**COMP90043: Cryptography and security**

**Week 7: RSA Signatures and Hash Functions**

1. **Assume that Alice chooses two primes 43 and 47 to construct her RSA key prime factors. Help her to set up public and private keys and demonstrate encryption and decryption with an example. Choose the smallest possible exponent for the public key.**

p=43; q:=47; n:=p*q= 2021; $\phi(n) = (p-1)*(q-1) = 1932$;

Let us choose a smallest possible for illustration.

e=1 is not a good choice (then encryption will be trivial).

e=2; is impossible because e is even and hence GCD(e, $\phi(n)$) is always even. Note that $\phi(n)$ is always even.

e=3; In this case GCD(e, 1932) = 3 so we cannot choose this number for e.

e=5; GCD(e, $\phi(n)$) = 1 a valid choice. Let e=5, d:= Modinv(e, $\phi(n)$) = 773;

m:=313; $c:=m^e \bmod n = 464$;

Note that $m:=c^d \bmod n = 313$.

2. **a.** $n = 403; \phi(n) = 360; d = 19; C = 388.$

For decryption, we have $388^{19} \bmod 403$

$= 388^{16} \times 388^{12} \times 388^{1} \bmod 403 = 2$

**b.** $n = 341; \phi(n) = 300; d = 43; C = 16.$

For decryption, we have $16^{43} \bmod 341$

$= 16^{32} \times 16^{8} \times 16^{2} \times 16^{1} \bmod 341 = 4$

**c.** $n = 51; \phi(n) = 32; d = 13; C = 14.$

For decryption, we have $14^{13} \bmod 51$

$= 14^{8} \times 14^{4} \times 14^{1} \bmod 51 = 5$

**d.** $n = 85; \phi(n) = 64; d = 55; C = 31.$

For decryption, we have $31^{55} \bmod 85$

$= 31^{32} \times 31^{16} \times 31^{4} \times 31^{2} \times 31^{1} \bmod 85 = 6$

**e.** $n = 119; \phi(n) = 96; d = 53; C = 12.$

For decryption, we have $12^{53} \bmod 119$

$= 12^{32} \times 12^{16} \times 12^{4} \times 12^{1} \bmod 119 = 3$

3. **CCA Attack**

**Consider n = 91, e =7 and d=31.**

**M= 5; C = M^e mod n = 47**

**Decrypt C without using d:**

**X = (C * 2^e) mod n**

**Compute 2^7 mod 91 = 128 mod 91 = 128 – 91 = 37**

**X = 47 x 37 mod 91 = 10**

**In CCA attack, Adversary is able to get decryption of X:**

**Compute Y = X^d mod n; = 10 ^ (31) mod 91 = 10 (can you do it without calculator)?**

**10^31 = 10 (10^3)^10 = 10 (-1)^10 = 10; because (100 mod 91 = 9) and 1000 mod 91= 90 = -1**

**But note that X^d == (2M) mod N = Y = 10**

**M = (Inverse(2)* Y;**

**Inverse (2) mod n = 46;**

**(Observe 2* 45 = 90 = -1 mod n; then 2 ( - 45) = 1 mod n; hence -45 = 91-45 = 46 is the inverse of 2 mod n**

**M = 46* Y = 460 = 5 mod n as required.**

**4.** Yes. If a plaintext block has a common factor with n modulo n then the encoded block will also have a common factor with n modulo n. Because we encode blocks, which are smaller than pq, the factor must be p or q and the plaintext block must be a multiple of p or q. We can test each block for primality. If prime, it is p or q. In this case we divide into n to find the other factor. If not prime, we factor it and try the factors as divisors of n.

**5. What are the advantages using Hash functions in digital signatures?**

Some uses are:

- You can sign arbitrary long messages.
- You can use to build authentication mechanisms (see text/slides).
- Assuring integrity of messages.

**6. Explain how you can use RSA encryption function to construct a digital signature scheme.**

Public Key Parameters: (n, e)

Private Key parameter: (n, d)

Hash Function: (H)

Compute Signature (S) = $(H(M)^d)$ mod n; [M, s] form message signature pair.

Verification Algorithm: If H(M) == $((s^d)$ mod n) then accept the signature else declare verification failure.

**7. What characteristics are needed in a secure hash function?**

1. H can be applied to a block of data of any size.
2. H produces a fixed-length output.
3. H(x) is relatively easy to compute for any given x, making both hardware and software implementations practical.

4. For any given value h, it is computationally infeasible to find x such that H(x) = h. This is sometimes referred to in the literature as the one-way property.

5. For any given block x, it is computationally infeasible to find $y \neq x$ with H(y) = H(x).

6. It is computationally infeasible to find any pair (x, y) such that H(x) = H(y).

**8. What is the difference between weak and strong collision resistance?**

Weak collision resistance: Property 5 of answer in previous question;

Strong collision resistance: Property 6 of answer in previous question.

**9. Is it possible to use a hash function to construct a DES like block cipher? How is it possible?**

If you examine the structure of a single round of DES, you see that the round includes a one-way function, f, and an XOR: $R_i = L_i – 1\ f(R_i – 1, K_i)$

For DES, the function f maps a 32-bit R and a 48-bit K into a 32-bit output (see the slides/text for details). That is, it maps an 80-bit input into a 32-bit output. This is clearly a one-way function. Any hash function that produces a 32-bit output could be used for f. The demonstration in the text that decryption works is still valid for any one-way function f.

**10. Explain the birthday paradox in simple words? What is the main implication of this for hash function?**

The problem for adversary in collision resistant attack is to produce to messages x and y which give rise to same hash value, i.e. H(x) = H(y).

Birthday paradox is not a paradox! If we choose random variables from a uniform distribution in the range 0 to N-1, then the probability that a repeated element is encountered exceeds 0.5 after about √N (square root N) choices have been made.

You can see Appendix 1A of Chapter 11 of the text for precise derivation of this result.

Hence for m bit hash, if we pick messages at random, we can expect to find two messages with the same hash value with about $\sqrt{2^m} = 2^{m/2}$ attempts. Hence two break the collision resistant property of an m bit hash function; the effort for adversary is upper bounded by a quantity approximately $2^{m/2}$.

This implies that m should be chosen reasonably high.

**11. Name three important hash functions used in practice.**

MD4, MD5, SHA-1/2/3.

**12. Discuss how the security of the hash functions depends on the length of the hash.**

(Discuss Brute force attack and birthday attacks and show how they influence the decision on length of the hash.) 32 bit hash is trivial to break. The length of the hash should be at least 160 bits or more.

**13. Why CRC checksum cannot be used as a secure hash function?**

Collisions are easy to find and does not satisfy one way property requirement of hash functions.

**14. Consider the following questions in regards to Timing Attacks:**

**a. What is a Timing Attack?**

This is a very dangerous attack side channel attack, as it factors in the time complexity involved in deciphering a message by the intended receiver in order to be able to recover the private key.

**b. How can Timing Attacks be prevented?**

The following are examples of approaches which can be used to prevent timing attacks:

- **Constant Exponentiation Time:** Requires the modifications of algorithms so as to ensure that all exponentiation operations take a fixed amount of time to execute before producing a result. This can be detrimental on performance however.

- **Random Delay:** Furthers on the exponentiation time problem by introducing random delay periods during each exponentiation operation to throw off an attacker from knowing how long the operation actually took. Offers better performance than constant exponentiation time operations.

- **Blinding:** Involves the multiplication of the plaintext by a random integer prior to performing any exponentiation operations on it.

**15. With RSA, discuss how the concept of Blinding can be implemented?**

RSA allows the incorporation of Blinding in two ways:

- By allowing the multiplication of the plaintext message with an arbitrary number before performing exponentiations which allows the operations to be performed not on the real input or the real output. Not all functions can use this approach. One example is when Alice wants Bob to return to her the result of a function F which only Bob has access to. But Alice doesn't want Bob to know the input M. By using blinding, Alice blinds, encrypts, and sends M' to Bob. Bob then calculates and returns the value of F(M'). Alice then decrypts F(M') and reverses the blinding to get F(M).

- RSA supports Blind Signatures, wherein similar to the above concept; the content of the passed message is oblivious to the person signing it. This is applicable when the authenticity of the message needs validation from an entity who did not write the message.