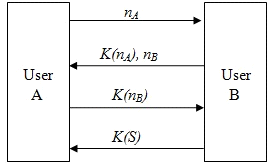
**Cryptography Systems KARTHIK SRINIVAASAN AYENGAR DEVANATHAN - 1209320095**

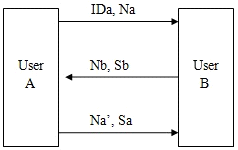
**1.**



In the above system, an attacker can send a list of nonces of his choice to User B and store the responses K(nA) and also make a note of the responses of User A for different values of nB that User B sends User A. Thereby later on, the attacker has a list of plain-texts(challenges) and their Signed responses.

Thus the attacker can then authenticate himself to User B, if an earlier repeated nonce was sent again and the corresponding response can be sent back. Upon sending the right response, user B would send him K(S), which the attacker also collects. Over a period of time the attacker can create a dictionary of new shared key S signed by K.  
  
This information can be misused and should not be made available and reduces the strength of the key K.

**2.**



The attacker can change the value of IDa to his own ID, say IDc and send it to User B posing as User A. User B thinks it is User A and sends Sb, his identity IDb (we assume that IDb has been missed out in the diagram unintentionally) and nonce Nb . Now Sb = Encskb(Nb, Na, IDc) which vouches for the attacker's identity. User A only checks if Na is present in Sb after decrypting Sb, hence, User A wouldn't know that it has been modified. Also, the attacker may choose to send IDc instead of IDb to User A. When User A sends Na' and Sa, User B wouldn't know that the attacker has changed User B's identity to his own since User B also checks only if Nb is present in Sa.

There is one more possibility for this particular question. The attacker may keep recording all the Na and Nb that he receives and record the corresponding values of Sa and Sb. The next time when he sees the users using Na or Nb, he immediately sends back Na or Nb to pose as the other user and continue his communication from then on.

**3.** The validation method explained in the situation isn’t secure. The Hash of the zip file and the file being sent are asynchronous and can be altered by the attacker. The attacker could change the original zip file with a file of his/her choice. He/ She can send the hash of his altered file to Bob. Bob would never know that he received an incorrect version of the file he wanted as the hash of the new file and the hash calculated by Bob would be the same.

The system should be modified as follows:

The zip file should be hashed and Alice should sign the document along with its hash using her Private key and send the same in a single transaction. That way the hash cannot be modified. Even if the attacker knows the public key of Alice and opens and reads the document and hash, he/she cannot modify the file as the attacker doesn’t know the private key of Alice. As an additional layer of security to prevent forging existing signatures and also encrypt , Alice can optionally choose to perturb the (document, hash) pair with a nonce concatenated at the end.

**Existing system:** 1. Send file 2. Send Hash

**New System:** 1. Find hash of document 2. Optionally concatenate (document, hash) with nonce. 3. Sign compounded data with Private key and send to recipient.

**4. RSA Particulars:**

We are given that the public keys are n=33 and e=13 which are both small primes. In order to encrypt the plain text m=6, RSA uses the function c = me mod n. Therefore, c = 613 mod 33. Since the exponential is quite high, we find the values:

1. 61 mod 33 = 6 2. 62 mod 33 = 3 3. 64 mod 33 = 9 4. 68 mod 33 = 15

From here, we can find that

613 mod 33 = (686461) mod 33 = (6 x 9 x 15) mod 33 = 18. Therefore, c is found to be 18.

We know that p\*q = 33. By the method of prime factorization, we can find that p and q are 11 and 3. Now, from the values of p and q, the value of c can be found which is (p-1)(q-1) = 20. From the following equation, we can find the value of d:

ed = 1 + ϕ(n) k

=> 13d = 1+ 20k.

By trial and error method, it can be found that d=17.

Alice's public/private pair is weak because the public components of the system are not large primes. The numbers can be guessed easily using prime factorization. Therefore, it is recommended to use large prime numbers as the public key.