Lecture 1: Course Intro

6.5610 – Spring 2023
Henry Corrigan-Gibbs
MIT

Plan

- What is cryptography/security?
- Course info
- Logistics
- History of crypto
- First encryption system

Logistical Notes

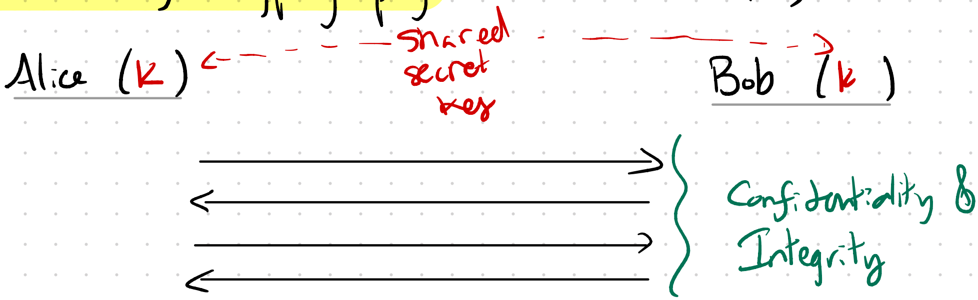
- Course site: 6S610.csail.mit.edu
- Sign up for Piazza & Gradescope
  - ↑ Use for all coms w/ course staff
- OH posted on Piazza

This class is about how to build and use cryptography.

↳ Should be useful even if you've taken 6.1600 already
(More technical, more formal)

The lectures are in three modules, mirroring history of cryptography...

I. Secret-Key Cryptography (??? BCE – 1976)

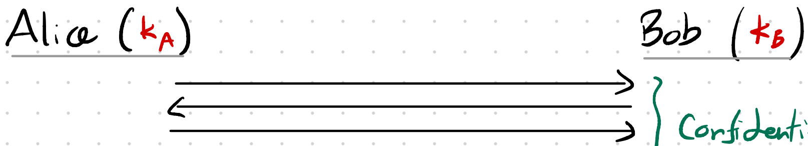


Initial applications: military, diplomatic, some biz (telegraph)
Now: File enc, disk enc, web traffic, cell traffic, ...

Qs: What does it mean for enc scheme to be secure?
How do we construct enc & auth schemes?
How have HW advances changed enc designs?
PRF, OWF, PRP, Enc, AE, MAC

Problem: Where do you get shared key?
↳ Manageable for high-sec settings, less so for commerce
↳ Internet, ATM, phone network, ...

II. Public-Key Cryptography (1974 - now)



} Confidentiality & integrity*

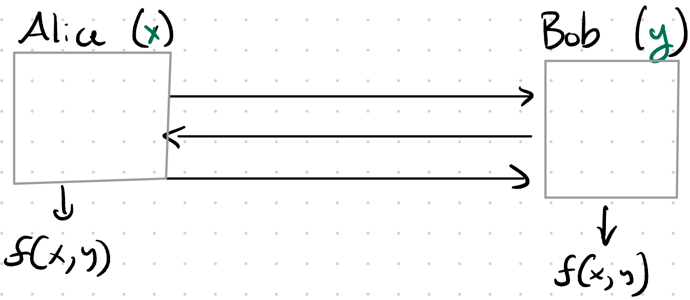
- First proposed by Ralph Merkle (1974) in undergrad sec class @ UCB
- First construction of key ex given by DH (1976)
- Public-key enc RSA @ MIT (1977)

↖ why 70s? Parallel to CS theory?

Applications: HTTPS, TLS, SSH, etc, Code signing, ...

Problem: Securing communication often insufficient. Want f. secure computation.

III. Cryptographic Protocols (1980s - now)



} A & B learn f(x,y) and "nothing more" + more props.

- Applications: outsourced computation, ZK proofs, e-voting, anon comm., private dB lookups, ...

Problem: Few of these advanced techniques work in practice.

What is security / cryptography?

Successfully performing a task (communicating, computing, ...) in presence of adversaries.

Examples: [VERY INFORMAL!]

- Given safe is locked, no adversary using only a screwdriver should be able to extract jewels from safe in ≤ 8 hrs.

- Given an encryption of a message m, no polynomial-time adversary should be able to recover m.

- Given complete control of one app on my phone, no attacker (running arbitrary code) should be able to extract my banking password.

Typically we define security in terms of

- Attacker's power: computational resources
  information
  influence

- Attacker's goal: what constitutes a successful attack?
  what is attacker trying to do?

Recipe for building secure systems:

1) Define class of attackers
2) Define security goal
3) Construct system that protects goal against all attackers in class [with "formal" proof]
   ↳ Often using assumptions (e.g. that no poly-time alg for factoring)

Ways systems break

1) Don't protect against large enough class of adversaries
   ↳ e.g. attacker uses a blowtorch

2) Don't define strong enough security goal
   ↳ e.g. attacker steals safe

3) Assumption is false.
   ↳ e.g. factoring turns out to be easy.

⇒ Want to achieve strongest possible security goal against largest class of advs, under min assumptions

* When security break happens, it's worth thinking about which of these three things failed.

Course Staff

* Yael Kalai, Henry Corrigan-Gibbs (profs)
* Andrea Lin, Kelsey Merrill, Simon Langowski (TAs)
* Alexandra Henzinger, Kyle Hogan (LAs)

→ Anon feedback form on website

Course Structure

- Lecture M/W, recitation F (except as noted on calendar)
- Assignments
  * Four psets (some coding) – 40%
    ↳ Must typeset in Latex!
  * One Quiz – 20% [April 5]
  * Final project – 40% [See course site]
    ↳ See details on course site.

Policies (see website)

- Four HW - first 3 groups assigned by TAs
- 4th group up to you (or ask TAs)
  - ↳ Also project group
- MUST write up sdns on your own
- MAY discuss HW w/ group members
  - ↳ NOT anyone else
- MAY use external sources - CITE ALL!

Also, late HW:

- 48 hrs ext w/ advance TA permission
- >48 hrs (or many exts) requires S^3 dean email

Which reminds me...

* College / grad school can be very stressful
* Life happens: illness, family trouble, financial issues, etc.

There are many resources at MIT to help YOU! (Yes, you!)

- Undergrads: S^3 for anything
- Grad students: GradSupport
- All: Student mental health – confidential by law*

  ...feeling sad? hopeless? not sleeping enough? too much?
  ...not enjoying things you usually like? There's help!

- Always: TAs, me, Yael... will connect you to the right support on campus (even after 6.5610)

Questions?

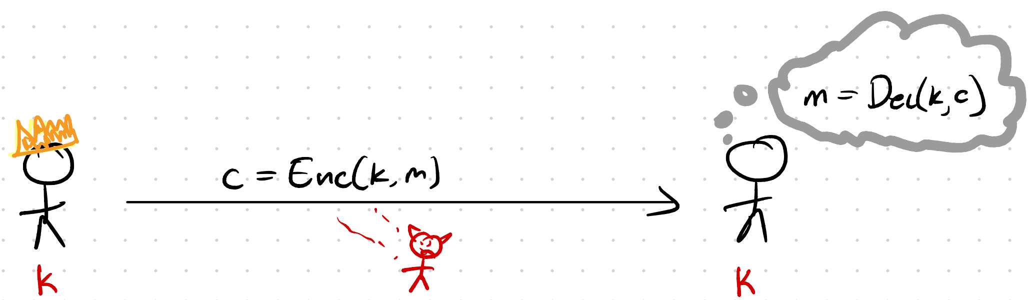
To set the scene for this course, let's look back at the history of crypto... See Kahn "Codebreakers"

Symmetric/
Secret-key crypto has existed for thousands of years. Depending on what you consider "crypto," as early as 1900 BCE.

A sym-key enc scheme over keyspace $\mathcal{K}$, msg space $\mathcal{M}$ consists of two eff algs (Enc, Dec) s.t. $\forall k \in \mathcal{K}, m \in \mathcal{M}$

$$\text{Dec}(k, \text{Enc}(k, m)) = m$$

We also need a security property, but leave that aside for now.



Through the 19th century, substitution ciphers were common

Caesar Cipher (≈ 50 BCE, Julius Caesar)

$\mathcal{K} = \{0, \ldots, 25\}$

$\mathcal{M} = \{0 = A, 1 = B, 2 = C, 3 = D, \ldots, 25 = Z\}$

$\text{Enc}(k, m) := m + k \mod 26$

$\text{Dec}(k, c) := c - k \mod 26$

Correct! $\forall\, m \in \mathcal{M},\ k \in \mathcal{K}$

$$\text{Dec}(k, \text{Enc}(k, m)) = (m + k) - k \mod 26$$
$$= m \mod 26.$$

To encrypt longer msgs, just run $\text{Enc}(k, \cdot)$ on each letter of msg.

Problem 1: Keyspace is too small - only 26 keys. Attacker can try them all!

You might object to this criticism, since attacker has to know that msg is encrypted w/ Caesar cipher to attack...

Kerckhoffs's Principle: (1880s)

"The attacker knows the system."

or

"The key is the only secret; all algs are public."

Why?

* Empirically, attacker usually gets the algorithm
  - ↳ Many people have to know alg
  - ↳ Few need to know the keys
  - (Many many examples...

* After compromise, easier to change key than to change alg.

* Simplifies analysis: If alg is secret, just consider it part of the key. So key is the secret stuff, alg is all else.

Substitution Cipher [Attempt II]

$\mathcal{K} = S_{26}$ — set of all perms on $\{0, \ldots, 25\}$

e.g. [A → C, B → Y, C → R, ..., Z → H]

$\mathcal{M} = \{ 0 = A,\ 1 = B, \ldots \}$

$Enc(\pi, m) := \pi(m)$ // e.g. "ABC" ↦ "CYR"

$Dec(\pi, c) := \pi^{-1}(c)$ // "CYR" ↦ "ABC"

Now $|\mathcal{K}| = 26 \cdot 25 \cdot 24 \cdot \cdots \cdot 3 \cdot 2 \cdot 1 \approx 2^{88}$ keys

↳ Trying all keys would take ≥1 million CPU-years of compute (energy required to boil a lake of 200 sq miles) {Lenstra, Kleinjung, Thomé}

Are we safe?

Problem: Frequency Analysis ... back to Al-Kindi 800s CE

* Substitution cipher preserves letter frequencies.

THE␣HOUSE␣IS␣ON␣FIRE

XQR␣QFABR␣LM␣NZ␣CLOR

→ $E \approx 12\%$, $T \approx 9\%$, $A \approx 8\%$, ...

→ Can also look at bigrams & trigrams ("the", etc)

* More on pset 1.

Having a large keyspace is necessary, but not sufficient, for security.

In Renaissance and after, polyalphabetic substitution ciphers became common.
↳ Freq analysis still a problem.

[See Bellovin 2011]

In late 1800s, early 1900s, a few people independently developed a cryptosystem that we now call the "One-time pad."

One-Time Pad

- Just the Caesar cipher with a fresh key for each msg symbol encrypted.

$\mathcal{M} = \{0,1\}^n$ $\qquad$ $Enc(k,m) := k \oplus m$

$\mathcal{K} = \{0,1\}^n$ $\qquad$ $Dec(k,c) := k \oplus c$

$\forall k \in \mathcal{K}, \forall m \in \mathcal{M} \quad Dec(k, Enc(k,m)) = (k \oplus m) \oplus k$
$= m$

$$\begin{array}{rl} m = & 01101 \\ k & 10011 \\ \hline c = & 11110 \end{array} \oplus$$

Problem: Key is as long as the message.
And, can only use key to enc one msg

BUT, OTP has one major benefit...

Perfect (One-time) Security

[Shannon 1949]

An enc scheme (Enc, Dec) over $\mathcal{K}, \mathcal{M}, \mathcal{C}$ (ct space) has perfect/information-theoretic security if

$\forall\, m_0, m_1 \in \mathcal{M} \quad \forall\, c \in \mathcal{C}$

$$\Pr[\mathrm{Enc}(k, m_0) = c : k \xleftarrow{R} \mathcal{K}] = \Pr[\mathrm{Enc}(k, m_1) = c : k \xleftarrow{R} \mathcal{K}]$$

Assume all msgs in $\mathcal{M}$ have same length.

$\Rightarrow$ Seeing ct leaks no information about plaintext to adversary.

For all advs $\mathcal{A}$, even running in unbounded time, $\forall\, m_0, m_1 \in \mathcal{M}$

$$\Pr\left[\mathcal{A}(c) = 1 : \begin{array}{l} k \xleftarrow{R} \mathcal{K} \\ c \leftarrow \mathrm{Enc}(k, m_0) \end{array}\right] = \Pr\left[\mathcal{A}(c) = 1 : \begin{array}{l} k \xleftarrow{R} \mathcal{K} \\ c \leftarrow \mathrm{Enc}(k, m_1) \end{array}\right]$$

Thm [Shannon]: One-time pad has perfect one-time security.

Pf idea: For all $m \in \mathcal{M}$, $c \in \mathcal{C}$, $\Pr[c = \mathrm{Enc}(k, m) : k \xleftarrow{R} \mathcal{K}] = 1/|\mathcal{K}|$.

Why we are not done...

1. Perfect secrecy isn't enough.

$$\begin{array}{r} m = 0110 \\ k = 1101 \\ \hline c = 1011 \end{array} \quad \longrightarrow \quad \begin{array}{r} c = 1010 \\ k = 1101 \\ \hline 0111 \end{array}$$

2. Key is too long / want to reuse key.

Dan Boneh's formulation

Thm [Shannon '49] Perfect security $\Rightarrow |\mathcal{K}| \geq |\mathcal{M}|$

Intuition: If $|\mathcal{K}| < |\mathcal{M}|$ then some (m, c) pairs are infeasible.

$\Rightarrow$ If you want short keys (& we do) you'll have to settle for "computational security." ← Next time

↳ Get security only against time-bounded attackers

But the story is even sadder...

Since the existence of computationally secure ciphers ⇒ $P \neq NP$ (& more!), we prove security under computational assumptions.

- Some are nice (e.g. ∄ poly time alg for factoring integers)
- Some are not so nice (e.g. my cipher is secure)

The good news!

→ The most exciting developments in 4000 years of cryptography have happened in the last 50.

→ CS is the star of the show.

→ We have precise & elegant ways to define diff types of security and to relate security of primitives to each other

→ Crypto draws on all the best of CS & more... algs, complexity, data structs, low-level hacking, number theory, policy, etc...

We love cryptography! Ask us Qs on Piazza, in OH, in class, etc. Any time!