Applied Cryptography

CO 487

Alfred Menezes

LaTeXed by Sibelius Peng

Preface

Disclaimer Much of the information on this set of notes is transcribed directly/indirectly from the lectures of CO 487 during Winter 2021 as well as other related resources. I do not make any warranties about the completeness, reliability and accuracy of this set of notes. Use at your own risk.

For any questions, send me an email via https://notes.sibeliusp.com/contact/.

You can find my notes for other courses on https://notes.sibeliusp.com/.

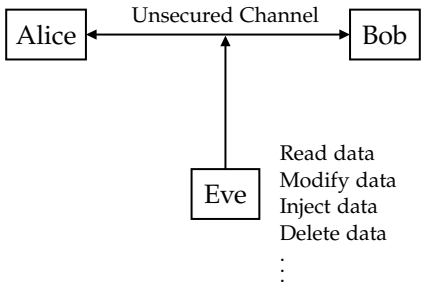
Sibelius Peng

Contents

1

Introduction

Cryptography is about securing communications in the presence of *malicious* adversaries.



Note that even if we call him "Eve", he can do more than eavesdropping... The adversary is malicious, powerful and unpredictable.

Fundamental Goals of Cryptography

1. Confidentiality: Keeping data secret from all but those authorized to see it.
2. Data integrity: Ensuring data has not been altered by unauthorized means.
3. Data origin authentication: Corroborating the source of data.
4. Non-repudiation: Preventing an entity from denying previous commitments or actions.

Some examples of unsecured channel:

- Secure Browsing: The Internet
- Online Shopping: The Internet
- Automatic Software Upgrades: The Internet
- Cell Phone Service: Wireless
- Wi-Fi: Wireless
- Bluetooth: Wireless
- Messaging: Wired/Wireless

Communicating Parties

Alice and Bob are two communicating devices.

Alice	Bob	Communication channel
person	person	telephone cable
person	person	cellular network
person	web site	internet
iPhone	wireless	router wireless
iPhone	headphones	wireless
iPhone	service provider	cellular network
your car's brakes	another car	wireless
smart card	bank machine	financial network
smart meter	energy provider	wireless
military commander	satellite	space

1.1 Secure Web Transactions

Transport Layer Security (TLS): The cryptographic protocol used by web browsers for secure web transactions for secure access to amazon, gmail, hotmail, facebook etc.

TLS is used to assure an individual user (called a client) of the authenticity of the web site (called the server) he or she is visiting, and to establish a secure communications channel for the remainder of the session.

Symmetric-key cryptography: The client and server a priori share some secret information k, called a key.

They can subsequently engage in secure communications by encrypting their messages with AES and authenticating the resulting ciphertexts with HMAC.

How do they establish the shared secret key k?

Public-key cryptography: Communicating parties a priori share some authenticated (but non-secret) information.

To establish a secret key, the client selects the secret session key k, and encrypts it with the server's RSA public key. Then only the server can decrypt the resulting ciphertext with its RSA private key to recover k.

How does the client obtain an authentic copy of the server's RSA public key?

Signature scheme: The server's RSA public key is signed by a Certifying Authority using the RSA signature scheme.

The client can verify the signature using the Certifying Authority's RSA public verification key which is embedded in its browser. In this way, the client obtains an authentic copy of the server's RSA public key.

1.2 The TLS Protocol

1. When a client first visits a secured web page, the server transmits its certificate to the client.
   - The certificate contains the server's identifying information (e.g., web site name and URL) and RSA public key, and the RSA signature of a certifying authority.
   - The certifying authority (e.g., Verisign) is trusted to carefully verify the server's identity before issuing the certificate.
2. Upon receipt of the certificate, the client verifies the signature using the certifying authority's public key, which is embedded in the browser. A successful verification confirms the authenticity of the server and of its RSA public key.
3. The client selects a random session key k, encrypts it with the server's RSA public key, and

transmits the resulting ciphertext to the server.

4. The server decrypts the ciphertext to obtain the session key, which is then used with symmetric-key schemes to encrypt (e.g. with AES) and authenticate (e.g. with HMAC) all sensitive data exchanged for the remainder of the session.
5. The establishment of a secure link is indicated by a closed padlock in the browser. Clicking on this icon reveals the server's certificate and information about the certifying authority.

TLS is one of the most successful security technologies ever deployed. But is TLS really secure?

There are many potential security vulnerabilities:

1. The crypto is weak (e.g., AES, HMAC, RSA).
2. Quantum attacks on the underlying cryptography.
3. Weak random number generation.
4. Issuance of fraudulent certificates
   - In 2001, Verisign erroneously issued two Class 3 code-signing certificates to a person masquerading as a Microsoft representative.
   - Mistake due to human error.
5. Software bugs (both inadvertent and malicious).
6. Phishing attacks.
7. TLS only protects data during transit. It does not protect your data when it is collected at the server.

   Many servers store large amounts of credit card data and other personal information.
8. The National Security Agency (NSA)

1.3 Cryptography in Context

Cybersecurity is comprised of the concepts, technical measures, and administrative measures used to protect networks, computers, programs and data from deliberate or inadvertent unauthorized access, disclosure, manipulation, loss or use. Also known as information security. Cybersecurity includes the study of computer security, network security and software security.

Note that Cryptography $\neq$ Cybersecurity.

- Cryptography provides some mathematical tools that can assist with the provision of cybersecurity services. It is a small, albeit an indispensable, part of a complete security solution.
- Security is a chain
  - Weak links become targets; one flaw is all it takes.
  - Cryptography is usually not the weakest link. However, when the crypto fails the damage can be catastrophic.

2

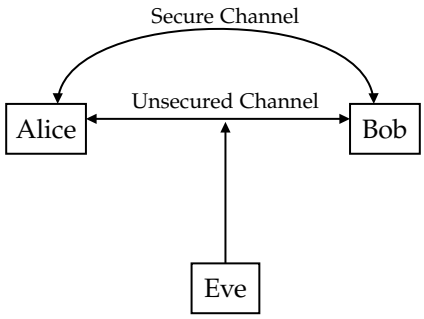
Symmetric-Key Cryptography

2.1 Basic concepts

symmetric-key encryption scheme

A **symmetric-key encryption scheme (SKES)** consists of:

- M - the plaintext space,
- C - the ciphertext space,
- K - the key space,
- a family of encryption functions: $E_k : M \to C$, $\forall k \in K$,
- a family of decryption functions: $D_k : M \to C$, $\forall k \in K$, such that $D_k(E_k(m)) = m$ for all $m \in M, k \in K$.



1. Alice and Bob agree on a secret key $k \in K$ by communicating over the secure channel.
2. Alice computes $c = E_k(m)$ and sends the ciphertext c to Bob over the unsecured channel.
3. Bob retrieves the plaintext by computing $m = D_k(c)$.

Some examples:

- WWII: Enigma Machine, Alan Turing,

  https://en.wikipedia.org/wiki/Cryptanalysis_of_the_Enigma
- WWII: Lorenz Machine, Bill Tutte, Colossus

  http://tinyurl.com/COBillTutte, http://billtuttememorial.org.uk/

2.1.1 The Simple Substitution Cipher

- M = all English msgs.
- C = all encrypted msgs.
- K = all permutations of the English alphabet.
- $E_k(m)$: Apply permutation k to m, one letter at a time.
- $D_k(c)$: Apply inverse permutations k^{-1} to c, one letter at a time.

Example:

$$k = \begin{array}{cccccccccccccccccccccccc} a & b & c & d & e & f & g & h & i & j & k & l & m & n & o & p & q & r & s & t & u & v & w & x & y & z \\ D & N & X & E & S & K & O & J & T & A & F & P & Y & I & Q & U & B & R & Z & G & V & C & H & M & W & L \end{array}$$

Encryption: $m = the\ big\ dog$, $c = E_k(the\ big\ dog) = GJS\ NTO\ EQO$.

Decryption: $c = GJS\ NTO\ EQO$, $m = E_k^{-1}(GJS\ NTO\ EQO) = the\ big\ dog$.

Is the simple substitution cipher a secure SKES? but wait, what does it mean for a SKES to be secure? We need a **security definition**.

1. What are the computational powers of the adversary?
2. How does the adversary interact with the two communicating parties?
3. What is the adversary's goal?

Security model: Defines the computational abilities of the adversary, and how she interacts with the communicating parties.

Basic assumption: The adversary knows everything about the SKES, except the particular key k chosen by Alice and Bob. (Avoid security by obscurity!!) Security should only depend on the secrecy of the secret keying material, not on the secrecy of the algorithms that describe the cryptographic scheme.

Let's now consider the computational power of the adversary.

- **Information-theoretic security**: Eve has infinite computational resources.
- **Complexity-theoretic security**: Eve is a 'polynomial-time Turing machine'.
- **Computational security**: Eve has 36,768 Intel E5-2683 V4 cores running at 2.1 GHz at her disposal. See: Graham in the basement of MC

  We say: Eve is "*computationally bounded*".

In this course, we will be concerned with computational security.

Next consider how the adversary can interact with Alice and Bob.

- Passive attacks:
  - **Ciphertext-only attack**: The adversary knows some ciphertext (that was generated by Alice or Bob).
  - **Known-plaintext attack**: The adversary also knows some plaintext and the corresponding ciphertext. (stronger because more info than before)
- Active attacks:
  - **Chosen-plaintext attack**: The adversary can also choose some plaintext and obtains the corresponding ciphertext. This is at least as strong as known-plaintext attack because the attacker gets to choose the plain text for which it is given the corresponding ciphertext.
- Other attacks (not considered in this course):

- Clandestine attacks: bribery, blackmail, etc.
- Side-channel attacks: monitor the encryption and decryption equipment (timing attacks, power analysis attacks, electromagnetic-radiation analysis, etc.)

Finally, we will consider the adversary's goal (from strongest to weakest).

1. Recover the secret key.
2. Systematically recover plaintext from ciphertext (without necessarily learning the secret key).
3. Learn some partial information about the plaintext from the ciphertext (other than its length).

If the adversary can achieve 1 or 2, the SKES is said to be **totally insecure** (or **totally broken**).

If the adversary cannot learn any partial information about the plaintext from the ciphertext (except possibly its length), the SKES is said to be **semantically secure**. So the ciphertext hides all the meaning about the corresponding plaintext.

security of SKES

A symmetric-key encryption scheme is said to be **secure** if it is semantically secure against chosen-plaintext attack by a computationally bounded adversary.

Note that the definitions captures the computational abilities of the adversary, namely some concrete upper bound on its computational resources. It captures how the adversary interacts with Alice and Bob, namely by mounting a chosen-plaintext attack. And finally, it captures the goal of the adversary, namely breaking semantic security. Note also that in the definition the attacker's capabilities are *as strong as possible*, namely a chosen-plaintext attack and its goal is *as weak as possible* namely, breaking semantic security

From the definition, we can also deduce what it means to break an encryption scheme.

1. The adversary is given a challenge ciphertext c (generated by Alice or Bob using their secret key k).
2. During its computation, the adversary can select plaintexts and obtains (from Alice or Bob) the corresponding ciphertexts.
3. After a feasible amount of computation, the adversary obtains some information about the plaintext m corresponding to the challenge ciphertext c (other than the length of m).

If the attacker can win this game specified in these three steps, then we'll say that the encryption scheme is in secure.

Desirable Properties of a SKES

1. Efficient algorithms should be known for computing E_k and D_k (i.e., for encryption and decryption).
2. The secret key should be small (but large enough to render exhaustive key search infeasible).
3. The scheme should be secure.
4. The scheme should be secure even against the designer of the system.

The last point is to ensure that the designer hasn't somehow inserted a backdoor into the encryption scheme whereby the adversary can learn a secret keying material by looking at ciphertext, while this method would be hard to find by other people. This fourth property can sometimes be very difficult to obtain in practice.

Now we can see whether the simple substitution cipher is secure: Totally insecure against a chosen-

plaintext attack. This is because the adversary can simply give the plaintext message that's comprised of the English alphabet from a to z to Alice for encryption. Alice returns to the adversary the ciphertext, which it turns out is the secret key itself. And so by simply giving Alice a short plaintext and getting back the corresponding ciphertext the adversary has learnt Alice's secret key.

What about security under a ciphertext-only attack? Is exhaustive key search possible?

- Given sufficient amounts of ciphertext c, decrypt c using each possible key until c decrypts to a plaintext message which "makes sense".
- In principle, 30 characters of ciphertext are sufficient on average to yield a unique plaintext that is a sensible English message. In practice, a few hundred characters are needed.

Exhaustive search:

- Number of keys to try is $26! \approx 2^{88}$
- If the adversary uses 10^6 computers, each capable of trying 10^9 keys per second, then exhaustive key search takes about 10^4 years.
- So, exhaustive key search is infeasible.

In this course,

- 2^{40} operations is considered very easy.
- 2^{56} operations is considered easy.
- 2^{64} operations is considered feasible.
- 2^{80} operations is considered barely feasible.
- 2^{128} operations is considered infeasible.

The Bitcoin network is presently performing hash operations at the rate of $2^{66.4}$ per second (or $2^{91.3}$ per year).

The Landauer limit from thermodynamics suggests that exhaustively trying 2^{128} symmetric keys would require $\gg$ 3000 gigawatts of power for one year (which is $\gg$ 100% of the world's energy production). http://en.wikipedia.org/wiki/Brute-force_attack

security level

A cryptographic scheme is said to have a **security level** of ℓ bits if the fasted known attack on the scheme takes approximately 2^{ℓ} operations.

As of the year 2021, a security level of 128 bits is desirable in practice.

Of course, simple frequency analysis of ciphertext letters can be used to recover the key. One would simply find the most frequently occurring ciphertext letter; this would most likely correspond to the English letter e which is the most frequently occurring letter in the English alphabet. Hence, the simple substitution cipher is totally insecure even against a ciphertext-only attack.

2.1.2 Polyalphabetic Ciphers

Basic idea: Use several permutations, so a plaintext letter is encrypted to one of several possible ciphertext letters.

Example: Vigenère cipher

Secret key is an English word having no repeated letters. E.g., $k = CRYPTO$.

Example of encryption:

	m	=	t	h	i	s		i	s		a		m	e	s	s	a	g	e
+	k	=	C	R	Y	P		T	O		C		R	Y	P	T	O	C	R
	c	=	V	Y	G	H		B	G		C		D	C	H	L	O	I	V

Here, $A = 0, B = 1, \ldots, Z = 25$; addition of letters is mod 26.

Decryption is subtraction modulo 26: $m = c - k$.

Frequency distribution of ciphertext letters is flatter (than for a simple substitution cipher).

The Vigenère cipher is totally insecure against a chosen-plaintext attack. Not unexpectedly, the Vigenère cipher is also totally insecure against a ciphertext-only attack.

2.2 The One-Time Pad

The one-time pad is a symmetric-key encryption scheme invented by Vernam in 1917 for the telegraph system. The key is a random[1] string of letters. Example of encryption:

	m	=	t	h	i	s		i	s		a		m	e	s	s	a	g	e
+	k	=	Z	F	K	W		O	G		P		S	M	F	J	D	L	G
	c	=	S	M	S	P		W	Y		P		F	Q	X	C	D	R	K

Note that the key is as long as the plaintext. One major disadvantage of the one-time pad is that the secret key is of the same length as the plaintext. This is especially inconvenient if Alice and Bob are exchanging long messages over a period of time, and so it might be tempting in practice to reuse the secret key. The key should *not* be re-used: If $c_1 = m_1 + k$ and $c_2 = m_2 + k$, then $c_1 - c_2 = m_1 - m_2$. Thus $c_1 - c_2$ depends only on the plaintext (and not on the key) and hence can leak information about the plaintext. In particular, if m_1 is known, then m_2 can be easily computed.

Convention: From now on, unless otherwise stated, messages and keys will be assumed to be binary/bit strings.

$\oplus$

$\oplus$ is bitwise exclusive-or (XOR) i.e., bitwise addition modulo 2. For example:

$$1011001010 \oplus 1001001001 = 0010000011.$$

Note that $x \oplus x = 0$ and $x \oplus y = y \oplus x$. Hence if $x = y \oplus z$, then $x \oplus y = z$.

So, for the one-time pad:

Encryption: $c = m \oplus k$.

Decryption: $m = c \oplus k$.

https://cryptosmith.com/2008/05/31/stream-reuse/ is an example which reuses the same key...

Security of One-Time Pad

Perfect secrecy: The one-time pad is semantically secure against ciphertext-only attack by an adversary with infinite computational resources. This can be proven formally using concepts from information theory [Shannon 1949].

The bad news: Shannon (1949) proved that if plaintexts are m-bit strings, then any symmetric-key encryption scheme with perfect secrecy must have $|K| \geq 2^m$. So, perfect secrecy (and the one-time

[1] independently and uniformly at random

pad) is fairly useless in practice. However, all is not lost. One can use one-time pads to inspire the construction of so-called stream ciphers.

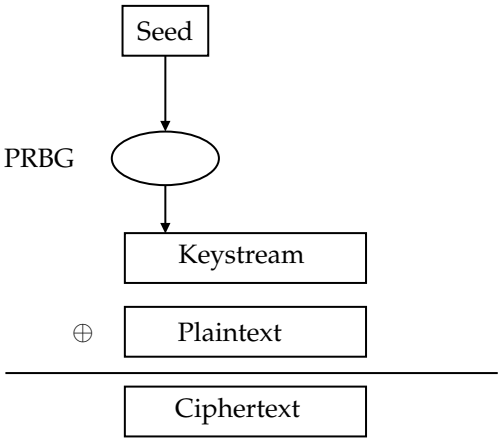
2.3 Stream Ciphers

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

A **pseudorandom bit generator** (PRBG) is a deterministic algorithm that takes as input a (relatively small random) seed, and outputs a longer "pseudorandom" sequence called the keystream.

Using a PRBG for encryption (a stream cipher):

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



The XOR portion of the stream cipher emulates a one-time pad. Note though that we no longer have perfect secrecy because the keystream is no longer purely random but pseudo-random. Thus security depends on the quality of the PRBG.

Security Requirements for the PRBG

The keystream should be "indistinguishable" from a random sequence (the **indistinguishability requirement**).

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 keystream. Thus, given portions of the keystream, it should be infeasible to learn any information about the rest of the keystream (the **unpredictability requirement**).

Aside: Don't use UNIX random number generators (`rand` and `srand`) for cryptography!

- $X_0 = \text{seed}$, $X_{i+1} = aX_i + b \mod n$, $i \geq 0$.
- a, b and n are fixed and public constants.

And so the UNIX routines `rand` and `srand` do not meet the unpredictability requirement.

One difficulty with using stream ciphers in practice is that we still have the requirement as we did with the one-time pad that keystream should never be reused.

2.4 The RC4 Stream Cipher

Designed by Ron Rivest in 1987.

Until recently, was widely used in commercial products including Adobe Acrobat, Windows password encryption, Oracle secure SQL, TLS, etc.

- *Pros*: Extremely simple; extremely fast; variable key length. No catastrophic weakness has been found since 1987.
- *Cons*: Design criteria are proprietary; not much public scrutiny until the year 2001.

In the past 10 years, many weaknesses have been found in RC4 and so its use has rapidly declined in practice.

RC4 has two components: (i) a *key scheduling algorithm*, and (ii) a *keystream generator*.

In the following, $K[i], \bar{K}[i]$ and $S[i]$ are 8-bit integers (bytes).

Algorithm 1: RC4 Key Scheduling Algorithm

Input: Secret key $K[0], K[1], \ldots, K[d-1]$. (Keylength is $8d$ bits.)
Output: 256-long array: $S[0], S[1], \ldots, S[255]$.

for $i = 0, \ldots, 255$ **do**
  $S[i] \leftarrow i$
  $\bar{K}[i] \leftarrow K[i \mod d]$
$j \leftarrow 0$
for $i = 0, \ldots, 255$ **do**
  $j \leftarrow (\bar{K}[i] + S[i] + j) \mod 256$
  Swap$(S[i], S[j])$

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

Algorithm 2: RC4 Keystream Generator

Input: 256-long byte array: $S[0], S[1], \ldots, S[255]$ produced by the RC4 Key Scheduling Algorithm
Output: Keystream.

$i \leftarrow 0; j \leftarrow 0$
while *keystream bytes are required* **do**
  $i \leftarrow (i+1) \mod 256$
  $j \leftarrow (S[i] + j) \mod 256$
  Swap$(S[i], S[j])$
  $t \leftarrow (S[i] + S[j]) \mod 256$
  Output$(S[t])$

ENCRYPTION: The keystream bytes are stored with the plaintext bytes to produce ciphertext bytes.

2.4.1 Wired Equivalent Privacy

Wireless Security

Wireless networks have become prevalent. Popular standards for wireless networks: IEEE 802.11 (longer range, higher speeds, commonly used for wireless LANs) and Bluetooth (short range, low speed).

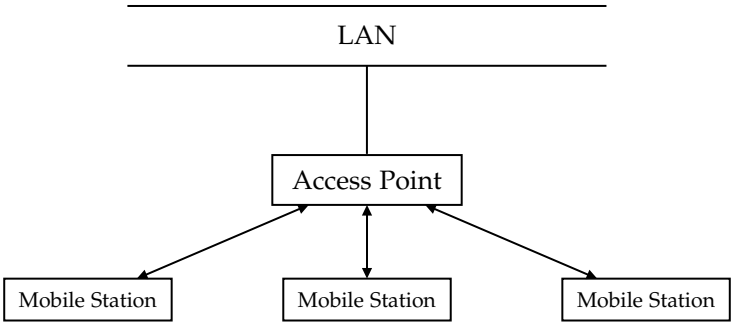
New security concerns:

- 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). This standard was ratified in September 1999. This was a very exciting time because

this was the first time that wi-fi was available to the consumer market. 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. A mobile station might be your cell phone or your laptop, and an access point is a wi-fi router somewhere in the building.



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 stations share 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.

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,

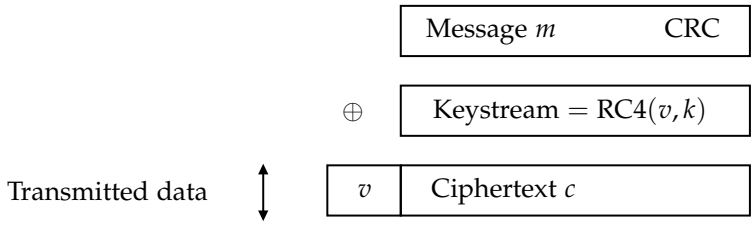
   $$\text{CRC}(m_1 \oplus m_2) = \text{CRC}(m_1) + \text{CRC}(m_2).$$

   (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 the RC4 stream cipher.

4. Send (v, c) over the wireless channel.



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 $S' \neq s$.

Are confidentiality, data integrity, and access control achieved? No (Borisov, Goldberg & Wagner; 2001),

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, them 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 is predictable.)

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.

It is easy to make controlled changes to (encrypted) packets: Suppose (v, c) is an encrypted packet. Let $c = \text{RC4}(c\|k) \oplus (m\|S)$, 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'.

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 $c' = \text{RC4}(v\|k) = c \oplus (m' \| \text{CRC}(m'))$.

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.

A More Devastating Attack

Fluhrer, Mantin & 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.]

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.

aircrack-ptw: Breaks WEP in under 60 seconds (only about 40,000 packets are needed).

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.

See: http://tinyurl.com/WEP-TJMaxx

IEEE 802.11 Update

http://en.wikipedia.org/wiki/Wi-Fi_Protected_Access

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 disclosed in October 2017.

WPA3

- Adopted in January 2018.
- Further improvements to WPA2.
- Dragonblood attack disclosed in April 2019.

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 TLS traffic was secured using RC4.

2013-2021: Because of several new weaknesses discovered, RC4 has been deprecated in applications such as TLS. (As of October 2019, $\approx$ 11.6% of websites have RC4 enabled, and only $\approx$ 1.1% actually use it.)

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

Index