Q1. SHA 계열의 해시함수나 CBC-MAC 등은 대칭암호알고리즘과 마찬가지로 채우기가 필요하다. 채우기가 필요한 이유와 보안적으로 그것의 중요한 이유(모두 0으로 채우기를 하거나 비트 채우기만 하였을 때 문제점 을 제시)를 설명하시오. 또 대칭 암호알고리즘의 채우기와 중요한 차이점을 설명하시오. 힌트. SHA 계열은 Merkle과 Damg˙ard 구조 기반이기 때문에 내부적으로 블록 단위의 함수를 활용한다.

A1.
SHA 계열의 해시함수나 CBC-MAC 등에서 채우기가 필요한 이유는 입력 데이터의 크기가 블록 크기의 배수가 아닐 경우에 발생한다. 이 경우 마지막 블록은 부족한 부분을 채우기 위해 추가적인 비트나 바이트를 더해야 한다.

채우기를 하지 않으면 암호화나 인증 작업을 할 때 마지막 블록의 크기가 부족하면 처리할 수 없기 때문에 처리 오류가 발생할 수 있다.

보안적으로 중요한 이유는 채우기를 하지 않으면 입력 데이터를 블록 단위로 나누어 암호화하거나 인증하는 과정에서 데이터 유출이나 변조 등의 공격이 가능해진다. 예를 들어, 마지막 블록을 모두 0으로 채우면 모든 블록이 같아져서 암호화나 인증에 취약해진다.

SHA 계열의 해시 함수에서는 Merkle-Damgård 구조를 사용하기 때문에 내부적으로 블록 단위의 함수를 반복해서 적용한다. 따라서 입력 데이터가 블록 크기의 배수가 아닐 경우 마지막 블록을 채워주어야 하는데, 이 때 사용하는 채우기 방식은 대칭암호알고리즘에서 사용하는 채우기 방식과 동일하다.

하지만 대칭암호알고리즘의 채우기와 SHA 계열의 해시함수에서의 채우기의 중요한 차이점은 채우기 방식을 정의하는 규칙이 다르다는 점이다. 대칭암호알고리즘에서는 PKCS#7 패딩이나 ISO/IEC 7816-4 패딩 등의 규격이 존재하지만, 해시함수에서는 주로 1을 추가하는 비트 채우기 방식이 사용된다.

Q2.

RSA 암호알고리즘에서 공개키가 (n, e)이고 개인키가 (n, d)일 때, $M < n$인 메시지의 암호화는 $C = M^e \mod n$을 이용한다. 이 암호문은 $C^d \mod n$을 통해 복호화한다. 복호화 결과의 정확성은 $\gcd(M, n) = 1$일 때, 다음을 통해 증명할 수 있다.

$$C^d = M^{ed} = M^{k\phi(n)+1} \equiv \left(M^{\phi(n)}\right)^k M \equiv M \pmod{n}$$

참고. 오일러 정리. $\gcd(m, n) = 1$이면 $m^{\phi(n)} \equiv 1 \pmod{n}$이 성립한다.
$\gcd(M, n) \neq 1$이면 M은 p의 배수이거나 q의 배수이다. 따라서 M은 p와 서로소이거나 q와 서로소이다.
$M = ap$이고 q와 서로소라고 가정하면 다음이 성립한다.

$$M\left(M^{\phi(n)}\right)^k \equiv M\left(M^{\phi(q)}\right)^{\phi(p)k}$$

M은 q와 서로소이기 때문에 $M^{\phi(q)} \equiv 1 \pmod{q}$이다. 따라서 다음이 성립한다.

$$M\left(M^{\phi(q)}\right)^{\phi(p)k} = ap(1 + bq)$$

$ap(1 + bq) \equiv M \pmod{n}$임을 보이시오. 이것이 성립하면 모든 $M < n$에 대해 RSA의 정확성이 증명된다.

A2. RSA 암호 알고리즘에서 공개키가 (n, e)이고 개인키가 (n, d)일 때, M < n인 메시지의 암호화는 C = Me mod n을 사용합니다. 이 암호문은 C d mod n을 사용하여 복호화됩니다.

M과 n이 서로소이면, 오일러 정리에 따라 Mϕ(n) ≡ 1 (mod n)이 성립합니다. 여기서 ϕ(n)은 n보다 작은 양의 정수 중에서 n과 서로소인 수의 개수를 나타내며, n = p * q와 같이 두 소수의 곱으로 표현될 때 ϕ(n) = (p-1) * (q-1)입니다.

따라서 C d ≡ M ϕ(n)d ≡ M kϕ(n)+1 ≡ M (mod n)이 성립합니다. 이것은 복호화된 결과가 원래의 메시지 M과 일치한다는 것을 의미합니다.

반면에 gcd(M, n) ≠ 1인 경우, M은 p의 배수이거나 q의 배수입니다. 따라서 M은 p와 서로소이거나 q와 서로소입니다.

M = ap이고 q와 서로소라고 가정하면 다음이 성립합니다.

M ϕ(n)k ≡ M ϕ(q)ϕ(p)k

M은 q와 서로소이기 때문에 M ϕ(q) ≡ 1 (mod q)이 성립합니다. 따라서 다음이 성립합니다.

M ϕ(n)k ≡ apϕ(q)k ≡ aϕ(n)kqk ≡ 1qk ≡ 1 (mod q)

즉, M ϕ(n)k는 q로 나누어 떨어집니다. 따라서 다음이 성립합니다.

C d ≡ M ϕ(n)d ≡ (ap)ϕ(n)d ≡ apϕ(n)d ≡ apkϕ(n)+1 ≡ apkϕ(q)ϕ(p)+1 ≡ (apϕ(q)k)ϕ(p)ap ≡ 1ap ≡ aq ≡ 0 (mod q)

이것은 복호화된 결과가 원래의 메시지 M과 일치하지 않을 수 있다는 것을 의미합니다.

Q3. RSA 공개키가 (n, e)인 사용자에게 1부터 365 사이의 수 m을 암호화하여 전달하고자 한다. 이를 위해 매우 큰 수 x를 선택한 후에 x와 (m + x) e mod n를 전달하였다. 이 방식의 문제점을 제시하시오.

A3. RSA 암호화에서는 평문 m이 모듈러 n보다 작은 경우, 이를 곧바로 암호화할 수 있다는 장점이 있다. 하지만 문제에서 제시된 방식으로 암호화를 수행하면 평문 m을 암호화하는 대신, (m + x)^e를 암호화하게 된다. 이 때 x는 매우 큰 임의의 수로, 보통 2의 거듭제곱값으로 구성된다.

이 방식은 패딩 없이 암호화를 수행하고 있기 때문에, 평문 m이 작은 경우에도 암호문의 크기가 n보다 크게 될 수 있다는 문제점이 있다. 이는 암호문이 매우 예측하기 쉬워져서 보안성이 떨어지는 것을 의미한다.

또한, 암호화되는 m의 값에 따라서는 (m + x)^e가 n보다 큰 경우가 발생할 수 있다. 이 경우에는 모듈러 연산을 수행해야 하는데, 이는 암호화와 복호화 모두에서 추가적인 시간과 연산이 필요하게 된다. 따라서 이 방식은 비효율적이며, 일반적으로 사용되지 않는다. 대신에 RSA 패딩 방식을 사용하여 암호화를 수행하는 것이 좋다.

Q4. m(< n)에 대한 RSA 전자서명 값을 md mod n을 통해 계산한다고 가정하자. 서명자의 서명키가 (n, d)이고, 확인키가 (n, e)일 때, m = rem′ mod n을 계산하여 서명자로부터 m에 대한 서명값 md mod n을 확보하였다. 이 서명으로부터 서명자의 또다른 유효한 서명을 만들 수 있다. 해당 서명을 얻는 방법을 제시하시오.

A4. RSA 전자서명에서는 서명자는 메시지 m에 대한 해시 값을 계산한 후, 해당 값을 서명하는 것이 일반적입니다. 그러나 이 문제에서는 직접적으로 m을 서명하는 방식을 사용합니다.

서명자가 m에 대한 서명값 m^(d) mod n을 이미 확보했으므로, 다음과 같이 새로운 서명값을 생성할 수 있습니다.
1. 임의의 정수 k를 선택합니다. (단, gcd(k,n)=1)
2. r = m^(k*e) mod n을 계산합니다.
3. s = k^(-1) * (m^(d) * r^(d-1) - 1) mod n을 계산합니다.

위의 s는 서명자가 새로 만든 m에 대한 서명값입니다. 이 값을 서명 검증자가 확인했을 때, m^(e) = r^(s)*m' (mod n)이 되는 것을 알 수 있습니다. 이는 RSA 전자서명 검증 과정에서 검증되므로, 서명자가 새로운 유효한 서명값을 생성할 수 있다는 것을 보여줍니다.