Abstract. The dining cryptographers protocol provides information-theoretically secure sender and recipient untraceability. However, the protocol is considered to be impractical because a malicious participant may disrupt the communication. We propose an implementation which provides information-theoretical security for senders and recipients, and in which a disruptor with limited computational capabilities can easily be detected.

1 Introduction

The dining cryptographers protocol [3] implements a multiple access channel in which senders and recipients are anonymous. In one round of the protocol, each participant broadcasts a ciphertext ($O$), which may or may not contain a message ($M$). The encryption vanishes when the ciphertexts of all participants are combined (e.g., $\sum_i O_i$). If exactly one ciphertext contains a message, then this message appears (e.g., $\sum_i O_i = M$).

However, there is a collision when several ciphertexts contain a message (e.g., $\sum_i O_i = M + M' + M''$). A problem of the protocol is that a malicious participant may disrupt the communication by deliberately creating collisions in each round. The identification of such a disruptor is difficult, as an investigation may compromise the anonymity of honest senders.

In this paper, we propose a way to implement a verifiable collision resolution algorithm, which allows to verify the correct participation of every participant, without compromising anonymity. In our approach, we use superposed receiving [7,8] in conjunction with specially constructed Pedersen commitments and zero-knowledge proofs. When there is no disruption, we have an optimal throughput of one message per slot. Misbehaving participants can be identified during an investigation phase. We see possible applications in the fields of anonymous communication and secret shuffling.
2 Preliminaries

Let $G$ denote a group of large prime order $q$ with generators $g$ and $h$, in which the discrete logarithm problem is assumed to be hard. It is further assumed that $\log_h g$ is not known.

We use a special kind of Pedersen commitments. A Pedersen commitment $c \in G$ to a value $x \in \mathbb{Z}_q$ is of the form

$$c = g^x h^r,$$

wherein $r \in \mathbb{Z}_q$ is typically chosen at random. A Pedersen commitment is unconditionally hiding and computationally binding. Unconditionally hiding means that nobody can find $x$ if the commitment is not opened by providing $(x, r)$. Computationally binding means that it is computationally hard to come up with another pair $(x', r')$, such that $c = g^{x'} h^{r'}$.

3 Phases of the Protocol

In this section, we describe the five phases of the protocol. We have

- a setup phase,
- a commitment phase,
- a transmission phase,
- a collision resolution phase, and
- a verification phase.

We assume $N$ participants $\mathcal{P}_1 \ldots \mathcal{P}_N$ that have a reliable broadcast channel at their disposal.

3.1 Setup

In the setup phase, the participants $\mathcal{P}_1 \ldots \mathcal{P}_N$ establish shared secret keys, and compute commitments for these keys.

First, each pair of participants $\mathcal{P}_i$ and $\mathcal{P}_j$ secretly choses random keys $K_{ij}, K'_{ij}$ and $K_{ji} = -K_{ij}, K'_{ji} = -K'_{ij}$ in $\mathbb{Z}_q$.

Then, each participant $\mathcal{P}_i$ computes the commitments

$$c_{ij} := g^{K_{ij}} h^{K'_{ij}} \text{ and } c_{ji} := g^{K_{ji}} h^{K'_{ji}}.$$
for each \( j \). These commitments, which can be opened by \( P_i \) and \( P_j \), are unconditionally hiding and computationally binding. Further they have the property that \( c_{ij} \cdot c_{ji} = 1 \) holds.

Finally, each participant \( P_i \) sends for each \( c_{ji} \) a signature \( S_i(c_{ji}) \) to the corresponding participant \( P_j \). The receiving participant \( P_j \) verifies if the signature is correct. (The case where a participant refuses to provide a valid signature is discussed later in section 4.)

After these steps, each participant \( P_i \) has a signed commitment \( S_j(c_{ij}) \) for each of his secret keys \( K_{ij} \). Further, we define the commitment \( c_i \) for \( \sum_j K_{ij} \) as

\[
c_i := g^{\sum_j K_{ij}} h^{\sum_j K'_{ij}}
\]

so that

\[
c_i = \prod_j c_{ij}.
\]

A participant \( P_i \) is now able to prove that \( c_i \) is correct by showing the commitments \( c_{i1}...c_{iN} \) and the signatures \( S_1(c_{i1})...S_N(c_{iN}) \).

**Remark 1.** Each round of the protocol requires a new commitment. Instead of signing each individual commitment, one can use a Merkle tree to sign multiple commitments at once.

### 3.2 Commitment

Each participant \( P_i \) publishes the commitment \( c_i \). After receiving all commitments \( c_i \), each participant verifies if

\[
\prod_{i=1}^{N} c_i = 1
\]

holds. If (3.1) does not hold, then one of the participants cheated and an investigation phase is started. In this investigation phase, each participant \( P_i \) publishes his commitments \( c_{i1}...c_{iN} \) and his signatures \( S_1(c_{i1})...S_N(c_{iN}) \), and then performs the steps of:

1. verifying the signatures \( S_i(c_{ji}) \) of the commitments \( c_{ij} \);
2. verifying that \( c_i = \prod_j c_{ij} \) holds for all \( i \); and
3. verifying that \( c_{ij} \cdot c_{ji} = 1 \) holds for all \( c_{ij}, c_{ji} \).

At least one participant who provided a wrong signature, a wrong \( c_i \) or a wrong \( c_{ij} \) is detected in this process. This malicious participant can then be excluded from the group of participants.
If \((3.1)\) holds, the commitments \(\hat{c}_i\) are correct and the actual transmission can take place. Each participant \(P_i\) publishes a ciphertext \(O_i\), of the form
\[
O_i = X_i + M_i
\]
to send a message \(M_i\), or of the form
\[
O_i = X_i
\]
to send no message. The value \(x_i\) is defined as
\[
X_i = \sum_j K_{ij}.
\]

### 3.4 Collision resolution

We use superposed receiving \([7,8]\) to resolve collisions. This collision resolution technique achieves an optimal throughput of 1 message per transmission slot. As illustrated in Figures 3.1 and 3.2 it is a tree based collision resolution algorithm. When a collision occurs in round \(k\), two rounds rounds \(2k\) and \(2k+1\) are used to 'retransmit' the involved messages. Actually, the outcome of round \(2k+1\) is
Fig. 3.2. Exemplary binary collision resolution tree with superposed receiving. In rounds 1, 2, 4, 6 and 14, ciphertexts $O^{(k)}$ are transmitted, and $C^{(k)}$ is computed using these ciphertexts. In rounds 3, 5, 7 and 15, no data is transmitted and $C^{(k)}$ is computed using data from the parent and the sibling node.

Inferred using the outcome of the rounds $k$ and $2k$, and thus no real transmission takes place for round $2k + 1$.

For superposed receiving, a tuple of the form $(1, m)$ is encoded in a message $M$, such that when 2 messages $M$ and $M'$ collide in round $k$, then $(2, m + m')$ is received. The senders of involved messages then compute the average, e.g. $(m + m')/2$, and retransmit in round $2k$ only if their message is below this average.

This collision resolution is performed in blocked access mode, which means that no new message may be sent until all collisions are resolved. I.e., as illustrated in Figure 3.3 only a message that was sent in round $k$ may be retransmitted in round $2k$. We can ensure that a participant does not infringe this rule, as we see next.
round id

\[ k \]

\[ 2k \]

\[ 2k + 1 \]

(a) No message. (b) Retransmit left. (c) 'Retransmit' right.

Fig. 3.3. Retransmission in superposed receiving. Only message involved in the in the collision in round \( k \) may be retransmitted either in round \( 2k \). No new message may be sent during the collision resolution process.

3.5 Verification

We can ensure that a participant either retransmits no message or the same message using zero-knowledge proofs. The commitment \( c_i \) is computationally binding \( P_i \) to the value

\[ X_i := \sum_j K_{ij}. \]

Instead of opening a commitment \( c_i \), we can use it like an algebraic pad in \([6]\). That is, we can use a commitment \( c_i \) to construct a zero-knowledge proof.

Zero-knowledge proofs allow a prover to prove to a verifier that he knows a witness which verifies a given statement, without giving the verifier any other information. A system for proving general statements about discrete logarithms was presented in \([2]\). One can for instance prove the knowledge of a discrete logarithm, the equality of discrete logarithms with different bases, and logical \( \land \) (and) and \( \lor \) (or) combinations thereof. In our notation based on \([1]\), secrets are represented by greek symbols. For instance, we write a proof of knowledge of the discrete logarithm of \( y \) to the base \( g \) as \( \mathcal{PK}\{\alpha : y = g^\alpha \} \).

We can for instance construct a zero-knowledge proof for an individual ciphertext. E.g., the proof

\[ \mathcal{PK}\{\alpha : (c_i^{(2k)} / g_{i}^{(2k)} = h^\alpha) \} \]
proves that the ciphertext $O_i^{(2k)}$ (of round $2k$) is empty (i.e., when $O_i^{(2k)} = X_i^{(2k)}$ then $\mathcal{P}_i$ knows $\alpha = \sum_j K_{ij}^{(2k)}$). We can also construct statements that hold when there is a relation between two or more ciphertexts. E.g., the proof

$$\mathcal{PK}\{\alpha : (c_i^{(2k)} c_i^{(2k)-1} / g^{O_i^{(2k)}-O_i^{(2k)}} = h^\alpha)\}$$

can be used to prove that both ciphertexts $O_i^{(k)}$ and $O_i^{(2k)}$ encode the same message $M$ or that both ciphertexts $O_i^{(k)}$ and $O_i^{(2k)}$ encode no message. The proof works because a message $M$ contained in the ciphertexts $O_i^{(k)}$ and $O_i^{(2k)}$ cancels in the difference $O_i^{(k)} - O_i^{(2k)}$. In this case $\mathcal{P}_i$ knows $\alpha = \sum_j K_{ij}^{(k)} - \sum_j K_{ij}^{(2k)}$.

Thus, to prove that a retransmission in round $2k$ was correctly performed, we use $O_i^{(k)}$ and $O_i^{(2k)}$ and the proof

$$\mathcal{PK}\{\alpha : (c_i^{(2k)} / g^{O_i^{(2k)}} = h^\alpha) \lor (c_i^{(k)} c_i^{(2k)-1} / g^{O_i^{(k)}-O_i^{(2k)}} = h^\alpha)\}$$

holds.

When no parent $O_i^{(k)}$ exists; i.e., when no transmission took place and the outcome of round $k$ was computed, then it is necessary to consider the siblings up to the last transmitted parent node. This is illustrated in the following example.

**Example 1.** In the collision resolution process shown in Figure 3.2 a participant proves for $O_2$ that

$$\mathcal{PK}\{\alpha : (c_i^{(2)} / g^{O_i^{(2)}} = h^\alpha) \lor (c_i^{(1)} c_i^{(2)-1} / g^{O_i^{(1)}-O_i^{(2)}} = h^\alpha)\}$$

holds, then for $O_4$ that

$$\mathcal{PK}\{\alpha : (c_i^{(4)} / g^{O_i^{(4)}} = h^\alpha) \lor (c_i^{(2)} c_i^{(4)-1} / g^{O_i^{(2)}-O_i^{(4)}} = h^\alpha)\}$$

holds, then for $O_6$ that

$$\mathcal{PK}\{\alpha : (c_i^{(6)} / g^{O_i^{(6)}} = h^\alpha) \lor (c_i^{(1)} c_i^{(2)-1} c_i^{(6)-1} / g^{O_i^{(1)}-O_i^{(2)}-O_i^{(6)}} = h^\alpha)\}$$

holds, then for $O_{14}$ that

$$\mathcal{PK}\{\alpha : (c_i^{(14)} / g^{O_i^{(14)}} = h^\alpha) \lor (c_i^{(1)} c_i^{(2)-1} c_i^{(6)-1} c_i^{(14)-1} / g^{O_i^{(1)}-O_i^{(2)}-O_i^{(6)}-O_i^{(14)}} = h^\alpha)\}$$

holds.
This shows that it is possible to verify that a participant retransmitted his message in only one branch of the tree. Further problems that may occur and the corresponding countermeasures are discussed in the next section.

4 Possible Attacks

An attacker may attempt to disrupt the protocol in several ways, but this is easy to handle.

A first problem may occur during the setup phase. A malicious participant could refuse to agree on shared secret keys with another participant. I.e., the malicious participant could choose not to provide a proper signature to that participant. In such a situation, the participant who does not obtain the signature can publicly announce that he wants to drop the keys between himself and the malicious participant. These keys are then all set to zero, and no commitments and signatures are required for these keys.

A second attack could consist in not retransmitting in the correct branch of the tree. I.e. according to superposed receiving, only those messages with a value below the average should be retransmitted. A malicious node could do just the opposite, and thus cause a new collision. In such a case, where a collision does not split, it is possible to fall back to probabilistic splitting of collisions. I.e. each participant randomly chooses to retransmit or not. This allows to separate the honest nodes from the malicious ones. After this separation has taken place it is possible to determine the malicious nodes.

Finally, a participant could just choose not to submit a message of the proper format \((1, m)\) in order to make the collision resolution fail. But here again it is possible to fall back to a probabilistic algorithm. If a collision does not split after several attempts in the probabilistic algorithm, then the involved participants must be colluding, and all of these can be considered as malicious.

5 Related work

One technique to detect disruptors in an information-theoretically secure dining cryptographers protocol is based on trap rounds [35].
The idea is that when a disruption is detected, an investigation phase is started, in which each participant must reveals his secret keys. The problem of this technique is that it would also reveal the identity of a honest sender. Therefore transmission rounds are anonymously reserved, and some of them are used as trap rounds, in which no message is sent. If a participant detects that one of his trap rounds was disrupted, he reveals that it was trap and an investigation phase is started. A first problem of this technique is that it is complicated and lengthy. A further problem is that if a round containing a message is disrupted, then the disrupter can not be detected.

More recent techniques \cite{6,5,4} allow a more efficient detection of disruptors, but they do not provide information-theoretically secure untraceability. This is risky for application where a high level of security is required for many years. As the field of cryptography is permanently evolving, and as the computational capabilities are permanently increasing, one cannot say for sure for how long the anonymity will hold. A transmission that is considered to be secure today may be recorded and broken in a few years from now.

6 Applications

Applications exist in the fields of low-latency untraceable communication and secret shuffling. The advantage of our approach is that the anonymity of senders and recipients is information-theoretically secure.

7 Concluding Remarks

We have shown that it is possible to implement an information-theoretically secure dining cryptographers protocol in which malicious participants with limited computational capabilities are easy to detect. Unlike previous approaches, we do not rely probabilistic detection of disruptors.

References

1. J. Camenisch and M. Stadler. Efficient Group Signature Schemes for Large Groups. *LECTURE NOTES IN COMPUTER SCIENCE*, pages 410–424, 1997.
2. 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.

3. D. Chaum. The dining cryptographers problem: Unconditional sender and recipient untraceability. Journal of Cryptology, 1(1):65–75, 1988.

4. H. Corrigan-Gibbs, D. I. Wolinsky, and B. Ford. Proactively accountable anonymous messaging in verdict. In USENIX Security, 2013.

5. C. Franck. New Directions for Dining Cryptographers. Master’s thesis, University of Luxembourg, Luxembourg, 2008.

6. 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.

7. A. Pfitzmann. How to implement ISDNs without user observability – Some remarks. ACM SIGSAC Review, 5(1):19–21, 1987.

8. M. Waidner. Unconditional Sender and Recipient Untraceability in spite of Active Attacks. Lecture Notes in Computer Science, 434:302, 1990.