CS3302 Practical 2

200007413

Diffie-Hellman Key Exchange

1

1.1 Describe

Carol can sabotage the conversation between Alice and Bob given ability to intercept the initial symmetric key-setup communications. The methodology would be as follows:

1. Carol notes the agreed values of p and g
2. Carol chooses a secret value c, such that $c < p$, and calculates

$$C = g^c \mod p$$

3. Carol intercepts the sharing of A and B calculated by Alice and Bob respectively - stores their values, and sends the value of C to both Alice and Bob pretending to be the other party.

$$A = g^a \mod p$$
$$B = g^b \mod p$$

4. (a) Alice and Bob will then calculate keys based upon their own secret values a or b, and the calculated value C they believe comes from the other party.

$$\text{Alice's Key} = C^a \mod p$$
$$\text{Bob's Key} = C^b \mod p$$

   (b) Carol calculates different keys for both Alice and Bob, using the intercepted values A and B, and Carol's secret value c.

$$\text{Carol's Key 1 (For Alice)} = A^c \mod p$$
$$\text{Carol's Key 2 (For Bob)} = B^c \mod p$$

This will give Carol and Alice the same symmetric key 1, and Carol and Bob the same symmetric key 2.

5. Carol can then intecept messages from Alice to Bob, decrypt the message using key 1, and then re-encrypt with key 2, and send that to Bob pretending to be Alice.

   Carol can also intercept Bob's messages to Alice, decrypting messages using key 2, then re-encrpyting with key 1, then sending that to Alice pretending to be Bob.

   Carol also then has the ability to modify these messages after decrypting them.

1.2 Identify

To mitigate this issue, Alice and Bob can setup their symmetric key using a public-key cryptography system such as RSA for the initial communication.

RSA

2

- n should be known to Bob and Alice, and doesn't have to be kept secret from everyone.

  It should be known as it acts as half of both Alice's public and private keys, and so is required by others to encrypt data to Alice, and required by Alice to decrypt those messages.

- p needs to be known to Alice, but should be kept secret from Bob and everyone else.

  This is because it is used by Alice to generate the value of $\phi(n)$, which then is used to generate the public and private key pair.

  Given p by Bob or others, it would greatly reduce the difficulty of finding the other half of the private key j, due to p's use in generating the auxiliary notation. $\phi(n) = (p-1)(q-1)$. Given p, only the value q is required to be guessed. This can then be used to try values of j which match the formula $ij = 1 + k\phi(n)$.

- q should also be known only to Alice, and kept secret from everyone else.

  This is because it is also used in a similar manner to p in generating the public and private keys.

  Given q, Bob or anyone would have more information to reduce the difficulty of finding the value of $\phi(n)$ in the same was as with knowing q as $\phi(n) = (p-1)(q-1)$.

- i should be known to Bob, and doesn't have to be kept secret.

  It acts as one half of Alice's public key which Bob and others require to encrypt information for Alice to then decrypt.

  It is also used to help generate Alice's private key as $ij = 1 \pmod{\phi(n)}$ helps determine what values of j are valid.

- j should be known only to Alice, and should be kept secret from Bob and others.

With j, an attacker has access to the trapdoor function, and so decryping any information encrypted with Alice's public key is easy. The other half of the private key n is meant to be public, and so with j an attack is trivial.

- $\phi(n)$ should be known only to Alice, and should be kept secret.

  It is used to generate Alice's public and private keys. This is because $ij = 1 \pmod{\phi(n)}$. Therefore, given $\phi(n)$, and the public-key value i, it vastly reduces the number of values needed to be checked.

  An attacker could brute-force values of j which satisfy the formula $i \cdot j = 1 \pmod{\phi(n)}$ which is much less expensive than prime factorisation.

3

3.1 Discuss

CRT allows us to store our decryption formula as two parts because both halves hold all the information required to reform the original decrypted message.

$$m = y^j \pmod{n}$$

If we take $\pmod{p}$ from both sides, then we get

$$m \mod p = (y^j \mod n) \mod p$$

Because we know that p is a factor of n, we know that n will divide p, and so:

$$m \mod p = y^j \mod p$$

We can do the same but with $\pmod{q}$:

$$m \mod q = (y^j \mod n) \mod q$$

$$m \mod q = y^j \mod q$$

Therefore, we have our two halfs of m which are

$$m_p = y^j \mod p$$

$$m_q = y^j \mod q$$

3.2 Show

1. $m = y^j \pmod{p}$
2. $m = y^j \pmod{q}$
3. $m = (y^j \mod p)k_2q + (y^j \mod q)k_1p \pmod{pq}$
4. $1 = k_1p + k_2q$

Also, because we know that $m_p = y^j \pmod{p}$ and $m_q = y^j \pmod{q}$ we can write this as:

$$m = m_p k_2 q + m_q k_1 p \pmod{pq}$$

We can work out $\bar{q}$ from formula (4) because $\bar{q}q = 1 \pmod{p}$ which means we can write this as $\bar{q}q = rp + 1$. We can re-arrange this to the form $1 = \bar{q}q - rp$. This matches the structure of formula (4), and so we know that $k_1 = -r$ and $k_2 = \bar{q}$. Therefore:

$$\bar{q} = k_2$$

Therefore, we can rewrite our formulas as:

1. $m = y^j \pmod{p}$
2. $m = y^j \pmod{q}$
3. $1 = k_1 p + \bar{q}q$
4. $m = m_p \bar{q}q + m_q k_1 p \pmod{pq}$

We can then substitute $k_1 p$ in formula 3 for $1 - \bar{q}q$.

$$\begin{aligned} m &= m_p \bar{q}q + m_q(1 - \bar{q}q) \pmod{pq} \\ &= m_p \bar{q}q + m_q - m_q \bar{q}q \pmod{pq} \\ &= m_q + m_p \bar{q}q - m_q \bar{q}q \pmod{pq} \\ &= m_q + \bar{q}q(m_p - m_q) \pmod{pq} \\ &= m_q + q(\bar{q}(m_p - m_q)) \pmod{pq} \\ &= m_q + qh \pmod{pq} \end{aligned}$$

Therefore, we can recover m using the equation $m = m_q + qh \pmod{pq}$.

QED.

4

We can use Euler's thorem to optimize decryption because y and p are coprime positive integers.

Therefore, in our equation: $y^{\phi(p)} = 1 \pmod{p}$. We can then use this to factor out the $\phi(p)$ from j.

For as many multiples of $\phi(p)$ in j, we can factorize them out as they just equate to 1. This is done by rewriting j from a multiple of $\phi(p)$ plus the remainder of $j/\phi(p)$.

$$j = r\phi(p) + (j \mod \phi(p))$$

We can then work with this equation to get the following:

$$j = r\phi(n) + (j \mod \phi(p))$$

$$\begin{aligned} m_p &= y^{r\phi(p)+(j \mod \phi(p))} \pmod p \\ &= y^{r\phi(p)} y^{(j \mod \phi(p))} \pmod p \\ &= (y^{\phi(p)})^r y^{(j \mod \phi(p))} \pmod p \\ &= (1)^r y^{(j \mod \phi(p))} \pmod p \Leftarrow \text{as per Euler's theorem} \\ &= 1y^{(j \mod \phi(p))} \pmod p \\ &= y^{(j \mod \phi(p))} \pmod p \end{aligned}$$

Therefore, we can express the equation $m_p = y^j \pmod p$ in the form:

$$m_p = y^{(j \mod \phi(p))}$$

And, because p is prime, $\phi(p) = p - 1$ and so:

$$m_p = y^{(j \mod p-1)} = y^{j_p}$$

QED.

5

$$\begin{aligned} n &= 157 \cdot 163 \\ &= 25591 \\ \phi(n) &= (157-1)(163-1) \\ &= 156 \cdot 162 \\ &= 25272 \end{aligned}$$

For some k:

$$(7 \cdot j) = 1 + (k \cdot 25272)$$

Which can be written as:

$$(j \cdot 7) + ((-k) \cdot 25272) = 1$$

We can then apply the extended euclidian algorithm to find a suitable value for j and $-k$ (See *extendedEuclidian.c*).

$$j = 10831$$

$$-k = -3 \rightarrow k = 3$$

To decode $y = 2373$ using Euler's Theorem, we first need the values frof j_p and j_q:

$$\begin{aligned} j_p &= 10831 \mod (157-1) \\ &= 10831 \mod 156 \\ &= 67 \\ j_q &= 10831 \mod (163-1) \\ &= 10831 \mod 162 \\ &= 139 \end{aligned}$$

We can then use these to work out the values of m_p and m_q from the CRT $m_p = y^{j_p} \pmod p = 2373^{67} \mod 157 = 63$ $m_q = y^{j_q} \pmod q = 2373^{139} \mod 163 = 57$

We can then calculate the values of $\bar{q}$ because:

$$1 = kp + \bar{q}q = 157p + 163\bar{q}$$

To do so, we use the extended euclidian algorithm (see *extendedEuclidian.c*)

$$\bar{q} = -26$$

Then, we can use this to calculate h:

$$\begin{aligned} h &= \bar{q} \cdot (m_p - m_q) \mod p \\ &= -26 \cdot (63 - 57) \mod 157 \\ &= 1 \end{aligned}$$

Finally, we recover m using $m = m_q + (h \cdot q)$:

$$\begin{aligned} m &= 65 + (1 \cdot 163) \\ &= 65 + 163 \\ &= 220 \end{aligned}$$

The other messages decode following the same process. I wrote a program written in python, as it includes a built-in modular exponential function (see *decoder.py*)

- $y = 20360$
  $m = 521$
- $y = 19129$
  $m = 14565$
- $y = 23043$
  $m = 9$

Huffman codes

6

First, we must order our symbols and probabilities in order of increasing probability:

$$A = \{B, F, A, C, H, E, G, D\}$$

$$P = \{0.3, 0.15, 0.14, 0.12, 0.11, 0.1, 0.05, 0.03\}$$

As our code alphabet $B = \{0, 1, 2\}$ is non-binary with $|B| = k = 3$, we combine 3 symbols at each stage, except the first where we combine k_0 symbols which is found using:

$$2 \leq k_0 \leq 3$$

$$8 - k_0 \mod 3 - 1 = 0$$

This gives us $8 - k_0 \mod 2 = 0$, which must mean $k_0 = 2$. We can then construct our $S^*, A^* and P^*$ values as follows:

$$A^* = \{B, F, A, C, H, E, GD\}$$
$$P^* = \{0.3, 0.15, 0.14, 0.12, 0.11, 0.1, 0.08\}$$
$$S^* = \{A*, P*\}$$

By keeping the list ordered, we then recursively compute a Huffman code K^* for S^*:

B	F	A	C	H	E	G	D
0.3	0.15	0.14	0.12	0.11	0.1	0.05	0.03
B	F	A	C	H	E	GD	
0.3	0.15	0.14	0.12	0.11	0.1	0.08	
B	$HEGD$	F	A	C			
0.3	0.29	0.15	0.14	0.12			
FAC	B	$HEGD$					
0.41	0.3	0.29					

We then construct code words from bottom to top to get

B	F	A	C	H	E	G	D
1	00	01	02	20	21	220	221
B	F	A	C	H	E	GD	
1	00	01	02	20	21	22	
B	$HEGD$	F	A	C			
1	2	00	01	02			
FAC	B	$HEGD$					
0	1	2					

This gives us our final value of K:

$$K = \{1, 00, 01, 02, 20, 21, 220, 221\}$$

We can then work out the average code length. First, let $d_i = |K(a_i)|$, then:

$$L(S, K) = \sum_{i=1}^{n} P(a_i) \cdot d_i$$

$$\begin{aligned} L(S,K) &= (1 \cdot 0.3) + (2 \cdot 0.15) + (2 \cdot 0.14) + (2 \cdot 0.12) + (2 \cdot 0.11) + (2 \cdot 0.1) + (3 \cdot 0.05) + (3 \cdot 0.03) \\ &= 0.3 + 0.3 + 0.28 + 0.24 + 0.22 + 0.2 + 0.15 + 0.09 \\ &= 1.78 \end{aligned}$$