MOVA signature project: University Competition Application v2

Sebastien Duc

1 Overview

We consider an application where several universities are participating to a tournament. We can imagine that for the first round of the tournament, universities are organized into pairs. Let's consider one pair, and let's denote the two universities by U_1 and U_2. U_1 and U_2 are in competition. In this application, U_1 provides quizzes/challenges/riddles to students in U_2. Similarily U_2 provides quizzes to students in U_1. Students in a university are forming teams. We assume that teams have a certain amount of time (for example one week) to complete the current quiz/challenge/riddle. Each team in a university have a different quiz[1]. When completed they send it to the server of both universities and get a score. Then the university who provided the quiz also provides the solution of the quiz and the score the team did, to the students. At the end of the first round, all the best of the two universities (i.e. the ones with the highest cumulated score of all teams in the university) in the pairs are promoted to the second round. The game goes on the same way until round n with only on pair (only two universities) and the best of the two wins the game.

2 Security

For this part let us consider one pair with universities U_1 and U_2 as before. Let us denote the server of U_1 by S_1 and the server of U_2 by S_2. We can assume that communication between S_1 and S_2 are confidential, autenticated and preserve integrity. When U_1 creates a quiz/challenge for students of U_2 they sign it with MOVA and then send it to them. In that way, the students are ensured that the quiz comes indeed from U_1 and that it is not a fake

[1]If several teams in a university have the same quiz, then they can copy on each other to increase the final score.

quiz from other cheating students who wants to make them lose time, hence lose score. Furthermore, only the designated team can verify the signature. Then, when the students are sending back the quiz to S_2, S_2 can sign the filled quiz with MOVA and send it to back to the team. The team can verify that the signature is correct[2], and send it to S_1. Finally S_1 receives the filled quiz and verify both signatures. For signatures coming from S_2, S_1 can run a batch verification with S_2. Then when S_1 provides the solutions with score the the team, he signs everything and send. Therefore, students are ensured that the solution and the score are correct.

3 Problems

As opposed to the previous version of the application, we don't need to keep authentication and integrity between the students and their university server, because the server send the signed filled quiz back to the team and the team can verify the signature. In the previous version we had to assume that S_2 did not modify the results of the quizzes before signing them. This is not an issue anymore, because the students receive their signed filled quiz back.

4 Using classical Signatures

In this application we could try to use classical signature. For the signature of the unfilled quiz (the signature signed by S_1, when S_1 is the quiz provider), MOVA could bring something compared to classical signatures, because

- We use that MOVA is not universally verifiable, because only the designated team must verify the signature.
- But still, we cannot use batch verification
- But still, it is not really useful for the signatures to be very short

For the signature of the filled quiz (signed by S_2, when S_1 is the quiz provider), we exploit the fact that S_1 runs batch verification.

5 Conclusion

To conclude, it seems that MOVA could bring something for this application, but only in a weird threat model.

[2]In that model, we consider that the students don't trust the server and want to be sure that answer were not changed or deleted.