Dining Cryptographers with 0.924 Verifiable Collision Resolution

Christian Franck

Abstract. The dining cryptographers protocol implements a multiple access channel in which senders and recipients are anonymous. A problem is that a malicious participant can disrupt communication by deliberately creating collisions. We propose a computationally secure dining cryptographers protocol with collision resolution that achieves a maximum stable throughput of 0.924 messages per round and which allows to easily detect disruptors.

1 Introduction

Protocols for untraceable communication have received much attention recently as they can help us protect our privacy and avoid cyber espionage. The aim of these protocols is not to encrypt messages but to prevent an attacker from determining who is communicating with whom.

The dining cryptographers protocol [3] is the most secure protocol for untraceable communication known in computer science. This multi-party protocol implements a multiple access channel in which senders and recipients of messages remain anonymous. Unlike other primitives like mixes [4] and onion routing [9,7], it does not require a trusted third party and it is not vulnerable to network based attacks like traffic shaping.

The problem is that messages collide when multiple senders attempt to transmit a message at the same time. Even worse, malicious participants can disrupt the communication by deliberately creating collisions all the time. Such disruptors are hard to identify because the anonymity of the honest senders must be preserved.

Recent computationally secure variants of the dining cryptographers protocol use an anonymous reservation phase in order to avoid collisions and a technique based on zero-knowledge proofs to detect disruptors [8,6,5]. However, the implementation of such an anonymous reservation phase is complicated and reservations do not adapt well to situations where participants are frequently joining or leaving the group.

The present paper shows that one can address collisions as they occur using a collision resolution algorithm and still prevent disruption by a malicious participant. First, we show that with a modified SICTA collision resolution algorithm a maximum stable throughput (MST) of 0.924 packets per round can be achieved for the dining cryptographers protocol. Then, we show that it is possible to use zero-knowledge proofs to verify that each participant properly executes the collision resolution algorithm.

Compared to existing techniques our approach is easier to implement as there is no a reservation phase. Further, it adapts better to situation where participants are joining and leaving. We see possible applications in the fields of electronic voting and low latency anonymous communication.

The rest of the paper is organized as follows. Section 2 contains preliminaries and definitions. In section 3, we discuss collision resolution with SICTA. In section 4 we show how disruptors can be detected. In section 5 we discuss related work, in section 6 we present possible applications, and we conclude in section 7.

2 Preliminaries

In this section, we briefly review the principle behind the dining cryptographers protocol and a technique to implement it efficiently.

Dining Cryptographers

In one round of the dining cryptographers protocol [3], every participant broadcasts a ciphertext (O), which may or may not contain a message (M). (To keep the description simple, we assume that the participants have reliable broadcast channels at their disposal.) The encryption vanishes when the ciphertexts of all participants are combined (e.g., $C := \prod_i O^{(i)}$). If exactly one ciphertext contains a message, then this message appears (e.g., $C = M$). However, there is a collision when several ciphertexts contain a message (e.g., $C = M \cdot M' \cdot M''$). We assume that messages are encoded with a checksum, so that it is possible to distinguish between a message and a collision of messages.

Generation of Ciphertexts

We generate ciphertexts as described by Golle and Juels in [10]. The advantage of this technique, which is based on the Diffie-Hellman key agreement, is that after a single setup phase the participants can generate ciphertexts for a large number of rounds. To this effect, participants share finite groups $H = \langle h \rangle$ and $G = \langle g \rangle$ and a bilinear map $e : H \times H \to G$ such that $e(h^a, h^b) = e(h,h)^{ab} = g^{ab}$ for $a, b \in \mathbb{Z}$. The Bilinear Decisional Diffie-Hellman (BDDH) problem is assumed to be hard in H and G. Each participant has a private key x and the corresponding public keys $\bar{y} = h^x$ and $y = g^x$ are known to all participants.

In a round j of the protocol, each participant then generates a ciphertext $O_j \in G$ which has an algebraic structure. This means that O_j is either of the form

$$O_j = {A_j}^x$$

or of the form

$$O_j = {A_j}^x M,$$

wherein $x \in \mathbb{Z}$ is a secret key and $M \in G$ is a message. As we have the BDDH assumption, one cannot distinguish whether O_j contains a message M or not.

The value A_j is public and it is based on the public keys $\bar{y} = h^x$ of the other participants and on a public random value R_j which is different in every round. For example, n participants $P^{(1)}, ..., P^{(n)}$ compute

$$A_j^{(i)} = e\left(\prod_{k=1}^{i-1} \bar{y}^{(k)} \prod_{k=i+1}^{n} 1/\bar{y}^{(k)}, R_j\right),$$

The so obtained values $A_j^{(i)}$ are different in each round and have the property that they cancel when they are multiplied. I.e.,

$$\prod_{k=1}^{n} \left(A_j^{(k)}\right)^{x^{(k)}} = 1.$$

Therefore, only messages remain when a recipient multiplies the ciphertexts $O_j^{(1)}...O_j^{(n)}$ provided by all the participants.

3 Collision Resolution with SICTA

In this section we explain the SICTA algorithm (Successive Inference Cancellation Tree Algorithm) [14] and show that in the context of the dining cryptographers protocol we can reach throughput to 0.924 messages per round.

Collision Resolution

Assume in a round j each participant provides a ciphertext O_j. If several of these ciphertexts contain a message, the combination of all these ciphertexts $C_j := \prod_{i=1}^{n} O_j^{(i)}$ only provides a multiplication of all the messages and no meaningful information is transmitted. The purpose of a collision resolution algorithm is to resolve such a collision by resending the involved messages in later rounds.

SICTA is a binary tree algorithm, in which a collision of messages is repeatedly split until all messages have been transmitted. When there is a collision in one round, two subsequent rounds are dedicated to the resolution of this collision. Each message involved involved in the collision is then retransmitted at random in one of two dedicated rounds. This process is repeated recursively until all collisions are resolved. An example of a SICTA collision resolution tree is shown in Figure 3.1. To simplify the description we adapt our notation to binary trees; when a collision that occurs in round j we assume that the rounds $2j$ and $2j+1$ are dedicated for the resolution. SICTA uses a technique called inference cancellation to reduce the number of transmissions. As we have $C_j = C_{2j} \cdot C_{2j+1}$, it is not necessary to transfer any O_{2j+1} for round $2j+1$. The value C_{2j+1} can be inferred from C_j and C_{2j} by computing $C_{2j+1} = C_j/C_{2j}$. For this inference cancellation to work, the algorithm operates in blocked access mode, which means that no new message may be sent until all collisions are resolved.

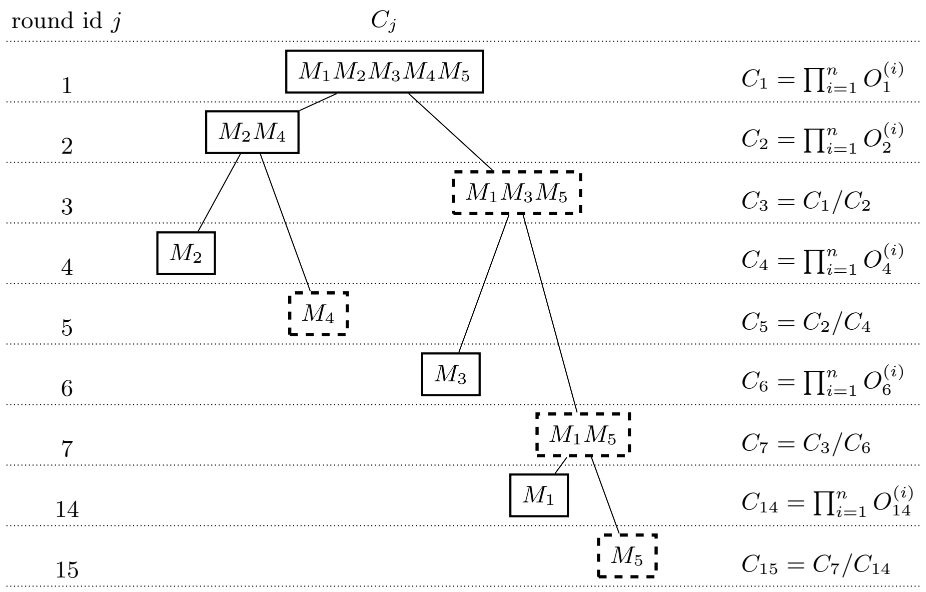

Fig. 3.1. Exemplary binary collision resolution tree with successive inference cancellation (SICTA). In rounds 1,2,4,6 and 14, ciphertexts O_j are transmitted, and C_j is computed using these ciphertexts. In rounds 3,5,7 and 15, no data is transmitted and C_j is computed using data from the parent and the sibling node.

Performance

Let us consider the maximum stable throughput (MST), which denotes the maximal input rate (messages/round) for which all messages have a finite delay. Therefore we define S_k as the average number of rounds needed to resolve a collision of k messages, and we consider the throughput k/S_k.

A collision of k messages is split into two collisions with i and $k-i$ messages with a probability $\binom{k}{i}2^{-k}$. Thus we have

$$S_k = \sum_{i=0}^{k} \binom{k}{i} 2^{-k}(S_i + S_{k-i}).$$

With $\binom{k}{i} = \binom{k}{k-i}$ this can be written as

$$S_k = \sum_{i=0}^{k} \binom{k}{i} 2^{1-k} S_i$$

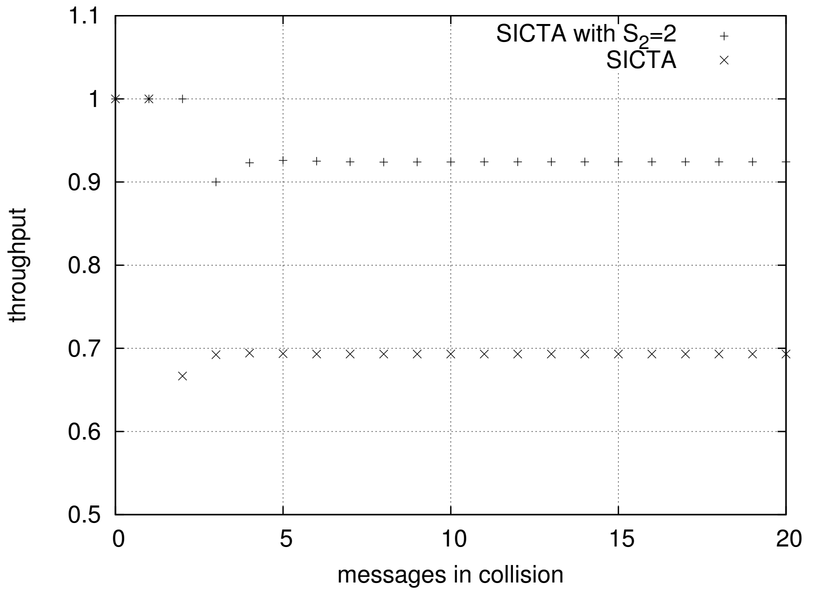

Fig. 3.2. Performance of collision resolution with SICTA.

and after removing the recursion we obtain

$$S_k = \frac{2^{1-k}}{1-2^{1-k}} \sum_{i=0}^{k-1} \binom{k}{i} S_i.$$

As 'collisions' with 0 or 1 messages take only 1 round, we have $S_0 = S_1 = 1$. The throughput k/S_k for increasing values of k is shown in Figure 3.2. We observe for SICTA the known MST of 0.693.

We can achieve a higher throughput by exploiting the fact that in the dining cryptographers protocol all senders are also receivers. After a collision of two messages, the two respective senders can recover each other's message by removing their own from the collision. They can then avoid a further collision by using a rule that for instance only the numerically smaller message is resent. This way, collisions of two messages are always resolved in two rounds. I.e., we have $S_2 = 2$, which leads to a MST of 0.924.

So we have just computed the possible throughput of the channel and seen that efficient collision resolution is possible. We have done this under the assumption that every participant is honest and that no disruption takes place. This assumption is reasonable, as we show in the next section that disruptors can easily be detected and eliminated from the group. Being exceptional events, disruptions have no impact on the asymptotic (number of rounds $\to \infty$) behavior of the channel.

4 Detecting Disruptors

In this section, we show that disruptors are easy to detect. We first present techniques using zero-knowledge proofs to prove statements about the retransmission of messages, and then we show how these techniques can be used to verify that each participant correctly performs the SICTA algorithm.

Zero-Knowledge Proofs for the Retransmission of Messages

It was shown in [10] that the algebraic structure of the ciphertexts makes it possible to prove statements about them using zero-knowledge proofs. Such a zero-knowledge proof allows a prover to prove to a verifier that a given statement holds, without giving the verifier any further information. I.e., the verifier cannot compute anything that he could not have computed before. One can for instance prove the equality of discrete logarithms to different bases, and logical $\wedge$ (and) and $\vee$ (or) combinations of such statements [1]. It is also possible to prove the inequality of logarithm to different bases [2].

Existing zero-knowledge proofs used in dining cryptographers protocols contain statements about individual ciphertexts. E.g., the statement

$$\log_{A_1}(O_1) = \log_g y$$

holds when ciphertext O_1 is empty (i.e., $O_1 = A_1^x$). As a reminder, x is a secret key of the participant and $y = g^x$ the corresponding public key.

To verify the correct execution of the SICTA collision resolution protocols we use a new kind of statements, which hold when there is a relation between two or more ciphertexts coming from the same participant. E.g., the statement

$$\log_{A_1/A_2}(O_1/O_2) = \log_g y$$

holds when both ciphertexts O_1 and O_2 encode the same message M (or when both encode no message). It is thus possible to construct more complex statements in order to verify the retransmission of a message.

Example 1. The ciphertext O_2 either contains no message, or the same message as O_1, when the statement

$$\Big(\log_{A_2/A_1}(O_2/O_1) = \log_g y\Big) \vee \big(\log_{A_2}(O_2) = \log_g y\big)$$

holds.

Example 2. At most one ciphertext out of $O_2, ..., O_k$ contains the same message as O_1 (and the rest of $O_2, ..., O_k$ contain no message), when the statement

$$\Big(\log_{A_2 ... A_j/A_1}(O_2 ... O_j/O_1) = \log_g y\Big) \vee \Big(\log_{A_j}(O_j) = \log_g y\Big)$$

holds for $j \in \{2, ..., k\}$. (Note that it is not sufficient to consider only the last statement, as a participant could encode $O_2 = A_2^x E$ and $O_3 = A_3^x E^{-1}$ instead of $O_2 = A_2^x$ and $O_3 = A_3^x$. In the multiplication $O_2 O_3 ...$, the factors E and E^{-1} would cancel, and the statement would hold. It is therefore necessary to consider each statement for $j \in \{2, ..., k\}$.)

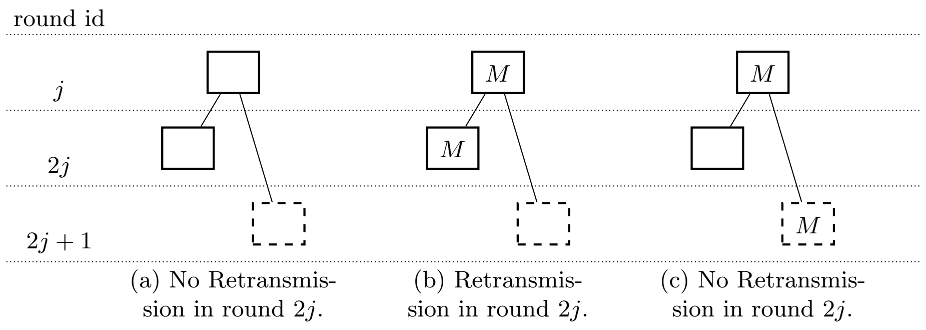
round id
j
$2j$
$2j+1$
M
M
M
M
(a) No Retransmission in round $2j$.
(b) Retransmission in round $2j$.
(c) No Retransmission in round $2j$.

Fig. 4.1. Collision resolution in SICTA. Only a message involved in a collision in round j may be retransmitted either in round $2j$. The round $2j+1$ is virtual; no transmission takes place. No new message may be sent until all collisions are resolved.

Verification of Standard SICTA with a MST of 0.693

We now show how the techniques from the previous section can be used by the participants to prove that they correctly executed the collision resolution algorithm, without revealing if they are sending a message or not (so that the senders of the messages remain anonymous).

Correct participation in the standard SICTA algorithm means that a participant may only retransmit a message in round $2j$ if he already transmitted that message in round j. Remember that SICTA operates in blocking mode an that no new message may be sent until the resolution has finished. This principle, which is illustrated in Figure 4.1, means that

– if a participant transmits $O_j = A_j^x$ in round j, then he must transmit $O_{2j} = A_{2j}^x$ in round $2j$; and
– if a participant transmits $O_j = A_j^x M$ in round j, then he must transmit either $O_{2j} = A_{2j}^x$ or $O_{2j} = A_{2j}^x M$ in round $2j$.
(Note that sending no message equals to sending $M = 1$. If the participant did not send a message in round j, this means that $O_j = A_j^x \cdot 1$. The participant then has the 'choice' between retransmitting $O_{2j} = A_{2j}^x$ or $O_{2j} = A_{2j}^x \cdot 1$ in round $2j$. I.e., he is forced to retransmit $O_{2j} = A_{2j}^x \cdot 1 = A_{2j}^x$ and may not send a message.)

Using the techniques from the previous section, each participant can prove that his ciphertext O_{2j} is correct, without revealing if whether it contains a message or not. To do this, the participant generates a zero-knowledge proof that proves that

$$\Big(\log_{A_j/A_{2j}}(O_j/O_{2j}) = \log_g y\Big) \vee \Big(\log_{A_{2j}}(O_{2j}) = \log_g y\Big) \tag{4.1}$$

holds. With this proof he can convince a verifier that he participated correctly, without compromising the anonymity of the protocol.

As described before, SICTA is a recursive algorithm and there are virtual rounds during which C_j is inferred, but no corresponding O_j is transmitted. It is then not possible to prove statement (4.1), but luckily it is still possible to prove that O_{2j} is correct. To do this, the participant proves that a message contained in the nearest transmitted parent round was transmitted at most once in all the branches down to O_{2j}. Akin to Example 2, a participant proves that

$$\Big(\log_{A_{j_t/2}/A_{j_1}\ldots A_{j_t}}(O_{j_t/2}/O_{j_1}\ldots O_{j_t}) = \log_g y\Big) \vee \Big(\log_{A_{2j}}(O_{2j}) = \log_g y\Big)$$

holds, wherein $j_1 := 2j$, $j_k := (j_{k-1}/2) - 1$ and t such that $j_t/2$ is the index of the nearest transmitted parent round of round j.

Example 3. In the collision resolution process shown in Figure 3.1, each participant shows for O_2 that

$$\Big(\log_{A_1/A_2}(O_1/O_2) = \log_g y\Big) \vee \big(\log_{A_2}(O_2) = \log_g y\big)$$

holds, then for O_4 that

$$\Big(\log_{A_2/A_4}(O_2/O_4) = \log_g y\Big) \vee \big(\log_{A_4}(O_4) = \log_g y\big)$$

holds, then for O_6 that

$$\Big(\log_{A_1/A_2A_6}(O_1/O_2O_6) = \log_g y\Big) \vee \big(\log_{A_6}(O_6) = \log_g y\big)$$

holds, then for O_{14} that

$$\Big(\log_{A_1/A_2A_6A_{14}}(O_1/O_2O_6O_{14}) = \log_g y\Big) \vee \big(\log_{A_{14}}(O_{14}) = \log_g y\big)$$

holds.

So we have shown that a participant can prove in zero-knowledge that he properly participates in the standard SICTA collision resolution algorithm. Any participant who is not able to prove that his output is correct can be excluded from the group. The corresponding round is lost and must be repeated by the remaining participants.

Additional Verification for Optimized SICTA with a MST of 0.924

The special technique we described in section to increase the MST from 0.693 to 0.924 uses a deterministic rule for the retransmission after a collision of two messages. E.g. only the message with the lower value must be retransmitted. So we need additional verification to detect participants that do not respect this rule. We cannot verify this with a single zero-knowledge proof, but we can for instance use the following approach.

If one of the two participants involved in the collision does not respect this rule, the other one can switch back to random retransmission in order to split the collision. Once the collision has been split and the two messages are out, everybody sees that there must have been a problem in a previous round, and an investigation can be started. The message M of the cheating participant is now known, and every participant must then for instance send a zero-knowledge proof that he did not send this message during the initial collision round (i.e. prove that $\log_{A_1} O_1/M \neq \log_g y$). The disruptor will not be able to come up with an appropriate proof and can be eliminated from the group.

Further Minor Security Considerations

Malicious participants may attempt to delay the collision resolution process or to prevent it from terminating. For instance,

- colluding participants can always chose the same round to retransmit their messages, or
- a malicious participant can wait until all other participants have transmitted and then choose to retransmit his message so that a collision occurs, or
- a malicious participant may not send a valid message in the first place.

However, such malicious behavior is easy to detect. In the previously described SICTA algorithm with a MST of 0.924, the probability that a collision does not split is less than or equal to 1/4 (it is exactly 1/4 for collisions with 3 messages). Thus, the probability that a collision does not split k times in a row is less than or equal to $1/4^k$. E.g., the probability that a collision does not split 5 times in a row is below 0.1%. When such malicious activity is detected, one can require commitment before transmission and one can use zero-knowledge proofs similar to the ones proposed in [10] to detect participants that are frequently involved in non-splitting collisions. If a lower throughput is acceptable, one can go for a simpler approach and just skip the branches of the resolution tree that do not split after several attempts, without trying to detect the malicious participants.

5 Related Work

Superposed receiving [12,13] is a collision resolution technique for the dining cryptographers protocol that achieves throughput of 100%. Therein, messages are elements of an additive group. When a collision occurs, the average of the messages values is computed and only messages whose value is less than this average are retransmitted. Like in SICTA, inference cancellation is used, which leads to the 100% throughput. However, this approach requires the use of an additive finite group and it cannot be implemented using the algebraic ciphertexts that we need for efficient ciphertexts generation and for zero-knowledge proofs.

A fully verifiable dining cryptographers protocol was proposed in [8] and rediscovered in [5]. In this protocol, we have 100% throughput. However, there is the need for a reservation phase which can be lengthy and cumbersome. Current systems are using mixnets to perform the reservations and therefore they are inefficient when only a few reservations are made. Further, they do not easily adapt to situations where participants join or leave frequently.

6 Applications

Our protocol can be used to implement computationally secure anonymous communication channels with a low latency. Another application is the realization of secret shuffle algorithms (e.g. [11]). A secret shuffle algorithm is used to obtain a shuffled list of values from a plurality of participants, while keeping it secret which value is coming from which participant. Existing solutions typically require each participant to submit a value. The protocol proposed herein also works efficiently if only a few participants have a value to submit. In particular it may be used to shuffle anonymous public keys for verifiable dining cryptographers protocols in which rounds are reserved [8,5].

7 Concluding Remarks

The main problems of the dining cryptographers protocol are collisions and malicious participants disrupting the communication.

We have shown that with a collision resolution algorithm it is possible to achieve a maximum stable throughput of up to 0.924 messages per round. Further, we have shown that if we use ciphertexts with an algebraic structure as proposed in [10], we can verify in zero-knowledge that each participant properly retransmits his message during the collision resolution process.

Compared to other dining cryptographer protocols, our approach does not need a reservation phase to avoid collisions. It is therefore easier to implement and it adapts more naturally to situations where participants are frequently joining and leaving the group.

We see possible applications in the fields of low-latency anonymous communication and secret shuffling.

References

1. J. Camenisch and M. Stadler. Proof systems for general statements about discrete logarithms. *Technical Report TR 260, Institute for Theoretical Computer Science, ETH Zurich*, Mar. 1997.
2. Jan Camenisch and Victor Shoup. Practical verifiable encryption and decryption of discrete logarithms. In *Advances in Cryptology-CRYPTO 2003*, pages 126–144. Springer, 2003.
3. D. Chaum. The dining cryptographers problem: Unconditional sender and recipient untraceability. *Journal of Cryptology*, 1(1):65–75, 1988.
4. D.L. Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM*, 24(2):84–88, 1981.
5. H. Corrigan-Gibbs, D. I. Wolinsky, and B. Ford. Proactively accountable anonymous messaging in verdict. In *USENIX Security*, 2013.
6. Henry Corrigan-Gibbs and Bryan Ford. Dissent: accountable anonymous group messaging. In *Proceedings of the 17th ACM conference on Computer and communications security*, pages 340–350. ACM, 2010.

7. R. Dingledine, N. Mathewson, and P. Syverson. Tor: the second-generation onion router. *Proceedings of the 13th conference on USENIX Security Symposium-Volume 13 table of contents*, pages 21–21, 2004.
8. C. Franck. New Directions for Dining Cryptographers. Master's thesis, University of Luxembourg, Luxembourg, 2008.
9. D. Goldschlag, M. Reed, and P. Syverson. Hiding Routing Information. *LECTURE NOTES IN COMPUTER SCIENCE*, pages 137–150, 1996.
10. P. Golle and A. Juels. Dining Cryptographers Revisited. *Advances in cryptology-EUROCRYPT 2004: International Conference on the Theory and Applications of Cryptographic Techniques, Interlaken, Switzerland, May 2-6, 2004: Proceedings*, 2004.
11. C Andrew Neff. A verifiable secret shuffle and its application to e-voting. In *Proceedings of the 8th ACM conference on Computer and Communications Security*, pages 116–125. ACM, 2001.
12. A. Pfitzmann. How to implement ISDNs without user observability – Some remarks. *ACM SIGSAC Review*, 5(1):19–21, 1987.
13. M. Waidner. Unconditional Sender and Recipient Untraceability in spite of Active Attacks. *Lecture Notes in Computer Science*, 434:302, 1990.
14. Yingqun Yu and Georgios B Giannakis. Sicta: a 0.693 contention tree algorithm using successive interference cancellation. In *INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE*, volume 3, pages 1908–1916. IEEE, 2005.