

A Mobile Agent Clone Detection System with Itinerary Privacy

T. C. Lam and Victor K. Wei

Department of Information Engineering

The Chinese University of Hong Kong

Shatin, NT Hong Kong

tclam0@ie.cuhk.edu.hk and kwwei@ie.cuhk.edu.hk

Abstract

We propose a system for the detection of unauthorized cloning of mobile agents by malicious hosts. The system can detect clones after the fact, identify the culprit, and then seek penalties. The agent migration protocol is offline (non-centralized). The itinerary of an agent is privacy-protected. Our scheme is based on Wong's transferable extension [Won01] of Ferguson's single-term offline untraceable e-cash [Fer93]. The main technique is to map the clone detection problem to the double-spending problem in transferable e-cash, and then adopt and adapt existing solutions from the latter field.

1. Introduction

Mobile agent technology offers a new computing paradigm in which an autonomous program, working on behalf of its owner, can suspend its execution on a host computer, migrate itself to another agent-enabled host on the network, and resume execution on the new host [JK99]. Thanks to its autonomy and mobility, the concept of a mobile agent is applied to areas such as workflow systems, electronic commerce and information retrieval systems. However, security threats pose a significant deployment bottleneck.

A number of surveys on these security threats and its counter-measures have been studied, ranging from risking a host to a malicious agent, to risking an agent to a malicious host [FGS96, Che98, JK99, NC99, GM00, Jan99]. One particular threat, unauthorized mobile agent cloning, is an open issue that is difficult to solve. Baek [Bae98] proposed a solution in which a central site oversees every agent migration.

In this paper, we proposed a new scheme for the detection of unauthorized cloning of mobile agents. Our scheme can detect clones after they are produced, and identify the culprit host which performed the cloning. Penalties can then be sought. However, our scheme does not prevent

unauthorized cloning from happening.

There are several added benefits in our system. One is "itinerary privacy" – the identities of honest hosts traversed by an agent cannot be traced from the contents of that agent. Another is offline computation. The agent migration protocol is a "local" computation involving only the agent, the sending host, and the receiving host – but no other party. No centralized oversight is needed. In comparison, Baek's [Bae98] clone detection system requires centralized oversight.

Our scheme is based on a cryptographic technique called *e-cash*. An *e-cash* is the electronic counterpart to the physical cash (paper bills and coins) from real life. It can be used in electronic transactions as a medium of payment. In a *transferable e-cash*, the payment recipient can immediately use the received e-cash to pay another entity.

An e-cash is ultimately an electronic file. Therefore, it is easy to produce an exact duplicate of an e-cash. One of the security requirements in the design of an e-cash system is that any duplicate of an e-cash cannot be spent again. The problem of e-cash duplication and its multiple uses is called the *double spending* problem. E-cash is an active area of research in cryptography [CFN88, vA90, OO91, Fer93, Bra93, Won01]. Quite a few schemes have been devised to combat the double spending problem while achieving various other objectives. In many schemes, double spending cannot be prevented from happening. Rather, the e-cash system examines the circulations in the system and checks for duplicates. When duplicates are observed, the e-cash system has a designed method to subsequently determine the identity of the offender who performed the double spending.

One of our central motivations is the observation that double spending in e-cash has several similarities to the problem of unauthorized cloning of mobile agents. Mobile agents are also ultimately electronic files, and thus can be exactly duplicated with ease. In order to devise a

mobile agent clone detection system, one can look for existing techniques from e-cash systems.

Our scheme for detecting unauthorized cloning of mobile agent is motivated by Wong's [Won01] transferable extension of Ferguson's single-term offline untraceable e-cash [Fer93]. By modifying and extending some core techniques, we are able to design a system that can determine whether two given agents are clones of each other. In the case when two clones are observed, our system can determine, from the information contained in the clones, the identity of the host which performed the cloning.

Bredin, et al. [BKR98] also proposed an e-cash based system for agent security.

The rest of the paper is organized as follows: Section 2 summarizes the main results of our scheme. Section 3 defines the clone detection system environment. Section 4 gives the specifications of the protocols. Security and privacy analysis is described in Section 5. Finally we have a conclusion in Section 6.

2. Main Results

Two agents with the same identity and the same fixed codes are unauthorized clones. The itineraries of two cloned agents can be used to reveal the identity of the host which performed the cloning. The two itineraries should have a common beginning, then a two-prong fork some time later. The host at the forking point should be the culprit.

If each agent is required to record and carry its itinerary (i.e. the sequence of hosts it has visited), then we already have a clone detection system. However, itinerary privacy is an issue. Traversed hosts may not want subsequent hosts to know they are in the itinerary. If the agent encrypts the itinerary, then only those who have the decryption key can identify the culprit of cloning, but they can also read the itinerary.

Our system achieves the following goals simultaneously:

- Detection of unauthorized clones and identification of the culprit.
- Public verification – any host can perform the clone detection and the culprit identification.
- Itinerary privacy – an honest host's presence in an itinerary is never revealed.
- Offline (distributed) computation – agent migration involves only local entities: the agent, the sending host, and the receiving host.

In the following, we give an overview of our

clone detection system. Each agent records and carries its itinerary in a file called the *passport*. When two agent clones are captured, their passports reveal the identity of the culprit host.

In order to protect itinerary privacy, the passport is encrypted. The encryption is such that (1) whether an honest host is contained in an itinerary cannot be determined from the passport, and (2) the identity of culprit hosts can be computed from clones' passports.

Each host is required to expend an "e-token" for the purpose of receiving, hosting, and sending away an agent. It conducts a multi-round interactive protocol to withdraw e-tokens from a supervisory body called *EPA (E-token and Passport Authority)*. The use of e-tokens enforces that the hosts help agents record their itineraries honestly. The e-tokens can be withdrawn early and stockpiled. Then the migration of an agent from one host to another requires only offline computation involving the agent, the sending host, and receiving host but no other party.

The main techniques used in the design of the passport is motivated by Wong's transferable extension of Ferguson's e-cash.

3. The Clone Detection Environment

In our mobile agent clone detection environment, there are hosts which provide computing resources, mobile agents who migrate from hosts to hosts, a particular central host site called *EPA (E-token and Passport Authority)*. Each agent contains a file called "passport" as part of its state. When a host creates an agent, it needs to conduct an interactive protocol with the EPA to initialize the agent's passport. Each host obtains, through interactive protocols with the EPA, multiple files called "agent-hosting e-tokens".

Assumptions about public parameters: The RSA public key v and the RSA modulus n of the EPA are known to all. The parameter v is assumed to be a large prime. The private key $1/v$ of the EPA is kept secret in the EPA. The parameter p is a large prime where $p-1$ is a multiple of n . Parameters g_1, g_2, g_3 are publicly known elements of large orders in the multiplicative group of Z_n^* , h_3, h_4 are publicly known elements of orders n in the multiplicative group Z_p^* . The mappings f is a suitable one-way function from Z_n^* to Z_v^* , f_1 is a suitable one-way function from Z to Z_p^* with output of order n , f_2 is a suitable one-way function from $Z_n^* \times Z_n^*$ to

Z_v^* , f_3 is a suitable one-way function from $Z_n^* \times Z_n^* \times Z_n^*$ to Z_v^* .

Each host has a unique identification number (identity), denoted U .

Throughout this paper, we assume computations are done modulo n unless explicitly specified otherwise.

Definition 1: An *agent* is defined as $\mathcal{A} = (I, \mathcal{K}, S, \mathcal{P})$ where I is the identity of the agent, \mathcal{K} is the code of the agent, S is the state of the agent, and \mathcal{P} is the passport of the agent.

Remark: In classic notation, the agent consists only of I , \mathcal{K} and S , i.e. $\mathcal{A} = (I, \mathcal{K}, S)$. In that case, the passport \mathcal{P} is a part of S .

Definition 2: A *passport* is a 7-tuple $\mathcal{P} = (\ell, \mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{x}, \mathbf{r}, \mathbf{R})$ where ℓ is a positive integer, $\mathbf{a} = (a_1, a_2, \dots, a_\ell)$, $\mathbf{b} = (b_1, b_2, \dots, b_\ell)$, $\mathbf{c} = (c_1, c_2, \dots, c_\ell)$ are integer vectors of length ℓ , and $\mathbf{x} = (x_1, x_2, \dots, x_{\ell-1})$, $\mathbf{r} = (r_1, r_2, \dots, r_{\ell-1})$, $\mathbf{R} = (R_1, R_2, \dots, R_{\ell-1})$ are integer vectors of length $\ell-1$. The *length of the passport* is ℓ .

The following technical definition is useful.

Definition 3: An integer 6-tuple (a, b, c, x, r, R) is a *Type-I vector* if $R^v = C^r A^x B$, where $A = ag_1^{f(a)}$, $B = bg_2^{f(b)}$, $C = cg_3^{f(c)}$. An integer 8-tuple $(a, b, c, x, r, R, I, \mathcal{K})$ is a *Type-II vector* if $R^v = C^r A^x B$, where $h_1 = f_1(I)$, $h_2 = f_1(\mathcal{K})$, $A = ag_1^{f(a)}$, $B = bg_2^{f(b)}$, $C = cg_3^{f(c)}$.

Definition 4: A passport \mathcal{P} of length $\ell > 1$ is a *valid passport* for agent $\mathcal{A} = (I, \mathcal{K}, S, \mathcal{P})$ if

- (1) $(a_1, b_1, c_1, x_1, r_1, R_1, I, \mathcal{K})$ is of Type-II, and
- (2) $(a_i, b_i, c_i, x_i, r_i, R_i)$ is of Type-I, for each i , $2 < i < \ell - 1$, and
- (3) $x_i = f_3(a_{i+1}, b_{i+1}, c_{i+1})$, for each i , $1 < i < \ell - 1$.

By convention, all passports of length 1 are valid.

Figure 1 illustrates the internal “chaining” structure of the passport. Each column corresponds to an agent-hosting e-token. The value x_ℓ is mapped from $(a_{\ell+1}, b_{\ell+1}, c_{\ell+1})$ via some one-way mechanism. The values (r_ℓ, R_ℓ) is mapped from $(a_\ell, b_\ell, c_\ell, x_\ell)$ and some other information via another one-way mechanism.

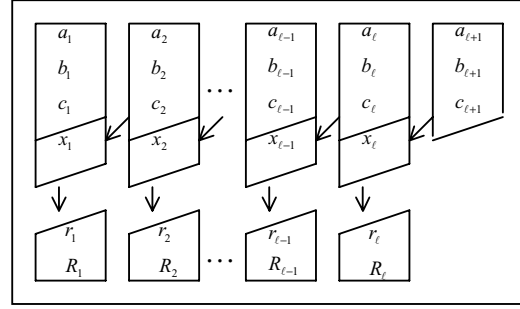


Figure 1: The internal “chaining” structure of the passport.

Definition 5: An *agent-creation e-token* for Host U and agent $\mathcal{A} = (I, \mathcal{K}, S, \mathcal{P})$ is a 7-tuple of integers

$$(a, b, c, U, k, S, T)$$

satisfying $S = (C^k A)^{1/v}$, $T = (C^U B)^{1/v}$,

with $h_1 = f_1(I)$, $h_2 = f_1(\mathcal{K})$

$$A = ag_1^{f(a)}, B = bg_2^{f(b)}, C = cg_3^{f(c)}$$

Definition 6 An *agent-hosting e-token* for Host U is a 7-tuple of integers,

$$(a, b, c, U, k, S, T)$$

satisfying $S = (C^k A)^{1/v}$, $T = (C^U B)^{1/v}$

with $A = ag_1^{f(a)}$, $B = bg_2^{f(b)}$, $C = cg_3^{f(c)}$

4. Protocols

4.1. Withdrawing E-tokens

A host interactively conducts these protocols with EPA for the withdrawal of e-tokens which will be needed for creating agents and hosting agents.

Agent-Hosting E-tokens Withdrawal Protocol

Host U obtains an agent-hosting e-token (a, b, c, U, k, S, T) from the EPA. This protocol is identical to the Withdrawal Protocol in Ferguson’s e-coin [Fer93]. Details are in the Appendix. The following summarizes the main features:

- Host presents its identity, U , to EPA.
- Host and EPA jointly generate the numbers a, b, c, k, S and T , in such a way that these values are known to Host but unknown to EPA.
- It is hard for Host to generate agent-hosting e-tokens without knowing $1/v$, the private

key of EPA.

As we shall see below, a host needs to expend one agent-hosting e-token for each agent that travels through it.

A host repeats the protocol and obtains multiple (and distinct) agent-hosting e-tokens from the EPA.

Agent-Creation E-tokens Withdrawal Protocol

This protocol is almost identical to the Agent-Hosting E-tokens Withdrawal Protocol. Host U conducts an interactive multi-round protocol with EPA for the purpose of obtaining an agent-creation e-token for a new agent it just created, \mathcal{A} . Details are contained in the Appendix. Host needs I and \mathcal{X} of the agent as the inputs to this protocol.

4.2 The Agent Creation Protocol

Host U creates a new agent via these steps:

Create: Host U creates the first three components of an agent $\mathcal{A} = (I, \mathcal{X}, S, \mathcal{P})$, where S is initialized.

Get Token: Host U performs Agent-Creation E-token Withdrawal with EPA, and obtains an agent-creation e-token (a, b, c, U, k, S, T) for \mathcal{A} .

Initialize Passport: Host U initializes $\mathcal{P} = (\ell, \mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{x}, \mathbf{r}, \mathbf{R})$, with the agent-creation e-token, by setting $\ell = 1, a_1 = a, b_1 = b, c_1 = c$.

Remark: The values of a, b, c, k, S, T (i.e. all values in the token except U) are unknown to EPA.

4.3. The Agent Migration Protocol

In this protocol, the agent $\mathcal{A} = (I, \mathcal{X}, S, \mathcal{P})$ migrates from the sending host with identity U to the receiving host with identity U' , where $\mathcal{P} = (\ell, \mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{x}, \mathbf{r}, \mathbf{R})$. After the protocol, the passport is updated to $\mathcal{P}' = (\ell' = \ell + 1, \mathbf{a}', \mathbf{b}', \mathbf{c}', \mathbf{x}', \mathbf{r}', \mathbf{R}')$. Its length is incremented by one.

Host U' checks passport validity, perform challenge-and-response with U , and then “chops” the passport.

Verify Passport Validity: Host U' verifies that \mathcal{P} is a valid passport for \mathcal{A} . Proceed if OK.

Challenge: Host U' selects an unused agent-hosting e-token $(a', b', c', U', k', S', T')$ in its possessions and issues a challenge x to Host U with $x = f_3(a', b', c')$.

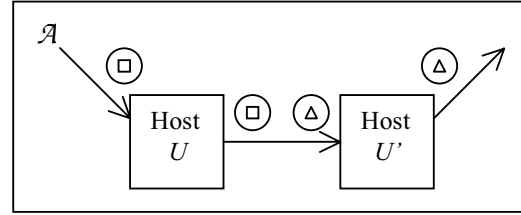


Figure 2: Each host is required to use the same token in receiving and sending an agent.

Response: Host U computes, based on the e-token (a, b, c, U, k, S, T) which it used on \mathcal{A} when \mathcal{A} entered Host U , the following response (r, R) and send it to U' : $r = xk + U \pmod{v}$, and $R = S^x T$.

[Exception: If Host U , has just created the agent. i.e. $\ell = 1$, then the agent-creation e-token (a, b, c, U, k, S, T) of the agent is used instead of the agent-hosting e-token above.]

Verify Response: Host U' verifies that (a, b, c, x, r, R) is a Type-I vector. [Exception: If $\ell = 1$, then U' verifies that $(a, b, c, x, r, R, I, \mathcal{X})$ is a Type-II vector.]

Chop: Host U' “chops” (increments) the passport \mathcal{P} to \mathcal{P}' by setting $x_\ell = x, r_\ell = r, R_\ell = R, a_{\ell+1} = a', b_{\ell+1} = b', c_{\ell+1} = c'$

Remark: Our scheme enforces that the same agent-hosing e-token is used for reception and send-away of the same agent (Figure 2). This is a crucial part of our security.

4.4 Clone Detection and Culprit Identification

Clone Detection Protocol. Two agents $\mathcal{A} = (I, \mathcal{X}, S, \mathcal{P})$ and $\mathcal{A}' = (I', \mathcal{X}', S', \mathcal{P}')$ are unauthorized clones if and only if $I = I'$ and $\mathcal{X} = \mathcal{X}'$. Any party can independently determine whether two given agents, whether visiting its site or being transmitted from elsewhere for examination, are clones.

For identifying the culprit, we need the following result:

Proposition 1: Except a negligible probability, all e-tokens (including both agent-creation and agent-hosting) have unique headers (a, b, c) .

Sketch of Proof: Ferguson’s e-cash has unique headers except negligible probability. The agent-hosting e-tokens withdrawal is identical to Ferguson’s e-cash withdrawal. The agent-hosting

e-token withdrawal is almost identical to it.

Culprit Identification Protocol

Given two clones \mathcal{A} and \mathcal{A}' with valid passports \mathcal{P} and \mathcal{P}' , the host which performed the cloning can be computed as follows:

Remark: Clones without valid passports will not be able to travel onward. We consider them less harmful and do not investigate them in this paper. For simplicity, we also assume no agent is allowed to revisit any host that it has traversed before.

Let i be the smallest positive integer such that $(a_i, b_i, c_i) \neq (a'_i, b'_i, c'_i)$. We first focus on the case $i > 1$. From Proposition 1, there is only one token (a, b, c, U, k, S, T) in our entire system whose header $(a, b, c) = (a_{i-1}, b_{i-1}, c_{i-1})$. The host which withdrew this token from the EPA is the culprit. His identity, U , was privacy-protected but now it can be computed as follows. From the $(i-1)$ -th entries of the passports \mathcal{P} and we have $r_{i-1} = kx_{i-1} + U \pmod{\nu}$ and $r'_{i-1} = kx'_{i-1} + U \pmod{\nu}$. These two linear equations enable us to solve for U .

If $i=1$, then these two clones are still in the possession of their original agent-creating host and have never migrated yet. If i does not exist and $\ell = \ell'$, then Host U who withdrew the token (a, b, c, U, k, S, T) with header $(a, b, c) = (a_\ell, b_\ell, c_\ell)$ made the clones but has not passed either of them out. If i does not exist and $\ell < \ell'$, then Host U who withdrew the token (a, b, c, U, k, S, T) with header $(a, b, c) = (a_\ell, b_\ell, c_\ell)$ made the clones but has passed only one of them out. These cases are not considered crimes.

5. Security and Privacy Analysis

We state several propositions concerning the security and privacy in our system.

Theorem 1: In the Agent Migration Protocol, if \mathcal{P} is a valid passport for \mathcal{A} , of any length, then \mathcal{P}' is also valid passport for \mathcal{A} provided the challenge-response is verified.

Proof. Straightforward.

Proposition 2. To produce an agent-hosting e-token without knowing $1/\nu$ is as hard as forging Ferguson's e-cash without $1/\nu$.

Proof Sketch: The two withdrawal protocols are identical.

Proposition 3. To produce a valid passport of length greater than one without knowing $1/\nu$

is as hard as forging Wong's transferable e-cash without $1/\nu$.

Proof Sketch: The passport is a modification of Wong's transferable e-cash. The only alterations are the passport initialization and the first agent migration. To cryptanalyze the alternations require inverting f_1 , or cryptanalyze the agent-creation e-token.

Proposition 4. Assume Host U has not committed any unauthorized cloning. To determine whether U is in the itinerary of a passport is as hard as deciding whether a transferable e-cash of Wong has passed through U' , where U' is a party in Wong's system which has not committed e-cash forgery.

Proof Sketch: Similar arguments to Proposition 3.

Propositions 1 to 3, together with Theorem 1, assure the security of our system. Proposition 4 states that our system ensures itinerary privacy.

The clone detection and the culprit identification are obviously publicly verifiable – the computations can be performed by any host. The identity of an honest host cannot be determined from the passports. This is true even if the honest host is part of an itinerary used to successfully identify a cloner. Only the cloner's identity is revealed, other honest hosts on the clone's itinerary remain anonymous.

6. Conclusion

We introduced a mobile agent clone detection system based on transferable e-cash. In our specific implementation, Wong's transferable extension of Ferguson's single-term offline untraceable e-cash is used. The central idea is to mimic a paper passport, where host identity is "chopped" to an agent passport for each migration. Records on passport are used for clone detection and culprit identification.

As an offline scheme, clones are detected after the fact. Therefore, halting of clone's malicious actions is delayed. Moreover, the passport grows in size as the agent travels [ChaPe93]. The complexity of the passport validity verification also grows.

Ferguson's e-cash is notoriously complex for 1024-bit RSA. Our system inherits such high complexity. We have no current plan of implementation to test out system performance.

References

[Bae98] J. Baek, "A design of a protocol for detecting a mobile agent clone and its correctness proof using Coloured Petri Nets." *Technical Report*

TR-DIC-CSL-1998-002, *Info. & Comm., K-JIST*, (1998).

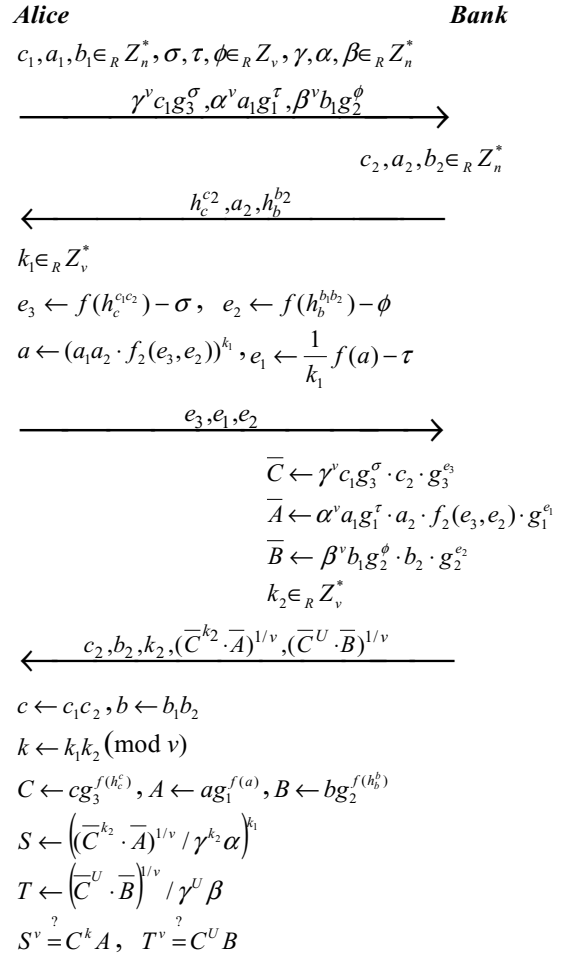
<http://atom.kjist.ac.kr/~jsbaek/pub/tr-dic-1998-02.ps>

- [BKR98] J. Bredin, D. Kotz, and D. Rus, "Market-based resource control for mobile agents." *Proc. of the Second Int'l Conf. on Autonomous Agents* (1998) 197-204.
- [Bra93] S. Brands, "Untraceable Off-line Cash in Wallet with Observers." in *Advances in Cryptology-CRYPTO'93* (1993) 302-318.
- [CFN88] D. Chaum, A. Fiat, and M. Naor. "Untraceable electronic cash." In *Advances in Cryptology - CRYPTO'88* (1988) 319 - 327.
- [ChaPe93] D. Chaum and T. P. Pedersen. "Transferable Cash Grows in Size." In *Advances in Cryptology-EUROCRYPT'93* (1993) 390 - 407.
- [Che98] D. M. Chess. "Security Issues in Mobile Code Systems." In *Mobile Agents and Security. LNCS 1419* (1998) 1-14.
- [Fer93] N. Ferguson. "Single Term Off-line Coins". In *Advances in Cryptology - EUROCRYPT'93* (1993) 318-328.
- [FGS96] W. M. Farmer, J. D. Guttman, and V. Swarup. "Security for mobile agents: Issues and requirements." *Proc. of the 19th National Information Systems Security Conf.* (1996) 591 - 597.
- [Jan99] W. A. Jansen. "Countermeasures for Mobile Agent Security." <http://www.itl.nist.gov/div893/staff/jansen/ppcounterMeas.pdf>
- [JK99] W. Jansen and T. Karygainnis, "NIST Special Publication 800-19 -- Mobile Agent Security." (1999). <http://csrc.nist.gov/mobileagents/publication/sp800-19.pdf>
- [GM00] M. J. Grimley and B. D. Monroe. "Protecting the Integrity of Agents: An Exploration into Letting Agents Loose in an Unpredictable World." *ACM Crossroads*, July 2000. <http://www.acm.org/crossroads/xrds5-4/integrity.html>
- [NC99] S. K. Ng and K. W. Cheung. "Protecting Mobile Agents against Malicious Hosts by Intention Spreading." *Proc. Int. Conf. On Parallel and Distributed Processing Techniques and Applications (PDPTA'99)* (1999) 725-729. .
- [OO91] T. Okamoto and K. Ohta. "Universal electronic cash." In *Advances in Cryptology - CRYPTO'91* (1991) 324 - 337.
- [vA90] H. van Antwerpen. *Off-line electronic cash*. Master's Thesis, Eindhoven University of Technology, Department of Mathematics and Computer Science (1990).
- [Won01] H. Y. Wong. *Issues in Electronic Payment Systems: A New Off-line Transferable E-coin Scheme and a New Off-line E-check Scheme*. Master's Thesis, Dep. of Information Engineering, The Chinese University of Hong Kong (2001).

Appendix

The e-coin withdrawal protocol of Ferguson [Fer93], below, is useful.

Ferguson's E-Coin Withdrawal Protocol:



Remark: This contains a sophisticated blind signature where Alice obtains S, T which depends on $1/v$. It is assumed that Alice authenticated her identity U with the bank prior to the protocol.

The Agent-Hosting E-token Withdrawal Protocol is identical, by setting Alice \leftarrow Host, Bank \leftarrow EPA, $h_b \leftarrow h_3$ and $h_c \leftarrow h_4$.

The Agent-Creation E-token Withdrawal Protocol is almost identical: we set Alice \leftarrow Host, Bank \leftarrow EPA, $h_b \leftarrow h_1 = f_1(I)$ and $h_c \leftarrow h_2 = f_1(\mathcal{X})$. Ferguson requires h_b and h_c to be elements of order n in the multiplicative group Z_p^* . We can set $f_1(I) = h_3^{\text{hash}(I)} \pmod{p}$, where $\text{hash}(\cdot)$ is a suitable one-way function mapping from I (and \mathcal{X}) to Z_n^* . Then h_1 and h_2 are qualified except negligible probability.