Security 1: Mandatory hand in 2

Author: Nedas Surkus

October 2022

Contents

Introduction

In this specific problem. Bob and Alice do not trust each other or the network they are sending data in. Thus, the following questions need to be answered:

(a) How can Bob and Alice ensure that their dice rolls are fair, unbiased and cannot be influenced by either party.

(b) How can Bob and Alice ensure that data they send between each other is not manipulated in any way or form?.

Protocols

Fair dice roll protocol

To answer the first question, we must establish a way that neither Alice nor Bob can't influence the roll. For this we can take an inspiration from the online gambling scene.

Most modern gambling sites use one of two algorithm types to calculate their rolls. "Black box" algorithms which are algorithms that don't reveal any information on the algorithm used and "Provably Fair Gaming Algorithms". There are many different Provably Fair algorithms but, in this assignment, we will use the most basic version which utilizes hashing to create a pseudo random dice roll.

The most basic version follows these steps:

1. Alice will pick some "random" string. From this point we will refer to this string as "Alices secret string".
2. Alice then will hash this string using a SHA256 hash. From this point we will refer to this as a commitment.
3. Alice will then send this commitment to Bob.
4. Bob will receive the commitment.
5. Bob will pick some random string B. This string will be generated in a similar way that Alice generated hers.
6. Bob will then send string B to Alice without encrypting.
7. Once Alice retrieves string B from Bob. She will combine string B with her own secret string and then hash it using SHA512 hash.
8. Alice will send SHA512 hash to Bob and she will also include her secret key.
9. Once Bob gets the SHA512 hash both Alice and Bob will calculate the roll.
10. To get the roll. Bob and Alice will take the first 5 characters of SHA512 hash and convert them from hex to integers.
11. If this integer is bigger than one million or smaller than 0, Bob and Alice will take the next 5 characters from the hash.Otherwise they will divide this integer by 10000 and take the reminder. They will divide this reminder by 1000 and round up the results.
12. If the result is bigger than 6 or smaller or equals to 0, Bob and Alice will take the next 5 characters in the hash and repeat Steps 10-12. If it is between 1 and 6, this will be the roll result.

13. Bob and Alice can confirm that the roll is unbiased by hashing Alice's secret string to SHA256 and combining Bobs string B and Alice's secret string and hashing it with SHA512. If both Bobs generated hashes and the hashes Alice sent are the same. Then Alice has not interfered with the dice and the result.

Nuances:

(a) The string which Alice and Bob picks need to have both upper- and lower-case characters and numbers. This random string can be any size, but to prevent Bob from brute-forcing the string. Alice will use at least a 10-character string.

(b) Alice cannot interfere with the results of the dice roll because she must first send the commitment to Bob. If Alice changes her string, the SHA256 encryption will not match the commitment she sent to Bob.

(c) Bob cannot interfere with the dice roll. Because to calculate the SHA512 hash, he needs to know Alice's secret key. To brute force this secret key would take an long time if Alice picked a big enough string.

(d) Utilizing 2 different hashing methods allows the dice roll to be more randomized and unpredictable.

(e) Lastly to reduce the chance that either Bob or Alice would try to cheat. Bob and Alice should agree on a maximum response time, 3 seconds would be long enough to account for any delays in internet connection but would be too small of a time to try out every combination of hashing to get a the keys used to generate the hashes.

In general we can reasonably that by utilising this protocol. The dice roll will be unbiased and unpredictable to both parties.

Prevent external attacks

Signatures

However, the biggest threat comes from external actors trying to influence Alice or Bobs rolls.

This attacker C can attempt to either modify commitment Alice sends, the string Bob sends to Alice or the SHA512 hash and secret Alice sends to Bob. Thus, potentially influencing the roll results.

While this attempt at modification would be detected quickly by verifying if results match the hashes, it does not prevent from attack C hijacking Alice's connection and impersonating her. Thus, it is crucial that we could verify that the content of the message is not modified somehow, and that the sender is Alice.

Additionally, the attacked might guess they are playing a dice roll game which would give further hints on how attacker C could influence the game. There are 2 ways that Alice and Bob can ensure that they are playing against each other and not some third party (ensure Authenticity) and ensure that the messages they send to each other are not modified (ensure Integrity). For the first way we can take inspiration from one way message authentication and for each message, create an unique signature for each message.

Note: For this specific example, we assume that Both Alice and Bob have agreed on a secret key or have exchanged a secret key. This secret key will be "SecretK" for simplicity's sake.

Thus, the following protocol was designed:

1. At the start of the round. Alice and Bob will generate an SHA256 key by combining SecretK and Salt (in this scenario - round number). From this point, we will refer to this key as "authentication key".

2. Once Alice generates a SHA256 hash using her secret string, she will create a Hmac (Hash-based message authentication code). This Hmac will be generated with SHA256, authentication key and the message contents.
3. Once Alice has the Hmac she will attach it to the message she will send to Bob.
4. Once Bob receives the message, he can see that the message contents have not been tempered with by creating another Hmac with SHA256 encryption, the shared authentication key, and the message he received from Alice.
5. The Hmac should match the one provided by Alice. If they do not, Bob knows that Alice's message has been tampered with and should close the connection.
6. Repeat this for each message Bob or Alice will send this round.
7. Once the round finishes, Bob and Alice should increase the round counter by 1 and repeat this process for as long as they want to play the game.

Why is this secure?

For attacker C to successfully impersonate either Alice or Bob he must know what salt they are using and he must know the secret Key Alice and Bob agreed on. If he knows both, he can generate his own Hmac and impersonate both Alice and Bob.

However, each round Alice and Bob would hash a new and unique Authentication key, this means that the Attacker C has a very short window to brute force the signature in order to generate Bobs or Alice's Hmac. However, even if he does brute-force the Hmac. He will only find out a SHA256 authentication key which will change each round. Further limiting his ability to successfully impersonate either.

Additionally, Bob and Alice both know what kind of authentication key will be generated, as they use the same encryption, the same string and same salt to generate the authentication key. Meaning both Bob and Alice generate the same authentication key. This means that in the case Attacker C tries to high jack Alice's connection and pretend he is Alice. Bob will know its not Alice as attacker C will not know the authentication key used to generate the Hmac. Thus, easily verifying Alice's identity.

In case the attacker C modifies either the signature or the message contents. Bob and Alice can easily verify if the signatures match.

Thus, we can assume that utilizing one way message authentication can help prevent external attackers from compromising the integrity of Bobs and Alices messages and allowing Bob and Alice to quickly confirm authenticity of the message sender.

However, it has to be noted that this protocol can only help upkeep integrity and authenticity, but not confidentiality. Additionally, it can be quite exhausting and not very practical in encrypting each message with a unique signature.

TLS

Another and a more secure way Alice and Bob can ensure that attacker C would not interfere with their game is by utilising TLS (Transport Layer Security) and switching their connection from HTTP to HTTPS. To do this, both Alice and Bob will generate a secret key and a certificate.

As the game should be private between Bob and Alice only. A digital certificate will be generated and will be used to sign both Alice's and Bob's certificates. In this assignment we will assume that both Alice and Bob have this CA certificate but not attacker C.

As Bob and Alice will try to connect with each other. Bob will start a digital handshake. He will send a connection request to Alice. Alice will then engage in a handshake and exchange secret keys and certificates with Bob and allow him to send messages to her.

Why is this secure?

Both Alice's and Bobs certificates are signed using a common shared DA certificate. Thus is very easy to verify if Bob and Alice are who they claim to be as only they could sign their certificates using this common shared DA certificate.

TLS encrypts data between Alice and Bob so attacker C does not know the contents of the messages being sent.

Lastly Attacker C cannot influence the data in any form or way as his attempt to connect to the server would be rejected as his certificate would not be signed by the shared DA.

Thus, we can be certain that TLS would provide Alice and Bob their desired Confidentiality, Authenticity and Data integrity from outside attackers.

However, we must take into account that there is a slight chance that attacker C might somehow acquire the CA certificate and use that to imitate that he is Alice and Bob. To counter this, Alice and Bob would still have the signatures to verify that both are who they claim to be.

Lastly, if anything out of the ordinary happens. (A certificate is expired, a certificate is not signed by shared DA or Bob and Alice would not provide the agreed on signature each round.) The connection will be terminated between Alice and Bob and the game ends.

Thus with all these systems and protocols in place, we can reasonably assume that that it would be very hard for any attacker C to influence and spy on Alice's and Bobs game.

Code

Code Nuances

To illustrate both protocols in action. Two NodeJS programs were created. One representing Alice, the other representing Bob.

Alice and Bob both are communicating using sockets to illustrate them communicating over some network that anyone can listen to.

To establish a TLS connection Both Alice and Bob have generated their own keys and will utilise a shared CA certificate. Anyone that does not provide a certificated signed by this shared CA certificate, will be automatically rejected and disconnected.

Alice and Bob will play 6 rounds. In other words, they will roll the dice 6 times. 3 dice rolls will be performed by Bob and 3 by Alice.

Both Alice and Bob will contain a hard-coded Hmac secret "SecretK" and will contain mostly identical methods with some exceptions.

Bob has 1/9 chance each round to become "compromised" and if he is. He has a 50

How to run the code

Ensure that you got NodeJS v16.17.1 installed.

Navigate to the Bobs folder and in the terminal write: node app.js

Navigate to Alice's folder and in the terminal write: node app.js

Both instances need to be running at the same time to play the game.