Cryptographic Secret Sharing

Girls Talk Math

Introduction

In this problem set, you will learn about ...

One last note about reading mathematical texts: it is very normal when reading math to read a passage or even a single sentence several times before understanding it properly. Also, never trust the author! Check every claim and calculation (time permitting). Take your time and never give up. Let's talk math!

Contents

1 Probability and Randomness

1.1 Introduction

(NG: Define probability, define what it means to be random. Introduce basic notation.)

1.2 Randomness in Cryptography

(NG: Define *information-theoretic security*. Define what a security parameter is (λ).)

1.3 Sharing Secrets

Secret sharing is a way to "split" a secret value (call it x) into pieces, called *shares*. Two useful properties a secret-sharing might have are *correctness* and *privacy*. Informally, correctness means that if we put shares back together, we get back the original secret; privacy says that each share by itself reveals nothing about the secret x.

These properties are important for practical uses of secret sharing, for example distributing shares among a large set of people so that no one knows the secret but some subset of them can recover the secret if they pool their information.

Example 1.0 (NG: Simple additive secret-sharing.)

Let's be a little more rigorous about these definitions, using the standard notation in cryptography. First, we'll define what a secret sharing scheme does without giving implementation details (there could be multiple ways of achieving the same thing, after all).

Definition 1 (Secret sharing scheme)**.** *Let $\mathcal{D}$ be the input domain and $\mathcal{D}_s$ be the share domain. A secret sharing scheme is a pair of efficient algorithms* $(\mathsf{Share}, \mathsf{Rec})$ *such that*

- Share *takes as input a secret $x \in \mathcal{D}$ and two numbers t, n and outputs n shares in $\mathcal{D}_s$.*
- Rec *takes as input m shares $s_1, \ldots, s_m \in D_s$. If $m < t$, it outputs a special symbol $\perp$ indicating failure; otherwise, it outputs some value $y \in \mathcal{D}$.*

The scheme is correct *if for all* $x \in \mathcal{D}$, $\mathsf{Rec}(\mathsf{Share}(x)) = x$.

Here's a visual representation:

(NG: Picture of two domains, with Share mapping in one direction and Rec back in the other.)

(NG: Define the privacy property and the cryptographic privacy game? The goal can be to have a (bonus) exercise in which they write a real cryptographic proof of privacy of a secret sharing scheme (likely the XOR scheme).)

Example 1.0 (NG: Secret sharing without correctness.)

Exercise 1.1 (NG: Is this scheme private?)

In the next sections, we'll see some examples of private secret sharing schemes.

2 A simple secret sharing

2.1 Binary Arithmetic

The exclusive-OR (XOR) operation is denoted by the symbol $\oplus$ and defined by the following truth table:

a	b	$a \oplus b$
0	0	0
0	1	1
1	0	1
1	1	0

(NG: explain binary representations of numbers and binary arithmetic.)

Exercise 2.1 Write the following numbers in binary.

1) 5
2) 12
3) 15
4) 16
5) 23

Exercise 2.2 Convert the following numbers to base 10.

1) 11
2) 110
3) 1001
4) 11110
5) 110101

Exercise 2.3

1) $101 \oplus 110$
2) $1 \oplus 111$

3) $1101001 \oplus 0010110$

4) $1101001 \oplus 1101001$

Exercise 2.4

1) $4 \oplus 5$

2) $11 \oplus 1$

3) $8 \oplus 4$

4) $7 \oplus 3$

2.2 Sharing Secrets using XOR

$\underline{\mathsf{Share}(s)}$
$s_1 \leftarrow\!\!\$\, \{1, \ldots, 2^\lambda\}$
$\mathsf{return}\ (s_1, s \oplus s_2)$

$\underline{\mathsf{Reconstruct}(s_1, s_2)}$
$\mathsf{return}\ s_1 \oplus s_2$

Exercise 2.5 Pick your favorite number and use the Share algorithm defined above to secret share it.

Exercise 2.6 What is $\mathsf{Rec}(14, 21, 49)$?

2.2.1 Proving Security*

(NG: Optional section. Introduce the privacy game, have them act as the adversary and try to beat the privacy and get some intuition about why it's hard, then work through a simple proof.)

In cryptography, security properties like privacy are defined using what are called "games". A game is a challenge in which an attacker (called the *adversary* and usually denoted by a curly $\mathcal{A}$) is given some information and has the goal of providing an answer that somehow breaks the security property of a scheme.

For example, in the case of privacy for a secret sharing scheme, we give the attacker a share and ask it to give us some information about the secret it came from. This should be almost impossible if the scheme is private.

Here is the *privacy game* for any secret sharing scheme $\mathcal{S} = (\mathsf{Share}, \mathsf{Rec})$:

1. The adversary $\mathcal{A}$ picks two values x_0, x_1 and a number i between 1 and n.

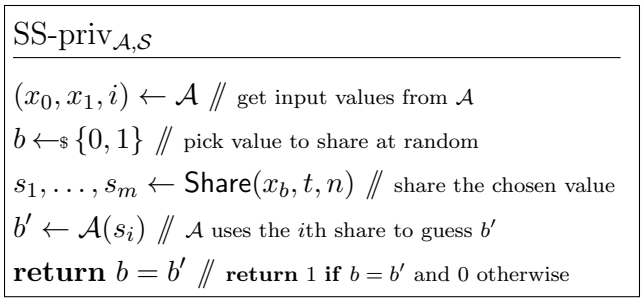
2. The game flips a coin to randomly choose one of those two values. This is usually written as picking a random value b from the set $\{0, 1\}$.

3. Now the game runs the Share algorithm on this randomly chosen value x_b to get shares $s_1, \ldots, s_m$. It gives the ith share to $\mathcal{A}$.

4. $\mathcal{A}$ tries to guess which of the two values x_0, x_1 were shared. More specifically, it outputs a guess $b' \in \{0, 1\}$.

5. If $b = b'$, we say the result of the game is 1 (to signify **true** or **success**), in which case we say that $\mathcal{A}$ "wins" the game; otherwise, it's 0.

In cryptography, these games are generally written much more compactly by using symbols. The privacy game above would be written as follows (without the comments, which are there to explain the notation):



Exercise 2.7 With a partner, play through the security a couple times using the XOR secret sharing scheme. One of you should take on the role of the game while the other acts as the adversary. After you've done this a couple of times, switch roles and repeat.

How successful was the adversary? If you were the adversary, what was difficult about your role? What was the key part of the scheme that ensured privacy?

(NG: Maybe try again with another scheme that is not deterministic but gives the adversary ¿1/2 odds of winning?)

Now that we have an idea of why the game is difficult to win consistently, let's rigorously define privacy by specifying how often the attacker should be able to win:

Definition 2 (secret sharing privacy)**.** *A secret sharing scheme* $\mathcal{S} = (\mathsf{Share}, \mathsf{Rec})$ *is* private *if, for all realistic*[1] *adversaries* $\mathcal{A}$*,*

$$\Pr[\text{SS-priv}_{\mathcal{A},\mathcal{S}}(t, n) = 1] - \frac{1}{2}$$

is small[2]*.*

What does this actually mean? No matter what adversary $\mathcal{A}$ we are dealing with, the probability that it can win the privacy game should be very close to one-half, which is what it should be if $\mathcal{A}$ were to randomly guess which value was shared.

[1] In cryptography we usually consider what are called *probabilistic polynomial-time*, or PPT, adversaries.

[2] bounded by a *negligible* function.

3 Shamir's Secret Sharing

3.1 Polynomials

(NG: y-intercept, zeroes)

3.1.1 Uniqueness

(NG: How many points uniquely define a polynomial)

3.1.2 Lagrange Interpolation

Exercise 3.1 (NG: Practice some manual Lagrange interpolation. Pick a polynomial, evaluate it at 3 points, then use those 3 points in Lagrance Interpolation and recover the polynomial.)

3.2 Sharing Secrets Using Polynomials

Shamir secret sharing is a $(t+1)$-out-of-n secret sharing protocol, for some numbers t and n. This means that we split the secret s into n values and distribute them to n people. Then, at least $t+1$ of those people must work together to recover s.

(NG: Don't introduce finite fields but maybe make a note that this should be done over finite fields.)

(NG: Use the Jupyter Notebook to play around with this.)