UConn, School of Computing
Fall 2024
CSE 3400/CSE 5850: Introduction to Computer and Network Security / Introduction to Cybersecurity

Assignment 5

Instructor: Prof. Ghada Almashaqbeh
Posted: 11/8/2024
Submission deadline: 11/17/2024, 11:59 pm

Notes:

- Solutions **must be typed** (using latex or any other text editor) and must be submitted as a pdf (not word or source latex files).

Problem 1 [30 points]
This problem is about shared key protocols:

1. For the 2PP protocol in slide 16, lecture 8, we modify the protocol as follows: the nonce sent by Bob is computed as $N_B = 3 \cdot N_A$. Eve, is a MitM attacker who is trying to impersonate Alice, i.e., claim to be Alice and complete a successful handshake with Bob. How can Eve do that under this modified protocol? Justify you answer.

   Eve can impersonate Alice to Bob by exploiting the dependence of N_B on N_A. Below is the attack:

   (a) Eve stores Alice's MAC from a previously used nonce.
   (b) Bob sends back the same nonce.
   (c) Eve sends the previous MAC from Alice.
   (d) Bob now believes Eve is Alive.

   Since Bob receives a valid MAC over $N'_A \parallel N_B$, he believes he's communicating with Alice. Eve successfully completes the handshake by relaying messages and without knowing the shared key K.

2. For the 2PP protocol in slide 16, lecture 8, Eve is a MitM who has planned the following attack. Every time Alice sends a nonce N_A (i.e., the first message), Eve modifies that nonce to another value she chooses and then send the modified value to Bob. Why is Eve able to do that? what is the impact of this attack on Alice and Bob? Justify you answer.

   Eve can modify Alice's nonce N_A because it is sent in plaintext without any protection. By changing N_A to N'_A and sending it to Bob, Eve causes Bob to responses to be based on N'_A, not the original N_A. This leads to mismatched nonces when Alice and Bob compute their MACs, resulting in failed authentication. The impact is that Alice and Bob cannot successfully complete the protocol, allowing Eve to disrupt their communication.

3. For the 2RT-2PP protocol in slide 7, lecture 9, Charlie wants to modify the protocol by replacing the MAC with a keyed hash function. So, the third and fourth messages will have $h_k(...)$ computed over the same messages you see in the protocol, instead of $MAC_k(...)$. Charlie claims that security of this protocol will not be impacted if we are using a CRHF function since still integrity will be preserved (that is, any changes to the nonces will change the output of the hash function). Is Charlie's claim correct? why?

   No, Charlie's claim is incorrect. Replacing the MAC with a keyed hash function h_k weakens the protocol's security because a keyed hash does not provide the same integrity guarantees as a MAC. An attacker can exploit properties of the hash function (e.g., length extension attacks if h is based on Merkle-Damgård construction) to forge valid h_k outputs without knowing the key k. This allows the attacker to manipulate messages and compromise the protocol's integrity.

4. For the counter-based RR protocol in slide 8, lecture 9, Alice resets her counter back to 0 after each successful run of this protocol with Bob. What is the impact of this change? Justify your answer.

   If Alice resets her counter to 0 after each successful run with Bob, it enables replay attacks. An attacker can record messages from previous sessions and replay them when the counter resets, causing Bob to accept outdated messages as fresh. This compromises the protocol's security by allowing the attacker to impersonate Alice or disrupt communication.

5. For the 2RT-2PP protocol in slide 10, lecture 9, Roger wants to modify the protocol to have the MAC computed over the *req* and *resp* without encrypting them, i.e., the third message will be $E_{k.e}(req), MAC(3 \parallel A \rightarrow B \parallel N_A \parallel N_B \parallel req)$ and the fourth message will be $E_{k.e}(resp), MAC(4 \parallel A \leftarrow B \parallel N_A \parallel N_B \parallel resp)$. Will that impact the security of the protocol (as compared to the original protocol)? why?

   No, this modification does not impact the security of the protocol. Computing the MAC over the plaintext req and resp messages and sending it alongside the encrypted messages still ensures both confidentiality and integrity. The encryption $E_{k.e}$(req) and $E_{k.e}$(resp) protect the messages from being read by unauthorized parties, while the MAC computed over NA, NB, and the plaintext ensures that any tampering with the messages will be detected. Since the nonces are included, replay attacks are also difficult. Thus, the protocol's security remains effectively the same as the original.

Note: if the scheme is insecure then provide a successful attack against it. If the scheme is secure, just provide a convincing argument (formal security proofs are not required).

Problem 3 [40 points]
Compute the following expressions (please read the note below carefully):

1. $192 \cdot 17^{1700} + 986 \cdot 2024^{56} \mod 37$

$$\Big[\big((192 \cdot 17^{1700}) \mod 37\big) + \big(((986 \mod 37) \cdot (2024^{56} \mod 37)) \mod 37\big)\Big] \mod 37$$
$$= \Big[\big(((192 \mod 37) \cdot (17^{1700} \mod 37)) \mod 37\big) + \big(((986 \mod 37) \cdot ((2024 \mod 37)^{56} \mod 37)$$
$$= \Big[\big(((192 \mod 37) \cdot ((17 \mod 37)^{1700 \mod 36}) \mod 37)\big) + \big(((986 \mod 37) \cdot ((2024 \mod 37)^{56}$$
$$= \Big[\big((7 \cdot (17^{8} \mod 37)) \mod 37\big) + \big((24 \cdot (26^{20} \mod 37)) \mod 37\big)\Big] \mod 37$$
$$= \Big[\big((7 \cdot 33) \mod 37\big) + \big((24 \cdot 10) \mod 37\big)\Big] \mod 37$$
$$= 27$$

2. $576(23023^{1024000} + 365 \cdot 856985^{7202}) \mod 73$

$$\Big[\big((576 \cdot 23023^{1024000}) \mod 73\big) + \big(((365 \mod 73) \cdot (856985^{7202} \mod 73)) \mod 73\big)\Big] \mod 73$$
$$= \Big[\big(((576 \mod 73) \cdot (23023^{1024000} \mod 73)) \mod 73\big) + \big(((365 \mod 73) \cdot ((856985 \mod 73)^{7202}$$
$$= \Big[\big(((576 \mod 73) \cdot ((23023 \mod 73)^{1024000 \mod 72} \mod 73)) \mod 73\big) + \big(((365 \mod 73) \cdot ((856$$
$$= \Big[\big((65 \cdot (28^{16} \mod 73)) \mod 73\big) + \big((37 \cdot 1) \mod 73\big)\Big] \mod 73$$
$$= \Big[\big((65 \cdot 37) \mod 73\big) + (0)\Big] \mod 73$$
$$= (2405 \mod 73)$$
$$= 69$$

3. $\frac{1436}{8} + \frac{568}{17} + 10785 \mod 101$

$$\frac{1436}{8} + \frac{568}{17} + 10785 \mod 101$$
$$= (179 + 33 + 10785) \mod 101$$
$$= (179 \mod 101) + (33 \mod 101) + (10785 \mod 101) \mod 101 \quad \text{(apply}$$
$$= (78) + (33) + (56) \mod 101 \quad \text{(reduce e}$$
$$= (167) \mod 101$$
$$= 66$$

4. $2048 - \frac{8}{1436} + \frac{10785}{20 \cdot 17} \mod 99$

$$
\begin{aligned}
&2048 - \frac{8}{1436} + \frac{10785}{20 \cdot 17} \mod 99 \\
&\qquad = 2048 - 0 + \frac{10785}{340} \mod 99 && \text{(reduce fractions)} \\
&\qquad = 2048 + 31 \mod 99 && \text{(reduce } \frac{10785}{340} = 31\text{)} \\
&\qquad = (2048 \mod 99) + (31 \mod 99) \mod 99 && \text{(apply modular addition)} \\
&\qquad = 67 + 31 \mod 99 && \text{(reduce each term mod 99)} \\
&\qquad = 98 \mod 99 \\
&\qquad = 98
\end{aligned}
$$

5. $10245 \cdot (24 + (345 \cdot 10^{77}(34^{381} + 18976))) \mod 67$

$$
\begin{aligned}
80 &= 2 \cdot 2 \cdot 2 \cdot 2 \cdot 5 = 2^4 \cdot 5 \\
2728 &= 2 \cdot 2 \cdot 2 \cdot 341 = 2^3 \cdot 341 = 2^3 \cdot 31 \cdot 11 \\
80 \cdot \left(1 - \frac{1}{2}\right) \cdot \left(1 - \frac{1}{5}\right) &= 32 \\
2728 \cdot \left(1 - \frac{1}{2}\right) \cdot \left(1 - \frac{1}{11}\right) \cdot \left(1 - \frac{1}{31}\right) &= 1200 \\
1200 + 32 &= 1232
\end{aligned}
$$

6. $\phi(80) + \phi(2728)$ (First represent each of 80 and 2728 as a product of powers of distinct primes and then compute the Euler's Totient function.)

$$
\begin{aligned}
&\Big[(10245 \cdot (24 + (345 \cdot 10^{77}(34^{381} + 18976))) \mod 67)\Big] \mod 67 \\
&= \Big[((10245 \mod 67) \cdot ((24 \mod 67) + ((345 \cdot 10^{77}(34^{381} + 18976)) \mod 67))) \mod 67\Big] \\
&= \Big[((10245 \mod 67) \cdot (((24 \mod 67) + ((345 \mod 67) \cdot ((10^{77}(34^{381} + 18976)) \mod 67))) \mod 67)) \\
&= \Big[((10245 \mod 67) \cdot (((24 \mod 67) + ((345 \mod 67) \cdot ((((10^{77}) \mod 67) \cdot ((34^{381} + 18976) \mod 67)) \\
&= \Big[((10245 \mod 67) \cdot (((24 \mod 67) + ((345 \mod 67) \cdot ((((10^{77} \mod 67) \cdot (((34 \mod 67)^{381 \mod 66}) \\
&= \Big[(61 \cdot (((24) + ((440) \mod 67)) \mod 67)) \mod 67\Big] \\
&= \Big[(61 \cdot ((62) \mod 67)) \mod 67\Big] \\
&= \Big[(61 \cdot 62) \mod 67\Big] \\
&= (3782 \mod 67) \\
&= 30
\end{aligned}
$$

Note: The goal of this problem is to use the simplification techniques we studied in class. Computing the whole expression (as one shot) using a calculator or some software without

showing the simplification steps will be considered a wrong answer. Also, whenever applicable indicate the theorem/lemma/property that allowed the simplification step that you applied. You can use the calculator for intermediate steps that cannot be simplified (for example, multiply numbers or compute a mod operation that cannot be simplified)

Problem 3 [30 points]
This problem is about public key cryptographic primitives.

1. Alice and Bob want to run the DH key exchange protocol for multiplicative cyclic groups. Alice selected the prime $p = 47$ (which satisfies $47 = 2 \cdot 23 + 1$), and the generator $g = 5$. Show an execution of the protocol where Alice chooses the secret key $a = 9$, and Bob chooses the secret key $b = 7$. Show both Alice's computations and Bob's computations in the protocol as well as their outputs.

   Alice and Bob execute the Diffie-Hellman key exchange using prime $p = 47$ and generator $g = 5$.

   Alice selects her secret key $a = 9$ and computes:

   $$A = g^a \mod p = 5^9 \mod 47 = 40,$$

   which she sends to Bob.

   Bob selects his secret key $b = 7$ and computes:

   $$B = g^b \mod p = 5^7 \mod 47 = 11,$$

   sending this to Alice.

   Upon receiving B, Alice computes the shared secret:

   $$s = B^a \mod p = 11^9 \mod 47 = 38.$$

   Similarly, Bob computes:

   $$s = A^b \mod p = 40^7 \mod 47 = 38.$$

   Both Alice and Bob arrive at the shared secret key $s = 38$.

2. Extend the Diffie-Hellman key exchange protocol to allow four parties (say Alice, Bob, Cameron, and Doug) to exchange a key. So at the end of this protocol, all four parties will establish the same key, say g^{abcd} where c is the secret key that Cameron will choose and d is the secret key that Doug will choose (there is no limit on the number of exchanged messages, so a party may need to send two ore more rounds of messages instead of one as in the 2-party DH key exchange. However, try to minimize the number of sent messages as possible). Consider eavesdropper security as well here.

   In the first round, Alice sends her public value $A = g^a$ to Bob, Bob sends $B = g^b$ to Cameron, Cameron sends $C = g^c$ to Doug, and Doug sends $D = g^d$ to Alice.

In the second round, each party raises the received value to their secret exponent and passes it on:

Alice computes D^a and sends it to Bob,
Bob computes A^b and sends it to Cameron,
Cameron computes B^c and sends it to Doug,
Doug computes C^d and sends it to Alice.

In the third round, the process repeats:

Alice computes $(C^d)^a = g^{acd}$ and sends it to Bob,
Bob computes $(D^a)^b = g^{abd}$ and sends it to Cameron,
Cameron computes $(A^b)^c = g^{abc}$ and sends it to Doug,
Doug computes $(B^c)^d = g^{bcd}$ and sends it to Alice.

In the final round, each party raises the received value to their secret exponent to obtain the shared key:

$$s = g^{abcd}.$$

This protocol ensures that all four parties arrive at the same key $s = g^{abcd}$ while maintaining eavesdropper security, as an eavesdropper cannot compute s without knowing the private exponents.

3. Consider the El-Gamal PKE scheme, and a given a ciphertext (x, y) encrypting some unknown message m (that is, $(x, y) := (g^b, e^b \cdot m)$ where the secret key is a, the public key is $e = g^a$, b is some random integer selected for encryption as we saw in class). Show that given the public key $e = g^a$ and the ciphertext (x, y), it is possible in polynomial time to generate a ciphertext (x', y') encrypting the message km for any desired integer $k \in \mathbb{Z}_p^*$, and such that $x' \neq x$ and $y' \neq y$.

Given the ElGamal ciphertext $(x, y) = (g^b, e^b \cdot m)$ and the public key $e = g^a$, we can generate a new ciphertext (x', y') encrypting the message $k \cdot m$ without knowing m.

Select a random integer b' in $\mathbb{Z}_p^*$ and compute:

$$x' = x \cdot g^{b'} = g^{b+b'}$$

and

$$y' = y \cdot e^{b'} \cdot k = e^{b+b'} \cdot k \cdot m.$$

This new ciphertext (x', y') is not equal to (x, y) since $b' \neq 0$, and it correctly encrypts $k \cdot m$ under the ElGamal scheme, as it follows the encryption process with exponent $b + b'$.