r is randomly chosen and the longer the same key is used the probability that r is repeated will be increased (less choices for r). Applying the generalised birthday problem approximation for r, it will be repeated with probability of .5 after $\sqrt{2^{16}} = 2^8$ messages (the formula $n(p; d) \approx \sqrt{2 \times d \times ln(\frac{1}{1-p})} \approx 300$ where p=.5 and d=65536).

(c) Yes, the argument is that RC4 key material will be repeated if the initial key is the same which in this case would be $h(r)||k$ when the value of r is the same, and to counter that the unique hash values must be more than the number of messages for as long as the secret shared key **k** is used. Increasing the bit size will decrease the probability of the same seed being used which decreases the probability of the RC4 key stream being repeated.

3. Alice is using CFB mode of operation to encrypt a 16KB file to send it to Bob (1KB=1024 bytes).

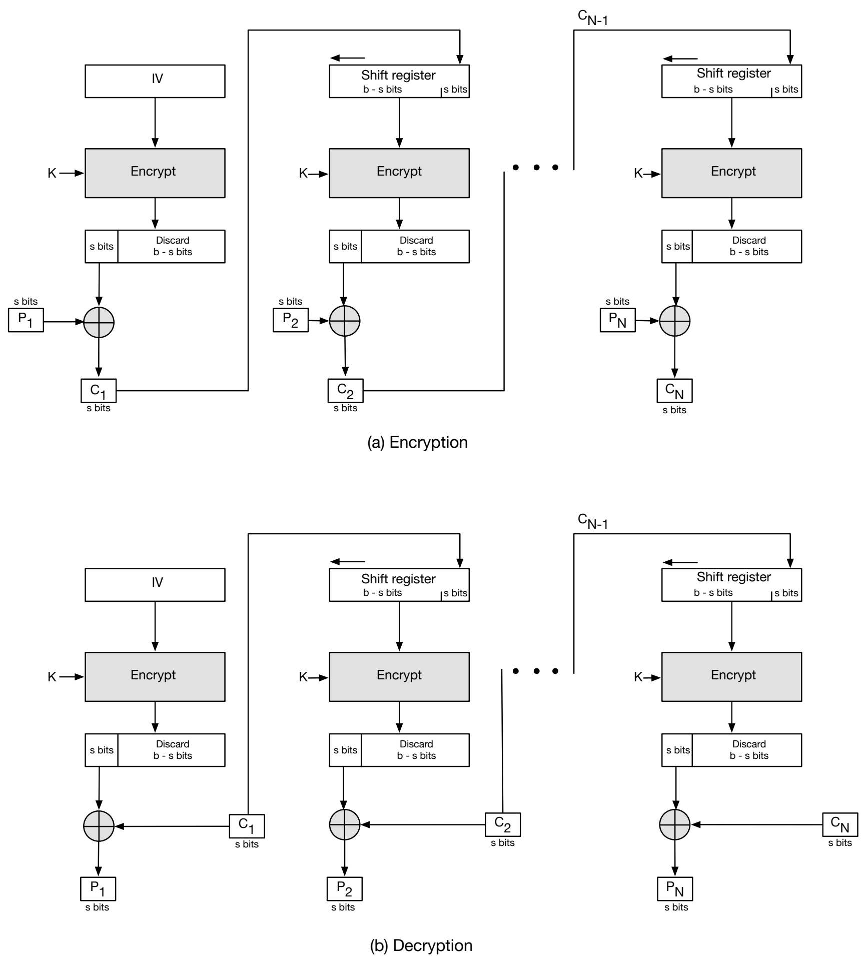

Figure 1: s-bit Cipher Feedback Mode of Operation

(a) How many ciphertext blocks will be produced if Alice uses 8-bit CFB-DES (DES or Data Encryption Standard algorithm as the block cipher in Figure 1)?

(b) How many ciphertext blocks will be produced if Alice uses 16-bit CFB-AES (AES or Advanced Encryption Standard algorithm as the block cipher in Figure 1)?

(c) If there is an error in transmitted ciphertext block C_1, how many plaintext blocks will be corrupted when Bob decrypts the file for 8-bit CFB-DES?

(d) If there is an error in transmitted ciphertext block C_1, how many corrupted plaintext blocks when 16-bit CFB-AES is used?

[20 Marks]

b: the block size of the block cipher algorithm
s: number of bits used in CFB mode of operation

(a) for 8-bit CFB the number of bits per blocks is 8 hence total number of blocks = $\frac{\text{file size in bits}}{\text{8bits per block}} = \frac{16\times1024\times8}{8} = 16384$, the block size of the block cipher has no effect on the number of ciphertext blocks.

(b) for 16-bit CFB as the bit size is doubled the block numbers will be halved so: 8192

(c) using the formula: 1 (by XOR) + $\frac{b}{s}$ (by shift register) $=1+\frac{64}{8}=9$, so P_1 to P_9, since $C_2, C_3, \ldots, C_9$ are all received correctly when C_1 is shifted out the plaintexts will be recovered correctly onward.

(d) using the formula: 1 (by XOR) + $\frac{b}{s}$ (by shift register) $=1+\frac{128}{16}=9$ so P_1 to P_9
(**Note:** even though the number of blocks are the same the extent of error propagation is twice in terms of affected bits compared to 8-bit CFB-DES as the affected plaintext blocks are 16-bit each)

4. Joe is an overworked and under-appreciated programmer at "This Secure World" company. He is asked to write an RSA key generation algorithm that performs better than the competition. To increase the efficiency of his algorithm he decides instead of generating two random prime factors for the modulus part of every RSA key pair, to reuse one of the previous factors and only generate one new random prime number for the new pair. For instance if the algorithm is generating x RSA key pairs the value of n for these keys would be as:
$n_1 = p_1 \times q_1$
$n_2 = p_2 \times q_1$
$n_3 = p_2 \times q_2$
$n_4 = p_3 \times q_2$
...
etc.
This has increased the efficiency of his algorithm by reducing the time required to test the primality of the randomly generated numbers and for the first time in quite a while he is praised by his supervisor for the surprisingly good performing algorithm. The company is going to embed this algorithm in all of their hardware and software products. You are tasked with evaluation of the security of Joe's approach by either approving or rejecting Joe's idea. For either case you need to provide a compelling argument for or against the explained approach. You can use the following four values of n to argue your case (if needed).

n_1=6707746877476258769687006155346667312135818167434246494098858837590302781071
n_2=10154884310969769321427673109695437709100362015315810046444550032439143118393
n_3=16589923032327209104008156227749326050568372812838589709379351079078499097041
n_4=1575487835625540131816354221478432115692476442918739107617773841300854393383

[20 Marks]
The method is not secure as the values of n have common factor with each other and can be factored by the "common factor attack". The attack takes advantage of the efficient Euclidean algorithm to find the GCD of the modulus of the RSA key pairs. As each number only has two large prime factors once one of the factors is discovered the other can be recovered by a simple division. Having the factors, recovering the private key is as simple as calculating $\phi(n) = (p-1) \times (q-1)$ and $e \times d \bmod \phi(n) = 1$. This can be shown simply by using the WolframAlpha web site:
GCD(n_1, n_2)=31867979191070123899911747301895093233
GCD(n_2, n_3)=318654792953276345216131749790350968521
GCD(n_3, n_4)=52062367801132535308762955797447186121
which would result in factoring all four modulus:
n_1 other factor by performing n_1/GCD(n_1, n_2)=210485479397949024065459709030935810687.
Using the common factors as above all modulus can be factored as follows:
n_1=210485479397949024065459709030935810687 $\times$ 31867979191070123899911747301895093233
n_2=31867979191070123899911747301895093233 $\times$ 318654792953276345216131749790350968521
n_3=318654792953276345216131749790350968521 $\times$ 52062367801132535308762955797447186121
n_4=52062367801132535308762955797447186121 $\times$ 30261547873573046755505273091340640623

5. Eve (the adversary to Alice and Bob) intercepts the following communication between Alice and Bob:

   - Alice: Let's use Diffie-Hellman key exchange algorithm to share a secret key
   - Bob: Ok, let the prime be
     $p = 21764036621498105787560256376487687131351962980131606042071496200654587822341$ and the primitive root (generator) $g = 5$
   - Alice: Using your selected parameters my public key is
     $Y_A = 7268937201146768929726789293736151850236699943837892054706367857897711254113$
   - Bob: My public key is
     $Y_B = 276234186282125810839720184172914090708234028484467282900291170147002076321$
   - The rest of the communication is encrypted with K_{AB}

   (a) In the above scenario can Eve recover the shared secret key K_{AB} using the captured messages? Explain what stops Eve from recovering the key or how she can calculate the value of the shared secret.
   (b) Knowing the private key of Alice as $X_A = 278623657769$ what is the value of K_{AB}?

   [20 Marks]

   (a) No she cannot, the DH algorithm relies on the difficulty of discrete logarithm, that is to calculate the shared secret key $K_{AB} = {Y_B}^{X_A} \bmod p = {Y_A}^{X_B} \bmod p = g^{X_A \times X_B} \bmod p$, Eve must know one of the values of X_A or X_B. Even though Eve knows the values of p, g, Y_A, Y_B there is no efficient algorithm known to find the value of X_A in the equation $Y_A = g^{X_A} \bmod p$ (same for X_B), beside a brute force approach (trying different values of X_A to find which would satisfy the equation). More advanced discussion can be found in the following paper: "The Decision Diffie-Hellman problem" by Dan Boneh doi:10.1007/BFb0054851 for interested students.

(b) using the formula $K_{AB} = {Y_B}^{X_A} \bmod p$ we can calculate $K_{AB} =$
`10463734079768228575442231996896482266264025572388267597729737683934651234709`
In the provided text box in the wolframalpha web page at `http://www.wolframalpha.com/` enter:
`27623418628212581083972018417291409070823402848446728290029117014700207632 1 ^278623657769 mod`
`21764036621498105787560256376487687131351962980131606042071496200654587822341`
All without line breaks (with space between mod and the numbers).