COM S 6830 – Cryptography Aug 27, 2009

Lecture 1: Introduction

Instructor: Rafael Pass *Scribe: Eleanor Birrell*

1 Logistics

- CS 6830 - Introduction to Cryptography
- Prof: Rafael Pass (rafael@cs)
- TA: Dustin Tseng (wdtseng@cs)

- 4-5 HW assignments
- Scribe notes (24 hours to hand in)
- HW 1 out (due Sept. 10)
- Handout: Notation and Probability

2 Cryptography Basics

- "crypto" = secrets, "graphy" = writing, invented 2500-400
- Traditional Purpose: secure communication. A (Alice) wants to send a message m to B (Bob) with out the adversary E (Eve) intercepting or reading the message. Ie, A wants a sealed envelope that B can open but E cannot.
- Simple crypto: agree on "secret code"

Definition 1 (Private Key Encryption)**.** A triplet of algorithms (Gen, Enc, Dec) is called a *private key encryption scheme* over the message space M and keyspace K if the following three things hold:

1. Gen is a randomized algorithm $k \leftarrow Gen$ s.t. $k \in K$
2. Enc is a (randomized) algorithm that takes a key $k \in K$ and a message $m \in M$ and outputs a ciphertet $c = Enc_k(m)$.
3. Dec is an algorithm $m' = Dec_k(c) \in M$ such that for all $m \in M, k \in K$,

$$\Pr_r[Dec_k(Enc_k(m)) = m] = 1.$$

Kerchef's Principle (1884): Encryption and decryption algorithms should be public. (Corollary: key generation algorithm Gen *must* be randomized)

$A(k,m)$		$B(k)$
$c = Enc_k(m)$	$\rightarrow$	
		$m' = Dec_k(c)$

Table 1: A simple private-key encryption scheme.

Example Private-key Encryption Schemes

Caesar Cipher: Do a cyclic shift on each letter in the message.

$$
\begin{aligned}
&M = \{A, \ldots, Z\}^* \\
&K = \{0, \ldots, 25\} \\
&Gen\text{: } k \leftarrow_R \{0, \ldots, 25\} \\
&Enc_k(m_1 m_2 \ldots m_n) = c_1 \ldots c_n \text{ where } c_i = m_i + k \mod 26 \\
&Dec_k(c_1 c_2 \ldots c_n) = m'_1 \ldots m'_n \text{ where } m'_i = c_i - k \mod 26
\end{aligned}
$$

Eg, If $k = 3$, GOD MORN $\rightarrow$ JRG PRUQ.

Substitution Cipher: Perform a permutation on the letters of the alphabet.

$$
\begin{aligned}
&M = \{A, \ldots, Z\}^* \\
&K = \{ \text{ permutations on the letters}\} \\
&Gen\text{: } k = \text{ random permutation of } \{A, \ldots, Z\} \\
&Enc_k(m_1 m_2 \ldots m_n) = c_1 \ldots c_n \text{ where } c_i = k(m_i) \\
&Dec_k : (c_1 c_2 \ldots c_n) = m'_1 \ldots m'_n \text{ where } m'_i = k^{-1}(c_i)
\end{aligned}
$$

Proposition 1. *Caesar and Substitution Ciphers are private-key encryption schemes.*

Approaches to Cryptograph

Classic Crypto Cycle (an art)

1. Person A invents a cipher.
2. A claims (or proves) that known attacks don't work.
3. Cipher is employed.
4. Cipher gets broken.
5. Patch/improve/reject old cipher.

Modern Cryptography (a science)

1. Principles: start by defining in a precise, mathematical way what it means to be secure.
2. Provide precise mathematical assumptions (axioms – eg, factoring is intractable)
3. Construct schemes and *prove* that they are secure based on the assumed axioms. (Ie, if scheme S can be broken, then so can assumption A, therefore A is false.)

3 Secure computations protocols

Applications beyond secure communication. What if B is dishonest/malicious?

Example 1 (List comparison)**.** A, B have lists L_A, L_B and they want to compute $L_A \cap L_B$. Solution: both encrypt data and compare encrpyted databases

General Case: Secure Computation Consider A, B with inputs a, b respectively. They wish to compute $f(a, b)$ but A, B do not want to reveal their private inputs. There are two esired properties:

1. Privacy: a protocol does not reveal "any more" that $f(a, b)$
2. Correctness: output really is $f(a, b)$

Definition 2 (Informal)**.** A *secure 2-party computation* allows A, B to compute $f(a, b)$ with the same privacy and correctness as if a trusted third party had done it for them.

Theorem 1 (Informal)**.** *Under some number-theoretic assumptions, there exists a protocol for secure computation*

This problem (and result) can be generalized to secure multi-party computation with even more applications (Voting, auctions, etc.)

Example 2 (Love)**.** Question: Do Alice and Bob love each other? Function: Love $= \wedge$ (If love each other, then they want to know, but if only one loves the other, this should have the same result as if neither were in love.) The following protocol solves this problem:

1. Take five playing cards (3 identical hearts and 2 identical diamonds) Each has one heart, one diamond in their hand (to start). There are five places on the table; the third heart is placed facedown in the third (middle) position.
2. If Alice loves Bob, she places her diamond in position 1 and her heart in position 2, if not she places them the other way around.
3. If Bob loves Alice, he places his heart in position 4 and his diamond in position 5, if not he places them the other way around.

4. Put cards in a pile (in order), Alice cuts (permutes cards), then Bob cuts (permutes cards).

5. Reveal the cards in order. If three hearts in a row, then they both love each other, else they don't.

The correctness of this protocol relies on the fact that the only permutation in which three hearts are placed in a row is when Alice places a heart in position two and Bob places a heart in position 4. The privacy arises from the fact that the other three layouts are permutations of each other.

Zero-Knowledge Proofs are a special case of secure 2-party computations. For example, say that A has a number N which she claims is the product of two primes. She wants to prove to B that $N = pq$ is actually a product of two prime while only revealing veracity of claim. We will see such proofs later in the course.

4 Scope of CS6830

- Not in this course:
  - AES,DES,3DES
  - most efficient/practical schemes (opt for easily-explainable, intuitive)
  - how to break crypto
  - how to build secure systems
  - everything about crypto (or even everything about foundations/theory)
- The Layers of Modern Cryptography:
  - Computational Hardness Assumptions (Number-theoretic – eg, factoring is hard, Complexity-theoretic – eg, exist one-way functions, $NP \neq P$)
  - Primitives (encryption, pseudorandom generation, signatures)
  - Basic protocols (ZK, Authentication protocols, Identification protocols)
  - Advanced protocols (Secure computation, voting, auctions)
  - Secure systems
- Outline of Course:
  - Hardness, one-wayness
  - Indistinguishability
  - Randomness and Pseudorandomness
  - Knowledge
  - Secure computation (computing on secret inputs)