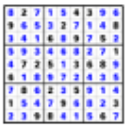
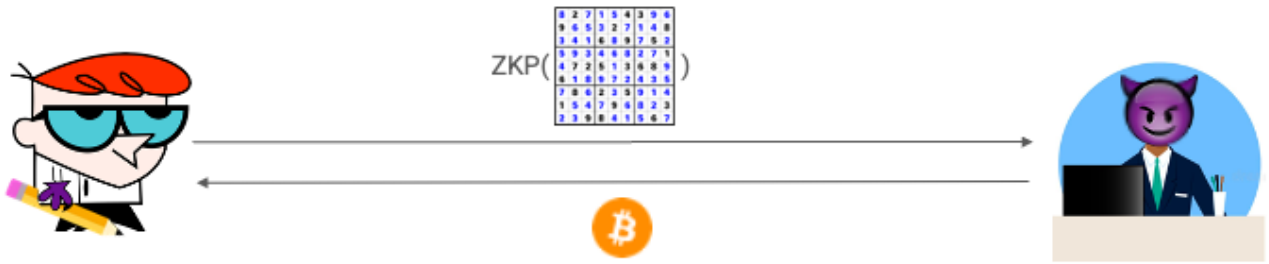
Live Exercises

Privacy-Preserving Authentication

You are a professional sudoku solver. A company needs a sudoku to be solved, and offers a reward of 1 BTC. After hours of continuous effort, you finally solve it. You now want to claim your prize. But… you don't trust the company, and think that they might "steal" the solution or refuse to pay. You decide to use **Zero-Knowledge Proofs**.

ZKP()

You are a professional sudoku solver. A company needs a sudoku to be solved, and offers a reward of 1 BTC. After hours of continuous effort, you finally solve it. You now want to claim your prize. But... you don't trust the company, and think that they might "steal" the solution or refuse to pay. You decide to use **Zero-Knowledge Proofs**.



What happens if one of those properties is not provided?

1. Completeness
2. Soundness
3. Zero-Knowledge

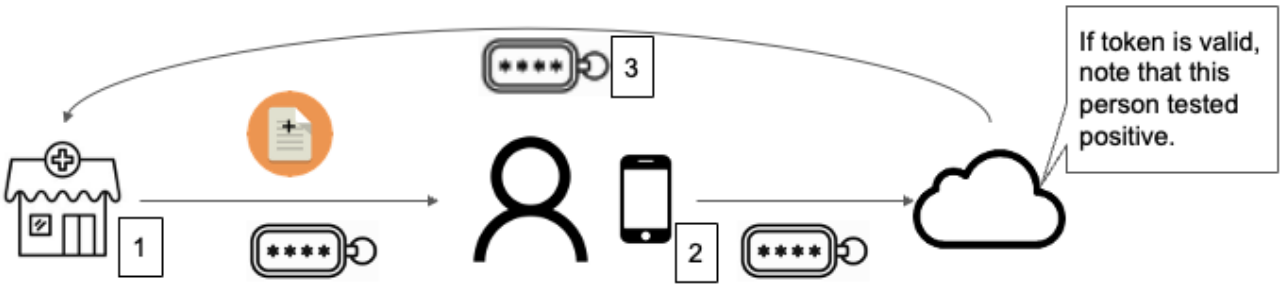
no completeness: you cannot convince the company even if you have a correct solution
no soundness: you can fool the company even though you do not have a correct solution
not zero-knowledge: the company learns the solution and does not pay you

Digital proximity tracing systems such as SwissCovid use smartphone apps to warn people that they have been close to someone that tested positive for COVID-19. To enable these warnings, the app of a positive person must upload data to a central server. **For the security of the system it is important that updates can only be done by people that tested positive.** Therefore, SwissCovid uses an authorization mechanism that works like this (see next slide for figure):

1. Together with a positive test result, the user also receives from the testing center a one-time token generated by the central upload server.

2. The user enters this token into the app, and the app uploads the relevant data together with the token to the central upload server.

3. The server checks that the token is valid and was not used before. If both checks pass, it accepts the uploaded data. The token proves that the user was authorized to make an upload (i.e., because the user tested positive).



The current system already has many protections in place to guarantee anonymity of positive users. However, you realize that **the upload server together with the testing center could determine which users uploaded data, and which did not.** Indeed, the testing center knows which users received which tokens, and the server knows which tokens were used to upload data.

You decide to see if you can design a protocol that provides better privacy for positive users. For this exercise, we say that everybody that got tested, can do an upload. (**This is not a good idea in practice.**) You envision the following high-level solution based on attribute-based credentials (ABCs):

1. Every user that gets tested, receives a one-time-use token generated by the upload server.

2. Immediately after the test, users enter this token into their app. The app sends the token to the server to prove that the user received a test. The server then issues an ABC to the app.

3. If the user later receives a positive test result, they click the upload button in the app. Thereafter, the app sends the relevant data and a proof of having a credential back to the server. After performing some checks, the server accepts the uploaded data.

Part 1. What security and privacy properties do you want?

Inspired by 2021 Midterm (but there we wrote down the properties ;))

You want to ensure two properties:

Deniability. The server cannot determine whether a user with token X uploaded their data or not.

One-upload per test. No one should be able to make more than one upload for each test they take.

As stated, the system provides anonymity (think that users use an anonymous communication system (you will learn more about those later in the semester) so that the server cannot distinguish between users based on network traffic metadata).

Part 2. How would you use ABCs in your protocol to ensure these properties? What data, if any, would the user and/or server encode into the attributes? What does the server check upon receiving the upload in step 3?

Describe your solution. You don't have to give mathematical details such as zero-knowledge proofs, of the associated ABC scheme. Your description should include who provides which attributes in step 2, and what is disclosed in step 3.

The attributes can be designed as follow:
- no attribute and a show protocol indicates a positive result;
- something random (but unlinkable);
- status attribute (then it is always positive, so this is kind of useless)

The server needs to perform only one check which is uniqueness check.

Following the approaches described aboves, what is disclosed is either nothing or a random value.

Part 3. Which type of ABC would you use? What should your ABC scheme satisfy?

- ☐ Unforgeability
- ☐ Selective disclosure
- ☐ Issuer unlinkability
- ☐ Verifier unlinkability

Argue based on this that your proposal provides your chosen properties.

We want to use a blind signature without attribute (or again, random but unlinkable) and with a uniqueness check.

Unforgeability is needed, but needs to be combined with the uniqueness check to ensure the one-upload-per-test.
We need issuer-unlinkability to ensure that the server cannot recognize in step 3 any credentials that it issued in step 2.
If no attribute: no need for selective disclosure.
A credential is used at most once, so verifier unlinkability is not needed.