Ro Sham Bo

HW6 - CNS Sapienza

Pietro Spadaccino 1706250

7 December 2018

1 Introduction

We will discuss about a secure protocol enabling two parties Alice and Bob to play rock paper scissors' game without a centralized trusted authority.

2 Technologies

Before describing the protocol, we show the utilized technologies and the reason why we used them:

- Public key cryptography. We will make use of it only for granting non-repudiation on sent messages, like a signature. We denote by $E_A(msg)$ and $E_B(msg)$ the content msg encrypted by means of the private key of Alice and Bob;
- Hashing, used as a one-way function hard to invert. We denote the hash of a content msg by $H(msg)$;
- Nonces, used to scramble sent hashes, denoted by n.

3 Protocol

We now describe how the protocol works, reported also in Figure 1. When it starts, both Alice and Bob have done their move $a, b \in \{P, R, S\}$. The protocol to perform a turn it's made by few different messages, divided into 4 phases:

1. Alice and Bob exchange $E_A(H(a||n_A))$ and $E_B(H(b||n_B))$, which is the hash of the performed move encrypted by the private key of the sender. Since the message is hashed, no party can retrieve the move of the other one. This hash represents indeed the proof that an action has been done, but the adversary doesn't know yet which one. Additionally, each party adds a nonce n_A and n_B, that must be different each time this protocol is executed, as explained in section 3.4.

2. Alice sends the content of the previous message received from Bob encrypted by its own private key $E_A(H(b, n_B))$, and Bob does the same $E_B(H(a, n_A))$. By checking the message of the adversary, each party is sure that the other received what was sent during the first phase. This step enforces data integrity and it is explained in section 3.5.

3. Alice and Bob exchange a, n_A and b, n_B in plaintext, which is the move performed alongside the nonce used in the phase before. By now, each party can verify if the received move is the real one performed: if the hash constructed using the move and the nonce is equal to the one received before then the move is valid, otherwise the party is cheating.

4. Once the parties have agreed on the move, they can exchange the score of the game. The score is represented by a tuple containing the number of games won by Alice and the ones won by Bob. If no one arrived to $p + 1$ then another game is started.

3.1 Performances

The protocol requirements are very light and can be run on devices with limited computation resources. The most complex operation is to generate a public/private key pair if the parties don't have one, but this has to be done only once. Moreover, it requires the execution of some hashing functions, which are not very demanding, and the generation of some random nonces, which is easy.

3.2 Impossibility to change action

When we play the game rock player scissors non-virtually, a player wants to choose a move without making the other aware of which move he has taken. When both players have selected a move, they can show it at the same time,

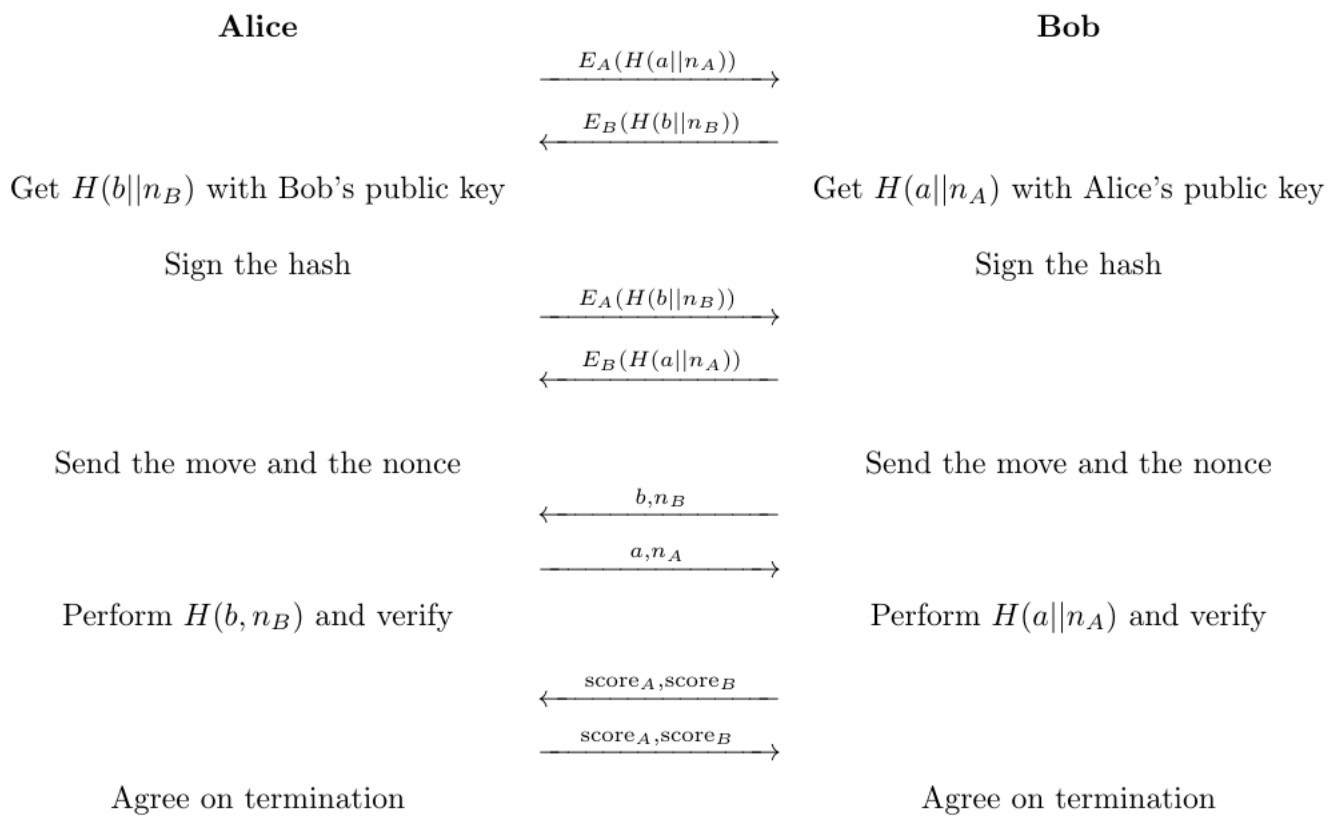
Alice
Bob
$E_A(H(a||n_A))$
$E_B(H(b||n_B))$
Get $H(b||n_B)$ with Bob's public key
Get $H(a||n_A)$ with Alice's public key
Sign the hash
Sign the hash
$E_A(H(b||n_B))$
$E_B(H(a||n_A))$
Send the move and the nonce
Send the move and the nonce
b, n_B
a, n_A
Perform $H(b, n_B)$ and verify
Perform $H(a||n_A)$ and verify
$\text{score}_A, \text{score}_B$
$\text{score}_A, \text{score}_B$
Agree on termination
Agree on termination

Figure 1: Message flow of the two parties. We note that Bob is the first to send b, n_B because he's the first to have received the hash of the move. This overcomes replay attack, as discussed in section 3.3.

agreeing on the termination and on the result of the game. Indeed, a player must not be able to change its decision after looking at the adversary move.

Our protocol does so by exchanging the hash of the action in the first phase and then later showing those actions. In this way, an adversary can't retrieve the move of the other by looking only at the hash, but when the moves are shown publicly both can check if the hashed value of the action corresponds to the hash received before. As an example, let's consider the step of the protocol where the moves are exchanged between the parties (the hashes of the moves were already exchanged). Bob receives Alice's move R and he may not have sent yet his move S to Alice. Bob now knows that he lost the game and he can try to send move P to Alice. When the move is received, Alice performs the hash to check whether or not it corresponds to the action declared. In this case the hash doesn't match and Bob is caught cheating. But Bob, since he's smart, can claim that Alice is lying: the verification was positive but Alice cheats and refuses to accept Bob's move. Also in this case Bob fails, since Alice possesses the hash of Bob's move encrypted by means of Bob's private key, therefore Alice can prove that the hash is original and Bob can't repudiate it.

The last chance of Bob is to send its new move and trying to trick the verification done by comparing the hashes: in order to send P to Alice and to not get caught, he must find a collision such that $H(S, n_1) = H(P, n_2)$, where n_1, n_2 are nonces. We know that by using secure hashing functions, like SHA256, this is almost impossible to achieve given any computational power existing today.

3.3 Replay Attack

We now analyze the possibility for a malicious party to perform a replay attack. We remind that after the first phase, the hashes of the performed actions are exchanged. A party cannot know which move has performed the adversary, but knows for sure that if he replies with the same hash received, the game must end in a tie. Bob may follow this scheme to perform a replay attack:

1. Alice sends $H(a, n_A)$ to Bob;
2. Bob sends $H(a, n_A)$ to Alice;
3. Alice sends a, n_A to Bob;
4. Bob sends a, n_A to Alice;

5. Both parties positively perform the check on the hash of the move and the nonce and, since the move is the same, the game is tie.

While this behavior is suspicious, it may be admissible by the protocol. To overcome this difficulty, we can force Bob to be the first to send the move and the nonce after he received the hash from Alice. In this case we would have:

1. Alice sends $H(a, n_A)$ to Bob;
2. Bob sends $H(a, n_A)$ to Alice;
3. Bob must send a, n_A to Alice, but since he doesn't know them and he can't bruteforce, his attack fails.

3.4 Nonces

By using nonces, it is not possible to make an "inverse" replay attack. For "inverse" replay attack we mean a situation where a malicious party can analyze different messages of the adversary and retrieve his move before sending its own move. If we didn't use nonces, meaning that the hash is composed only by the move and, let's say, some other secret not variable during time $H(move||secret)$, then there can be only 3 different hashes, since there are only 3 possible moves. A malicious party can then collect these hashes with a few games and understand the action selected by the player from the hash.

3.5 Agreement and termination

If a malicious party decides to cheat by changing the move, he is caught by the adversary because in case of a cheat the hashing verification yields a negative result as discussed earlier. But a party can also cheat in the last phase, when the twos agree on the termination of the current game exchanging the result of the overall match. Indeed, a party can change the score of the match adding points to himself and then sending it to the adversary. To overcome this possibility, each player must save the encrypted hashes sent by the adversary, so that if a player decides to cheat the other can prove that he's actually cheating, and the malicious party cannot repudiate it. Let's make an example: Alice is malicious and claims to have some points more over Bob. But Bob had saved Alice's hashes of the moves encrypted by her private key. This is a proof that Alice performed that moves and she cannot repudiate them.

But what would happen if Alice claims that Bob is cheating? In other words, while Bob can prove by non-repudiation that the moves performed by Alice were indeed performed by her, how can he prove what were the moves performed by him? This is where the second phase of the protocol comes in help, the one where the content of the first phase get signed and exchanged: Bob can prove his own moves because during this phase he received from Alice $E_A(H(b, n_B))$, therefore Bob is sure that Alice received his moves and she can't repudiate the message since it's encrypted by means of her private key. Therefore Bob can prove all the moves of the match and its overall score, and the parties can agree on the termination.