Stream Ciphers

- Data is encrypted a bit or byte at a time
- **Confusion sequence** is created at encryption time and is combined with the plaintext to form the ciphertext stream.
- The key allows recreation of the confusion sequence for recreation of the plaintext at the receiving end.
One Stream Cipher

- Parties share key k.
- RC4 is an encryption algorithm that deterministically generates a string of random bits given two parameters
- An initialization vector is created by the originator
- $CT = PT \oplus RC4(IV, k)$
- Transmit $(CT, IV)$

Stream Cipher with RC4

| Plaintext | CRC | $\oplus$ | RC4 (v,k) | $\oplus$ | RC4 (v,k) | $\oplus$ | Plaintext | CRC |

RC4 Algorithm

1. $i := (i + 1) \mod 256$
2. $j := (j + S_i) \mod 256$
3. swap $S_i$ and $S_j$
4. $t = (S_i + S_j) \mod 256$
5. $K = S_t$
RC4 Demonstration

Algorithm
1. i := (i + 1) mod 8
2. j := (j + S_i) mod 8
3. swap S_i and S_j
4. t = (S_i + S_j) mod 8
5. K = S_t

---

RC4 Demonstration

1. i := (i + 1) mod 8
2. j := (j + S_i) mod 8
3. swap S_i and S_j
4. t = (S_i + S_j) mod 8
5. K = S_t

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>2</td>
<td>4</td>
<td>6</td>
<td>5</td>
<td>1</td>
<td>0</td>
<td>7</td>
</tr>
</tbody>
</table>

i = 1
j = 7
t_1 = 2 + 7 mod 8 = 1
K_1 = S_t = 2 = 010

---

RC4 Demonstration

Second 3 bits

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>2</td>
<td>6</td>
<td>4</td>
<td>5</td>
<td>1</td>
<td>0</td>
<td>7</td>
</tr>
</tbody>
</table>

i = 2
j = 3
t_2 = 6 + 4 mod 8 = 2
K_2 = S_t = 6 = 110
RC4 Demonstration

1. \( i := (i + 1) \mod 8 \)
2. \( j := (j + S_i) \mod 8 \)
3. swap \( S_i \) and \( S_j \)
4. \( t := (S_i + S_j) \mod 8 \)
5. \( K = S_t \)

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>2</td>
<td>6</td>
<td>7</td>
<td>5</td>
<td>1</td>
<td>0</td>
<td>4</td>
</tr>
</tbody>
</table>

\( i = 3 \quad j = 7 \)
\( t_2 = 7 + 4 \mod 8 = 3 \quad K_3 = S_1 = 7 = 111 \)

Confusion Sequence

- **One Time Pad**
  - Generate one random bit at a time & XOR
  - One time pad, with all its up and down sides
- **Keyed Deterministic Random Bit Generator**
  - Use the same RBG and the same seed (IV \( \oplus \) key)
  - Both sides generate the same random bit stream

Hash

- Function that transforms any size message input into a fixed size message output
- Since the output is usually smaller than the input, it is called a Message Digest
- Deterministic: The same input will ALWAYS produce the same digest
- One way: Given the output, it is computationally infeasible to find the input
Some Hash Functions Producing 8 Bit Digests

1. Take the first 8 bits of the message

2. Add 511, then take the first 8 bits

3. XOR 8 bits at a time

<table>
<thead>
<tr>
<th>Message</th>
<th>Digest</th>
</tr>
</thead>
<tbody>
<tr>
<td>catapillar</td>
<td>0110 0011 0110 0001 0111 0100 1111 1111</td>
</tr>
<tr>
<td>a</td>
<td>0110 0010</td>
</tr>
<tr>
<td>t</td>
<td>0000 0010</td>
</tr>
<tr>
<td>g</td>
<td>0111 0110</td>
</tr>
</tbody>
</table>

Secure Hash

- The Function is One Way
- The Digest is Random
- The Function is Sparse
- Diffusion Occurs

Secure Hash

- The Function is One Way
  - There is no practical way to find input that corresponds to a given output.
  - Distinct from encryption in that
    - A hash function is one-way absent the "backdoor"
    - The output is smaller than the input
Secure Hash

- The Function is One Way
- The Digest is Random
  1. Each output should, with high probability, have about half of the bits on.
  2. For arbitrary inputs, the likelihood that any given bit will be on is about 50%.
  3. Any two outputs should be uncorrelated
  4. For any two outputs, about half the bits should differ
  5. Etc.

Secure Hash

- The Function is One Way
- The Digest is Random
  - The Function is Sparse:
    There is no practical way to find two inputs that map to the same output

Sparse Digest

- Minimum 128 bits
- Requires searching $2^{64}$ messages to have 50% chance of finding two messages that are the same
Secure Hash

- The Function is One Way
- The Digest is Random
- The Function is Sparse
- Diffusion Occurs

A single bit change in the input ensures that each bit of the output changes with 50% probability

Hashing for Integrity

1. Originator
   1. Hash the message to produce the digest
   2. Transmit both message and digest

2. Recipient
   1. Re-hashes the message
   2. Compare the value to the received digest

Using a Hash to Ensure Integrity
Hashing for Secure Integrity

1. Originator
   1. Hash the message using the shared key to produce the digest
   2. Transmit both message and digest

2. Recipient
   1. Receive the message
   2. Re-hash the message using the shared key
   3. Compare the value to the received digest

Hashing for Secure, Private Integrity

1. Originator
   1. Hash the message to produce the digest
   2. Encrypt the message
   3. Transmit ciphertext and digest

2. Recipient
   1. Decrypts the message
   2. Re-hashes the message
   3. Compares the digest to the received digest
Using a Hash for Authentication

1. Verifier
   1. Generate/transmit a random number to the prover
   2. XOR the random number with the shared key
   3. Hash the result to produce a digest

2. Prover will
   1. XOR the random number with the shared key
   2. Hash the result to produce the digest
   3. Transmit the digest to the verifier

3. Verifier
   1. Compare the generated and received digests

*Start by sending $R_1$
Some Secure Hashing Algorithms

- MD2  ➢ 8 bit words, 128 bit digest, 2 passes
- MD4  ➢ 32 bit words, 128 bit digest, 3 passes
- MD5  ➢ 32 bit words, 128 bit digest, 4 passes
- SHS  ➢ 32 bit words, 160 bit digest, 5 passes

Review

- Hashing Defined
- Hash Uses
- Hash Functions

Preview

- Public Key Cryptography