Topic 1.1
Symmetric encryption – Stream ciphers

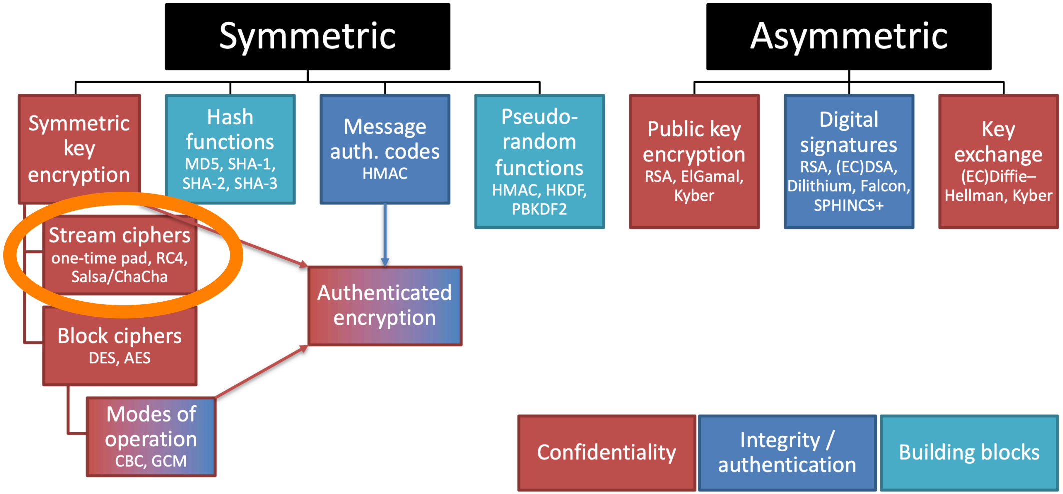
Douglas Stebila

CO 487/687: Applied Cryptography

Fall 2024

UNIVERSITY OF WATERLOO

Map of cryptographic primitives



Symmetric
Asymmetric
Symmetric key encryption
Hash functions MD5, SHA-1, SHA-2, SHA-3
Message auth. codes HMAC
Pseudo-random functions HMAC, HKDF, PBKDF2
Public key encryption RSA, ElGamal, Kyber
Digital signatures RSA, (EC)DSA, Dilithium, Falcon, SPHINCS+
Key exchange (EC)Diffie–Hellman, Kyber
Stream ciphers one-time pad, RC4, Salsa/ChaCha
Block ciphers DES, AES
Modes of operation CBC, GCM
Authenticated encryption
Confidentiality
Integrity / authentication
Building blocks

Outline

Pseudorandom bit generator

Basic idea: Instead of using a random key in the one-time pad, use a “pseudorandom” key.

Definition

A pseudorandom bit generator (PRBG) is a deterministic algorithm that takes as input a short random seed, and outputs a longer pseudorandom sequence, also known as a keystream.

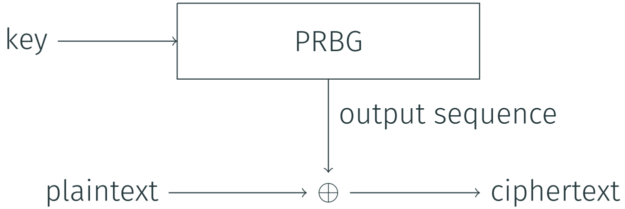
Equivalent terminology:

- pseudorandom bit generator $\equiv$ pseudorandom number generator $\equiv$ keystream generator $\equiv$ deterministic random bit generator
- seed $\equiv$ key
- pseudorandom output sequence $\equiv$ keystream

Stream cipher

We can use a pseudorandom bit generator to construct a stream cipher.

The seed is the secret key shared by Alice and Bob.



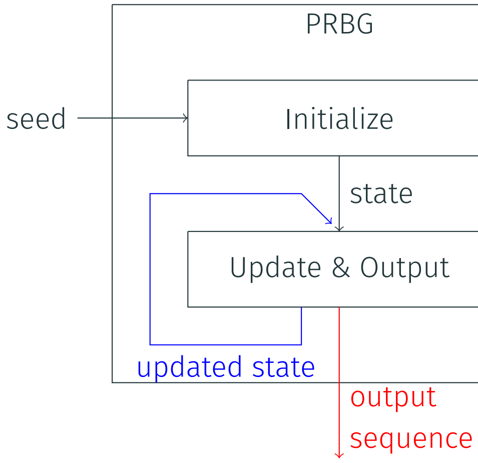
Compared to the one-time pad, a stream cipher will not have perfect secrecy.

Instead, security depends on the quality of the PRBG.

Security Requirements for the PRBG

- Indistinguishability: The output sequence should be indistinguishable from a random sequence.
- Unpredictability: If an adversary knows a portion c_1 of ciphertext and the corresponding plaintext m_1, then she can easily find the corresponding portion $k_1 = c_1 \oplus m_1$ of the output sequence. Thus, given portions of the output sequence, it should be infeasible to learn any information about the rest of the output sequence.

Components of a pseudorandom bit generator



Definition (Pseudorandom bit generator)

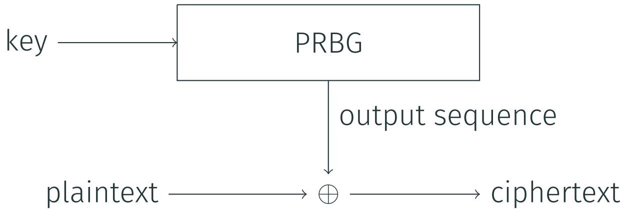
A pseudorandom bit generator consists of:

- **Initialize**(seed) $\rightarrow$ state:
  A deterministic algorithm that takes as input a seed and outputs a state
- **Update&Output**(state) $\rightarrow$ (state′, out):
  A deterministic algorithm that takes as input a state and outputs an updated state and a output sequence a.k.a. keystream.

Call **Update&Output** as many times as needed until we have a sufficiently long

Stream ciphers from PRBG's with 1 input

Original stream cipher designs used a pseudorandom generator with just one input: the key.

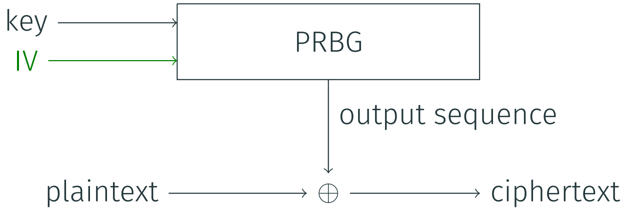


Drawbacks:

- PRBG is deterministic so each key produces a single output sequence
- $\implies$ cannot use the same key to encrypt multiple messages

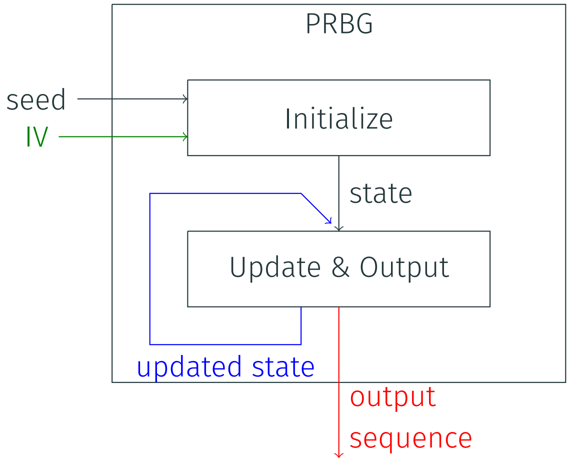
Stream ciphers from PRBG's with 2 inputs

Modern stream cipher designs use pseudorandom generators with two inputs: the key and an initialisation vector.



The key is kept private, but the IV can be made public.
$\Longrightarrow$ Can generate multiple (different) output sequences from the same key but different IV's.

Components of a pseudorandom generator with 2 inputs



Definition (Pseudorandom bit generator)

A 2-input PRBG consists of:

- **Initialize**(seed, IV) $\rightarrow$ state:
  A deterministic algorithm that takes as input a seed and initialization vector and outputs a state
- **Update&Output**(state) $\rightarrow$ (state$'$, out):
  A deterministic algorithm that takes as input a state and outputs an updated state and a output sequence a.k.a. keystream.

Stream ciphers from PRBG's with 2 inputs

Pros:

- can reuse the same key with different IVs
  $\implies$ can safely encrypt many messages using the same key with different IVs

Cons:

- need to transmit the IV with the ciphertext
  $\implies$ constant overhead

Outline

Linear feedback shift registers (LFSR)

- An LFSR is a common component in design of stream ciphers.
- LFSRs are simple and fast to implement in hardware and software and have good pseudorandomness properties.
- Any infinite periodic sequence (s_t) can be defined via a recurrence relation:

$$s_{t+n} = c_{n-1}s_{t+n-1} \oplus c_{n-2}s_{t+n-2} \dots c_1 s_{t+1} \oplus c_0 s_t$$

  where c_i are binary constants.
- The vector $(s_0, s_1, \dots, s_{n-1})$ is called the initial state vector.
- The initial state vector together with the above equation defines all terms of the sequence.

LFSR example

Suppose that the sequence (s_t) defined by the recurrence relation

$$s_{t+6} = s_{t+5} \oplus s_{t+4} \oplus s_{t+1} \oplus s_t.$$

Some example terms:

$$\begin{aligned} s_6 &= s_5 \oplus s_4 \oplus s_1 \oplus s_0 \\ s_7 &= s_6 \oplus s_5 \oplus s_2 \oplus s_1 \\ s_8 &= s_7 \oplus s_6 \oplus s_3 \oplus s_2 \end{aligned}$$

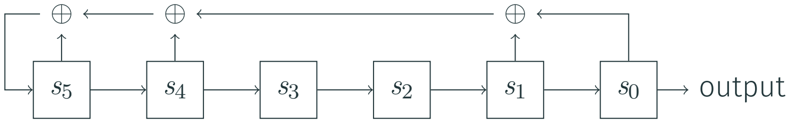
and so forth.

Example: When the initial state is 010110, the output sequence is

$$010110010101001001111000001101110011000111010111111011010001000 \ldots$$

Linear feedback shift registers

LFSR diagram for the recurrence relation $s_{t+6} = s_{t+5} \oplus s_{t+4} \oplus s_{t+1} \oplus s_t$



This represents the following algorithm:

Update&Output(state=s)

```
1 : $output \leftarrow s_0$
2 : $s'_5 \leftarrow s_5 \oplus s_4 \oplus s_1 \oplus s_0$
3 : $\textbf{for } i = 0 \ldots 4 : s'_i \leftarrow s_{i+1}$
4 : $\textbf{return } (state' = s', output)$
```

Period

A deterministic pseudorandom bit generator produces a periodic sequence, meaning it repeats after a while.

Definition

The period of a binary sequence $s(t)$ is said to be p if $s_{t+p} = s_t$ for all t, and p is the smallest such number.

Example

The sequence

$$0111\,0111\,0111\,0111\,0111\ldots$$

has a period of 4.

Attack on small period

For a good stream cipher we need that the period of the keystream output sequence is larger than the length of the plaintext.

- If the period of the keystream is less than the length of a ciphertext then two sections of message are encrypted using the same portion of keystream.
- By XORing these two sections, the keystream cancels out and one obtains the XOR of the plaintext strings.
- It is then possible to attack such a sum by exploiting the redundancy of the plaintext.

Security Requirements for the PRBG

Recall our main security requirements for a PRBG:

- Indistinguishability: The output sequence should be indistinguishable from a random sequence.
- Unpredictability: Given portions of the output sequence, it should be infeasible to learn any information about the rest of the output sequence.

In order to achieve this, the sequence generated by the PRBG must have:

1. large period;
2. large linear complexity;[1]
3. random noise-like characteristics.[2]

[1] Linear complexity: degree of the smallest recurrence relation that could generate the sequence

[2] Typically measured by statistical tests.

Pseudorandom generation using LFSRs

A single LFSR on its own is not a good pseudorandom generator:

- not very high linear complexity
- knowing the output leads to knowing the intermediate state so you can predict all future output

Three methods of using LFSRs to form a pseudorandom generator:

1. Multiple outputs from the sequence can be sampled and passed through a non-linear filter.
2. Multiple LFSRs can be advanced separately using an irregular clock. (Example to follow.)
3. The outputs from multiple LFSRs can be passed through a non-linear combiner.
    - Example: The A5 cipher used in the GSM mobile phone standard.

Outline

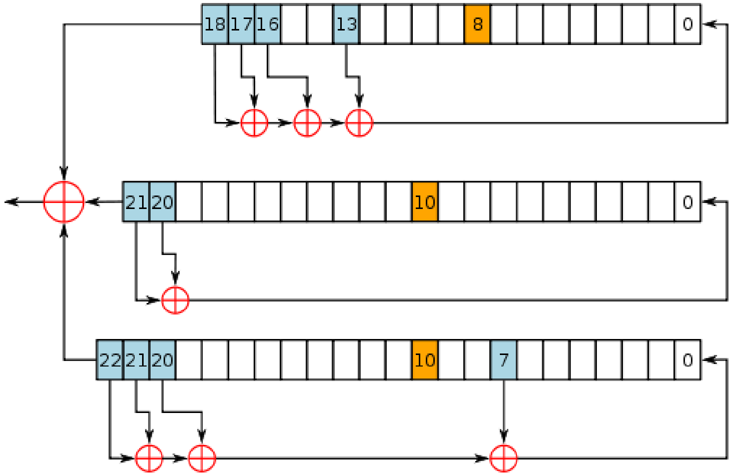
A5 cipher

- The A5 cipher is a binary stream cipher applied today in most GSM mobile telephones.
- A5 has three variants:
    1. A5/1 is the original algorithm defined in 1987
    2. A5/2 is a *weakened* version of A5/1, originally intended for deployment outside Europe, but no longer allowed under GSM standards
    3. A5/3 is a recent algorithm for deployment in third generation mobile systems.
- The design of A5/1 was originally kept confidential by its designers but became public in 1994.

A5/1 design

- The A5/1 algorithm uses three LFSRs whose output is combined.
- The three LFSRs are irregularly clocked which means that the overall output is non-linear.
- Each LFSR has a clocking bit (coloured orange in the following diagram). Each LFSR is clocked (i.e., updated using the Update function) if its own clocking bit is in majority agreement with the clocking bits of the other LFSRs.
- The key length is 64 bits, which define the state vector for each LFSR. In deployment it is said the 10 bits of the key are *fixed*. (Note that each LFSR must have a non-zero initial state.)

A5/1 keystream generator (courtesy of Wikimedia Commons)



Cryptanalysis of A5/1

- In 1997, Golic published the first cryptanalysis of A5/1, requiring much less time ($2^{40.16}$) than brute force search (2^{64}).
- ...
- In 2011, a normal PC with a fast GPU and terabytes of flash memory can break an A5/1-encrypted GSM session in a few seconds with probability above 90%.

Subsequent standards

- For 3G: A5/3 a.k.a. KASUMI
- For 4G: SNOW

Outline

The RC4 Stream Cipher

Designed by Ron Rivest in 1987.

- Was widely used in commercial products from its inception until mid 2010s.
- Pros: Extremely simple; extremely fast; variable key length.
- Cons: Design criteria are proprietary; not much public scrutiny until the year 2001. Has small biases exploitable with ≈millions of ciphertexts. Not recommend for use today.
- RC4 has two components: A key scheduling algorithm (Initialize), and a keystream generator (Update&Output).

RC4 Key Scheduling Algorithm

```
Initialize($K = K[0], K[1], \ldots, K[d-1]$)
1 : // $K[i]$, $\overline{K}[i]$ and $S[i]$ are 8-bit integers (bytes)
2 : for $i = 0, \ldots, 255$ :
3 :     $S[i] \leftarrow i$
4 :     $\overline{K}[i] \leftarrow K[i \bmod d]$
5 : $j \leftarrow 0$
6 : for $i = 0, \ldots, 255$ :
7 :     $j \leftarrow (\overline{K}[i] + S[i] + j) \bmod 256$
8 :     Swap($S[i], S[j]$)
9 : return $S$    // a 256-byte array $S[0], S[1], \ldots, S[255]$
```

Idea: S is a "random-looking" permutation of $\{0, 1, 2, \ldots, 255\}$ that is generated from the secret key

I don't want you to memorize this algorithm or look into the details too much. Some basic takeaways:

- Simple description
- Easy to implement in software
- But not secure any more

RC4 Keystream Generator

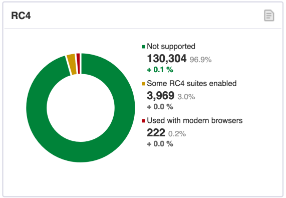
```
Update&Output($S = S[0], S[1], \ldots, S[255]$)
1 : $i \leftarrow 0; j \leftarrow 0$
2 : while more keystream bytes are required
3 :     $i \leftarrow (i + 1) \bmod 256$
4 :     $j \leftarrow (S[i] + j) \bmod 256$
5 :     Swap($S[i], S[j]$)
6 :     $t \leftarrow (S[i] + S[j]) \bmod 256$
7 :     return $S[t]$
```

Encryption: The keystream bytes are XORed with the plaintext bytes to produce ciphertext bytes.

Lifecycle of RC4

- The Fluhrer-Mantin-Shamir attack exploits known biases in the first few bytes of the keystream. The attack can be defeated by discarding the first few bytes (e.g., 768 bytes) of the keystream. [Details omitted]
- As of 2013, approximately 50% of all SSL traffic was secured using RC4.

- 2013-2019: Because of several new weaknesses discovered, RC4 is deprecated for applications such as SSL.
- September 2024: ∼3% of websites have RC4 enabled, ∼0.2% will actually use it.



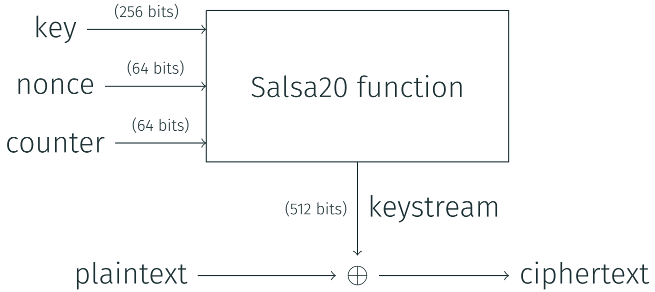
`https://www.ssllabs.com/ssl-pulse/`

Conclusions: Don't use RC4. Instead use AES-CTR (more on this later) or Salsa20/ChaCha20

Outline

Salsa20

- Salsa20: Modern stream cipher, designed by Daniel J. Bernstein in 2005.
- ChaCha20: Variant of Salsa20, 2008. Faster on some architectures.

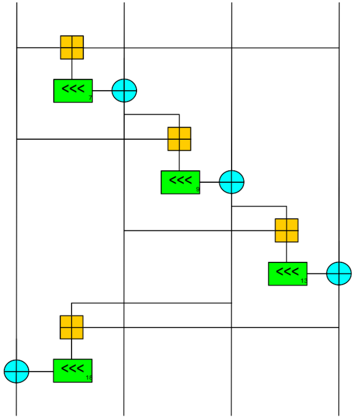


Salsa20 initial state

4×4 array of 32-bit words

constant_1	key	key	key
key	constant_2	nonce	nonce
counter	counter	constant_3	key
key	key	key	constant_4

Salsa20 quarter-round function (image from Wikipedia)



Legend:

- Yellow square plus: integer addition modulo 2^{32}
- Blue circle plus: exclusive OR
- Green left arrows: cyclic shift by 7 / 9 / 13 / 18 positions

Salsa20

Salsa20 Initialize

1. Load the key, nonce, and counter into the state

Salsa20 Update&Output

2. Make a copy of the initial state
3. Do the following 10 times:
    - 3.1 Apply the quarter-round function to each column of the state (4 columns)
    - 3.2 Apply the quarter-round function to each row of the state (4 rows)
4. XOR the current state with the copy of the initial state
5. Output that as the keystream

Salsa20 intuition

- Do some simple operations many times
- XORing in the initial state acts a little like the Vigenère cipher
- The cyclic shifts and applying the quarter-round function alternately to columns and rows are a little like the transposition cipher
- Together these ensures that the effect of key bits gets spread out across the entire state ("diffusion") in a way that's hard to reverse

Salsa20 characteristics

- Easy to implement in software and hardware
- Extremely fast in software: 4–14 cycles / per byte encrypted
- Successfully resisted cryptanalysis since 2005
- Variant ChaCha20 increasingly adopted in Internet protocols like TLS, QUIC, IPsec

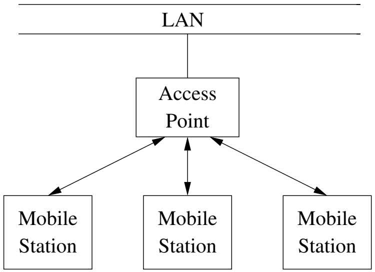
Outline

Wireless Security

- Wireless networks have become prevalent.
- Popular standards for wireless networks:
  - IEEE 802.11 (longer range, higher speeds, commonly used for wireless LANs).
  - Bluetooth (short range, low speed).
- New security concerns:
  - More attack opportunities (no need for physical access).
  - Attack from a distance ($>$ 1 km with good antennae).
  - No physical evidence of attack.

Case Study: IEEE 802.11 Security

- IEEE 802.11 standard for wireless LAN communications includes a protocol called Wired Equivalent Privacy (WEP).
- Ratified in September 1999.
- Multiple amendments: 802.11a (1999), 802.11b (1999), 802.11g (2003), 802.11i (2004), 802.11n (2009)....
- WEP's goal is (only) to protect link-level data during wireless transmission between mobile stations and access points.



Main Security Goals of WEP

1. Confidentiality: Prevent casual eavesdropping.
   - RC4 is used for encryption.
2. Data Integrity: Prevent tampering with transmitted messages.
   - An ‘integrity checksum’ is used.
3. Access Control: Protect access to a wireless network infrastructure.
   - Discard all packets that are not properly encrypted using WEP.

Description of WEP Protocol

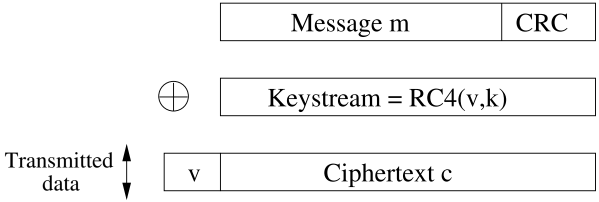
- Mobile station shares a secret key k with access point.
    - k is either 40 bits or 104 bits in length.
    - The standard does not specify how the key is to be distributed.
    - In practice, one shared key per LAN is common; this key is manually injected into each access point and mobile station; the key is not changed very frequently.
- Messages are divided into packets of some fixed length (e.g. 1500 bytes).
- WEP uses a per-packet 24-bit initialization vector (IV) v to process each packet. WEP does not specify how the IVs are managed. In practice:
    - A random IV is generated for each packet; or
    - The IV is set to 0 and incremented by 1 for each use.

Description of WEP Protocol (2)

To send a packet m, an entity does the following:

1. Select a 24-bit IV v.
2. Compute a 32-bit checksum: $S = \text{CRC}(m)$.
   - 802.11 specifies that a CRC-32 checksum be used. CRC-32 is linear. That is, for any two messages m_1 and m_2 of the same bitlength,
     $$\text{CRC}(m_1 \oplus m_2) = \text{CRC}(m_1) \oplus \text{CRC}(m_2).$$
     (The details of CRC-32 are not important to us)
3. Compute $c = (m \| S) \oplus \text{RC4}(v \| k)$.
   - $\|$ (and also a comma) denotes concatenation
   - $(v \| k)$ is the key used in the RC4 stream cipher.
4. Send (v, c) over the wireless channel.

Description of WEP Protocol (3)



The receiver of (v, c) does the following:

1. Compute $(m\|S) = c \oplus \text{RC4}(v\|k)$.
2. Compute $S' = \text{CRC}(m)$; reject the packet if $S' \neq S$.

Question: Are confidentiality, data integrity, and access control achieved?

Answer: NO! [Borisov, Goldberg & Wagner; 2001]

Outline

Problem 1: IV Collision

- Suppose that two packets (v, c) and (v, c') use the same IV v. Let m, m' be the corresponding plaintexts. Then $c \oplus c' = (m \| S) \oplus (m' \| S')$. Thus, the eavesdropper can compute $m \oplus m'$.
- If m is known, then m' is immediately available.
- If m is not known, then one may be able to use the expected distribution of m and m' to discover information about them. (Some contents of network traffic are predictable.)

Finding IV Collisions

- Since there are only 2^{24} choices for the IV, collisions are guaranteed after enough time — a few days on a busy network (5 Mbps).
- If IVs are randomly selected, then one can expect a collision after about 2^{12} packets.
    - Birthday paradox: Suppose that an urn contains n numbered balls. Suppose that balls are drawn from the urn, one at a time, with replacement. The expected number of draws before a ball is selected for a second time (called a collision) is approximately $\sqrt{\pi n/2} \approx \sqrt{n}$.
- Collisions are more likely if keys k are long-lived and the same key is used for multiple mobile stations in a network.
- Conclusion: WEP does not provide a high degree of confidentiality.

Problem 2: Checksum is Linear

- CRC-32 is used to check integrity. This is fine for random errors, but not for deliberate ones.
- CRC is linear: $\text{CRC}(a \oplus b) = \text{CRC}(a) \oplus \text{CRC}(b)$
- It is easy to make controlled changes to (encrypted) packets:
  - Suppose (v, c) is an encrypted packet.
  - Let $c = \text{RC4}(v\|k) \oplus (m\|S)$, where k, m, S are unknown.
  - Let $m' = m \oplus \Delta$, where Δ is a bit string.
    (The 1's in Δ correspond to the bits of m an attacker wishes to change.)
  - Let $c' = c \oplus (\Delta\|\text{CRC}(\Delta))$.
  - Then (v, c') is a valid encrypted packet for m'.
    [Exercise: Prove this]
- Conclusion: WEP does not provide data integrity.

Problem 3: Integrity Function is Unkeyed

- Suppose that an attacker learns the plaintext m corresponding to a single encrypted packet (v, c).
- Then, the attacker can compute the RC4 keystream $\text{RC4}(v\|k) = c \oplus (m\|\text{CRC}(m))$.
- Henceforth, the attacker can compute a valid encrypted packet for any plaintext m' of her choice: (v, c'), where $c' = \text{RC4}(v\|k) \oplus (m'\|\text{CRC}(m'))$.
- Conclusion: WEP does not provide access control.

Optional Reading: Sections 1, 2, 3, 4.1, 4.2, 6 of “Intercepting mobile communications: The insecurity of 802.11”, by N. Borisov, I. Goldberg and D. Wagner.

Outline

A More Devastating Attack

- Fluhrer, Mantin and Shamir, 2001.
- Assumptions:
  1. The same 104-bit key k is used for a long period of time. [Most products do this.]
  2. The IV is incremented for each packet, or a random IV is selected for each packet. [Most products do this.]
  3. The first plaintext byte of each packet (i.e. the first byte of each m) is known to the attacker.
     (Most wireless protocols prepend the plaintext with some header bytes which are non-secret.)
- Attack: A passive adversary who can collect about 5,000,000 encrypted packets can very easily recover k (and thus totally break the system). [Details not covered in this course.]

Implementing the Fluhrer-Mantin-Shamir Attack

- The attack can be easily mounted in practice:
  - Can buy a $100 wireless card and hack drivers to capture (encrypted) packets.
  - On a busy wireless network (5Mbps), 5 million packets can be captured in a few hours, and then k can be immediately computed.
- Implementation details: A. Stubbefield, J. Ionnidis, A. Rubin, "Using the Fluhrer, Mantin and Shamir attack to break WEP", AT&T Technical Report, August 2001.
- Script kiddies:
  - Aircrack-ng: `http://www.aircrack-ng.org/doku.php`
  - WEPCrack: `https://sourceforge.net/projects/wepcrack`
  - aircrack-ptw: Breaks WEP in under 60 seconds (only about 40,000 packets are needed).

Outline

Implications of WEP Insecurity

- WEP was blamed for the 2007 theft of 45 million credit-card numbers from T.J. Maxx. (T.J. Maxx is an American department store chain)
- A subsequent class action lawsuit settled for $40,900,000.

`https://www.informationweek.com/it-leadership/t-j-maxx-data-theft-likely-due-to-wireless-wardriving-`

Updates to Wi-Fi security – IEEE 802.11

WPA2 (Wi-Fi Protected Access)

- Implements the new IEEE 802.11i standard (2004).
- Uses the AES block cipher instead of RC4.
- Deployed ubiquitously in Wi-fi networks.
    - But deployments of WEP are still out there :-(
    - See `https://wigle.net/stats`
- KRACK attack in October 2017

WPA3

- Adopted in January 2018.
- Further improvements to WPA2.
- Dragonblood attack in April 2019, FragAttacks in May 2021