Cody Clark 30010560
CPSC418 Exam 2020

1. Alice and Bob both construct a handshaking protocol. Bob first generates his random number, N, and then computes h(N). He sends h(N) to Alice. Alice is now certain that Bob has generated an N. Alice then makes a guess of the parity of N and sends it to Bob. Bob knows now that Alice has received his hash and has made her guess without knowing its parity. Then Bob finally sends his number N to Alice. Alice computes the hash of this number. If the hash of the random number N Bob sent matches the h(N) Bob sent earlier, then she knows that Bob is being honest and they both know if she won or not.

   Alice cannot cheat in this protocol because she doesn't know the number generated until after she's already sent her guess to Bob and Bob replies with his number. Bob cannot cheat in this situation because he must generate and send a number to Alice before she makes her guess, and the number he sent is then verified.

2. **a)** Because $C \equiv M^e \pmod p$, then $M \equiv C^d \equiv (M^e)^d \pmod p$.

   $(M^e)^d \equiv M^{ed} \equiv M^{k(p-1)+1} \pmod p$ - Because $ed = k(p-1) + 1$
   $M^{k(p-1)+1} \equiv MM^{k(p-1)} \equiv M(M^{(p-1)})^k \equiv M(1)^k \equiv M \pmod p$

   Because p is prime $(p-1) = \phi(p)$, then applying Euler's Theorem where $M^{\phi(p)} \equiv 1 \pmod p$. Thus the decryption procedure is correct. QED.

   b) This scheme uses the intractability of the IFP (Integer Factorization Problem). The security of the system rests on the privacy of the keys (e, d). Because $ed = k(p-1) + 1$, an attacker can recover a key if they discover either e or d, as all they would need to do afterwards would be to compute the modular inverse under (mod p-1). (They would need to find a factor of $(k(p-1)+1)$, which are e and d)

   In a known plaintext attack Eve would have access to the message M and its ciphertext C. If Eve can solve instances of the IFP, then she can find either e or d. Once Eve has found a factor, let's call it f, then she would just need to compute M^f and C^f. Because f = e or d, then either the ciphertext will be decrypted such that $C^f = M$, or the plaintext will be encrypted such that $M^f = C$. She would at that point know whether f = d or f = e.

   In order to find the other part of the key, she would then solve the linear congruence of $fx \equiv 1 \pmod{p-1}$ for x using the extended euclidean algorithm which she can do because $gdc(e, p-1) = gcd(d, p-1) = 1$. After which she will have her key $(f, x) = (e, d)$.

3. **a)** A cryptosystem provides perfect secrecy only if $p(C) = p(C|M)$ for all plaintexts M with $p(M) > 0$ and ciphertexts $p(C) > 0$. By Bayes' theorem $p(C|M) = (p(C)p(M|C))/p(M)$ for all $M \in M$, $C \in C$ with $p(M) > 0$, $p(C) > 0$.

   To prove that this system does not have perfect secrecy we must prove that $p(C) \neq p(C|M)$. We will do this by contradiction. Suppose a one-time pad where $K = 0^n$ has been removed from the key space K has perfect secrecy. We will construct a counterexample. Let n = 1. Then $K = \{1\}$ (Because K = 0 has been removed from the keyspace), $M = \{0, 1\}$ and $C = \{0, 1\}$. Consider the following cases:

   Case 1 (M = 0)**:** Then $E_K(M) = C = 1$ - Because K = 1, $0 \oplus 1 = 1$

   Case 2 (M = 1)**:** Then $E_K(M) = C = 0$ - Because K = 1, $1 \oplus 1 = 0$

   Then p(M) = 0.5, p(C) = 0.5, and p(M|C) = 1 (Given any ciphertext we know the plaintext with certainty).

   By Bayes' theorem $p(C|M) = (p(C)p(M|C))/p(M) = ((0.5)(1))/(0.5) = (0.5)/(0.5) = 1$.
   This shows that $1 = p(C|M) \neq p(C) = 0.5$, proving that this system cannot have perfect secrecy. QED

   b) Because M, C, and $K \in \mathbb{N}$, it implies that the message space is the same as the cipher space. Then there's a possibility that when encrypting with a key K, a plaintext M can be transformed into any number of possible, valid ciphertexts that look like other, reasonable plaintexts. Suppose someone wants to send a message of M = 1010. Transformed with a key of K = 0000, the ciphertext will be C = 1010. It was also possible that the message to send might be M = 0101. If the key was K = 1111 (an equally likely possibility), then the ciphertext would also be C = 1010. If one does not know the key and the key is generated randomly, then the fact C = 1010 gives no information about which plaintext was actually encrypted due to perfect secrecy.

4. **a) i)** Let x be a hash value such that H(M) = x for a message M.

   Consider the two possible cases:

   Case 1:
   A preimage i of x is found such that H(i) = x, and the last bit of i is 0. Because the last bit is 0, H(i) = h(i), thus h(i) = x and i is a preimage under h as well.

   Case 2:
   A preimage i of x is found such that H(i) = x, and the last bit of i is 1. Because H(M) = h(M'), then H(i) = h(i'), where i' = i with the last bit of i being replaced by 0. In this case i' is the preimage under h which was easy to compute from i.

   Thus, if a preimage can feasibly be found for H, then a preimage can be feasibly found for h. h is known to be preimage resistant, therefore H must also be preimage resistant. QED

   ii) To prove this we must find two distinct inputs (bit strings) M and M' such that H(M) = H(M').

   Let M = 11 and M' = 10. Then H(M) = h(M') → H(11) = h(10). (Because M' = M where the last bit is replaced by 0) When the last bit is already 0 as in M', H(M) = h(M), thus H(M') = h(M') → H(10) = h(10). Therefore H(10) = h(10) = H(11) and a strong collision has been found. QED

   This example exploits the fact that for any message M, H(M) can only be the result of bit strings with the last bit being 0 being hashed under h. If a bit string doesn't end in 0, then it's essentially converted before being hashed. Therefore any bit strings that differ in the last bit are essentially treated as the same number under H.

   b) i) We will prove that H is weakly collision resistant through contradiction.

   Suppose that having been given an input M, it is computationally feasible to find another input M' such that H(M') = H(M).

   Consider the case when M ≠ 0. Then this implies that H(M') = H(M) → 0||h(M') = 0||h(M). This then suggests that the only part of the hash value which may differ is h(M') = h(M). If it is computationally feasible to find two values M' and M such that h(M') = h(M), then the function h would be vulnerable to weak collisions. Because h is weakly collision resistant there exists a contradiction, therefore it is also infeasible to compute a weak collision for the function H when M ≠ 0.

Consider the case when M = 0, then H(M) = $1||0^n$. In order to find a weak collision we would need to find an M' ≠ M, such that H(M') = $1||0^n$. However, if M' ≠ 0, then H(M') = $0||h(M') \neq 1||0^n$. Therefore H is weakly collision resistant. QED

ii) x = $1||0^n$ is an example where the preimage under H is M = 0. This is a good example because we have this example shown within the definition of the function H itself. If H(M) = $1||0^n$ then it's not possible for M to be anything else except for 0. QED

5. **a)** $BMAC_K(M_3) = BMAC_K(M_2)$.

$BMAC_K(M_1) = E_K(0^n \oplus M_1)$
$M_2 = BMAC_K(M_1) = E_K(0^n \oplus M_1)$
$BMAC_K(M_2) = E_K(0^n \oplus BMAC_K(M_1)) = E_K(0^n \oplus E_K(0^n \oplus M_1))$
$M_3 = M_1 || 0^n$
Therefore,
Round 1 of $BMAC_K(M_3)$: $C \leftarrow E_K(0^n \oplus M_1) = M_2$
Round 2 (final round) of $BMAC_K(M_3)$:
$C \leftarrow E_K(M_2 \oplus 0^n) = E_K(E_K(0^n \oplus M_1) \oplus 0^n) = E_K(0^n \oplus E_K(0^n \oplus M_1)) = BMAC_K(M_2)$

For a MAC to be computation resistant it must satisfy the following requirement: For any fixed but unknown key K, it should be computationally infeasible to compute a a new message/MAC pair (M, $MAC_K(M)$), even if provided with many pairs (M_i , $MAC_K(M_i)$). We have been provided with a few pairs and have been able to compute a new message/MAC pair (M_3, $MAC_K(M_3)$) without knowledge of the key. Therefore, BMAC is *not* computation resistant.

b) I will use the same calculations as made in 5a) in order to save some space.

$M_4 = M_2 || Y = E_K(0^n \oplus M_1) || (BMAC_K(M_1) \oplus BMAC_K(M_2))$
$= E_K(0^n \oplus M_1) || (E_K(0^n \oplus M_1) \oplus E_K(0^n \oplus E_K(0^n \oplus M_1)))$

Round 1 of $BMAC_K(M_4)$:
$C \leftarrow E_K(0^n \oplus E_K(0^n \oplus M_1)) = BMAC_K(M_2)$

Round 2 (final round) of $BMAC_K(M_4)$:
$C \leftarrow E_K(E_K(0^n \oplus E_K(0^n \oplus M_1)) \oplus Y)$
$= E_K(E_K(0^n \oplus E_K(0^n \oplus M_1)) \oplus (E_K(0^n \oplus M_1) \oplus E_K(0^n \oplus E_K(0^n \oplus M_1))))$
$= E_K(BMAC_K(M_2) \oplus (BMAC_K(M_1) \oplus BMAC_K(M_2)))$
$= E_K(BMAC_K(M_2) \oplus BMAC_K(M_1) \oplus BMAC_K(M_2))$
$= E_K(BMAC_K(M_2) \oplus BMAC_K(M_2) \oplus BMAC_K(M_1))$
$= E_K(0^n \oplus BMAC_K(M_1))$ - Because $BMAC_K(M_2) \oplus BMAC_K(M_2) = 0^n$
$= BMAC_K(M_2)$

Therefore $BMAC_K(M_4) = BMAC_K(M_2)$

Using the same definition of computation resistance in (a), we have seen that we can generate a new message/MAC pair (M_4, $BMAC_K(M_4)$) without any knowledge of the key using a select few previously obtained message/MAC pairs. Therefore, once more BMAC has been shown to not possess computation resistance.

6. **a)** Because $x_i \equiv g^{ai} \pmod p$, then $y_i \equiv x_i^b \equiv (g^{ai})^b \pmod p$. Therefore the participants need to extract g^b from g^{aib}. The most efficient way to do this is by solving the linear congruence $a_i x \equiv 1 \pmod{\phi(p)}$ for x such that $x = a_i^{-1}$. (Using the extended Euclidean algorithm) Then $g^{aibx} \equiv g^b \equiv K \pmod p$, the shared key.

   Suppose $a_i x \equiv 1 \pmod{\phi(p)}$. Then $a_i x = r(p-1) + 1$ for some integer r. Thus, we have:

$$
\begin{aligned}
g^{aibx} &\equiv g^{aixb} \\
&\equiv g^{b(r(p-1)+1)} \\
&\equiv g^{br(p-1)+b} \\
&\equiv g^{br(p-1)} g^b \\
&\equiv (1)^{br} g^b && \text{- By Euler's theorem} \\
&\equiv g^b \pmod p
\end{aligned}
$$

   Because the EEA is feasible to compute, we have proved that each conference participant can efficiently compute the secret key K. QED

   (Note: Throughout these problems when ai appears in the exponents it should be interpreted as a_i)

   b) If an eavesdropper, Eve, were to be able to extract discrete logarithms (can solve the discrete logarithm problem) then she could find the shared conference key.

   Suppose Eve has the ability that when given p, g, and $g^x \pmod p$, then she can feasibly find x (Solve the DLP). In this scenario p and g are known. Eve could listen to a message from a participant A_i when they initiate the key agreement protocol so that Eve knows the value A_i sent to B, $x_i \equiv g^{ai} \pmod p$. She would then intercept B's response to A_i, $y_i \equiv x_i^b \pmod p$. Knowing y_i, x_i, and $x_i^b \pmod p$, she would be able to extract the exponent b (by solving the DLP), which she then uses to calculate $K \equiv g^b \pmod p$, the shared key.

7. **a)** Suppose an eavesdropper, Eve, knows that the same message M was encrypted and sent to two different people who have used the same RSA modulus, n. We will see how Eve can recover M from C without needing to factor n.

   Because Eve knows e_1, e_2, and $\gcd(e_1, e_2) = 1$, she can then solve the linear Diophantine equation $e_1x + e_2y = 1$ for x and y using the extended Euclidean algorithm. With x and y computed, Eve can then compute $C^d \equiv M^{ed} \equiv M \equiv M^{e1x+e2y} \equiv M^{e1x}M^{e2y} \equiv C_1^{x}C_2^{y} \pmod n$. Either x or y will be a negative number, therefore Eve must also solve the modular inverse $C_iC_i^{-1} \equiv 1 \pmod n$, where i = 1 or 2, depending on whichever number is negative. If x is negative, then $(C_1^{-1})^{x}C_2^{y} \equiv M \pmod n$. If y is negative then $C_1^{x}(C_2^{-1})^{y} \equiv M \pmod n$. This retrieves the encrypted message M without needing to factor n.

   b) In this situation we will solve $e_2x + e_1y = 1$, simply because $e_2 > e_1$. It could easily be solved where $e_1x + e_2y = 1$ but that requires another step for both algorithms.

   The attack: 49(x) + 13(y) = 1. Solving for x and y by the Euclidean and extended Euclidean algorithms:

   Euclidean algorithm:

$a = 49 = 13(3) + 10 = 39 + 10$	$q_0 = 3, r_0 = 10$
$b = 13 = 10(1) + 3 = 10 + 3$	$q_1 = 1, r_1 = 3$
$r_0 = 10 = 3(3) + 1 = 9 + 1$	$q_2 = 3, r_2 = 1$
$r_1 = 3 = 1(3) + 0 = 3 + 0$	$q_3 = 3, r_3 = 0$

   Extended Euclidean Algorithm (Linear recurrence):

   $A_0 = (3)(1) + (0) = 3$
   $B_0 = (3)(0) + (1) = 1$
   $A_1 = (1)(3) + (1) = 4$
   $B_1 = (1)(1) + (0) = 1$
   $A_2 = (3)(4) + (3) = 15$
   $B_2 = (3)(1) + (1) = 4$
   $A_3 = (3)(15) + (4) = 49 = e_2$
   $B_3 = (3)(4) + (1) = 13 = e_1$

   $x = (-1)^{3-1}(4) = (-1)^2(4) = 1(4) = 4$
   $y = (-1)^3(15) = (-1)(15) = -15$

   $49(4) = 196$
   $13(-15) = -195$

   $49(4) + 13(-15) = 196 - 195 = 1$ confirming x = 4 and y = -15.

Then $M \equiv M^{e2x+e1y} \equiv M^{e2x}M^{e1y} \equiv C_2^{x}C_1^{y} \equiv (74)^4(46)^{-15} \equiv (4)(46)^{-15} \equiv (4)(46^{-1})^{15} \pmod{n}$
Now we must solve for 46^{-1}. Which means solving $46x \equiv 1 \pmod{77}$ for x, where $46x - 77y = 1$ using the extended Euclidean algorithm.

Euclidean algorithm:

$a = 46 = 77(0) + 46$	$q_0 = 0$
$b = 77 = 46(1) + 31$	$q_1 = 1$
$r_0 = 46 = 31(1) + 15$	$q_2 = 1$
$r_1 = 31 = 15(2) + 1$	$q_3 = 2$
$r_2 = 15 = 1(15) + 0$	$q_4 = 15$

Extended Euclidean algorithm (Linear recurrence):
$A_0 = (0)(1) + (0) = 0$
$B_0 = (0)(0) + (1) = 1$
$A_1 = (1)(0) + (1) = 1$
$B_1 = (1)(1) + (0) = 1$
$A_2 = (1)(1) + (0) = 1$
$B_2 = (1)(1) + (1) = 2$
$A_3 = (2)(1) + (1) = 3$
$B_3 = (2)(2) + (1) = 5$
$A_4 = (15)(3) + (1) = 46$
$B_4 = (15)(5) + (2) = 77$

$x = (-1)^{4-1}(5) = (-1)(5) = -5$
$y = (-1)^4(3) = (1)(3) = 3$

$46(-5) + 77(3) = 1$, so $x = 77 - 5 = 72$.

We can see $(46)(72) \equiv 1 \pmod{77}$ Therefore $72 = 46^{-1}$. Then we have:

$$
\begin{aligned}
(4)(46^{-1})^{15} &\equiv (4)(72)^{15} \\
&\equiv (4)(43) \\
&\equiv 172 \\
&\equiv 18 \pmod{77}
\end{aligned}
$$

Therefore, M = 18.

We can then confirm this by encrypting 18 with the e_i for each user i.
User 1: $18^{e1} \equiv 18^{13} \equiv 46 \equiv C_1 \pmod{77}$
User 2: $18^{e2} \equiv 18^{49} \equiv 74 \equiv C_2 \pmod{77}$

This proves that we have correctly found the encrypted message M without needing to factor n.

8. **a) i)** Eve knows the following public values: M_1, M_2, s_1, s_2, r, p, g, y, and the hash function H. She knows that s_1 and s_2 were computed using the following equation: $ks_i \equiv H(M_i||r) - xr \pmod{p-1}$, where i is the message number 1 or 2.

   Then $ks_1 + xr \equiv H(M_1||r) \pmod{p-1}$, and $ks_2 + xr \equiv H(M_2||r) \pmod{p-1}$. Therefore she can subtract the second equation from the first as this:

   $ks_1 + xr - (ks_2 + xr) \equiv H(M_1||r) - H(M_2||r) \pmod{p-1}$
   $ks_1 + xr - ks_2 - xr \equiv H(M_1||r) - H(M_2||r)$
   $ks_1 - ks_2 \equiv H(M_1||r) - H(M_2||r)$
   $k(s_1 - s_2) \equiv H(M_1||r) - H(M_2||r) \pmod{p-1}$

   Eve can calculate $s_1 - s_2$ and $H(M_1||r) - H(M_2||r)$ because all of the values and functions involved are public. Let $s_1 - s_2 = a$, and $H(M_1||r) - H(M_2||r) = b$, then Eve just needs to solve the linear congruence $ak \equiv b \pmod{p-1}$ for k using the EEA. This works because as outlined in the question $\gcd(s_1-s_2, p-1) = 1$.

   ii) If Eve knows k (possibly through the attack in the previous question) then she can consider the equation $ks + xr \equiv H(M||r) \pmod{p-1}$. Because she knows k, s, r, m, p, and the function H, she can rearrange the equation as follows: $xr \equiv H(M||r) - ks \pmod{p-1}$. Then, let r = a, and $H(M||r) - ks = b$, then she once more needs to use the EEA to solve the linear congruence $ax \equiv b \pmod{p-1}$ for x. This is possible because as mentioned in the question, $\gcd(r, p-1) = 1$. After solving for x then Eve would know Alice's private key.

   b) i) To prove this we will show that $y^r r^s \equiv g^M \pmod{p}$ with the substitutions Mallory has calculated using (u, v). (I will use the ^ symbol to denote exponentiation when necessary and omit the modulus until the end in order to provide a clearer picture.)

$y^r r^s$	$\equiv (g^x)^r r^s$	- From $y \equiv g^x \pmod{p}$
	$\equiv g^{xr}(g^u y^v)^s$	- From $r \equiv g^u y^v \pmod{p}$
	$\equiv g^{xr}(g^{su} y^{sv})$	
	$\equiv g^{xr}(g^{su} g^{xsv})$	- From $y \equiv g^x \pmod{p}$
	$\equiv g^{xr+su+xsv}$	
	$\equiv g\text{^}(xr + (-rv^*)u + x(-rv^*)v)$	- From $s \equiv -rv^* \pmod{p-1}$
	$\equiv g\text{^}(xr - rv^*u - xrvv^*)$	
	$\equiv g\text{^}(xr - rv^*u - xr(1))$	- From $vv^* \equiv 1 \pmod{p-1}$
	$\equiv g\text{^}(xr - rv^*u - xr)$	
	$\equiv g\text{^}(-rv^*u) \pmod{p}$	
g^M	$\equiv g^{su}$	- From $M \equiv su \pmod{p-1}$
	$\equiv g\text{^}((-rv^*)u)$	- From $s \equiv -rv^* \pmod{p-1}$
	$\equiv g\text{^}(-rv^*u) \pmod{p}$	(Thus $y^r r^s \equiv g^M \pmod{p}$)

This proves that anyone will accept the signature (r, s) to the “message” M as a valid signature coming from Alice, so Mallory has committed existential forgery against Alice. QED

ii) By using H(M||r) Alice is providing a mechanism by which to ensure data integrity. In Mallory’s attack she has constructed a new “message” M’ where M’ ≡ -rv*u (mod p-1), using her own values (u, v) which allow her to masquerade as Alice when creating a signature, replacing Alice’s M with her M’. Because a recipient of the signature is able to calculate H(M||r) themselves, they’re able to tell if the message has been changed. Without a hash function they’re unable to ensure the same integrity. (As we’ve seen with (b) (i)) By using a hash function H(M||r), the only way to replicate the same effect is by finding some message M’ such that H(M’) ≡ H(M||r) (mod p), a preimage attack. If the hash function H is preimage resistant, then it’s computationally infeasible to generate such an M’, negating the possibility of a preimage attack and rendering Mallory’s previous attack infeasible.

9. **a)** When n is prime then $\mathbb{Z}_n^* = \{1, 2, \ldots, n-2, n-1\}$, so that $1 \le a \le n-1$. When n is prime, then the Jacobi symbol and the Legendre symbol are the same: (a/n). Then by Euler's Criterion: $a \in QR_n$ iff $a^{(n-1)/2} \equiv 1 \pmod n$. The Legendre symbol is defined as (a/n) = 1 if $a \in QR_n$, -1 if $a \in NR_n$, and 0 if n|a. Consider the following cases:

Case 1: $a^{(n-1)/2} \equiv 1 \pmod n$
By Euler's criterion, then $a \in QR_n$, therefore the Jacobi and Legendre symbol (a/n) = 1. This matches the computation and returns that n is prime, which is true.

Case 2: $a^{(n-1)/2} \equiv -1 \pmod n$
By Euler's criterion, then $a \in NR_n$, therefore the Jacobi and Legendre symbol (a/n) = -1. This matches the computation and returns that n is prime, which is true.

We can use Fermat's theorem to prove that these are the only two possible cases when n is prime. $a^{n-1} \equiv 1 \pmod n$, therefore n divides $a^{n-1}-1 = (a^{(n-1)/2}-1)(a^{(n-1)/2}+1)$ by the binomial theorem. Then either n divides $(a^{(n-1)/2}-1)$ or n divides $(a^{(n-1)/2}+1)$. This implies $a^{(n-1)/2} \equiv \pm 1 \pmod n$, proving that these are the only two cases and that this testing algorithm correctly answers when the input n is prime. QED

b) $247 = 13 \cdot 19$.

c) (179/247) = (179/13)(179/19) - By property (3)
= (10/13)(8/19) - By property (1) $179 \equiv 10 \pmod{13}$
- $179 \equiv 8 \pmod{19}$
= (5/13)(2/13)(4/19)(2/19) - By property (2)
= (5/13)(2/13)(2/19)(2/19)(2/19)
= (5/13)((-1)^((13^2 - 1)/8)(((-1)^((19^2 - 1)/8))^3 - By property (4)
$(13^2 - 1)/8 = 21$
$(19^2 - 1)/8 = 45$
= $(5/13)(-1)^{21}((-1)^{45})^3$
= $(5/13)(-1)(-1)^3$
= (5/13)(-1)(-1)
= (5/13)
= $(13/5)(-1)^{((5-1)/2)((13-1)/2)}$ - By property (5) 5 is odd
= $(13/5)(-1)^{(2)(6)}$
= $(13/5)(-1)^{12}$
= (13/5)
= (3/5) - By property (1) $13 \equiv 3 \pmod 5$
= $(5/3)(-1)^{((3-1)/2)((5-1)/2)}$ - By property (5) 3 is odd
= $(5/3)(-1)^{(1)(2)}$
= $(5/3)(-1)^2$
= (5/3)
= (2/3) - By property (1) $5 \equiv 2 \pmod 5$

$= (-1)^{((3^2-1)/8)}$ - By property (4)
$= (-1)^1$
$= -1$

Thus, the Jacobi symbol of (179/247) = -1.

d) a = 179, n = 247. $179^{(247-1)/2} = 179^{123} \equiv 246 \equiv -1 \pmod{247}$, and the Jacobi symbol *e* = (179/247) = -1, as we've seen in part (c). Thus, the algorithm will output that 247 is prime, however we know that 247 can be factored by $247 = 13 \cdot 19$. Therefore this is a case when the algorithm lies and gives the wrong answer. QED