Cryptography
Lecture 1

Lecture by Dr. Alon Rosen
Typeset by Steven Karas

2019-03-05
Last edited 18:12:07 2019-03-05

Disclaimer These notes are based on the lectures for the course Cryptography, taught by Dr. Alon Rosen at IDC Herzliyah in the spring semester of 2018/2019. Sections may be based on the lecture slides prepared by Dr. Alon Rosen.

1 Grading

90% exam, 10% homeworks. There will be 6 homework assignments.

2 History of Cryptography

$$\underbrace{\text{CRYPTO}}_{\text{"hidden"}}\underbrace{\text{GRAPHY}}_{\text{"to write"}}$$

Despite being a field of study for over 3000 years, only in the last 25-50 years it's become a proper field of scientific study. The field covers the concept of secrets from symmetric encryption through secure computation.

From around 500 BCE until World War 2, most cryptography followed a simple cycle of design, break, repair, break, ... [1]

Shannon in 1949[6] published a paper that established Cryptography as a rigorous study of mathematics. He described perfect secrecy, and came to a pessimistic conclusion that the security of a message is limited by the availability of pure entropy.

Diffie and Hellman in the 1976[1] published a paper that described the first asymmetric key exchange cryptosystem. They described the difference between infeasible versus impossible computational security, and based their key exchange system on complex computation. They won the Turing Award for their work in 2015.

Ron Rivest, Adi Shamir and Leonard Adelman from MIT published in 1978[5] the first practical asymmetric cryptosystem. They published a set of keys of various sizes for the public to factor for prize money. The largest yet factored is called RSA-768.

Shafi Goldwasser and Silvio Micali in the 1984[3] published definitions of computational security building on top of Shannon's work. They won the Turing Award in 2012.

Scope of cryptography

private-key most communication uses this method. DES and AES are the most common ciphers.
public-key DH, Rabin, RSA, etc.
authentication and integrity ensuring that messages are not changed or forged.
secure multi-party computation computing answers without sharing too much information
blockchain
private information retrieval

[1] True story: at one point, the Americans were using onetime pads for encrypting messages between the American embassy in Moscow and Washington. The Soviets were able to plant microphones inside the embassy, and then hired hundreds of people who were able to decode the letters typed by the Americans before they were encrypted.

Basic modern approach

1. Define - has to be meaningful
2. Construct - efficient, but initially just feasible
3. Prove - assumptions have to be reasonable

There is a central assumption that there exist hard problems.

It's more or less safe to assume that we have 2^{100} computational cycles worldwide in a year (as an upper bound). It would take us at least 2^{28} years to brute force a 128-bit key.

3 Basic concepts

In the basic model, consider two parties Alice and Bob communicating over an insecure channel. We will consider a message being sent by Alice to Bob, and two parties Eve and Mallory. Eve's goal is to discover the contents of the message, whereas Mallory's goal is to change it in a way that Bob won't know. Eve and Mallory may or may not be the same entity.

Symmetric Encryption Alice and Bob can agree on a secret code (cryptosystem) where they are the only ones who know that secret.

Asymmetric Encryption Alice and Bob can agree on a secret code without using a secure channel.

Layers of cryptography

1. Hard problems
2. Cryptographic primitives such as encryption, signatures, pseudorandom generators, zero-knowledge proofs, etc
3. Protocols
4. Secure systems and implementations

More advanced layers require deeper knowledge of previous layers, and are generally more difficult to construct.

Base assumptions All the following logically follow from each other:

$$\exists = \text{we can construct}$$

$$\nexists \text{ poly-time algorithm for factoring integers}$$

$$\exists \text{ one-way functions}$$

$$\exists \text{ PRG}$$

$$\exists \text{ secret-key encryption}$$

$$\exists \text{ secure election protocols}$$

3.1 Hard Problems

An algorithm A is defined by $A(x),\ x \in \Sigma^*$ where usually $\Sigma = \{0,1\}$, and denoted as:

$$A : \Sigma^* \to \Sigma^*$$

Computation We say that A computes

$$f : \{0,1\}^* \to \{0,1\}^*$$

if for an input x, A outputs $f(x)$.

Decision We say taht A decides $L \subseteq \{0,1\}^*$ if:

$$A(x) = 1 \Leftrightarrow x \in L$$

Time Complexity We say that A runs in time T where $|x|$ denotes the length of an input x:

$$T : \mathbb{N} \to \mathbb{N} : \forall x,\ A(x) \text{ halts after } T(|x|) \text{ steps}$$

We say that A runs in polynomial time if it runs in time $O(n^c)$ for some constant c.
We consider polytime as being efficient and feasible.

Examples of polytime problems

$$\text{Div}(x, y) = (q, r) \text{ s.t. } x = y \cdot q + r,\ 0 \leq r < y$$

$$\text{GCD}(x, y) = \text{largest } z \text{ s.t. } z|x \text{ and } z|y$$

$$\text{ModExp}(x, y, z) = x^y \mod z$$

Problems believed to not be polytime *NP* is the set of all decision problems that have short proofs of membership. $L \in NP$ means that $\exists V \in P$ and a polynomial q such that:

$$x \in L \Leftrightarrow \exists w, |w| \leq q(|x|), V(x, w) = 1$$

where w is the witness for $x \in L$.
We say that L is *NP*-complete if a polytime algorithm for L implies a polytime algorithm for any $L \in NP$.

Examples of such problems

$$\text{SAT} = \{\phi(x_1, \ldots, x_n) | \exists w\ \phi(w) = 1\}$$

$$\text{Factor}(N) = \exists P, Q : PQ = N$$

3.1.1 Repeated squaring

A naive approach to implementing ModExp is to first compute $a = x^y$ and then $a \mod z$. However, this is not polytime.
To find a polytime approach, consider the identity:

$$(a \mod z)(b \mod z) = ab \mod z$$

Which gives us an efficient construction:

$$y = y_0 + y_1 \cdot 2 + y_2 \cdot 4 + \cdots + y_t 2^t$$
$$x^y = x^{y_0} \cdot (x^2)^{y_1} \cdot (x^4)^{y_2} \cdot \cdots \cdot (x^{2^t})^{y_t}$$

Where each step is $\mod z$. This gives us $t = \log y = |y|$ steps that each finish in polytime.

3.1.2 Integer Factorization

Given n, we want to find a nontrivial factor ($\neq 1$) of n.
We do not know if this is in P or *NP*-complete.
Some algorithms:

- Exhaustive search (trial division). Takes $\widetilde{\Theta}(n)$. Can be improved to $\widetilde{\Theta}(\sqrt{n})$.
- Quadratic sieve. Takes $2^{\widetilde{\Theta}(\sqrt{|N|})}$
- Number field sieve. Takes $2^{\widetilde{\Theta}(\sqrt[3]{|N|})}$

4 Next week

We did not cover randomized algorithms, but it is strongly recommended to go over the material before next week.

References

[1] Whitfield Diffie and Martin Hellman. New directions in cryptography. *IEEE transactions on Information Theory*, 22(6):644–654, 1976.

[2] Oded Goldreich. *Foundations of Cryptography: Volume 1, Basic Tools*, volume 1. Cambridge University Press, 2000.

[3] Shafi Goldwasser and Silvio Micali. Probabilistic encryption. *Journal of computer and system sciences*, 28(2):270–299, 1984.

[4] Jonathan Katz and Yehuda Lindell. *Introduction to Modern Cryptography, Second Edition*. Chapman & Hall/CRC, 2nd edition, 2014.

[5] R. L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM*, 21(2):120–126, February 1978.

[6] C. E. Shannon. Communication theory of secrecy systems. *Bell System Technical Journal*, 28(4):656–715, 1949.