Secure Computational Models: Weekly Paper 3

Siddarth Ijju

October 1st, 2018

1 Introduction

This week's lectures focused on how to prove the security of an example t-out-of-n SSS, and also introduced the concept of indistinguishability and negligible functions

2 Shamir SSS

2.1 Linear Algebra Base

We discussed the fact that any degree d polynomial requires at least $d + 1$ points to define it. We defined a degree d polynomial to be a polynomial of the form: $f(x) = a_d x^d + a_{d-1} x^{d-1} ... + ax + a_0$, where d is the degree of the function and $a_d ... a_0 \epsilon \mathbb{C}$.

2.2 Shamir SSS

We then applied the Linear Algebra Base to the concept of a secure t-out-of-n SSS. The Shamir SSS works by generating a polynomial with degree $t-1$ for the Share function and returns n possible evaluations of the function as shares. The Reconstruct function then must have at least t shares in order to uniquely determine the function used to create the shares, and thus can be used to find the message m . In more formal terms, the Shamir SSS has three components: library variables, the share function, and the reconstruct function.

2.2.1 Variables

The Shamir SSS has 3 library variables. They are; p which is a randomly selected prime number; $\mathbb{Z}_p$, which is the set of all integers less than p; and $n$which is the number of shares the Share function will split the message $m \epsilon \mathbb{Z}_p$.

2.2.2 Share Function

The Share Function takes an input of the message m and performs the following steps:

1. Select coefficents randomly where $a_1 ... a_{t-1} \epsilon \mathbb{Z}_p$
2. Define degree $t-1$ polynomial as: $f(x) = a_{t-1} x^{t-1} + a_{t-2} x^{t-2} + ... + a_1 x + m$
3. Generate shares s_i where $s_i = (i, f(i) mod p)$ for $i = 1...n$
4. Return $s = \{s_1, s_2, s_3, ...\}$

2.2.3 Reconstruct Function

The Reconstruct function takes as input a set of user ids $\mathcal{U}$ and performs the following steps:

1. if(size of $\mathcal{U} < t$) return ERROR
2. Take t shares (ids from $\mathcal{U}$)
3. Reconstruct the polynomial $f(x)$
4. Return $f(0)$

3 Indistinguishability

The concept of indistinguishability is based in the idea that a model can be deemed secure if the computation time required to break the model is sufficiently lengthy. In other words, in this section we move away from the idea that to prove a model's security, you have to prove that it is impossible to break it. Rather, you can simply prove if it is computationally infeasible to break the model instead.

3.1 Proof of Security

We then went through the method of proving the security of the Shamir SSS. We used probability distribution analysis to show $\mathcal{L}_{\text{SSS-Real}} \equiv \mathcal{L}_{\text{SSS-Rand}}$,which are defined below.

3.1.1 $\mathcal{L}_{\text{SSS-Real}}$

$\mathcal{L}_{\text{SSS-Real}}$ is a library containing an interface Query$(m, t, \mathcal{U})$, where m is the message, t is the number of shares, and $\mathcal{U}$ is the set of user ids. Query performs the following:

1. if $|\mathcal{U}| \geq t$ return ERROR
2. Perform Share Function
3. Return $\{s_i | i \epsilon \mathcal{U}\}$

3.1.2 $\mathcal{L}_{\text{SSS-Rand}}$

$\mathcal{L}_{\text{SSS-Rand}}$ contains the same interface as $\mathcal{L}_{\text{SSS-Real}}$ except the Query function performs the following:

1. if $|\mathcal{U}| \geq t$ return ERROR
2. Sample $y_i \leftarrow \mathbb{Z}_{\text{p}}$ for $i \epsilon \mathcal{U}$
3. $s_i = (i, y_i)$ for $i \epsilon \mathcal{U}$
4. return $\{s_i | i \epsilon \mathcal{U}\}$

3.1.3 Proof of Interchangeability

$\mathcal{L}_{\text{SSS-Rand}}$

1. $| \mathcal{U} | = t - 1$
2. Pick any $m \epsilon \mathbb{Z}_{\text{p}}$
3. Pick any set of $t - 1$ ids
4. $\mathcal{L}_{\text{SSS-Rand}}$'s output would then be $\{(i, y_i) \mid i \epsilon \mathcal{U}\}$

The probability of any given output of $\mathcal{L}_{\text{SSS-Rand}}$ would be $\frac{1}{p^{t-1}}$, because there are $t - 1$ possibilites in which we can select up to p numbers. Similarly, $\mathcal{L}_{\text{SSS-Real}}$ would also have the same probability because it is selecting up to $p$values for $t - 1$ possibilities for reconstructing the function.

3.2 Polynomial Time Computation

The general premise behind polynomial time computation is that no adversary would be able to run a sufficiently large exponentional function due to physical limitations. Thus we can say that an algorithm is secure if it is computationally infeasible to break it, even if the algorithm isn't perfectly secure in principle. Algorithms such as RSA depend on this type of security. We define polynomial computation to be of the following form: Given a sufficiently large input x, there exists some constant c greater than zero such that any program ends in no more than O($| x |^c$) steps.

3.2.1 Security Parameters

The secruity parameter λ measures the random space of the algorithm in question. When λ increases, the algorithm's computation increases as a polynomial function.

3.2.2 Negligible Functions

Negligible functions are our way of determining where the line is between a computation that is too slow and a computation that is practically impossible to guess. We will use negligible functions to determine this. We will define a function as negligible if for any polynomial p, we have $\lim_{\lambda \to \infty} p(\lambda) f(\lambda) = 0$. We then defined for $f \approx g$ for two functions f and g if $| f(\lambda) - g(\lambda) |$ is negligible.

3.3 Indistinguishablity (formal definition)

We can now define indistinguisable in mathematical terms. Let $\mathcal{L}_{\text{left}}$ and $\mathcal{L}_{\text{right}}$ be two libraries with a common interface. We say $\mathcal{L}_{\text{left}}$ is indistinguishable from $\mathcal{L}_{\text{right}}$ if $\Pr(A \cdot \mathcal{L}_{\text{left}} = 1) \approx \Pr(A \cdot \mathcal{L}_{\text{right}} = 1)$. We then went on in class to use this definition to prove multiple lemmas for sample libraries.