A crypto protocol for the Rock Paper Scissors game

HW6 - CNS Sapienza

Andrea Fioraldi 1692419

2018-12-7

1 The Problem

Alice and Bob want to play Rock Paper Scissors game using their smartphones. We must design a protocol in order to ensure a fair game. Our protocol must be secure and it must not permit the cheating. A match is composed of N games where N is an odd number. The game is live, there is no time for a brute force attack.

2 Our Solution

Our solution is based on the SHA256 hashing algorithm. We use the hash extension feature of the hashing algoriths based on MerkleDamgard. The method is the following: Given $hash(message_1)$ and the length of $message_1$ we can calculate $hash(message_1||message_2)$[1] where $message_2$ is an arbitrary string of our choice. The procedure that can be applied for the computation of such hash is well described in this blogpost [1].

This feature is often considered a security issue in fact SHA3 dropped it. In our specific problem, however, it can be used to design a very simple protocol that ensures data integrity and avoid cheating.

We encode the moves of the game using a 2-bits number:

- Rock: $R = 00$;
- Paper: $P = 01$;

[1] $||$ is the concatenation operation

- Scissors: $S = 10$;

This is the protocol for a game:

1. Alice generates a 32-bit random nonce;
2. Alice choose a move $M_A \in \{R, P, S\}$ and in parallel Bob choose $M_B \in \{R, P, S\}$;
3. Alice compute $H_1 = SHA256(nonce||M_A)$;
4. Alice sends H_1 to Bob;
5. Bob waits for H_1 and then compute $H_2 = extension(H_1, M_B)$ which is equivalent to $SHA256(nonce||M_A||M_B)$ but Bob does not know the nonce and neither M_A;
6. Bob sends H_2 to Alice.
7. When Alice receives H_2 she sends the nonce to Bob.
8. Alice compute 3 hashes $SHA256(nonce||M_A||probe)\ \forall\, probe \in \{R, P, S\}$;
9. Alice checks if one of the hashes is H_2, if yes now she knows M_B, if not Bob inserted an invalid move and so stop the game.
10. Alice updates its local record of winners with the winner of this game.
11. Bob compute 3 hashes $SHA256(nonce||probe||M_B)\ \forall\, probe \in \{R, P, S\}$;
12. Bob checks if one of the hashes is H_2, if yes now he knows M_A, if not Alice inserted an invalid move or lied about the nonce and so stop the game.
13. Bob updates its local record of winners with the winner of this game.

This method works when a player does not trust the other player. The nonce avoids the replay attacks. If someone cheats the other player will know. Alice cannot send a wrong hash at step 4 otherwise Bob will compute 3 wrong hashes at step 11 revealing the cheat. Bob cannot cheat at step 6 otherwise Alice will compute 3 wrong hashes at step 8 revealing the cheat. Inserting a fake result is useless, a player cannot change the internal record of the other.

In a game the exchanged messages are only three and the total hashes computed are 8, 4 for each player.

In order to ensure data integrity, we can insert an CRC for each message.

With this protocol each user compute the score locally without the possibility to affect the score computed by the other. So at the end all the users have the same scoreboard stored locally and know who is the winner.

If for some reason a comparison of the two scoreboard is needed at the end of the match, in order to avoid that the loser will declare an invalid scoreboard to invalidate the match, we can sign every exchanged message using a digital signature to ensure the not repudiation of the results.

So an user can show that his scoreboard is valid simply showing that all the messages are original.

References

[1] *Everything you need to know about hash length extension attacks SkullSecurity.*

`https://blog.skullsecurity.org/2012/everything-you-need-to-know-about-hash-length-extension-attacks`

Accessed: 2018-12-7.