COMP 4901W – Final Project

A. K. Goharshady

Release Date: April 21, 2022
Deadline: May 11, 2022 (23:59 HKT)

This project accounts for 30% of your total grade. You should submit your solution as two files, one `pdf` and one `sol`. As usual, handwritten and scanned solutions will not be accepted since the TAs might be unable to read your handwriting. However, you can draw your figures, if any, by hand. Unlike the homeworks, you are **NOT** allowed to discuss the problems with your classmates. You have to come up with ideas and write your solution and code entirely on your own. All submissions will go through a strict plagiarism check. If we detect plagiarism, both sides will receive a grade of F for the whole course. As mentioned in the course announcements, we have a zero-tolerance policy and several students have already been disqualified due to this policy and received an F. The deadline is firm and no extensions can be granted. No resubmissions are allowed for the final project.

Background

Consider one of the early scenarios we talked about in the course: A bank has two branches, one in Paris and one in Berlin. A customer deposits money in Paris and wants to receive it in Berlin. However, suppose that our customer has new requirements and she wants to do this with complete privacy. Specifically, she would like to follow a procedure that consists of the following steps:

1. The customer deposits money in the Paris branch. Let's assume that the deposited amount is always a multiple of 100 Euros.
2. For every 100 Euros of deposit, the Paris branch issues a "certificate of deposit" (CD) and gives it to the customer.
3. Anyone who presents a valid CD to the Berlin branch will be paid 100 Euros.

Additionally, we would like to have the following properties:

- *Privacy:* The customer can give the CD to anyone else in order to withdraw the funds in Berlin on her behalf. Upon seeing the CD, the Berlin branch should be able to verify that it was issued by the Paris branch. However, the two branches should not be able to find the identity of the original depositor, even if they collaborate and share their information.
- *No double-spending:* No one should be able to cash the same CD twice.

In 1983, David Chaum introduced the concept of a blind signature, which can be used to solve this problem. The idea is that the Paris branch should sign a CD without actually seeing it! This might sound odd, but here is how it works:

1. The Paris branch uses RSA signatures. So, it announces a modulus N and a public key e for everyone to see, while keeping a secret key d to itself. The branch periodically changes its keys, e.g. it might use a new key pair and modulus every week.
2. To deposit 100 Euros, the customer first creates a bill. A bill is simply a string $\mathbf{s}$ of the form $\mathbf{s}$ = "This is a bill issued by the Paris branch with serial number XXXXXXXXXX", in which the X's are replaced by a 10-digit alphanumeric serial number, chosen randomly by the customer herself. She then computes the hash of $\mathbf{s}$. Let's call this hash $h(\mathbf{s})$.
3. Anyone who can provide a valid bill $\mathbf{s}$ and a signature from the Paris branch on $h(\mathbf{s})$ will be paid 100 Euros by the Berlin branch. In other words, the pair $(\mathbf{s}, h(\mathbf{s})^d \mod N)$ is the CD. Moreover, the Berlin branch keeps track of the serial numbers that are already cashed and will not cash the same bill twice.
4. The customer should somehow obtain the Paris branch's signature on $h(\mathbf{s})$ without actually revealing $\mathbf{s}$ or $h(\mathbf{s})$ to the Paris branch. To do so, the customer first chooses a random number r such that $\gcd(r, N) = 1$ and then computes $h' = h(\mathbf{s}) \cdot r^e \mod N$. She then pays the Paris branch 100 Euros and asks them to sign h'. The Paris branch would sign any hash provided by the customer as long as they are paid 100 Euros. In other words, they charge 100 Euros per signature, but do not care what it is that they are signing.
5. The customer now has a signature on $h' = h(\mathbf{s}) \cdot r^e$, but she needs a signature on $h(\mathbf{s})$. However, note that a signature on h' is of the form

$$h'^d = h(\mathbf{s})^d \cdot r^{e \cdot d} = h(\mathbf{s})^d \cdot r \mod N.$$

So, the customer can simply multiply the signature by r^{-1} and obtain $h(\mathbf{s})^d \mod N$, which is a valid signature on $h(\mathbf{s})$.

6. The customer or a proxy provides the CD $(\mathbf{s}, h(\mathbf{s})^d \mod N)$ to the Berlin branch. The Berlin branch verifies the signature and pays the money.

Note that the procedure above protects the customer's privacy. The bank has no way to tell which withdrawal corresponded to which deposit, since they do not know r. More specifically, all withdrawals that are done using signatures created by the key of the current week are indistinguishable from the bank's point-of-view. On the other hand, there can be no double-spending as long as the function h is a cryptographic hash function.

After this long introduction of the technique of blind signatures, we are now finally ready to explain the project!

Your Project

Recall that one of the main challenges when using cryptocurrencies is to remain anonymous. For example, if you use a particular identity (public key) to buy something from an online merchant, the merchant can look up your public key on the blockchain and see all the previous transactions that you have made with the same identity. This is of course not desirable. So, we would like to have a so-called "mixer", i.e. a smart contract in which people can make deposits with one identity and then withdraw their money with another identity such that their privacy is preserved and no one can connect the deposits with the withdrawals. Note that simple solutions such as using a trusted third-party, e.g. a broker/exchange service who would receive your money and pay it back to you at a different address, are not sufficient, since the broker can know both of your identities and make the connection. Additionally, we do not want to exchange our cryptocurrencies for fiat and then exchange them back.

1. (15 points) Design a protocol using the blind singatures scheme outlined above that can be implemented as a smart contract on the Ethereum blockchain and would serve as the mixer. Specifically, the protocol should have the following properties:

   - Anyone can deposit 1 ETH in the smart contract and receive a certificate of deposit (CD). If the CD is not issued in time (e.g. during 24 hours after the deposit), the depositor should be able to collect her money back.
   - After another fixed deadline, e.g. a week after deployment of the contract, anyone can provide a valid CD and receive 1 ETH.
   - No one should be able to withdraw the same deposit twice.
   - No one, except for the customer herself, should be able to connect the deposit made by a customer to her withdrawal. In other words, if the customer deposits using some identity and then withdraws using another identity, there should be no way for anyone else to link the two identities.

   You are allowed to introduce third-parties, e.g. those who perform the operations of a bank in the blind signature scheme, as long as this is done in a trustless manner, i.e. the third-parties should not be able to steal anyone's money or connect anyone's deposit and withdrawal identities. Also, you can assume that many people will take part in the mixing and that everyone wants to deposit and withdraw exactly 1 ETH.

Explain your solution in detail in the `pdf` file you submit and argue why it satisfies the required security properties.

2. (15 points) Implement your approach in Solidity and submit it as a `sol` file. Make sure your code does not suffer from any of the common vulnerabilities or bugs that were discussed in the course.
3. (5 points – extra credit) Make your approach decentralized, i.e. make sure that anyone on the blockchain has the exact same privileges in your smart contract as anyone else. Note that you should keep your approach trustless, as well.
4. (5 points – extra credit) When a customer wants to use her new identity to withdraw her money from the mixer contract, she needs to pay for gas. However, this is impossible if the new identity does not already hold some ether. Extend your smart contract so that one can withdraw to an empty account. To achieve this, you are allowed to introduce more third-parties and assume that all participants have access to a forum on which they can post anonymous messages that will be seen by everyone else.

   For both extra-credit parts, you should explain your approach in your `pdf` submission and apply the required changes in your `sol` code.

Notes and Tips:

- If you introduce third-parties in the contract, make sure they are incentivized to act according to your protocol, i.e. assume that all third-parties are rational and are simply trying to maximize their own payoff. Specifically, the customers might have to pay some fees to the third-parties to incentivize them to take part. This is fine, as long as the security requirements are satisfied.
- Be very careful about the attacks that can happen on your smart contract. Analyze it adversarially and cynically from the point-of-view of any customer, third-party, miner or normal node on the blockchain and make sure they cannot steal money or connect other people's identities. In short, you are now not only designing and implementing a smart contract, but also doing a security audit on it. So, you have to combine everything you have learned from Homeworks 4, 5 and 6.
- You can find excellent tutorials on blind singatures on YouTube.