Introduction to Cryptography

Lecture #1, September 11, 1991

1 Course Overview

Today, we give a brief overview of the topics that will be covered in this course.

1.1 Secure Communication

The central problem in cryptography is the problem of secure communication across an insecure channel. This problem is often stated in the following setting: person A and person B wish to communicate in secrecy across a line to which an adversary (or line-tapper) has access. The traditional solution to this problem requires that A and B hold a secret meeting before the transmission takes place and agree upon a secret key $SK$ which will be kept private and used during transmission for encryption and decryption. Also, traditionally, these systems were invented in an ad hoc manner and were all eventually broken. An example of this is the substitution cipher $f : \Sigma \rightarrow \Sigma$ which to encrypt and decrypt their messages (so if the message is $m = m_1m_2\ldots m_n$, where $m_i \in \Sigma$, then the encryption function is $E(m) = f(m_1)\ldots f(m_n)$.)

Such traditional systems had two shortcomings:

1. They required a secret meeting before transmission.
2. They did not give an a priori guarantee of security.

In 1949, Claude E. Shannon produced a theory of perfect secrecy (see [11]). In his framework, the adversary was considered to have available unlimited computational resources. This work studies the above situation (where A and B use secretly defined $SK$ for encryption and decryption and the adversary has unlimited computational resources) and bounds the amount of information that can be securely transmitted as a function of the amount of information agreed upon beforehand: $|m| \leq |SK|$, where $m$ is the transmitted message. For Shannon, securely meant with perfect secrecy, which he defined in the following way.

**Definition 1** Consider a finite set of messages $M$, a finite set of keys $K$, a set of ciphertexts $C$, and an encryption function $f : M \times K \rightarrow C$. Each message $m \in M$ and each key $SK \in K$ has an a priori probability, $Pr[m]$ and $Pr[SK]$, of being selected by the sender. (It is assumed that the adversary has access to these distributions and the crypt function $f$.) After the adversary has seen $c = f(m, SK)$, he may compute the a

---

1 These notes were scribed by Alexander Russell.
posteriori probability of \( m \), \( \Pr[m \mid c] \), the probability that \( m \) produces ciphertext \( c \) (where the probability is taken over the choices of \( SK \in K \)). A cryptosystem is perfectly secure when for all messages \( m \) and ciphertexts \( c \), \( \Pr[m] = \Pr[m \mid c] \).

**Example 1** Perfect security is achievable using the Vernam one time pad cryptosystem. In this system, the two parties A and B secretly agree upon a random bit string \( SK = b_1b_2\ldots b_n \), where \( b_i \in \{0, 1\} \). A and B then encrypt a message \( m = m_1m_2\ldots m_n \), where \( m_i \in \{0, 1\} \) with the function \( E_{SK} : \{0, 1\}^n \rightarrow \{0, 1\}^n \) defined by \( E_{SK}(m) = m \oplus SK \) (bitwise exclusive or). Similarly, they decrypt the ciphertext \( c = c_1c_2\ldots c_n \) by computing the function \( D_{SK}(c) = c \oplus SK \). It is easy to verify that for every \( m, y \in \{0, 1\}^n \), \( \Pr_{SK}[E_{SK}(m) = y] = \frac{1}{2^n} \), so that the one time pad is perfectly secure.

From Shannon’s work, should the length of the clear text (message) exceed the length of the secret key (\( SK \)), then some information will be revealed. For example, in the case of the one time pad, should another message \( m' \) be sent using \( E_{SK}(m') = m' \oplus SK \), the adversary can compute \( E_{SK}(m) \oplus E_{SK}(m') = m \oplus m' \). (The structure \( \{0, 1\}^n, \oplus, 0\cdots0 \) is an abelian group with \( x \oplus x = 0 \).) However, if the length of the one time pad is the same as the number of bits ever to be transmitted over the insecure channel using the one time pad, then the system is perfectly secure. Still, a secret meeting must be held prior to any transmission.

Modern cryptography abandons the assumption that the adversary has available infinite computing resources, assuming rather that his computation must be resource bounded in some reasonable way (for instance, polynomially bounded in the size of the clear text). For a discussion of this, see [4]. Within this framework, many new questions have been asked:

1. Can A and B agree on a key \( SK \) in a secret way and subsequently exchange \( p(|SK|) \) secure bits where \( p \) is a polynomial?

2. Can A and B exchange secure messages without even meeting?

3. Can A sign messages digitally so that everyone can verify the authenticity of the signature (that is, everyone can check that A produced the signature but nobody can forge A’s signature)?

With respect to an infinitely computationally powerful adversary, the answer to questions 2 and 3 is negative.

The next topic of interest in this course is pseudo-random generators.
1.2 Pseudo-Random Generators

Question 1 has been addressed by studying the notion of a pseudo-random generator.

**Definition 2** A pseudo-random bit generator is a function \( G : \{0,1\}^n \rightarrow \{0,1\}^{p(n)} \) (where \( p \) is a polynomial) computed by a deterministic algorithm which “stretches” a truly random string, \( s \), (called a seed) into a longer string, \( G(s) \), in such a way that the resulting strings are “indistinguishable” from truly random strings of length \( p(n) \). (The term indistinguishable will be precisely defined later in the course.)

The existence of a pseudo-random bit generator would give an affirmative answer to question 1 because the original pad \( SK \) could be stretched to a long pad \( G(SK) \) and used in \( |m| \) bit blocks. It is not known if such generators exist but we will show that pseudo-random generators do exist if one way functions exist under certain complexity assumptions.

**Definition 3** Informally, a one way function is an efficiently computable function \( f \) with a non-efficiently computable inverse.

A central result that will be shown is the following.

**Theorem 1** One way functions exist if and only if there exist pseudo-random number generators.

There are further issues related to efficiency in the study of pseudo-randomness. For instance, what is the complexity of the computation of a pseudo-random number generator \( G \)? Is it possible to design a \( G \) which can be quickly computed in parallel?

1.3 Public Key Cryptography

Question 2 is addressed by the field of public key cryptography. In a public key cryptosystem, each user \( A \) makes public certain information (denoted \( PK \) for public key) and keeps secret other information (denoted \( SK \) for secret key). The public information \( PK \) should enable anyone to send \( A \) an encrypted message, whereas the corresponding secret information \( SK \) enables \( A \) to decrypt the encrypted messages received. Diffie and Hellman introduced such systems in their seminal 1976 paper [3]. It is unknown if secure public key cryptosystems exist. Clearly, from an information theoretic point of view, they cannot. However, we will define security with respect to a restricted adversary and under certain complexity assumptions, they can be shown to exist. See also [10].

**Definition 4** Informally, a trapdoor function is an efficiently computable function \( f \) such that \( f^{-1} \) is not efficiently computable unless some extra information (the trapdoor) is known.

**Theorem 2** If there exist trapdoor functions, then there exist secure public key cryptosystems.
1.4 One Way Functions

As we saw above, pseudo-random number generators and public key cryptosystems (the cornerstones of modern cryptography) exist if one way functions exist. As of today, we cannot prove that such functions exist. However, certain functions have been conjectured to be one way functions. Proving that a function \( f \) is one way involves

1. Showing that \( f \) is easy to compute.
2. Showing that \( f^{-1} \) is not easy to compute.

Let’s examine several proposed candidates.

1. **Factoring.** The function \( f : (x, y) \mapsto xy \) where \( x, y \in \mathbb{Z} \) is conjectured to be a one way function. The fastest proven factoring algorithms to date are all variations on Dixon’s random squares algorithm (see [7]) with expected running time \( L(n) \sqrt[2]{2} \) where \( L(n) = e^{\sqrt{\log n \log \log n}} \) and \( n \) is the integer to be factored. Recent progress in factoring was made by Lenstra, Lenstra, Manasse, and Pollard who introduced the number field sieve algorithm; a factoring algorithm proved under a certain set of assumptions to factor integers of form \( r^e \pm s \) in expected time \( e^{((c+o(1))(\log n)^{1/3}(\log \log n)^{2/3})} \) where \( r \) and \( |s| \) have a fixed upper bound (see [8]). Adleman generalized this algorithm to arbitrary integers, maintaining the same conjectured running time (see [1]).

2. **The Discrete Logarithm Problem.** The multiplicative group \( \mathbb{Z}_p^* = \{x < p \mid (x, p) = 1\} \) where \( p \) is a prime is cyclic, so that \( \mathbb{Z}_p^* = \{g^i \mod p \mid 1 \leq i \leq p - 1\} \) for some generator \( g \in \mathbb{Z}_p^* \) (see [5, 6]). The function \( f : (p, g, x) \mapsto (g^x \mod p, p, g) \) where \( p \) is a prime, \( g \) is a generator for \( \mathbb{Z}_p^* \) and \( x \in \mathbb{Z}_p^* \) is conjectured to be a one way function. Computing \( f(p, g, x) \) can be done in polynomial time using repeated squaring. However, the fastest proven algorithm for its inverse, the discrete logarithm, is the index-calculus algorithm with expected running time \( L(p)^{\sqrt[2]{2}} \) (see [7]). For the history of this algorithm refer to section 4 of [9]. An interesting problem is to find an algorithm which will generate a prime \( p \) and a generator \( g \) for \( \mathbb{Z}_p^* \). It is not known how to find a generator for \( \mathbb{Z}_p^* \). However, in [2], Bach shows how to generate random factored integers (in a given range \( [N/2, N] \)). Coupled with a fast primality tester (as described in [7], for example), this can be used to efficiently generate random vectors \( (p - 1, q_1^{a_1}, \ldots, q_k^{a_k}) \) with \( p \) and \( q_1, \ldots, q_k \) prime and such that \( p - 1 = \prod q_i^{a_i} \). Then picking \( g \in \mathbb{Z}_p^* \) at random, it can quickly be checked if \( (g, p - 1) = 1, g^{q_i} \neq 1 \mod p \) for \( i = 1, \ldots, k, \) and \( g^{p-1} \equiv 1 \mod p, \) in which case \( g \) has order \( p - 1 \) and thus \( g \) is a generator for \( \mathbb{Z}_p^* \). It can be shown that the density of the generators in \( \mathbb{Z}_p^* \) is high so that few choices are required. For a given prime \( p \), the problem of efficiently determining a generator for \( \mathbb{Z}_p^* \) is an intriguing open research problem.
3. Subset Sum. Let $\vec{a} = (a_1, \ldots, a_n)$ where $a_i \in \{0, 1\}^m$ and $\vec{s} = (s_1, \ldots, s_n)$ where $s_i \in \{0, 1\}$. Let $f : (\vec{a}, \vec{s}) \mapsto (\vec{a}, \sum_{i=1}^{n} s_i a_i)$. An inverse of $(\vec{a}, \sum_{i=1}^{n} s_i a_i)$ under $f$ is any pair $(\vec{a}', \vec{s}')$ for which $\sum_{i=1}^{n} s_i a_i = \sum_{i=1}^{n} s_i' a_i$. The function $f$ is a candidate for a one way function. The associated decision problem (given $(\vec{a}, y)$, does there exists $\vec{s}$ so that $\sum_{i=1}^{n} s_i a_i = y$?) is NP-complete.

A one way function is said to be weak if it is hard to invert only on a small fraction of its domain. If a one way function is not efficiently invertible almost everywhere in its domain, it is said to be strong. We will see in the next lecture that a weak one way function may be used to construct a strong one way function.

1.5 Two Party Protocols

For discussions concerning two party protocols, it is assumed that A and B cannot trust each other.

Example 2 A and B have $x_a$ and $x_b$, respectively, which they wish to remain secret. A must compute $g(x_a, x_b)$ and B must compute $h(x_a, x_b)$.

One solution to such problems is to use a trusted center; specifically, an agent C who, after receiving $x_a$ and $x_b$, will return $g(x_a, x_b)$ to A and $h(x_a, x_b)$ to B. One of the goals of cryptography is to provide a solution to this problem that does not require a trusted center. Such a result is stated next.

Theorem 3 If there exists a trapdoor function, then there exists a secure method (for the above two party protocol) without a trusted center.

This problem can be extended to $N$ parties in which case some unconditional results can be proven as well.

Theorem 4 Consider a collection of $N$ parties, $t$ of which may be dishonest. For $i = 1, \ldots, N$, let $x_i$ be the input of party $i$ which he wishes to remain secret. The goal of party $i$ for each $i$ is to compute some function $f_i(x_1, \ldots, x_N)$.

1. If trapdoor functions exist then there exists a secure method for solving this problem without the use of a trusted center.

2. If $N \geq 3t+1$ and every two parties can communicate secretly, then, unconditionally, there exists a secure method for computing $f_i(x_1, \ldots, x_N)$ for each $1 \leq i \leq N$ without the use of a trusted center.

Some examples of $N$ party computation are the following.

Example 3 Electronic Voting. $N$ parties wish to vote, maintaining the secrecy of their votes, yet being guaranteed that the final count is correct.
Example 4 List Comparison. Two government agencies each have private lists of people and wish to compute the intersection of the lists without revealing their entire lists.

An important concept related to the proof of Theorem 3 is that of zero knowledge interactive proofs. Zero knowledge interactive proofs are methods to bound the amount of knowledge necessary to be transmitted in order to convince an untrusted and nontrusting party of the truth of a proposition.

References


