6.857 Computer and Network Security **Fall Term, 1997**

Lecture 5 : September 18, 1997

Lecturer: Ron Rivest *Scribe: Roberta Braum*

1 Topics Covered

- CBC MAC
- Stream Ciphers
- Pay-TV and Related Applications
- Key Distribution Centers

2 Cipher Block Chaining Message Authentication Code (CBC-MAC)

As discussed in the last lecture, the CBC method can be used for encrypting data. The method also works fine for calculating MACs for fix length messages, but may be insecure for messages of variable length.

For example:

$f(M1, k^{)} = \mathrm{MAC1}$

$f(M2, k^{)} = \mathrm{MAC2}$

This is breakable because you can find a way to encrypt the last block again.

$f(M1, k^{)} = \mathrm{MAC1}$

$f((MAC1 \oplus M2), k^{)} = \mathrm{MAC2}$

In order to fix this problem, the last block in the chain can be decrypted with $k^{'}$ and then encrypted with k.

3 Stream Ciphers

Stream ciphers are utilzed in high speed channels that need fast encryption. Many of these ciphers are Feedback Shift Registers (FSR), and incorporate very fast pseudo random number generators. The cipher is linear if f is the $\oplus$ of the inputs. This method can guarantee a long period of $2^n - 1$ for an n-stage FSR. The weakness of the linear case is that it is insecure once you have more than 2n bits of data.

In order to improve on this method, it is useful to use non-linear feedback. This has a shorter period, though, as illustrated by the birthday paradox.

Birthday Paradox Take elements at random from a set of size n, replacing each one afterwards. After drawing around square root of n elements, you will have seen a repetition. The period in this case is $2^{n/2}$.

Additionally, methods can be combined such as having three shift registers - A, B, and C. Shift register A would be used to determine which of B or C the output would come from. The message can also be decimated by running until you have encountered a 1, then outputting the next bit.

3.1 RC4

RC4 is a variable-size stream cipher. The algroithm utilizes the Ouput Feedback mode (OFB) and the key stream is independent of the plaintext. It has an 8*8 S-box, which has 255 1-byte positions. The entries are a permutation of the numbers 0 through 255, and the permutation is a function of the variable length key. The algorithm has two counters i and j, which are initialized to 0. The S-box is initialized by first filling up the S values such that S[0] = 0, s[1] = 1...s[255] = 255. Then fill another 256 byte array with the key, repeating the key as necessary to fill the entire array.

j = 0

for i = 0 to 255

j = (j + Si +Ki) mod 256

swap (Si, Sj)

Once the table is created, random bytes can be created by:

i =(i+1) mod 256

```
j = (j+S[i]) mod 256
output S(S[i] +S[j])
swap(S[i], S[j])
Repeat
```

RC4 has similar error properites to a one-time pad. The algorithm is immune to differenetial and linear cryptanalysis, and is highly non-linear.

4 Pay-TV and Related Applications

One area in which stream ciphers may be used is for Pay-TV. The programs being distributed can be encrypted so that only certain viewers may watch a particular program or channel. In order to do this, the client must have a unique key which is shared with the TV center. Program keys can be changed dynamically through a secure channel between the client and the center.

For instance, if Alice wants to watch pay-per-view, she can send a message to the center through the secure channel requesting the key. The center then responds with the key for the PPV channel encrypted and MAC'd with Alice's set- top box key. (For services such as HBO, a longer term group key may be used.)

Software distribution via CD can also be key encrypted. Each program on the CD can have a separate key, and the user can purchase the key for a particular desired program. A problem with this distribution method is that the CD does not have a unique key, and the keys could be posted on the internet.

5 Key Distribution Center

One problem with maintaining security is the creation and distribution of keys. One way of handling this is through a Key Distribution Center (KDC). In order for this to work, the KDC must be a trusted entity, and be accessible by all parties. Examples of how the KDC works can be illustrated through the users Alice(A) and Bob(B). Alice shares Ka with KDC and Bob shares Kb with the KDC.

Methodologies of communication: 1) Have KDC encrypt/decrypt messages to and from Alice and Bob. KDC is the intermediary.

m1 $A->KDC$: $"Hi, Bob..."[Ka]$

m2 $KDC->Bob$: $"Hi, Bob.."[Kb]$

The message is encrypted and MACed by Alice with her key [Ka], the message is sent to the KDC, who decrypts the message.

2) Have KDC help Alice and Bob establis a common key Kab.

m1 $A->KDC$: $"IwanttotalktoBob"[KA]$

m2 $KDC->Bob$: $"Alicewantstotalkprivatelytoyou, usekeyKab = 029"[KB]$

m3 $KDC->Alice$: $"UsekeyKab = 029totalktoBob"[Ka]$

m4 $A->B$: $"Hi"[Kab]$

Variations:

- Alice making up the key.
- Alice could be sent m3, who forwards it to Bob
- IDs could be sent in clear so that Bob could know which key to use
- If there are multiple keys/sender, the keys may need IDs
- Keys could be made up by Alice, Bob, the KDC or a combination
- Expiration. How long should Bob keep 'Kab'.

One way of handling the expiration problem would be:

m1 $A->KDC : A, keyID, A, B, "request", newkeyid, newkey, timeoutKa$

m2 $KDC->A : C, keyID, C, A, "reply", newkeyid, newkey, timeoutKa$

m3 $KDC->B : C, keyID, C, B, "newkey", newkeyid, newkey, timeoutKb$

m4 $A->B : A, newKeyID, "Hi"newkey$

where Key denotes an encrypted MACed message.

The KDC can be used for establishing keys with other KDCs. The KDCs establish contact with each other, and key transactions are handled at a higher level.

Problems with the KDC method is that the KDC must be trusted completely, must be available, and that there are no digital signatures.