Ayush Jain (UNI : aj2672)
COMS W4180 Network Security
Written Assignment 1

1 Problem 1

a) Since UDP prefers speed over reliability, it may lead to packet loss and/or reordering. In such a situation **CTR** mode should be preferred because it provides for random read access(decryption of a particular cipher text does not depend on the previous one) and both encryption and decryption is parallelizable.

b) CBC should be used here because the message is small and hence can be padded with extra digits in the beginning. In that way, the decryption becomes independent of the IV used and can be easily decrypted.

c) Again, **CTR** should be used here as well. It is strong in terms of security and is parallelizable in encryption and decryption.

d) EME should be used for this use case. This is because it is more reliable as compared to XTS and CMC in terms of security. It is a bit slower as it requires two fold encryption, but at the same time it is parallelizable which can be a plus for distributed log files.

2 Problem 2

In the study, the set of collected keys were partitioned into different clusters shared the same RSA modulus. Out of about 6 million certificates collected about 200k certificates contain RSA modulus that is shared with another certificate.

We know that moduli that share one prime factor result in complete loss of security for all moduli involved. This means that two out of every one thousand RSA moduli that were collected offer no security at all.

This puts into question the assumption that different random choices are made each time keys are generated. The paper also concludes that generating keys in the real world for multiple-secrets cryptosystems such as RSA is significantly riskier than for single-secret ones such as ElGamal or (EC)DSA which are based on Diffie-Hellman.

3 Problem 3

a) The steps to encrypt and sign a file are:

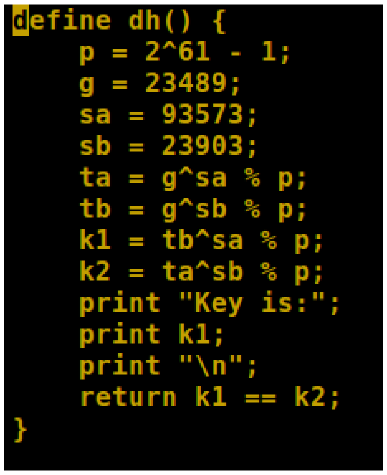
1. Using the public key algorithm P and private key privkey and public key pubkey, compute and exchange the secret K.
2. use hash function H to hash the message.
3. use block cipher B and key K to encrypt the file and produce the digital signature.

b) The steps to verify the signature on the file are:

1. Decrypt signature using secret key K and block cipher B.
2. hash original data using H
3. compare results from the two.

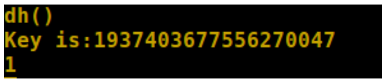
4 Problem 4

Following is the code snippet used to compute the shared key using Diffie-Hellman key exchange.



```
define dh() {
    p = 2^61 - 1;
    g = 23489;
    sa = 93573;
    sb = 23903;
    ta = g^sa % p;
    tb = g^sb % p;
    k1 = tb^sa % p;
    k2 = ta^sb % p;
    print "Key is:";
    print k1;
    print "\n";
    return k1 == k2;
}
```

The output produced using the above code was:



```
dh()
Key is:1937403677556270047
1
```

The dh() is the function call, followed by the key computed. 1 in the last line proves that the two keys were equal.

5 Problem 5

When selecting a new algorithm, the simplicity of the algorithm reduces implementation errors. It also makes sure that the costs such as power consumption, number of hardware gates and execution time is minimal.

6 Problem 6

a) It is necessary to start devising algorithms for quantum computing because it usually takes years, close to a decade to devise and accept new algorithms. This is because of the time and thoroughness that goes into testing the algorithm and pronouncing it safe. At the same time, elimination of existing algorithms also takes an equivalent amount of time. So if we do not start coming up with quantum safe algorithms, current ones dependent on mathematically hard problems will still be in use when quantum computers come around. And with quantum computers cracking such algorithms will not be a problem because of the speed of such computers.

b) Diffie-Hellman is a bigger problem and needs to be replaced with quantum safe algorithm. This is because the mathematical hard problem on which DH algorithm depends upon can be easily solved by a sufficiently large quantum computer. AES can still be effective in presence of quantum safe computers if we increase the key size.

7 Problem 7

In my view, this can be a problem, because measuring the time can be an effective hint to the range in which the shared secret lies. The algorithm are based on mathematical hardness of the problem of checking each key. However, if we know the approximate range of our key, the test range decreases and hence figuring out the key would be much simpler, thus posing a security risk.