Cristina Nita-Rotaru

CS355: Cryptography

Lecture 6: Stream ciphers.

Modern cryptography

- One-time pad requires the length of the key to be the length of the plaintext and the key to be used only once. Difficult to manage.
- Alternative: design cryptosystems, where a key is used more than once.
- What about the attacker? Resource constrained, make it infeasible for adversary to break the cipher.

Stream ciphers

- In OTP, a key is described by a random bit string of length n
- Stream ciphers:
  - Idea: replace "rand" by "pseudo rand"
  - Use Pseudo Random Number Generator
  - PRNG: $\{0,1\}^s \rightarrow \{0,1\}^n$
    - expand a short (e.g., 128-bit) random seed into a long (e.g., 10^6 bit) string that "looks random"
  - Secret key is the seed
  - $E_{seed}[M] = M \oplus PRNG(seed)$

Properties of stream ciphers

- Do not have perfect secrecy
  - Security depends on PRNG
- PRNG must be "unpredictable"
  - Given consecutive sequence of bits output (but not seed), next bit must be hard to predict
- Typical stream ciphers are very fast
- Used in many places, often incorrectly
  - DVD (LFSR), SSL(RC4), WEP (RC4), etc.

Weaknesses of stream ciphers

- If the same keystream is used twice ever, then easy to break – decipher the text.
- Highly malleable
  - Easy to change ciphertext so that plaintext changes in predictable, e.g., flip bits
- Weaknesses exist even if the PRNG is strong

Randomness and pseudorandomness

- Random is not a property of one string
  - Is "000000" "less random" than "011001"?
  - Random is the property of a distribution, or a random variable drawn from the distribution
- Similarly, pseudo-random is property of a distribution
- We say that a distribution D over strings of length-l is pseudorandom if it is indistinguishable from a random distribution.
- We use "random string" and "pseudorandom string" as shorthands

Distinguisher

- A distinguisher D for two distributions works as follows:
  - D is given one string sampled from one of the two distributions
  - D tries to guess which distribution it is from
  - D succeeds if guesses correctly
- How to distinguish a random binary string of 256 bits from one generated using RC4 with 128 bites seed?

Pseudorandom generator definition

- We say an algorithm G, which on input of length n outputs a string of length l(n), is a pseudorandom generator if
  1. For every n, $l(n) > n$
  2. For each PPT distinguisher D, there exists a negligible function negl such that

     $|Pr[D(r)=1] - Pr[D(G(s))=1]| \leq negl(n)$

  Where r is chosen at uniformly random from $\{0,1\}^{l(n)}$ and s is chosen at uniform random from $\{0,1\}^s$

Variable length messages

- A variable output-length pseudo-random generator is $G(s, 1^l)$ that output l such that
  - Any shorter output is the prefix of the longer one
  - Fix any length, this is a pseudo-random generator
- Given such a generator, can encrypt messages of different length by choosing l to be length of the message.

Multiple encryptions

- How to encrypt multiple messages with one key?
  - What is wrong with using the standard way of using stream cipher to encrypt?
- How to define secure encryption with multiple messages?
- No deterministic encryption scheme is secure for multiple messages

Single message vs. multiple messages

- Give an encryption scheme that has indistinguishable encryptions in the presence of an eavesdropper
  - i.e., secure in single message setting
- But does not have indistinguishable multiple encryptions in the presence of an eavesdropper.
  - i.e., insecure for encrypting multiple messages?

- No deterministic encryption scheme is secure for multiple messages

Multiple messages: Synchronized mode

- Use a different part of the output stream to encrypt each new message
- Sender and receiver needs to know which position is used to encrypt each message
- Often problematic

Multiple messages: Unsynchronized mode

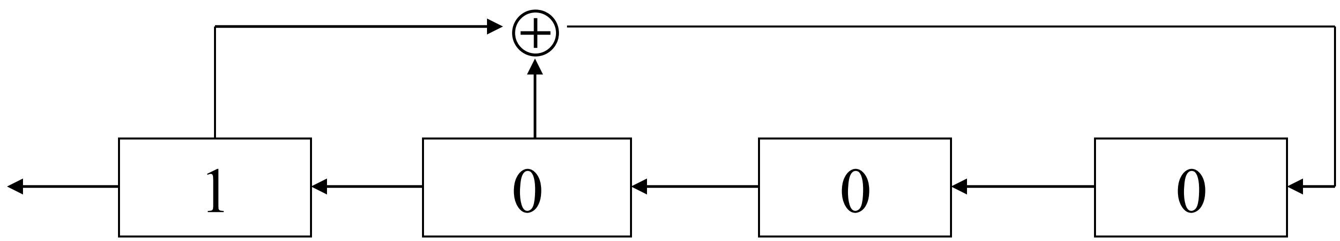
- Use a random Initial Vector (IV)
- $\mathbf{Enc}_k(m) = \langle IV, G(k,IV) \oplus m \rangle$
  - IV must be randomly chosen, and freshly chosen for each message
  - How to decrypt?
- What G to use and under what assumptions on G such a scheme has indistinguishable multiple encryptions in the presence of an eavesdropper
  - What if $G(k,IV) \equiv G'(k||IV)$, where G' is a pseudorandom generator

Security of unsynchronized mode

- Recall that IV is sent in clear, so is known by the adversary
- For each IV, $G(\cdot,IV)$ is assumed to be pseudorandom generator;
- Furthermore, when given multiple IVs and outputs under the same randomly chosen seed, the combined output must be pseudo-random
- Stream ciphers in practice are assumed to have the above **augmented pseudorandomness** property and used this way

Linear Feedback Shift Register (LFSR)

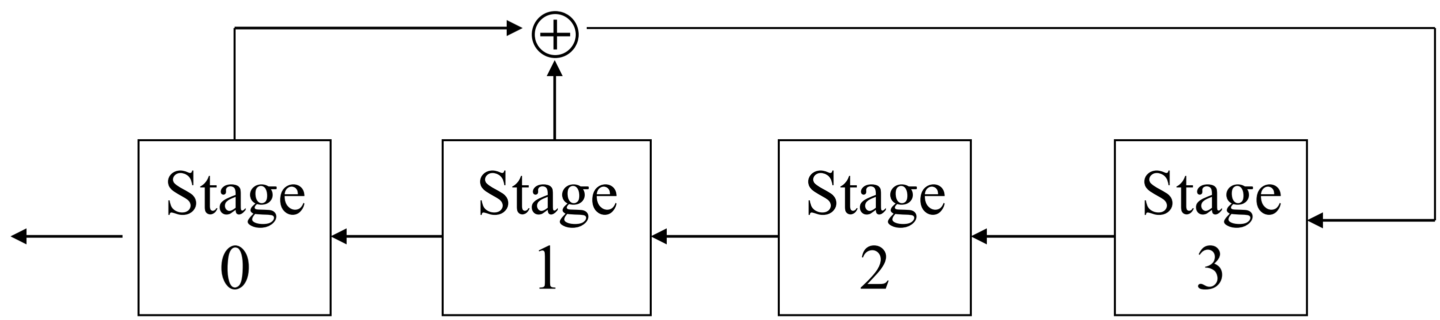
- Example:



  - Starting with 1000, the output stream is
    - 1000 1001 1010 1111 000
  - Repeats every $2^4 - 1$ bit
  - The seed is the key, in this case 1000

Linear Feedback Shift Register (LFSR)

- Example:



- $z_i \quad = (z_{i-4}+z_{i-3}) \bmod 2$
  $\quad\quad = (0 \cdot z_{i-1} + 0 \cdot z_{i-2} + 1 \cdot z_{i-3} + 1 \cdot z_{i-4}) \bmod 2$

- We say that stages 0 & 1 are selected.

Properties of LFSR

- Fact: given an L-stage LFSR, every output sequence is periodic if and only if stage 0 is selected
- Definition: An L-stage LFSR is maximum-length if some initial state will results a sequence that repeats every $2^L - 1$ bit
- Whether an LFSR is maximum-length or not depends on which stages are selected

Cryptanalysis of LFSR

- Vulnerable to know-plaintext attack
- A LFSR can be described as

$$z_{m+i} = \sum_{j=0}^{m-1} c_j \, z_{i+j} \bmod 2$$

- Knowing 2*m* output bits, one can
  - Construct *m* linear equations with *m* unknown variables $c_0, \ldots, c_{m-1}$
  - Recover $c_0, \ldots, c_{m-1}$

Cristina Nita-Rotaru

Cryptanalysis of LFSR

- Given a 4-stage LFSR, we know
  - $z_4=z_3c_3+z_2c_2+z_1c_1+z_0c_0 \bmod 2$
  - $z_5=z_4c_3+z_3c_2+z_2c_1+z_1c_0 \bmod 2$
  - $z_6=z_5c_3+z_4c_2+z_3c_1+z_2c_0 \bmod 2$
  - $z_7=z_6c_3+z_5c_2+z_4c_1+z_3c_0 \bmod 2$
- Knowing $z_0,z_1,\ldots,z_7$, one can compute c_0,c_1,c_2,c_4.
- In general, knowing 2n output bits, one can solve an n-stage LFSR

$$z_j = c_1 z_{j-1} + c_2 z_{j-2} + \cdots + c$$

RC4

- A proprietary cipher owned by RSA DSI, designed by Ron Rivest.
- Simple and effective design.
- Variable key size, byte-oriented stream cipher.
- Widely used (web SSL/TLS, wireless WEP).
- Key forms random permutation of all 8-bit values.
- Uses that permutation to scramble input info processed a byte at a time.

RC4 Key Schedule

- Walks each entry in an array S of numbers: 0..255 turn, using its current value plus the next byte of key to pick another entry in the array, and swaps their values over.
- Total number of possible states is 256!, very big number
- S forms **internal state** of the cipher, L is the size of the key k

```
for i = 0 to 255 do
    S[i] = i
j = 0
for i = 0 to 255 do
    j = (j + S[i] + k[i mod L])(mod 256)
    swap (S[i], S[j])
```


Cristina Nita-Rotaru

RC4 encryption

- Encryption continues shuffling array values
- Sum of shuffled pair selects the "stream key" byte value
- XOR with next byte of message to en/decrypt

$i = j = 0$

for each message byte m_i

$i = (i + 1) \pmod{256}$

$j = (j + S[i]) \pmod{256}$

$swap(S[i], S[j])$

$t = (S[i] + S[j]) \pmod{256}$

$C_i = m_i \oplus S[t]$

RC4 cryptanalysis

- The algorithm was kept secret however…
- In 1994 the source code was leaked on the to *cyberpunks* mailing list.
- The external analysis of RC4 was done on the source code that leaked in 1994.
- Fluhrer showed two weaknesses:
  - The first byte generated by RC4 leaks information about individual key bytes.
  - Found a large number of weak keys, in which knowledge of a small number of key bits suffices to determine many state and output bits with non-negligible probability.

Fluher, Mantin, and Shamir Attack

- This is an known-plaintext attack against RC4, that allows attackers to eventually recover a key.
- Attack is based on an assumption that the attacker is able to guess the first byte of plaintext used by the victim.
- Stubblefield, Ionnandis, and Rubin showed that the attack is possible in practice

Take home lessons

- Keystream should never be reused for stream ciphers
- When encrypting with a stream cipher in unsynchronized mode IV must be randomly chosen, and freshly chosen for each message
- LFSR is vulnerable to known plaintext attacks

Example: WEP

Wired Equivalent Privacy

- Security goals: protect link-level transmission
  - Confidentiality
  - Access control
  - Data integrity
- Security relies on the difficulty of discovering the secret key through a brute-force attack
- Uses stream cipher RC4 for encryption and CRC32 for integrity

WEP details

- RC4 is a stream cipher: based on key k and initialization vector (IV) v, generates a keystream RC4(v,k)
- To send a message M from A to B
  - Compute integrity checksum (CRC32): c(M)
  - plaintext P = {M, c(M)}
  - Encrypt P using RC4: ciphertext $C = P \oplus RC4(v,k)$
  - Transmit **$C' = v, (P \oplus RC4(v,k))$**
- To decipher an encrypted message C', the encryption process is reversed

Some observations

- The integrity check does not depend on a key, but just on the message M, so anybody can create a pair M and CRC32(M)
- The WEP standard specifies 64-bit key = **40** bit key and 24 IV. Some vendors implemented 128-bit keys (24 IV and **104** bit key).
- The IV is sent in clear, so is available to the attacker as well.

Risk of keystream reuse

$$C1 = P1 \oplus RC4(v, k)$$
$$C2 = P2 \oplus RC4(v, k)$$
$$C1 \oplus C2 = P1 \oplus P2$$

- If P1 or P2 is also known by the attacker, the other plaintext is easy to compute
- If n ciphertexts using the same keystream are available makes reading traffic easier (frequency analysis, etc)
- *Find plaintext P and the encryption C with keystream k, then it is easy to decipher any ciphertext C' encrypted with the same keystream k.*

Is keystream reused?

- The pseudorandom keystream is based on the shared key k and the initialization vector IV. Since the key k is secret and is difficult to be changed for every packet, changing the IV is important to prevent keystream reuse.
- The IV is sent in clear, so is available to the attacker as well.
- The WEP standard recommends, but does not require that the IV be changed every packet, also does not say anything about how to select the IV.
- An implementation can reuse the same IV for all packets without risking non-compliance !

24-bit IV space

- Busy access point sending 1500 byte packets, at an average of 2 Mbps, exhausts the IV space in **half a day.**
- Random generation of IV can produce collisions every **5000** packets (due to the *birthday paradox*).
- Many implementations use for IV a counter that is incremented for each packet sent and reset every time the card is inserted in the computer.

Exploiting keystream reuse

- Methods to obtain pairs (plaintext, ciphertext):
  - IP fields predictable: login sequences, recognize shared libraries transfer
  - Send email and wait for the user to check it via wireless links
  - Send data to access-points that have access control disables and observe the encrypted data

Dictionary attack

- Goal: Decrypt traffic
- How: Store keystream in a table, indexed by IV.
- Remember the IV is sent it clear
- When the attacker sees a packet with an IV stored already in the table, look up the corresponding keystream, XOR it against the packet, and read the data!
- Table is at most 1500 * 2^24 bytes = 24 GB

Packet modification

- CRC32 is linear: $c(M \oplus D) = c(M) \oplus c(D)$
- Message M was transmitted, and the ciphertext was C and the IV was IV, C and IV are known to the adversary.
- Attacker can find C' s. t. it decrypts to $M' = M \oplus D$
  D = arbitrarily chosen by the attacker
- $C' = C \oplus \langle D, c(D) \rangle$
  $= RC4(v,k) \oplus \langle M, c(M) \rangle \oplus \langle D, c(D) \rangle$
  $= RC4(v,k) \oplus \langle M \oplus D, c(M) \oplus c(D) \rangle$
  $= RC4(v,k) \oplus \langle M', c(M \oplus D) \rangle$
  $= RC4(v,k) \oplus \langle M', c(M') \rangle$

Packet injection

- The attacker knows the keystream, he can select any message and compute CRC of the message without knowing the key.
- The base station will accept the packet as valid

WEP authentication

- *Base station verifies that a client joining the network really knows the shared secret key k.*
- The base station sends a challenge string to the client, and the client sends back the encrypted challenge
- The base station checks if the challenge is correctly encrypted, and if so, accepts the client.
- If adversary sees a challenge/response pair for a given key k; he can perform the packet injection attack previously describe, and trick the base station.

Lessons learnt

- Engineering network protocols vs. security:
  - CRC-32 and RC4 are fast and simple, but they have problems
  - Being stateless is good for networking, but dangerous for security because they give an attacker more freedom
- Learn from previous works: see IPSEC, TLS.
- Public review is important: international standards should be examined by the cryptographic community

3G encryption also a stream cipher

- 2010, reports of a new attack that had "broken Kasumi" (also known as A5/3), the standard encryption algorithm used to secure traffic on 3G GSM wireless networks, by means of a sandwich attack (a type of related-key attack), allowing them to identify a full key

Take home lessons

- The strongest attack is finding the key just by observing the traffic and exploiting a known-attack on RC4, the encryption algorithm
- Decrypting traffic looking for pairs of plaintext, ciphertext and look for text encrypted with the same keystream
- Packet modification and injection

  exploiting the fact that integrity

  was implemented using CRC32