CO 487 Winter 2024:
Lecture Notes

Lecture notes taken, unless otherwise specified, by myself during the Winter 2024 offering of CO 487, taught by Alfred Menezes.

Lectures

Chapter 1

Introduction

Lecture 1
Jan 8

Cryptography is securing communications in the presence of malicious adversaries. To simplify, consider Alice and Bob communicating with the eavesdropper Eve. Communications should be:

- Confidential: Only authorized people can read it
- Integral: Ensured that it is unmodified
- Origin authenticated: Ensured that the source is in fact Alice
- Non-repudiated: Unable to gaslight the message existing

Examples: TLS for intenet browsing, GSM for cell phone communications, Bluetooth for other wireless devices.

Overview: Transport Layer Security The protocol used by browsers to visit websites. TLS assures an individual user (a client) of the authenticity of the website (a server) and to establish a secure communications session.

TLS uses symmetric-key cryptography. Both the client and server have a shared secret k called a key. They can then use AES for encryption and HMAC for authentication.

To establish the shared secret, use public-key cryptography. Alice can encrypt the session key k can be encrypted with Bob's RSA public key. Then, Bob can decrypt it with his private key.

To ensure Alice is getting an authentic copy of Bob's public key, a certification authority (CA) signs it using the CA's private key. The CA public key comes with Alice's device preinstalled.

Potential vulnerabilities when using TLS:

- Weak cryptography scheme or vulnerable to quantum computing
- Weak random number generation for the session key
- Fraudulent certificates
- Implementation bugs

- Phishing attacks
- Transmission is secured, but the endpoints are not

These are mostly the purview of cybersecurity, of which cryptography is a part. Cryptography is not typically the weakest link in the cybersecurity chain.

Chapter 2

Symmetric Key Encryption

2.1 Basic concepts

Lecture 2
Jan 10

Definition 2.1.1 (symmetric-key encryption scheme)

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

- plaintext space M,
- ciphertext space C,
- key space K,
- family of encryption functions $E_k : M \to C$ for all keys $k \in K$, and
- family of decryption functions $D_k : C \to M$ for all keys $k \in K$

such that $D_k(E_k(m)) = m$ for all m and k.

For Alice to send a message to Bob:

1. Alice and Bob agree on a secret key k *somehow* (assume a secured channel)
2. Alice computes $c = E_k(m)$ and sends c to Bob
3. Bob recovers the plaintext by computing $m = D_k(c)$

Examples include the Enigma and Lorenz machines.

Cryptoscheme 1 (simple substitution cipher)

Let:

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

We want a system to have:

1. Efficient algorithms should be known for computing (encryption and decryption)

2. Small keys but large enough to render exhaustive key search infeasible
3. Security
4. Security against its designer

To determine how secure the protocol is, we have to define security.

Definition 2.1.2 (security model)

Some parameters which define the strength of the adversary, specific interaction with the "secure" channel, and the goal of the adversary.

Some options for strength:

- Information-theoretic security: Eve has infinite resources.
- Complexity-theoretic security: Eve is a polynomimal-time Turing machine.
- Computational-theoretic security: Eve has a specific amount of computing power. In this course, Eve is computationally bounded by 6,768 Intel E5-2683 V4 cores running at 2.1 GHz at her disposal.

For the interaction:

- Ciphertext-only attack: Eve only knows the ciphertext.
- Known-plaintext attack: Eve knows some plaintext and the corresponding ciphertext.
- Chosen-plaintext attack: Eve picks some plaintext and knows the corresponding ciphertext.
- Clanedestine attack: Eve resorts to bribery, blackmail, etc.
- Side-channel attack: Eve has physical access to hardware and has some monitoring data.

And for the goal:

- Recovering the secret key k
- Systematically decrypt arbitrary ciphertexts without knowing k (total security)
- Learn partial information about the plaintext (other than the length) (semantic security)

Definition 2.1.3 (security)

An SKES is secure if it is semantically secure against a chosen-plaintext attack by a computationally bounded adversary.

Equivalently, an SKES is broken if:

1. Given a challenge ciphertext c for m generated by Alice,
2. …and access to an encryption oracle for Alice,
3. …Eve can obtain some information about m other than its length,
4. …using only a feasible amount of computation.

Note: this is IND–CPA from CO 485.

Example 2.1.4. Is the simple substitution cipher secure? What about under a ciphertext-only attack?

Solution. Under CPA, encrypt the entire alphabet. Then, the entire key k is recovered.

With a ciphertext-only attack, an exhaustive key search would take $26! \approx 2^{88}$ attempts. This would take over 1,000 years, which is pretty infeasible, so it is secure. □

Can we quantify how feasible something is?

Definition 2.1.5 (security level)

A scheme has a security level of ℓ bits if the fastest known attack on the scheme takes approximately 2^ℓ operations.

Convention. In this course:

- 40 bits is very easy to break
- 56 bits is easy to break
- 64 bits is feasible to break
- 80 bits is barely feasible to break
- 128 bits is infeasible to break

Lecture 3
Jan 12

The simple substitution cipher can be attacked by frequency analysis, since, for example, if "e" is the most common English letter, we check the ciphertext for the most common letter and identify it with "e".

Cryptoscheme 2 (Vigenère cipher)

Let the key K be an English word with no repeated letters, e.g., $K = \text{CRYPTO}$.

To encrypt, add letter-wise the key modulo 26, where k is K repeated until it matches the length of the message:

	$m =$	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

To decrypt, just take $c - k$.

This solves our frequency analysis problem. However, the Vigenere cipher is still totally insecure.

Exercise 2.1.6. Show that the Vigenere cipher is totally insecure under a chosen-plaintext attack and a ciphertext-only attack.

Cryptoscheme 3 (one-time pad)

The key is a random string of letters with the same length as the message.

Repeat the process for Vigenere. To encode, add each letter. To decode, subtract each letter.

Example 2.1.7. We can encrypt as follows:

	$m =$	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

This is semantically secure as long as the key is never reused. Formally, there exist keys that can decrypt the ciphertext into *anything*, so there is no way for an attacker to know the plaintext. If it is reused, i.e., if $c_1 = m_1 + k$ and $c_2 = m_2 + k$, then $c_1 - c_2 = (m_1 + k) - (m_2 + k) = m_1 - m_2$. Since this is a function only of messages, it can leak frequency information etc.

Also, since the key is never reused, this is secure against a chosen plaintext attack, since one would only recover the already used key.

Convention. From now on, messages and keys are assumed to be binary strings.

Definition 2.1.8 (bitwise exclusive or)
For two bitstrings $x, y \in \{0,1\}^n \cong \mathbb{Z}/2\mathbb{Z}^n$, the bitwise XOR $x \oplus y$ is just addition mod 2.

Unfortunately, due to Shannon, we have this theorem:

Theorem 2.1.9
A perfectly secure symmetric-key scheme must have at least as many keys as there are messages.

2.2 Stream ciphers

Instead of using a random key in the OTP, use a pseudorandom key.

Definition 2.2.1 (pseudorandomness)
A pseudorandom bit generator (PBRG) is a deterministic algorithm that takes as input a seed and outputs a pseudorandom sequence called the keystream.

Then, we can construct a stream cipher by defining the key as the seed and the ciphertext as the keystream XOR'd with the plaintext. To decrypt, use the seed to generate the same keystream and XOR with the ciphertext.

For a stream cipher to be secure, we need:

- Indistinguishability: the keystream is indistinguishable from a truly random sequence; and
- Unpredictability: given a partial keystream, it is infeasible to learn any information from the remainder of the keystream.

Remark 2.2.2. Do not use built-in UNIX `rand` or `srand` for cryptography!

Now, we introduce ChaCha20, a stream cipher actually uesd in the real world. The algorithm works entirely on words (32-bit numbers). It has no known flaws (other than people bungling the implementation).

Cryptoscheme 4 (ChaCha20)

First, define a helper function $QR(a, b, c, d)$ on 32-bit words:

1. $a \leftarrow a \boxplus b$, $d \leftarrow d \oplus a$, $d \leftarrow d \lll 16$
2. $c \leftarrow c \boxplus d$, $b \leftarrow b \oplus c$, $b \leftarrow b \lll 12$
3. $a \leftarrow a \boxplus b$, $d \leftarrow d \oplus a$, $d \leftarrow d \lll 8$
4. $c \leftarrow c \boxplus d$, $b \leftarrow b \oplus c$, $b \leftarrow b \lll 7$

where $\oplus$ is bitwise XOR, $\boxplus$ is addition mod 2^{32}, and $\lll$ is left bit-rotation.

Given a 256-bit key $k = (k_1, \dots, k_8)$, a selected 96-bit nonce $n = (n_1, n_2, n_3)$, a 128-bit given constant $f = (f_1, \dots, f_4)$, and 32-bit counter $c \leftarrow 0$, construct an initial state:

$$S := \begin{bmatrix} f_1 & f_2 & f_3 & f_4 \\ k_1 & k_2 & k_3 & k_4 \\ k_5 & k_6 & k_7 & k_8 \\ c & n_1 & n_2 & n_3 \end{bmatrix} = \begin{bmatrix} S_1 & S_2 & S_3 & S_4 \\ S_5 & S_6 & S_7 & S_8 \\ S_9 & S_{10} & S_{11} & S_{12} \\ S_{13} & S_{14} & S_{15} & S_{16} \end{bmatrix}$$

Keep a copy $S' \leftarrow S$, then apply:

$$QR(S_1, S_5, S_9, S_{13}), \quad QR(S_2, S_6, S_{10}, S_{14}), \quad QR(S_3, S_7, S_{11}, S_{15}), \quad QR(S_4, S_8, S_{12}, S_{16})$$
$$QR(S_1, S_6, S_{11}, S_{16}), \quad QR(S_2, S_7, S_{12}, S_{13}), \quad QR(S_3, S_8, S_9, S_{14}), \quad QR(S_4, S_5, S_{10}, S_{15})$$

ten times (for 80 total calls to QR) and output $S \oplus S'$. This gives us 64 keystream bytes.

Increment $c \leftarrow c + 1$ and repeat as necessary to generate more keystream bytes.

To encrypt, XOR the keystream with the plaintext, then append the nonce.

To decrypt, pop off the nonce, then XOR the keystream with the ciphertext.

Lecture 4
Jan 15

One must be careful never to reuse nonces, since this results in the same keystream, leading to recoverable messages. In practice, this is hard (e.g., two devices with the same key).

Miscellaneous remarks:

- Why is ChaCha20 so good? The QR function is very fast at the hardware level and there is wide adoption/standardization by experts.
- Why 10 rounds? If you do 1 or 2 rounds, there is a trivial attack. The latest theoretical attacks can attack 7 rounds (currently infeasible, but still better than exhaustive key search). So 8 rounds is secure and we do 10 to be safe.
- Is this secure forever (i.e., can we always just increase rounds)? No. Nothing in this course is. Someone could find a super crazy PMATH theorem that shows predictability of the QR scramble.

2.3 Block ciphers

Definition 2.3.1 (block cipher)
Like a stream cipher, but instead of processing one character at a time, we break up the plaintext into blocks of equal length and encrypt block-wise.

Example 2.3.2. The Data Encryption Standard (DES) is a standard 56-bit key and 64-bit blocks.

Aside: History and the NSA doing ratfuckery In 1972, the National Institute of Standards and Technology (NIST)[1] puts out an RfP for encryption algorithms.

IBM developed and proposed 64-bit DES, but then the NSA reduced it in 1975 to 56-bit so they can do some spying. This made DES feasible to break by nation-states but not smaller organizations.

The National Security Agency (NSA) is the US' signals intelligence (SIGINT; hacking foreign intelligence) and information insurance (IA; defending domestic intelligence) agency. They have a history of regulating how strong cryptoraphic products can be by banning the export of strong cryptography.

Canada has an NSA equivalent: the Communications Security Establishment (CSE). Along with the Kiwi CCSA, British GCHQ, and Australian ASD, these are the Five Eyes who spy on just about everyone.

We only really know stuff about the NSA/Five Eyes due to the Snowden leaks. For example, the SIGINT Enabling Project attempts to influence/blackmail companies to weaken their security with backdoors.

Throughout the course, we will use the NSA to mean "generic nation-state level adversary", since if you can defeat the NSA, you can defeat basically anyone.

Anyways, weakened DES was adopted by NIST in 1977 as FIPS 46 in 1977, then as a banking standard as ANSI X3.92 in 1982 (replaced by Triple-DES in 1988). From 1997–2001, a new contest developed the Advanced Encryption Standard (AES), which is the current standard block cipher.

Desired properties of block ciphers (Shannon, 1949):

1. Diffusion: Each ciphertext bit should depend on all plaintext bits.
2. Confusion: The key–ciphertext relationship should be complicated.
3. Key length: Keys should be small but not too small to be searchable.
4. Simplicity: Ease of implementation and analysis.
5. Speed: Runs quickly on all reasonable hardware.
6. Platform: Can be implemented in hardware and software.

[1] of standardized peanut butter fame

Cryptoscheme 5 (DES)

The design principles of DES are still classified, so we just treat it as a black box for this course. We only need to know that there is a 56-bit key and 64-bit blocks.

The DES key space is not very big. Exhaustive search on DES takes 2^{56} operations. In 1997, this took three months. In 2012, it takes 11.5 hours.

The blocks are also not very large. By the birthday paradox, there is a collision every 2^{32} blocks. This is an information leak, breaking semantic security.

These are the only (known) weaknesses in DES.

Definition 2.3.3 (multiple encryption)

Re-encrypt the ciphertext more times with different keys.

This is not always more secure. For example, in the simple substitution cipher, permutations can be composed and do not introduce more security.

Cryptoscheme 6 (Double-DES)

Pick a secret key $k = (k_1, k_2) \in_R \{0,1\}^{112}$.

Then, encrypt $E_{k_2}(E_{k_1}(m))$ where E is DES encryption

Likewise, decrypt $E_{k_2}^{-1}(E_{k_1}^{-1}(m))$ where E^{-1} is DES decryption.

We now have an exhaustive key search of 2^{112} operations, which is better. However, there is an attack which reduces this to breaking DES.

List of Named Results

List of Cryptosystems

Index of Defined Terms