Anonymous Credentials through Acid Mixing*

Alessandro Acquisti
Carnegie Mellon University

June 2003

Abstract

Reliability and trust are open issues in the MIX-net literature. We present a variation on the MIX-net approach that can be used to issue anonymous credentials and protect the privacy of participants to various types of transactions. In our protocol, users first identify themselves to and then request credentials from a MIX they do not trust. Transactions satisfy ACID properties, thus enhancing reliability. While several applications are possible, the protocol is best suited for environments where robustness is critical.

Keywords: MIX-nets, privacy enhancing technologies, anonymity, ACID properties, credentials.

*We thank John Chuang, Rachna Dhamija, Adrian Perrig, Doug Tygar, Hal Varian, and participants to workshops and seminars at UC Berkeley and CSCW '02 for comments and suggestions.

1 Introduction

In 1981 [6] proposed an electronic mail system to hide the identity of a party who wants to communicate anonymously with other parties. The concept of a "MIX" was introduced: a third party that hides a sender's message with those of many other senders, so that no observer can link a particular sender to a particular recipient.

MIXes have been used in several applications: synchronous and asynchronous communication systems (see [15], [19], [14], and [20]); electronic voting (see e.g. [16]); and - already in [6] - "untraceable" digital credentials. Their drawbacks lie in their reliability (the probability that the MIX will actually forward the messages it receives) and the trust users must place on the MIX's honesty (see [13] and [12]). [6], for example, already described an application to electronic voting through digital pseudonyms. In that application, users have to send to a pseudonym authority acting as the MIX *both* their proposed pseudonyms *and* the identifying information required by the authority for the acceptance decision *in the same message*. More generally, in the vanilla version of the protocol the MIX must be trusted to forward messages reliably and not to reveal to others the linkages between senders and recipients.

A vast literature has attempted to address those issues. A family of solutions to the problem of trust relies on distributing trust among several third parties in a cascade of MIXes (forming a "MIX-net") that clouds the relation between messages (or pseudonyms) and their senders. In such a cascade, each message must go through a certain number of MIXes before reaching its final destination. However, collusion among MIXes in a cascade can expose the identity of the sender of a certain message. In addition, any MIX in a cascade can stop forwarding the messages transiting its server. Thus, addressing *trust* decreases the *reliability* of the system.

In this paper we discuss a variation of the MIX approach that allows anonymous credentials to be issued for various types of transactions while addressing simultaneously trust and reliability issues of traditional MIX-net systems. The method presented here is based on a new primitive - that we will call acid mixing - through which parties send at different moments to the same MIX their proposed credentials *and* the information needed for those credentials' acceptance; several users' credentials are processed in batches, and all transactions in a batch either verifiably complete or abort.

This variation on the MIX-net approach does not rely on a trusted third party because the MIX no longer can link an user to the credential he requests. The protocol also allows users to choose the credentials they want the MIX to validate, and the MIX to actually observe the credential without compromising anonymity. These properties make the protocol suited for applications where robustness is critical (for example, anonymOus payments or electronic voting) in addition to more traditional applications such

as anonymous communications.

The rest of this paper introduces the acid mixing primitive at the heart of the protocol (Section 2) and then examines some of its features (Section 3). Comparisons with related technologies and possible attacks on the protocol are described in Sections 4 and 5 respectively.

2 Description

2.1 Approach

We refer to a "credential" as a cryptographic key signed by a third party authority. An anonymous credential cannot be linked to the party that obtained it. We focus on a scenario where, in order to complete a *transaction* (such as purchasing a good, sending a message, voting in an election, and so on) "anonymously" with Alice, Bob needs to receive an anonymous credential valid for that transaction and show it to Alice; in order to receive the credential, Bob needs to show to the third party that he satisfies certain criteria set for the completion of that type of transaction.

Acid mixing, in a nutshell, works the following way: after Bob has identified himself to the third party and provided evidence that he meets the criteria to complete a certain type of transaction, the third party signs and gives him a key (that is linkable to Bob). Then, Bob and the set of other users who received similar keys create *each* a new key and broadcast them to be signed by the third party. To complete the transaction, each user must then "redeem" the key he had originally received by sending it back to the third party. Because the new keys are processed in a batch and each user sends the old key and requests the new one at *two different moments* and through broadcasting, the third party cannot link the user owning a certain initial key to the user who has requested a certain new key. Hence, unlike [6] and other MIX-net systems, users never send pseudonyms *and* personal information in the same message.

In the rest of this section we provide a formal description of the protocol.

2.2 Assumptions

We make the following assumptions (their strength will be discussed in Sections 3 and 5):

1. There exist several "sending" parties S_Is, with $I = 1, ..., N$, each trying to complete a transaction t_Y of type T that requires such party to send messages to other parties.

2. There are several "receiving" parties R_Is, with $I = 1, ..., T$, that sending parties want to complete transactions of type T with.

3. There is a third party M, able to send and receive messages, and able to perform a function F on a message (I, T) sent by a sending party S_I. Such function F maps a message (I, T) sent by S_I to a binary outcome $(0, 1)$. It is: $F(I, T) : (I, T) \rightarrow (0, 1)$, where $F(I, T) = 1$ means that the data I included in the message meets certain criteria set by the third party to allow the sending party to receive the credentials to complete a transaction of type T.

4. All parties can send and receive messages to and from other parties, and can perform basic cryptographic operations (like creating cryptographic keys, signing or encrypting and decrypting messages and keys using public key cryptosystems, such as [22]). In particular, the parties are able to create keys Ks which are random and long enough to be unique with high probability, and the third party is able to create "credentials" for transactions of type T by signing the keys sent by the sending parties with a private key used for that type of transaction. The third party can also keep track of the keys it has signed.

5. All parties participate to a network D. There exists some communication channel inside the network through which the parties exchange information. Parties can broadcast messages to all parties in D.

6. Parties can also broadcast anonymously (see [24]), in the sense that receivers of a message broadcast anonymously cannot identify the sender of the message.[1]

7. There exists a Log party L that logs all communications going to and coming from the third party M.

The goal of the protocol is to let sending parties S_Is complete transactions t_Y "anonymously" with receiving parties R_Is after exhibiting their credentials. "Anonymously" means that no party is able to find a correspondence between a sending party S_I and the transaction t_Y (or the receiving party R_I) completed by that sending party. Hence, no party is able to determine S_I from t_Y and R_I or viceversa.

2.3 Notation

The following notation is adopted in the description of the protocol:

- $E_X\{.\}$ means that message (.) has been encrypted with key X.
- $E_{C\_tPB}\{.\}$ means that message (.) has been encrypted with the public key t of C.

[1] See 5.

- $E_{C\_dPR}\{.\}$ means that message (.) has been signed with the private key d of C.
- $A \rightarrow B : t$ represents the communication of t from A to B.
- $A \rightarrow * : t$ represents a broadcast communication of t from A to all other parties.
- $A? \rightarrow * : t$ represents an anonymous broadcast communication of t from A to all other parties.
- T (in caps) represents a category or "type" of transaction (for example, a voting transaction, or a payment transaction). t_Y (in small caps, with $y = 1, ..., W$) represents a specific instance of transaction of type T (for example, a particular casted vote). Hence $t_Y \epsilon T$.

2.4 Steps

The protocol is composed of six steps:

1. $S_I \rightarrow M : E_{M\_PB}\{E_{S_{I\_PR}}\{(I, T)\}\}$

   A sending party S_I that wants to complete a transaction of type T sends a message (I, T) to the third party M. The third party operates a function F that maps (I, T) to a binary outcome $(0, 1)$ (Assumption 3). If $F(I, T) = 1$, it means that the data I included in the message includes data that meet the criteria set by the third party for allowing the sending party S_I to perform a transaction of type T.

   If $F(I, T) = 1$ (Assumption 3), M creates a long random key KT_{I_1} and signs it (Assumption 4). This signed key represents a non-anonymous credential for a transaction of type T. In other words, KT_{I_1} would allow S_I to complete a transaction of type T, altough *not* anonymously. M sends the signed key to S_I:

2. $M \rightarrow S_I : E_{S_{I\_PB}}\{E_{M\_PR}\{KT_{I_1}\}\}$
3. $S_I? \rightarrow *_D : E_{M\_PB}\{KT_{I_2}, KT_{I_3}\}$

   Every S_I broadcasts anonymously (Assumption 6) to the network D two newly created long random keys. Each sending party wants the second key signed by M and then encrypted with the third key, so that only the sending party that created it can decrypt the signed key. Under Assumption 6, the keys are such that given S_I, M cannot determine K_{I_2} or KT_{I_3}; and given KT_{I_2} or KT_{I_3}, M cannot determine S_I.

   M gathers the pairs of second and third keys broadcast by all sending parties and defines:

$$\bar{KT} = \left[E_{KT_{1_3}}\left\{E_{KT_{1_2}}\right\}, ..., E_{KT_{I_3}}\left\{E_{KT_{I_2}}\right\}, ..., E_{KT_{N_3}}\left\{E_{KT_{N_2}}\right\}\right]$$

Then, M creates an ID for $\bar{KT}$ called *idlist* and broadcasts:

4. $M \rightarrow *_D : E_{M\_PR}\left\{\text{idlist},\bar{KT}\right\}$

   Each sending party S_I, upon recognizing his key pairs in the broadcast message, sends a signed message to M quoting the *idlist* and returning the original key KT_{I_1}.

5. $S_I \rightarrow M : E_{M\_PB}\left\{E_{S_{I\_PR}}\left\{\text{idlist},KT_{I_1}\right\}\right\}$

   If and only if M receives as many KT_{I_1}s as key pairs associated to *idlist*, then it can proceed to sign each KT_{I_2} with the associated KT_{I_3} and then broadcasts the message:

6. $M \rightarrow *_D : E_{M\_PR}\left\{E_{KT_{I_3}}\left\{E_{M\_PR}\left\{E_{KT_{I_2}}\right\}\right\}\right\}$

Each sending party has now received a new credential KT_{I_2}, based on a key he chose, signed by M. This credential can be used to complete a transaction of type T (for example, the credential could represent a payment token to pay for an anonymous purchase). Because from S_I M cannot determine KT_{I_2} or KT_{I_3} (see step 3 above and Assumption 6), therefore M cannot determine KT_{I_2} given KT_{I_1} either. Therefore, S_I now owns a key that will allow him to complete a transaction t_y of type T with a receiving party R_I, without the receiving party (or anybody else) being able to determine S_I from t_y, QED.

3 Anonymity, Reliability, Acidity, and Other Properties

In this section we discuss some key properties of the protocol. In Section 4 we discuss related technologies, and in Section 5 we analyze possible attacks.

The protocol is based on the synchronous activities of parties that do not know or trust each other. Users of the protocol choose keys and have them signed by a third party, in a way that no other party can determine the identity of an user from the key it has received. Anonymity is protected because M cannot determine KT_{I_2} given KT_{I_1}, and therefore no other party can determine S_I from t_y.

This form of anonymity relies on Assumption 6 - the existence of an anonymous broadcasting channel. [24] argue that anonymous broadcasting is a more accurate model of what can actually happen during anonymous communications, and that it can provide efficient implementation techniques for several anonymity-related protocols. We note here that even in the absence of an anonymous broadcasting channel, each user can take advantage of two different "pseudo-identities" to send messages 1 and 3 respectively

and achieve the same outcome. Each pseudo-identity is an identity that an user can adopt to contact the third party, and that the third party can use to contact the user: for example, an email address, or an account on a computer shared by several users.[2] Users may also send messages through email mixers such as [14]. Even in this latter case, unlike [6]'s application to electronic voting through digital pseudonyms and unlike traditional MIX-nets, in this protocol users never need to send to the third party their proposed pseudonyms (KT_{I_2}) and the information required by the authority for the acceptance decision (I) *in the same message.*

The existence of a Log L does not undermine anonymity even in case of collusion with M. The Log is only used to prevent or resolve disputes and ensure atomicity.[3] A global observer able to intercept all messages in the network D cannot compromise anonymity as long as Assumption 6 is maintained. On the other side, intersection attacks on the anonymity set (see [23]) are possible. We discuss these attacks in Section 5.

Since a single MIX is sufficient to ensure anonymity, there is no need for a cascade of MIXes. This enhances the reliability of the system. In addition, transactions completed through the single MIX satisfy ACID properties (atomicity, consistency, isolation, and durability - see [17] and [5]). Since M needs to receive as many KT_{I_1}s as key pairs associated to *idlist* in order to sign all the credentials in a batch, each batch is either collectively completed or aborted. If too many new credentials are requested (or too few valid KT_{I_1}s are redeemed), M aborts the transaction after the chosen delay period has elapsed since step 2. M must then request users to start again from step 3 by issuing new key-pairs to be signed. (Potential delay of service and other attacks on this feature are discussed in Section 5.) Transactions are consistent because, when a batch is aborted, the keys KT_{I_1} that certain users attempted to redeem in step 5 are still valid and can be reused: the Log can prove that a certain KT_{I_1} was sent for a batch that was aborted. L can also verify that M does not issue signed keys without corresponding legitimate users, or that M is not aborting a batch even if it has received the correct number of valid keys KT_{I_1}s.[4] If L stores the information it gathers by observing communications involving M, durability is also achieved. Finally, because each batch (and the associated keys) are identified by an unique *idlist*, transactions are also isolated.

[2]We note that, in this case, the requirements are not different from those involving the actual spending of anonymous cash - such as ECash ([7]): the analysis of the traffic between a customer and the party blindly signing the customer's tokens, and between the same customer and the merchant where the tokens are spent may also undermine anonymity if the customer is not using different pseudo-identities. See also [5].

[3]See below.

[4]Collusion between L and M disrupts atomicity but cannot undermine anonymity. In addition, this form of collusion would be easily detected by other users. Hence we rule out this case by also noting that L would be an independent authority or a tamper-proof device (see also [5]).

The above description does not specify what type of transaction T the sending party wants to complete. Given that the information I needed for the third party's acceptance decision is customized for the transaction T, and given that users choose their keys KT_{I_2}s and M observes the keys before signing them, the acid mixing primitive can be used for several applications - such as anonymous payments, electronic voting, anonymous communications, and so on. We discuss some of these applications in Section 6.

4 Related Technologies

Acid mixing combines elements from various strands of the cryptographic literature: MIX-nets ([6], and related applications such as [14] and [21]); ANDOS protocols (e.g., [3]); group signatures ([10]); and the "cocaine auction" protocol (see [24]).

In terms of functionalities, the protocol presented here can be related to other anonymous credentials protocols (such as [7], [8], [9], [11], [2], [18], and [4]) as well as to protocols based on blind signatures ([7]) that also produce unlinkable signed certificates.

MIX-nets [6] have been discussed above. The commonalities between MIX-nets and the approach described here lie in the presence of a third party that mixes keys from several users. The method presented here, however, addresses some trust and reliability issues in MIX-net systems.

The "All-Or-Nothing-Disclosure-Of-Secrets" (or "ANDOS;" see [3]) protocols allow "secrets" to be sent to unidentifiable recipients. Group signatures (see [10]) protocols allow an unidentifiable member of a group to sign on behalf of the group. The assumptions and features of these two approaches are quite different from those of the protocol presented here. Group signatures do not offer the same level of anonymity provided by the method described here. ANDOS protocols require in general more steps, messages, and complexities than our protocol. However, some elements are also in common. In particular, all these protocols rely on different parties acting as members of a group in order to make each particular member unidentifiable.

[7], [8], [9] describe anonymous credential systems which all rely on trusted or semi-trusted third parties that transfer credentials between different organizations - unlike the untrusted third party described in this protocol. [18] and [2] propose credentials system that do not rely on trusted parties, but are based on functions and proofs (for example, zero-knowledge proofs) that make their use unpractical.

[4] propose a credential systems that can be used in applications such as anonymous payments. Their system has a novel feature: an "all-or-nothing" share of credentials, under which a party that allows another party to use one of his credentials, loses control on *all* of his other credentials. Some of the goals of the protocol we present are comparable to those described in [4]:

producing keys which are "credentials" that strangers can exhibit to act in environments where parties do not know and trust each other. The protocol presented here, however, is based on fewer messages and simpler cryptographic operations (only 4 messages are exchanged for each user, and two additional messages are broadcasted by M to the network D for the entire batch). More generally, acid mixing offers a different strategy for anonymity and different functionalities compared to the anonymous credential systems discussed above or to blind signatures-based systems (through [7]'s blind signatures, a sending party can manipulate data signed by a third party in a way that the third party's signature can still be recognized but not linked to the data originally signed). First, under our protocol, the collusion of the third party with all other $N-1$ sending parties is in general required in order to expose the connection between a certain N-th sending party S_I and the K_{I_2} he requested. This means that the collusion of the very same users of the system is needed in order to attack it. In other words, in the protocol described here there is no cryptographic relation between the exchanged keys, but only among the participants to the method itself. Second, unlike the protocols described by [3], [10], or [4], this method allows sending parties to choose the keys they want the third party to sign, and it allows the third party to see the actual keys before signing them, while preserving anonymity. These features becomes useful when the method is used in applications such as electronic voting (see Section 6).

5 Adversary Model and Attacks

A global adversary who can observe all communications in D does not compromise the anonymity of the system. An adversary controlling $N-1$ users, however, does. More generally, in this protocol the anonymity of each sending party depends on his synchronously blending and hiding his messages (and keys) with those of other sending parties. Hence, anonymity depends on the entropy of the "anonymity set" (see [23]) that the N sending parties form as a group, and is exposed to intersection attacks on these sets. Hence the best use of the protocol is for synchronous systems such as elections or for systems with large numbers of transactions such as payment systems or communication systems with many users.

If the third party impersonates or colludes with all other $N-1$ parties, it compromises the anonymity of the N-th party (as in any anonymity system where all participants except one are colluding). When more than one batch is used in a certain application of the protocol, the third party (who knows that a certain user is in the batch) may, in fact, try to select the batches in order to decrease the entropy of the anonymity set where the user is trying to hide his transactions. One possible defense is to force the third party to cluster all parties in the same batch (if the transactions are synchronous, as

they may be in an election), or select parties for batches randomly, with a procedure similar to the one proposed in [13]. Another defense is to allow the user himself to select his peers and form groups. Such groups may still contain users colluded with the third party, although now with reduced probability.[5]

Denial of service attacks may be implemented by malicious users who flood the system with requests of new credentials they are not able to satisfy, because they do not have (or are not willing to send) the original KT_{I_1} keys. In fact, if and only if M receives as many KT_{I_1}s as key pairs associated to *idlist*, then it can proceed to sign each KT_{I_2} with the associated KT_{I_3}. However, given that transactions are ACID, a certain batch can be aborted with no consequence if the right number of keys have not been received. Still, while this form of attack would bring no gain to the attacker, it may nevertheless slow or halt the third party. Hence, to further disincentive attackers, their strategy can be made inordinately costly by asking a monetary, non anonymous deposit at step 1 and by providing - in return - in step 2 a group key for the batch, that the user must add to this message in step 3. The deposit is reimbursed after the batch transaction is completed or aborted, but only users who left a deposit are able to see the group key needed to request the credentials KT_{I_2}s to be validated.

6 Applications

Acid mixing allows users to reliably obtain anonymous credentials from a MIX they do not trust. Since the nature of the information I that must be provided by users to obtain a credential can change with the transaction T, and given that the credentials KT_{I_2}s can be chosen by the users and observed by M, the primitive can be used for several applications. A few examples are discussed below.

Anonymous payments. I contains financial information that M needs in order to charge the sending party's personal account. The anonymous credentials KT_{I_2}s, once signed by M, represent payment tokens that the sending party can use with a merchant.

Electronic voting. I contain information that proves the sending party's eligibility to vote in a certain election. The write-in ballot vote is expressed through the keys KT_{I_2}s. After the batch is complete, M can publish the keys. Hence the method allows for write-in ballots *and* universally verifiable votes (see [1]).

[5]For additional protection, users may also engage in the protocol iteratively. Once obtained the signed keys KT_{I_2}, users can resume the protocol from step 3, in order to obtain yet again new keys by sending back the ones just received, rather than using the latter to complete a transaction. Through these repetitions of the protocol, "old" keys are "redeemed" into new ones, in order to cloud further the relation between each original sending party and the signed keys it obtains.

Anonymous communications. I contains a message, encrypted with the receiving party's public key, that the sending party wants to send anonymously. The keys KT_{I_2}s contain the addresses of the receiving parties. M prepares a batch of the messages and forward the batch to the addresses listed in the keys KT_{I_2}s - a receiving party will only be able to decrypt the part of the batch meant for her.

References

[1] Acquisti, A. Write-in ballots in universally verifiable elections. Working Paper, 2003.

[2] Brands, S. *Rethinking Public Key Infrastructure and Digital Certificates - Building in Privacy.* MIT Press, Cambridge, MA, 2000.

[3] Brassard, G., Crepeau, C., and Robert, J.-M. All-or-nothing disclosure of secrets. In *Advances in Cryptology - Crypto '86* (1987), Springer Verlag, LNCS 263, pp. 234–238.

[4] Camenisch, J., and Lysyanskaya, A. An efficient system for non-transferable anonymous credentials with optional anonymity revocation. In *Advances in Cryptology - EUROCRYPT '01* (2001), Springer-Verlag, LNCS 2045, pp. 93–118. http://citeseer.nj.nec.com/camenisch01efficient.html.

[5] Camp, J., Harkavy, M., Tygar, J. D., and Yee, B. Anonymous atomic transactions. In *USENIX Workshop on Electronic Commerce* (1996), pp. 123–133. citeseer.nj.nec.com/camp96anonymous.html.

[6] Chaum, D. Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM 24*, 2 (1981), 84–88.

[7] Chaum, D. Blind signatures for untraceable payments. In *Advances in Cryptology - Crypto '82* (1983), Plenum Press, pp. 199–203.

[8] Chaum, D. Security without identification: Transaction systems to make big brother obsolete. *Communications of the ACM 28* (1985), 1030–1044.

[9] Chaum, D., and Evertse, J.-H. A secure and privacy-protecting protocol for transmitting personal information between organizations. In *CRYPTO - '86* (1985), pp. 118–167.

[10] Chaum, D., and van Hejist, E. Group signatures. In *Advances in Cryptology - EUROCRYPT '91* (1991), Springer Verlag, LNCS 547, pp. 257–265.

[11] Chen, L. Access with pseudonyms. In *Cryptography: Policy and Algorithms* (1995), E. Dawson and J. Golie', Eds., Springer-Verlag, LNCS 2045, pp. 232–243.

[12] Dingledine, R., Freedman, M. J., Hopwood, D., and Molnar, D. A reputation system to increase MIX-net reliability. In *Information Hiding - IH '01* (2001), I. Moskowitz, Ed., Springer-Verlag, LNCS 2137, pp. 126–141. http://www.freehaven.net/papers.html.

[13] Dingledine, R., and Syverson, P. Reliable MIX cascade networks through reputation. In *Financial Cryptography - FC '02* (2002), M. Blaze, Ed., Springer Verlag, LNCS (forthcoming). http://www.freehaven.net/papers.html.

[14] Goldschlag, D., Reed, M., and Syverson, P. Onion routing for anonymous and private internet connections. *Communications of the ACM 42*, 2 (1999), 39–41.

[15] Gulcu, C., and Tsudik, G. Mixing email with BABEL. In *Symposium on Networked and Distributed System Security* (1996).

[16] Hirt, M., and Sako, K. Receipt-free voting based on homomorphic encryption. In *Eurocrypt* (2000).

[17] Lynch, N., Merritt, M., Weihl, W., and Fekete, A. *Atomic Transactions*. Morgan Kaufmann, San Mateo, CA, 1994.

[18] Lysyanskaya, A., Rivest, R. L., Sahai, A., and Wolf, S. Pseudonym systems. In *Proceedings of the Sixth Annual Workshop on Selected Areas in Cryptography (SAC '99)* (Cambridge, MA, 1999), H. Heys and C. Adams, Eds., Springer Verlag LNCS 1758.

[19] Mazières, D., and Kaashoek, M. F. The design, implementation and operation of an email pseudonym server. In *Computer and Communications Security - CCS '98* (1998), ACM Press.

[20] Pfitzmann, A., Pfitzmann, B., and Waidner, M. ISDN-mixes: Anonympous communication with very small bandwidth overhead. In *GI-ITG Conference: Communicartion in Distributed Systems* (1991), pp. 451–462.

[21] Reiter, M. K., and Rubin, A. D. Anonymous web transactions with Crowds. *Communications of the ACM 42*, 2 (1999), 32–38.

[22] Rivest, R. L., Shamir, A., and Adleman, L. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM 21* (1977), 120–126.

[23] Serjantov, A., and Danezis, G. Towards an information theoretic metric for anonymity. In *Privacy Enhancing Technologies - PET '02* (2002), P. Syverson and R. Dingledine, Eds., Springer Verlag, LNCS (forthcoming).

[24] Stajano, F., and Anderson, R. J. The cocaine auction protocol: On the power of anonymous broadcast. In *Information Hiding Workshop* (1999), Springer Verlag, LNCS 1768, pp. 434–447.