CO 487 - Applied Cryptography

Cameron Roopnarine

Last updated: January 15, 2021

Contents

Contents 1

V1: Introduction 2
V1a: Course Preview 2
V1b: Course Outline 8

V2: Symmetric-Key Cryptography 11
V2a: Basic Concepts 11

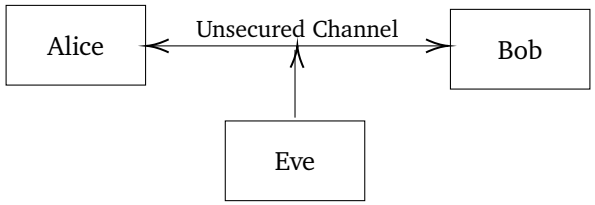
V1: Introduction

V1a: Course Preview

What is Cryptography?

DEFINITION 1.1.1: Cryptography

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



In the basic communications model we have two parties: Alice and Bob. They are exchanging messages over an *unsecured* channel such as the internet. By *unsecured* we mean that their communications might be under attack by an adversary who we call Eve, or eavesdropper.

Eve can certainly read data exchanged between Alice and Bob, but Eve can also modify data, inject data, delete data, and so on. Therefore, the term Eve can be a bit misleading since Eve can do *a lot* more than just eavesdrop. For the remainder of the course, when the term Eve is used please do not think of Eve as an eavesdropper, but rather as a very powerful, malicious, unpredictable, and evil adversary. What kind of security can Alice and Bob hope to achieve in the presence of such strong adversaries?

Fundamental Goals of Cryptography

1. *Confidentiality*: Keeping data secret from all but those authorized to see it.

   When Alice sends Bob a message, even though Eve can read all the transmitted data, Alice and Bob are the only parties who can read the actual underlying message. In practice, confidentiality is achieved through the use of encryption.

2. *Data integrity*: Ensuring data has not been altered by unauthorized means.

   When Bob receives a message from Alice, Bob has the assurance that the message wasn't modified during transmission.

3. *Data origin authentication*: Corroborating the source of data.

   When Bob receives a message supposedly sent by Alice, Bob can verify that the message indeed was sent by Alice.

4. *Non-repudiation*: Preventing an entity from denying previous commitments or actions.

Broadly speaking, this is preventing an entity from denying previous commitments or actions. In the physical world, non-repudiation is provided through the use of handwritten signatures. Cryptography though can be used to provide the same security as handwritten signatures, but in the digital world.

We'll see in the remainder of the course that cryptography can be used to provide confidentiality, data integrity, data origin authentication, and non-repudiation.

It should be evident that these four security services provided by cryptography are very important in many applications where sensitive data might be communicated or stored. Let's look at some examples.

EXAMPLE 1.1.2: Email

Firstly, when Alice connects to the website, she needs the assurance that she really is visiting g-mail's website, not the website of an impersonator. Next, when she enters her email address and password, these are transmitted over the internet and so need to be secured through the use of encryption. Lastly, when Alice uploads and downloads emails, these are also transmitted over the internet and so have to be encrypted and authenticated.

EXAMPLE 1.1.3: Online Shopping

Cryptography helps engender trust in e-commerce applications for both consumers and vendors.
This is accomplished by encrypting and authenticating all sensitive financial and personal data that might be transmitted between Alice and Bob.

EXAMPLE 1.1.4: Automatic Software Upgrades

Cryptography helps in guaranteeing the safety of automatic software upgrades.
Imagine Microsoft broadcasting upgrades of its operating system to hundreds of millions of devices around the world. When such a device receives an upgrade, it needs the assurance that the software upgrade indeed is from Microsoft and hasn't been modified during transmission. In practice, these assurances are provided through the use of digital signatures.
Note: Confidentiality is not required in this scenario because the software upgrades themselves are not considered to be private.

EXAMPLE 1.1.5: Cell Phone Service

When you make a call on your cell phone the connection is established to a nearby cell phone tower. The cell phone tower needs the assurance that you have the authorization to use the cell phone service. Furthermore, your voice data needs to be encrypted and authenticated as it is transmitted over the air to the cell phone tower.

EXAMPLE 1.1.6: WiFi

When you connect to a WiFi network, your cell phone has to authenticate itself to the WiFi router to prove that it has the authorization to use the WiFi service. Furthermore, all internet communications between your cell phone and the router have to be encrypted and authenticated.

EXAMPLE 1.1.7: Bluetooth

When two devices such as your cell phone and your headset establish a Bluetooth connection this means that all data transmitted between the two devices is encrypted and authenticated using cryptographic mechanisms.

EXAMPLE 1.1.8: Messaging

Cryptography is used to secure instant messaging, most famously by WhatsApp which provides very strong *end-to-end encryption*. End-to-end encryption is when *only* the two users who are exchanging messages can read them. In particular, WhatsApp itself cannot read the exchange messages.

Hopefully, these examples convince the reader that cryptography is extremely useful and important in securing all kinds of applications where sensitive data might be communicated or stored.

One shouldn't think of Alice and Bob as human beings but rather as two communicating devices.

Table 1.1: Communicating Parties

Alice	*Bob*	*Communications channel*
person	person	telephone cable
person	person	cellular network
person	website	internet
iPhone	wireless router	wireless
iPhone	headphones	wireless
car's brakes	another car	wireless
smart card	bank machine	financial network
smart meter	energy provider	wireless
military commander	satellite	space

Secure Web Transactions

EXAMPLE 1.1.9: Secure Web Transactions

Cryptography is used to secure web transactions via *Transport Layer Security*.

DEFINITION 1.1.10: Transport Layer Security (TLS)

The cryptographic protocol used by web browsers for secure web transactions for secure access to amazon, g-mail, Facebook, etc.

TLS is used to assure an individual user (*client*) in this case Alice, of the authenticity of the website (*server*), in this case g-mail, he or she is visiting, and to establish a *secure communications channel* for the remainder of the session.
All data communicated between Alice and the website need to be encrypted and authenticated which is accomplished using *symmetric-key cryptography*.

DEFINITION 1.1.11: Symmetric-key Cryptography

The *client* and *server*, a priori, share some *secret* information k, called a *key*.

Once Alice and Bob have such a secret key, they can subsequently engage in secure communications by encrypting their messages with Advanced Encryption Standard (AES) and authenticating the resulting ciphertexts with HMAC. We'll study AES and HMAC later on in the course.

Question: Alice and Bob establish the shared secret key k in the first place?

Alice cannot simply select k and transmit it over the internet to Bob because then the eavesdropper would learn k.

This is where *public-key cryptography* is very useful.

DEFINITION 1.1.12: Public-key Cryptography

Communicating parties a priori, share some *authenticated* (but non-secret) information.

Now, to establish a secret key, the client Alice selects the secret session key k, and encrypts it with the servers *RSA public key*, and sends the resulting ciphertext over the internet to Bob. Only Bob the server can decrypt the resulting ciphertext, because decryption requires the use of the corresponding RSA private key which only Bob knows. And so Bob decrypts the ciphertext to recover k, and k is known only to Alice and Bob.

Question: How does Alice obtain an authentic copy of the server's RSA public key?

Bob cannot just send the public key to Alice over the internet because Eve could intercept this public key and replace it with her own public key.

This problem is solved using signature schemes.

DEFINITION 1.1.13: Signature Scheme

The server's RSA public key is *signed* by a *Certifying Authority* (CA) using the **RSA signature scheme**.

The client can verify the signature using the CA's RSA public verification key which is embedded in Alice's browser.

In this way, the client obtains an authentic copy of the server's RSA public key.

The TLS protocol

1. When a client first visits a secured website, the server transmits its *certificate* to the client.
   - The certificate contains the server's identifying information (e.g. website name and URL) and RSA a 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?

TLS: Potential Vulnerabilities

1. The cryptography is weak (e.g. AES, HMAC, RSA).

   Recall from MATH 135 that the security of RSA is based on the presumed difficulty of factoring large numbers. No one knows how to factor large numbers efficiently today, however there is a possibility that such an algorithm might be discovered tomorrow.

2. *Quantum attacks* on the underlying crpytography.

We do know how to factor large numbers very quickly on quantum computers,although it isn't known as yet whether one can actually build large-scale quantum computers.

3. Weak random number generation.

   In TLS, browsers select random session keys. In practice, selecting random numbers is a very difficult problem in software applications.

   Many deployments of TLS have been found to be insecure in the past 20 years due to poor random number generation.

4. Issuance of fraudulent certificates.

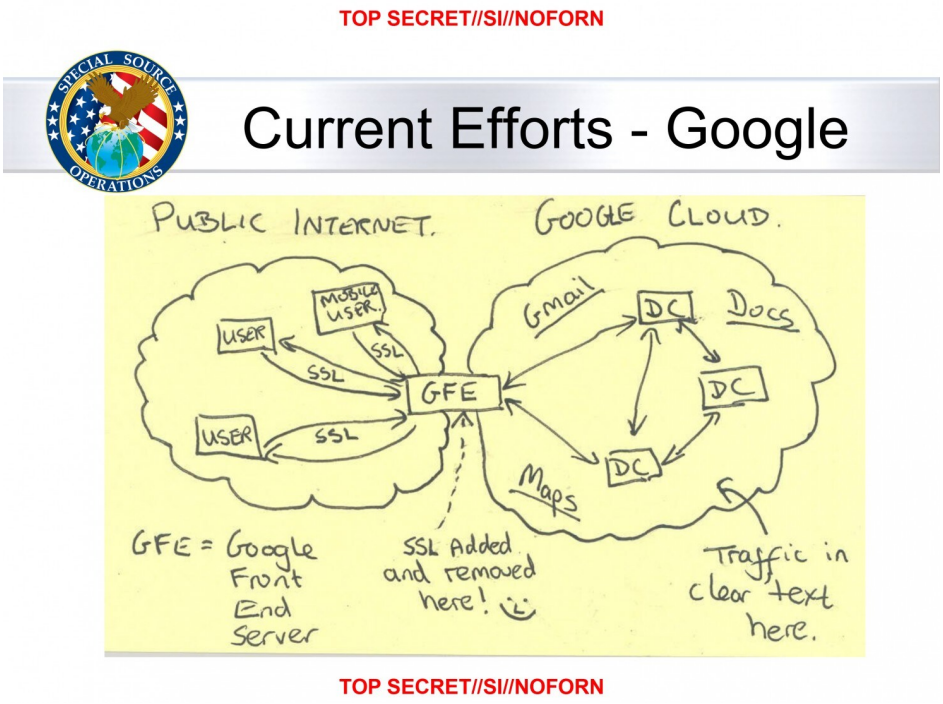
   - In 2001, Verisign erroneously issued two Class 3 code-signing certificates to a person masquerading as a Microsoft representative.

     In principle, this person could have generated malware, signed it with its RSA private key, broadcast this malware to computers around the world, and the computers would have accepted the malware as originating from Microsoft. This problem was caused by *human error*, not due to any deficiency in the underlying cryptography.

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).



We give an example from the Snowden revelations. In 2013, Edward Snowden, an NSA employee, leaked a large number of highly classified NSA documents to the press. One such document was headed "Current Efforts Google."

On the left, the slide showed the public internet. So it showed users connecting using TLS, also known as SSL, to Google's front-end servers. So all data exchange between users and Google servers are protected

using TLS.

However, the slide also had the statement "SSL added and removed here" with a smiley face, and the note that "traffic in cleartext here."

So Google's front-end servers would then transmit the decrypted data to its data centers around the world. These transmissions took place in the so-called Google cloud, where the data centers were connected by private links. In particular, the data transmitted between these links were not encrypted. As it turns out, NSA had access to one of these links between two Google data centers and NSA was capturing and storing gigantic amounts of user data transmitted between these two data centers.

Therefore, even though TLS itself was secure, what was not secure was the way in which data was stored and transmitted in Google's back end.

Cryptography in Context

These examples of potential vulnerabilities in TLS tell us that cryptography must be studied in a context, and that broader context is information security, now known as Cybersecurity. Cybersecurity is a very broad field and it's difficult to give it a precise definition. Nonetheless, here's one attempt.

DEFINITION 1.1.14: Cybersecrity

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*.

Table 1.2: Cybersecrity Includes the Study of:

Computer security	*Network security*	*Software security*
Security models and polices	Internet Protocols and their security	Detecting and preventing buffer overflows
Secure operating systems	Viruses and worms	Programming languages and compilers
Virus protection	Denial-of-service (DoS) attacks	Specifying and enforcing security policies
Auditing mechanisms	Firewalls	Digital rights management
Risk analysis	Intrusion detection systems	Code obfuscation
Risk management	Wireless communications	Software tamper resistance
		Trusted computing

Cryptography $\neq$ Cybersecrity

- 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 cryptography fails the damage can be catastrophic.
  - This course will focus on cryptography.

Syllabus

Cryptography primitives

- Symmetric-key encryption
- Hash functions
- Message authentication
- Authenticated encryption
- Public-key encryption
- Signature schemes
- RSA
- Elliptic curve cryptography
- Key establishment
- Post-quantum cryptography

Cryptography deployments

- IEEE 802.11 WEP/WPA2
- Google's KMS
- GSM security
- QQ browser
- FIDO U2F
- Bluetooth security
- Key management (PKIs)
- Web security protocols (TLS)
- Signal protocol (WhatsApp)
- Cryptocurrencies (Bitcoin)

V1b: Course Outline

About the Course

- Coverage will favour breadth at the expense of depth
  - For depth, try the optional readings
  - See also: *CO 485* (Mathematics of Public-Key Crypto)
  - See also: *CS 458* (Computer Security and Privacy)
- This course is not a traditional textbook course!
  - *Watching the video lectures* is strongly recommended
  - There are no good sources of "practice questions"
  - *Your job*: Identify and understand the important (technical and non-technical) *concepts* presented in the lectures
- Optional text: Understanding Cryptography
  - Available for free download from the UW library
  - Optional readings are posted on the course website

Learning Outcomes

1. Understand the fundamental cryptographic tools of symmetric-key encryption, message authentication, authenticated encryption, hash functions, public-key encryption, and signatures;
2. Appreciate the challenges with assessing the security of these tools;
3. Gain exposure to how these cryptographic tools are used to secure large-scale applications;
4. Understand why key management is an essential process that underpins the security of many applications.

Course Administration

- Website: http://learn.uwaterloo.ca
  - Course outline and *weekly schedule*.
  - Videos, slides, assignments & solutions, handouts.
  - Optional readings.

- Assistance:
  - Instructor and TA online office hours (see LEARN).
  - Piazza (see the Course Outline for the password).
  - Participation is *strongly encouraged*, but optional.
- Communications: All important course announcements will be sent to you by *email*. (Apologies in advance for the spam.)
- Assignments (50%): submitted via Crowdmark.
  - Due dates: *Jan 29*, *Feb 12*, *Mar 5*, *Mar 26*, *Apr 9*.
- Quizzes (15 / 7.5 / 0%): *Feb 26* and *Mar 19* (on LEARN).
- Final Assessment: (35 /42.5 / 50%).

Academic Integrity

- Assignments:
  - No collaboration permitted on one or two questions.
  - Collaboration permitted (and *encouraged*) on the other questions (but solutions must be written up independently).
- Quizzes:
  - *Open book*: slides, your notes, assignments, solutions (ONLY).
- Final Assessment:
  - *Open book*: slides, your notes, assignments, solutions (ONLY).
- General: no cheating, Course Hero, Chegg, online discussion groups, solutions from previous course offerings, etc.
- Policy 71 *will be enforced*.

Should You Take This Course?

Course philosophy: CO 487 is an *elective course* for everyone.

As such, it is intended to be:

- *interesting*,
- *fun*,
- *relevant*, and
- *not-too-difficult*.

Amount of Material

Course content:

- 65%: Material you need to know for quizzes and the final exam.
- 25%: Show-and-tell, including “Crypto Cafe.”
- 10%: Cryptography gossip.

Guidance for the quizzes and the final exam:

There will not be any "unreasonable" expectations of what you need to memorize for the quizzes and the final assessment. For example, you do *not* need to memorize the descriptions of RC4, the WEP security protocol, ChaCha20, AES, AES-GCM, SHA-256, RSA-OAEP, the elliptic curve addition formulae, ECDSA, the Bluetooth security protocol, the Signal protocol, etc. If asked any questions on such topics, then all the relevant background information will be provided. You also don't need to memorize any *trivia*, e.g. historical notes, dates, spelling out of acronyms, etc.

V2: Symmetric-Key Cryptography

V2a: Basic Concepts

DEFINITION 1.1.15: Symmetric-key Encryption Scheme (SKES)

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: $C \to M$, $\forall k \in K$, such that $D_k(E_k(m)) = m$ for all $m \in M$, $k \in K$.

Using a SKES to Achieve Confidentiality

TODO Image

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 $m = D_k(c)$.

The Simple Substitution Cipher

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

EXAMPLE 1.1.16: The Simple Substitution Cipher

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

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.

Question: Is the simple substitution cipher a secure SKES?

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!!)

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 2.1GHz 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.

Adversary's Interaction

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.

Active attacks:

- Chosen-plaintext attack: The adversary can also choose some plaintext and obtains 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.)

Adversary's Goal

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*.

Definition of a Secure SKES

DEFINITION 1.1.17: Secure 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.

To *break* a symmetric-key encryption scheme, the adversary has to accomplish the following:

1. 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).

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.

Security of the Simple Substitution Cipher

Totally insecure against a chosen-plaintext attack. [Why?]

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 $26! \approx 4 \times 10^{26} \approx 2^{88}$.
- 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.

Work Factor

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).

Brute-force attack

Security Level

DEFINITION 1.1.18: Security level

A cryptographic scheme is said to have a **security level** of ℓ bits if the fastest 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. Hence, the simple substitution cipher is totally insecure even against a ciphertext-only attack.

[TODO Image, 52]

Polyalphabetic Ciphers

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

EXAMPLE 1.1.19: Vigenere cipher

- Secret key is an English word having no repeated letters s e.g. $k =$ CRYPTO.
- Example of encryption: [TODO]
- 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).

Security of the Vigenere Cipher

- The Vigenère cipher is totally insecure against a chosen-plaintext attack. [Why?]
- What about security under a ciphertext-only attack?
- Not unexpectedly, the Vigenère cipher is also totally insecure against a ciphertext-only attack.

The One-Time Pad

Invented by Vernam in 1917 for the telegraph system.

The key is a random string of letters.

EXAMPLE 1.1.20: Encryption

REMARK 1.1.21

The key as as long as the plaintext.

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.

Binary Messages

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

REMARK 1.1.22

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

EXAMPLE 1.1.23: Bitwise Addition Modulo 2

$$1011001010 \oplus 1001001001 = 0010000011$$

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: $m = c \oplus k$.

EXAMPLE 1.1.24: Reusing a Key in the One-Time Pad

TODO

Security of the 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 pad) is fairly useless in practice.
- All is not lost: stream ciphers

Stream Ciphers

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

DEFINITION 1.1.25: Pseudorandom bit generator (PRBG), Seed, Keystream

A **pseudorandom bit generator** (PRBG) is a deterministic algorithm that takes as input a (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.

[TODO Image] No more perfect secrecy — 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!
  - $X_0 =$ seed, $X_{i+1} = aX_i + b \bmod n$, $i \geq 0$.
  - a, b and n are fixed and public.

The RC4 Steam 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.
- Cons: Design criteria are proprietary; not much public scrutiny until the year 2001.
- RC4 has two components: (i) a key scheduling algorithm, and (ii) a keystream generator.

RC4 Key Scheduling Algorithm

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

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

RC4 Keystream Generator

[TODO Alg]

ENCRYPTION: The keystream bytes are XOR'd with the plaintext bytes to produce ciphertext bytes.

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), etc.
- WEP's goal is (only) to protect link-level data during wireless transmission between mobile stations and access points. TODO

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 bit-length,
   $$\text{CRC}\,(m_1 \oplus m_2) = \text{CRC}\,(m_1) \oplus \text{CRC}\,(m_2)$$
   (The details of CRC-32 are not important to us)
3. Compute $c = (m \| S) \oplus \text{RC}\,4(v \| k)$.
   - $\|$ (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.

TODO image

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

1. Compute $(m \| S) = c \oplus \text{RC}\,4(v \| k)$.
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)

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 is 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.
- It is easy to make controlled changes to (encrypted) packets:
- Suppose (v, c) is an encrypted packet.
- Let $c = \text{RC}\,4(v \| k) \oplus (m \| S)$, 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 $\text{RC}\,4(v \| k) = c \oplus (m \| \text{CRC}(m))$.

- Henceforth, the attacker can compute a valid encrypted packet for any plaintext m' of her choice: (v, c'), where $c' = \text{RC}\,4(v \| k) \oplus (m' \| \text{CRC}\,(m'))$
- Conclusion: WEP does not provide access control.

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

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 $\$40,900,000$
- See URL

IEEE 802.11 Update

WiFi-Protected Access

WPA2 (WiFi Protected Access)

- Implements the new IEEE 802.11i standard (2004).
- Uses the AES block cipher instead of RC4.
- Deployed ubiquitously in WiFi networks.

- But deployments of WEP are still out there.
- URL
- KRACK attack disclosed in October 2017.

WPA3

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

RC4 Update

- 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).