Mini Project 2 - Security 1

Adam Hadou Temsamani (ahad)

October 31, 2023

0.1 Solution

The solution can be found in the root folder at *main.go* file. The grpc interface can be found in the subfolder */proto/proto.proto*. The certificates for running TLS can be found in the subfolder */certificates*.

0.2 Adversary Model

I am assuming an insecure network, with a group of mutually distrustful parties. But with a honest majority. We assume the adversary to a *Doliv-Yao* attacker. Meaning that it is a static and active adversary. Each of the parties need to share a secret value with the hospital without leaking/revealing any information about it.
Due to the nature of the adversary, I assume that they will try to break security by either eavesdropping on the messages in the network, impersonating parties or dropping messages. Which means we somehow needs to guarantee *confidentiality*, *integrity* and *authenticity*. This can be guaranteed with the cryptographic protocol *Transport Layer Security*.
In the event that the adversary can decrypt messages, it is important to make sure the adversary learns nothing about the secrets of the parties. It is also important that in the event of the hospital getting compromised that it itself does not learn anything about the individual secrets of the parties, but only an aggregate value. This can be done through *MPC*, specifically; *n-out-of-n Additive Secret Sharing*.

0.3 Building Blocks

The solution is implemented using *Golang* with the *grpc* library. I am using a *Peer-to-Peer* connection. This allows us to easily implement TLS. We do this by creating self-signed certificates, which the parties will use to communicate over the network. This is done using the *OpenSSL* tool, with the locations of the certificates in the subfolder */certificates*. The implimentation can be found in the *main()* method. It is important to note, since the solution does not use a centralized server for the hospital. Each of the peers (including the hospital) have a specific Id. *Id 0* is reserved for the hospital. While *Id 1-3* are reserved for the three-way communication between Alice, Bob, and Charlie.
When disclosing information with the hospital we need to create an aggregate values, with *n-out-of-n Additive Secret Sharing*. Each of the parties compute three shares, which can be found in *splitShare()*. These value are computed from each of their chosen secrets, which they want to send to the hospital. I have computed each of these secrets using *circular groups*, with the properties of the *modulo* opperator and *prime*-numbers. This means that the *splitShare()* method only supports secrets in the range between *1 to 514229*. Each of these shares are sent the other peers, while keeping the last share for themselves. This can be found in the method *ShareSecret()*. When each of the parties have received three shares (including their own). They sum up the shares and broadcast them to the hopstital. This can be found in *ShareSecret()* and *CombineSharesAndSend()*. After which the hopsital sums up these values to create a final sum. This can be found in the method *Send()*.

0.4 Security Guarantee

When working with a Dolev-Yao adversary it is important that the application ensures the different security guarantees.

Using the aforementioned building blocks and techniques allows us to create a program where each of the communicating parties are authenticated and know who they are communicating with. The *authentication* property is provided through TLS (certificates). While also making sure that the messages they receive and send are encrypted as well as not modified. Providing us with *confidentiality* and *integrity*. Additive Secret Sharing also allows us to keep the secrets of each peer a secret even if the shares do leak, and even if the hospital is compromised only the sum is known. This is included in the property of aggregate sharing, where if less than all shares is known, it computationally impossible to compute the actual value. We provide this through *n-out-of-n* additive secret sharing. The only security guarantee that the program does not provide with a Dolev-Yao adversary is if the adversary decides to drop messages.

0.5 Running Example

To run the application you need to open four terminals in the root folder. Run and execute main.go in each of the terminals. An example command would be: *go run . 0*, with the last number being the id of the peer.
The figures below show the application running. The example shows each peer using "1000" as their secret. This is chosen simply to verify that the shares are random and well computed, as well as the final sum that the hospital produces is correct. It is important to note that when the program is done running the algorithm/protocol once, you need to terminate the program and re-execute the program.

```
PS C:\Users\Adamh\OneDrive\Documents\GitHub\MiniProject1_Security1\Mini_Project_2> go run . 0
Trying to dial: 5051
Trying to dial: 5052
Trying to dial: 5053
Welcome to the service!
You are the hospital. Please wait, while the peers are sending you their secrets.
I have received sum number: 1, which is: -184934.
I have received sum number: 2, which is: 162384.
I have received sum number: 3, which is: 25550.
I have received sums from all peers.
The final sum is 3000.
```

```
PS C:\Users\Adamh\OneDrive\Documents\GitHub\MiniProject1_Security1\Mini_Project_2> go run . 1
Trying to dial: 5050
Trying to dial: 5052
Trying to dial: 5053
Welcome to the service!
To share a secret with the hospital, enter a number between 0 and 500.000
1000
You have chosen the secret: 1000
These are my shares: [92835 33625 -125460]
I am sending (92835) to peer: (5052)
I am sending (33625) to peer: (5053)
I am keeping (-125460) for myself
I am peer (5051) and the share is (-125460)
I am peer (5051) and the share is (455154)
I am peer (5051) and the share is (210085)
I am sending the sum 25550 to the hospital.
```

```
PS C:\Users\Adamh\OneDrive\Documents\GitHub\MiniProject1_Security1\Mini_Project_2> go run . 2
Trying to dial: 5050
Trying to dial: 5051
Trying to dial: 5053
Welcome to the service!
To share a secret with the hospital, enter a number between 0 and 500.000
1000
You have chosen the secret: 1000
These are my shares: [455154 209341 -149266]
I am sending (455154) to peer: (5051)
I am sending (209341) to peer: (5053)
I am keeping (-149266) for myself
I am peer (5052) and the share is (92835)
I am peer (5052) and the share is (-149266)
I am peer (5052) and the share is (218815)
I am sending the sum 162384 to the hospital.
```

```
PS C:\Users\Adamh\OneDrive\Documents\GitHub\MiniProject1_Security1\Mini_Project_2> go run . 3
Trying to dial: 5050
Trying to dial: 5051
Trying to dial: 5052
Welcome to the service!
To share a secret with the hospital, enter a number between 0 and 500.000
1000
You have chosen the secret: 1000
These are my shares: [210085 218815 -427900]
I am sending (210085) to peer: (5051)
I am sending (218815) to peer: (5052)
I am keeping (-427900) for myself
```

Figure 1: Example Program.