Problem Set 8

This problem set is due on *Thursday, November 16* at the end of class.

Each solution is to be handed in *separately*, i.e. on separate sheets of three-hole punched paper. If the solution to a single problem takes more than one page, staple together only the pages relative to the same problem. Mark the top of each sheet with your name, 6.915, the problem number, and the date.

If you collaborate with other students, you MUST write up solutions on your own and acknowledge the people you work with.

Problem 8-1. Plaintext-aware encryption

Read the bottom of page 9 of the paper *iKP: A Family of Secure Electronic Payment Protocols* (Handout 27.) They say that they require encryption to provide not only confidentiality, but also integrity. Decryption of a ciphertext results either in a message or in a flag indicating non–validity. Correct decryption convinces the receiver that the sender *knows* the plaintext that was encrypted.

Definition 1 *We say that an encryption scheme is* **plaintext–aware** *if it is impossible to produce a valid ciphertext without knowing the corresponding plaintext.*

(a) Why is plaintext-awareness needed in iKP? Explain one possible problems if a "regular" (non-plaintext-aware) encryption function (as RSA) is used.

The concept of plaintext–awareness is related to the concept of malleability:

Definition 2 *We say that an encryption scheme E is* **non–malleable** *if it given a ciphertext $c = E(m)$ it is impossible to produce a valid ciphertext c' of a related message m'.*

(b) Compare Definitions 1 and 2 and tell us if one implies the other.

Problem 8-2. Electronic Cash

Real-life cash has two main properties:

- It is *anonymous*: meaning when you use cash to buy something your identity is not revealed, compare with credit cards where your identity and spending habits are disclosed
- It is *transferable*: that is the vendor who receives cash from you can in turn use it to buy something else. He would not have this possibility if you had payed with a non-transferable check.

The electronic cash proposals we saw in class are all "non–transferable". that is the user gets a coin from the bank, spends it, and the vendor must return the coin to the bank in order to get credit. As such they really behave as anonymous non-transferable checks. In this problem we are going to modify such proposals in order to achieve transferability.

The proposal we saw in class can be abstracted as follows: we have three agents: the Bank , the User and the Vendor .

The Bank has a pair of keys (S, P). A signature with S is a *coin* worth a fixed amount (say \$1.). It is possible to make blind signatures, meaning the User gets a signature $S(m)$ on a message m, but the Bank gets no information about m.

Withdrawal protocol

1. The User chooses a message m
2. The Bank blindly signs m and withdraws \$1 from User 's account.
3. The User recovers $S(m)$. The coin is the pair $(m, S(m))$.

Payment Protocol

1. The User gives the coin $(m, S(m))$ to the Vendor .
2. The Vendor verifies the Bank signature and sends a random challenge c to the User .
3. The User replies with an answer r
4. the Vendor verifies that the answer is correct.

The challenge–response protocol is needed in order to detect double–spending. Indeed the system is constructed in such a way that if the User answers two different challenges on the same coin (meaning he's trying to spend the coin twice) his identity will be revealed to the Bank when the two coins return to the bank. This is why the whole history of the payment protocol must be presented to the Bank when the Vendor deposits the coin.

Deposit protocol

1. The Vendor sends $m, S(m), c, r$ to the Bank
2. The Bank verifies it and add \$1 to the Vendor 's account.
3. The Bank searches its database to see if the coin was deposited already and if it was reconstruct the identity of the double–spender User .

In order to make the whole scheme transferrable we give the bank a different pair of keys $(\mathcal{S}, \mathcal{P})$. It is still possible to make blind signatures with $\mathcal{S}$. However messages signed with $\mathcal{S}$ have no value. We will call them *pseudo-coins*. When people open an account with the Bank , they get a lot of these anonymous pseudo–coins by running the withdrawal protocol with $\mathcal{S}$ as the signature key.

Suppose now the Vendor received a payed coin $m, S(m), c, r$ and instead of depositing it wants to use it to buy something from OtherVendor . What she could do is the following:

Transfer protocol

1. The Vendor sends $m, S(m), c, r$ and a pseudo–coin $m', \mathcal{S}(m')$ to OtherVendor
2. OtherVendor verifies all signatures and the pair (c, r). Then sends a random challenge c' for the pseudo–coin.
3. Vendor replies with r'
4. OtherVendor checks the answer.

Notice however that Vendor can still double–spend the coin $m, S(m), c, r$ if she uses two different pseudo–coins to transfer it to two differnt people. Indeed since she will never answer two different challenges on the same pseudo–coin, her identity will never be revealed. The problem is that there is no link between the real coin and the pseudo-coin used during the transfer protocol. If we could force Vendor to use only one pseudo–coin for each real coin she wants to transfer then the problem would be solved.

Show how to achieve the above goal. You will need to modify both the payment and the transfer protocol.

Hint: If Vendor wants to transfer the true coin she is receiving during the payment protocol, she must be forced *then* to create a link between the true coin and the pseudo–coin she will use for the transfer later. Notice that Vendor chooses c at random, maybe c can be chosen in some different way?