CAB340 - Cryptography
Queensland University of Technology
Semester 2, 2018

Assignment 3

Student: John Santias (N9983244)
Due: Friday 26th October 2018

Contents

Number Theory.....3

RSA Encryption.....5

Digital Signatures.....6

Diffie-Hellman Key Agreement.....8

Secret Sharing.....9

References.....12

Number Theory

a. Bezout's identity states that for positive integers a and b, there always exist integers m and n such that

$$am + bn = gcd(a, b)$$

Also, recall that $a \equiv b \ (mod\ n)$ means that there exists an integer ℓ such that

$$a - b = \ell n$$

This is called modular equivalence or modular congruence.
Recall that (if it exists) the multiplicative inverse of p modulo q, is defined to be the unique number p^{-1} such that $p^{-1}p \equiv 1 \ (mod\ q)$.

i. Apply the definition of modular equivalence and write down what $p^{-1}p \equiv 1 \ (mod\ q)$ means.

If we apply modular equivalence to $p^{-1}p \equiv 1(mod\ q)$, both sides would still be equivalent and will have the same remainder when divided by q. For example, if p is equal to 5 and q is an integer number:

$$5^{-1} \times 5 \equiv 1 \ (mod\ 5)$$
$$1 \equiv 1 \ (mod\ 5)$$
$$1 \equiv 1$$

ii. Rearrange what you get and apply Bezout's identity to conclude that if $gcd(p, q) = 1$ then $p^{-1}\ exists$ (modulo q).

p^{-1} can be rearranged to:

$$p^{-1} = \frac{1}{p} \neq 0$$

If $gcd(p, q) = 1$ then p^{-1} will exist. This is evident with $p = 3, q = 4$:

$$\gcd(3,4) = 1$$
$$\frac{1}{3} \times 3 \equiv 1\ mod\ 4$$
$$1 \equiv 1$$

$$\therefore \gcd(3,4) \equiv 1\ mod\ 4 \equiv \frac{1}{3} \times 3$$

b. Division algorithm tells us that, given positive numbers a and b, with $a \geq b$, there always exist integers q and r, with $0 \leq r < b$ such that

$$a = bq + r.$$

(think of primary school division before you learned about decimals) q is called the $quotient$ and r is called the remainder. It can be shown that

$$\gcd(a, b) = \gcd(b, r)$$

The Euclidean algorithm makes use of this fact to calculate $gcd(a, b)$ by repeatedly replacing the larger number by a remainder (which is always smaller). The algorithm is:

i. $Set\ a_0 = a, b_0 = b, j = 0.$

ii. Use the division algorithm to find q_j and r_j with $0 \leq r_j < b_j$ such that aj = bj qj + rj

iii. If rj > 0 then set aj+1 = bj , bj+1 = rj , increment j and go back to the previous step iv. Output rj−1 as gcd(a, b). We can keep track of all values in a table. For example, we can calculate gcd(27, 16) by: j aj bj qj rj 0 27 16 1 11 (27 = 16(1) + 11) 1 16 11 1 5 (16 = 11(1) + 5) 2 11 5 2 1 (11 = 5(2) + 1) 3 5 1 5 0 (5 = 1(5) + 0) and we find gcd(27, 16) = 1 from the second last value of rj . The final column is included for illustration and is not necessary.

j	a_j	b_j	q_j	r_j	Illustration
0	31	19	1	12	$(31 = 19(1) + 12)$
1	19	12	1	7	$(19 = 12(1) + 7)$
2	12	7	1	5	$(12 = 7(1) + 5)$
3	7	5	1	2	$(7 = 5(1) + 2)$
4	5	2	2	1	$(5 = 2(2) + 1)$
5	2	1	2	0	$(2 = 1(2) + 0)$

c.

j	q_j	m_j	n_j	Illustration
4	2	1	-2	$(5(1) + 2(-2) = 1)$
3	1	-2	3	$(7(-2) + 5(3) = 1)$
2	1	3	-5	$(12(3) + 7(-5) = 1)$
1	1	-5	8	$(19(-5) + 12(8) = 1)$
0	1	8	-13	$(31(8) + 19(-13) = 1)$

d. Bezout's identity is $am + bn = \gcd(a, b)$

j	a_j	b_j	m_j	n_j	$am + bn$	$\gcd(a, b)$	$am + bn = \gcd(a, b)$
0	31	19	8	-13	$(31(8) + 19(-13) = 1)$	1	Yes
1	19	12	-5	8	$(19(-5) + 12(8) = 1)$	1	Yes
2	12	7	3	-5	$(12(3) + 7(-5) = 1)$	1	Yes
3	7	5	-2	3	$(7(-2) + 5(3) = 1)$	1	Yes
4	5	2	1	-2	$(5(1) + 2(-2) = 1)$	1	Yes
5	2	1	0	1	$(2(0) + 1(1) = 1)$	1	Yes

RSA encryption

For the following questions, we are using the RSA cryptosystem. For each of the questions below, write down the operation you need to perform in addition to the answer that you get. Wolfram Alpha is suitable for all the calculations you will need to do.

a. Given the primes $p = 2027$ and $q = 2593$, and using $e = 7$

i. Generate the corresponding public RSA key.

$$n = pq$$
$$n = 2027 \times 2593$$
$$n = 5256011$$

The public key is the pair n and e,

$$K_E = (n, e)$$
$$K_E = (5256011, 7)$$

ii. Generate the corresponding private RSA key.

The private key consists of the values p, q and d.

$$\varphi(n) = (p-1)(q-1) = 5251392$$
$$d = e^{-1} \, mod \, \varphi(n)$$
$$d = 7^{-1} \, mod \, 5251392$$
$$d = 750199$$

The private key is $K_D = d = 750199$

b. Using the public key above, encrypt the message 1024.

$$C = E(M, K_E)$$
$$C = M^e mod \, n$$
$$C = 1024^7 mod \, 5256011$$
$$C = 3406593$$

c. Using the private key above, decrypt the message 3054908.

$$D = (C, K_D)$$
$$M = C^d mod \, n$$
$$M = 3054908^{750199} mod \, 5256011$$
$$M = 100001$$

d. You are given the public key (n = 7354943, e = 7). Determine the private key by first factoring the public key. Wolfram Alpha is able to perform the necessary factorisation.

Public key: $K_E = (7354943, 7)$
Using wolframalpha: "Factorise 7354943"
Result = 2711×2713

$$\varphi(n) = (p-1)(q-1) = 7349520$$
$$d = e^{-1} \, mod \, \varphi(n)$$
$$d = 7^{-1} \, mod \, 7349520$$

$$d = 2099863$$

The private key is $K_D = d = 2099863$

Digital Signatures

a. Digital signatures are sometimes used for authenticating users in network protocols. SSH and TLS, for example, can both use such a mechanism. Consider the following protocol:
   - The server sends Alice a randomly chosen number.
   - Alice digitally signs the number and sends the signature back to the server.
   - The server checks the signature using Alice's public key. If the signature is valid, then the server accepts Alice's connection request.

Suppose that Alice uses the same private key to log in to a server and sign her emails. Show how a server could forge emails from Alice. For this exercise, assume that the authentication mechanism signs the bare challenge without hashing, while for emails Alice signs the hash of the message.

The server does not have any authentication mechanism or hashing function which makes it vulnerable to forgery attacks. There could be people on the server or connected to it. If Alice uses the private key to sign in to the server anyone could be listening for inputs and record it. Once the key is captured, anyone can act as Alice and send emails to anyone. The receiver will not know if the email was from the original sender because it will have the same hash values for authentication.

If there is no hashing on the server it will just encrypt messages without hashing. The receiver may not notice about the missing hash but can easily view the message. They won't be able to authenticate that the message was from the original sender. The aim of digital signatures is to achieve authentication, integrity, and non-repudiation but in this case it doesn't.

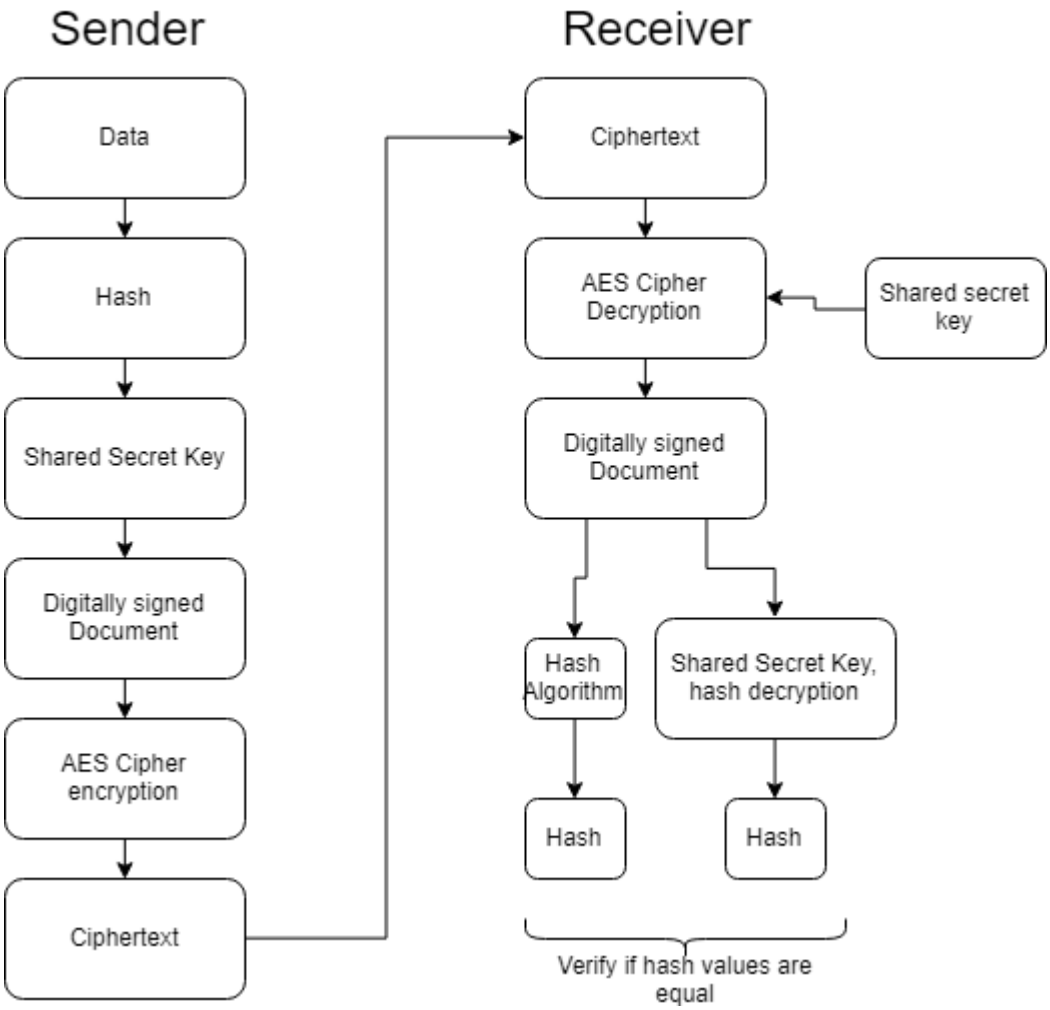
b. Suggest a modification to the authentication protocol which would defeat this attack.

To improve the security and prevent forgery attacks, we could change the hash function to the most secure type available and make sure that it is collision resistant. Without it, we can't prevent forgery attacks. An untrusted party can choose a message m and m' so that $h(m) = h(m')$ (TheBestVPN.com. 2018.).

A couple of other methods like adding XOR gates and timers or strengthening the RSA keys to 4096 bits could help but may not completely prevent the attack. However, by implementing AES cipher through the digital signature process, we can achieve authentication, non-repudiation and defeat forgery attacks.

When implementing AES, we remove the private and public keys. The hosts will have to negotiate a shared secret key for encryption and decryption, and it will need to be highly confidential. This more secure digital signature process is run by hashing the data, then using the secret key to get a digitally signed document. Afterward, it is encrypted with the AES cipher to produce the ciphertext and sent to the receiver.

The receiver then decrypts the ciphertext with AES to get the document, then produces two different hashes for authentication. The first is using the hash algorithm to produce the hash value, and another hash produced by decrypting with the public key. We would have to implement AES that is 128-bits or above to achieve integrity and authentication (TheBestVPN.com. 2018.). Forgery attack may be launched but may just make very minimal changes to the message like a bit flip.



Diffie-Hellman Key Agreement

For this question we will be considering the original Diffie-Hellman key agreement algorithm, which is like the authenticated version, but excludes the signatures and verifications.

a. For Diffie-Hellman we need to choose a public generator $g\ modulo\ p$ (where p is prime). If g is a generator, then for every x with $\gcd(p, x) = 1$, there must exist a k such that

$$g^k \equiv x\ (mod\ p)$$

This means that we can take discrete logarithms with base g for any x which is not a multiple of p. We can test if g is a generator by finding its order, which is the smallest k such that $g^k \equiv 1\ (mod\ p)$. g is a generator modulo p if and only if the order of g is $p - 1$. Using $p = 2027$, find all generators between 1 and 10. You can use Wolfram Alpha to find the order of g modulo p by typing, for example, *order of 2 modulo 2027,* substituting in your value of g for 2.

x	p	Generator (g)	Illustration (using wolframalpha)
1	2027	1	Order of 1 mod 2027 = 1
2	2027	2026	Order of 2 mod 2027 = 2026
3	2027	1013	Order of 3 mod 2027 = 1013
4	2027	1013	Order of 4 mod 2027 = 1013
5	2027	2026	Order of 5 mod 2027 = 2026
6	2027	2026	Order of 6 mod 2027 = 2026
7	2027	2026	Order of 7 mod 2027 = 2026
8	2027	2026	Order of 8 mod 2027 = 2026
9	2027	1013	Order of 9 mod 2027 = 1013
10	2027	1013	Order of 10 mod 2027 = 1013

b. Using $p = 2027$ and g the smallest generator that you found in the previous question, suppose that Alice and Bob perform Diffie-Hellman key agreement, where Alice's secret is 424 and Bob's secret is 1746. What messages do Alice and Bob send to each other?

Using $p = 2027$ and the smallest generator found from the previous question, $g = 1013$, we can perform a Diffie-Hellman key agreement. The smallest generator is not 1 because if we generate the keys for Alice and Bob, we'd have keys with a value of 1 which anyone can use.

Firstly, Alice and Bob pick a random number as their private (or secret) key. Alice chose 424 and Bob had chosen 1746. Then Alice computes $A = g^a (mod\ p)$,

$$A = 1013^{424} (mod\ 2027)$$
$$A = 680$$

and sends the value to Bob. Bob also does the same computation but using his secret key, $B = g^b (mod\ p)$ and sends it to Alice (Diffie-Hellman Protocol -- from Wolfram MathWorld. 2018).

$$B = 1013^{1746} (mod\ 2027)$$
$$B = 1521$$

Alice and Bob send their values that were computed using the generator value, their chosen secret key, and a prime number.

c. What is the key that Alice and Bob agree on?

Once Alice and Bob have exchanged values, they can then compute their shared key $K = g^{ab} (mod\ p)$.

Alice calculates the shared key using the value Bob sent:

$$K = B^a (mod\ p) = (g^b)^a (mod\ p)$$
$$K = 1185$$

Bob also calculates the shared key with the value Alice sent:

$$K = A^b (mod\ p) = (g^a)^b (mod\ p)$$
$$K = 1185$$

As calculated in the question above, the key Alice and Bob agree on is 1185.

Secret Sharing

Consider the following secret sharing scheme for m parties for sharing a secret n-bit string x.

- The dealer generates $m - 1$ random n-bit strings $r_1 \dots r_{m-1}$
- The dealer sends r_j to party j for $j = 1 \dots m - 1$
- The dealer sends $r_1 \oplus \dots . \oplus r_{m-1} \oplus x$ to party m.

a. Explain how all of the parties together can reconstruct the secret x.

The two parties can reconstruct the secret x by XORing their shares together (Secret Sharing - Wikipedia. 2018.). For example,

Our secret is an 8-bit string,

$$x = 11110000$$
$$x = j \oplus m$$

Party j obtains a random 8-bit string r,

$$r = 10101010 = j$$

Party m uses binary XOR operations,

$$m = r \oplus x\ = 01011010$$

To reconstruct secret x, we XOR the shares,

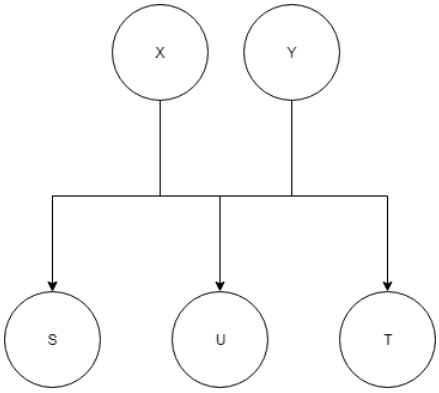
$$\text{Secret } x = j \oplus m = 01011010 \oplus 10101010$$
$$\text{Secret } x\ = 11110000$$

b. Explain why any $m-1$ parties do not have enough information to reconstruct the message. It may be helpful to note that for $m=2$ this scheme corresponds to the one-time-pad. You may also, for the purposes of your explanation, consider a single bit secret (i.e. $n-1$).

A party by itself does not have enough information to reconstruct the message because they each receive a random string. It will be harder for a party that receives the last share because it would have been calculated with XORing the random string with the secret message. With the example provided in the previous question, all parties will need to XOR each other's shares for secret reconstruction. If there is only one party then the secret is trivial and can be to the party (Secret Sharing - Wikipedia. 2018.).

For $m=2$ this scheme corresponds to the one-time-pad because it cannot be cracked. It is a perfect cipher. An attacker can't obtain the secret if they don't have a share. If we consider a single bit secret it can easily be guessed by anyone by a bit flip.

c. Suppose that 3 parties share two secrets x and y using the above scheme. Their shares are s_j for x and t_j for y. Show that if each party forms $u_j = s_j \oplus t_j$ and then they get together to reconstruct the secret from the u_j's, the result is $x \oplus y$.



For example,
Secret $x = 1010$, secret $y = 0101$, $x \oplus y = 1111$

secret	s	t	$u\ (s \oplus t \oplus secret)$
$x = 1010$	1001	1011	1001⊕1011⊕ 1010=1000
$y = 0101$	1000	0011	1000⊕0011⊕ 0101=1110

Referring to the table above, each party has been given a random 4-bit number. For party u, it is the last party that will XOR the secret with $s \oplus t$. Following the scheme from the previous question, the last party always XORs with the secret. To reconstruct the secret x, we use a party's computation of $s \oplus t$,

$$secret\ x = s \oplus t \oplus u$$
$$secret\ x = 0010 \oplus 1000$$

$$secret\ x = 1010$$

Similarly, with $secret\ y$,

$$secret\ y = s \oplus t \oplus u$$
$$secret\ y = 1011 \oplus 1110$$
$$secret\ y = 0101$$

Now we have been able to retrieve the secrets, the secrets can be combined.

$$secret\ x \oplus y = 1010 \oplus 0101$$
$$secret\ x \oplus y = 1111$$

References

Diffie-Hellman Protocol -- from Wolfram MathWorld. 2018 - Diffie-Hellman Protocol -- from Wolfram MathWorld. 2018. Diffie-Hellman Protocol -- from Wolfram MathWorld. [ONLINE] Available at: http://mathworld.wolfram.com/Diffie-HellmanProtocol.html. [Accessed 20 October 2018].

TheBestVPN.com. 2018. - TheBestVPN.com. 2018. What is Advanced Encryption Standard (AES): Beginner's Guide. [ONLINE] Available at: https://thebestvpn.com/advanced-encryption-standard-aes/. [Accessed 20 October 2018].

Secret Sharing - Wikipedia. 2018. - Secret sharing - Wikipedia. 2018. Secret sharing - Wikipedia. [ONLINE] Available at: https://en.wikipedia.org/wiki/Secret_sharing#1_%3C_t_%3C_n,_and,_more_general,_any_desired_subset_of_n. [Accessed 25 October 2018].