Ripping Coins for a Fair Exchange

Markus Jakobsson*

Department of Computer Science and Engineering,
University of California, San Diego,
La Jolla, CA 92093

Abstract. A fair exchange of payments for goods and services is a barter where one of the parties cannot obtain the item desired without handing over the item he offered. We introduce the concept of ripping digital coins to solve fairness problems in payment transactions. We demonstrate how to implement coin ripping for a recently proposed payment scheme [9, 8], giving a practical and transparent coin ripping scheme. We then give a general solution that can be used in any payment scheme with a challenge. We also indicate how fairness can be obtained by building a contract into the coin.

1 Introduction

Many payments are practically concurrent with the delivery of the purchase; when you buy milk in a grocery store you cannot leave with the milk without paying, but neither can the store charge you and then not give you your milk. In this situation, there should not be any problems with trust since both parties are physically present. However, if you call a taxi and ask the driver to pick up some goods for you, and then return to deliver them to you, there could be a problem with trust: the taxi driver doesn't want to go and pick the items up without getting paid first (since you will not go with him,) but you don't want to pay him in advance risking that he is not honest. Let us say that the taxi driver's payment would be \$100. Paying him half the amount before he leaves, and the rest upon delivery would not solve the problem. However, if you tear the \$100 bill in two parts and give the driver one half before he leaves, and the other upon delivery, this *would* solve the problem. Neither you nor the driver will be able to use half a bill, since this will be without value, but when he returns and you give him the second half, he can tape the parts together and use the bill. If you are concerned that the taxi driver might not return just on spite, thus making you lose \$100 although *he* will not get it, you can both tear bills and give each other halves in order to keep each other honest. When he returns, you will exchange the second halves, and then perform a standard payment.

Similar issues of trust can arise in many digital cash applications [1, 2, 5, 6, 7, 8, 9, 10, 11, 13], since the two parties in the transaction may be geographically

* Research partly done at DigiCash, the Netherlands. Email address: markus@cs.ucsd.edu

large distances apart, and also because of the inherent problems of proving that a payment or delivery did or did not take place. In section 2 and 3 we analyze the problem in the context of anonymous electronic cash protocols, and discuss the intuition behind our solution based on coin ripping. In section 4, we show how to rip coins in a newly proposed privacy protecting payment system [9]. In section 5 we show how to rip a coin in any challenge based payment scheme.

A different solution to achieve a fair exchange is to let the challenge be a (randomly) hashed contract. This allows the buyer to prove that a certain deposited coin was designated for a certain purchase by showing the hash preimage to the challenge. However, it gives a solution with the power balance on the Shop's side; if the Shop does not deliver the item paid for, the buyer will have to sacrifice his anonymity in order to make a complaint at the Bank. We will show how this solution works in the Appendix.

Our coin ripping solution gives the buyer the advantage, and lets an honest buyer remain anonymous at all times. If the buyer decides not to fulfil the payment after the ordered item or service is delivered, neither the buyer nor the seller will get the coin, and moreover, the seller will not be able to prove what the buyer has done, but only blacklist him for future transactions (assuming the seller knows who the buyer is, which often is the case.) Often, though, the seller will be the most powerful party of the two and this balance might therefore be preferable. If such pranks are a persistent problem, the solution can be modified to increase the costs of pranks by requiring rip-spent coin payments in both directions.

2 Our Requirements

The payment systems we will apply our coin ripping method to will be assumed to have the following properties:

1 Representations of coins cannot be created without the help of the Bank.
2 The withdrawer will be able to prevent the Bank from correlating the withdrawn coins to the withdrawer when spent and deposited. In existing schemes, withdrawal protocols with this property often use what is called a "blind signature."
3 The receiver of the payment will be able to verify the correctness of a coin without the help of the Bank.
4 Any party or coalition of parties that attempts to spend more coins than they have withdrawn (so called "over-spending,") will be detected by the Bank, and moreover, the Bank will be able to calculate the identity of the over-spender with an overwhelming probability.

We will let the payer *rip-spend* a coin, and add the following requirements while keeping the above properties for both rip-spent and fully spent coins.

5 A rip-spent coin as such is useless for the receiver of the payment, as he will not be able to deposit it, i.e., it is infeasible to create a depositable coin from any number of rip-spent coins.

6 The receiver of the payment will be able to verify that a rip-spent coin is of the right form.

7 If a payer cheats the receiver of the payment by not giving him the second part of the coin later, the payer is prevented from using the coin in any other payments.

8 The receiver of the payment will be able to verify that the second part of the coin fits with the first part, and thereby be convinced that he will be able to deposit it.

A trivial solution is to use a two-spendable coin, i.e., a coin that can be spent twice without revealing the identity of its owner [2, 9] and let the payer rip-spend a coin by performing one of the two spendings, and fully spend it by spend the second. The Bank will only give a person money or credit when *both* of the two corresponding coins are deposited. If the payer decides to cheat the receiver of the ripped coin, he will not be able to use the coin for a payment later, as he will then have spent the coin for a total of three times, and will thus be identified by the Bank as an overspender. This solution does not, however, have the following desirable feature that we will require:

9 A payer may not spend or rip-spend more coins than he has withdrawn, or he will be detected and identified.

(This requirement is a little bit different from requirement 5, as we are referring to fully spent coins in requirement 5, but relate to rip-spent coins here. For a two-spendable coin, the coin owner could instead of correctly spending a coin, spend it twice in a rip-spending fashion without getting caught. We want a solution such that one coin can only be rip-spent once without detection.)

Another setback with this trivial solution is that there will be an added cost of letting users rip-spend coins; instead of one-spendable coins, all users, Shops, and the Bank will have to store and send two-spendable coins, which have larger representations. Two desirable (but not necessary) properties we will require of our new coins will be:

10 Coins that can be ripped will not have a larger representation than normal coins.

11 The Bank can never know whether a coin will be or has been ripped unless the payer defaults on a rip-spent coin; in other words, the ripping will be transparent to the Bank.

3 Intuition

We will here discuss a sketch of a general solution that has all the listed features of the last section. First, we will describe a rather general format for electronic cash systems fitting many proposed schemes [1, 2, 5, 7, 8, 9, 10] to a high degree.

Bank Signatures: The Bank authenticates coins using a signature function pair (S_{Bank}, V_{Bank}), where S_{Bank} is the private function used for producing a signature, and V_{Bank} is the public function used for verification of a signature.

A Coin: A coin is a pair (x, s), where x is a random number known only to the withdrawer and $s = (S_{Bank} \circ g)(x)$ for some function g. The coin scheme must be secure against existential forgery [12], and a signature of a message must not be useful in producing a signature of another legal message. The pair (x, s) must be chosen by both the withdrawer, Alice, and the Bank in a way that achieves *blinding*, i.e., so that the Bank can not know which message was signed. We will in this section disregard many details for the sake of clarity, and avoid the discussion of how schemes are made to be safe against existential forgery and how the blinding will be done.

Spending a Coin: The user spends a coin by engaging in a protocol with the Shop that gives the Shop the transcript $(\overline{x}, c, \overline{s})$, where c is a challenge chosen by the Shop and $\overline{x}$ and $\overline{s}$ are functions of x and s, and of c. The triple $(\overline{x}, c, \overline{s})$ is non-forgeable and its correctness can be verified by the Shop. In many implementations, including this, a one-spendable coin corresponds to a line whose equation in its turn corresponds to the identity of the withdrawer. For each time a coin is spent, the payer has to give a point on this line. Answering two challenges for one coin will reveal the equation of the line, and thereby also the identity of the over-spender. Answering only one challenge per coin will give no Shannon information about the identity of the payer.

Depositing a Coin: The Shop sends the Bank the triple $(\overline{x}, c, \overline{s})$, and the Bank verifies its correctness. The triple is related to the withdrawal session where (x, s) was obtained; thus, the Bank is able to trace coins given two transcripts

Ripping a Coin: The user can rip-spend a coin by applying a one-way function f to either $\overline{x}$, $\overline{s}$, or parts of these. The Shop will at the end of the payment protocol have received the transcript $(\overline{x}', c, \overline{s}')$, where either $\overline{x}'$ or $\overline{s}'$ is the result of applying f to $\overline{x}$ or $\overline{s}$, or parts thereof. We will do this so that the Shop can still verify the correctness of the triple in order to be convinced that it is a valid coin. However, the Bank will not accept the triple $(\overline{x}', c, \overline{s}')$ for credit, as it will not be of the correct form, and so, the Shop cannot deposit it for credit. It is important that the Shop will not be able to produce a valid, depositable transcript $(\overline{x}, c, \overline{s})$ from $(\overline{x}', c, \overline{s}')$. The payer completes the payment (gives the second part of the coin) by sending the Shop information that allows the Shop to calculate the depositable transcript $(\overline{x}, c, \overline{s})$. Should the payer not do this, then the Shop gives the triple $(\overline{x}', c, \overline{s}')$ to the Bank. This will prevent the payer from spending the coin again without being detected, as the identity of the payer will be possible to obtain from two transcripts from the same coin, even though f is applied to the transcript.

It should be apparent that our general outline of the payment protocol using ripping does not necessarily lose any of the listed properties, but we will certainly have to specify and analyze our solution before it is clear that all the properties are indeed retained. We will in the next section give an example of a payment protocols, and show specifically how coins in this can be ripped.

4 Coin-Ripping for a Specific Protocol

We will show how to rip a coin in an abstraction of the protocol by Ferguson [9], but the method is applicable to many other schemes as well. We will make only minor changes to the protocols, and our solution will satisfy all the requirements listed. In our description, Alice will be the payer and Shop the receiver of the payment, and we will use the denotation introduced in the previous section.

Bank Signatures: The Bank authenticates coins using a signature function pair (S_{Bank}, V_{Bank}), where $S_{Bank}(x) = x^{1/e}$ modulo a composite N. Here, $e = e_1 e_2$, where $e_1 \neq \pm 1$ is an odd integer and e_2 is a large prime. (In the original protocol, [9], $e_1 = 1$ was used.)

Original Payment Protocol: The following protocol is executed:

1. Alice sends $\overline{x}$ to the Shop.
2. The Shop constructs a challenge c and sends it to Alice.
3. Alice sends the answer $\overline{s}$ to the Shop, who verifies that $V_{Bank}(\overline{x}) = \Theta(\overline{s}, c)$ for a function Θ.

Rip Coin Payment Protocol: The following protocol is executed to rip-spend the first part of a coin:

1. Alice sends $\overline{x}$ to the Shop.
2. The Shop constructs a challenge c and sends it to Alice.
3. Alice sends the answer $\overline{s}' = \overline{s}^{e_1}$ to the Shop, who verifies that $V_{Bank}(\overline{x}) = (\Theta(\overline{s}, c))^{e_1}$.

Thus, the difference is that instead of giving an e^{th} root as a signature, the payer only gives an ${e_2}^{th}$ root. This way, the Shop does not get a transcript that he can deposit in return for money, unless it can calculate ${e_1}^{th}$ roots. Alice spends the second part of the coin by giving $\overline{s}$ to the Shop, who verifies that this is correct. Should Alice not give the Shop this, then the Shop can give the transcript $(\overline{x}, c, \overline{s}')$ to the Bank, who after verifying that it is a correct transcript for a rip-spent coin stores it. Should Alice ever spend the same coin again, then the Bank can find out her identity exactly as for a normal over-spending, and punish her for over-spending. Note that giving the second part of the rip-spent coin does not constitute an over-spending as only one challenge will have been responded to.

Theorem 1: The rip-spending protocol preserves the anonymity of the payer.

Proof: Assume that there is a strategy for the Shop and the Bank to collaborate in the rip-spending protocol to find out any information about the identity of the payer. They can use this strategy to find the same information in the original coin spending protocol as follows: Let $(\overline{x}, c, \overline{s})$ denote a fully spent coin. Let $(\overline{x}, c, \overline{s}')$ be the same coin, but ripped.The Shop can calculate the transcript $(\overline{x}, c, \overline{s}')$ from $(\overline{x}, c, \overline{s})$ as f is public. Therefore, if any information about the identity of a spender of a rip-spent coin can be found, the same information must be possible to obtain for a fully spent coin. Thus, since the original protocol preserves the privacy of the honest user, so must our rip-spending protocol. □

Theorem 2: If the payer after withdrawing k coins successfully can spend or rip-spend $k+1$ coins without detection, then he could do the same for Ferguson's protocol with $S_{Bank}(x) = x^{1/e_2} \, mod \, N$.

Proof: This holds since the rip-spend protocol is identical to Ferguson's original protocol for $S_{Bank}(x) = x^{1/e_2} \, mod \, N$, and any spent coin (using $S_{Bank}(x) = x^{1/(e_1 e_2)}$) can be used to construct a rip-spent coin (for which $S_{Bank}(x) = x^{1/e_2}$.)
□

We will now prove that the receiver of a rip-spent coin will not be able to cash it unless he can invert a one-way function:

Theorem 3: The Shop will not be able to construct a cashable coin $(\hat{x}, \hat{c}, \hat{s})$ from a rip-spent coin $(\overline{x}, c, \overline{s}')$ unless he can calculate ${e_1}^{th}$ roots.

Proof: Assume that there is a polynomial-time cheating strategy for the Shop, allowing it to construct a coin $\hat{C}_1 = (\hat{x}, \hat{c}, \hat{s})$ that can be deposited from a rip-spent coin $C_{1/2} = (\overline{x}, c, \overline{s}')$. Call the algorithm for this A.

Let Alice spend a coin $C_1 = (\overline{x}, c, \overline{s})$ in a Shop. The Shop can produce a rip-spent coin $C_{1/2} = (\overline{x}, c, \overline{s}')$ from this by applying f to the $\overline{s}$. Let $C_{1/2}$ be the input to A and call the output $\hat{C}_1$. There are three possibilities:

- $C_1 = \hat{C}_1$:
  This means that given input $C_{1/2} = (\overline{x}, c, \overline{s}')$, where c is chosen according to some strategy $\Phi(\overline{x})$, A produces an output $C_1 = (\overline{x}, c, \overline{s})$, where $\overline{s} = \overline{s}'^{1/e_1}$. In the original protocol, different exponents are used to encode different coin denotations. Thus, being able to produce this kind of transcript $\hat{C}_1$ from $C_{1/2}$ allows us to change the value of a coin in the original protocol.
- $C_1 = (\overline{x}, c_1, \overline{s}_1)$, $\hat{C}_1 = (\overline{x}, c_2, \overline{s}_2)$, $c_1 \neq c_2$:
  This means that given a correct payment transcript for some coin with one challenge answered, we can produce a correct payment transcript for the

same coin but with another challenge answered. Two such transcripts counts as two spendings, and given a one-spendable coin, the Bank could given these two calculate the identity of the spender. This contradicts the fact that the original protocol which we build our protocol on is privacy preserving, as this would give an algorithm for deciding the identity of the spender of a coin in that protocol.

– $C_1 = (\overline{x}_1, c_1, \overline{s}_1)$, $\hat{C}_1 = (\overline{x}_2, c_2, \overline{s}_2)$, $\overline{x}_1 \neq \overline{x}_2$:
  Both C_1 and $\hat{C}_1$ are valid coins. However, since $\overline{x}_1 \neq \overline{x}_2$, they are two coins with *distinct* secret representations, i.e., two different coins. Since they both stem from one withdrawal session, and both will be accepted by the Bank as distinct and valid coins, this will allow a coalition of Shops and Payers to make the Bank accept $k+1$ deposited coins after only having withdrawn k coins in the original protocol using $S_{Bank}(x) = x^{1/(e_1 e_2)}$. Since coins using $S_{Bank}(x) = x^{1/e_2}$ can be calculated from such coins, this gives us a contradiction if coins cannot be forged in the original protocol. □

We have thus proved that the introduction of coin ripping can be made safely for the protocol by Ferguson.

5 The General Solution

We have in our example assumed the use of RSA-signatures to authenticate the coins. However, we will show how any digital cash scheme where the payment protocol is of a certain general form can be modified to allow ripping. The form is a three move protocol with the payer sending a message $\overline{x}$ to the Shop, who responds with a challenge c of some minimal size followed by the payer sending a response $\overline{s}$ to the challenge c. In this general solution, however, rippability will incur a small storage overhead, as well as a minor communication overhead. The idea is to use a special form of challenge jointly set by the payer and the receiver of the payment.

Let $(f_1, f_2, \oplus)$ be a triple of functions such that it is hard to find a collision (x_1, x_2), (x_1', x_2') such that $f_1(x_1) \oplus f_2(x_2) = f_1(x_1') \oplus f_2(x_2')$. An example of such a triple $(f_1, f_2, \oplus)$ is

$$\begin{cases} f_1(x) = {g_1}^x & mod \text{ p} \\ f_2(x) = {g_2}^x & mod \text{ p} \\ a \oplus b = ab & mod \text{ p} \end{cases}$$

where p is a prime. This is collision-free if the representation problem is hard; this has been shown being as hard as the discrete logarithm problem [3].

The participants jointly calculate the challenge, c, the following way:

1. Alice chooses a random number r_A and calculates $c_A = f_1(r_A)$. She calculates a commitment *com* to c_A, unconditionally safe for the receiver. Alice sends *com* to the Shop.
2. The Shop sets his share of the challenge, c_S, as $f_2(r_S)$ for some r_S, and sends this c_S to Alice.
3. Alice opens up her commitment.
4. The Shop verifies the correctness of the commitment *com*. The challenge is $c = c_A \oplus c_S$.

The user then rip-spends the coin using this challenge c, spending the coin as he usually would. He spends the other half by revealing r_A to the Shop. In order to deposit the coin, the Shop will have to give the Bank the spent coin *and* the pair (r_A, r_S), thus proving that the second part of the coin has been spent. In order to file a complaint (if Alice did not give the second part of the coin,) the Shop just sends the Bank the spent coin. A coin can be spent all in one part by letting the Shop set both r_A and r_S, and construct the challenge the usual way from these numbers.

Theorem 4: The rip-spending protocol preserves the anonymity of the payer.

This proof is similar to the proof of Theorem 1. Assume that there is a strategy for the Shop and the Bank to collaborate in the rip-spending protocol to find out any information about the identity of the payer. They can use this protocol to help them find the same information in the original coin spending protocol as follows: The Shop sets the challenge as $c = f_1(r_A) \oplus c_S$, where r_A is chosen at random and c_S is chosen according to the cheating strategy for the rip-spent coin. Thus, if the Shop can learn any secret information from a ripped coin, it could learn exactly the same kind of information in the original protocol, so ripping a coin will leak no information. $\Box$

Theorem 5: If Alice does not send the second part of the coin then she will be prevented from spending this coin later.

Proof: This holds since two rip-spent coins will correspond to an over-spent coin in the original protocol if the special form of the challenge is not considered, and so the Bank will be able to find the identity of the over-spender with overwhelming probability. To see that the probability of finding a cheating payer's identity remains the same as in the original protocol, note that Alice cannot influence what the final challenge will be, since she cannot find collisions for the commitment scheme, as we are using a commitment scheme unconditionally safe for the receiver. Thus, the probability of finding Alice's identity is not affected by the fact that she sets part of the challenge. $\Box$

Theorem 6: The Shop will not be able to construct a cashable coin $(\hat{x}, \hat{c}, \hat{s})$ from a rip-spent coin $(\overline{x}, c, \overline{s}')$ unless it can invert f_1.

Proof: The Shop cannot find an r_A such that $c_A = f_1(r_A)$; since *com* is a commitment of c_A and not r_A, the commitment cannot give any help. Therefore, he cannot find another representation of the challenge used for that coin. The rest of the proof is analogous to the proof of Theorem 3, with the small difference that the Shop in the first case instead will be able to invert f_1. □

Thus, we have proved that rip-spending can be safely introduced in any payment protocol with a challenge of sufficient length.

6 Conclusion

We have shown how coins can be rip-spent in order to achieve fairness in payment protocols. We have given an efficient solution based on an existing payment protocol, and then a general solution for any challenge-based scheme with large enough challenge.

There are many alternative settings and extensions to the protocol we have looked at. For example, we can obtain extensions to multi-spendable coins and divisible coins without any further changes to the protocol; for each rip-spent coin or rip-spent share of coin, the spender will have lost his ability to fully spend this coin or share of coin without being charged with over-spending.

Similarly, observers can be used at the same time as coin ripping. An observer is a tamper-safe piece of hardware that keeps part of the coins to prevent over-spending. It will only answer the correct number of challenges for each coin, e.g., one challenge for a one-spendable coin, two for a two-spendable, etc. Since only one challenge will need to be answered for each one-spendable coin, even if it is ripped and spent in two parts, the observer will not be affected. The ripping algorithm described can be located entirely in the non-observer part of the device.

We note that the efficiency will be basically unchanged with ripping of coins. We will have to communicate one extra message, the "opening up" of the hidden information, and both parties will have to compute one extra function. Also, they will have to store the rip-spent coin along with some tag specifying the transaction involved until the second part will be released.

7 Acknowledgements

Thanks to David Chaum for suggesting the problem and Russell Impagliazzo, Stefan Brands, Niels Ferguson, Giovanni Di Crescenzo and Karin Högstedt for valuable feedback.

References

1. S. Brands, "An Efficient Off-line Electronic Cash System Based On The Representation Problem," CWI Technical Report CS-R9323, April 11, 1993
2. S. Brands, "Untraceable Off-line Cash in Wallet with Observers," Crypto '93, pp. 302-318
3. S. Brands, "An Efficient Off-line Electronic Cash System Based On The Representation Problem", CWI Technical Report CS-R9323, April 11, 1993. Can be obtained by ftp from ftp.cwi.nl, directory pub/CWIreports/AA, filename CS-R9323.ps.Z.
4. D. Chaum, "Achieving Electronic Privacy," Scientific American, August 1992, pp. 96-101
5. D. Chaum, A. Fiat, M. Naor, "Untraceable Electronic Cash," Crypto '88, pp. 319-327
6. D. Chaum, T. Pedersen, "Wallet databases with observers," Crypto '92, pp. 89-105
7. R. Cramer, T. Pedersen, "Improved Privacy In Wallets With Observers," Eurocrypt '93, pp. 329-343
8. N. Ferguson, "Single term off-line coins," Eurocrypt '93, pp. 318-328
9. N. Ferguson, "Extensions of Single-term Coins," Crypto '93, pp. 292-301
10. N. Ferguson, "Single term off-line coins," Technical Report CS-R9318, CWI, Amsterdam, 1993. Anonymous ftp: ftp.cwi.nl:/pub/CWIreports/AA/CS-R9318.ps.Z
11. M. Franklin, M. Yung, "Secure and efficient off-line digital money," Automata Languages and Programming, 20th International Colloquium, ICALP '93, pp. 265-276
12. S. Goldwasser, S. Micali, R. Rivest, "A 'Paradoxical' Solution to the Signature Problem", 25th Annual Symposium on Foundations of Computer Science, 1984, pp. 441-448
13. T. Okamoto, K. Ohta, "Universal Electronic Cash," Crypto '91, pp. 324-337

8 Appendix: Another Approach for Fairness

Another method for achieving fairness is to let the Shop and the payer, Alice, agree on a contract C , the Shop pick a random value r, and use $c = h(C, r)$ as the challenge, where h is a collision-free hash function. When Alice receives the item paid for, she will give a signature $\sigma = S_{Alice}(r)$ as a receipt. We will assume that this signature and the delivery will be simultaneous. (Note that this need not be assumed for ripped coins.) Here, we will require S_{Alice} to be a signature function that is *not* safe against existential forgery, i.e., for which it will be possible to find correct message-signature pairs for random messages. This holds for RSA since we can set σ and then calculate $r = \sigma^{e_{Alice}}$ modulo N_{Alice}.

If the Shop cashes the coin but does not deliver the item ordered, then Alice can go to the Bank and show the Bank (C, r) as a proof that she has paid. The Bank finds the spent coin protocol $(\overline{x}, c, \overline{s})$ for which $c = h(C, r)$ and asks the Shop to show the receipt σ that the ordered item was received. If the receiver of the payment cannot give the Bank σ, then the receiver must not have delivered the item ordered, and the payer wins the case; if the receiver has been given the receipt σ, then he shows this to the Bank and the payer loses the case.

Theorem 7: It is not possible to win a case for a contract that was never agreed on, or where the receipt of delivery was given to the Shop.

Proof: This holds since h is collision-free, meaning that it will not be possible to find two pairs $(r, C), (r', C')$ such that $h(C', r') = h(C, r)$. Thus, it is not possible to find a new contract for a certain challenge, and therefore, Alice cannot falsely claim that the Shop did not follow the contract used for the challenge. □

Theorem 8: The privacy of the payment scheme will not be compromised by the use of signed receipts.

Proof: We will show that The Shop can produce a transcript $(\overline{x}, \overline{s}, C, r, \sigma)$ such that for an arbitrary supposed payer Alice

$$\begin{cases} c = h(C, r) \\ \sigma = S_{Alice}(r), \end{cases}$$

and $(\overline{x}, c, \overline{s})$ is a correct transcript for a spent coin.

Let (x, s) be the representation of a coin *withdrawn by the Shop.* The Shop performs the following steps:

1. Select a random σ.
2. Calculate $r = \sigma^{e_{Alice}}$.
3. Set an arbitrary contract C.
4. Calculate a challenge $c = h(C, r)$.
5. Spend the withdrawn coin using the challenge c.

It is not possible to see that the coin was withdrawn by the Shop and not Alice; therefore, and because of the random invertibility of the signature function S_{Alice}, the Shop can fake transcripts by using his own coins. Thus, he cannot prove that Alice in fact did buy something from him, and the protocol is privacy preserving with the use of contracts if it is privacy preserving without. □