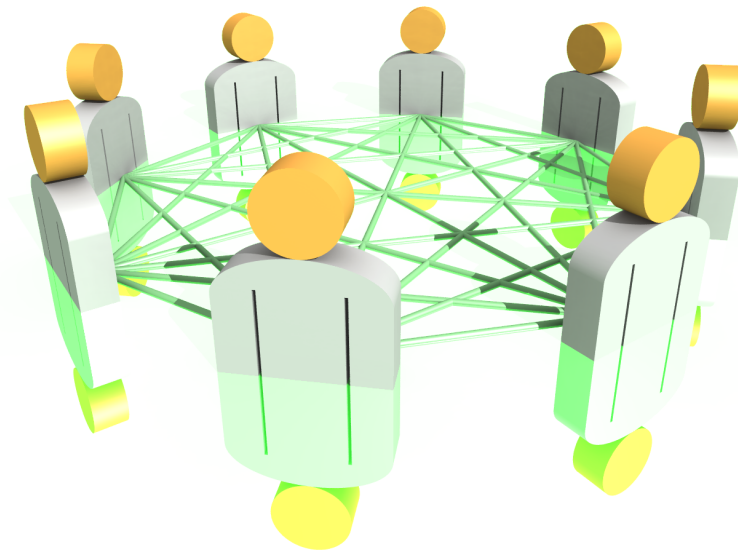


New Directions for Dining Cryptographers

Computationally Secure Sender and Recipient Untraceability



Christian Franck

New Directions for Dining Cryptographers

by Christian Franck

Submitted in partial fulfillment of the requirements for the degree of
Master of Science in Information and Computer Sciences
at the University of Luxembourg

First Supervisor: Prof. Dr. Thomas Engel

Second Supervisor: Prof. Dr. Ulrich Sorger

Assistant: Dipl. Inform. Volker Fusenig

November 2008

Abstract

In the dining cryptographers protocol [7], a malicious participant can disrupt communication by providing a wrong output. Existing techniques only allow to limit the damage to some extent. This is a major obstacle to the practical usage of the protocol. The objective of this thesis is to provide a better response to this problem. Our starting point are the ideas presented by Golle and Juels in [21], where outputs of algebraic structure are generated based on the Diffie-Hellman key exchange [12], and where non-interactive zero-knowledge proof techniques allow to prove the correctness of these outputs. We present verifiable superposed sending, a computationally secure technique which allows participants to generate verifiable outputs. We use a reservation phase which provides participants with a digital pseudonym of the sender. They use this digital pseudonym to generate an output of algebraic structure by directly establishing a shared secret with the sender. A honest participant can prove that his output is correct with respect to the reservation. The dining cryptographers protocol becomes a practicable alternative to mixnets, especailly when strong anonymity is required, and it has the advantage that it can transfer messages with lower latency. We present possible applications in the fields of electronic voting, low latency anonymous communication and secure multiparty computation.

Acknowledgements

I would like to thank my first supervisor Prof. Dr. Thomas Engel, his assistant Dipl. Inform. Volker Fusenig, and last but not least my tutor and my second supervisor Prof. Dr. Ulrich Sorger, for allowing me to work on this interesting topic, for the interesting discussions, and for all the kind support during the realisation of this work!

Contents

1	Introduction	4
2	Anonymous Communication	7
2.1	Anonymity and Untraceable Communication	7
2.2	Concepts for Online Communication	10
2.3	Attacker Models, Trust and Strong Anonymity	10
2.4	Mixnets	11
2.5	Summary	12
3	The Dining Cryptographers	14
3.1	The Story	14
3.2	Superposed Sending	16
3.2.1	Key Establishment	18
3.2.2	Reservation	18
3.2.3	Detection of Disrupters	20
3.3	Reliable Broadcast	23
3.4	Bandwidth Usage	24
3.5	Summary	25
4	A Verifiable Dining Cryptographers Protocol	26
4.1	Verifiable Superposed Sending	26
4.1.1	Purpose	26
4.1.2	Preliminaries	27
4.1.3	The Short DC-Net Protocol	27
4.1.4	Remaining Problems	29
4.1.5	Our Solution	29
4.1.6	Long Messages	31
4.1.7	Variations	31

4.1.8	Discussion	31
4.2	Reservation	32
4.2.1	What We Need	32
4.2.2	A Mixnet based Reservation	33
4.3	Effects on the Broadcast	34
4.4	Applications	34
4.4.1	Electronic Voting	35
4.4.2	Low Latency Anonymous Communication	35
4.4.3	Secure Multi-Party Computation	40
4.5	Summary	40
5	Conclusions	42
A	Cryptographic Primitives	46
A.1	Computationally Hard Problems	46
A.2	Diffie-Hellman Key Exchange	47
A.3	Public Key Encryption Schemes	47
A.3.1	El Gamal Encryption	47
A.3.2	Homomorphic Encryption	48
A.3.3	Public Key Signature Schemes	48
A.3.4	El Gamal Signature Scheme	48
A.3.5	Aggregate Signatures	49
B	Proofs of Knowledge	50
B.1	Zero-Knowledge Proofs of Knowledge	50
B.2	Sigma Protocols	50
B.3	Proving Statements about Discrete Logarithms	51

Chapter 1

Introduction

In this chapter, we discuss the aims and goals of this thesis and briefly discuss what is coming in the following chapters.

What is this Thesis About?

Anonymous communication is important to protect privacy and freedom of speech. Several concepts have been invented to allow anonymous communication over a public network. The strongest known concept is the dining cryptographers protocol, presented by Chaum in [7]. It guarantees unconditional (information theoretical) sender and recipient untraceability, but it is very inefficient and therefore considered to be unpractical.

As illustrated in Figure 1.1, the participants P_1, P_2, \dots, P_n respectively announce the outputs O_1, O_2, \dots, O_n . These outputs are indistinguishable from random outputs, but they allow to compute an anonymous message M , typically according to the formula: $M = O_1 \oplus O_2 \oplus \dots \oplus O_n$. The anonymous message appears but the sender and the recipient remain unknown.

A major obstacle to the practical application is that malicious participants may disrupt communication by providing wrong outputs. Current

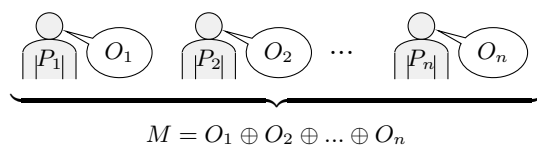


Figure 1.1: The dining cryptographers

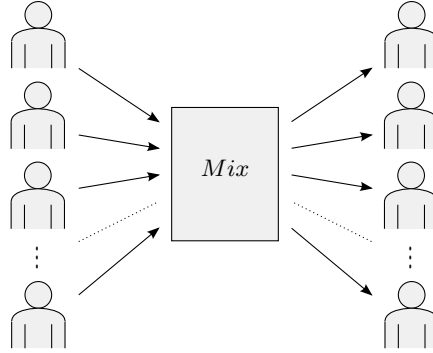


Figure 1.2: Chaum's mix

techniques for the detection of these disrupters are complicated and only limit the disruptions to a certain extent.

The goal of this thesis was use cryptographic techniques to realize a computationally secure dining cryptographers protocol, in which malicious participants who disrupt the communication are easy to detect.

Why is this Interesting?

Currently, most anonymisation systems that are used in practice are based on mixes. This concept was proposed by Chaum in [9]. As illustrated in 1.2, many senders send their messages to a common mix which shuffles the messages and forwards them to the receivers. This way it is difficult to link a message to a sender. Mixes are used because they are efficient, and they scale to a large number of participants.

However, the problem is that this mix concept is weak. First, participants must trust the mix owner, who knows which participant is the sender of a message. Furthermore, the messages of all participants must be equal in size. Otherwise a global observer may deduce who is the sender of a message, by comparing the size of the messages entering and leaving the mix.

Over the last years, many efforts have been made to create strong anonymisation systems based on this weak concept. Typically, efficiency is sacrificed to obtain more security. Multiple mixes are combined to a mixnet, in which messages are arranged into packets of equal size, and then mixed multiple times before reaching a receiver. However, these new constructions have again new weaknesses and there is like a competition between new attacks

and new designs.

I believe that in order to realize a strong anonymisation system, it is not a good approach to start from a weak and efficient concept and then to try to make it stronger (at the cost of efficiency). I think it is better to start from a very strong and inefficient concept, and then to try to make it more efficient (and a bit weaker).

Therefore this thesis starts from the dining cryptographers protocol, which is very strong and very inefficient, and then tries to make it more efficient. Unlike existing approaches, wherein a message is routed through several small dining cryptographer networks [28][15][31][18], this thesis is about improving the dining cryptographers protocol itself.

What is the Proposed Solution?

The key contribution of this work is the presentation of verifiable superposed sending, a new technique which allows participants of the dining cryptographers protocol to generate verifiable outputs. We use a new kind of reservation phase which provides all participants with a digital pseudonym¹ of the sender. Based on the ideas presented by Golle and Juels in [21], zero-knowledge proof techniques allow honest sender to prove that their outputs are correct, without who is the sender.

Organisation

This document is organised as follows:

The first chapter introduces the subject of the thesis, the motivation, and provides an brief outlook to what can be found in the rest of this document.

The second chapter provides the reader with some background on anonymity, untraceable communication and mixnets.

The third chapter presents the dining cryptographers protocol, explains the underlying techniques and the related problems.

The fourth chapter presents our new technique "verifiable superposed sending", and possible applications.

The fifth and last chapter contains a conclusion and indicates possible future work that can be done in this area.

¹A digital pseudonym is a public key, and the holder of the corresponding private key is anonymous. The term was introduced by Chaum in [9].

Chapter 2

Anonymous Communication

This chapter is an introduction to anonymity and untraceable communication. We present existing concepts for untraceable communication, and explain what we expect from a protocol to obtain strong anonymity. We conclude with an introduction to mixnets.

2.1 Anonymity and Untraceable Communication

We want anonymity to protect our privacy against people who could use personal information about us to harm us. This is particularly important in many critical areas like voting, freedom of speech, treatment of medical data, financial data and similar.

One possibility to achieve anonymity is to physically access the network in a way that nobody can associate the usage of the access point to your person. Internet cafés and public hotspots offer opportunities to access the web in an anonymous way. Prepaid SIM cards allow to log into the GSM network and into the internet anonymously. Unprotected LAN or WLAN access points can serve as anonymous entry points to the internet. However, most of these methods are expensive to realise and some are not legal. We would prefer to have anonymity while using our own internet connection. This is why we need protocols for untraceable communication.

When talking about untraceable communication, we assume that the different participants have names and IP addresses. So there will not be anonymity in a literal sense. Untraceability means that people within the group will be able to communicate in a way that nobody can tell who is

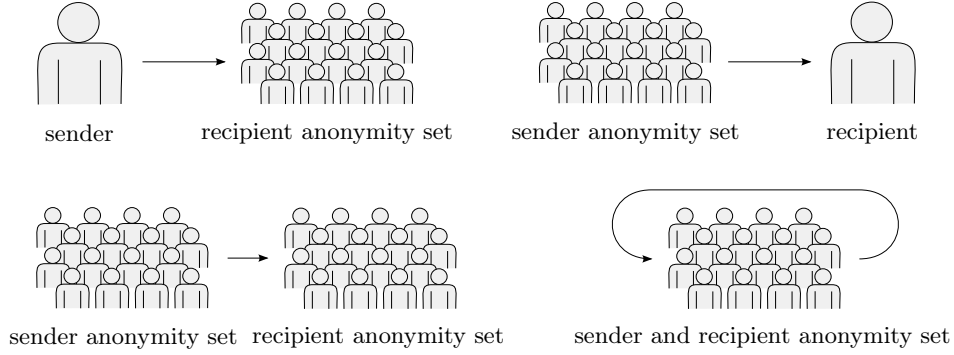


Figure 2.1: Different anonymity sets

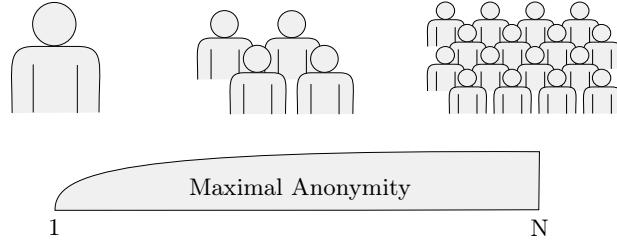


Figure 2.2: Maximal degree of anonymity

communication with whom. It should not be possible to follow the path of a message from one participant to another. This is the role of protocols for untraceable communication.

Anonymity Sets

In [29], we find the definition: "Anonymity is the state of being not identifiable within a set of subjects, the anonymity set". The requirement to have an anonymity set is something special that makes anonymity differ from other security properties like secrecy or authenticity. As illustrated in Figure 2.1, we can distinguish between sender and receiver anonymity sets. Often all participants are into both sets, to allow bidirectional anonymous communication. Anonymity is not an absolute value, and one can have more or less anonymity. The maximal amount of anonymity that can be reached is dependent on the size of the anonymity set, as illustrated in Figure 2.2. The more people there are, the more anonymity one can achieve.

Related Concepts

Pseudonymity Pseudonymity is the usage of a false name within a specific context. For example in internet chat rooms users will typically use a pseudonym. Using a false name is a form of anonymity in the sense that nobody will know your real name. But it will be known that everything that is done with the same pseudonym is normally done by the same person. This already gives a lot of information and can possibly be use by an attacker .

Plausible deniability Plausible deniability means that an action could also have been made by someone else or could not have happened at all. This concept is often used in the context of politics and military. In case of problems with the public opinion, it can be a good strategy for politicians to claim that they were not involved and to come up with a plausible scenario in which someone else was the author.

Fundamental Limitations and Other Problems

Size of the anonymity set If some action X was done anonymously, this means that nobody knows who has done X. However the group of all possible persons who could have done X is a finite number, and the degree of anonymity is related to the size of the group. This implies that there must be at least two entities in a group to achieve anonymity.

Collusion of malicious participants A problem is that if all the other members of the set work together, they can determine what a user is doing by exclusion. If several members of the set cooperate and share their information, this is called collusion.

Detection of misbehaving participants One fundamental problem of anonymous communication systems is the detection of misbehaving participants. These could try to disrupt the system by dropping packets, modifying packets or by generating nonsense traffic. Detection of the disrupters is difficult since everything is done anonymously.

2.2 Concepts for Online Communication

Mixes The mix concept was proposed by David Chaum in [9]. Multiple senders send their encrypted messages to a common mix, which decrypts and shuffles the messages, and forwards them to the receivers. This way it is difficult to link one message to one sender. This is illustrated in Figure 1.2. There is the possibility to add encrypted information to the message that will allow the receiver of the message to respond via a special return mix. This will not compromise the anonymity of the initial sender.

Crowds Crowds were presented by Reiter and Rubin in [30]. To become a member of a crowd, the user must register at a central server. The members of a crowd are called Jondos. A Sender creates a packet containing a message and the address of the intended recipient. This packet is then randomly forwarded from one Jondo to another, and finally sent to the recipient. Each intermediary Jondo throws a coin to decide if he should forward the message to another Jondo or to the recipient.

Onion routing Onion routing was presented by Goldschlag et al. in [19]. A sender sends his message through several onion routers. He selects a path and encrypts the message several times. Each onion router decrypts one layer, and forwards the packet to the next router. The messages is routed through all the onion routers and finally sent to the destination.

The dining cryptographers The strongest known concept is the dining cryptographers protocol, presented by Chaum in [7]. It was introduced in the introduction, and will be presented in detail in the next chapter. It guarantees unconditional (information theoretical) sender and recipient untraceability, but it is very inefficient and therefore considered to be unpractical.

2.3 Attacker Models, Trust and Strong Anonymity

Dolev-Yao Concepts like crowds and onion routing consider attackers to have only a restricted view of the network. For example, the attacker will be the destination node and can only see who is directly communicating with him. They do not resist a strong attacker, like in the Dolev-Yao model. In the Dolev-Yao attacker model [14], the attacker is the network. It is assumed

that every message that is send over the network is forwarded by the attacker. He can observe, create, drop or modify all the packages he wants. This model is often used to formally verify security protocols.

Trust Chaum’s mix concept only works, if the mix owner is assumed to be honest. Senders must trust the mix owner. If the mix is operated by an attacker, he knows everything. A possibility to approach (and eliminate) the problem of trust is to send message through a sequence of mixes. This is then called a mixnet, and we will see this in the next section.

Strong Anonymity Ideally we would like to have a system, which can guarantee anonymity against a Dolev-Yao attacker, without requiring trust in any third party. This is what we will call strong anonymity. The only concept that offers this level of security out of the box, is the dining cryptographers protocol. Unfortunately it is not practical as such, because it is too inefficient. Especially the problems related to malicious participants disrupting the communication are not solved in a satisfactory manner. Therefore systems used in practice are based on mixnets, that we will see next.

2.4 Mixnets

To avoid that senders must trust one single mix owner, often in practice a mixnet is used, wherein a set of messages is decrypted and shuffled consecutively by a set of mixes $Mix_1, Mix_2, \dots, Mix_k$. If there is one honest mix in the mixnet, messages will be shuffled and be untraceable. Trust can be completely eliminated, if each sender owns one mix. This gives each sender the opportunity to shuffle the whole set of messages, ensuring that the messages are honesty shuffled at least once. However, this is expensive in terms of bandwidth usage, and in terms of latency.

Evolution of Mixnets

The evolution of cryptography has lead to new types of mixnets. This has been a very active area of research in the last 20 years.

Early Mixnets In a mixnet based on Chaum’s initial approach, senders encrypt their messages using a public key cryptosystem. As a message must

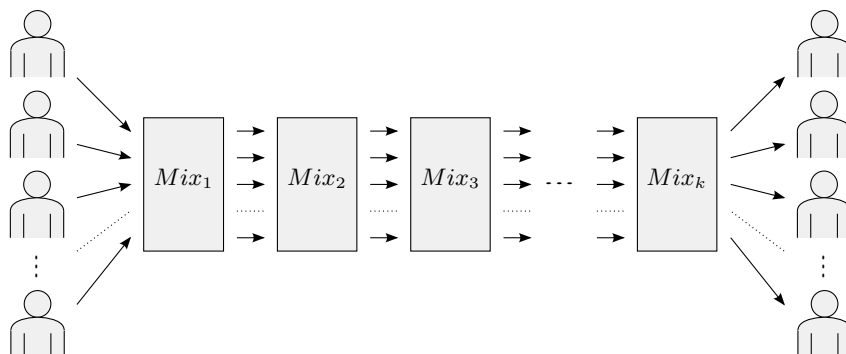


Figure 2.3: Mixnet

be encrypted once for each mix, messages are initially very large. On their way through the mixnet, one layer of encryption is removed at every mix, which makes the packages shrink at every mix. This is not very efficient, because the packages are initially be very large.

Reencryption Mixnets Reencryption mixnets use homomorphic encryption schemes, as described in appendix A.3.2, which allows to encrypt a message multiple times, without increasing the packet size. This principle was first presented by Park et al. in [25]. This requires less bandwidth, since the initial packets are kept small.

Verifiable Mixnets Verifiable Mixnets prevent a malicious mix server from manipulating the packets. Cryptographic proof techniques are used to construct a proof that the set of packets leaving a mix is a permutation and reencryption of the packets entering the mix. Zero knowledge proofs allow the verify that the packets were correctly mixed, without revealing the permutation. These types of mixnets are expensive in terms of computation.

2.5 Summary

We want anonymity to protect our privacy. Anonymity is the property of an element to be indistinguishable within a set, the anonymity set. Untraceable communication is communicating while making it impossible to follow the path of the message. Different concepts exist to realize untraceable communication. Some do not resist a strong attacker, others require trust in a third

party. The dining cryptographers protocol is the strongest known concept, but it is not used in practice. In a mixnet, a list of messages is sent through a sequence of mixes. At each mix, the messages are shuffled. Mixnets are currently the most widely used anonymisation systems.

Chapter 3

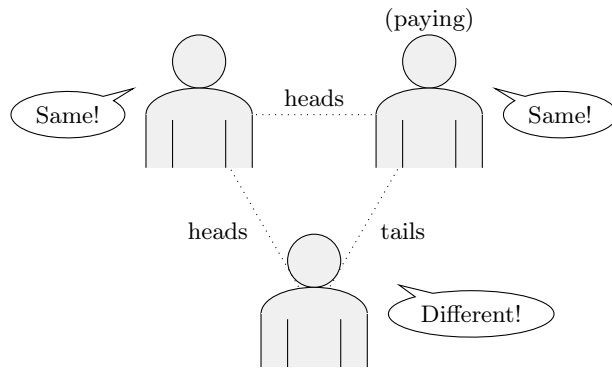
The Dining Cryptographers

The dining cryptographers protocol was described by Chaum in [7], and allows to achieve unconditional sender and recipient untraceability. This chapter presents the dining cryptographers protocol, and explains the underlying techniques. The different phases of a protocol are analysed, and the existing problems are explained.

3.1 The Story

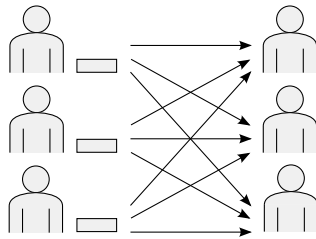
Chaum presents his protocol with the following story (Illustrated in Figure 3.1). Three cryptographers are dining in a restaurant, and the bill was paid anonymously. They want to find out if one of them paid, or if it was their employer, the National Security Agency. If it was one of them, they want this person to remain anonymous, and therefore they decide to run the following protocol. Each cryptographer flips a coin and shows it to his right neighbour. Then everybody compares the coin on his right and the coin on his left and announces "same" or "different". Normally, there should be an even number of people which announce "different". Now, if one of them is paying, he will lie and announce the opposite of what he normally would. This will result in having an odd number of participants announcing "different". If this is the case, everybody knows that one of them lied, but nobody can tell who is the liar. So this allows to determine if one of them paid, while keeping this person anonymous.

This story was used by Chaum to illustrate more general principles, which we will introduce next. The Dining Cryptographers protocol is based on

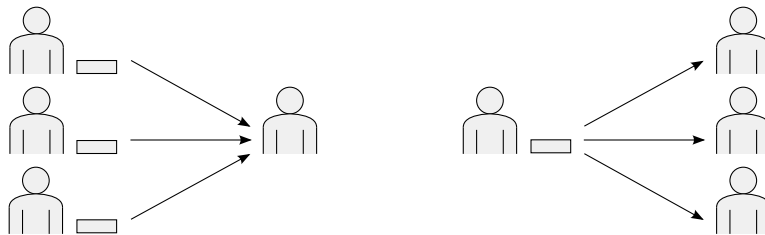


Odd number of "Different" → One of them is paying!

Figure 3.1: The Dining Cryptographers



Dining Cryptographers (sender and receiver untraceability)



Superposed Sending (sender untraceability) Reliable Broadcast (receiver untraceability)

Figure 3.2: Sender and receiver untraceability

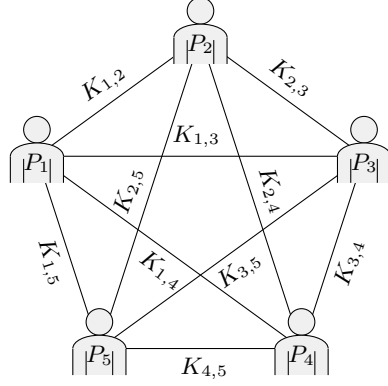


Figure 3.3: Complete key graph for 5 participants

two techniques: Superposed Sending for sender untraceability, and Reliable Broadcast for recipient untraceability. This is illustrated in figure 3.2.

3.2 Superposed Sending

The technique used to achieve sender untraceability is called superposed sending. It is described by Chaum in [7], by Pfitzmann in [28] and by Waidner in [34]. The participants P_1, P_2, \dots, P_n respectively announce their outputs O_1, O_2, \dots, O_n . Each value is indistinguishable from a random value, but the combination of all these values allows to obtain the anonymous message. The message M is obtained by a computation of the form: $M = O_1 \odot O_2 \odot \dots \odot O_n$. Which participant is the sender remains unknown.

Key graph For each round of superposed sending, pairs of participants must first establish secret keys. A key graph \mathbf{K} represents which participants share a secret key. The vertices of the graph are the participants, and the edges indicate which participants share a secret key.

$$\mathbf{K} = \{\mathbb{V}, \mathbb{E}\}, \text{ with } \begin{cases} \mathbb{V} = \{P_1, P_2, \dots, P_n\} \\ \mathbb{E} = \{\{P_i, P_j\} : P_i \text{ and } P_j \text{ share secret key } K_{i,j}\} \end{cases}$$

Figure 3.3 shows an example of a key graph with 5 participants. Any biconnected graph will work, but a complete graph offers the best protection against a collusion of malicious participants. A secret key $K_{i,j}$ is a random

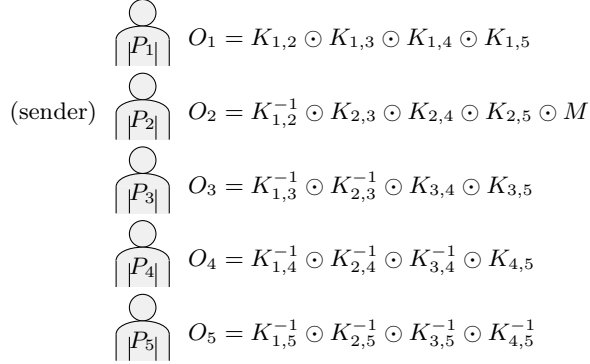


Figure 3.4: Output computation

element of a group (G, \odot) , and we define that $K_{j,i} = K_{i,j}$. We also define $\mathbb{S}_i = \{k : \{P_i, P_k\} \in \mathbb{E}\}$, which means \mathbb{S}_i is the set containing the indexes of the participants which share a secret key with participant P_i .

Computation of Outputs The participants compute their outputs based on the secret keys. The function $\text{sign}(x)$ is 1 for $x > 0$, and -1 otherwise. Each participant P_i , except the sender, computes his output O_i with:

$$O_i = \bigodot_{j \in \mathbb{S}_i} K_{i,j}^{\text{sign}(j-i)}$$

The sender P_i sends the message M . He computes his value O_i , using the secret keys he knows, according to:

$$O_i = \left(\bigodot_{j \in \mathbb{S}_i} K_{i,j}^{\text{sign}(j-i)} \right) \odot M$$

An example is given in figure 3.4. One can see that every key is used twice. One participant uses the key itself ($K_{i,j}$), while another participant uses the inverse of the key ($K_{i,j}^{-1}$). The combination of all the outputs O_i combines of all the $K_{i,j}$ and $K_{i,j}^{-1}$ elements. Therefore the message M is obtained by computing $M = O_1 \odot O_2 \odot \dots \odot O_n$. As $K_{i,j} \odot K_{i,j}^{-1} = id$, all the secret keys cancel and only the message M remains.

3.2.1 Key Establishment

In order to obtain different outputs for each round, the pairs of participants must agree on new secret keys for each round. From a conceptual point of view, participants secretly toss coins over a secure channel. Different techniques have been suggested to realize practical protocols. As suggested in [7], participants may exchange harddiscs with random data, or they may agree to use synchronised pseudo random generators. Golle and Juels present a technique in [21], which is based on Diffie-Hellman key exchange, and which allows to generate keys which have an algebraic structure. (We will see this in section 4.1.3.)

3.2.2 Reservation

In one round of superposed sending, only one participant may send a message, otherwise messages collide. Therefore, rounds should be reserved prior to transmission. Several techniques have been suggested for the anonymous reservation of rounds. After the reservation, each participant knows which rounds he can use, but does not know who is using the other rounds. Normally several rounds are reserved at a time. Such a set of rounds is called a transmission slot. We present now a brief overview of various reservation techniques that have been suggested.

Chaum's Reservation Vector The reservation technique suggested by Chaum in [7] is to use a vector full of 0 bits, where each participants would randomly set one bit to 1. Participants use superposed sending with addition in $GF(2)$. The resulting vector appears after adding all the outputs. By looking at the resulting vector, every participant can tell how many 1's there are on the left of his 1 and thereby determine the transmission slot that he will use. If two or more participants happen to chose the same position, there is a collision and the whole reservation has to be repeated. The more participants there are, the larger the vector must be. Because of the birthday paradox, the required size of the vector grows rapidly with an increasing number of participants.

Reservation based on the Factorisation of a Polynomial The following technique is proposed by Bos and de Boers in [4]. Each participant

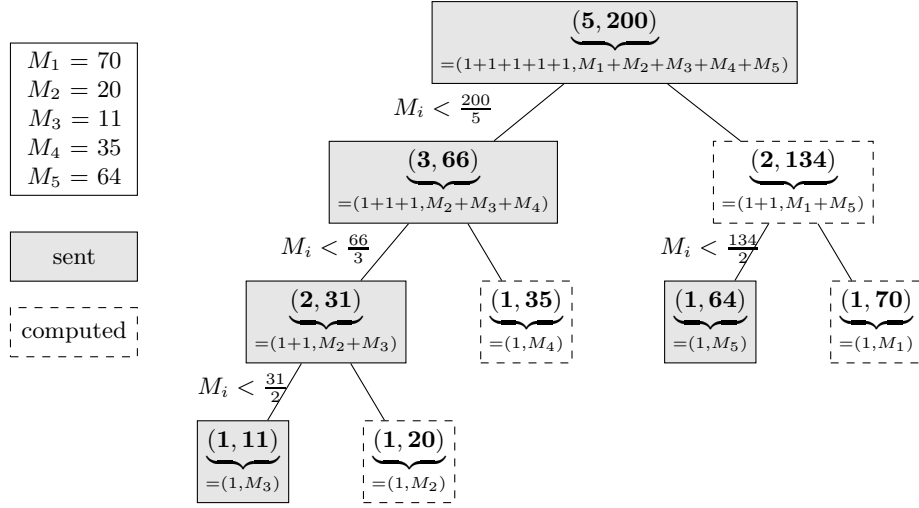


Figure 3.5: Superposed Receiving

P_i sends a vector $(R, A_i, A_i^2, \dots, A_i^n)$. The packets collide and one obtains the vector $(\sum_i R, \sum_i A_i, \sum_i A_i^2, \dots, \sum_i A_i^n)$. From this, a system of linear equations is build to find the coefficients of a polynomial P. This polynomial is then again decomposed into factors which are ordered. The transmission slot that a participant receives, depends on the position of the factor related to his value A_i . R is random value which will be used to shuffle the results to prevent participants from influencing their position.

Reservation Map Technique The reservation map technique was described by Pfitzmann in [27] and by Waidner in [34]. It is based on superposed sending, where G is an additive group modulo m . Each participant chooses one position in a vector of k elements, and sends a 1 in that position. In all other positions he sends 0. The resulting vector will indicate how many participants chose a particular position. For example if 3 participants send a 1 in a position, the result will contain a 3. The positions which contain a 1 indicate a successful reservation, all others are skipped. Participants then use the transmission slots accordingly.

Reservation based on Superposed Receiving Superposed receiving is a transmission technique described by Pfitzmann in [28], and by Waidner in [34], which takes advantage of the algebraic properties of superposed sending.

Superposed sending is used with an additive group, such that a collision is the sum of all values. The transmission of n messages takes n rounds. It is illustrated in Figure 3.5. In the first round, all participants send their messages and create a big collision. This gives the total sum of all values. In the next round, only the participants who had a value below the average send their messages, which divides the total sum into two sub sums. This process is repeated until all the messages have appeared. To compute the average, it must be known how many messages are colliding. Therefore participants send a vector $(1, M_i)$ instead of just M_i . Reservation based on superposed receiving is done by having each participant chose a random number as his message. Then, similarly to the order of the bit in Chaum's vector, the relative position of his number will tell a participant which transmission slot he can use.

Reservation using a Mix Network In [33], Studholme and Blake suggest to make reservations for the Dining Cryptographers using a mixnet to create a secret permutation. Each participant secretly creates a ciphertext. All these ciphertexts are sent through a mixnet, and a vector is obtained, which contains a permutation and a reencryption of these ciphertexts. Each participant can then recognise his ciphertext in the list, and use the corresponding transmission slot.

3.2.3 Detection of Disrupters

Malicious participants may disrupt communication by providing the wrong values. Such participants are called disrupters. The detection of these is difficult, because of the anonymity of the protocol.

Trap based detection A first technique was presented by Chaum in [7] and refined by Waidner in [34]. The principle is that in case of disruption, an investigation phase is started, in which each participant reveals his secret keys and explains how he computed his output. This does however also reveal the sender and his message, and can therefore only be used for rounds which contain no message. Therefore, participants use some of the rounds they reserve as trap rounds, in which they will not send any message. 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 problem is that

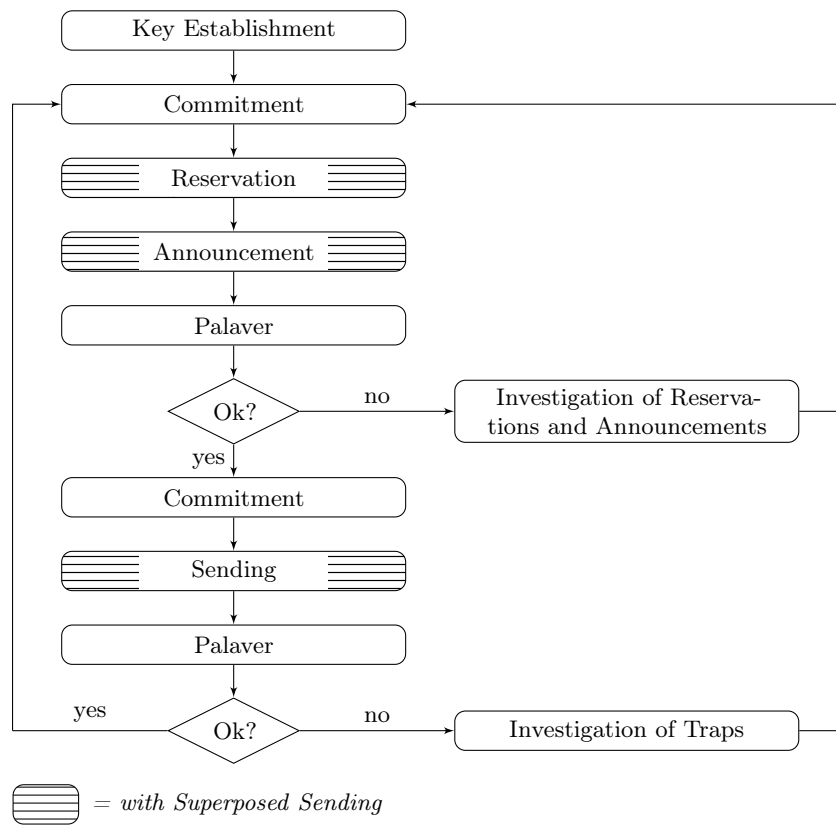


Figure 3.6: Phases of Chaum's protocol with Waidner's improvements

if a round containing a message is disrupted, then the disrupter can not be detected.

The different phases of Chaum’s protocol improved by Waidner are illustrated in detail in Figure 3.6. First, participants establish the secret keys that they need for the superposed sending. Then follows a reservation phase which is preceded by a commitment phase. A commitment phase ensures that an attacker cannot change his output after seeing the output of another participant. After the reservation phase, participants commit to encrypted trap announcements, which contain the information they want to use a round as a trap or not. After this comes a palaver phase, in which participants verify if the reservation and the announcements are accepted by everybody (If nobody noticed a disruption). If there are complaints, an investigation phase is started where all the keys used in the first phases are revealed. If there are no complaints in the palaver phase, participants commit to their outputs for the sending phase. In the sending phase, the messages are sent, trap rounds are left empty. After the sending phase, there is another palaver phase, where participants may ask for investigation if one of their traps was disrupted. Participants who asked for an investigation then open their trap announcements, to prove that the disrupted rounds are their traps, and everybody has to reveal the keys he used for those rounds, and to explain how he computed his output.

Zero-knowledge proof techniques In [21], Golle and Juels present two protocols which are based on zero knowledge proofs. The first protocol is called the Short DC-net protocol, because it is designed to send single elements. Participants generate secret keys that have an algebraic structure. Each participants sends a vector containing n outputs, and can prove that $n-1$ of these outputs are correct and do not contain a message. This restricts a disrupter to disrupt 1 out of n rounds. The second protocol is called the Long DC-net protocol, and it can be used for longer messages. It is based on technique which is comparable to randomised partial checking [22]. Again, each participant sends a vector containing n outputs, and a participant can generate a proof that a high number of his outputs are correct.

3.3 Reliable Broadcast

Receiver untraceability is based on reliable broadcast [34], which is defined by the following two properties [26]:

- Every honest participant receives the same data.
- If the sender is honest, every honest participant receives the (unfalsified) data sent by the sender.

The sender broadcasts his message to all potential recipients, so every recipient is equally likely to be the real receiver. If confidentiality is required, the sender can encrypt the message, so that only the unknown recipient can decrypt it. Reliable broadcast is important to be sure that every receiver receives the same unfalsified message. Otherwise an attacker (which could be the sender) can send different messages to different receivers, in order to identify the real receiver by his response.

Physical Reliable Broadcast In the original story [7], the three cryptographers are sitting in a star restaurant and the reliable broadcast is physically guaranteed.

The Byzantine Generals Problem In case the broadcast is not physically guaranteed, like in the story of the three cryptographers in the restaurant, we have the Byzantine Generals problem, which is introduced by Lamport et al. in [24].

Several generals hold Constantinople under siege. To conquer the city they must attack all the same time, otherwise they will lose the battle. So they must agree on what time to attack. Their problem is that they can only send messages to each other, and there can be traitors who intercept or modify the messages, so they are not sure if each general will really be there at the desired time.

Byzantine Agreement Byzantine Agreement protocols are solutions to the Byzantine Generals problem. They are used to realize a reliable broadcast on networks which do not physically guarantee reliable broadcast. Depending on the initial assumptions, different solutions exist. Several possibilities are presented in [35] and [34].

One example is a computationally secure protocol suggested by Dolev and Strong in [13]. The sender signs his message and sends it to all other participants. Recipients create signatures of the messages they received and send them to other participants.

Fail-Stop Broadcast The idea of fail-stop broadcast is that the communication is interrupted as soon as two honest participants receive different messages. Waidner shows in [34], how this can be done in the Dining Cryptographers protocol. The keys for a round of superposed sending are computed such that they depend completely, but not exclusively, on the values that a participant received in the previous round. If two participants receive different values, their keys will become unsynchronised, and it will not be possible to transfer a message anymore. The computation of the message will then just return a random value.

3.4 Bandwidth Usage

The dining cryptographers protocol combines superposed sending and reliable broadcast. Superposed Sending requires each participant to generate one output per message, and Reliable Broadcast requires each participant to send his output to all other participants. For n participants, this means $n(n - 1)$ bits are sent for one message bit. This is expensive for a high number of participants.

It is more efficient if participants locally combine their outputs and then forward the results only. In [7], Chaum gives the example of a ring network, in which each participant combines his output with the value received from his predecessor, and then forwards the result to the next participant. After one round, the message is computed, and it takes a second round to forward the message to all participants. This reduces the cost to $2n$ bits per message bit. In [31], Goel et al. use a star topology to achieve the same effect. One of the participants collects all the outputs, computes the result and forwards it to the other participants. The cost is again $2n$ bits per message bit.

The problem of these more efficient techniques is that there is no broadcast from the sender to recipients anymore, and an intermediary may falsify the message.

3.5 Summary

The Dining Cryptographers protocol combines sender untraceability and receiver untraceability:

Sender untraceability is based on superposed sending. A key graph represents which pairs of participants share a secret key. The output of any participant except the sender is a combination of his secret keys. The output of the sender is a combination of his secret keys, plus his message. The combination of all outputs make all the secret keys cancel, and the sender's message is revealed. For each round, participants must establish new secret keys. Only one participant may send a message per round, otherwise messages collide. Therefore a reservation phase is used to anonymously reserve rounds. A malicious participant may disrupt the communication by not providing the correct output. The detection of such a disrupter is difficult.

Receiver untraceability is based on reliable broadcast. Every participant receives the same data, so each one could be the receiver. Reliable broadcast means that every honest participant receives the same data, and that if the sender is honest, every honest participant receives the (unfalsified) data sent by the sender. If a reliable broadcast is not physically guaranteed, we have the byzantine generals problem. Byzantine agreement protocols are used to realize a reliable broadcast in presence of the byzantine generals problem. A fail-stop broadcast will interrupt communication, as soon as two honest participants receive different values.

A simple combination of the two techniques is expensive in terms of bandwidth. For n participants, $n(n-1)$ bits have to be sent for one message bit. If the message is computed by an intermediary, and then forwarded to all participants, the costs go down to $2n$. However, an intermediary may then falsify the message.

Chapter 4

A Verifiable Dining Cryptographers Protocol

The key idea presented in this chapter is a new method for superposed sending, which is based on digital pseudonyms. We first present this method based on the assumption that a reservation phase provides participants with a digital pseudonym of the sender. Then we see how such a reservation can be realized. Finally we show what are the possible effects for reliable broadcast, and conclude by describing possible applications of the new methods.

4.1 Verifiable Superposed Sending

In this section we present how a new method for superposed sending, which allows to generate verifiable outputs.

4.1.1 Purpose

A major problem of superposed sending (introduced in section 3.2) is the disruption of the transmission by malicious participants who provide a wrong output. Such malicious participants are called disrupters. Current methods to detect disrupters are not very effective. Chaum's method based on traps requires multiple rounds to detect a disrupter, and the methods presented by Golle and Juels [21] only limit the damage that a disrupter can cause.

We want a superposed sending technique which allows participants to generate verifiable outputs. Each participant should be able to prove the

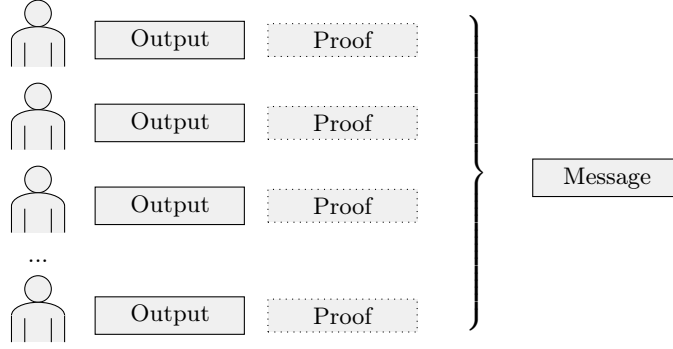


Figure 4.1: Participants generate outputs and prove correctness

correctness of his output. This is illustrated in Figure 4.1. This will discourage malicious participant from providing wrong outputs, as it will be possible to detect them immediately.

4.1.2 Preliminaries

The Diffie-Hellman key exchange and other cryptographic primitives are described in appendix A. Proofs of knowledge are described in appendix B.

4.1.3 The Short DC-Net Protocol

It is possible to prove statements about outputs if they are of algebraic structure. This idea was presented by Golle and Juels in [21]. We use their 'Short DC-Net Protocol' as our starting point.

The Idea The idea is that participants establish their secret keys using the Diffie-Hellman technique. A multiplicative group $G = \langle g \rangle$ is known to all participants. Each participant P_i has a private key a_i and a public key g^{a_i} . The secret key established between P_i and P_j is then $K_{i,j} = g^{a_i a_j}$. We define S_i as the set containing the indexes of the participants with whom P_i is sharing a secret key. The function $\text{sign}(x)$ evaluates to 1 for $x > 0$, and to -1 otherwise. Each participant P_i , except the sender, computes his output

by multiplying all his secret keys:

$$O_i = \prod_{j \in \mathbb{S}_i} K_{i,j}^{\text{sign}(j-i)} = \prod_{j \in \mathbb{S}_i} g^{a_i a_j \text{sign}(j-i)} = \left(\prod_{j \in \mathbb{S}_i} g^{a_j \text{sign}(j-i)} \right)^{a_i}$$

This output has an algebraic structure. More precisely, O_i is of the form Y^{a_i} , where Y can be computed by everybody. We can use this property to prove statements about O_i using zero-knowledge proof techniques, as we will see later. First we have to see how we can extend the technique to multiple rounds.

Multiple Rounds The Diffie-Hellman based technique that was just described allows to generate one set of secret keys, but if we have multiple rounds, we need a new set of secret keys for every round. To extend the technique to multiple rounds, we use the multiplicative groups $G = \langle g \rangle$ and H , and a bilinear map $e : G \times G \rightarrow H$, wherein $e(g^x, g^y) = e(g, g)^{xy}$. Additionally a public random generator is used, which provides a random value $R \in G$ for each round. The secret key established between P_i and P_j is then $K_{i,j} = e(g^{a_i a_j}, R)$. The so obtained secret key is still based on the Diffie-Hellman technique but it is randomised for every new round. If P_i is not the sender, then his output is a multiplication of all his secret keys:

$$O_i = \prod_{j \in \mathbb{S}_i} K_{i,j}^{\text{sign}(j-i)} = \prod_{j \in \mathbb{S}_i} e(g^{a_i a_j}, R)^{\text{sign}(j-i)} = e \left(\prod_{j \in \mathbb{S}_i} g^{a_j \text{sign}(j-i)}, R \right)^{a_i}$$

This output has again the desired algebraic structure, so that we can prove statements about it using zero-knowledge proof techniques, as we will see next.

Proving Correctness of Outputs We can now use the algebraic structure of the outputs to construct proofs. The output O_i of each participant P_i , except the sender, should be of the form Y^{a_i} . Everybody can compute Y , and (g, g^{a_i}) and are publicly known. The output O_i of every honest participant, except the sender, must satisfy $\log_Y(O_i) = \log_g(g^{a_i})$. This can be used to prove that an output is correct, without revealing a_i . Using the notation introduced by Camenisch and Stadler in [5] (described in appendix B.3), this corresponds to the statement $\mathcal{PK}(\alpha : (g^\alpha = g^{a_i}) \wedge (Y^\alpha = O_i))$.

The correct behaviour of participants cannot be verified by directly verifying this statement, as it would prevent the sender from sending a message. The statement is however very useful as a part of a larger proof. In the Short DC-Net Protocol, n participants submit a vector containing n outputs at a time, and they prove that $n - 1$ of these outputs are of the form Y^{a_i} . This means that a participant can use one of the n outputs to send his message. As a consequence, a disrupter can maximally create one collision per n rounds.

4.1.4 Remaining Problems

The Short DC-Net protocol presents very interesting ideas, but it has the following problems.





A first problem is that no reservation is made, and collisions of messages are accepted. Even if it was extended with a reservation phase, a disrupter would be able to disrupt 1 round out of n , while remaining undetected.

A second problem is coming from the bilinear map, which is used to generate secret keys for multiple rounds. The only currently known ways to realize a bilinear map are based on elliptic curve cryptography, and then the group H is a group based on the elliptic curve. The secret keys and the outputs are elements of H , but our messages usually are in \mathbb{Z}_p . The mapping of messages from \mathbb{Z}_p to H and back again is not trivial, and we would like to avoid this.

4.1.5 Our Solution

The Short DC-Net Protocol introduces interesting techniques, but because of the remaining problems we need something else. This leaves us with the questions: Can we create outputs of algebraic structure for multiple rounds, without using a bilinear map? Can we add a reservation phase, and consider the reservations in the proofs? We will now see what is possible if we assume that we have a digital pseudonym of the sender.

Assumptions Let $G = \langle g \rangle$ be a group in which the Decision Diffie-Hellman problem is assumed to be infeasible. We assume that $G = \langle g \rangle$ is known to all participants. Each participant P_i has a private key a_i and a public key g^{a_i} . Let us assume that from a reservation phase we have a digital pseudonym of the sender g^x which is known to all participants, and only the

		Digital pseudonym g^x
(sender)	 public: g^{a_1} private: a_1	$O_1 = (g^x)^{a_1}$
	 public: g^{a_2} private: a_2, M, x	$O_2 = \frac{M}{(g^{a_1} g^{a_3} g^{a_4} \dots g^{a_n})^x}$
	 public: g^{a_3} private: a_3	$O_3 = (g^x)^{a_3}$
	 public: g^{a_4} private: a_4	$O_4 = (g^x)^{a_4}$


	 public: g^{a_n} private: a_n	$O_n = (g^x)^{a_n}$
		$M = \prod_{i=1}^n O_i$

Figure 4.2: Computing outputs based on a digital pseudonym

sender knows the private key x .

Computation of Outputs Using the digital pseudonym g^x , a participant P_i can establish a secret directly with the sender, and use this as his output:

$$O_i = (g^x)^{a_i}$$

This output has the required algebraic structure. Furthermore, it is indistinguishable from a random output because the Decision Diffie-Hellman problem is infeasible in $G = \langle g \rangle$. Let \mathbb{T} denote the set containing the indexes of all participants except the sender. The sender P_i computes his output to contain his message M , as follows:

$$O_i = \frac{M}{(\prod_{i \in \mathbb{T}} g^{a_i})^x}$$

The message M can then be obtained by multiplying all outputs. An example is shown in Figure 4.2.

Multiple rounds If for every reservation a new digital pseudonym is chosen, the outputs will be different for each round. The randomness is coming from the digital pseudonym.

Proving Correctness of Outputs An output O_i is correct if it has the form $O_i = (g^x)^{a_i}$, or if P_i is the sender. This means that either $\log_{(g^x)}(O_i) = \log_g(g^{a_i})$ or P_i knows x . P_i can prove that one of both statements is true, without revealing which one, by proving the following statement:

$$\mathcal{PK}\{\alpha : ((g^\alpha = g^{a_i}) \wedge ((g^x)^\alpha = O_i)) \vee (g^\alpha = g^x)\}$$

How to prove this statement is given as an example at the end of appendix B.3. We have achieved our goal, the outputs are indistinguishable from random outputs, but verifiable.

4.1.6 Long Messages

If the messages are longer, of the form (M_1, M_2, \dots, M_k) instead of just M , then it is possible to create multiple outputs $(O_{i1}, O_{i2}, \dots, O_{ik})$ from one digital pseudonym. Each participant P_i must have a secret vector $(a_{i1}, a_{i2}, \dots, a_{ik})$ and a corresponding public vector $(g^{a_{i1}}, g^{a_{i2}}, \dots, g^{a_{ik}})$, instead of having only one private value a_i and one public value g^{a_i} . Then a new pair (a_i, g^{a_i}) is used for each output. This allows to adjust the protocol to the desired maximal message size.

4.1.7 Variations

We assumed until now, that the message is computed as a multiplication of all outputs. As the sender knows the precise outputs of all other participants, other functions of the more general form $M = f(O_1, O_2, \dots, O_n)$ are also possible. The sender must then chose his output, such that the function evaluates to his message.

4.1.8 Discussion

In our solution, all other participants use the Diffie-Hellman technique to establish a shared secret with the sender, while keeping the identity of the latter secret. This completely eliminates the need for pairs of participants

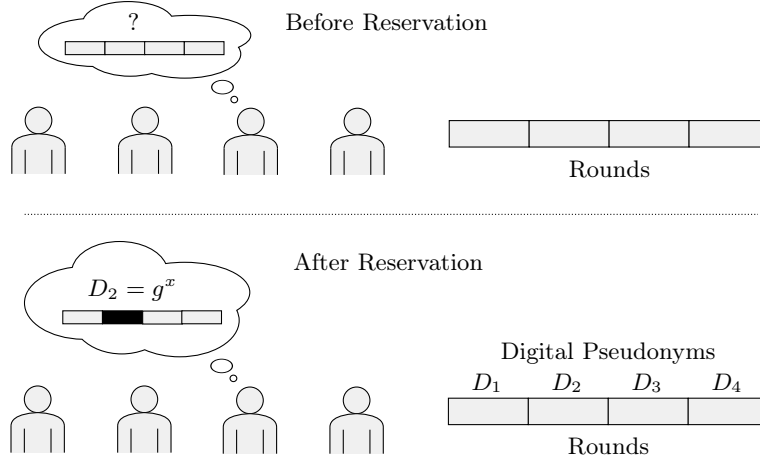


Figure 4.3: Before and After Reservation

to share secret keys. Outputs are indistinguishable from random outputs because of the Decision Diffie-Hellman assumption. The sender has more knowledge than before, when the outputs were generated based on a key-graph, as he knows in advance the precise outputs of all other participants. This is not a problem, since he is the sender himself. Any other participant does not have more information than before.

Until now, we assumed that we have a reservation phase which provides us with the digital pseudonym of the sender. We will see next, how to actually realize such a reservation phase.

4.2 Reservation

We assumed in the previous section that we had a reservation phase which provides the participants with a digital pseudonym of the sender. In this section we will see how this can be realized.

4.2.1 What We Need

Existing reservation techniques secretly inform each participant if he is the sender or not. We need a reservation technique which informs each participant if he is the sender or not, and additionally provides every participant with a digital pseudonym of the sender. This is illustrated in figure 4.3.

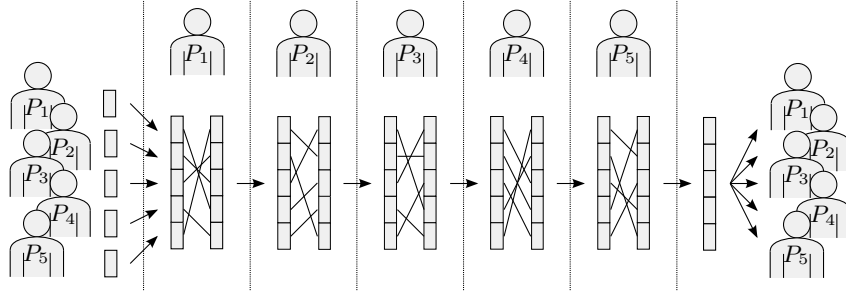


Figure 4.4: Mix based reservation

4.2.2 A Mixnet based Reservation

We present a solution which is based on a mixnet (illustrated in Figure 4.4). Each participant first privately generates a digital pseudonym, then participants use a mixnet to obtain a shuffled list of all their digital pseudonyms. A mixnet is the best known practical method to obtain such a shuffled list.

Description Each participant P_i secretly chooses a number $x_i \in \mathbb{Z}_q$, and computes g^{x_i} . All participants send their values g^{x_i} through a mixnet to get (D_1, D_2, \dots, D_n) , a permutation of $(g^{x_1}, g^{x_2}, \dots, g^{x_n})$. To obtain computational security, a mixnet is used, in which each participant shuffles the list one time, such that no trust in a set of mix owners is required. The values D_1, D_2, \dots, D_n are the digital pseudonyms and determine the order in which the participants will send their messages. Each participant P_i recognises his g^{x_i} , and knows which round he can use to send his message.

To prevent an attacker from reusing a digital pseudonym from another participant, each participant P_i should generate a non-interactive proof of knowledge of the secret key corresponding to his digital pseudonym ($\mathcal{PK}\{\alpha : g^{x_i} = g^\alpha\}$) and send a packet through the mixnet which contains the public key and the proof of knowledge, so that other participants can verify that the participant who submitted a digital pseudonym also knows the corresponding private key.

After receiving the list (D_1, D_2, \dots, D_n) , participants verify that everybody received the same list, and each participant P_i verifies that his value g^{x_i} is in the list.

Discussion For a computationally secure protocol, a mixnet seems to be the best choice. A mixnet is naturally suited to do the reservation, since the keys are small and of equal size. Each participant mixes the list one time, so that no trust is required. The mixnet is only used to shuffle randomly generated keys, and no message information is sent through the mixnet. This has two advantages. First, reservations can be made long time before the actual transmission of a message. Even if the reservation is a high latency process, this has no negative impact on the latency of the transmission of a message. Second, it is not necessary to use a verifiable mixnet, with zero knowledge proofs about the inputs and the outputs of each mix. A verification at the end is sufficient. If a participant sees that his key is not in the list as it should, he can initiate an investigation phase. In order to find the misbehaving mix, all participants disclose the values they used. This does not compromise any message information, and the disclosed keys can just be ignored and replaced by new ones. This means that a simple reencryption mixnet can be used, which is easier to realize than a verifiable mixnet.

A possible alternative could be a reservation based on superposed receiving, but it would be more complicated to realize.

4.3 Effects on the Broadcast

The advantage of having a digital pseudonym of the sender for the broadcast, is related to what is described in section 3.4. The sender can sign the message, and a recipient can use the digital pseudonym to verify the signature. This allows to realize efficient topologies, where for example one participant collects all outputs, computes the message and sends it to all participants. Because the message is signed, the intermediary can not replace the message with a fake message.

To obtain a reliable broadcast, it is however still necessary to verify that everybody received the same message. In case the intermediary was the malicious sender, he could send different signed messages to different recipients.

4.4 Applications

In last chapter, we saw the fundamental principles of the new protocol. In this chapter, we describe how this can be used to realize systems for:

- Electronic voting,
- Low latency anonymous communication, and
- Fast multiparty computation.

4.4.1 Electronic Voting

Using the new techniques to implement a simple electronic voting system is straightforward. What is described next is illustrated in Figure 4.5.

First, the voters execute a reservation phase to obtain a list of digital pseudonyms. Each voter has one digital pseudonym in the list. This list is signed by all voters to ensure that the list is correct and that everybody has the same list.

Then, each voter generates a ballot which contains an output and a proof for each digital pseudonym. When a voter encounters his digital pseudonym, he can cast his vote. All outputs look like random outputs, so that the vote is not revealed. The proofs allow to verify that the voter behaved correctly.

Finally, all the ballots are combined and the votes appear. The votes are pseudonymously signed, so it is possible to use the list of digital pseudonyms to verify the pseudonymous signatures. This shows that each vote is originating from the voter who holds the secret key corresponding to the digital pseudonym. The signed list of digital pseudonyms and the list of the pseudonymously signed votes can be used to prove the correctness of the result to any third party.

The advantages of the proposed system are that it is easy to implement, and that the result that can be used to prove the correctness of the results to a third party. A disadvantage is that every voter must submit his ballot, otherwise the votes cannot be computed.

4.4.2 Low Latency Anonymous Communication

We can also build low latency communication systems, using the techniques described in previous chapter.

Reservation of Rounds As described previously, participants use a mixnet to anonymously reserve the rounds for transmission. In communication system, they just continuously generate new random keys and send them

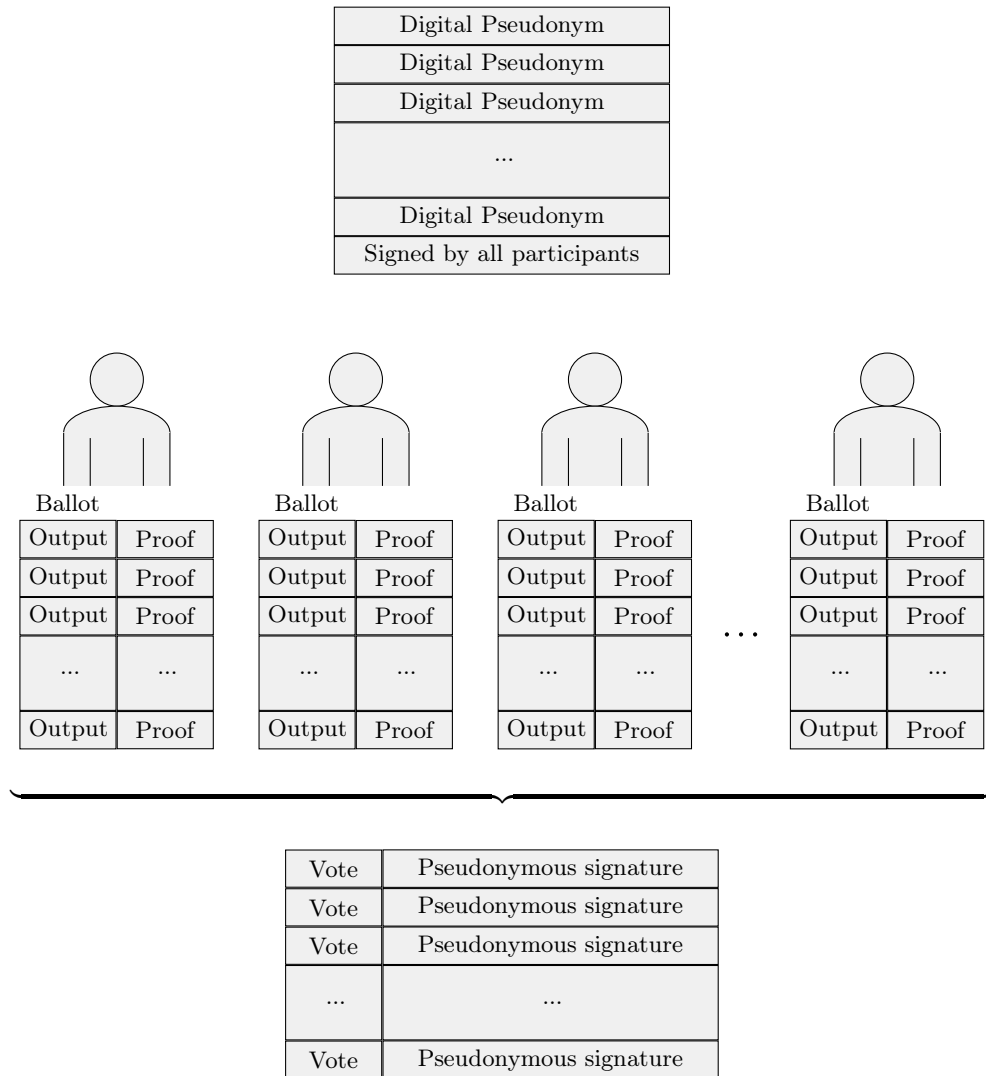


Figure 4.5: Electronic voting

through the mixnet. There will be an initial delay until the first set of keys is coming out of the mixnet, but then each step provides a new set of digital pseudonyms.

Transmission of Messages For the transmission of messages, participants can use a star or a tree topology. The combination of an pseudonymously signed message and the proofs that the individual values are correct, allows to efficiently aggregate the values and broadcast the message. This is described in the next paragraphs, and it is illustrated in Figure 4.6. (We use the notation defined in the appendix A.3.3, and y_i is the public key of participant P_i).

A message is pseudonymously signed by the respective sender. Intermediary nodes can then aggregate the received outputs with their own and forward the result. The participant at the root of the tree computes the message and verifies the signature. If the signature is not valid, an investigation phase is started and the participants must provide the proofs that their values were correct. If the signature is valid, then the pseudonymously signed message is sent to the other participants. Each participant uses again the signature to verify that the root did not cheat. This gives total bandwidth costs that are linear relative to the number of participants.

To achieve receiver untraceability, participants must ensure that all of them received the same message. Otherwise a malicious sender could generate several signed messages, and provide different participants with different messages. Therefore, after reception of the message, each participant generates a signature of the message he received and forwards his signature to all other participants. Using aggregate signatures (see appendix A.3.5), participants can use the same star or tree topology that they use for message transmission to combine and broadcast these signatures.

Traffic Announcement It is possible to announce the traffic, in order to efficiently use the available bandwidth. This is especially useful for large messages. In a system without traffic announcement, as illustrated in Figure 4.7, a message is transmitted for each digital pseudonym, and all these messages are of the same size. In a system with traffic announcement, we will have two phases, illustrated in Figure 4.8. First, we make traffic announcements, in which each sender indicates the size of his message. Then, the messages

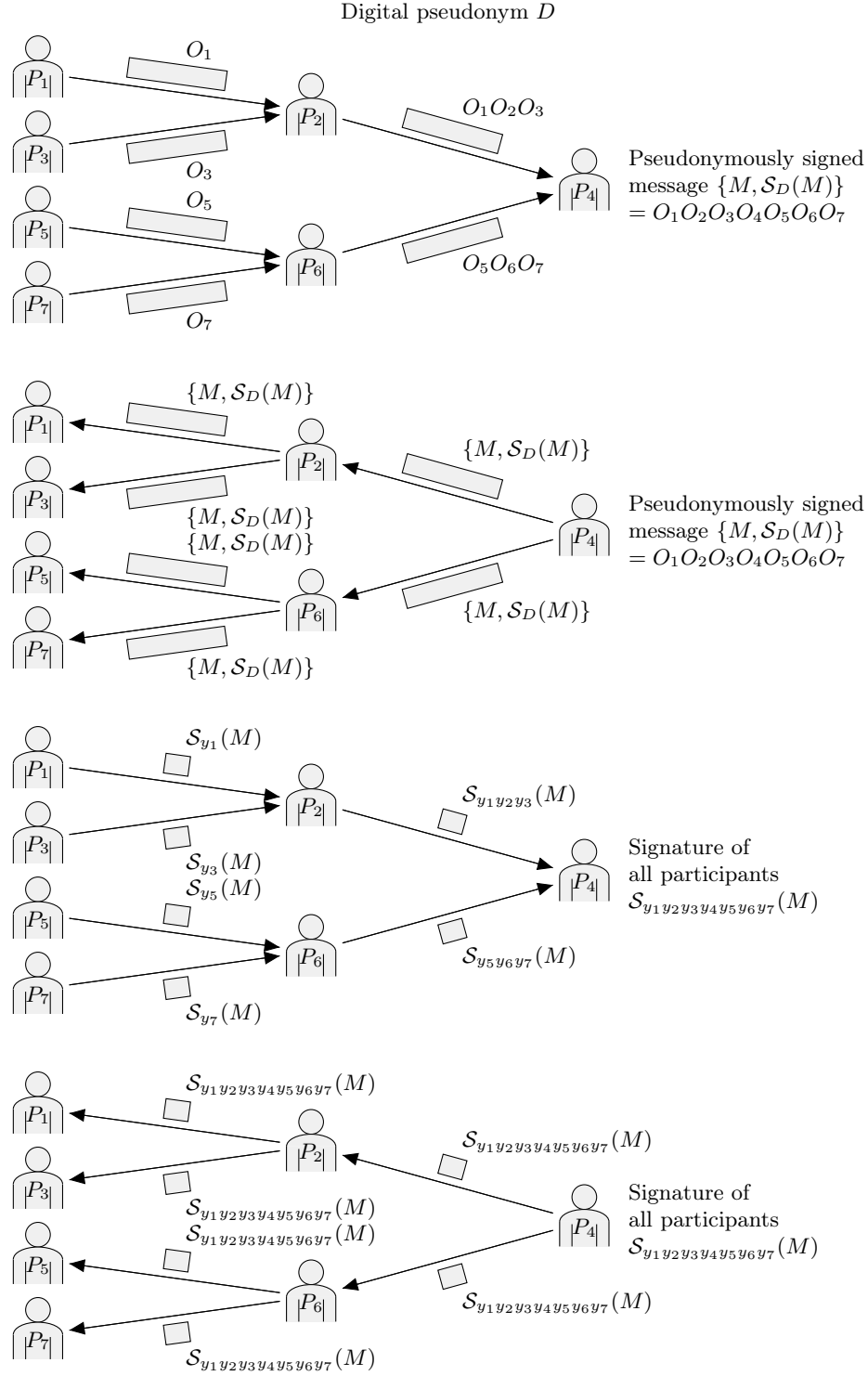


Figure 4.6: Transmission in a tree topology

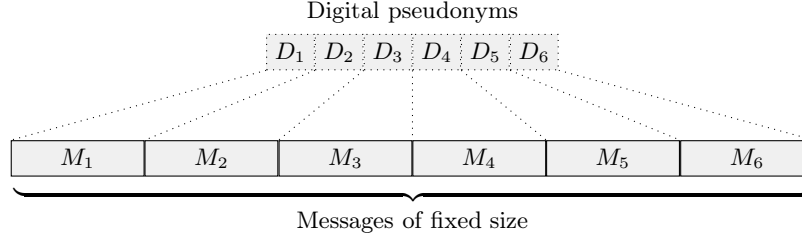


Figure 4.7: Message transmission without traffic announcement

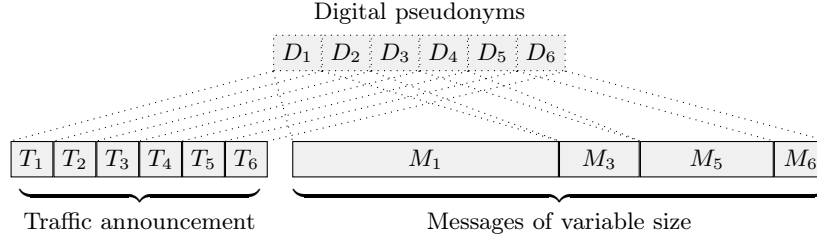


Figure 4.8: Message transmission with traffic announcement

are transmitted according to the announced sizes. So, participants which do not have a message to transmit do not unnecessarily waste bandwidth. The message size can be arbitrarily long, and the interval between the reservation (an traffic announcement) steps can be chosen. This allows to realize networks with a high number of participants, which use low bandwidth when nobody is sending a message.

Comparison with Mixnets The new solution is much more flexible than a mixnet of the same level of security. It allows to obtain a better performance in terms of bandwidth usage and latency. A mixnet is sequential, and for n participants to achieve computational security (without trust), a mixnet will require n steps. The new solution works with any tree topology, so the latency can be chosen to be very low. It should for example be possible for two participants to communicate using Voice Over IP technologies, without having annoying transmission delays. Mixnets also require to have one message from each participant, and the messages must all be of the same size. The new solution allows to use messages of variable size, and to use traffic announcement techniques to efficiently use the bandwidth.

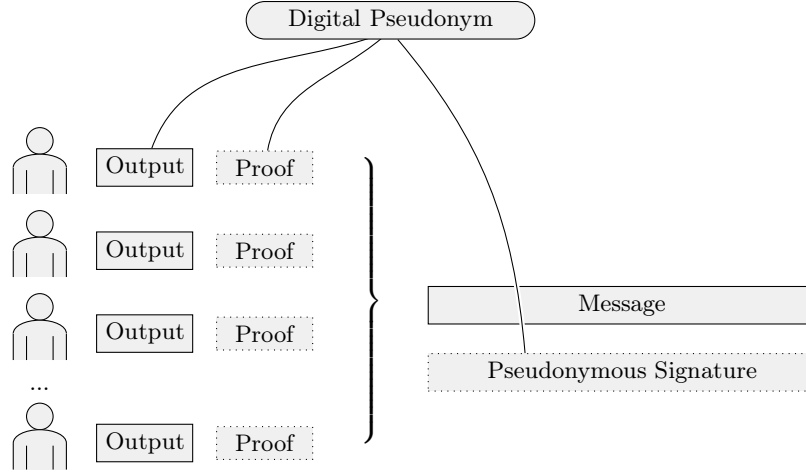


Figure 4.9: Usage of a Digital Pseudonym of the Sender

4.4.3 Secure Multi-Party Computation

Another area of application is secure multi-party computation, first presented by Yao in [36]. Participants perform a computation where the input from each participant remains secret, and where only the result is publicly known. Sometimes intermediary results have to be passed through a mixnet. The sequential mixing is time consuming and slows down the multiparty computation. The techniques presented herein allow to realize mixing rounds with a very low latency, which makes these multiparty computations faster.

4.5 Summary

In this chapter we presented verifiable superposed sending, a computationally secure version of superposed sending, which allows participants to generate verifiable outputs. We used a digital pseudonym, which is a public key of the form g^x , and only the anonymous sender know the corresponding secret key x . This digital pseudonym is very useful, as illustrated in figure 4.9. Participants use the digital pseudonym and the Diffie-Hellman key exchange to establish a shared secret with the sender, while keeping the identity of the latter secret. This allows to generate outputs of algebraic structure, which are indistinguishable from random outputs in a group where the Decision Diffie Hellman problem is assumed to be infeasible. The discrete logarithms

based nature of the digital pseudonym and of the outputs allows to use standard proof techniques to prove statements about them. A participant can prove that he participated correctly, without revealing if he is the sender or not. The sender can also use his private key x to sign his message, and all other participants can verify the signature using the digital pseudonym. This allows combine computation and broadcast, which reduces bandwidth usage. For example one participant can collect all the outputs, compute the message and then forward it to the other participants. The pseudonymous signature allows other participants to verify that the intermediary did not cheat, and that the message is really coming from the sender. We saw how to obtain a digital pseudonym from the sender, using a reservation phase based on a mixnet. The new methods can be applied to electronic voting, low latency anonymous communication, and multiparty computation.

Chapter 5

Conclusions

Our initial problem was the detection of malicious participants which disrupt communication in a dining cryptographers protocol by providing wrong outputs. We presented the verifiable superposed sending technique, which allows participants to prove the correctness of their outputs, without compromising the anonymity of the sender. This solves our problem, as disrupters are now immediately detectable. The possibility to verify the correct behaviour of a participant, which exists for many years in mix based networks, is now available in the dining cryptographers protocol, which makes it a practicable alternative to mixnets. In fact, it can even outperform a mixnet in terms of latency, and it allows to use traffic announcement techniques to keep the bandwidth usage low. We presented possible applications of our new techniques in the fields of electronic voting, low latency anonymous communication and secure multi-party computation.

Interesting activities for future research, are the study of the behaviour of our techniques in a real network. Especially the presented idea for a low latency anonymous communication systems, in combination with the presented traffic announcement technique, has the potential to generate interesting applications, for a high number of anonymous senders and recipients.

Bibliography

- [1] M. Bellare. O. Goldreich. On defining proofs of knowledge. In *Advances in Cryptology-CRYPTO '92*, volume 740, pages 390–420, 1992.
- [2] M. Bellare and P. Rogaway. Random oracles are practical: a paradigm for designing efficient protocols. In *Proceedings of the 1st ACM conference on Computer and communications security*, pages 62–73. ACM New York, NY, USA, 1993.
- [3] D. Boneh, C. Gentry, B. Lynn, and H. Shacham. A survey of two signature aggregation techniques. *RSA CryptoBytes*, 6(2):1–10, 2003.
- [4] J. Bos and B. den Boer. Detection of disrupters in the DC protocol. *Advances in Cryptology-EUROCRYPT'89*, pages 320–327, 1989.
- [5] J. Camenisch and M. Stadler. Efficient Group Signature Schemes for Large Groups. *LECTURE NOTES IN COMPUTER SCIENCE*, pages 410–424, 1997.
- [6] 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.
- [7] D. Chaum. The dining cryptographers problem: Unconditional sender and recipient untraceability. *Journal of Cryptology*, 1(1):65–75, 1988.
- [8] D. Chaum and T.P. Pedersen. Wallet Databases with Observers. *LECTURE NOTES IN COMPUTER SCIENCE*, pages 89–89, 1993.
- [9] D.L. Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM*, 24(2):84–88, 1981.
- [10] R.J.F. Cramer. *Modular design of secure yet practical cryptographic protocols*. Universiteit van Amsterdam, 1997.
- [11] I. Damgård. Discrete log based cryptosystems. *Manuscript, www.daimi.au.dk/ivan/DL.pdf*, 31, 2004.

- [12] W. Diffie and M. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976.
- [13] D. Dolev and H.R. Strong. Authenticated Algorithms for Byzantine Agreement. *SIAM J. Comput.*, 12(4):656–666, 1983.
- [14] D. Dolev and A. Yao. On the security of public key protocols. *Information Theory, IEEE Transactions on*, 29(2):198–208, 1983.
- [15] M. Robson E. G. Sirer, M. Polte. Cliquenet: A self-organizing, scalable, peer-to-peer anonymous communication substrate. 2001.
- [16] T. El Gamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *Proceedings of CRYPTO 84 on Advances in cryptology*, pages 10–18, 1985.
- [17] A. Fiat and A. Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *Advances in Cryptology*, volume 86, pages 186–194. Springer, 1986.
- [18] V. Fussenig, D. Spiewak, and T. Engel. Acimn protocol: A protocol for anonymous communication in multi hop wireless networks. In *Information Security 2008*, volume 81. Conferences in Research and Practice in Information Technology (CRPIT), 2008.
- [19] D. Goldschlag, M. Reed, and P. Syverson. Hiding Routing Information. *LECTURE NOTES IN COMPUTER SCIENCE*, pages 137–150, 1996.
- [20] S. Goldwasser, S. Micali, and C. Rackoff. The knowledge complexity of interactive proof-systems. In *Proceedings of the seventeenth annual ACM symposium on Theory of computing*, pages 291–304. ACM New York, NY, USA, 1985.
- [21] 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.
- [22] M. Jakobsson, A. Juels, and R.L. Rivest. Making Mix Nets Robust for Electronic Voting by Randomized Partial Checking. In *Proceedings of the 11th USENIX Security Symposium*, pages 339–353. USENIX Association Berkeley, CA, USA, 2002.
- [23] A. Joux and K. Nguyen. Separating Decision Diffie–Hellman from Computational Diffie–Hellman in Cryptographic Groups. *Journal of Cryptology*, 16(4):239–247, 2003.

- [24] L. Lamport, R. Shostak, and M. Pease. The Byzantine Generals Problem. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 4(3):382–401, 1982.
- [25] C. Park, K. Itoh, and K. Kurosawa. Efficient anonymous channel and all/nothing election scheme. *Workshop on the theory and application of cryptographic techniques on Advances in cryptology*, pages 248–259, 1994.
- [26] M. Pease, R. Shostak, and L. Lamport. Reaching Agreement in the Presence of Faults. *Journal of the ACM (JACM)*, 27(2):228–234, 1980.
- [27] A. Pfitzmann. How to implement ISDNs without user observability—Someremarks. *ACM SIGSAC Review*, 5(1):19–21, 1987.
- [28] A. Pfitzmann. *Diensteintegrierende Kommunikationsnetze mit teilnehmerüberprüfbarem Datenschutz*. Springer, 1990.
- [29] A. Pfitzmann and M. Kohntopp. Anonymity, Unobservability, and Pseudonymity-A Proposal for Terminology.
- [30] M.K. Reiter and A.D. Rubin. Crowds: anonymity for Web transactions. *ACM Transactions on Information and System Security*, 1(1):66–92, 1998.
- [31] M. Polte S. Goel, M. Robson and E. G. Sirer. Herbivore: A scalable and efficient protocol for anonymous communication. 2002.
- [32] CP Schnorr. Efficient identification and signatures for smart cards. In *Advances in Cryptology–EUROCRYPT’89: Proceedings*. Springer Verlag, 1990.
- [33] C. Studholme and I. Blake. Multiparty Computation to Generate Secret Permutations. 2007.
- [34] M. Waidner. Unconditional Sender and Recipient Untraceability in spite of Active Attacks. *Lecture Notes in Computer Science*, 434:302, 1990.
- [35] M. Waidner and B. Pfitzmann. The Dining Cryptographers in the Disco: Unconditional Sender and Recipient Untraceability with Computationally Secure Serviceability. *Advances in Cryptology - EUROCRYPT*, 89:10–13, 1989.
- [36] A.C. Yao. Protocols for secure computations. In *Proceedings of the 23rd Annual IEEE Symposium on Foundations of Computer Science*, pages 160–164, 1982.

Appendix A

Cryptographic Primitives

A.1 Computationally Hard Problems

It is possible to chose $G = \langle g \rangle$, a group of large prime order, such that for $x, y \in \mathbb{Z}$ and $z \in G$, the following problems are hard:

- Discrete logarithm (DL) problem:

given (g, g^x) , find x

- Computational Diffie-Hellmann (CDH) problem:

given (g, g^x, g^y) , find g^{xy}

- Decision Diffie-Hellmann (DDH) problem:

given (g, g^x, g^y, z) , decide if $z = g^{xy}$

The problems are dependent on each other. For the DDH problem to be hard, the CDH problem must be hard, and for the CDH problem to be hard, the DL must be hard.

If a problem is hard or not, depends on the choice of the group G . Typically it is possible to use finite fields based on integers or based on elliptic curves. To achieve the same level of security, integer based groups require more bits than elliptic curve based groups. However integer based groups are simpler to implement, and one example can be found in [11].

Gap Diffie-Hellman Groups and Bilinear Maps A group in which the CDH is hard and in which the DDH is easy, is called a Gap Diffie-Hellman group. The only known way to implement such a group is based on elliptic curve cryptography. It is possible to define a bilinear map $e : G \times G \rightarrow G_T$,

with $G = \langle g \rangle$, such that the CDH is hard in G , and that:

$$e(g^{xy}, g) = e(g^y, g^x)$$

The DDH can be solved by comparing $e(z, g)$ and $e(g^x, g^y)$, so G is a Gap Diffie-Hellman group. More information can be found in [23].

A.2 Diffie-Hellman Key Exchange

The Diffie-Hellman key exchange was presented by Whitfield Diffie and Martin Hellman in their famous paper [12]. It allows two participants to establish a shared secret over an insecure channel.

1. Alice and Bob share a group $G = \langle g \rangle$, in which the Computational Diffie-Hellmann (CDH) problem is assumed to be infeasible.
2. Alice chooses a random value a .
Bob chooses a random value b .
3. Alice sends g^a to Bob.
Bob sends g^b to Alice.
4. Alice computes $(g^b)^a$.
Bob computes $(g^a)^b$.
5. The secret value $g^{ab} = (g^b)^a = (g^a)^b$ is shared by Alice and Bob.

A.3 Public Key Encryption Schemes

In a public key encryption scheme, there is a key pair consisting of a private key x and a public key y . A randomization value r is used to obtain different ciphertexts for multiple encryptions of the same message. The encryption of the message m using the public key y and the randomization r is written:

$$\mathcal{E}_y(m, r)$$

The decryption of the ciphertext c using the private key x is written:

$$\mathcal{D}_x(c)$$

A.3.1 El Gamal Encryption

The El Gamal encryption is a public key encryption system based on the Diffie-Hellman key exchange [16].

1. Initially: A group $G = \langle g \rangle$ is publicly known.

2. Public key generation: A secret key x is chosen, and the public key is $y = g^x$.
3. Encryption: To encrypt a message m , a random value r is chosen, and the ciphertext (c_1, c_2) is computed with $(c_1, c_2) = \mathcal{E}_y(m, r) = (g^r, y^r m)$.
4. Decryption: From the ciphertext (c_1, c_2) , the message m is obtained by computing $m = \mathcal{D}_x(c_1, c_2) = c_2/c_1^x$.

To achieve semantic security, a group G must be used in which the Decision Diffie-Hellmann (DDH) problem is assumed to be infeasible.

A.3.2 Homomorphic Encryption

A homomorphic encryption scheme is an encryption scheme, in which we have the following property:

$$\mathcal{E}_y(m_1, r_1) \odot \mathcal{E}_y(m_2, r_2) = \mathcal{E}_y(m_1 \otimes m_2, r_1 \oplus r_2)$$

The operations \odot , \otimes and \oplus depend on the specific encryption scheme that is used.

El Gamal based Homomorphic Encryption In the El Gamal encryption scheme, we have $\mathcal{E}_y(m, r) = (g^r, y^r m)$. We can create an El Gamal based homomorphic encryption scheme by defining \cdot to be the pairwise multiplication of ciphertext elements. This gives the following property:

$$\begin{aligned} \mathcal{E}_y(m_1, r_1) \cdot \mathcal{E}_y(m_2, r_2) &= (g^{r_1}, y^{r_1} m_1) \cdot (g^{r_2}, y^{r_2} m_2) \\ &= (g^{r_1+r_2}, y^{r_1+r_2} m_1 m_2) \\ &= \mathcal{E}_y(m_1 m_2, r_1 + r_2) \end{aligned}$$

A.3.3 Public Key Signature Schemes

In a public key signature scheme, there is a key pair consisting of a private key x and a public key y . It is possible to create a signature for message using the private key x . The authenticity of the signature can then be verified with the public key y . The signature of the message m using the private key x is written:

$$\mathcal{S}_y(m)$$

A.3.4 El Gamal Signature Scheme

The El Gamal signature is a public key signature scheme based on the discrete log problem [16].

1. Initially: A group $G = \langle g \rangle$ is publicly known.

2. Public key generation: A secret key x is chosen, and the public key is $y = g^x$.
3. Signature: A random value r is chosen, and the signature (s_1, s_2) of a message m is then computed by $(s_1, s_2) = \mathcal{S}_y(m) = (g^r, \mathcal{H}(m) - xg^r)$.
4. Verification: The signature $\mathcal{S}_y(m) = (s_1, s_2)$ is valid if $g^{\mathcal{H}(m)} = y^{s_1} s_2^{s_2}$.

A.3.5 Aggregate Signatures

An aggregate signature scheme allows to combine the signatures of multiple participants. This allows to sign a message multiple times, without increasing its size. In [3], an aggregate signature scheme is described, which is based on bilinear maps. A single signature is generated with the following steps:

1. Initially: A bilinear map $e : G \times G \rightarrow G_T$, with $G = \langle g \rangle$, and where the CDH is hard in G (see A.1).
2. Public key generation: A secret key $x \in \mathbb{Z}$ is chosen, and the public key is $y = g^x$.
3. Signature: A random value r is chosen, and the signature s of a message m is then computed by $s = \mathcal{S}_y(m) = \mathcal{H}(y, m)^x$.
4. Verification: The signature $\mathcal{S}_y(m) = s$ is valid if $e(\mathcal{H}(m, y), y) = e(s, g)$.

Several signatures can now be aggregated as follows:

$$\begin{aligned} \mathcal{S}_{y_1 y_2 \dots y_n}(m) &= \mathcal{H}(y_1, m)^{x_1} \cdot \mathcal{H}(y_2, m)^{x_2} \cdot \dots \cdot \mathcal{H}(y_n, m)^{x_n} \\ &= \mathcal{S}_{y_1}(m) \cdot \mathcal{S}_{y_2}(m) \cdot \dots \cdot \mathcal{S}_{y_n}(m) \end{aligned}$$

To verify the aggregated signature $AS = \mathcal{S}_{y_1 y_2 \dots y_n}(m)$, the verifier checks if:

$$\prod_{i=1}^n e(\mathcal{H}(m, y_i), y_i) = e(AS, g)$$

Appendix B

Proofs of Knowledge

B.1 Zero-Knowledge Proofs of Knowledge

An interactive proof allows a prover to prove a statement to a verifier. It can be used to verify that a party correctly participated in the execution of a protocol, without disclosing the details of his action. An interactive proof has the following properties:

1. Completeness: A honest prover can convince a verifier of the correctness of a true statement.
2. Soundness: A cheating prover cannot convince a honest verifier of the correctness of a false statement.

Proofs of Knowledge The concept of a "proof of knowledge" was introduced by Goldwasser et al. in [20], and defined by Bellare and Goldreich in [1]. Informally, a proof of knowledge allows a prover to prove to a verifier that he knows a witness which verifies a given statement.

Zero-Knowledge Proofs A zero-knowledge proof allows a prover to prove to a verifier that a statement is true, without giving the verifier any other information. This means that the proof does not enable the verifier to compute anything that he could not have computed before. There exist many flavors of zero-knowledge. As we try to realize a protocol that is computationally secure, zero knowledge in our context means computational zero-knowledge.

B.2 Sigma Protocols

A sigma protocol [10], illustrated in Figure B.1, is a three step proof protocol, which contains the three steps of: commitment, challenge and response, and which has the following properties:

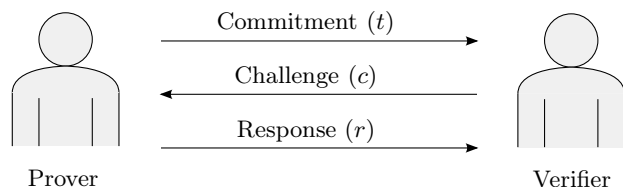


Figure B.1: A sigma (Σ) protocol

1. Completeness: A honest prover can convince a verifier of the correctness of a true statement.
2. Special Soundness: Two successful conversations (t, c, r) and (t, c', r') , with $c \neq c'$ allow to compute the witness.
3. Special Honest Verifier Zero-Knowledge: There exists a simulator which on an input c can produce (t, c, r) , which looks like a real conversation with the real prover.

Honest Verifier Zero-Knowledge A honest verifier is supposed to chose his challenge randomly. A proof where a dishonest verifier could get some knowledge by choosing c in a special way, is called a honest-verifier zero knowledge (HVZK) proof. As one cannot expect the verifier to be honest, HVZK proofs may seem uninteresting a first sight. It is however possible to transform these proofs into non-interactive proofs, which eliminates the problem of the dishonest verifiers.

Non Interactive Sigma Protocol The Fiat Shamir heuristic is a technique that allows to turn a three step sigma protocol into a non-interactive (one step) protocol. The challenge is generated by the prover using a hash function, with $c = \mathcal{H}(t)$. The prover then transmits (c, r) to the verifier in one step. The Fiat-Shamir heuristic was presented in [17] and is provably secure in the random oracle model [2]. The term heuristic is due to the fact that a hash function is used instead of a real random oracle.

B.3 Proving Statements about Discrete Logarithms

It is possible to use sigma protocols to prove general statements about discrete logarithms. It is for example possible to prove: knowledge of the discrete log of a value to a base, equality of discrete logs to different bases, and logical combinations thereof. A system for proving general statements about discrete logarithms is presented in [6].

Notation

He will use hereinafter a notation introduced by Camenisch and Stadler in [5]. The notation is illustrated by the following examples. The secrets are represented by greek symbols, and the known values are represented by non-greek symbols.

- Proof of knowledge of the discrete logarithm of y to the base g :

$$\mathcal{PK}\{\alpha : y = g^\alpha\}$$

- Proof of knowledge of the discrete logarithm of y_1 to the base g_1 and of the discrete logarithm of y_2 to the base g_2 :

$$\mathcal{PK}\{(\alpha, \beta) : (y_1 = g_1^\alpha) \wedge (y_2 = g_2^\beta)\}$$

- Equivalence of the discrete logarithms of y_1 to the base g_1 and of y_2 to the base g_2 :

$$\mathcal{PK}\{\alpha : (y_1 = g_1^\alpha) \wedge (y_2 = g_2^\alpha)\}$$

Examples

Example 1 Proof of knowledge of the discrete logarithm of y to the base g :

$$\mathcal{PK}\{\alpha : y = g^\alpha\}$$

1. The prover randomly chooses $a \in \mathbb{Z}$, computes $t = g^a$ and sends t to the verifier.
2. The verifier randomly chooses $c \in \mathbb{Z}$, and sends c to the prover.
3. The prover computes $r = a + c\alpha$, and sends r to the verifier.
4. The verifier accepts if $g^r = ty^c$.

This proof was introduced in [32] and is known as the Schnorr protocol. If we apply the Fiat-Shamir heuristic, we obtain:

1. The prover randomly chooses $a \in \mathbb{Z}$, and computes $t = g^a$.
2. The prover computes $c = \mathcal{H}(t)$.
3. The prover computes $r = a + c\alpha$, and sends (c, r) to the verifier.
4. The verifier computes $t' = g^r y^{-c}$ and accepts if $\mathcal{H}(t') = c$.

Example 2 Proof of the equality of discrete logarithms to different bases ($\log_{g_1} y_1 = \log_{g_2} y_2$):

$$\mathcal{PK}\{\alpha : (y_1 = g_1^\alpha) \wedge (y_2 = g_2^\alpha)\}$$

1. The prover randomly choses $a \in \mathbb{Z}$, computes $t_1 = g_1^a$, $t_2 = g_2^a$ and sends (t_1, t_2) to the verifier.
2. The verifier randomly choses $c \in \mathbb{Z}$, and sends c to the prover.
3. The prover computes $r = a + c\alpha$, and sends r to the verifier.
4. The verifier accepts if $g_1^r = t_1 y_1^c$ and $g_2^r = t_2 y_2^c$.

If we apply the Fiat-Shamir heuristic, we obtain:

1. The prover randomly choses $a \in \mathbb{Z}$, computes $t_1 = g_1^a$, $t_2 = g_2^a$.
2. The prover computes $c = \mathcal{H}(t_1, t_2)$.
3. The prover computes $r = a + c\alpha$, and sends (c, r) to the verifier.
4. The verifier computes $t_1' = g_1^r y_1^{-c}$, $t_2' = g_2^r y_2^{-c}$ and accepts if $\mathcal{H}(t_1', t_2') = c$.

This principle was described in [8], and is known as the Chaum Pedersen proof. Here it is only optimized by passing (c, r) instead of (t_1, t_2, r) to the verifier.

Example 3 Proof of knowledge of the discrete logarithm of y_1 to the base g_1 , or of y_2 to the base g_2 :

$$\mathcal{PK}\{\alpha : (y_1 = g_1^\alpha) \vee (y_2 = g_2^\alpha)\}$$

- If the prover knows α , such that $y_1 = g_1^\alpha$:
 1. The prover randomly choses $a, r_2, c_2 \in \mathbb{Z}$, computes $t_1 = g_1^a$, $t_2 = g_2^{r_2} y_2^{-c_2}$ and sends (t_1, t_2) to the verifier.
 2. The verifier randomly choses $c \in \mathbb{Z}$, and sends c to the prover.
 3. The prover computes $c_1 = c - c_2$, $r_1 = a + c_1 \alpha$ and sends (c_1, c_2, r_1, r_2) to the verifier.
 4. The verifier accepts if $c = c_1 + c_2$, $g_1^{r_1} = t_1 y_1^{c_1}$ and $g_2^{r_2} = t_2 y_2^{c_2}$.
- If the prover knows α , such that $y_2 = g_2^\alpha$:
 1. The prover randomly choses $a, r_1, c_1 \in \mathbb{Z}$, computes $t_1 = g_1^{r_1} y_1^{-c_1}$, $t_2 = g_2^a$ and sends (t_1, t_2) to the verifier.

2. The verifier randomly chooses $c \in \mathbb{Z}$, and sends c to the prover.
3. The prover computes $c_2 = c - c_1$, $r_2 = a + c_2\alpha$ and sends (c_1, c_2, r_1, r_2) to the verifier.
4. The verifier accepts if $c = c_1 + c_2$, $g_1^{r_1} = t_1 y_1^{c_1}$ and $g_2^{r_2} = t_2 y_2^{c_2}$.

The prover decomposes the challenge c into the challenges c_1 and c_2 , such that $c = c_1 + c_2$. He can then choose the value of either c_1 or c_2 , but he must must have the secret knowledge to respond to the remaining challenge.

If we apply the Fiat-Shamir heuristic, we obtain:

- If the prover knows α , such that $y_1 = g_1^\alpha$:
 1. The prover randomly chooses $a, r_2, c_2 \in \mathbb{Z}$, computes $t_1 = g_1^a, t_2 = g_2^{r_2} y_2^{-c_2}$.
 2. The prover computes $c = \mathcal{H}(t_1, t_2)$.
 3. The prover computes $c_1 = c - c_2$, $r_1 = a + c_1\alpha$ and sends (c_1, c_2, r_1, r_2) to the verifier.
 4. The verifier computes $t_1' = g_1^{r_1} y_1^{-c_1}$, $t_2' = g_2^{r_2} y_2^{-c_2}$ and accepts if $\mathcal{H}(t_1', t_2') = c_1 + c_2$.
- If the prover knows α , such that $y_2 = g_2^\alpha$:
 1. The prover randomly chooses $a, r_1, c_1 \in \mathbb{Z}$, computes $t_1 = g_1^{r_1} y_1^{-c_1}, t_2 = g_2^a$.
 2. The prover computes $c = \mathcal{H}(t_1, t_2)$.
 3. The prover computes $c_2 = c - c_1$, $r_2 = a + c_2\alpha$ and sends (c_1, c_2, r_1, r_2) to the verifier.
 4. The verifier computes $t_1' = g_1^{r_1} y_1^{-c_1}$, $t_2' = g_2^{r_2} y_2^{-c_2}$ and accepts if $\mathcal{H}(t_1', t_2') = c_1 + c_2$.

Example 4 Proof of knowledge of the following statement:

$$\mathcal{PK}\{\alpha : (y_1 = g_1^\alpha) \vee ((y_2 = g_2^\alpha) \wedge (y_3 = g_3^\alpha))\}$$

- If the prover knows α , such that $y_1 = g_1^\alpha$:
 1. The prover randomly chooses $a, r_2, c_2 \in \mathbb{Z}$, computes $t_1 = g_1^a, t_2 = g_2^{r_2} y_2^{-c_2}, t_3 = g_3^{r_2} y_3^{-c_2}$ and sends (t_1, t_2, t_3) to the verifier.
 2. The verifier randomly chooses $c \in \mathbb{Z}$, and sends c to the prover.
 3. The prover computes $c_1 = c - c_2$, $r_1 = a + c_1\alpha$ and sends (c_1, c_2, r_1, r_2) to the verifier.
 4. The verifier accepts if $c = c_1 + c_2$, $g_1^{r_1} = t_1 y_1^{c_1}$, $g_2^{r_2} = t_2 y_2^{c_2}$ and $g_3^{r_2} = t_3 y_3^{c_2}$.

- If the prover knows α , such that $y_2 = g_2^\alpha$:
 1. The prover randomly choses $a, r_1, c_1 \in \mathbb{Z}$, computes $t_1 = g_1^{r_1} y_1^{-c_1}, t_2 = g_2^a, t_3 = g_3^a$ and sends (t_1, t_2, t_3) to the verifier.
 2. The verifier randomly choses $c \in \mathbb{Z}$, and sends c to the prover.
 3. The prover computes $c_2 = c - c_1, r_2 = a + c_2 \alpha$ and sends (c_1, c_2, r_1, r_2) to the verifier.
 4. The verifier accepts if $c = c_1 + c_2, g_1^{r_1} = t_1 y_1^{c_1}, g_2^{r_2} = t_2 y_2^{c_2}$ and $g_3^{r_2} = t_3 y_3^{c_2}$.

If we apply the Fiat-Shamir heuristic, we obtain:

- If the prover knows α , such that $y_1 = g_1^\alpha$:
 1. The prover randomly choses $a, r_2, c_2 \in \mathbb{Z}$, computes $t_1 = g_1^a, t_2 = g_2^{r_2} y_2^{-c_2}, t_3 = g_3^{r_2} y_3^{-c_2}$ and sends (t_1, t_2, t_3) to the verifier.
 2. The prover computes $c = \mathcal{H}(t_1, t_2, t_3)$.
 3. The prover computes $c_1 = c - c_2, r_1 = a + c_1 \alpha$ and sends (c_1, c_2, r_1, r_2) to the verifier.
 4. The verifier computes $t_1' = g_1^{r_1} y_1^{-c_1}, t_2' = g_2^{r_2} y_2^{-c_2}, t_3' = g_3^{r_2} y_3^{-c_2}$ and accepts if $\mathcal{H}(t_1', t_2', t_3') = c_1 + c_2$.
- If the prover knows α , such that $y_2 = g_2^\alpha$:
 1. The prover randomly choses $a, r_1, c_1 \in \mathbb{Z}$, computes $t_1 = g_1^{r_1} y_1^{-c_1}, t_2 = g_2^a$.
 2. The prover computes $c = \mathcal{H}(t_1, t_2, t_3)$.
 3. The prover computes $c_2 = c - c_1, r_2 = a + c_2 \alpha$ and sends (c_1, c_2, r_1, r_2) to the verifier.
 4. The verifier computes $t_1' = g_1^{r_1} y_1^{-c_1}, t_2' = g_2^{r_2} y_2^{-c_2}, t_3' = g_3^{r_2} y_3^{-c_2}$ and accepts if $\mathcal{H}(t_1', t_2', t_3') = c_1 + c_2$.