Eindhoven university of technology
Department of mathematics and computer science

Master's thesis

Encrypted certificate schemes and their security and privacy analysis

B.J.M. (Bart) Mennink

Eindhoven, June 2009

Supervisors:
Dr.ir. L.A.M. (Berry) Schoenmakers
Dr. J. (Jorge) Guajardo Merchan

Master's thesis

Encrypted certificate schemes and their security and privacy analysis

Eindhoven university of technology
Department of mathematics and computer science

Author:
B.J.M. (Bart) Mennink

Supervisors:
Dr.ir. L.A.M. (Berry) Schoenmakers
Dr. J. (Jorge) Guajardo Merchan

Eindhoven, June 2009

Abstract

Certificate schemes are well-known cryptographic primitives, where an authority certifies certain attributes (e.g. name, date of birth, bank account and salary) of a user, who can then show that certificate to someone else. There are however scenarios where the authority as well as the user may not want to disclose or publish the attribute values (which for instance might be the result of some secure multiparty computation). For instance in medical surveys, we would like to use certificate schemes to guarantee that the users' inputs to the survey are correct without disclosing (in plaintext) private sensitive medical data. Formal protocols for solving these problems are derived in this thesis. More generally, certificate schemes and secure multiparty computation are combined, where the encrypted outcomes of a multiparty computation are certified, either with or without the user learning the plaintext attributes, and where the user can disclose certified attributes in encrypted form to a verifier. The main aim of this combination is to obtain a scheme that provides the possibility to compute statistics on encrypted inputs, while still offering privacy and security.

In the first part of the thesis, in a general approach the algorithms and protocols required for the combination of the two primitives, certificate schemes and secure multiparty computation, are derived. This approach applies to any certificate scheme, and is therefore named universal. Then we construct the required protocols for each of these two primitives. In the area of secure multiparty computation, protocols are constructed that are either required for the extended certificate scheme to work correctly (e.g. a Paillier to ElGamal conversion gate), or for the computation of statistics of encrypted data (e.g. a modulo reduction gate). In the second part of the thesis certificate schemes are considered, and the algorithms and protocols introduced in the universal approach are constructed. In particular, an *encrypted certificate scheme* is constructed where the user is issued a certificate on ElGamal encryptions without learning the plaintext values. To this end, the definition of certificate schemes is reformulated, formalized and extended.

It turns out that it is possible to generalize the certificate scheme of Brands [Bra99] to the case where the attributes to be certified are encrypted and unknown to the user and/or certification authority. This extension is considered for both the ElGamal and Paillier cryptosystem. Following this generalization together with the universal construction, a complete set of algorithms and protocols for combining the two primitives is possible. This extension is efficient as it only requires linear overhead (in the number of attributes) when compared to the original scheme of Brands. By mixing different protocols one can reduce the cost (even to constant order). Security of the new scheme is slightly weaker as it requires stronger assumptions. Although verification is not publicly possible anymore in case the user does not learn the plaintext attributes, this verification *is* possible for participants that *in practice* are expected to be able to verify, i.e. the certification authorities.

Acknowledgements

I would like to thank the people that have been important for the accomplishment of this thesis.

Firstly, the people that played a significant role during my study career. Among the TU/e employees, especially Georg Prokert who supervised my bachelor's thesis on 'pointwise convergence of Fourier series', and the ones that brought me in contact with the area of cryptology, namely Henk van Tilborg and Berry Schoenmakers. Moreover thanks to Boris Škorić and Benne de Weger for being in the exam committee. Among my fellow students, a special word for Jochem Berndsen, whose bright view on mathematics is magical.

The people at Philips, in particular Milan Petković for his help on medical applications of my project, and all of his useful advises. Guido Hermans, Luan Ibraimi, Qing Liu and Martin Franz for sharing the office. The latter moreover for sharing interesting discussions.

Also a special word of gratitude to the home front, including my friends and family, Bert, Els and Paul and my girlfriend Audrey, who have been very valuable during my study. Audrey is also thanked for acting in the introducing example of Chapter 1 $\ddot\smile$.

But most of all, thanks to my supervisors Berry Schoenmakers and Jorge Guajardo Merchan. Berry, the 'walking cryptology encyclopedia', whose comments and advises to my work proved themselves very useful to me. Jorge, whose masterly sharp eyes have been very valuable to this thesis, and who always had some time free to talk! The cooperation with both of them was very important and educational to me. Both deserve my greatest respects; They showed me the way, and I am glad to follow in their footsteps!

Table of Contents

List of Figures

1. Introduction

This thesis is concerned with two major primitives in cryptography, namely certificate schemes[1] and secure multiparty computation schemes.

In a certificate scheme, users obtain certificates from organizations certifying them that they comply with some condition. These certificates can subsequently be shown to other parties in order to obtain access to a service or to obtain another certificate. For instance, the government issues driving licenses proving the right of the owner to drive a car. These can in turn be shown at the car rental office to rent a car (this property is commonly called 'disclosure', as the owner discloses the data). Digital certificate schemes need to provide two major demands: security and privacy. Security roughly means that it is difficult to obtain a false certificate. In practice this would for instance mean that it should be impossible for an underage person to obtain a driving license. Privacy means that the secret values in the certificate cannot leak and the different executions (for instance showing a driving license at two places) cannot be linked. For real-life certificates, the privacy requirement is not always satisfied: a passport can be used by its owner to buy beer (more formally: he can prove that he is sufficiently old to buy beer), but the seller learns the consumer's name, address, etc. as well. In digital certificate schemes, preserving privacy turns out to be possible, as it has a.o. been shown by [Bra99, CL02, CL04].

In a secure multiparty computation scheme (MPC), a number of parties jointly compute a given function on data private to these parties, or more generally on any set of secret data. The output of that computation can be either public (as in Yao's millionaires problem [Yao82]), or private (if the output will be used as input to other multiparty computations). Of course, the main goal of multiparty computation is that the output is correct. Moreover, it needs to be secure, in the sense that no information leaks (except anything that can be concluded from the output), even if a subset of parties is actively malicious (i.e., deviates from the protocol). Normally, the function to be computed itself is publicly known, but it can be secret input of one of the participants as well.

These two primitives are well understood in the literature, but combining them yields a scheme that offers a higher level of privacy and security. For instance, Yao's millionaires problem [Yao82] is concerned with two millionaires, say Audrey and Bart, who want to find out who is richer without revealing their fortunes. This can be achieved by letting the participants both encrypt their wealths x_a and x_b, and use a so-called 'comparison gate' to compute $f(x_a, x_b) = [x_a < x_b]$. Multiparty protocols for this function are known that are correct and secure, yet these do not prevent Audrey and Bart from initially lying about their wealths (a drawback already noticed by Yao). This additional security property can be guaranteed by introducing a certificate scheme. A service provider, the bank, provides the users with a certificate on their bank deposits and upon comparing their values, the millionaires also verify each other's certificate. By construction of the certificate scheme, this verification does not leak the certified value, yet at the end of the protocol one party convinces the other that the correct value is encrypted. This idea is visualized in Figure 1.1.

[1]These schemes are sometimes called 'credential schemes'. Throughout the thesis, we use the term 'certificate scheme' instead.

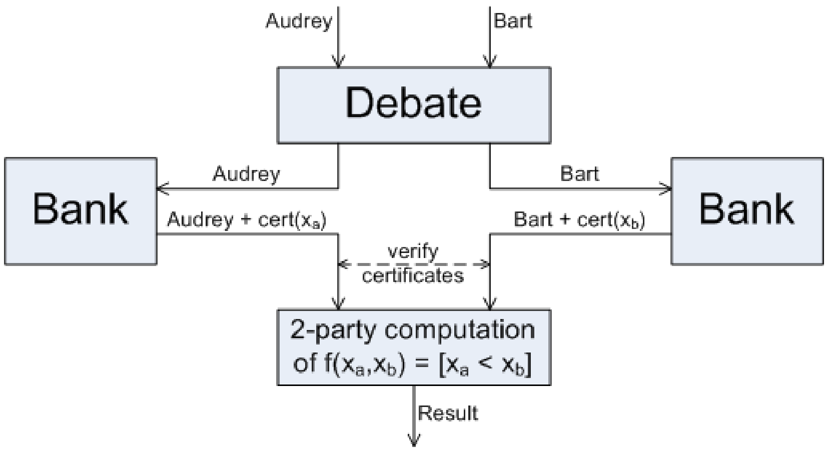

Figure 1.1: Yao's millionaires problem, preventing the participants from lying.

This millionaires example is an intuitive example, but not a very realistic one. More realistic examples are found in the medical domain. Indeed, medical data are usually very sensitive and many people are reluctant to distribute such information[2] So the interesting question and the main problem to be solved is how to unite certificate schemes and secure multiparty computation, in order to construct a secure scheme for (among others) multiparty computation of statistics. In particular, the development of the protocols for the interface between these schemes is the main goal of this thesis. The following three examples will allow the reader to visualize how this question can help to solve practical medical security problems in practice. Applications in non-medical domains are imaginable as well, as follows from the two subsequent examples. In Section 3.3, these examples are studied more extensively, along the lines of a detailed universal description of our construction.

Example 1.0.1 (Je echte leeftijd)**.** A well-known survey in the Netherlands is the 'je echte leeftijd'-survey: a website that computes 'your real age', given your lifestyle, eating behavior, medical history, etc. [web09b]. In this survey participants enter lots of private and sensitive information about themselves, which would not be a big problem if privacy was guaranteed[3]. The result is that it is very tempting to enter slightly different data, and this clearly influences the correctness and reliability of the results. So on the one hand, you want the inputs to be correct, which can be achieved by deploying a certificate scheme. On the other hand, you do not want the website to learn anything about your lifestyle, your illnesses or any other sensitive private information, while still having the possibility to do statistical analysis on the inputs. This can be achieved using a multiparty computation scheme.

Example 1.0.2 (Je echte leeftijd, extended)**.** The purpose of the 'je echte leeftijd'-survey is in fact to compute the 'real age' of the participant. Naturally, you want this data to be private, so the outcome of the multiparty computation should only be learned by the user, and by no-one else (not even the website itself)[4]. So, again the website obtains several *certified* values and does multiparty computation to obtain a result, but unlike Example 1.0.1, now the result is in encrypted form. In addition, one can think of a scenario where the output needs to be certified as well (for further use by the user) in which case the website is a service provider that also issues certificates.

[2]Some governments already require persons concerned with medical data to comply with privacy standards, like for instance the Health Insurance Portability and Accountability Act (HIPAA) in the United States [web09a].

[3]Ironically, the website states: 'In order to ensure your privacy, you need to register with your name, date of birth, e-mail, gender and address'.

[4]In this case the computation is done on private inputs of just one participant, but for extension to multiple participants one can think of a computation that results in the quartile the 'real age' of a certain user is in, with respect to the 'real ages' from all participants with the same year of birth.

Example 1.0.3 (Health monitoring)**.** The idea of health monitoring is that a participant wears a device, that records medical data about that person. Philips for example offers such a device and associated service that allows users to reach certain exercise level which, in turn, allows them to have a healthier life style [web09c]. More generally, the government (or another institute) can decide to supply a certain part of the population with a device that measures cardiac data (blood pressure, heartbeat, etc.), and sends these measured values to a central server where statistical analysis can be executed (for instance, the differences in the heart activity in different age groups or social classes). Moreover, if it were possible to revoke anonymity (the certificate scheme is naturally required to offer privacy), then it could for instance also be used to study ways in which heart attacks can be predicted. The function of the certificate scheme in this case is to let the observed values be certified, and the function of the multiparty computation is to compute statistics on the measured data. Again, the schemes need to be combined to allow the complete system to work correctly.

Example 1.0.4 (Activity monitoring)**.** 'Stichting Kijkonderzoek' is a Dutch institute that measures and analyses the television viewing behavior of the Ducth audience [web09d]. Briefly, the idea is that a representative set of households have a device attached to their TV which registers the users' watching preferences and sends these to a server. The server then does statistical analysis on the data and publishes the results (for example, 'yesterday $\pm 3.141.000$ people watched the 8 p.m. news broadcast'). For privacy reasons, you do not want the server to learn the watching behavior of the users, but for the statistical analysis you do want the possibility to execute computations, and in particular to link different certified data (with respect to age groups, religion, social classes, etc.). The use of the two schemes and their connection becomes clear now.

Example 1.0.5 (Social networks)**.** Consider a dating network where members make some of their private information public (name, date of birth, hobbies, favorite television programs, etc.)[5]. Then, a man can search over the network for a woman of his interests (or vice versa). Clearly, this system is not anonymous, nor is it secure (users can enter wrong data). Now, we are aiming at a completely secure and anonymous social network where, a priori, no one can be trusted. To this end, we need a set of servers that can find out a possible match between two users, given their private encrypted inputs. One can see a match as a session key to a virtual chat room. The relevance of a certificate scheme becomes clear as the users' inputs need to be correct and anonymous. The multiparty computation system is used to match two users securely.

Related work

Put forward by Chaum [Cha83, Cha84] and Damgård [Dam88], the first certificate schemes in the literature were meant for online payment systems [CFN90, Bra93, Bra95a]. Later, more general certificate schemes have been constructed, merely due to Brands [Bra95b, Bra99], Lysyanskaya et al. [LRSW99] and Camenisch and Lysyanskaya [CL01, CL02, CL04]. Recently, Belenkiy et al. [BCKL08] constructed a non-interactive certificate scheme. These certificate schemes differ in various aspects (type of certificates, underlying assumption, efficiency, etc.). For reasons argued in Section 3.4, in this thesis we will use Brands' discrete log based certificate scheme [Bra99]. Brands constructed several certificate schemes, based on the discrete log or the RSA assumption, and all schemes providing the possibility to include attributes in a certificate and to selectively disclose such attributes. We note that also [CL02, CL04] offer these two possibilities but unlike those,

[5]We note that this is a known example from literature, for instance studied by Juels and Sudan [JS06]. In their construction, Alice locks a vault with her secret key. The key contains a set of Alice's interests. Bob can then open the vault if he has similar interests (which need not be exactly the same). Their construction is based on error-correcting codes. In our example the interests are the encrypted values instead of the key, and therefore the possibility for Bob to unlock the vault if he has similar interests is difficult. In fact some problems pointed out in [JS06, Sec. 1.1] still apply to our solution. However, our focus is merely on the combination of certificate schemes with multiparty computation, and as a consequence encryptions of hobbies are certified (for instance, by sporting clubs), which makes it difficult for Alice (or Bob) to cheat. It is not studied whether the ideas by Juels and Sudan can be combined with certificate schemes as well.

Brands' certificates are one-show certificates, meaning that a user owning a certificate can only show it once (otherwise the different showing executions can be linked and privacy is violated). To our knowledge, no research has been done to combining certificate and encryption schemes, which is the main result of this thesis. The reader is referred to Section 2.8 for an extensive and more fundamental survey of certificate schemes.

In 1982, Yao introduced multiparty computation [Yao82], and initiated a new area within cryptography. Mainly, two flows of research on multiparty computation are known: multiparty computation based on verifiable secret sharing and multiparty computation based on threshold homomorphic encryption. Differences between these branches are for instance in the security setting (verifiable secret sharing based multiparty computation is unconditionally secure, while threshold homomorphic encryption is in the cryptographic model). This thesis focuses on threshold homomorphic encryption based multiparty computation, originally introduced by Franklin and Haber [FH96] and the independent papers by Jakobsson and Juels [JJ00] and Cramer, Damgård and Nielsen [CDN01]. These three papers all introduce a scheme for securely computing any circuit, but differ in a few aspects: while [FH96] only considers passive adversaries, [JJ00, CDN01] consider active adversaries as well. Moreover, the encryption scheme used by Franklin and Haber requires all participants to help upon decryption. The mix and match scheme by Jakobsson and Juels relies on mixing truth tables of gates in a boolean circuit, and is applicable to any discrete logarithm based encryption scheme. The framework of [CDN01] is restricted to an RSA-setting, but can be used for any arithmetic circuit and moreover has a smaller round complexity than the approach of [FH96]. Cramer et al. introduced a protocol for secure multiparty multiplication for any general additively homomorphic encryption scheme (and therewith implying a protocol for any secure function evaluation). Succeeding [CDN01], publications by Schoenmakers and Tuyls [ST04, ST06] and by Garay et al. [GSV07] are of particular interest, concerning several protocols for secure computation (conditional multiplication, bit extraction and comparison). This thesis extends these works in the sense that new protocols are constructed based on those. The reader is referred to Section 2.9 for an extensive and more fundamental survey of secure multiparty computation, including a mathematically oriented study of the named relevant publications.

Our Contributions

This thesis considers the combination of certificate schemes and multiparty computation. Although much research has been done already in both areas, the combination has not been studied yet. Moreover, our new scheme is of particular interest in many existing practical examples, especially examples where either the privacy issues are ignored (e.g. in the current solution for Example 1.0.4) or an important security item is ignored (in almost any solution to Yao's millionaires problem in literature it is possible for an active adversary to initially enter a wrong value, a.o. [Yao82, FH96, Fis01, ST04, LT05, GSV07]). Our focus will be on multiparty computation of statistics (such as computing the mean and the variance), although the proposed construction can in principle be applied to any secure function evaluation scheme. This choice is made because of the practical relevance of multiparty computation of statistics (compare for instance Example 1.0.3 with Yao's millionaires problem).

Because our construction involves the combination of two existing schemes from literature, our contributions are twofold. Firstly, in the area of multiparty computation, protocols for statistics are considered. At first sight such protocols are straightforward, but many statistics involve integer division, which is undefined in rings (and therefore in particular undefined for the ElGamal and Paillier cryptosystems). Therefore, a smart integer division protocol is constructed, and via this protocol statistical multiparty analysis is made possible. Furthermore, for the functionalities of the new certificate scheme, additional gates are needed, including a gate for converting Paillier encryptions to ElGamal and an extended gate for checking plaintext equalities of several relations (for instance $E(x_1)^{y_1} \overset{?}{=} E(x_2)^{y_2}$ for secret y_1, y_2).

A certificate scheme by Stefan Brands [Bra99] is extended such that it can is combined with multiparty computation. Specifically, this is done as follows:
Firstly, a general construction is given, describing the protocols and algorithms required for the certificate scheme and multiparty computation scheme, as well as the protocols required for joining these schemes. This construction is high-level and universal in the sense that it applies to any certificate scheme with the option of selective disclosure of attributes[6]. Roughly, these components are (**A**) a protocol for a user to disclose attributes *in encrypted form*, (**B**) a protocol for a set of multiparty computation servers to certify an encrypted value to a user *without the servers learning it* but where the user still learns the decryption and (**C**) an extension to **B** where now *even the user* does not learn the encrypted value. For the mentioned scheme of Brands, these three protocols are constructed for the ElGamal cryptosystem, and protocols **A** and **B** also for the Paillier cryptosystem. It turns out that the protocols **A** and **B** are small extensions to Brands' scheme, but for **C** a completely new scheme is constructed, and to this end the definition of 'certificate scheme' is revisited. More concretely, we reformulate and formalize Brands' definition of certificate schemes in such a way that it can be easily adapted to a definition of 'encrypted certificate scheme', scheme **C**.

The newly constructed encrypted certificate scheme turns out to be more expensive than Brands' scheme, in terms of communication complexity and certificate size. This is completely intuitive and no more than logical, but the efficiency loss turns out to be linear in the number of encrypted attributes only! This means that if the certificate only involves two (or in general: 'few') attributes, the complexity difference is constant as well. Moreover, the schemes can be easily mixed, for instance by using the encrypted certificate scheme for the first two attributes and a 'normal' certificate scheme for the other attributes, in which case the complexity difference with respect to the basic scheme becomes constant in the number of attributes again. We believe that this general complexity loss is not dreadful, in fact the main application of this scheme is where a service provider outputs an encrypted multiparty computation result that needs to be certified. In many practical examples this result is just one encryption, and thus we only need two encrypted attributes (a certificate with *one* encrypted attribute turns out to be impossible as it would leak something about the encrypted value), so with complexity difference with respect to Brands' scheme of constant order only.

Outline

First of all, Chapter 2 introduces the mathematical/cryptographical prerequisites for our scheme. Of special interest are Sections 2.8-2.9, where the related work with respect to certificate schemes and multiparty computation schemes is discussed. Experienced readers can skip this chapter.
Chapter 3 formalizes the problem statement. This is done in a natural and gradual way, until a description of the required protocols for the interface between the schemes is obtained. This formalization is universal: it is not restricted to one particular certificate scheme, nor does it restrict the multiparty computation scheme. Its description is done in such a way that separate discussions on the multiparty computation protocols and the certificate schemes complete the solution to the main problem of this thesis.

The multiparty computation protocols are discussed in Chapter 4. This chapter mainly includes an efficient protocol for multiparty modulo reduction in Section 4.3, as well as interesting protocols for Paillier to ElGamal conversion and extended plaintext equality tests in Sections 4.1 and 4.2, respectively. Section 4.5 briefly discusses multiparty computation of statistics.

The main point of this thesis, the extension of certificate schemes, is divided into three chapters. Firstly, Chapter 5 includes a reformulated definition of certificate schemes, as well as a revision of that definition in order to make it applicable for encrypted certificate schemes. Section 5.2 briefly

[6]This means that the user can for instance show that age ≥ 18, without revealing any other data in the passport.

discusses the certificate scheme of Brands, which is the main building block of our scheme. Chapter 6 considers the extension of Brands' scheme to multiparty computation with ElGamal encryption. More concretely, the components **A**-**C** are constructed for in case the underlying cryptosystem is ElGamal. In particular the algorithms and protocols for scheme **C** are discussed very detailed, together with an extensive security discussion. In Chapter 7, protocols **A** and **B** are constructed for the Paillier cryptosystem. We also considered scheme **C** for the Paillier cryptosystem, but did not find a provably secure scheme. Therefore, a construction of this scheme remains as an open problem.

Finally, Chapter 8 discusses some remarks on the three certificate schemes, including an extensive efficiency analysis in which the introduced schemes and protocols are compared mutually and with respect to Brands' scheme. This chapter also concludes this thesis, and shows possible directions for further research.

2. Preliminaries

In this chapter, basic cryptographic tools and concepts are discussed. The reader is assumed to be familiar with many of these notions, and therefore this discussion is done concisely. Sections 2.1-2.5 discuss the cryptographic basics, including the relevant commitment schemes and cryptosystems, and the security model on which the thesis is built. Sections 2.6-2.7 discuss some basic cryptographic primitives, in particular 'proofs of knowledge' and 'signature schemes'. Finally, Sections 2.8-2.9 contain a concise survey of certificate schemes and multiparty computation. Readers well acquainted with these subjects can skip directly to Chapter 3.

2.1 Cryptographic tools

A very concise introduction to the basic cryptographic tools, including some group theory, is discussed following [Sch09, Ch. 1]. In particular, only the definitions of significant importance (for the thesis or for understanding the used notations) are introduced here. Definitions and notations for complexity theory are completely taken from literature.
Participants are denoted with calligraphic letters, e.g.: $\mathcal{P}$. If that participant is or might be malicious, it is denoted with an apostrophe: $\mathcal{P}'$. It becomes clear from the context in what degree $\mathcal{P}$ is malicious (see Section 2.3.1). Sets of participants are denoted with Latin capital letters, e.g.: $P = \{\mathcal{P}_1, \ldots, \mathcal{P}_n\}$.
A *p.p.t. Turing machine*, or 'probabilistic polynomial time' Turing machine, is a Turing machine that on input of x halts within $p(|x|)$ time steps, where p is some polynomial function, and which is equipped with a random tape. Throughout this thesis, all algorithms are considered to be p.p.t. Turing machines, unless explicitly stated. A *protocol* is a distributed algorithm, describing the interactions between two or more entities. By '$x \leftarrow \text{prot}_{\mathcal{A}(s_a);\mathcal{B}(s_b)}(p)$' we denote an execution of the protocol 'prot' for participants $\mathcal{A}$ and $\mathcal{B}$ with private inputs s_a and s_b respectively, and with common input p, which results in output x. This output is meant for one or more of the involved parties, depending on the context. During a protocol execution, each participant also learns a *view*, which is the set of all information that a party has 'seen' during the protocol execution. This view is implicit output of the protocol execution. A *scheme* is an ensemble of algorithms and protocols.
Throughout, 'log' always denotes the discrete logarithm with respect to some base, e.g. $\log_g h$ (introduced in Section 2.1.1). In case of taking logarithms over the reals, we only consider base 2, and to this end we use the notation 'lg'.

2.1.1 Group theory

Formally, a *group* is a structure $(G, *, 1, x \mapsto x^{-1})$ consisting of a set G, an associative operation $*$ with unit element 1, and an operation $x \mapsto x^{-1}$ such that for all $x \in G$ we have $x^{-1}*x = x*x^{-1} = 1$. Normally, we call G itself the group, and assume the group operation understood. In this thesis only *finite cyclic groups*, implicitly still called 'groups', will be discussed. These are groups G that can be generated by a single element $g \in G$, we write $G = \langle g \rangle$, such that there exists an $n \in \mathbb{N}$, called the *order*, satisfying $g^n = 1$ but $g^m \neq 1$ for all $1 \leq m < n$. Concretely this means that

$$G = \langle g \rangle = \{g^0, g^1, \ldots, g^{n-1}, g^n = g^0 = 1\} = \{g^i \mid i \in \mathbb{Z}_n\},$$

and therefore many computations will simply be done over $\mathbb{Z}_n$. The set $\mathbb{Z}_n^*$ is defined as the set of all elements $x \in \mathbb{Z}_n$ with $\gcd(x, n) = 1$. These elements have a multiplicative inverse in $\mathbb{Z}_n$, almost by definition of the gcd. By definition of the Euler totient function ϕ, we have $|\mathbb{Z}_n^*| = \phi(n)$. More definitions and properties regarding groups can be found in [CCS99].

For an element $h \in \langle g \rangle$, the unique value $x \in \mathbb{Z}_n$ such that $h = g^x$ is called the *discrete log* of h, or simply $\log_g h$. It is believed that finding x given (g, h, n) is hard if n is prime (see Assumption 2.2.1). By 'hard' we mean that no p.p.t. algorithm can compute x, but as a direct consequence of the definition of 'p.p.t. algorithm' we require n to be of exponential size. Therefore we introduce a *security parameter* $k = \lfloor \lg n \rfloor$. As of now, for prime values we will use p, q. Composite numbers are commonly denoted by m.

2.1.2 Probability theory

For an n we consider $\mathbb{Z}_n$. By '$x \in_R \mathbb{Z}_n$' we denote that x is taken uniformly at random from the set $\mathbb{Z}_n$. A basic property is that for $x, y \in_R \mathbb{Z}_n$ we also have $x + y \in_R \mathbb{Z}_n$. Also, if $y \in_R \mathbb{Z}_n^*$, then $xy \in_R \mathbb{Z}_n$. The proofs are trivial probability theory. However, if both $x, y \in_R \mathbb{Z}_n$, the value $xy \in \mathbb{Z}_n$ is not anymore uniformly at random: suppose for example that $n = q$ prime, and consider $x, y \in_R \mathbb{Z}_q$. Then for any $\alpha \in \mathbb{Z}_q$:

$$Pr(xy = \alpha) = \begin{cases} \frac{2q-1}{q^2}, \text{ if } \alpha = 0; \\ \frac{q-1}{q^2}, \text{ otherwise.} \end{cases} \tag{2.1}$$

But one immediately sees that for q large xy is *almost* randomly distributed over $\mathbb{Z}_q$. In order to formalize this, we need the notion of '*negligibility*' which can be found in [Sch09]. Without going into detail, we pose that the function $k \mapsto 1/2^k$ is negligible. It now also follows that the distribution (2.1) is indeed negligibly close to uniformly random: we define $f : \mathbb{N} \to \mathbb{R}$ to be the maximum difference between the function (2.1) and the uniformly random probability density function:

$$f : q \mapsto \max_{\alpha \in \mathbb{Z}_q} \left| Pr(xy = \alpha \mid x, y \in_R \mathbb{Z}_q) - \frac{1}{q} \right|. \tag{2.2}$$

One observes that $f(q) \leq \frac{q-1}{q^2} < 1/2^k$ (remember that we have $q > 2^k$ for some k), which is indeed negligible as $1/2^k$ is negligible. These types of *blinding* (additive and multiplicative) turn out to be very useful, for instance in blind signature schemes (Section 2.7). They will be used implicitly throughout the thesis.

In many cases, we just say that a function is negligible, by writing '$f(k) < \varepsilon(k)$' , where k is the security parameter. A function is called 'non-negligible', if it is not negligible. An event A will happen '*with negligible probability*' if $Pr(A) < \varepsilon(k)$, and '*with overwhelming probability*' if $1 - Pr(A) < \varepsilon(k)$.

Other relevant notions are 'statistical distance' and 'indistinguishability'. These will only be introduced briefly. Indistinguishability was introduced by Goldwasser and Micali [GM84], this notation is taken from [Sch09].

Definition 2.1.1 (Statistical distance)**.** Let X, Y be random variables two random variables. Then the *statistical distance* $\Delta(X, Y)$ is defined to be the following quantity:

$$\Delta(X, Y) = \frac{1}{2} \sum_{v \in V} \Big| Pr(X = v) - Pr(Y = v) \Big|,$$

where V denotes all possible outcomes of X and Y.

Definition 2.1.2 (Indistinguishability)**.** Let $X = \{X_i\}_{i \in L}$ and $Y = \{Y_i\}_{i \in L}$ be two probability ensembles. These ensembles X and Y satisfy the following level of *indistinguishability*:

- *perfectly indistinguishable*, if $\Delta(X_i, Y_i) = 0$ for all $i \in L$;
- *statistically indistinguishable*, if $\Delta(X_i, Y_i)$ is negligible in $|i|$ for all $i \in L$;
- *computationally indistinguishable*, if for all p.p.t. distinguishers $\mathcal{D}$ we have that $\mathrm{adv}_{\mathcal{D}}(X_i, Y_i)$ is negligible in $|i|$ (for all $i \in L$), where

$$\mathrm{adv}_{\mathcal{D}}(X_i, Y_i) = \Big| Pr(1 \leftarrow \mathcal{D}(X_i)) - Pr(1 \leftarrow \mathcal{D}(Y_i)) \Big|.$$

In other words: X and Y are perfectly indistinguishable if X_i and Y_i are identically distributed for all $i \in L$, statistically indistinguishable if the distribution of X_i is negligibly close to Y_i, and computationally indistinguishable if no polynomial time algorithm can see a significant difference.

Clearly, perfectly indistinguishable implies statistically/computationally indistinguishably. Also, if X, Y are statistically indistinguishable, they are computationally indistinguishable [Sch09]. In many cases it suffices to state that X and Y are 'indistinguishable', regardless of which kind. This will be denoted by '$X \overset{d}{\approx} Y$'.

2.1.3 Hash functions

For the construction of our certificate scheme, we use a cryptographic hash function.

Definition 2.1.3. A function $\mathcal{H} : \{0,1\}^* \to \{0,1\}^n$, mapping bit strings of arbitrary length to a bit string of length n, for some predefined $n \in \mathbb{N}$, is called a *cryptographic hash function* if it can be evaluated efficiently and satisfies the following properties:

- *pre-image resistance*: given a $y \in \{0,1\}^n$ it is hard to find a bit string x such that $\mathcal{H}(x) = y$;
- *second pre-image resistance*: given a tuple $(x, \mathcal{H}(x))$, it is hard to find a bit string $x' \neq x$ such that $\mathcal{H}(x') = \mathcal{H}(x)$;
- *collision resistance*: it is hard to find two different bit strings x, x' such that $\mathcal{H}(x') = \mathcal{H}(x)$.

Without going into detail, we assume we have a cryptographic hash function at our disposal. We assume that this function is modeled as a random oracle over $\mathbb{Z}_q$ [BR93].

2.2 Assumptions

Just like many cryptographic systems around, the protocols in this thesis are based on some assumptions. In the previous section, the 'random oracle model' (ROM) is introduced, which guarantees nice properties of the cryptographic hash function $\mathcal{H}$. Normally, this assumption does not suffice to assure security and most cryptographic protocols are only secure under one or more computationally hard problems. The most relevant assumptions for this thesis are discussed here. Less important assumptions, that are only relevant in a certain part of the thesis will be introduced where needed. For the discrete log family of cryptographic assumptions, we will consider the discrete log (DL), the Diffie-Hellman (DH), the decisional DH (DDH) and the DL representation (DLREP) assumptions, and for the prime factoring family we will consider the Rivest-Shamir-Adleman (RSA), the strong RSA (SRSA) and decisional composite residuosity (DCR) assumptions. The reader is referred to [DH76, Bon98, Bra99, BDZ03, RSA78, Pai99, DJ01] for more extended discussions. In Section 2.2.3, these assumptions are compared.

2.2.1 Discrete log family

For the discrete log family, we consider a group $G = \langle g \rangle$ of prime order q, with length security parameter k. Therefore, all probabilities in the coming assumptions are taken over all constructions of (q, g) such that $q > 2^k$. The discrete log assumptions became of interest after the paper by Diffie and Hellman [DH76].

Assumption 2.2.1 (The discrete log assumption (DL))**.** For any p.p.t. algorithm $\mathcal{A}$, there exists a negligible $\varepsilon(k)$ such that

$$Pr(x \leftarrow \mathcal{A}(q, g, g^x) \mid x \in_R \mathbb{Z}_q) < \varepsilon(k).$$

In other words: given g and $g^x \in_R G$, it is hard to compute x.

Related to the discrete log assumption are the DH and the DDH assumptions. These extensions are studied in [Bon98], variations of these two assumptions are discussed in [BDZ03]. It is clear that DL $\Leftarrow$ DH $\Leftarrow$ DDH.

Assumption 2.2.2 (The Diffie-Hellman assumption (DH))**.** For any p.p.t. algorithm $\mathcal{A}$, there exists a negligible $\varepsilon(k)$ such that

$$Pr(g^{xy} \leftarrow \mathcal{A}(q, g, g^x, g^y) \mid x, y \in_R \mathbb{Z}_q) < \varepsilon(k).$$

In other words: given $g^x, g^y \in_R G$, it is hard to compute g^{xy}.

Assumption 2.2.3 (The decisional DH assumption (DDH))**.** For any p.p.t. algorithm $\mathcal{D}$, there exists a negligible $\varepsilon(k)$ such that

$$\left| Pr(1 \leftarrow \mathcal{D}(q, g, g^x, g^y, g^z) \mid x, y, z \in_R \mathbb{Z}_q) - Pr(1 \leftarrow \mathcal{D}(q, g, g^x, g^y, g^{xy}) \mid x, y \in_R \mathbb{Z}_q) \right| < \varepsilon(k).$$

In other words: given $g^x, g^y \in_R G$, it is hard to distinguish g^{xy} from a random element from G.

Finally, for the certificate scheme in this thesis we need the DLREP assumption, by Brands [Bra99, Sec. 2.2.2]. It is proven by Brands that DLREP $\iff$ DL [Bra99, Prop. 2.3.3].

Assumption 2.2.4 (The discrete log representation assumption (DLREP))**.** Let $l \in \mathbb{N}$, and let $g_1, \ldots, g_l \in_R G$. Then, for any p.p.t. algorithm $\mathcal{A}$, there exists a negligible $\varepsilon(k)$ such that

$$Pr(\underline{0} \neq (x_1, \ldots, x_l) \leftarrow \mathcal{A}(q, g, g_1, \ldots, g_l) \text{ such that } g_1^{x_1} \cdots g_l^{x_l} = 1) < \varepsilon(k).$$

In other words: given a natural number l and a list $g_1, \ldots, g_l \in_R G$, it is hard to find a non-trivial list of values $x_1, \ldots, x_l \in \mathbb{Z}_q$ such that $g_1^{x_1} \cdots g_l^{x_l} = 1$.

2.2.2 Prime factoring family

In the setting for the prime factoring family, we consider a composite number $m = pq$ of length k. It is generally assumed that m is hard to factor. In order to withstand Pollard's $p - 1$ algorithm [Pol74], we need p and q to be such that $p - 1$ and $q - 1$ are non-smooth. For simplicity we just consider p and q to be *safe primes*, i.e. $p = 2p' + 1$ and $q = 2q' + 1$ for some primes p', q'. This family of assumptions became of big interest after the paper by Rivest, Shamir and Adleman [RSA78], who used it for their cryptosystem. All probabilities are taken over all constructions of $m = pq$, with p, q safe primes of length $k/2$.

Assumption 2.2.5 (The Rivest-Shamir-Adleman assumption (RSA))**.** For any p.p.t. algorithm $\mathcal{A}$, there exists a negligible $\varepsilon(k)$ such that

$$Pr(x \leftarrow \mathcal{A}(m, e, x^e) \mid x \in_R \mathbb{Z}_m, e \in_R \mathbb{Z}^*_{\phi(m)}) < \varepsilon(k).$$

In other words: given $x^e \in_R \mathbb{Z}_m$ for some $e \in_R \mathbb{Z}^*_{\phi(m)}$, it is hard to compute x.

Note that if $d := e^{-1} \bmod \phi(m)$ is known, the RSA assumption does not hold anymore as $(x^e)^d \equiv x^{1+\alpha\phi(m)} \equiv x \bmod m$ is polynomial time computable. However, by the factoring assumption, finding $e^{-1} \bmod \phi(m)$ is hard. It has actually been proven recently by Aggarwal and Maurer that in the generic group model the RSA and the factoring assumptions are equivalent [AM09].
An extension is the strong RSA assumption, first introduced by Barić and Pfitzmann and independently by Fujisaki and Okamoto [BP97, FO97], and widely used since, i.e. in certificate schemes [CL02] and integer commitment schemes [DF02]. The only difference with the RSA assumption is that e can be chosen freely now.

Assumption 2.2.6 (The strong RSA assumption (SRSA))**.** For any p.p.t. algorithm $\mathcal{A}$, there exists a negligible $\varepsilon(k)$ such that

$$Pr((x, e) \leftarrow \mathcal{A}(m, y) \mid y \in_R \mathbb{Z}_m, e > 1, y = x^e) < \varepsilon(k).$$

In other words: given a $y \in_R \mathbb{Z}_m$, it is hard to find non-trivial x, e such that $y = x^e$.

Another assumption related to RSA is the decisional composite residuosity assumption, which is the basis for the security of the Paillier cryptosystem [Pai99].

Assumption 2.2.7 (The decisional composite residuosity assumption (DCR))**.** For any p.p.t. algorithm $\mathcal{D}$, there exists a negligible $\varepsilon(k)$ such that

$$\left| Pr(1 \leftarrow \mathcal{D}(m, x^m) \mid x \in_R \mathbb{Z}_{m^2}) - Pr(1 \leftarrow \mathcal{D}(m, y) \mid y \in_R \mathbb{Z}_{m^2}) \right| < \varepsilon(k).$$

In other words: given a $y \in_R \mathbb{Z}_{m^2}$, it is hard to verify whether $y = x^m \bmod m^2$ for some x.

Clearly RSA $\Leftarrow$ SRSA. It has been proven by Stern [Ste99] that RSA $\Leftarrow$ DCR in case $e = m$.

2.2.3 Comparison

Many papers have been written concerning the lengths of the security parameter. The document [ECR08, Sec. 6.2] includes a survey of existing guidelines of these recommendations. Without going into detail, the recommended security parameter is:

- for DL, a group with prime order q with bit length varying between 141 and 160, for groups constructed as subgroups of appropriate finite fields or for elliptic curve constructions [Bra99, Sec. 2.2.2];
- for RSA, a modulus $m = pq$ with bit length varying between 1024 and 1536.

So, at first sight, DL would be more suitable, but it also has disadvantages, for instance RSA turns out false if the algorithm knows some additional information (the trapdoor), while no such trapdoor is believed to exist for discrete log systems.
The NIST (National Institute of Standards and Technology) suggests however to change to 224 bits and 2048 bits, as computers become more powerful. As of now, we will just assume the setup of the system parameters to be such that the protocols are sufficiently secure.

2.3 Multiparty computation security model

Multiparty computation involves a number of participants who jointly securely compute a function f. Informally, this computation should be 'correct' and 'secure'. Yao and Goldreich et al. [Yao82, Yao86, GMW87] were the first to study the security of multiparty protocols, and many followed [GL90, Bea91, MR92, Can95, Can00]. We consider the framework of Cramer, Damgård and Nielsen [CDN01], which is based on the security model of Canetti [Can00]. Except when explicitly stated, all security proofs of multiparty computation protocols in this thesis are based on this model. In Section 2.3.2, an informal description of this framework is given. Firstly, the possible adversarial settings are discussed in Section 2.3.1.

2.3.1 Adversaries

An *adversary* is a party that corrupts some subset of participants. The adversary is commonly denoted by $\mathcal{A}$, and the corrupted or malicious participants are denoted with an apostrophe: $\mathcal{P}'$. An adversary is either *active or passive*, and either *static or adaptive*. In case of passive corruption, the malicious participants follow the protocol, but the adversary obtains all their private information. In case of active corruption, the adversary takes full control over the cheating participants and may let them deviate from the protocol. In case of a static adversary, the subset of malicious participants lead by the adversary is fixed from the start. In case of an adaptive adversary, the corrupted subset is subject to change.
Apart from that, we can distinguish between an information-theoretic model and a cryptographic model. In an information-theoretic model, it is assumed that the players can communicate over pairwise secure channels, so the adversary only learns about communication involving at least one corrupted party. In the cryptographic model, all participants are assumed to have access to the data exchanged between the players. The important difference is that in the information-theoretic model the adversary may have unlimited computer power. Other variants in adversarial behavior are also possible [Can00].

Preferably, we want our multiparty computation scheme to be perfectly secure against active and dynamic adversaries, since this is the strongest variant. Many protocols can however only be proven secure against active and static adversaries. For instance, threshold decryption (defined in Section 2.5) is a building block in many multiparty protocols that is clearly not secure against *adaptive* adversaries. We will only consider active and static adversaries in the cryptographic model.

2.3.2 Multiparty computation model

Informally, in the security framework of [CDN01] we have n parties $P = \{\mathcal{P}_1, \ldots, \mathcal{P}_n\}$, of which $P' \subset P$ are corrupted by the adversary $\mathcal{A}$, who want to execute a protocol for a function f. The protocol is said to securely evaluate that function f if the real execution of the protocol can be simulated in such a way that the two executions (real and simulated) are indistinguishable. This simulator may use the malicious part P' as a subroutine. It practically means that anything an adversary can learn from the real execution of the protocol, he could have simulated himself as well. This model is commonly called the 'real/ideal-model', as the simulated executions can be considered to be ideal executions where the adversary cannot learn anything from the honest parties. This security model is hybrid in the sense that it suffices to prove separate sub-protocols secure. For a secure function evaluation, these sub-protocols can be executed sequentially (but not concurrently), and then the security of the function evaluation then holds via the security of the separate protocols. Thus, except explicitly stated, multiparty computation protocols are secure if they can be simulated given an in- and output. For a more formal description, the reader is referred to [Can00] and [CDN01, full version].

2.4 Commitment schemes

The idea of a commitment scheme is to allow someone to 'testify' to a value x, such that he cannot change his mind afterwards, but without leaking x to anyone. This idea was introduced by Blum [Blu81] and formalized by Brassard et al. [BCC88]. A commitment scheme generally exists of a *commit* phase, where a prover takes a uniformly random value r, computes $c \leftarrow \text{commit}(x, r)$ and publishes the result, and a *reveal* phase, where the prover publishes x and r in order to let the verifier check $c \stackrel{?}{=} \text{commit}(x, r)$. The scheme should be *hiding*, meaning that for any x, x' and uniform values r, r' the outcomes $\text{commit}(x, r)$ and $\text{commit}(x', r')$ are indistinguishable. It should also be *binding*, in the sense that it is hard for a prover to output non-trivial values x, x', r, r' such that $\text{commit}(x, r) = \text{commit}(x', r')$. Many commitment schemes are known nowadays, but we only make use of the Pedersen commitment scheme [Ped92] and integer commitments [DF02].

Definition 2.4.1 (Pedersen commitment scheme)**.** Let $G = \langle g \rangle$ be a group of prime order q and let $h \in G\backslash\{1\}$, for which $\log_g h$ is unknown. The Pedersen commitment scheme is defined by the function $C : \mathbb{Z}_q \times \mathbb{Z}_q \to G$ as

$$C(x, r) = g^x h^r,$$

where r is meant to be taken uniformly at random.

Without going into detail, the Pedersen commitment function is computationally binding (under the DL assumption) and perfectly hiding. In other words, once a user committed to a value he cannot change it, but still $C(x, r)$ does not leak any information about x. If we work over a group of composite order m, a similar commitment scheme can be found in [FO97].
The Pedersen commitment scheme can be used to show that for committed values a, b, c the equation $ab = c$ holds. However, as G is a group of order q, this equation only holds modulo q. To get around this problem, Damgård and Fujisaki introduced 'integer commitments' [DF02], that generalizes the previous work of Fujisaki and Okamoto [FO97]. Briefly, their integer commitment scheme is the following, taken from [BCL06].

Definition 2.4.2 (Integer commitment scheme)**.** Let $m = pq$ be a composite number of length security parameter k, with $p = 2p' + 1$ and $q = 2q' + 1$ safe primes. Let $h_1 \in \mathbb{Z}_m^*$ be an element of order $p'q'$, and $h_2 \in_R \langle h_1 \rangle$ with $\log_{h_1} h_2$ unknown. The integer commitment scheme is defined by the function $IC : \{0, 1\}^{\ell_x} \times \{0, \ldots, \lfloor m/4 \rfloor - 1\} \to \mathbb{Z}_m$ as

$$IC(x, r) = h_1^x h_2^r,$$

where r is meant to be taken uniformly at random. Parameter ℓ_x is supposed to satisfy $\ell_x + \ell_s = k/4$ for some security parameter ℓ_s.

The scheme is statistically hiding and computationally binding (under the SRSA assumption) [DF02]. It is a useful tool to prove properties of values over different groups.

2.5 Cryptosystems

The idea of a cryptosystem is clear: instead of sending a message in plaintext, it is encrypted so that an outsider cannot learn the message. A cryptosystem consists of three algorithms, a key generation algorithm together with an encryption and decryption algorithm. The main security issue is that a ciphertext does not leak *any* information about the encrypted plaintext. To formalize this, the notion of 'semantic security' is used, due to [GM84]. Many encryption schemes are known, e.g. [GM84, ElG85, CS98, Pai99, BGN06, DGK07]. These cryptosystems differ in several aspects, e.g. group settings, required computational assumptions. In this thesis, only the ElGamal and Paillier cryptosystems are considered, both of which are 'homomorphic' and 'threshold'. These cryptosystems are introduced in Sections 2.5.1 and 2.5.2 respectively. In Section 2.5.3, the security of the schemes is considered. Finally, Section 2.5.4 introduces some convenient notation. Firstly, the notions 'homomorphic cryptosystem' and 'threshold decryption' are discussed.

Homomorphic cryptosystem

Both the ElGamal and Paillier cryptosystem consider three groups: the message group $(\mathfrak{M}, *)$, the randomization group $(\mathfrak{R}, \circ)$ and the encryption group $(\mathfrak{C}, \star)$. For both schemes, we have encryption and decryption functions

$$E : \mathfrak{M} \times \mathfrak{R} \to \mathfrak{C}, \qquad D : \mathfrak{C} \to \mathfrak{M}.$$

The cryptosystem is called 'homomorphic' if both functions are homomorphisms. That is, for all $c, d \in \mathfrak{C}$ we have $D(c \star d) = D(c) * D(d)$, and similarly for all $(x, r), (y, s) \in \mathfrak{M} \times \mathfrak{R}$ we have $E(x, r) \star E(y, s) = E(x * y, r \circ s)$. Preferably, we want the functions to be additively homomorphic in the message parameter.

Threshold cryptosystem

A cryptosystem is called a (t, n)-*threshold cryptosystem* if the decryption key is shared among n participants $P = \{\mathcal{P}_1, \dots, \mathcal{P}_n\}$, such that any t participants can jointly decrypt, but any set of less than t participants cannot learn anything about the encrypted value. The notion of (t, n)-threshold is derived from the analogue from secret sharing, due to Shamir and Blakley [Sha79, Bla79]. The first to use this idea in the context of cryptosystems were Desmedt and Frankel [DF90]. Threshold variants for the ElGamal and Paillier cryptosystems are due to Pedersen [Ped91] and Damgård and Jurik [DJ01], respectively.

2.5.1 ElGamal cryptosystem

The original ElGamal cryptosystem was introduced by ElGamal [ElG85]. The current notation is taken from [Sch09]. On input of a security parameter k, a key generation algorithm outputs a group $G = \langle g \rangle$ of prime order $q > 2^k$, together with a secret key $\lambda \in_R \mathbb{Z}_q$ and corresponding public key $f = g^\lambda$. The encryption and decryption functions are defined as

$$\begin{aligned} E_G &: G \times \mathbb{Z}_q \to G^2, & D_G &: G^2 \to G, \\ E_G &: (x, r) \mapsto (g^r, xf^r), & D_G &: (a, b) \mapsto b/a^\lambda. \end{aligned}$$

Clearly, the cryptosystem is correct: for any $x \in G, r \in \mathbb{Z}_q$ we have

$$D_G(E_G(x, r)) = D_G(g^r, xf^r) = xf^r/(g^r)^\lambda = x(f/g^\lambda)^r = x.$$

The cryptosystem is indeed homomorphic in the message parameter, yet multiplicatively. By applying the isomorphism $x \mapsto g^x$, mapping $(\mathbb{Z}_q, +)$ to $(G, \cdot)$, the encryption function changes to $E_G : \mathbb{Z}_q \times \mathbb{Z}_q \to G^2$ defined as

$$E_G : (x, r) \mapsto (g^r, g^x f^r).$$

Consequently, the decryption function needs to be changed as well by taking $D_G(a, b) = \log_g(b/a^\lambda)$, but this only works if x is from a restricted domain, by the DL Assumption 2.2.1.

The ElGamal cryptosystem can be easily turned into an efficient threshold variant. A (t, n)-threshold scheme for the ElGamal cryptosystem is due to [Ped91], and can be found in [Sch09, Ch. 6]. Now, each participant $\mathcal{P}_i$ has a polynomial decryption share λ_i $(i = 1, \dots, n)$. To avoid the use of a dealer for the key generation, an efficient method for 'distributed key generation' (DKG) [GJKR99] can be used.

2.5.2 Paillier cryptosystem

The Paillier cryptosystem was introduced by Paillier [Pai99], and generalized by Damgård and Jurik [DJ01]. We use the generalization. On input of a security parameter k, a key generation algorithm outputs a composite number $m = pq$ of length k, with $p = 2p' + 1$ and $q = 2q' + 1$ safe primes, and a value $s \in \mathbb{N}$. We consider the group $\mathbb{Z}^*_{m^{s+1}}$ and we have as message space $\mathbb{Z}_{m^s}$. The public key is (m, s), with secret key a value μ satisfying $\mu \equiv 1 \bmod m^s$ and $\mu \equiv 0 \bmod \operatorname{lcm}(p-1, q-1)$. The encryption and decryption functions are defined as

$$\begin{aligned} E_P &: \mathbb{Z}_{m^s} \times \mathbb{Z}^*_{m^{s+1}} \to \mathbb{Z}^*_{m^{s+1}}, & D_P &: \mathbb{Z}^*_{m^{s+1}} \to \mathbb{Z}_{m^s}, \\ E_P &: (x, r) \mapsto (m+1)^x r^{m^s} \bmod m^{s+1}, & D_P &: c \mapsto L(c^\mu \bmod m^{s+1}), \end{aligned}$$

Where $L : (1+m)^{x\mu} \bmod m^{s+1} \mapsto x \bmod m^s$ is a well-defined function [DJ01, Sec. 3]. Clearly, the cryptosystem is correct: for any $x \in \mathbb{Z}_{m^s}, r \in \mathbb{Z}^*_{m^{s+1}}$ we have

$$D_P(E_P(x, r)) = L((m+1)^{x\mu} r^{\mu m^s} \bmod m^{s+1}) = L((m+1)^{x\mu} \bmod m^{s+1}) = x \bmod m^s,$$

where the second equality holds as $r^{\mu m^s} \equiv 1 \bmod m^{s+1}$ due to the properties of Carmichael's function: Carmichael's function is $\lambda(m) = \operatorname{lcm}(p-1, q-1)$, and for each a such that $\gcd(a, m) = 1$

we have $a^{\lambda(m)} \equiv 1 \bmod m$. But $\lambda(m^{s+1}) = m^s\lambda(m)$ and as $\lambda(m)|\mu$, this equation is implied.

The scheme is indeed homomorphic, in the desired fashion. To this end, observe that for any $x_1, x_2 \in \mathbb{Z}_{m^s}$ and any $r_1, r_2 \in_R \mathbb{Z}^*_{m^{s+1}}$ we have:

$$E_P(x_1, r_1)E_P(x_2, r_2) = (m+1)^{x_1} r_1^{m^s} (m+1)^{x_2} r_2^{m^s} = (m+1)^{x_1+x_2}(r_1r_2)^{m^s} = E_P(x_1 + x_2, r_1r_2).$$

Damgård and Jurik showed that their extension to the Paillier cryptosystem can be easily extended to an efficient (t, n)-threshold variant [DJ01, Sec. 4.1]. Now, the decryption key is taken such that $\mu \equiv 1 \bmod m^s$ and $\mu \equiv 0 \bmod p'q'$, and each participant $\mathcal{P}_i$ has a polynomial share μ_i $(i = 1, \ldots, n)$. However, distributed key generation protocols for Paillier are more complicated compared to for the discrete log case. Boneh and Franklin [BF97] were the first to investigate distributed key generation for shares of composite numbers pq, initiating a series of other works [FMY98, PS98, Gil99], yet none of them considered the case of p and q being *safe primes*. Algesheimer et al. [ACS02] constructed a key generation algorithm with p, q safe. Still, the generation of such a key is far less efficient than for the ElGamal cryptosystem.

2.5.3 Security

Standardly, encryption schemes are proven secure using the definition of *semantic security*, introduced by Goldwasser and Micali [GM84]. Many equivalent notions of semantic security are known [MRS87, TY98, DJ01, Sho04]. We follow [Sho04], but this choice is arbitrary.

Definition 2.5.1 (Semantic security)**.** Let (pk, sk) be a public/secret key pair generated on input of security parameter k. Let $\mathcal{A}$ be a p.p.t. algorithm attacking the cryptosystem (E, D). The scheme is called *semantically secure* if there exists a negligible $\varepsilon(k)$ such that for any $x_0, x_1 \in \mathbb{Z}_q$

$$Pr\Big(b \leftarrow \mathcal{A}(pk, x_0, x_1, c) \;\Big|\; b \in_R \{0,1\}; c \leftarrow E(x_b)\Big) < \frac{1}{2} + \varepsilon(k).$$

The ElGamal cryptosystem is proven semantically secure under the DDH Assumption 2.2.3 [TY98], and Paillier under the DCR Assumption 2.2.7 [DJ01, Thm. 1]. Trivially (as they are homomorphic), both schemes are only secure against chosen plaintext attacks, not against chosen ciphertext attacks where the adversary can do decryption oracle queries as well.

Using the model of Section 2.3, it turns out that for both ElGamal and Paillier the threshold decryption protocol can be simulated on input $E(x)$ and x, where the simulator may use the malicious part of P, being P', as a subroutine. For ElGamal, the simulated and real distributions turn out to be perfectly indistinguishable [ST04]. For Paillier, the distributions are statistically indistinguishable [DJ01, Thm. 2]. Both threshold cryptosystems are secure against static active adversaries in the cryptographic model (see Section 2.3.1).

2.5.4 Notation

Instead of $E_G(x, r)$ we will write $[\![x, r]\!]_G$, and similarly for Paillier. The subscript is left out if it is irrelevant. In many cases also the randomization r is not important, and left out as well. By '$c = c'$' we denote that the two encryptions are equal in $\mathfrak{C}$. If $c = [\![x, r]\!]$ and $c' = [\![x', r']\!]$, this just means that $(x, r) = (x', r')$. More interesting is the meaning of '$c \doteq c'$', saying that the both values *encrypt* the same value: $c \doteq c' \iff D(c) = D(c')$. In particular, $c = c' \Rightarrow c \doteq c'$, but the converse is not true. The notation will also be used if c and c' are from different encryption schemes, in which case we will say $[\![x]\!]_G \doteq [\![x']\!]_P \iff (x \bmod q) \equiv (x' \bmod m^s) \bmod \min\{q, m^s\}$. Under the restriction that $x', q < m^s$, this means that $x \equiv x' \bmod q$, a property that will be exploited in Sections 4.1 and 4.2.

2.6 Proofs of knowledge

Proofs of knowledge have been studied widely in the literature for years. They originate from Goldwasser et al. [GMR85], and studied extensively, a.o. in [FFS87, FS90, BG92, CDS94, Cra97]. Informally, a *zero-knowledge proof of knowledge* is a protocol for a prover to convince a verifier that he knows something, without leaking any information other than the value of the assertion that the prover is trying to prove (hence 'zero-knowledge'). More specifically, we have a relation $R = \{(x; w)\}$ and for an x, common input for the prover and verifier, the prover proves in zero-knowledge that he knows a value w (the '*witness*') such that $(x; w) \in R$. System parameters are normally not included in x. In this thesis, unless noted explicitly, we will only use the stronger but more practical definition of 'Σ-protocols', due to Cramer [Cra97]. A convenient definition can be found in [Sch09, Ch. 5] as well.

Definition 2.6.1 (Σ-protocol)**.** A Σ-protocol for a relation $R = \{(x; w)\}$ is a protocol between two parties $(\mathcal{P}, \mathcal{V})$, also called the 'prover' and 'verifier', that is a three-move proof of knowledge with conversations of the form (a, c, r) such that c chosen uniformly at random and upon receiving r, the verifier either accepts or rejects, and that satisfies the following properties:

- (Completeness.) For any $(x; w) \in R$ and $(\mathcal{P}, \mathcal{V})$ following the protocol, the verifier accepts;
- (Special soundness.) Given a common input x and two successful conversations (a, c, r) and (a, c', r') with $c \neq c'$, one can efficiently compute a witness w such that $(x; w) \in R$. In other words: $\mathcal{P}$ cannot prepare for two different challenge values, unless he knows the witness;
- (Honest-verifier zero-knowledge.) There exists a p.p.t. simulator that given any common input x produces conversations of the protocol execution that are indistinguishable from real conversations between honest $\mathcal{P}$ and $\mathcal{V}$, where $\mathcal{P}$ uses any witness w such that $(x; w) \in R$. If, additionally, the simulator takes c uniformly at random and simulates a conversation with that challenge c, it is called *special* honest-verifier zero-knowledge.

Σ-protocols provide many possibilities. A simple example is Schnorr's identification protocol given in Figure 2.1 [Sch91]. In this case, we have $G = \langle g \rangle$ of order q and a value $h \in \langle g \rangle$, and the prover proves knowledge of x such that $h = g^x$. More formally, Figure 2.1 is a Σ-protocol for the relation $\{(h; x) \mid h = g^x\}$. Using the Fiat-Shamir heuristic [FS87], the Schnorr protocol can be turned into a non-interactive Σ-protocol. Here, the prover takes $c \leftarrow \mathcal{H}(a)$ for some suitable hash function. This will become interesting in the following section, where signature schemes are discussed.

$$\begin{array}{rcl}
\textbf{Prover} & & \textbf{Verifier} \\
(\text{knows: } h; x) & & (\text{knows: } h) \\
u \in_R \mathbb{Z}_q, \quad a \leftarrow g^u & \xrightarrow{\quad\quad a \quad\quad} & \\
 & \xleftarrow{\quad\quad c \quad\quad} & c \in_R \mathbb{Z}_q \\
r \leftarrow u + cx \bmod q & \xrightarrow{\quad\quad r \quad\quad} & \\
 & & g^r \overset{?}{=} ah^c
\end{array}$$

Figure 2.1: Schnorr's identification protocol [Sch91].

2.6.1 Proof of knowledge with integer commitments

As already mentioned in Section 2.4, integer commitments are a powerful way to prove something over different groups. This will be used in the thesis, for instance in Sections 4.1 and 7.1. As an example, suppose we have two different groups $G_1 = \langle g_1 \rangle$ and $G_2 = \langle g_2 \rangle$ of prime orders q_1 and q_2, respectively. Prover $\mathcal{P}$ knows an x and published $y_1 \overset{G_1}{\leftarrow} g_1^x$ and $y_2 \overset{G_2}{\leftarrow} g_2^x$ and wants to prove that both commit to the same value (over the integers). Say $-2^{\ell_x} < x < 2^{\ell_x}$ for some parameter $\ell_x \ll k$, where k is the security parameter which is the bit length of m (see Definition 2.4.2). The idea is that $\mathcal{P}$ uses an integer commitment to commit to x by taking an $\tilde{r} \in_R \{0, \ldots, \lfloor m/4 \rfloor - 1\}$ and then executes a Σ-protocol for the relation

$$\{(y_1, y_2, y; x, \tilde{r}) \mid y_1 \overset{G_1}{=} g_1^x \wedge y_2 \overset{G_2}{=} g_2^x \wedge y \overset{\mathbb{Z}_m^*}{=} h_1^x h_2^{\tilde{r}} \wedge (-2^{\ell_x+\ell_c+\ell_s} < x < 2^{\ell_x+\ell_c+\ell_s})\},$$

where ℓ_c is the challenge length and ℓ_s a security parameter and we require that $2^{\ell_x+\ell_c+\ell_s+1} < \min\{q_1, q_2\}$. We require $-2^{\ell_x} < x < 2^{\ell_x}$ for the protocol to succeed. For making the scheme practical for proving, the construction of y is included in the protocol. The protocol can be found in Figure 2.2, taken from [BCL06], and is a Σ-protocol for relation

$$\{(y_1, y_2; x) \mid y_1 \overset{G_1}{=} g_1^x \wedge y_2 \overset{G_2}{=} g_2^x \wedge (-2^{\ell_x+\ell_c+\ell_s} < x < 2^{\ell_x+\ell_c+\ell_s})\}. \tag{2.3}$$

$$\begin{array}{rcl}
\textbf{Prover} & & \textbf{Verifier} \\
(\text{knows: } y_1, y_2; x) & & (\text{knows: } y_1, y_2) \\
\tilde{r} \in_R \{0, \ldots, \lfloor m/4 \rfloor - 1\}, \quad y \leftarrow h_1^x h_2^{\tilde{r}} & & \\
u \in_R \{0,1\}^{\ell_x+\ell_c+\ell_s} & & \\
v \in_R \{0,1\}^{k/4+\ell_c+\ell_s} & & \\
a_1 \overset{G_1}{\leftarrow} g_1^u, \quad a_2 \overset{G_2}{\leftarrow} g_2^u & & \\
a_y \overset{\mathbb{Z}_m^*}{\leftarrow} h_1^u h_2^v & \xrightarrow{y, a_1, a_2, a_y} & \\
 & \xleftarrow{c} & c \in_R \{0,1\}^{\ell_c} \\
r \leftarrow u + cx, \quad s \leftarrow v + c\tilde{r} & \xrightarrow{r, s} & \\
 & & g_1^r \overset{?}{=} a_1 y_1^c, \quad g_2^r \overset{?}{=} a_2 y_2^c \\
 & & h_1^r h_2^s \overset{?}{=} a_y y^c \\
 & & r \overset{?}{\in} \{0,1\}^{\ell_x+\ell_c+\ell_s}
\end{array}$$

Figure 2.2: A Σ-protocol using integer commitments [BCL06].

As an example, we prove correctness of this protocol completely. This proof will be used as a reference in future proofs.

Proposition 2.6.2. *The protocol in Figure 2.2 is a Σ-protocol for (2.3).*

Proof. (Completeness). For $(y_1, y_2; x)$ in relation (2.3) and honest $(\mathcal{P}, \mathcal{V})$, we prove that the verifier always accepts. For '$g_2^r \overset{?}{=} a_2 y_2^c$', the proof is similar to the first:

$$g_1^r = g_1^{u+cx} = g_1^u (g_1^x)^c = a_1 y_1^c,$$
$$h_1^r h_2^s = h_1^{u+cx} h_2^{v+c\tilde{r}} = h_1^u h_2^v (h_1^x h_2^{\tilde{r}})^c = a_y y^c.$$

Remains to prove that $r \in \{0,1\}^{\ell_x+\ell_c+\ell_s}$. For convenience, we denote $K := 2^{\ell_x+\ell_c}$. Note that $-K < cx < K$ and $0 \le u < K2^{\ell_s}$. So $r \in (-K, K2^{\ell_s} + K) =: R$, and this is negligibly close (in ℓ_s) to $[0, K2^{\ell_s}) =: S$: denote $X = \{x \mid x \in_R R\}$ and $Y = \{y \mid y \in_R S\}$, then the statistical distance $\Delta(X, Y)$ is (Definition 2.1.1):

$$\Delta(X,Y) = \frac{1}{2} \sum_{v \in R \cup S} \Big| Pr(X = v) - Pr(Y = v) \Big| = \frac{1}{2} \sum_{v \in S} \left| \frac{1}{|R|} - \frac{1}{|S|} \right| + \frac{1}{2} \sum_{v \in R \setminus S} \left| \frac{1}{|R|} - 0 \right|$$
$$= \frac{2K}{2K + |S|} = \frac{1}{1 + 2^{\ell_s - 1}} < 2^{-\ell_s+1}.$$

(Special soundness). Suppose we have two successful conversations $(y, a_1, a_2, a_y; c; r, s)$ and $(y, a_1, a_2, a_y; c'; r', s')$ with $c \neq c'$. In particular, this means that

$$h_1^r h_2^s = a_y y^c, \qquad\qquad h_1^{r'} h_2^{s'} = a_y y^{c'},$$

which upon combination result in $h_1^{r-r'} h_2^{s-s'} = y^{c-c'}$. Following [FO97], $c - c'$ *must* divide $r - r'$ and $s - s'$ over $\mathbb{Z}$, since otherwise we find a non-trivial factor of m. Thus we find $h_1^{\frac{r-r'}{c-c'}} h_2^{\frac{s-s'}{c-c'}} \overset{\mathbb{Z}_m^*}{=} y$, which gives us desired witnesses $x, \tilde{r}$. The value for x satisfies $y_1 \overset{G_1}{=} g_1^x$ and $y_2 \overset{G_2}{=} g_2^x$ as well. Remains to show that witness $x = \frac{r-r'}{c-c'}$ is in $(-K2^{\ell_s}, K2^{\ell_s})$. But $r, r' \in [0, K2^{\ell_s})$, so $r - r' \in (-K2^{\ell_s}, K2^{\ell_s})$, and as the factor $|c - c'| \geq 1$ makes no difference, we are done.
(Honest-verifier zero-knowledge). The distribution of the real and simulated conversations are, respectively:

$$\Big\{(y, a_1, a_2, a_y; c; r, s) \;\Big|\; \tilde{r} \in_R \{0, \ldots, \lfloor m/4 \rfloor - 1\}, y \leftarrow h_1^x h_2^{\tilde{r}}, u \in_R \{0,1\}^{\ell_x+\ell_c+\ell_s}, v \in_R \{0,1\}^{k/4+\ell_c+\ell_s},$$
$$c \in_R \{0,1\}^{\ell_c}; a_1 \overset{G_1}{\leftarrow} g_1^u, a_2 \overset{G_2}{\leftarrow} g_2^u, a_y \overset{\mathbb{Z}_m^*}{\leftarrow} h_1^u h_2^v; r \leftarrow u + cx, s \leftarrow v + c\tilde{r}\Big\},$$
$$\Big\{(y, a_1, a_2, a_y; c; r, s) \;\Big|\; c \in_R \{0,1\}^{\ell_c}, y \in_R \mathbb{Z}_m^*, r \in_R \{0,1\}^{\ell_x+\ell_c+\ell_s}, s \in_R \{0,1\}^{k/4+\ell_c+\ell_s};$$
$$a_1 \overset{G_1}{\leftarrow} g_1^r y_1^{-c}, a_2 \overset{G_2}{\leftarrow} g_2^r y_2^{-c}, a_y \overset{\mathbb{Z}_m^*}{\leftarrow} h_1^r h_2^s y^{-c}\Big\}.$$

But clearly $(y, a_1, a_2, a_y; c)$ are distributed perfectly indistinguishably. Only difference is in r, s. But in the real conversation we have $r \in_R (-K, K2^{\ell_s} + K)$ and in the simulated we have $r \in_R [0, K2^{\ell_s})$, and these two distributions are statistically indistinguishable. Similar for s. As in both distributions the c is chosen freely and uniformly at random, the protocol is *special* honest-verifier zero-knowledge. $\square$

2.6.2 Interval proof of knowledge

For the multiparty computation protocol in Section 4.3, a so-called 'interval proof of knowledge' is utilized. In this type of schemes, introduced by [BCDG87] and elaborated on by [Bou00], a prover proves that a committed value is in an interval $[a, b)$. We do not introduce the interval proof of knowledge here, we only mention its existence. This protocol operates in constant rounds and has broadcast complexity $O(k)$.

2.7 Signature schemes

The idea of digital signature schemes is that a signer can autograph a message, so that it can be verified by anyone else. Similarly to ordinary 'pencil-and-paper' signatures, the signer should be the only one capable of signing with his signature. This idea originates from Diffie and Hellman [DH76], and the first implementation is due to Rivest et al. [RSA78]. Informally, a signature scheme for a signer and a verifier consists of a key generation algorithm, a signing protocol that on input of a message m outputs a signature σ on it, and a verification algorithm. The signing protocol is in principle interactive, but can as well be non-interactive. The scheme should be secure, meaning that it is hard to 'forge' certificates in the sense that someone illegally can obtain a signature. Following Goldwasser et al. [GMR88], the strongest notion of security is that a scheme is secure against '*existential forgery under chosen message attacks*', meaning that it is hard to forge *at least* one signature, even under presence of a signing oracle. Other adversary models are considered in [GMR88] as well. As of now, following [PS00] we will use the notion of 'one-more forgery', meaning that the adversary queries the oracle on $K \geq 0$ messages and outputs $K + 1$ signatures. An interesting feature is that using the Fiat-Shamir heuristic, any Σ-protocol can be easily turned into a non-interactive signature issuing protocol, where $c \leftarrow \mathcal{H}(a)$ should be replaced by $c \leftarrow \mathcal{H}(a, m)$ for message m [FS87].

Additionally, in a *blind* signature scheme signing executions and issued signatures cannot be linked. Therefore, the property of 'blindness' is commonly called 'unlinkability' as well. Its idea is by Chaum [Cha83, Cha84], and is a basic ingredient in e-cash and certificate schemes. Concretely, it means that if a signer issues two signatures $(m, \sigma)_0, (m, \sigma)_1$, and receives for $b \in_R \{0, 1\}$ the signatures $(m, \sigma)_b, (m, \sigma)_{1-b}$ (in this order), he cannot guess b correctly with probability significantly larger than $\frac{1}{2}$. Clearly, blind signature schemes cannot be non-interactive. Formal definitions of (blind) digital signature schemes can be found in [GMR88, JLO97, Bra99, PS00, Lys02, CKW04].

Also a blind signature scheme can be easily obtained from a Σ-protocol [Sch09, Ch. 8]. Figure 2.3 shows the blind Schnorr signature scheme, deduced from Schnorr's identification protocol in Figure 2.1. The key generation algorithm outputs public key (q, g, h), and corresponding secret key x such that $h = g^x$ to the signer. The signature issuing protocol is given in Figure 2.3, which results in a signature (m, c', r') on m such that $c' = \mathcal{H}(m, g^{r'} h^{-c'})$, by which the verification algorithm is immediately specified. Note that the signer never learns m.

$$\begin{array}{rcl}
\textbf{Signer} & & \textbf{Verifier} \\
\text{(knows: } h; x) & & \text{(knows: } h) \\
u \in_R \mathbb{Z}_q & & \\
a \leftarrow g^u & \xrightarrow{\quad a \quad} & \\
 & & \alpha, \beta \in_R \mathbb{Z}_q \\
 & \xleftarrow{\quad c \quad} & c \leftarrow \mathcal{H}(m, a g^{\alpha} h^{\beta}) + \beta \bmod q \\
r \leftarrow u + cx \bmod q & \xrightarrow{\quad r \quad} & \\
 & & g^r \overset{?}{=} a h^c \\
 & & r' \leftarrow r + \alpha \bmod q \\
 & & c' \leftarrow c - \beta \bmod q
\end{array}$$

Figure 2.3: Blind Schnorr signature protocol [Sch09].

Another blind signature scheme being of great interest for this thesis is by Chaum and Pedersen [CP93]. In fact, it is an entanglement of two blind Schnorr signature schemes and it uses the same key generation. A signature on a message m is a tuple (m, z, c', r') such that $c' = \mathcal{H}(m, z, g^{r'} h^{-c'}, m^{r'} z^{-c'})$. The protocol is given in Figure 2.4.

2.7.1 Security of the blind signature schemes in Figures 2.3 and 2.4

The protocols in Figure 2.3 and 2.4 are supposed to induce blind signature schemes and hence need to satisfy unlinkability and unforgeability. Unlinkability is perfectly resp. statistically guaranteed due to the blinding methods. For unforgeability: a Chaum-Pedersen forgery is provably more difficult than a Schnorr forgery (this can be proven using similar techniques as in Section 6.3). However, the blind Schnorr signature scheme is not unforgeable if signatures may be issued in parallel! Pointcheval and Stern [PS00] showed for the witness-indistinguishable variant of the Schnorr signature scheme (the blind Okamoto-Schnorr signature scheme) that it is so-called 'strong one-more unforgeable', meaning that the scheme can only be one-more unforgeable if $K \leq (\lg k)^{\alpha}$ for some α, where K is the maximum number of parallel executions of the signing protocol, and $k = \lfloor \lg q \rfloor$ is the security parameter, as defined in Section 2.1.1. This theorem is based on the DL Assumption 2.2.1: a forgery to the signature scheme is reduced to breaking the DL assumption. Furthermore, Schnorr [Sch01] showed a method to forge signatures which cannot be translated to forging the DL assumption. It is related to the ROS assumption, roughly stated:

$$\begin{array}{rcl}
\textbf{Signer} & & \textbf{Verifier} \\
(\text{knows: } h; x) & & (\text{knows: } h) \\
 & \xleftarrow{\quad m_0 \quad} & t \in_R \mathbb{Z}_q^*, \quad m_0 \leftarrow m^{1/t} \\
z_0 \leftarrow m_0^x, \quad w_0 \in_R \mathbb{Z}_q & & \\
a \leftarrow g^{w_0}, \quad b \leftarrow m_0^{w_0} & \xrightarrow{\quad z_0;a,b \quad} & \\
 & & u \in_R \mathbb{Z}_q^*, \quad v \in_R \mathbb{Z}_q \\
 & & a' \leftarrow (ag^v)^u, \quad b' \leftarrow (b^t m^v)^u \\
 & & z \leftarrow z_0^t \\
 & & c' \leftarrow \mathcal{H}(m, z, a', b') \\
 & \xleftarrow{\quad c \quad} & c \leftarrow c'/u \bmod q \\
r \leftarrow w_0 + cx \bmod q & \xrightarrow{\quad r \quad} & \\
 & & g^r h^{-c} \stackrel{?}{=} a, \quad m_0^r z_0^{-c} \stackrel{?}{=} b \\
 & & r' \leftarrow (r+v)u \bmod q
\end{array}$$

Figure 2.4: Blind Chaum-Pedersen signature protocol [CP93].

Assumption 2.7.1 (ROS assumption (ROS))**.** Given a cryptographic hash function $\mathcal{H}$, modeled as a random oracle. It is hard to find an Overdetermined Solvable system of linear equations with Random values. That is, it is hard to find a matrix $A \in \mathbb{Z}_q^{(l+1)\times l}$ and $\mathbf{b} \in \mathbb{Z}_q^{l+1}$ with $b_i = \mathcal{H}(a_{i,1}, \ldots, a_{i,l})$ such that $A\mathbf{x} = \mathbf{b}$ is solvable modulo q.

Schnorr proved that if one can solve the ROS problem with non-negligible probability, then he also knows how to forge a blind Schnorr signature. In [Wag02], Wagner gave an algorithm for solving the ROS problem in a reasonably efficient way (but still of exponential order), exploiting the birthday attack.

Although upper bounds on the number of parallel executions are proven, unforgeability of the mentioned schemes is still an open problem, which as of now will just be conjectured to be true.

Remark 2.7.2. For later purposes we note that forging a Chaum-Pedersen signature (the remark also holds for the blind Schnorr signature scheme) is just as hard as forging a signature of the form (ξ, m, z, c', r') such that $c' = \mathcal{H}(\xi, m, z, g^{r'}h^{-c'}, m^{r'}z^{-c'})$ for any arbitrary bit string ξ. This is due to the properties of the cryptographic hash function.

2.8 Concise survey of certificate schemes

The onset of certificate schemes was put by Chaum [Cha83], and later by Damgård [Dam88]. Both papers concern themselves with payment systems, like many following papers [Bra93, Bra95a, CFN90], which are a special kind of certificate schemes. Later, more general certificate schemes became of interest, where a service provider issues a 'certificate' (in the literature, commonly the word 'credential' is used, which in the context of this thesis means the same) to a user. This certificate is in fact a signature on data about the user, e.g. name and date of birth. Pioneers in the area of certificate schemes are Brands [Bra93, Bra95a, Bra95b, Bra95c, Bra02], with his PhD dissertation [Bra99] comprising all of them, Lysyanskaya et al. [LRSW99], Camenisch and Lysyanskaya [CL01, CL02, CL04] and Verheul [Ver01]. Recently, Belenkiy et al. [BCKL08] constructed a non-interactive certificate scheme. The main security aspects a certificate scheme should satisfy are security and privacy. The former roughly means that the certificates are unforgeable and the second that different activities are unlinkable and do not leak information about the data being certified (unless if participants cheat, e.g. by sharing their certificates). For our purposes, we also

need the possibility to include multiple attributes in the certificate, and to selectively disclose some of those attributes. These properties are satisfied by the schemes in [Bra99] and the extension to [CL02, CL04] by [BCL06]. We briefly compare these schemes.
Brands' work includes a DL and an RSA variant, while [CL02, CL04] (abbreviated to 'CL' in this paragraph) need slightly stronger assumptions (the SRSA and the LRSW[1], respectively). The [CL04] system is based on a bilinear mapping. Brands' work is in the random oracle model, which is in the CL-work only optional for obtaining non-interactive proofs (using the Fiat-Shamir heuristic). The unlinkability property is satisfied by all schemes. Also, sharing of certificates would require sharing private values. However for unforgeability, the CL-work is proven secure under the underlying assumptions, while Brands' work is only assumed to be secure. Another difference is that Brands' schemes concern one-show certificates, which means that the certificates can only be used once. CL's certificates are multi-show, they can be used a multiple number of times. One-show certificates are useful for e-cash systems. Without going into detail, [BDD07, Sec. 2.2] compares the DL variant of [Bra99] with [CL02, CL04]. They claim that Brands' scheme is the most efficient. In particular, recertification (or issuing more certificates at once) can be done efficiently in Brands' scheme. This thesis is based on one of Brands' certificate schemes, as argued in Section 3.4.

2.9 Concise survey of secure multiparty computation

Multiparty computation (MPC) was introduced in the millionaires paper by Yao [Yao82]. It embraces the idea of n participants jointly computing a function (Yao's protocol for secure two-party computation is proven secure by Lindell and Pinkas [LP04]). Using the definitions introduced in Section 2.3.1, Ben-Or et al. [BGW88] showed that in the information-theoretic model every function can be securely computed in the presence of an adaptive passive (resp. active) adversary if and only if the adversary corrupts $< n/2$ (resp. $< n/3$) participants. For the cryptographic model, Goldreich et al. [GMW87] showed that, assuming the existence of trapdoor one-way permutations, every function can be securely computed if a static and active adversary corrupts $< n/2$ players. We already considered the security framework of Cramer et al. in Section 2.3, which is used for almost all multiparty computation protocols throughout this thesis.

After Yao's paper, research on multiparty computation mainly grew in two directions. On the one hand, there is multiparty computation based on verifiable secret sharing. Of interest is the paper by Cramer et al. [CDM00], which shows how to get a multiparty computation scheme from any monotone span program. Also a paper by Damgård et al. [DFK$^+$06] is interesting, which includes a whole spectrum of constant round complexity multiparty protocols for secret sharing (e.g. bit representation, random bit and modulo reduction).
The other branch of the tree is multiparty computation based on threshold homomorphic encryption, after the papers by Franklin and Haber [FH96] and the independent papers by Jakobsson and Juels [JJ00] and Cramer, Damgård and Nielsen [CDN01]. These three papers all introduce a scheme for securely computing any circuit, but differ in a few aspects: while [FH96] only considers passive adversaries, [JJ00, CDN01] consider active adversaries as well. Moreover, the encryption scheme used by Franklin and Haber requires all participants to help upon decryption. The mix and match scheme by Jakobsson and Juels relies on mixing truth tables of gates in a boolean circuit, and is applicable to any discrete logarithm based encryption scheme. The framework of [CDN01] is restricted to an RSA-setting, but can be used for any arithmetic circuit and moreover has a smaller round complexity than the approach in [FH96]. In this branch, most of the research concerns the ElGamal (additively homomorphic variant) and Paillier cryptosystems, although there are some exceptions [BGN06, DGK07]. By the homomorphic property, both cryptosystems support addition and multiplication with known constant. Cramer et al. show how to construct a secure multiplication gate, $([\![x]\!], [\![y]\!]) \mapsto [\![xy]\!]$, for general threshold homomorphic encryption schemes. However, the ElGamal scheme is not considered general in this context, and under the

[1]Roughly stated, given g^x, g^y and $m \in \mathbb{Z}_q$, it is hard to find a triple (a, a^y, a^{x+mxy}). Clearly, DH $\Leftarrow$ LRSW.

DH Assumption 2.2.2 it is even impossible to construct an efficient multiplication gate [ST04]: otherwise, on input of g^x, g^y for $x, y \in_R \mathbb{Z}_q$ one can set $[\![x]\!] \leftarrow (g^r, g^x f^r)$, $[\![y]\!] \leftarrow (g^s, g^y f^s)$, send these encryptions to the multiparty computation servers in order to obtain $[\![xy]\!]$, which upon decryption would result in g^{xy}.[2] This is a serious drawback of the ElGamal encryption scheme, and although [ST04] show that a multiplication gate for ElGamal *conditionally* is possible, most of the later papers use Paillier encryptions (in spite of the fact that distributed key generation for Paillier is less efficient)[3]. Some protocols relevant for this thesis are the following. Some of the protocols by Damgård et al. [DFK⁺06] also carry over to the threshold homomorphic encryption setting. The notion $[\![x]\!]^{b(\ell)}$ denotes the set $\{[\![x_0]\!], \ldots, [\![x_{\ell-1}]\!]\}$ of ℓ encrypted bits of x.

- A random value gate and random bit gate, which output an encrypted random value and bit respectively. For generating a random encrypted bit, several protocols are known differing in their complexities. One can minimize the round complexity using a technique of [CDN01], or minimize broadcast complexity using a technique of [ST06];
- A bitwise addition circuit, which given bitwise encryptions $[\![x]\!]^{b(\ell)}$ and $[\![y]\!]^{b(\ell)}$ outputs $[\![x + y]\!]^{b(\ell+1)}$ (including a final carry bit). Again, one can minimize with respect to broadcast or round complexity, following a technique of [ST06] or [DFK$^+$06], respectively;
- A bit representation gate, which outputs $[\![x]\!]^{b(\ell)}$ given $[\![x]\!]$. The protocol of [ST06] relies on constructing ℓ random bits and using an addition circuit, and its efficiency depends on these underlying primitives;
- A comparison gate, which given two encryptions $[\![x]\!]$ and $[\![y]\!]$ outputs an encrypted bit $[\![[x < y]]\!]$. Multiparty comparisons are typically done bitwise, for which the described bit representation gate can be deployed. Following [GSV07], a logarithmic round complexity protocol is known. [DFK$^+$06] contains a constant round complexity protocol but it has a considerably higher hidden constant, and for small values of ℓ (namely $\ell < 20$) the logarithmic round protocol is more efficient. The protocol in this thesis that requires a multiparty comparison gate, Section 4.3, is commonly concerned with a small value ℓ.

To summarize, some of the relevant building blocks for threshold homomorphic multiparty computation are given in Table 2.1, together with their complexities. We note that all these protocols are proven secure in the real/ideal-model from Section 2.3.2. These building blocks can be used to implement secure function evaluation protocols. Abusing the notation of the theory of boolean logic, such a building blocks are commonly called 'gates'. For notation, recall that we consider a (t, n)-threshold cryptosystem, and k is the security parameter defined in Section 2.1.1. Moreover we write $[\![x]\!]^{b(\ell)}$ to denote the set $\{[\![x_0]\!], \ldots, [\![x_{\ell-1}]\!]\}$ of encrypted bits of x.

We note that all protocols in the table, except for the threshold decryption protocol, result in encrypted output. The threshold decryption protocol results in plaintext output publicly known by every participant. It can also be the case that only one participant (or more generally, a restricted number of participants) is allowed to learn the decryption. In that case, the participants can deploy a *private output* protocol. As this type of protocol is used explicitly in the remainder of the thesis, e.g. in Sections 6.2 and 7.2, in Section 2.9.1 a protocol for private output is discussed.

2.9.1 Multiparty protocol for private output

In this section we briefly introduce a protocol for private output, taken from [ST04]. In a protocol for private output, an encryption $c = [\![x]\!]$ is decrypted such that only one participant learns the plaintext value x. We consider a set of n participants $P = \{\mathcal{P}_1, \ldots, \mathcal{P}_n\}$ who (t, n)-threshold

[2]Using pairings, [BGN06] shows a way in which multiplication in the group G *is* possible. This can only be done once, since the result will not be in G anymore, but in G_T (the image of the pairing). This approach however cannot be used for ElGamal encryptions since it makes the DDH assumption false in group G.

[3]Clearly, if for the ElGamal encryptions we work with the bit representation, so bitrep($[\![x]\!]$) and bitrep($[\![y]\!]$) are known, a general multiplication gate *is* possible. This is not considered in this work, simply because it is inefficient.

Function	broadcast[i]	round[i]	remark
add : $(\llbracket x \rrbracket, \llbracket y \rrbracket) \mapsto \llbracket x+y \rrbracket$	$O(1)$	$O(1)$	
pub. mult : $(\llbracket x \rrbracket, y) \mapsto \llbracket xy \rrbracket$	$O(1)$	$O(1)$	
pr. mult : $(\llbracket x \rrbracket, y) \mapsto \llbracket xy \rrbracket$	$O(k)$	$O(1)$	$\mathcal{P}_1$ knows y; [ST04]
thres : $\llbracket x \rrbracket \mapsto x$	$O(nk)$	$O(1)$	[DJ01, Sch09]
mult : $(\llbracket x \rrbracket_P, \llbracket y \rrbracket_P) \mapsto \llbracket xy \rrbracket_P$	$O(nk)$	$O(1)$	[CDN01][ii]
cmult : $(\llbracket x \rrbracket, \llbracket y \rrbracket) \mapsto \llbracket xy \rrbracket$	$O(nk)$	$O(n)$	$x \in \{0,1\}$; [ST04][ii]
rand : $\emptyset \mapsto \llbracket r \rrbracket$	$O(nk)$	$O(1)$	[ST06]
randbit : $\emptyset \mapsto \llbracket r \rrbracket$	$O(nk)$	$O(n)$	$r \in_R \{0,1\}$; [ST06]
	$O(n^2k)$	$O(1)$	$r \in_R \{0,1\}$; [CDN01]
bitadd : $(\llbracket x \rrbracket^{b(\ell)}, \llbracket y \rrbracket^{b(\ell)}) \mapsto \llbracket x+y \rrbracket^{b(\ell+1)}$	$O(nk\ell)$	$O(\ell)$	[ST06][iii]
	$O(nk\ell \lg \ell)$	$O(1)$	[DFK+06][iii]
bitrep : $(\llbracket x \rrbracket_P, \ell) \mapsto \llbracket x \rrbracket_P^{b(\ell)}$	$O(nk\ell)$	$O(n+\ell)$	[ST06]
	$O(n^2k\ell \lg \ell)$	$O(1)$	[ST06]
comp : $(\llbracket x \rrbracket^{b(\ell)}, \llbracket y \rrbracket^{b(\ell)}) \mapsto \llbracket [x<y] \rrbracket$	$O(nk\ell)$	$O(\lg \ell)$	[GSV07][iii]
	$O(nk\ell)$	$O(1)$	[DFK+06][iii]

[i]: broadcast and round complexity.
[ii]: is commonly denoted as $\llbracket x \rrbracket * \llbracket y \rrbracket$.
[iii]: in case of ElGamal encryptions we would need cmult gates, and hence have the round complexity multiplied with n.

Table 2.1: Several efficient multiparty computation gates.

share the secret key for an encryption scheme (ElGamal or Paillier, following Section 2.5). We have an encryption $c = \llbracket x \rrbracket$, whose encrypted value x may only be learned by $\mathcal{U}$ (who might be $\in P$). The protocol here discussed is interactive and works for both ElGamal and Paillier. It will be used explicitly in Sections 6.2 and 7.2. We note that also a non-interactive private output protocol is known, although it only works for the ElGamal cryptosystem [ST04]. This protocol is not considered in this thesis.

Protocol 2.9.1 (Private output)**.** Given $c = \llbracket x \rrbracket$, the following protocol outputs x to the user $\mathcal{U}$. User $\mathcal{U}$ and the participants $\mathcal{P}_i$ $(i = 1, \dots, n)$ perform the following steps:

1. $\mathcal{U}$ takes $r, t \in_R \mathbb{Z}_q$ (ElGamal) or $r \in_R \mathbb{Z}_{m^s}, t \in_R \mathbb{Z}^*_{m^{s+1}}$ (Paillier) and publishes $c' \leftarrow \llbracket r, t \rrbracket$ plus a proof of knowledge for relation $\{(c'; r, t) \mid c' = \llbracket r, t \rrbracket\}$;

2. The participants jointly compute $\llbracket x + r \rrbracket$ and threshold decrypt it, obtaining $d \leftarrow g^{x+r}$ (ElGamal) or $d \leftarrow x + r \bmod m^s$ (Paillier);

3. $\mathcal{U}$ computes $x \leftarrow \log_g(d/g^r)$ (ElGamal) or $x \leftarrow d - r \bmod m^s$ (Paillier).

The protocol is perfectly secure. If we use ElGamal, phase 3 can only be done efficiently if x is from a restricted domain, $x \in \pm\{0,1\}^{\ell_x}$ for some ℓ_x.

3. Detailed problem statement

This thesis is concerned with the problem of how to combine certificate schemes[1] and multiparty computation schemes. That is, we will construct protocols for users to obtain certificates on encryptions, and for service providers to certify encryptions. If we consider the certificate scheme as a black box which outputs 'some type of certificates', and the multiparty computation scheme as a black box which computes 'some result' on input of a list of encrypted values, then our problem is to find protocols for the communication between the boxes, and in particular a protocol for using a certificate on a list of attributes for constructing an encryption of one or more of the attributes and vice versa. Those attributes can be anything: name, date of birth, cardiac data, etc. A visualization of the problem is given in Figure 3.1. On the left-hand side, users can obtain certificates from service providers, and on the right-hand side, a set of computation parties P has the ability to securely evaluate functions. The 'certificates' going from left to right need to be accompanied with encryptions of the attributes, and the 'some output' going in the other direction might be certified (which implicitly means that the set of multiparty computation servers needs to be able to certify, hence is a service provider as well). The idea is that the users send certified encryptions to the multiparty computation participants, who do some computation (possibly statistical analysis) on these values and output the result in either encrypted or plaintext form. This output may be certified as well.

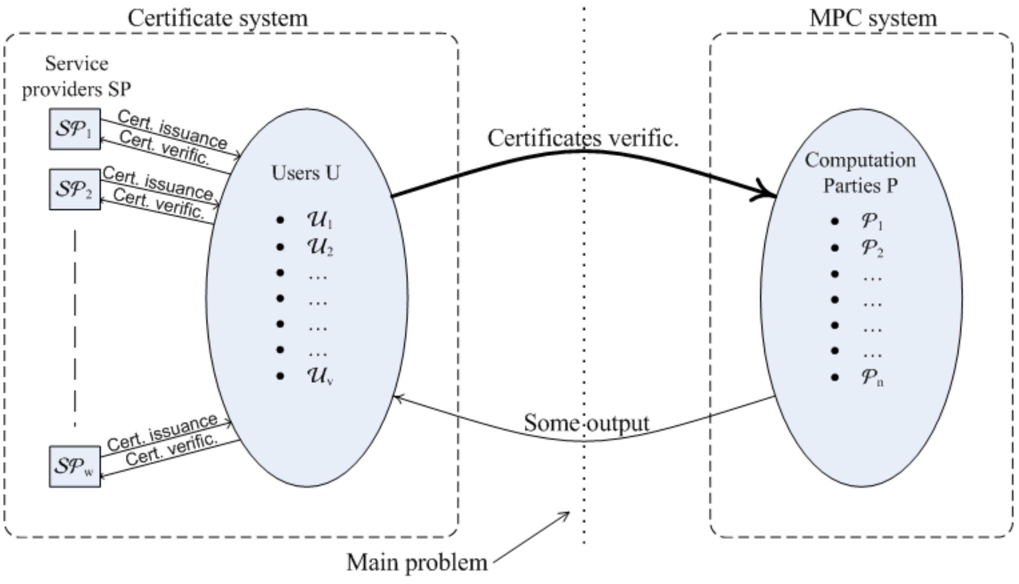

Figure 3.1: A high-level description of the problem.

[1]Recall that we use the word 'certificate scheme' rather than 'credential scheme', although they have the same meaning in the context of this thesis.

3.1 Participants

We can identify the following participants in Figure 3.1.

- *Users* $U = \{\mathcal{U}_1, \ldots, \mathcal{U}_v\}$: users obtain services from service providers. Notice that we consider both certificate issuance and verification as a type of service. In our scheme, the users provide the multiparty computation participants with encryptions, which might be certified;
- *Service providers* $SP = \{\mathcal{SP}_1, \ldots, \mathcal{SP}_w\}$: in the certificate scheme, the service providers provide services or certificates to users. Verification is seen as a service as well, and throughout this thesis the possibility to verify will be implicit. A verifier is also denoted as $\mathcal{V}$. A service provider can also be implemented as a group of participants, where a subset of the participants needs to be honest to offer the service correctly[2];
- *Computation parties* $P = \{\mathcal{P}_1, \ldots, \mathcal{P}_n\}$: the computation parties (in the context of multiparty computation simply called 'participants') jointly (t, n)-threshold share the secret key for an encryption scheme, and on input of encrypted values they evaluate secure multiparty protocols and output a result. This result can either be in plaintext or encrypted form. In general, it is assumed to be in encrypted form. In case only one participant, or another single entity may learn the plaintext, the participants can use a private output protocol, which guarantees that only one participant learns the outcome. In Section 2.9.1, a private output protocol is discussed.

Notice that there is some overlap among the different sets of participants. Most importantly, the set P is in fact a service provider. Indeed, the computation participants offer a 'service' to the users (notice that P is indeed implemented as a group of participants that threshold share a secret key). It can moreover be the case that the computation participants jointly issue a certificate on the output of their multiparty computation.

Remark 3.1.1. Possibly, a revoker can be added to the multiparty computation scheme. This revoker is one single party, or more generally a set of parties, that can decrypt as well (next to any t of the n participants), and can in particular revoke the anonymity of a user sending encryptions in case of danger/urgency. We do not consider such a structure, and assume that revocation is just implemented as a protocol for the n participants. For instance, in case anonymity has to be revoked if the encrypted value x is larger than a certain value α, the participants can execute a comparison protocol for $[x > \alpha]$ and decrypt the result. Also the certificate scheme might include a revocation algorithm, to revoke the anonymity of a cheating participant (e.g. in e-cash systems in case a participant spends a coin twice). We will not consider revocation algorithms for the certificate scheme.

3.2 Algorithms and protocols

The main algorithms and protocols we need for the system depicted in Figure 3.1 are the following. We first consider the certificate and multiparty computation schemes separately, and then the interface between them.

3.2.1 Certificate scheme

At a high level, for a certificate scheme, the left-hand side of Figure 3.1, we need three components:

- (Key generation.) An algorithm that, on input of a security parameter, outputs public and secret keys, including the system parameters;
- (Certificate issuance.) A protocol for a service provider to issue a certificate to a user. This certificate might certify some attributes (standard ones like 'name' or 'date of birth', or more sensitive ones like 'bank deposit' or 'criminal record');

[2]This is a natural way to restrict the adversarial character of a service provider to passive behavior only.

- (Certificate verification.) A protocol for a verifier to verify a certificate owned by a user. We require the ability of the user to selectively disclose attributes in plaintext.

These components need to provide some extra properties. Of course, the certificates should be unforgeable, roughly meaning that if a user is issued $K \geq 0$ certificates, it is hard to output $K+1$ certificates, or even to output any certificate on a *different* list of attributes [Bra99, PS00, CL01, CL02, CL04]. It is desirable that protocol executions of issuance and verification are unlinkable, but there are examples in which there must be a possibility to link certificates. Consider Example 1.0.3: if one wants the average of the heart rates of a user over some period, all relevant certificates need to be linked. This can however be settled easily by reserving one attribute item for some linkable serial number.

Certificate schemes may differ in the type of certificates. Firstly, certificates can either be *single- or multiple-use*. Single-use certificates are interesting if they testify to blood pressure values, as these values are changing continuously. For multiple-use certificate, one can moreover want them to be valid only a restricted number of times or in a restricted time period (for instance, passports have a restricted validity period). Also, certificates can either be *static or dynamic*. A static certificate can be a name, date of birth, nationality, whereas a dynamic certificate testifies to some value which is subject to continuous change (blood pressure values, cardiac cycle, etc.). These dynamic certificates are in fact just static certificates updated once a while. We refer to Section 2.8 for a discussion on different schemes.

3.2.2 Multiparty computation scheme

All certificate schemes discussed in Section 2.8 are based on either a discrete log or an RSA related assumption: in the discrete log based scheme of Brands [Bra99] the attributes are fixed in a DLREP representation (see Assumption 2.2.4), and Camenisch and Lysyanskaya [CL02, CL04] use Pedersen commitments in the verification protocol [BCL06]. Therefore, multiparty computation based on threshold homomorphic encryption suits better to certificate schemes than multiparty computation based on verifiable secret sharing, and thus the following components are required:

- (Cryptosystem.) A homomorphic cryptosystem (E, D) is required. We also require a (t, n)-threshold decryption protocol. In the absence of a trusted dealer, also a distributed key generation protocol is required. Both ElGamal and Paillier cryptosystems satisfy these requirements, see Section 2.5;

- (Secure multiparty protocols.) Depending on the scenario, we have a certain function to be computed. This function might be constructed from several building blocks, commonly called 'gates'.

Both the ElGamal and Paillier cryptosystem satisfy the requirements, and for both several multiparty computation gates are known, as shown in Section 2.9. Our focus is however on multiparty computation of statistics, and we will study protocols for these functions in this thesis. Following the recent developments in the literature [CDN01, ST04, ST06, GSV07], security is proven in the cryptographic model against static active adversaries. For discussion the reader is referred to Section 2.3.

3.2.3 Interface

In Figure 3.1 we saw that we need protocols for the interface between the certificate and the multiparty computation scheme: we need a protocol for users to disclose attributes in encrypted form, and service providers need to be able to issue certificates on encrypted values. These components are described more concretely below. As already stated, the complexity of these components highly depends on the chosen certificate and multiparty computation scheme, and therefore these schemes should be chosen so as to optimize certain parameters like communication and round complexity. Also, one might want to take the schemes so as to minimize the assumptions. In Sections 8.1 and 8.3, for several different choices the efficiency and the assumptions are compared.

A: (Encrypted disclosure of attributes.) User $\mathcal{U}$ owns a certificate on some private attributes, and discloses one or more attributes to the verifier *in encrypted form*;

B: (Attribute hiding issuance.) If the output of the secure multiparty computation is an encryption which needs to be accompanied with a certificate, the multiparty computation participants $P = \mathcal{SP}$ need to be able to issue a certificate on that encryption, without learning the encrypted value. Still, $\mathcal{U}$ will learn the encrypted value.

Protocol **B** only applies to the case where $\mathcal{U}$ learns the encrypted value. In this case, the standard verification protocol still works and we only need to adjust the issuing protocol of the original certificate scheme. We can also consider the case where the user does not learn the encrypted value. In this case we in fact need a completely new scheme, since $\mathcal{U}$ does not know the attributes which he should know for the verification.

C: (Encrypted certificate scheme.) The multiparty computation participants $P = \mathcal{SP}$ output one or more encryptions, which need to be accompanied with a certificate, but now also the user $\mathcal{U}$ does not learn the encrypted value.

In Section 3.4 it is shown that these three components really fulfill the requirements.

3.3 Applications of the components A-C

To motivate the interface components **A**-**C**, we show how these protocols may solve the problems described in the examples given in Chapter 1. Additionally, as scheme **C** is the most important result of this thesis, we will emphasize its relevance by illustrating some other practical applications of this scheme.

Secure and anonymous surveys

Recall Example 1.0.1, in which users send private data in encrypted form to the computation servers, obtaining a decrypted result. This setting satisfies precisely the setting of a standard certificate scheme (with the website being a verifier $\mathcal{V}$ and user $\mathcal{U}$ having a certificate issued by an external party), with the small adjustment that encrypted disclosure, protocol **A**, is required. For the certificate scheme we prefer the certificates to be unlinkable, although it is not a requirement. For the interface protocols, we do not need protocols for converting an encryption into a certificate as in this example the computation servers just publish the results. However in Example 1.0.2, extending Example 1.0.1, the output of the secure multiparty computation is required to be encrypted and certified. In particular, cases of medical surveys in which the user may not learn the attributes are also possible, as is explained below. Therefore, for Example 1.0.2 also protocol **B** and scheme **C** are desired.

Monitoring

Recall that in Examples 1.0.3 and 1.0.4 users are provided with a measurement device, which measures and certifies data about the users. The users subsequently send the certificates to the servers. As explained in the problem description, preferably the certificates can be linked. In particular, possibility to revoke anonymity of a user in case the certificates show deviances (like high blood pressure) would be desirable. For the properties of the required certificate scheme, as in the scenario participants utilize a measurement device that measures for instance blood pressure and heartbeat rate, we need dynamic certificates, for which single-use certificates can be used. For the interface protocols, certification on encryptions (components **B** and **C**) is not required, but we need protocol **A** for disclosure of encrypted attributes.

Secure and anonymous social networks

Recall Example 1.0.5, where we are looking for a completely secure and anonymous social network, meaning that a priori no-one can be trusted, not even the website. In case two members have the same or similar interests, the website issues the two matched members a certificate, being a key to a certain chat room where these two participants can meet in private. It means that we really need the computation servers to be able to issue certificates, but the possibility to certify encryptions is not needed, hence for the interface only protocol **A** is required. We prefer unlinkable certificates, since for instance otherwise multiple chat room sessions might be linkable. Some certificate can be multiple-use, like the name, date of birth, etc., but some need to be single-use, like the keys to the chat rooms. Finally, for the multiparty computation protocols, we really need the distributed key generation protocol as we do not allow any trusted party.

Other practical applications of scheme C

We can consider two types of applications of scheme **C**. In particular, scheme **C** applies to scenarios where the user *is not allowed* to learn the attributes, as well as to cases where the user *does not want* to learn the attributes. For the former possibility, one can for instance consider a scenario where a company needs to possess certified medical data of its employees. This can be achieved by letting doctors certify medical data of employees to the company (who acts as $\mathcal{U}$ in scheme **C**). For privacy reasons, however, the company should not be allowed to learn the users' data. Using scheme **C**, where the company is issued *encrypted* certificates, it is possible to solve this problem. One can also think of letters of recommendations as an application where the owner of the certificate (which is the person receiving the letter of recommendations) is not allowed to learn attributes.
Another relevant application of scheme **C** corresponds to the case where the user does not want to learn the attributes. Such application can be found in the medical domain. Indeed, some people do not want to learn about illnesses or (genetic) diseases they suffer, and by certifying these data in encrypted form, this problem can be solved. One can also think of two parents who do not want to learn the gender of their unborn baby, but whose friends already want to buy suitable presents for their baby.

3.4 Architectural setting

To verify that the protocols introduced in Section 3.2 indeed suffice to construct the complete scheme, consider one service provider, one user $\mathcal{U}$ and one verifier $\mathcal{V}$. Without loss of generality the service provider is a set of multiparty computation participants (the only difference between this set P and a normal $\mathcal{SP}$ is that P can moreover do multiparty computation). It can be the case that $P = \mathcal{V}$ as well. A discussion on considering $\mathcal{SP}$ and $\mathcal{V}$ as a set of multiparty computation participants is included in Section 8.1.1. All possible 'tracks' of a certificate issued by P are given in Figure 3.2: the result of the secure function evaluation by P is an encryption, which can be made known to $\mathcal{U}$ either in plaintext or encrypted form, and it can either be certified or not. This indeed results in the four tracks in Figure 3.2. Moreover, if $\mathcal{U}$ owns a certificate, he can show it to $\mathcal{V}$. He can do so by showing the certificates with plaintext[3] or encrypted disclosure, but clearly in case $\mathcal{U}$ is issued a certificate on encryptions, plaintext disclosure is impossible. In Figure 3.2 the required protocols and schemes are given between parentheses. Note that these protocols indeed satisfy the requirements.

- (encryption) $\rightarrow$ (plaintext): user $\mathcal{U}$ is the only participant allowed to learn the encrypted value. This can be done by using the private output protocol of Section 2.9.1;
- (encryption) $\rightarrow$ (plaintext + certificate): now, the user $\mathcal{U}$ obtains a certificate on the encrypted value, but $\mathcal{U}$ still learns the plaintext. This is exactly protocol **B** of Section 3.2.3.

[3]Implicitly, we mean 'with plaintext disclosure or with no disclosure'. Both options are supported by the underlying certificate scheme.

Now, plaintext disclosure is covered by the basic certificate scheme, and encrypted disclosure is possible due to protocol **A**;

- (encryption) $\rightarrow$ (encryption + certificate) $\rightarrow$ (encrypted disclosure): in this case, even the user does not learn the plaintext. The required protocols are exactly specified by scheme **C**.

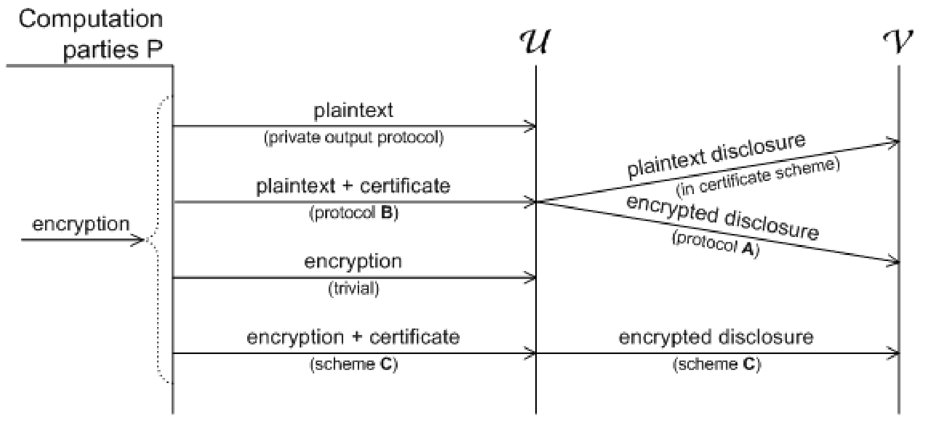

Figure 3.2: All possible transitions of an encryption in the extended certificate scheme.

This means that constructing the components **A**-**C** solves the problem for the interface between the certificate scheme and the multiparty computation scheme. As the main purpose of the scheme is doing multiparty computation of statistics on certified values, we also require the set of known multiparty gates to be extended with statistics gates. The approach of Section 3.2 is universal, in the sense that it applies to any certificate scheme and any multiparty computation scheme, as long as they satisfy the requirements of Sections 3.2.1 and 3.2.2, respectively. In the following chapters, the components **A**-**C** are considered for one certificate scheme in the literature, and for both the ElGamal and Paillier cryptosystem. We will use a certificate scheme of Brands [Bra99], for reasons explained below. Firstly, Chapter 4 gives several multiparty computation protocols, either required for the construction of the three components or for enabling multiparty computation of statistics. Then, in Chapter 5, Brands' certificate scheme is discussed, in Chapter 6 the components **A**-**C** are constructed for the ElGamal cryptosystem, and in Chapter 7 protocols **A** and **B** are considered for the Paillier cryptosystem. We also considered scheme **C** for the Paillier cryptosystem, but did not find a provably secure scheme. We will now introduce the basic cryptographic setting. Unless explicitly stated, we will assume this setting throughout the remainder of the thesis.

Certificate scheme

For the certificate scheme, we use Brands' discrete log scheme [Bra99], which is based on the DLREP assumption (Assumption 2.2.4). We think that this scheme is the best option of the ones discussed in Section 2.8: in particular, unlike Brands' scheme, the schemes introduced by Camenisch-Lysyanskaya [CL02, CL04] use multi-show certificates and it looks like therefore a construction for scheme **C** becomes more complex for these certificate schemes[4]. Still, the construction of the components **A**-**C** for the schemes of Camenisch-Lysyanskaya are briefly discussed in Section 8.3. Brands' scheme is provided with the algorithms and protocols needed according to Section

[4]It looks like a construction for scheme **C** would be more complex in case of multi-use certificates. As argued more extensively in Section 8.3: in the two schemes of Camenisch-Lysyanskaya [CL02, CL04] the user is issued a certificate on a list of attributes. In the verification protocol, the user has to re-blind the certificate and to prove that the certificate is correct. For scheme **C** this means that user is issued a certificate on a list of encryptions of attributes, and that in the verification for each certified encryption c the users takes a random r, sets $c' = c[\![0, r]\!]$ and sends c' to the verifier. However, as the verifier does not know c (otherwise the issuing and verification execution would be linkable), the user cannot simply prove knowledge of r. Therefore, if the user would set $c' = c[\![x]\!]$ for an unknown x, the verification protocol would succeed if $x = 0$, hence would succeed with probability $Pr(x = 0)$.

3.2.1, indeed the scheme includes a way to disclose attributes in plaintext. This scheme will be treated briefly in Section 5.2. For the general setting we have a finite cyclic group G generated by g of prime order $q > 2^k$ for some security parameter k (Section 2.1.1). These system parameters will be included explicitly in the formal key generation algorithms of the schemes, Sections 5.2.1 and 6.3.1. For simplicity, for the construction of the schemes we will consider only *one* service provider. In line with Brands, this service provider will be called the 'certification authority', or $\mathcal{CA}$. Also, we only consider one user $\mathcal{U}$ and one verifier $\mathcal{V}$. The certification authority issues a certificate on l attributes.

Multiparty computation scheme

For the encryption schemes, we have n participants $P = \{\mathcal{P}_1, \dots, \mathcal{P}_n\}$ doing multiparty computation. Both ElGamal and Paillier suffice for the protocols needed, and we consider both options. That is, for both a system parameter setup will be done, and the participants (t, n)-threshold share the secret key for both. For simplicity, it will be assumed that the group used for the ElGamal cryptosystem is the same G as is used for the certificate scheme, see Chapter 6. Sometimes it will be the case that the participants P jointly act as a verifier or certification authority. In this case, we just see them as one player. In particular, if in that case the certification authority has a secret value y, we implicitly mean that the participants share this value using a (t, n)-threshold secret sharing scheme. Also if the certification authority needs to generate a random value, we assume that the random gate of Table 2.1 is utilized. This justifies this generalization.

4. Protocols for secure multiparty computation

This chapter elaborates on and extends the set of multiparty gates we have at our disposal, Table 2.1. The discussed gates are interesting for several reasons. For instance, Sections 4.1 and 4.2 discuss Paillier to ElGamal conversion and an extended plaintext equality test respectively, and both turn out to be of importance in the certificate schemes of Chapters 5-7. In addition, in this chapter it is also studied in what way multiparty computation of statistics is possible, i.e. multiparty computation of statistics functions. A well-known statistic is the 'mean', computing the average of some values. Although this function looks trivial, it is not as it requires division *over the reals*. Indeed, if one computes the mean of two values x and y, their sum has to be divided by 2, and division is not defined on $\mathbb{Z}_{m^s}$ (if we use the Paillier cryptosystem) or $\mathbb{Z}_q$ (if we use the ElGamal cryptosystem), and moreover 2 need not be a divisor of $x + y$. This implies the need of an 'integer division' gate, or equivalently: a 'modulo reduction' gate. An efficient gate[1] for this purpose is discussed in Section 4.3. This modulo reduction gate and a sorting gate (Section 4.4) make several protocols for statistics possible, as shown in Section 4.5.

We consider the following setting for all protocols. We have n participants $P = \{\mathcal{P}_1, \ldots, \mathcal{P}_n\}$. For ElGamal we consider a group $G = \langle g \rangle$ of order $q > 2^k$ for security parameter k (see also Section 2.5.1). The participants threshold share the secret decryption key λ such that $f = g^\lambda$, and each participant holds a polynomial share λ_i. For Paillier, we consider a modulus $m = pq$ of length k, with p, q safe primes, and an $s \in \mathbb{N}$, and consider group $\mathbb{Z}^*_{m^{s+1}}$ with message space $\mathbb{Z}_{m^s}$ (see also Section 2.5.2). For the Paillier cryptosystem the participants share the secret decryption key μ, such that t honest participants can decrypt. Each participant has a polynomial share μ_i.

4.1 Paillier to ElGamal conversion

Suppose we are given an ElGamal encryption $[\![x]\!]_G$ with $x \in \mathbb{Z}_q$, and we want to compute a Paillier encryption $[\![x]\!]_P$ for the same x (provided $q < m^s$). A multiparty gate for this operation is infeasible. Indeed, since then using this gate one can break the DL Assumption 2.2.1 *via* Paillier. It turns out that the other way around, from Paillier to ElGamal, is possible though, if we assume $q < m^s$ (which is reasonable by Section 2.2.3). On input of $[\![x]\!]_P$, we obtain $[\![x]\!]_G$ such that $[\![x]\!]_P \doteq [\![x]\!]_G$ according to the definition given in Section 2.5.4. The main building block is integer commitments. The gate constructed in this section focuses on $x \in \{0, \ldots, q-1\}$, which implies that $D_P([\![x]\!]_P) = D_G([\![x]\!]_G)$ *over the integers*. If x was from a larger domain (with $x < m^s$), the gate is still possible, resulting in encryptions encrypting the same value mod q.

Although this section is about conversion for a general $x \in \mathbb{Z}_q$, we briefly consider the possibilities if x was from a two-valued domain. Then ElGamal to Paillier conversion *is* possible. Informally, the gate would be similar to the conditional gate [ST04]:
Say $x \in \{-1, 1\}$. The conditional gate is executed with $x_0 := x$ and $y_0 := 1$, but with the

[1] Only in case the modulus is publicly known, which is the case in our situation.

important difference that now y_0 is *Paillier* encrypted. The proofs of knowledge are therefore done using integer commitments (see Section 2.6.1), with $\ell_x = 1$. In phase 2, x_n is decrypted, and the protocol output is $[\![x]\!]_P \leftarrow [\![y]\!]_P^{x_n}$. This consequently means that if we have the bit representation $[\![x]\!]_G^{b(\ell)}$, we can get $[\![x]\!]_P^{b(\ell)}$, but this gate requires ℓ such conversion gates, meaning broadcast complexity $O(nk\ell)$ in $O(n)$ rounds (observe that the ℓ gates can be executed in parallel).

Back to the original problem, firstly an informal approach is discussed, to clarify the idea. Then the formal protocol is given. Finally in Section 4.1.1 the efficiency of the protocol is considered.

Intuition

The main idea of the protocol is to reconsider the proof of knowledge with integer commitment of Section 2.6.1, now with groups $G_1 = G$ and $G_2 = \mathbb{Z}^*_{m^{s+1}}$. The idea of that scheme is that a prover commits to an $x < 2^{\ell_x+1}$, and if $2^{\ell_x+\ell_c+\ell_s+1} < \min\{q, m^s\}$, he proves that the two commitments over different groups commit to the same value *over the integers*. But if we relax the requirement '$< q$', the committed value over the group G might have been computed modulo q (as the order of the group is q and x might be $\geq q$), but the commitment in group $\mathbb{Z}^*_{m^{s+1}}$ remains unchanged. As a consequence, both commitments commit to the same value mod q (formally, for the Σ-protocol of Section 2.6.1: $\log_{g_2} y_2 \bmod q = \log_{g_1} y_1$). We stress that this minor change in the protocol does not harm the security.

Now, given an encryption $[\![x]\!]_P$ for $x \in \mathbb{Z}_q$, the idea is that the n participants all publish $[\![r_i]\!]_P, [\![r_i]\!]_G$ (for some random r_i) plus the proof of knowledge of Figure 2.2. Then the participants individually compute $[\![x']\!]_P$, where

$$x' = x + \sum_{i=1}^{n} r_i.$$

We need the r_i's to statistically hide x, so at least we need $r_i \gg q$ for all i, which implies that we need the above described adaption of the proof of knowledge of Figure 2.2. Then the participants decrypt $[\![x']\!]_P$, obtaining $x' \bmod m^s$ (since we work in Paillier encryptions). Encrypting this value using ElGamal would result in $c' = [\![(x' \bmod m^s) \bmod q]\!]_G$, from which $[\![x]\!]_G$ is difficult to obtain. However, if the r_i are taken such that $x' < m^s$, this would mean that $c' = [\![x' \bmod q]\!]_G$. As in fact the encrypted values in $[\![r_i]\!]_G$ are also modulo q, we obtain

$$c' / \prod_{i=1}^{n} [\![r_i]\!]_G = \left[\!\!\left[\left(x' \bmod q - \sum_{i=1}^{n} r_i \bmod q \right) \bmod q \right]\!\!\right]_G = \left[\!\!\left[\left(x' - \sum_{i=1}^{n} r_i \right) \bmod q \right]\!\!\right]_G ,$$

which equals $[\![x \bmod q]\!]_G = [\![x]\!]_G$, as $x' = x + \sum_{i=1}^{n} r_i$ and $x \in \mathbb{Z}_q$.

Formal protocol

We recall that we have n participants (t, n)-threshold sharing the secret key μ for Paillier decryption, and that the public key for the cryptosystem is (m, s), with message space $\mathbb{Z}_{m^s}$. Notice that the participants do not need knowledge of the secret key for the ElGamal cryptosystem. We assume the setting for integer commitments in Definition 2.4.2, and we assume that therefore without loss of generality the Paillier modulus m is used. We recall that this setting includes the introduction of security parameters ℓ_c and ℓ_s. For the protocol, we introduce two additional security parameters: ℓ_r as the length of the r_i's and ℓ_{s2}, and we require that $2^{\ell_r} > q2^{\ell_{s2}}$ and $n2^{\ell_r+\ell_c+\ell_s+1} < m^s$.

Protocol 4.1.1 (Paillier to ElGamal (PtE))**.** Given an encryption $[\![x]\!]_P$ with $x \in \mathbb{Z}_q$, the following protocol outputs an encryption $[\![x]\!]_G$. The participants $\mathcal{P}_i$ $(i = 1, \ldots, n)$ perform the following steps:

1. Each participant individually generates $r_i \in_R \{0,1\}^{\ell_r}$ and publishes $c_{0i} = [\![r_i]\!]_P$ and $c_i = (c_{1i}, c_{2i}) = [\![r_i]\!]_G$, together with a Σ-protocol using integer commitment for relation:

$$\begin{aligned}\{(c_{0i}, c_{1i}, c_{2i}; r_i, s_1, s_2) \mid c_{0i} &\overset{\mathbb{Z}^*_{m^{s+1}}}{=} (m+1)^{r_i} s_1^{m^s} \wedge c_{1i} \overset{G}{=} g^{s_2} \wedge \\ c_{2i} &\overset{G}{=} g^{r_i} f^{s_2} \wedge (-2^{\ell_r+\ell_c+\ell_s} < r_i < 2^{\ell_r+\ell_c+\ell_s})\}.\end{aligned} \tag{4.1}$$

   The protocol is given in Figure 4.1. Failing participants are discarded immediately;

2. Each participant individually computes $[\![x']\!]_P \leftarrow [\![x]\!]_P \prod_{i=1}^{n} c_{0i}$;

3. Using threshold decryption, the participants obtain x';

4. Each participant individually sets $c' = (1, g^{x'})$ and computes the ElGamal encryption of x:

$$[\![x]\!]_G \leftarrow c' / \prod_{i=1}^{n} c_i. \tag{4.2}$$

$$\begin{array}{rcl}
\textbf{Prover} & & \textbf{Verifier} \\
(\text{knows: } c_0, c_1, c_2; r, s_1, s_2) & & (\text{knows: } c_0, c_1, c_2) \\
\tilde{r} \in_R \{0, \ldots, \lfloor m/4 \rfloor - 1\}, \quad y \leftarrow h_1^r h_2^{\tilde{r}} & & \\
u_r \in_R \{0,1\}^{\ell_r+\ell_c+\ell_s}, \quad u_1 \in_R \mathbb{Z}^*_{m^{s+1}} & & \\
u_{\tilde{r}} \in_R \{0,1\}^{k/4+\ell_c+\ell_s}, \quad u_2 \in_R \mathbb{Z}_q & & \\
a_0 \overset{\mathbb{Z}^*_{m^{s+1}}}{\leftarrow} (m+1)^{u_r} u_1^{m^s}, \quad a_1 \overset{G}{\leftarrow} g^{u_2} & & \\
a_2 \overset{G}{\leftarrow} g^{u_r} f^{u_2}, \quad a_y \overset{\mathbb{Z}^*_m}{\leftarrow} h_1^{u_r} h_2^{u_{\tilde{r}}} & \xrightarrow{y, a_0, a_1, a_2, a_y} & \\
& \xleftarrow{\quad c \quad} & c \in_R \{0,1\}^{\ell_c} \\
r_r \leftarrow u_r + cr, \quad r_{\tilde{r}} \leftarrow u_{\tilde{r}} + c\tilde{r} & & \\
r_1 \overset{\mathbb{Z}^*_{m^{s+1}}}{\leftarrow} u_1 s_1^c, \quad r_2 \leftarrow u_2 + cs_2 \bmod q & \xrightarrow{r_r, r_{\tilde{r}}, r_1, r_2} & \\
& & (m+1)^{r_r} r_1^{m^s} \overset{?}{=} a_0 c_0^c, \quad g^{r_2} \overset{?}{=} a_1 c_1^c \\
& & g^{r_r} f^{r_2} \overset{?}{=} a_2 c_2^c, \quad h_1^{r_r} h_2^{r_{\tilde{r}}} \overset{?}{=} a_y y^c \\
& & r_r \overset{?}{\in} \{0,1\}^{\ell_r+\ell_c+\ell_s}
\end{array}$$

Figure 4.1: Σ-protocol for relation (4.1).

Lemma 4.1.2. *The protocol in Figure 4.1 is a Σ-protocol for (4.1).*

Proof. (Completeness & honest-verifier zero-knowledge). Similar to the proof of Proposition 2.6.2, completeness is trivial and for honest-verifier zero-knowledge we only need to add $r_1 \in_R \mathbb{Z}^*_{m^{s+1}}, r_2 \in_R \mathbb{Z}_q$ and $a_0 \overset{\mathbb{Z}^*_{m^{s+1}}}{\leftarrow} (m+1)^{r_r} r_1^{m^s} c_0^{-c}$ and $a_1 \overset{G}{\leftarrow} g^{r_2} c_1^{-c}$ to the simulated conversation. (Special soundness). The proof of special soundness of Proposition 2.6.2 does not suffice anymore for finding a witness s_1. Consider two conversations $(y, a_0, a_1, a_2, a_y; c; r_r, r_{\tilde{r}}, r_1, r_2)$ and $(y, a_1, a_2, a_y; c'; r'_r, r'_{\tilde{r}}, r'_1, r'_2)$ with $c \neq c'$. Similarly to the compared proposition, we still take

$$r := \frac{r_r - r'_r}{c - c'}, \qquad \tilde{r} := \frac{r_{\tilde{r}} - r'_{\tilde{r}}}{c - c'}, \qquad s_2 := \frac{r_2 - r'_2}{c - c'}.$$

For s_1 we have the conversations satisfying:

$$(m+1)^{r_r} r_1^{m^s} = a_0 c_0^c, \qquad (m+1)^{r'_r} (r'_1)^{m^s} = a_0 c_0^{c'},$$

hence $(m+1)^{r_r - r'_r}(r_1/r'_1)^{m^s} = c_0^{c-c'}$. If $\gcd(c-c', m^s) \neq 1$, with overwhelming probability we obtained a non-trivial factor of m [Jur03, Sec. 4.3.4], which is hard by assumption. Therefore, using extended Euclid's algorithm we can find ρ, σ such that $\rho(c-c') + \sigma m^s = 1$. Now, exponentiating the mentioned equation to ρ gives:

$$(m+1)^{\rho(r_r - r'_r)}(r_1/r'_1)^{\rho m^s} = c_0^{\rho(c-c')} = c_0^{1-\sigma m^s},$$

which results in $c_0 = (m+1)^{\rho(r_r - r'_r)}((r_1/r'_1)^{\rho} c_0^{\sigma})^{m^s}$, but this does not satisfy the witness $r := \frac{r_r - r'_r}{c-c'}$. But similar to Proposition 2.6.2 we have $c - c' \mid r_r - r'_r$, and therefore:

$$\rho(r_r - r'_r) = \frac{r_r - r'_r}{c-c'}\rho(c-c') = \frac{r_r - r'_r}{c-c'}(1 - \sigma m^s),$$

which results in:

$$c_0 = (m+1)^{\frac{r_r - r'_r}{c-c'}} \left((r_1/r'_1)^{\rho} c_0^{\sigma} (m+1)^{-\frac{r_r - r'_r}{c-c'}\sigma} \right)^{m^s},$$

implying witness $s_1 := (r_1/r'_1)^{\rho} c_0^{\sigma}(m+1)^{-\frac{r_r - r'_r}{c-c'}\sigma}$. □

We are now ready to prove correctness of the protocol.

Proposition 4.1.3 (Correctness)**.** *On input of an encryption $[\![x]\!]_P$ with $x \in \mathbb{Z}_q$, the output in (4.2) equals $[\![x]\!]_G$.*

Proof. We need to prove that $c' / \prod_{i=1}^{n} c_i$ is an ElGamal encryption of x. This is proven using two equations.
Firstly, the value x' as computed in phase 3 satisfies $x' = x + \sum_{i=1}^{n} r_i \bmod m^s$, as we work over Paillier encryptions. But by phase 1, for each $i = 1, \ldots, n$ we have $r_i \in (-2^{\ell_r + \ell_c + \ell_s}, 2^{\ell_r + \ell_c + \ell_s})$. Therefore, $|x'| = |x + \sum_{i=1}^{n} r_i| < q + n2^{\ell_r + \ell_c + \ell_s + 1}$, which is $< m^s$ with overwhelming probability. So with overwhelming probability the equality holds over the integers:

$$x' = x + \sum_{i=1}^{n} r_i. \tag{4.3}$$

Secondly, as a consequence of Lemma 4.1.2, for each i the encryption c_i satisfies:

$$c_i = [\![r_i \bmod q]\!]_G. \tag{4.4}$$

Now we obtain for phase 4 of Protocol 4.1.1:

$$\begin{aligned} c' / \prod_{i=1}^{n} c_i &= \left[\!\!\left[\left(x' \bmod q - \sum_{i=1}^{n} (r_i \bmod q) \right) \bmod q \right]\!\!\right]_G && \{\text{phase 4 and (4.4)}\} \\ &= \left[\!\!\left[\left(x + \sum_{i=1}^{n} r_i - \sum_{i=1}^{n} r_i \right) \bmod q \right]\!\!\right]_G && \{\text{equation (4.3)}\} \\ &= [\![x \bmod q]\!]_G = [\![x]\!]_G && \{\text{as } x \in \mathbb{Z}_q\}. \end{aligned}$$ □

Security of the protocol is proven in light of the real/ideal-model, Section 2.3.

Proposition 4.1.4 (Security)**.** *On input of encryptions $[\![x]\!]_P, [\![x]\!]_G$ with $x \in \mathbb{Z}_q$, Protocol 4.1.1 can be simulated in a statistically indistinguishable way.*

Proof. We present a simulator that simulates the protocol in a statistically indistinguishable way. The simulator takes as input $[\![x]\!]_P, [\![x]\!]_G$ with $x \in \mathbb{Z}_q$, and may use the malicious parties $P' = \{\mathcal{P}_1, \ldots, \mathcal{P}_{t-1}\} \subset P$ as a subroutine.
Phase 1. The simulator lets the adversary run this phase for $\mathcal{P}_1, \ldots, \mathcal{P}_{t-1}$, obtaining $[\![r_i]\!]_P$, $[\![r_i$ mod

$q]\!]_G$ for r_i of length less than $2^{\ell_r+\ell_c+\ell_s+1} < m^s$, accompanied with a proof of knowledge. The simulator rewinds the proofs of knowledge, obtaining values $r_1, \ldots, r_{t-1}$ (by the special soundness property). For $\mathcal{P}_t, \ldots, \mathcal{P}_{n-1}$ he executes phase 1 as in the real protocol, obtaining values $r_t, \ldots, r_{n-1}$. For $\mathcal{P}_n$ however, he takes $r_n \in_R \{0,1\}^{\ell_r}$ and publishes $[\![\tilde{r}_n]\!]_P, [\![\tilde{r}_n \bmod q]\!]_G$ where $\tilde{r}_n = r_n - x$. Notice that he indeed can compute these encryptions. The required proof of knowledge is simulated.
Phase 2. Executed as is. By construction, $x' = x + \sum_{i=1}^{n-1} r_i + \tilde{r}_n = \sum_{i=1}^{n} r_i$.
Phase 3. Simulated on input $[\![x']\!]_P$ and $\sum_{i=1}^{n} r_i$ (see Section 2.5.3).
Phase 4. Executed as is.
Well now, the values $r_1, \ldots, r_{n-1}$ are from the same set as in the real protocol. Only difference is in $\tilde{r}_n$ versus r_n. In the real protocol it is from $R = [0, 2^{\ell_r})$, and in the simulated it is from $S = (-q, 2^{\ell_r})$. These sets are statistically indistinguishable in ℓ_{s2} since $2^{\ell_r} > q2^{\ell_{s2}}$ (similar to Proposition 2.6.2):

$$\frac{1}{2} \sum_{v \in R \cup S} \Big| P(v \in R) - P(v \in S) \Big| = \frac{q}{q + 2^{\ell_r}} = \frac{1}{1 + 2^{\ell_r}/q} < \frac{1}{2^{\ell_{s2}}}. \qquad \square$$

So concluding, the protocol is statistically secure under the DCR Assumption 2.2.7 (for Paillier encryption), the DDH Assumption 2.2.3 (for ElGamal encryption) and the SRSA Assumption 2.2.6 (for the integer commitment).

4.1.1 Efficiency

Phases 1, 2 & 4 clearly have round complexity $O(1)$ and broadcast complexity $O(nk)$ (caused by the proofs of knowledge). As phase 3 is only a threshold decryption, it has the same complexities (see Table 2.1). Thus the complete protocol has round complexity $O(1)$ and broadcast complexity $O(nk)$.

4.2 Plaintext equality tests

The idea of plaintext equality tests is by Jakobsson and Juels [JJ00]. For the purpose of our scheme, we however need extensions to this test. For convenience, we start completely from scratch and discuss the 'standard' plaintext equality test too. We will consider the following three functions. We recall that we have n participants (t, n)-threshold sharing ElGamal decryption key λ and Paillier decryption key μ.

- Standard plaintext equality test: on input of two ElGamal encryptions[2], the participants want to verify whether they encrypt the same value:

$$\mathrm{pet}_{P(\lambda)}([\![a]\!]_G, [\![b]\!]_G) = \beta = \begin{cases} 1, \text{ if } a = b; \\ 0, \text{ otherwise.} \end{cases} \tag{4.5}$$

  This gate is discussed in Section 4.2.1;

- Plaintext equality test with ElGamal representation: this function considers three ElGamal encryptions[2]. The participants P also (t, n)-threshold share two secret values $y, z \in_R \mathbb{Z}_q$, with $Y = g^y, Z = g^z$ known. To this end, the same distributed key generation protocol as for sharing λ can be used (Section 2.5.1). The function to be computed is:

$$\mathrm{pet}^{\mathrm{rep}}_{P(\lambda,y,z)}([\![a]\!]_G, [\![b]\!]_G, [\![c]\!]_G) = \beta = \begin{cases} 1, \text{ if } a \equiv by + cz \bmod q; \\ 0, \text{ otherwise.} \end{cases} \tag{4.6}$$

  The gate is discussed in Section 4.2.2. This function is called $\mathrm{pet}^{\mathrm{rep}}$ because its structure is similar to the one in the DLREP assumption (Assumption 2.2.4), where a tuple of secret values $(x_1, \ldots, x_l)$ is required to satisfy $1 = g_1^{x_1} \cdots g_l^{x_l}$;

[2]If all encryptions are Paillier, a similar protocol can be constructed.

- Plaintext equality test with Paillier representation: additionally to $\mathrm{pet}^{\mathrm{rep}}$, now b and c are Paillier encrypted instead (we require that $b, c < m^s$, but they may be $\geq q$). The function to be evaluated is:

$$\mathrm{pet}^{\mathrm{repP}}_{P(\lambda,y,z;\mu)}([\![a]\!]_G, [\![b]\!]_P, [\![c]\!]_P) = \beta = \begin{cases} 1, \text{ if } a \equiv by + cz \bmod q; \\ 0, \text{ otherwise.} \end{cases} \tag{4.7}$$

  The gate is discussed in Section 4.2.3.

We note that the latter two functions and corresponding protocols can be easily generalized to $l+1$ encrypted inputs. In Section 4.2.4 the efficiency of the protocols is considered.

4.2.1 Standard plaintext equality test

This section only involves ElGamal encryptions, therefore the subscript '$_G$' will be omitted.

Protocol 4.2.1 (Plaintext equality test (pet))**.** Given $[\![a]\!], [\![b]\!]$ for $a, b \in \mathbb{Z}_q$, the following protocol evaluates the function (4.5). The participants $\mathcal{P}_i$ $(i = 1, \ldots, n)$ perform the following steps:

1. Each participant individually computes $[\![c]\!] \leftarrow [\![a]\!]/[\![b]\!]$, takes $r_i \in_R \mathbb{Z}_q^*$ and publishes $[\![c_i]\!] \leftarrow [\![c]\!]^{r_i}$ together with a Σ-protocol for relation $\{([\![c_i]\!]; r_i) \mid [\![c_i]\!] = [\![c]\!]^{r_i}\}$. This protocol is trivial. Failing participants are discarded immediately;
2. Each participant individually computes $[\![\tilde{c}]\!] \leftarrow \prod_{i=1}^n [\![c_i]\!]$;
3. Using threshold decryption, the participants obtain $\tilde{c}$;
4. Each participant individually computes the output as

$$\beta \leftarrow \begin{cases} 1, \text{ if } \tilde{c} = 0; \\ 0, \text{ otherwise.} \end{cases} \tag{4.8}$$

Proposition 4.2.2 (Correctness)**.** *On input of encryptions $[\![a]\!], [\![b]\!]$ with $a, b \in \mathbb{Z}_q$, the output in (4.8) equals β as in (4.5).*

Proof. Notice that, where $\overset{*}{\iff}$ holds with overwhelming probability:

$$\beta = 1 \iff \tilde{c} = 0 \iff c\left(\sum_{i=1}^n r_i\right) \equiv 0 \bmod q \overset{*}{\iff} c = 0 \iff a = b. \qquad \square$$

Security of the protocol is proven in light of the real/ideal-model, Section 2.3.

Proposition 4.2.3 (Security)**.** *On input of encryptions $[\![a]\!], [\![b]\!]$ with $a, b \in \mathbb{Z}_q$, and a $\beta \in \{0, 1\}$, Protocol 4.2.1 can be simulated in a perfectly indistinguishable way.*

Proof. We present a simulator that simulates the protocol in a perfectly indistinguishable way, and fails with only negligible probability. The simulator takes as input $[\![a]\!], [\![b]\!]$ and a bit β, and may use the malicious parties $P' = \{\mathcal{P}_1, \ldots, \mathcal{P}_{t-1}\} \subset P$ as a subroutine.
Phase 1. The simulator lets the adversary run this phase for $\mathcal{P}_1, \ldots, \mathcal{P}_{t-1}$, obtaining $[\![c_i]\!]$, for $i = 1, \ldots, t-1$, accompanied with a proof of knowledge. The simulator rewinds the proofs of knowledge, obtaining values $r_1, \ldots, r_{t-1}$ (by the special soundness property). For $\mathcal{P}_t, \ldots, \mathcal{P}_{n-1}$ he executes phase 1 as in the real protocol, obtaining values $r_t, \ldots, r_{n-1}$. For $\mathcal{P}_n$ however, he takes $r_n \in_R \mathbb{Z}_q$, and publishes $[\![c_n]\!] \leftarrow \left[\!\!\left[(1-\beta)\sum_{i=1}^n r_i - c\sum_{i=1}^{n-1} r_i\right]\!\!\right]$. The required proof of knowledge is simulated. Notice that the simulator indeed can compute $[\![c_n]\!]$ as

$$[\![c_n]\!] = \left[\!\!\left[(1-\beta)\sum_{i=1}^n r_i - c\sum_{i=1}^{n-1} r_i\right]\!\!\right] = [\![1]\!]^{(1-\beta)\sum_{i=1}^n r_i}[\![c]\!]^{-\sum_{i=1}^{n-1} r_i}.$$

Phase 2. Executed as is.
Phase 3. The simulator actually knows $\tilde{c}$:

$$\tilde{c} \equiv c \sum_{i=1}^{n-1} r_i + c_n \equiv (1-\beta) \sum_{i=1}^{n} r_i \bmod q,$$

and therefore this phase is simulated on input $[\![\tilde{c}]\!]$ and $(1-\beta)\sum_{i=1}^{n} r_i \bmod q$ (see Section 2.5.3). Now, phase 4 with overwhelming probability succeeds due to the factor $1-\beta$ in $\tilde{c}$. As clearly all values $r_1, \dots, r_{n-1}, \tilde{r}_n$ are taken uniformly at random from $\mathbb{Z}_q$, we have perfectly indistinguishability. □

So concluding, the protocol is perfectly secure under the DDH Assumption 2.2.3 (for ElGamal encryption), and fails with only negligible probability.

4.2.2 Plaintext equality test with ElGamal representation

This section only involves ElGamal encryptions, therefore the subscript '$_G$' will be omitted. We construct a protocol for the function (4.6). The idea is to first compute $[\![\tilde{b}]\!] = \mathrm{rep}_{P(y,z)}([\![b]\!], [\![c]\!]) \leftarrow [\![b]\!]^y [\![c]\!]^z$, which is done in the following sub-protocol.

Protocol 4.2.4 (Sub-protocol for pet with ElGamal representation)**.** Given $[\![b]\!], [\![c]\!]$ for $b, c \in \mathbb{Z}_q$, the following protocol outputs an encryption $[\![\tilde{b}]\!] = [\![b]\!]^y [\![c]\!]^z$. The participants $\mathcal{P}_i$ $(i = 1, \dots, n)$ perform the following steps:

1. Each participant publishes $[\![\tilde{b}_i]\!] = [\![b]\!]^{y_i} [\![c]\!]^{z_i}$ together with Σ-protocol for relation:

   $$\{([\![\tilde{b}_i]\!]; y_i, z_i) \mid Y_i = g^{y_i} \wedge Z_i = g^{z_i} \wedge [\![\tilde{b}_i]\!] = [\![b]\!]^{y_i} [\![c]\!]^{z_i}\}.$$

   This protocol is trivial. Failing participants are discarded immediately;

2. Let A be the set of participants with a successful proof of knowledge. Outcome $[\![\tilde{b}]\!]$ is now computed as

   $$[\![\tilde{b}]\!] \leftarrow \prod_{i \in A} [\![\tilde{b}_i]\!]^{\lambda_{A,i}},$$

   where $\lambda_{A,i} = \prod_{j \in A \setminus \{i\}} \frac{j}{j-i}$ denote the Lagrange coefficients.

Notice that this is a slightly adjusted version of threshold decryption for ElGamal. In particular, the complexities are equal. Correctness of Protocol 4.2.4 follows from the correctness of the threshold decryption protocol [Sch09]. It is also secure as it can be simulated in a perfectly indistinguishable way on input $[\![b]\!], [\![c]\!], [\![\tilde{b}]\!]$. This simulator is a simplification of the simulator for threshold decryption for ElGamal [ST04]. We are now ready to construct a protocol for the $\mathrm{pet}^{\mathrm{rep}}$ function.

Protocol 4.2.5 (Plaintext equality test with ElGamal representation ($\mathrm{pet}^{\mathrm{rep}}$))**.** Given $[\![a]\!], [\![b]\!], [\![c]\!]$ for $a, b, c \in \mathbb{Z}_q$, the following protocol evaluates the function (4.6). The participants $\mathcal{P}_i$ $(i = 1, \dots, n)$ perform the following steps:

1. The participants jointly compute $[\![\tilde{b}]\!] \leftarrow \mathrm{rep}_{P(y,z)}([\![b]\!], [\![c]\!])$ using Protocol 4.2.4;

2. The participants jointly compute and output $\beta \leftarrow \mathrm{pet}_{P(\lambda)}([\![a]\!], [\![\tilde{b}]\!])$ using Protocol 4.2.1.

Proposition 4.2.6 (Correctness)**.** *On input of encryptions $[\![a]\!], [\![b]\!], [\![c]\!]$ with $a, b, c \in \mathbb{Z}_q$, the output of Protocol 4.2.5 equals β as in (4.6).*

Proof. Notice that $\beta = 1 \iff a = \tilde{b}$ by the correctness of Protocol 4.2.1. But in phase 1, $\tilde{b}$ is computed as $\tilde{b} \equiv \sum_{i \in A} \lambda_{A,i}(by_i + cz_i) \equiv by + cz \bmod q$ by the construction of the Lagrange coefficients. So concluding, we obtain $\beta = 1 \iff a \equiv by + cz \bmod q$. □

Proposition 4.2.7 (Security)**.** *On input of encryptions $[\![a]\!], [\![b]\!], [\![c]\!]$ with $a, b, c \in \mathbb{Z}_q$, and a $\beta \in \{0,1\}$, Protocol 4.2.5 can be simulated in a computationally indistinguishable way.*

Proof. We present a simulator that simulates the protocol in a computationally indistinguishable way, and fails with only negligible probability. The simulator takes as input $[\![a]\!], [\![b]\!], [\![c]\!]$ and a bit β, and may use the malicious parties $P' = \{\mathcal{P}_1, \ldots, \mathcal{P}_{t-1}\} \subset P$ as a subroutine. Following [CDN01, Thm. 1] and [ST04, Thm. 1], firstly we consider an oracle outputting $[\![\tilde{b}]\!] \leftarrow \mathrm{O}_{\mathrm{rep}}([\![b]\!], [\![c]\!])$.
Phase 1. Simulated using the simulator for $\mathrm{rep}_{P(y,z)}$, on input of $([\![b]\!], [\![c]\!], [\![\tilde{b}]\!])$.
Phase 2. Simulated using the simulator for $\mathrm{pet}_{P(\lambda)}$, on input of $([\![a]\!], [\![\tilde{b}]\!], \beta)$ (Proposition 4.2.3).
Both of the used simulators may use P' as a subroutine. As the two simulators simulate the parts in a perfectly indistinguishable way, and succeed with overwhelming probability, this simulator simulates Protocol 4.2.5 in a perfectly indistinguishable way, also succeeding with overwhelming probability.
Now, we replace the outcome of the oracle call by a random encryption of zero: $[\![\tilde{b}]\!] \leftarrow [\![0]\!]$. If this significantly changes the view of the conversation, as the simulator does not use any secrets of the honest parties, we obtain a distinguisher for $[\![0]\!]$ and $[\![a^y b^z]\!]$, which is hard as the ElGamal cryptosystem is assumed to be semantically secure. □

So concluding, the protocol is computationally secure under the DDH Assumption 2.2.3 (for ElGamal encryption), and fails with only negligible probability. Note that security is not perfectly: this is since we implicitly use a hybrid argument and the cryptosystem is only semantically secure (see Definition 2.5.1).

4.2.3 Plaintext equality test with Paillier representation

This gate is similar to the one discussed in Section 4.2.2, except that now the PtE gate in Protocol 4.1.1 is deployed to transform Paillier encryptions to ElGamal encryptions. For simplicity we require that $b, c < q$, but this restriction can be relaxed to larger domains, as argued in Section 4.1.

Protocol 4.2.8 (Plaintext equality test with Paillier representation ($\mathrm{pet}^{\mathrm{repP}}$))**.** Given $[\![a]\!]_G, [\![b]\!]_P, [\![c]\!]_P$ for $a, b, c \in \mathbb{Z}_q$, the following protocol evaluates the function (4.7). The participants $\mathcal{P}_i$ $(i = 1, \ldots, n)$ perform the following steps:

1. The participants jointly compute $[\![b]\!]_G \leftarrow \mathrm{PtE}_{P(\mu)}([\![b]\!]_P)$ and $[\![c]\!]_G \leftarrow \mathrm{PtE}_{P(\mu)}([\![c]\!]_P)$ using Protocol 4.1.1;

2. The participants jointly compute $[\![\tilde{b}]\!]_G \leftarrow \mathrm{rep}_{P(y,z)}([\![b]\!]_G, [\![c]\!]_G)$ using Protocol 4.2.4;

3. The participants jointly compute and output $\beta \leftarrow \mathrm{pet}_{P(\lambda)}([\![a]\!]_G, [\![\tilde{b}]\!]_G)$ using Protocol 4.2.1.

The proof of correctness is similar to Proposition 4.2.6, and security follows in the same way as Proposition 4.2.7, but now using oracle $([\![b]\!]_G, [\![c]\!]_G, [\![\tilde{b}]\!]_G) \leftarrow \mathrm{O}_{\mathrm{PtE,rep}}([\![b]\!]_P, [\![c]\!]_P)$.

So concluding, the protocol is computationally secure under the DCR Assumption 2.2.7 (for Paillier encryption), the DDH Assumption 2.2.3 (for ElGamal encryption) and the SRSA Assumption 2.2.6 (for the integer commitment), and fails with only negligible probability.

Remark 4.2.9. If we extend the protocol to $l+1$ encrypted inputs, the broadcast complexity of Protocol 4.2.8 grows with a factor l, due to the first phase. Under a certain constraint on the parameters, this can be avoided by swapping the first and second phase. More precisely, the participants then first execute a Paillier analogue of Protocol 4.2.4 on input $[\![b]\!]_P, [\![c]\!]_P$, obtaining $[\![\tilde{b}]\!]_P$, and then they execute the PtE gate on this encryption to obtain $[\![\tilde{b} \bmod q]\!]_G$. In order to ensure that this optimization works, we need that $\tilde{b} \equiv by + cz \bmod q$ really holds, and more generally that the value $\tilde{b}$ computed in the first phase does not exceed m^s. Indeed, the value $\tilde{b}$ computed in phase 1 satisfies

$$[\![\tilde{b}]\!]_P = \prod_{i \in A} [\![\tilde{b}_i]\!]_P^{\lambda_{A,i}} = \prod_{i \in A} [\![b]\!]_P^{y_i \lambda_{A,i}} [\![c]\!]_P^{z_i \lambda_{A,i}} = \left[\!\!\left[\left(b \sum_{i \in A} y_i \lambda_{A,i} + c \sum_{i \in A} z_i \lambda_{A,i} \right) \bmod m^s \right]\!\!\right]_P ,$$

where $\lambda_{A,i}$ are the Lagrange coefficients and A is the set of participants that successfully executed the first phase of Protocol 4.2.4. If $b \sum_{i \in A} y_i \lambda_{A,i} + c \sum_{i \in A} z_i \lambda_{A,i}$ does not exceed m^s, in phase two this encryption is converted to an ElGamal encryption $[\![\tilde{b} \bmod q]\!]_G$ with:

$$\begin{aligned} \tilde{b} \bmod q &= \left(b \sum_{i \in A} y_i \lambda_{A,i} + c \sum_{i \in A} z_i \lambda_{A,i} \right) \bmod m^s \bmod q \\ &= \left(b \sum_{i \in A} y_i \lambda_{A,i} + c \sum_{i \in A} z_i \lambda_{A,i} \right) \bmod q \\ &= by + cz \bmod q, \end{aligned}$$

where the last step follows by the construction of the shares y_i and z_i. Concluding, this means that the ciphertext obtained after conversion to ElGamal encrypts $by + cz \bmod q$.
To ensure that the value $b \sum_{i \in A} y_i \lambda_{A,i} + c \sum_{i \in A} z_i \lambda_{A,i}$ does not exceed m^s, we need a restriction on the involved values. We consider $b, c < 2^{\ell_x}$ for some parameter ℓ_x. As the shares y_i, z_i, as well as the Lagrange coefficients $\lambda_{A,i}$ are computed modulo q by the construction of the distributed key generation protocol [GJKR99], this restriction is certainly satisfied if $2 \cdot nq^2 2^{\ell_x + \ell_s} < m^s$ (for ℓ_s an additional security parameter required for the conversion from Paillier to ElGamal to function correctly). In case this protocol is extended to $l + 1$ encrypted inputs, we need $l \cdot nq^2 2^{\ell_x + \ell_s} < m^s$.

4.2.4 Efficiency

Firstly we consider the efficiency of the standard pet gate of Section 4.2.1. Phases 1, 2 & 4 clearly have round complexity $O(1)$ and broadcast complexity $O(nk)$ (caused by the proofs of knowledge). As phase 3 is only a threshold decryption (see Section 2.9), it has the same complexities. Thus the complete protocol has round complexity $O(1)$ and broadcast complexity $O(nk)$.
The pet$^{\text{rep}}$ gate is constructed from the pet gate and a sub-protocol having the same complexities as threshold decryption. Therefore, this gate also has round complexity $O(1)$ and broadcast complexity $O(nk)$. Also the pet$^{\text{repP}}$ gate has the same complexities as also the PtE gate has the same complexities. (The reader is referred to Table 2.1 for more information about the underlying gates.)

4.3 Modulo reduction

For some statistics such as the mean or variance, we need an integer division gate. Clearly, this construction is for free given the existence of a modulo reduction gate: for a given x and a we have:

$$x \operatorname{div} a = (x - x \bmod a) a^{-1}.$$

Our target therefore is to compute $[\![x \bmod a]\!]$. For our purposes, generally x is in encrypted form. The value a can either be in encrypted form or in plaintext. If a is in encrypted form, the most efficient algorithm known is a sequential protocol requiring $\lg x$ comparisons[3]. We only focus on public a, as this suffices for our purposes.
We note that if the encrypted bit representation of x is unknown, no efficient solution exists for ElGamal. Indeed, otherwise one can compute x's least significant bit $[\![x_0]\!] = [\![x \bmod 2]\!]$, which is

[3]For the modulus a in encrypted form, Algesheimer et al. [ACS02] constructed an algorithm, which is optimized to constant rounds by Damgård et al. [DFK$^+$06, Sec. 7.4]. Yet the performance of this protocol is generally low, in particular for a large number of participants, [FJ06, Ch. 8] and [Sec08, Sec. 4.6].

hard by the DL assumption [ST06]. If the bit representation is known, a protocol of Damgård et al. [DFK+06] allows to compute $[\![x \bmod a]\!]$. We consider the protocol for the Paillier cryptosystem.

We consider input $[\![x]\!]_P$ with $x \in \{0,1\}^{\ell_x}$ and a public value a such that $2^{\ell_a - 1} < a \leq 2^{\ell_a}$ for some ℓ_a, and construct a protocol for the computation of $[\![x \bmod a]\!]_P$. We only consider Paillier encryptions, and therefore the subscript '$_P$' is omitted. Without loss of generality, we assume that $\ell_a \leq \ell_x$: clearly, if $\ell_a > \ell_x$ then certainly $a > x$, in which case $x \bmod a = x$. The protocol relies on the fact that it is unnecessary to compute the ℓ_x bits of x if the modulus $a \leq 2^{\ell_a}$ is known for some $\ell_a \leq \ell_x$. In particular, if $a = 2^{\ell_a}$, computing $[\![x \bmod a]\!]$ can be easily done by computing the ℓ_a least significant bits of x, as $x \bmod a = \sum_{j=0}^{\ell_a - 1} x_j 2^j$. As in many cases ℓ_a is relatively small compared to ℓ_x (see for instance Section 4.3.1), this reduces the costs. The protocol requires a sub-protocol to bitwise generate a value $r \in_R [0, a)$. We refer to this sub-protocol as the *random bitwise value generation* protocol.

We recall that we have n participants (t, n)-threshold sharing the secret key for Paillier decryption, and that the public key for the cryptosystem is (m, s), with message space $\mathbb{Z}_{m^s}$. We introduce a security parameter ℓ_s, which we require to satisfy $an2^{\ell_x + \ell_s} < m^s$. We now discuss the protocol, and its efficiency is discussed in Section 4.3.1. The reader is referred to Table 2.1 for the underlying protocols.

Firstly, the sub-protocol for random bitwise value generation is considered. Then the modulo reduction protocol is constructed and correctness and security are proven. In Section 4.3.1 the efficiency is analyzed.

Protocol 4.3.1 (Random bitwise value generation)**.** Given a publicly known value a such that $2^{\ell_a - 1} < a \leq 2^{\ell_a}$, the following protocol generates an encrypted bit representation $\{[\![r_0]\!], \ldots, [\![r_{\ell_a - 1}]\!]\}$ of r such that $r \in_R [0, a)$. The participants $\mathcal{P}_i$ $(i = 1, \ldots, n)$ perform the following steps:

1. For $j = 0, \ldots, \ell_a - 1$, the participants jointly generate encryptions $[\![r_j]\!]$ for $r_j \in_R \{0, 1\}$, using ℓ_a randbit gates;
2. Using a comparison gate, $[\![[r < a]]\!]$ is computed and jointly decrypted. If $[r < a] = 0$, the protocol is restarted.

The proofs of correctness and security of this protocol are straightforward. Notice that in case $a \neq 2^{\ell_a}$, the number of restarts of the protocol is $2^{\ell_a}/a < 2$ on average: the success probability is $a/2^{\ell_a}$, and the number of restarts follows the geometric distribution (the participants restart until they have success). Hence, they need $2^{\ell_a}/a < 2$ restarts on average. We are now ready to construct the modulo reduction protocol.

Protocol 4.3.2 (Modulo reduction (mod))**.** Given $[\![x]\!]$ for $x \in \{0,1\}^{\ell_x}$ and a publicly known value a, the following protocol outputs an encryption $[\![x \bmod a]\!]$. The participants $\mathcal{P}_i$ $(i = 1, \ldots, n)$ perform the following steps:

1. The participants jointly generate a random encrypted bit representation $\{[\![r_0]\!], \ldots, [\![r_{\ell_a - 1}]\!]\}$ of $r \in_R [0, a)$, using Protocol 4.3.1. In parallel, each participant takes $s_i \in_R \{0,1\}^{\ell_x + \ell_s}$ and publishes $S_i = [\![s_i]\!]$ together with an interval proof of knowledge for relation (see Section 2.6.2)
$$\{(S_i; s_i) \mid S_i = [\![s_i]\!] \wedge s_i \in [0, 2^{\ell_x + \ell_s})\}. \tag{4.9}$$
Failing participants are discarded immediately;
2. Each participant individually computes
$$[\![\tilde{x}]\!] \leftarrow [\![x]\!][\![r]\!]^{-1} \left(\prod_{i=1}^{n} [\![s_i]\!] \right)^{a}; \tag{4.10}$$

3. Using threshold decryption the participants obtain $\tilde{x}$, and they compute $\bar{x} \leftarrow \tilde{x} \bmod a$;
4. Using a comparison gate the participants compute
$$[\![c]\!] \leftarrow [\![[a-1-\bar{x} < r]]\!]. \tag{4.11}$$
Notice that $a-1-\bar{x}$ is known in plaintext, and for r the participants know the encrypted bit representation. In particular, the comparison gates described in Section 2.9 can be used;
5. Each participant individually computes
$$\operatorname{mod}_{P(\mu)}([\![x]\!], a) \leftarrow [\![\bar{x}]\!][\![r]\!][\![c]\!]^{-a}. \tag{4.12}$$

Proposition 4.3.3 (Correctness)**.** *On input of $[\![x]\!]$ with $x \in \{0,1\}^{\ell_x}$, and a publicly known value a, the output in (4.12) equals $[\![x \bmod a]\!]$.*

Proof. Firstly, we note that $0 \le \tilde{x} < m^s$ holds with overwhelming probability: on the one hand, $-r + a\sum_{i=1}^{n} s_i \ge 0$ holds with overwhelming probability, and on the other hand:
$$|\tilde{x}| \le 2^{\ell_x} + a + a\left|\sum_{i=1}^{n} s_i\right| \le 2^{\ell_x+1} + an2^{\ell_x+\ell_s},$$
which is $< m^s$ with overwhelming probability. Secondly, observe that $\bar{x} + r \equiv x \bmod a$:
$$\begin{aligned}
\bar{x} + r &= (\tilde{x} \bmod a) + r && \{\text{phase } 3\}\\
&= \left(\left(x - r + a\sum_{i=1}^{n} s_i\right) \bmod m^s\right) \bmod a + r && \{\text{equation } (4.10)\}\\
&= \left(x - r + a\sum_{i=1}^{n} s_i\right) \bmod a + r && \{0 \le \tilde{x} < m^s\},
\end{aligned}$$
from which $\bar{x} + r \equiv x \bmod a$ clearly follows. Now, the value c computed in phase 4 satisfies:
$$c = 0 \iff a - 1 - \bar{x} \ge r \iff \bar{x} + r + 1 \le a \iff \bar{x} + r < a.$$
But as $\bar{x}, r \in [0, a)$, we have $\bar{x} + r \in [0, 2a)$, and thus the output in (4.12) clearly equals $\bar{x} + r \bmod a = x \bmod a$. □

Security of the protocol is proven in light of the real/ideal-model, Section 2.3. However, we need to adjust this model slightly using a technique of [CDN01], which is elaborated on by [ST06, GSV07]. This technique relies on 'Yet Another Distribution' (YAD). For a bit $b \in \{0,1\}$, YAD^b is defined such that for $b = 0$ it is perfectly indistinguishable from the distribution of ideal conversations, and for $b = 1$ it is statistically indistinguishable from the distribution of real conversations. Therefore, the real and ideal conversations are indistinguishable if and only if YAD^0 and YAD^1 are indistinguishable. But the difference between these two distributions only depends on the encrypted value $b \in \{0,1\}$, and hence the success probability of distinguishing between YAD^0 and YAD^1 equals the success probability of distinguishing between $[\![0]\!]$ and $[\![1]\!]$, which is negligible as the Paillier cryptosystem is semantically secure (Section 2.5.3). Therefore, given two values $x^{(0)}$ and $x^{(1)}$ drawn at random from the distributions YAD^0 and YAD^1, respectively, and a bit encryption $[\![b]\!]$, corresponding to either the ideal or real model, the real conversations need to be simulated for encrypted input $[\![x^{(0)}(1-b) + x^{(1)}b]\!]$. It is clear that this model is less general, but it is still sufficiently general to fit in the framework of [CDN01], [ST06]. In this model, we now construct a simulator that simulates Protocol 4.3.2 in a statistically indistinguishable way.

Proposition 4.3.4 (Security)**.** *On input of $x^{(0)}, x^{(1)}$ of length ℓ_x, $[\![b]\!]$ for $b \in \{0,1\}$, and a publicly known value a, Protocol 4.3.2 can be simulated in a statistically indistinguishable way.*

Proof. We construct a simulator for the computation of $\text{mod}_{P(\mu)}([\![x]\!], a)$, where $x = x^{(0)}(1-b) + x^{(1)}b$. The simulator may use the malicious parties $P' = \{\mathcal{P}_1, \ldots, \mathcal{P}_{t-1}\} \subset P$ as a subroutine. We note that the random bitwise value generation protocol, Protocol 4.3.1, can be simulated on input $([\![b]\!], r^{(0)}, r^{(1)})$ with $r^{(0)}, r^{(1)} \in [0, a)$. Moreover, threshold decryption can be simulated on input $[\![\tilde{x}]\!], \tilde{x}$ (see Section 2.5.3), and the comparison gate on input $([\![b]\!], r^{(0)}, r^{(1)})$ [GSV07].
Phase 1. The simulator takes $r \in_R [0, a)$ and simulates phase 1 with $\tilde{r} = \tilde{r}^{(0)}(1-b) + \tilde{r}^{(1)}b$, where:

$$\tilde{r}^{(b)} \leftarrow \left(r + x^{(b)}\right) \bmod a. \tag{4.13}$$

Note that the simulator knows the plaintext values $\tilde{r}^{(b)}$, and therefore the random bitwise value generation can be simulated on input $([\![b]\!], \tilde{r}^{(0)}, \tilde{r}^{(1)})$. For the parallel generation of the s_i's, the simulator lets the adversary publish $[\![s_i]\!]$ for $i = 1, \ldots, t-1$, rewinding the proof (special soundness) lets the simulator obtain s_i. For the participants $i = t, \ldots, n-1$ he executes this phase as is. For $\mathcal{P}_n$, he takes $s_n \in_R \{0,1\}^{\ell_x + \ell_s}$, but he publishes $[\![\tilde{s}_n]\!] = [\![\tilde{s}_n^{(0)}(1-b) + \tilde{s}_n^{(1)}b]\!]$, where:

$$\tilde{s}_n^{(b)} \leftarrow s_n - \left(r + x^{(b)}\right) \operatorname{div} a. \tag{4.14}$$

He simulates the proof of knowledge (4.9). By construction, $\tilde{r}$ and $\tilde{s}_n$ satisfy:

$$x - \tilde{r} + a\tilde{s}_n = -r + as_n. \tag{4.15}$$

Phase 2. Executed as is. Note that due to (4.15) $\tilde{x}$ satisfies:

$$\tilde{x} = x - \tilde{r} + a\sum_{i=1}^{n-1} s_i + a\tilde{s}_n = -r + a\sum_{i=1}^{n} s_i.$$

Phase 3. Simulated on input $[\![\tilde{x}]\!]$ and $-r + a\sum_{i=1}^{n} s_i$.
Phase 4. Simulated on input $([\![b]\!], \tilde{r}^{(0)}, \tilde{r}^{(1)})$. Notice that $a - 1 - \bar{x}$ is indeed known in plaintext.
Phase 5. Executed as is.
Finally, we need to prove that the simulated conversations are indistinguishable from the real conversations. To this end, first observe that by symmetry it suffices to consider $b = 0$. Now, r and $\tilde{r}$ are both $\in_R [0, a)$ and thus perfectly indistinguishably. For the indistinguishability of s_n first define $K := 2^{\ell_x + \ell_s}$. In the real conversations s_n is taken from $R := [0, K)$, and in the simulation we have $\tilde{s}_n = s_n - z$ with $s_n \in_R R$ and $z := (r + x^{(0)}) \operatorname{div} a < 2^{\ell_x}$ a fixed value. So $\tilde{s}_n$ is taken uniformly at random from $S := [-z, K - z)$. Define distributions $X = \{x \mid x \in_R R\}$ and $Y = \{y \mid y \in_R S\}$. Now, the statistical distance between X and Y is the following:

$$\begin{aligned} 2\Delta(X,Y) &= \sum_{v \in R \cup S} \Big| Pr(X = v) - Pr(Y = v) \Big| \\ &= \sum_{v \in [-z, 0)} Pr(Y = v) + \sum_{v \in [0, K-z)} \Big| Pr(X = v) - Pr(Y = v) \Big| + \sum_{v \in [K-z, K)} Pr(X = v) \\ &= z \cdot \frac{1}{|S|} + (K - z) \cdot 0 + z \cdot \frac{1}{|R|} = 2z/K, \end{aligned}$$

which is negligible in ℓ_s as $z < 2^{\ell_x}$. This completes the simulation, and by construction of the protocol (Proposition 4.3.3) the outcome in phase 5 indeed satisfies $\bar{x} + \tilde{r} \bmod a = x \bmod a$. □

4.3.1 Efficiency

Protocol 4.3.1 consists of ℓ_a random bit generations and one comparison gate on bit representations of length ℓ_a and this protocol needs to be executed < 2 times on average. Following Table 2.1, the broadcast complexity of this protocol varies between $O(nk\ell_a)$ (with round complexity $O(n)$) and $O(n^2k\ell_a)$ (in constant rounds).
The modulo reduction protocol, Protocol 4.3.2, requires one execution of the random bitwise

value generation protocol, and moreover needs one comparison on bit representations of length ℓ_a. Therefore, the modulo reduction protocol has average broadcast complexity varying between $O(nk\ell_a)$ (in $O(n)$ rounds) and $O(n^2k\ell_a)$ (in constant rounds).
In [DFK$^+$06], Damgård et al. construct a modulo reduction gate for verifiable secret sharing. The threshold homomorphic encryption analogue of this gate is less efficient than the one proposed in this section. In their proposal, they mainly need a bitrep(ℓ_x) gate and $\ell_x - 1$ parallel comp(ℓ_x) gates, resulting in a broadcast complexity varying between $O(nk\ell_x^2)$ and $O(n^2k\ell_x^2)$ (with round complexities $O(n + \ell_x)$ and $O(1)$, respectively). Even stronger however, in many practical applications of this protocol the value ℓ_a is rather small compared to ℓ_x: consider for example a scenario where the average fortune of 100 millionaires is securely computed. In this case $\ell_a = 7$, while $\ell_x = 37$ but needs to be extended to 47 to cover billionaires as well[4]. Note that Damgård et al.'s construction requires to compute the complete bit representation of x, while the idea of the proposed scheme relies on the form of a, as mentioned before.
Finally, one might want to skip the restriction that r needs to be less than a (which for Protocol 4.3.2 concretely means that in the execution of Protocol 4.3.1 phase 2 is omitted). In particular, the protocol is continued with $r \in_R [0, 2^{\ell_a})$ instead, but still with the same $\tilde{x}$ in (4.10). Consequently, either phase 4 requires two bit addition circuits or phase 5 requires two comparisons, in both cases costs are decreased. It is however not known if x is sufficiently hidden in $x - r + as$ then.

4.4 Sorting

Some statistics, for example the median and the MAD (median absolute deviation), can be computed by using a sorting gate. Also the range of l items, $x_{(l)} - x_{(1)}$, is an interesting value, but the computation of this value *does not need* a sorting gate: even the best sorting gate around needs $O(l \lg l)$ comparisons, while one can simply find the maximum value using a play-off format of $l - 1$ comparisons. Much study has been done on sorting protocols over plain values, and although there are many efficient gates around (merge, heap- and quicksort for just to mention some), these are not suited for secure multiparty computation. Indeed, their running time depends on the initial order of the list, through which information about the plaintexts leaks.

Yet there is a possibility. A well-known way of sorting is using a sorting network. The idea of a sorting network is understandably depicted in Figure 4.2 [Knu98]: vertical connections represent comparisons, and for each comparison the two values x_0 and x_1 are swapped if $x_0 > x_1$.

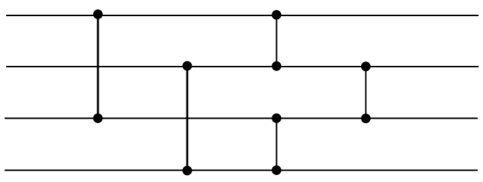

Figure 4.2: A sorting network. Connected wires are compared and interchanged if necessary.

The most efficient among all sorting network algorithms is the so-called 'odd-even mergesort' by Batcher [Bat68]. On input of l data items, it needs $O(l(\lg l)^2)$ comparisons in $O((\lg l)^2)$ rounds. The idea for the multiparty gate is that the n participants P jointly execute the algorithm by Batcher, and use a swap gate for each comparison. This swap gate can be constructed as follows (ℓ denotes the bit length of the encrypted values). We recall that we have n participants (t, n)-threshold sharing the secret key for the encryption scheme (either ElGamal or Paillier), and that $[\![x]\!]^{b(\ell)}$ denotes the set $\{[\![x_0]\!], \ldots, [\![x_{\ell-1}]\!]\}$ of encrypted bits of x.

[4]For simplicity we assume that the fortune of a millionaire is upper bounded by one billion. Now, the average is computed as $(x_1 + \ldots + x_{100})/100$, where x_i is the fortune of millionaire i. In the notation of the protocol, we have $x = \sum_{i=1}^{100} x_i$ and $a = 100$. Thus, $\ell_x = 37$ and $\ell_a = 7$.

Protocol 4.4.1 (Sorting two encryptions (swap))**.** Given $([\![x]\!]^{b(\ell)}, [\![y]\!]^{b(\ell)})$, the following protocol outputs encryptions $([\![\min\{x,y\}]\!]^{b(\ell)}, [\![\max\{x,y\}]\!]^{b(\ell)})$. The participants $\mathcal{P}_i$ $(i = 1, \dots, n)$ perform the following steps:

1. The participants jointly compute $c \leftarrow \text{comp}([\![x]\!]^{b(\ell)}, [\![y]\!]^{b(\ell)})$;

2. The participants jointly compute $s_j \leftarrow [\![x_j - y_j]\!] * c$ for $j = 0, \dots, \ell - 1$;

3. The participants individually simultaneously compute $[\![\bar{x}_j]\!] \leftarrow s_j[\![y_j]\!]$ and $[\![\bar{y}_j]\!] \leftarrow s_j^{-1}[\![x_j]\!]$ for $j = 0, \dots, \ell - 1$ and output $([\![\bar{x}]\!]^{b(\ell)}, [\![\bar{y}]\!]^{b(\ell)})$.

We note that the protocol is correct: if $x \geq y$ then $c = [\![0]\!]$ and hence the output is $([\![y]\!]^{b(\ell)}, [\![x]\!]^{b(\ell)})$, and otherwise nothing changes. Security follows from the security of the underlying gates. We note that phases 2 & 3 are done for each bit $j = 0, \dots, \ell - 1$ as we prefer the output to be in bit representation as well.

4.4.1 Efficiency

For complexity of the swap gate, we see that phase 1 is just one comparison (see Table 2.1), and phase 3 is trivial. Phase 2 involves ℓ multiplication gates, but they have a multiplier in common. This makes phase 2 almost as efficient as *one* normal multiplication gate, but now with broadcast complexity multiplied by ℓ. This results in broadcast complexity $O(nk\ell)$ in constant rounds. Consequently, the sorting protocol has broadcast complexity $O(l(\lg l)^2 nk\ell)$ in $O((\lg l)^2)$ rounds (remember that the sorting algorithm needs $O(l(\lg l)^2)$ comparisons and $O((\lg l)^2)$ rounds). Finally, if the encryptions are ElGamal, the round complexity needs a factor n due to the conditional gate. (The reader is referred to Table 2.1 for more information about the underlying gates.)

4.5 Multiparty computation of statistics

It turns out that using the constructions of the modulo reduction gate and the sorting gate, the participants P can do powerful statistical analysis. We briefly discuss the main statistics: the mean, variance, median, MAD, range and percentile. For one of them we need an implementation of the abs function, computing $[\![|x - y|]\!]$ from $[\![x]\!], [\![y]\!]$. This can be easily computed using one comparison: $[\![|x - y|]\!] \leftarrow [\![y - x]\!] * [\![2[x < y] - 1]\!]$. Now it is clear that the gates in Table 2.1 together with the modulo reduction and sorting gates of Sections 4.3 and 4.4 suffice to construct secure protocols for the following functions[5]. All functions below have input $([\![x_1]\!], \dots, [\![x_l]\!]) =: X$ for some l. Division is well-defined due to Section 4.3, and we use the notation $\div$.

$$\begin{aligned}
\text{mean} : X &\mapsto [\![\bar{x}]\!] = \left[\!\!\left[\frac{x_1 + \cdots + x_l}{l} \right]\!\!\right], \\
\text{var} : X &\mapsto [\![\sigma^2]\!] = \left[\!\!\left[\frac{1}{l-1} \sum_{i=1}^{l} (x_i - \bar{x})^2 \right]\!\!\right], \\
\text{med} : X &\mapsto [\![\tilde{x}]\!] = \left[\!\!\left[\frac{1}{2}(x_{(\lceil l/2 \rceil)} + x_{(\lfloor l/2+1 \rfloor)}) \right]\!\!\right], \\
\text{MAD} : X &\mapsto \left[\!\!\left[\text{med}\left([\![|x_1 - \tilde{x}|]\!], \dots, [\![|x_l - \tilde{x}|]\!]\right) \right]\!\!\right], \\
\text{range} : X &\mapsto [\![x_{(l)} - x_{(1)}]\!], \\
\text{percentile} : (X, k) &\mapsto \left[\!\!\left[\frac{100}{l} \sum_{i=1}^{l} [x_i \leq x_k] \right]\!\!\right].
\end{aligned}$$

[5]Although most of the functions can only be constructed for Paillier encryptions or for ElGamal encryptions with known bit representation.

As an example, we show how participants $P = \{\mathcal{P}_1, \dots, \mathcal{P}_n\}$, who (t, n)-threshold share secret key μ for the Paillier cryptosystem with public key (m, s), can compute the variance given l Paillier encryptions $\{[\![x_1]\!], \dots, [\![x_l]\!]\}$.

Example 4.5.1. Let $X := \{[\![x_1]\!], \dots, [\![x_l]\!]\}$ be a set of l Paillier encryptions, such that for each $i = 1, \dots, l$ we have $x_i < 2^{\ell_x}$. The participants P want to compute $\mathrm{var}(X)$. Notice that this function is equivalent to computing

$$\mathrm{var}(X) = \left[\!\!\left[\frac{1}{l-1} \sum_{i=1}^{l} x_i^2 - \frac{l}{l-1} \bar{x}^2 \right]\!\!\right] = \left[\!\!\left[\frac{1}{l(l-1)} \left(l \sum_{i=1}^{l} x_i^2 - \left(\sum_{i=1}^{l} x_i \right)^2 \right) \right]\!\!\right].$$

We denote $V := l \sum_{i=1}^{l} x_i^2 - (\sum_{i=1}^{l} x_i)^2$. Now, the participants can compute $\mathrm{var}(X)$ as follows.

1. Using $l+1$ multiplication gates (see Table 2.1), the participants jointly compute $[\![(\sum_{i=1}^{l} x_i)^2]\!]$ and $[\![x_i^2]\!]$ $(i = 1, \dots, l)$;
2. Each participant individually computes
$$[\![V]\!] \leftarrow \left(\prod_{i=1}^{l} [\![x_i^2]\!] \right)^{l} \cdot \left[\!\!\left[\left(\sum_{i=1}^{l} x_i \right)^2 \right]\!\!\right]^{-1};$$
3. The participants jointly compute and output $[\![V \bmod l(l-1)]\!] \leftarrow \mathrm{mod}_{P(\mu)}([\![V]\!], l(l-1))$ using Protocol 4.3.2.

From this output, the 'remainder after division', the encryption $\mathrm{var}(X) = [\![V \operatorname{div} l(l-1)]\!]$ is easily computed. In order for the modulo reduction protocol to succeed we need $nl^4 2^{2\ell_x + \ell_s} < m^s$, where ℓ_s is a security parameter. The complete protocol has broadcast complexity $O(lnk)$ and round complexity $O(n)$.

5. Certificate schemes: introduction

Digital certificate schemes provide certification authorities with the possibility to issue certificates, for instance digital driving licenses, to users. Users subsequently show these certificates to verifiers. A certificate generally certifies a list of attributes, like 'name, date of birth, etc.'. Main issues concerning certificate schemes are security and privacy, the former meaning that the scheme should be resistant against any possible cheating behavior (like attempts at forging a certificate), and the latter that executions cannot be linked, and that no private information leaks. For a more extended discussion on certificates, the reader is referred to Sections 2.8 and 3.2.1. In the remainder of this thesis, in line with Chapter 3 a certificate scheme will be extended so that it can be easily combined with multiparty computation schemes. In particular, the protocols mentioned in the architectural design in Section 3.2.3 are constructed for one particular certificate scheme so that all transitions in Figure 3.2 are well-defined. As mentioned in Section 3.4, the underlying scheme is chosen to be a discrete log based scheme of Brands [Bra99].

We recall some properties discussed in Section 3.4. Firstly, we consider the protocols for one user $\mathcal{U}$, one certification authority $\mathcal{CA}$ and one verifier $\mathcal{V}$. The certification authority is a set of n multiparty computation parties, $\mathcal{CA} = P$. Moreover, certificates issued by this authority include l attributes. Brands introduced several discrete log based certificate schemes, involving a group $G = \langle g \rangle$ of prime order q of size at least security parameter k, and all based on the DLREP assumption (Assumption 2.2.4). Informally, this assumption states that given $g_1, \ldots, g_l \in_R G$, it is hard to find values $x_1, \ldots, x_l \in \mathbb{Z}_q$ such that $g_1^{x_1} \cdots g_l^{x_l} = 1$, except for the trivial solution.

Based on the DLREP assumption, Brands introduced a 'standard' certificate scheme [Bra99, Fig. 4.7] where both the user and the certification authority need knowledge of the attributes before starting the issuing execution, a witness-indistinguishable extension to this standard scheme, which will not be considered, and a third scheme that is based on the blind Chaum-Pedersen signature protocol in Figure 2.4 [Bra99, Fig. 4.10], and for which $\mathcal{CA}$ need not know the plaintext attributes. The Chaum-Pedersen based certificate scheme is less efficient than the standard scheme, the certificates are larger, and moreover more pre-computation is required. Yet, the property that the $\mathcal{CA}$ need not know the attributes offers many advantages: the scheme is very useful for recertification, and in particular even the user does not need to know the attributes for certificate issuance [Bra99, Sec. 4.5.2]. The latter property turns out to be useful for constructing scheme **C**, the encrypted certificate scheme. In particular, the fact that in the standard scheme both $\mathcal{U}$ and $\mathcal{CA}$ need knowledge of the plain attributes before starting the issuing protocol would exclude the existence of protocol **B** and scheme **C**, or would make their constructions a lot more complicated. Therefore, the extensions that are introduced in Section 3.2.3 and discussed in this thesis, are based on Brands' Chaum-Pedersen based certificate scheme. For convenience, as of now it will simply be called 'Brands' certificate scheme'.

The extensions to Brands' certificate scheme are covered in Chapters 5-7. In Section 5.1, the formal definition of certificate schemes of Brands is reformulated and formalized, and extended to a definition for 'encrypted certificate schemes' (scheme **C** from Section 3.2.3). Brands' certificate scheme is introduced in Section 5.2. Chapters 6 and 7 discuss the extensions to Brands' scheme, namely the three components **A-C** from Section 3.2.3 for the ElGamal resp. the Paillier cryptosystem. Finally, some remarks and an efficiency analysis are included in Chapter 8.

5.1 Definition

Intuitively, a certificate scheme is constituted of three protocols and algorithms. Firstly, a key generation algorithm is needed for constructing public and secret keys. Secondly and thirdly, an issuing and a verification protocol are needed. A certificate scheme needs to satisfy security and privacy. Roughly stated, security means that a user cannot 'forge' certificates, either by coming up with $K+1$ certificates after $K \geq 0$ issuing executions[1], or by coming up with a certificate on a different attribute list. Even stronger, the user should not be able to convince the verifier with a corrupted certificate. Privacy is covered by requiring that no different protocol executions can be linked and that secret values do not leak. Now, it also becomes clear why two different definitions are required, one for the standard schemes where $\mathcal{U}$ learns the plaintext attributes, and one for the schemes where $\mathcal{U}$ only learns encryptions of the attributes: following Brands, in the first case the verification protocol is a zero-knowledge protocol for proving knowledge of the attributes, but this clearly does not work in the second case as $\mathcal{U}$ does not know the attributes at all. Moreover, as the second involves encryptions, it turns out that in its verification protocol also the verifier needs knowledge of some secret values, and therefore the definition of Σ-protocols would not suffice anymore.

We follow the definition by Brands, but we will deviate from his definition. This is done for several reasons. Firstly, we think that the new definition is better suited for constructing the extended definition of 'encrypted certificate schemes'. Secondly, Brands' definition [Bra99, Def. 4.1.1] is layered in the sense that the definition is founded on previous definitions. Our Definition 5.1.1 is stand-alone, which makes it easier to understand, and better comparable with the definition of the encrypted variant, Definition 5.1.2. Finally, Brands' definition only covers the issuing protocol, ours includes the specifications for the verification protocol as well. Moreover, the reformulated definition is more formal compared to Brands' definition. After Definition 5.1.1, the new definition will be compared with Brands' definition. Following Brands, we will call the two schemes 'restrictive blind certificate schemes' and 'restrictive blind *encrypted* certificate schemes', respectively. These two definitions will be compared at the end of this section.

Definition 5.1.1 (Restrictive blind certificate scheme)**.** A *restrictive blind certificate scheme* for a triple of parties $(\mathcal{CA},\mathcal{U},\mathcal{V})$, also called the 'certification authority', 'user' and 'verifier', consists of the following components:

- 'keygen' is an algorithm for $\mathcal{CA}$ that on input of security parameter k outputs a public/secret key pair (pk, sk), where pk includes the system parameters and a description of an *attributes set* S;
- 'issue' is a protocol for $(\mathcal{CA},\mathcal{U})$ that on input of pk, together with $\mathcal{CA}$'s private input sk and $\mathcal{U}$'s private input $s^* \in S$, called the '*attribute list*', outputs a certificate $(p, s, \sigma(p))$ for the user. This certificate satisfies that $\sigma(p)$ is a signature on p and that $\text{inv}(s) = s^*$, where inv is some polynomial time computable non-constant function, and that p is a public key part for which s is a secret key. (A part of) s^* might be known to $\mathcal{CA}$ before the issuing starts;
- 'verify' is a protocol for $(\mathcal{U},\mathcal{V})$ that on input of pk and $\mathcal{U}$'s input $(p, s, \sigma(p))$ outputs a bit, representing either acceptance or rejection.

These three components satisfy the following properties:

[1]This property of unforgeability trivially does not apply to certificate schemes with multi-show certificates. Brands' scheme, however, works with one-show certificates.

- (Completeness.) For any $s^* \in S$ and $(\mathcal{CA}, \mathcal{U}, \mathcal{V})$ following the protocol, the issuing protocol on input of s^* results in a valid certificate for $\mathcal{U}$. More formally, for any $s^* \in S$ we have

$$Pr\Big(1 \leftarrow \text{verify}_{\mathcal{U}(p,s,\sigma(p));\mathcal{V}}(pk) \;\Big|\; (pk, sk) \leftarrow \text{keygen}_{\mathcal{CA}}(k); \\ (p, s, \sigma(p)) \leftarrow \text{issue}_{\mathcal{CA}(sk);\mathcal{U}(s^*)}(pk)\Big) = 1;$$

- (Privacy for $\mathcal{U}$.) For any two issued certificates, if $\mathcal{U}$ followed the protocol, a malicious $\mathcal{CA}'$ cannot distinguish between the public key parts of these certificates. More formally, for any distinguisher $\mathcal{CA}'$ there exists a negligible $\varepsilon(k)$, such that for any $s_0^*, s_1^* \in S$ we have

$$Pr\Big(b \leftarrow \mathcal{CA}'(pk, sk, (p, \sigma(p))_b, (p, \sigma(p))_{1-b}, \text{view}_0, \text{view}_1) \;\Big|\; (pk, sk) \leftarrow \text{keygen}_{\mathcal{CA}'}(k); \\ b \in_R \{0, 1\}; (p, s, \sigma(p))_j \leftarrow \text{issue}_{\mathcal{CA}'(sk);\mathcal{U}(s_j^*)}(pk) \text{ for } j = 0, 1\Big) < \frac{1}{2} + \varepsilon(k),$$

where view_j denotes $\mathcal{CA}'$'s view on the j-th issuing execution ($j = 0, 1$), i.e. all values $\mathcal{CA}'$ sees during the execution (see also Section 2.1);

- (One-more unforgeability.) For any $(pk, sk) \leftarrow \text{keygen}_{\mathcal{CA}}(k)$ the following holds. Suppose that for any $K \geq 0$, malicious $\mathcal{U}'$ can perform K arbitrarily interleaving certificate queries on adaptively chosen attribute lists $s_j^* \in S$ ($j = 1, \ldots, K$). Then, the probability that $\mathcal{U}'$ outputs $K + 1$ distinct certificates is negligible in k. More formally, there exists a negligible $\varepsilon(k)$, such that for any $K \geq 0$ we have

$$Pr\Big(\forall_{i=1}^{K+1} \big[1 \leftarrow \text{verify}_{\mathcal{U}'((p,s,\sigma(p))_i);\mathcal{V}}(pk)\big] \;\Big|\; (pk, sk) \leftarrow \text{keygen}_{\mathcal{CA}}(k); \\ \{(p, s, \sigma(p))_i\}_{i=1}^{K+1} \leftarrow (\mathcal{U}')^{\text{issue}_{\mathcal{CA}(sk);\mathcal{U}'(\cdot)}(pk)}\Big) < \varepsilon(k),$$

where $\mathcal{U}'$ can query its oracle at most K times;

- (Blinding-invariance unforgeability.) For any $(pk, sk) \leftarrow \text{keygen}_{\mathcal{CA}}(k)$ the following holds. Suppose that for any $K \geq 0$, malicious $\mathcal{U}'$ can perform K arbitrarily interleaving certificate queries on adaptively chosen attribute lists $s_j^* \in S$ ($j = 1, \ldots, K$), and that $\mathcal{U}'$ outputs L certificates $((p, s, \sigma(p))_i)_{i=1}^{L}$ for some $L \leq K$. Then, with overwhelming probability for each i there exists a j such that $\text{inv}(s_i) = s_j^*$ (for the function inv described in the definition of the issuing protocol), and if moreover for multiple values of i that attribute list $\text{inv}(s_i)$ occurs, then for at least as many values j that list occurs as well. More formally, there exists a negligible $\varepsilon(k)$, such that for any $K \geq L \geq 0$ we have[2]

$$Pr\Big(\forall_{i=1}^{L} \big[1 \leftarrow \text{verify}_{\mathcal{U}'((p,s,\sigma(p))_i);\mathcal{V}}(pk)\big] \;\wedge\; R \not\subseteq \{s_j^*\}_{j=1}^{K} \;\Big|\; (pk, sk) \leftarrow \text{keygen}_{\mathcal{CA}}(k); \\ R := \{\text{inv}(s_i)\}_{i=1}^{L} \text{ a multiset}; \\ \big(\{(p, s, \sigma(p))_i\}_{i=1}^{L}; \{s_j^*\}_{j=1}^{K}\big) \leftarrow (\mathcal{U}')^{\text{issue}_{\mathcal{CA}(sk);\mathcal{U}'(\cdot)}(pk)}\Big) < \varepsilon(k),$$

where $\mathcal{U}'$ can query its oracle at most K times and $\{s_j^*\}_{j=1}^{K}$ is the multiset of queries by $\mathcal{U}'$;

- (Secure verification.) For any $(pk, sk) \leftarrow \text{keygen}_{\mathcal{CA}}(k)$ and any certificate $(p, s, \sigma(p))$, the protocol verify is a Σ-protocol for relation $R = \{(p, \sigma(p); s)\}$, where $\mathcal{U}$ pre-sent $(p, \sigma(p))$ to $\mathcal{V}$ (possibly together with additional information).

[2]We recall from general literature that a *multiset* is a set in which elements may occur multiple times.

Comparison of Definition 5.1.1 with Brands' [Bra99, Def. 4.1.1]. We briefly discuss the similarities and differences between Brands' definition and ours. First of all, it is clear that the components and security requirements in our definition are similar to the ones described in the corresponding definitions by Brands [Bra99, Defs. 4.1.1, 2.6.1, 2.5.1, 2.4.3, 2.4.1]. Moreover, these are indeed the properties required for a certificate scheme to offer 'security' and 'privacy'. The two unforgeability statements cover any possible forgery: a forger can either construct more certificates than he is issued on (one-more forgery), or less but on different attributes (blinding-invariance unforgeability). Also, a user cannot mislead a verifier with an incorrect certificate, by the special soundness property of the Σ-protocol. Privacy for all participants is also guaranteed: different executions cannot be linked by privacy for $\mathcal{U}$, and in the verification protocol the secret part of a certificate does not leak by the honest-verifier zero-knowledge property of the Σ-protocol. Also the secret key of $\mathcal{CA}$ does not leak, since otherwise forgeries would be possible.

Yet there are some differences between the two definitions, apart from the construction (Brands used instance generators, we use a key generation algorithm) and notations. Some observations:

- (Privacy for $\mathcal{U}$.) Brands' definition is stronger, saying that any issued public key part $(p, \sigma(p))$ and any view of $\mathcal{CA}'$ on an execution are statistically independent. We formalized it and switched over to indistinguishability (a generalization already suggested by Brands);
- (One-more unforgeability.) The definition is the same. Note that also in our definition the user can adaptively choose the attributes he will be issued on;
- (Blinding-invariance unforgeability.) Similar to Brands' definition, this second part of the definition may seem overly complex, but as we will explain, it is needed to capture all possible forgeries not being one-more forgeries. We note that, although not immediately clear, this definition drastically simplifies Brands' definition by means of using 'multisets', while still offering the same security property. Back to why this complex definition is necessary: suppose we would define this type of unforgeability roughly as 'given K issuing executions, it is hard to output any certificate on a *different* attribute list'. This would mean that the user 'sacrifices' the K issuing executions for *one* different certificate, but it does for instance not cover the case where a user is issued K certificates on different attribute lists $(s_j^*)_{j=1}^K$, and outputs 2 different certificates on s_1^*;
- (Secure verification.) We use the notion of Σ-protocols, which is stronger than the more general definition of honest-verifier zero-knowledge proof of knowledge [Bra99, Def. 2.4.3], but the notion of Σ-protocols suffices for the security of Brands' scheme of Section 5.2 and our schemes in Chapters 6 and 7.

Definition 5.1.2 (Restrictive blind encrypted certificate scheme)**.** A *restrictive blind encrypted certificate scheme* for a triple of parties $(\mathcal{CA}, \mathcal{U}, \mathcal{V})$, also called the 'certification authority', 'user' and 'verifier', consists of the following components:

- 'keygen' is an algorithm for $\mathcal{CA}$ that on input of security parameter k outputs a public/secret key pair (pk, sk), where pk includes the system parameters. It also includes a public/secret key pair for an encryption scheme and a description of an *encrypted attributes set* ES;
- 'issue' is a protocol for $(\mathcal{CA}, \mathcal{U})$ that on input of pk and $p^* \in ES$, called the '*encrypted attribute list*', together with $\mathcal{CA}$'s private input sk, outputs a certificate $(p, s, \sigma(p))$ for the user. This certificate satisfies that $\sigma(p)$ is a signature on p and that $\mathrm{inv}(p) \doteq p^*$, where inv is some polynomial time computable non-constant function, and that p is a public key part for which s is a secret key;
- 'verify' is a protocol for $(\mathcal{U}, \mathcal{V})$ that on input of pk and $\mathcal{U}$'s input $(p, s, \sigma(p))$ outputs a bit, representing either acceptance or rejection. $\mathcal{V}$ might have secret input as well;

These three components satisfy the following properties:

- (Completeness.) For any $p^* \in ES$ and $(\mathcal{CA}, \mathcal{U}, \mathcal{V})$ following the protocol, the issuing protocol on input of p^* results in a valid certificate for $\mathcal{U}$. More formally, for any $p^* \in ES$ we have

$$Pr\Big(1 \leftarrow \text{verify}_{\mathcal{U}(p,s,\sigma(p));\mathcal{V}}(pk) \;\Big|\; (pk, sk) \leftarrow \text{keygen}_{\mathcal{CA}}(k);$$
$$(p, s, \sigma(p)) \leftarrow \text{issue}_{\mathcal{CA}(sk);\mathcal{U}}(pk, p^*)\Big) = 1;$$

- (Privacy for $\mathcal{U}$.) For any two issued certificates, if $\mathcal{U}$ followed the protocol, a passively malicious p.p.t. $\mathcal{CA}'$ cannot distinguish between the public key parts of these certificates. More formally, there exists a negligible $\varepsilon(k)$, such that for any $p_0^*, p_1^* \in ES$ we have

$$Pr\Big(b \leftarrow \mathcal{CA}'(pk, sk, (p, \sigma(p))_b, (p, \sigma(p))_{1-b}, \text{view}_0, \text{view}_1) \;\Big|\; (pk, sk) \leftarrow \text{keygen}_{\mathcal{CA}'}(k);$$
$$b \in_R \{0, 1\}; (p, s, \sigma(p))_j \leftarrow \text{issue}_{\mathcal{CA}'(sk);\mathcal{U}}(pk, p_j^*) \text{ for } j = 0, 1\Big) < \frac{1}{2} + \varepsilon(k),$$

where view_j denotes $\mathcal{CA}'$'s view on the j-th issuing execution ($j = 0, 1$), i.e. all values $\mathcal{CA}'$ sees during the execution (see also Section 2.1);

- (One-more unforgeability.) For any $(pk, sk) \leftarrow \text{keygen}_{\mathcal{CA}}(k)$ the following holds. Suppose that for any $K \geq 0$, malicious $\mathcal{U}'$ can perform K arbitrarily interleaving certificate queries on encrypted attribute lists $p_j^* \in ES$ ($j = 1, \ldots, K$). Then, the probability that $\mathcal{U}'$ outputs $K + 1$ distinct certificates is negligible in k. More formally, there exists a negligible $\varepsilon(k)$, such that for any $K \geq 0$ and for any $p_j^* \in ES$ ($j = 1, \ldots, K$) we have

$$Pr\Big(\forall_{i=1}^{K+1}\big[1 \leftarrow \text{verify}_{\mathcal{U}'((p,s,\sigma(p))_i);\mathcal{V}}(pk)\big] \;\Big|\; (pk, sk) \leftarrow \text{keygen}_{\mathcal{CA}}(k);$$
$$\{(p, s, \sigma(p))_i\}_{i=1}^{K+1} \leftarrow \mathcal{U}'\Big(\{\text{issue}_{\mathcal{CA}(sk);\mathcal{U}'}(pk, p_j^*)\}_{j=1}^{K}\Big)\Big) < \varepsilon(k);$$

- (Blinding-invariance unforgeability.) For any $(pk, sk) \leftarrow \text{keygen}_{\mathcal{CA}}(k)$ the following holds. Suppose that for any $K \geq 0$, malicious $\mathcal{U}'$ can perform K arbitrarily interleaving certificate queries on encrypted attribute lists $p_j^* \in ES$ ($j = 1, \ldots, K$), and that $\mathcal{U}'$ outputs L certificates $((p, s, \sigma(p))_i)_{i=1}^{L}$ for some $L \leq K$. Then, with overwhelming probability for each i there exists a j such that $\text{inv}(p_i) \doteq p_j^*$ (for the function inv described in the definition of the issuing protocol), and if moreover for multiple values of i, $\text{inv}(p_i)$ encrypts the same attribute list, then for at least as many values j, p_j^* encrypts that list as well. More formally, there exists a negligible $\varepsilon(k)$, such that for any $K \geq L \geq 0$ and for any $p_j^* \in ES$ ($j = 1, \ldots, K$) we have

$$Pr\Big(\forall_{i=1}^{L}\big[1 \leftarrow \text{verify}_{\mathcal{U}'((p,s,\sigma(p))_i);\mathcal{V}}(pk)\big] \;\wedge\; R \not\subseteq Q \;\Big|\; (pk, sk) \leftarrow \text{keygen}_{\mathcal{CA}}(k);$$
$$R := \{D(\text{inv}(p_i))\}_{i=1}^{L} \text{ and } Q := \{D(p_j^*)\}_{j=1}^{K} \text{ multisets};$$
$$\{(p, s, \sigma(p))_i\}_{i=1}^{L} \leftarrow \mathcal{U}'\Big(\{\text{issue}_{\mathcal{CA}(sk);\mathcal{U}'}(pk, p_j^*)\}_{j=1}^{K}\Big)\Big) < \varepsilon(k);$$

- (Secure verification.) For any $(pk, sk) \leftarrow \text{keygen}_{\mathcal{CA}}(k)$ and any certificate $(p, s, \sigma(p))$, the protocol verify is a secure two-party protocol for proving knowledge of s such that $(p, s, \sigma(p))$ is a valid certificate, where $\mathcal{U}$ pre-sent $(p, \sigma(p))$ to $\mathcal{V}$ (possibly together with additional information).

Comparison of Definitions 5.1.1 and 5.1.2. There are three major differences between the schemes. Firstly, for the privacy for $\mathcal{U}$ of *encrypted* certificate schemes of Definition 5.1.2 the certification authority may only be passively malicious and operate in probabilistic polynomial

time. These requirements are trivially needed since the certificates involve encryptions: if $\mathcal{CA}'$ was actively malicious, he can use his decryption key to decrypt $\text{inv}(p_b)$ and p_0^*, in which case he outputs $b = 0$ if these encrypt the same values, and $b = 1$ otherwise. Also, if $\mathcal{CA}'$ would have unlimited computer power the encryption scheme is insecure and he can apply the same attack. We note that these restrictions are not harmful: in particular, as we require that $\mathcal{CA} = P$ is a set of multiparty computation participants, the passively adversarial behavior is naturally enforced.
Second major difference is in the way attribute lists are defined and processed. In Definition 5.1.2, attributes are in encrypted instead of plaintext form (to emphasize this difference, they are called *encrypted* attribute lists and the notation ES is used instead of S). As a consequence, in general also the certification authority knows p^* and it belongs to the public key part. This leads to a re-definition of how the attribute list is encapsulated in the final public key part p, as now the encryptions need to be re-blinded (to maintain unlinkability). This explains the changes in the definition of the issuing protocol. Consequently, for blinding-invariance unforgeability it should hold that each p encapsulates a blinded version of some p^*. An important difference is that in Definition 5.1.2 for unforgeability a forger is not anymore allowed to query the issuing protocol on *adaptively* chosen encrypted attribute lists. This is done for facilitating the proof of Proposition 6.4.5: the construction of this proof requires a simulator to have knowledge of the encrypted values in the encrypted attribute list, but for the ElGamal encryption scheme under the DL Assumption 2.2.1 this is hard to achieve, unless the simulator can construct the encryptions himself. Note that $\mathcal{U}$ choosing the attributes himself contradicts the idea of encrypted certificates. Moreover, this definition suffices for our purposes as in our scenario the $\mathcal{CA}$ himself comes up with an encryption to be certified.
Finally, as in the new definition the verifier may have secret input as well, it does not suffice to state that the verification protocol should be a Σ-protocol. See also Section 6.4.5 for a more extended discussion.

5.2 Brands' certificate scheme

In this section, Brands' certificate scheme is introduced. It extends the blind Chaum-Pedersen signature scheme as given in Figure 2.4. The scheme is discussed very briefly, the reader is referred to [Bra99] for a more extended discussion.

5.2.1 Key generation

On input of security parameter k, the certification authority obtains public key (q, g), with $q > 2^k$, and $(h_0, g_1, \dots, g_l)$, together with secret key $x_0, (y_i)_{i=1}^l \in_R \mathbb{Z}_q$ satisfying

$$h_0 = g^{x_0}, \qquad\qquad \forall_{i=1}^l : \; g_i = g^{y_i}.$$

The attributes set is $(\mathbb{Z}_q)^l$. A certificate on an attribute list $s^* = (x_i)_{i=1}^l \in (\mathbb{Z}_q)^l$ is a tuple $(h', (x_i)_{i=1}^l, \alpha_1, z', c_0', r_0')$ satisfying:

$$c_0' = \mathcal{H}(h', z', g^{r_0'} h_0^{-c_0'}, (h')^{r_0'} (z')^{-c_0'}) \text{ and } (g_1^{x_1} \cdots g_l^{x_l} h_0)^{\alpha_1} = h'. \tag{5.1}$$

Here, $p = h'$ is the public key part with corresponding secret key part $s = ((x_i)_{i=1}^l, \alpha_1)$, in light of Definition 5.1.1. The element h' is required to be different from 1. If in the issuing it turns out that $h' = 1$, it is declared invalid and the protocol is restarted.

5.2.2 Certificate issuance

On input of the attribute list s^*, firstly the value $h = g_1^{x_1} \cdots g_l^{x_l} h_0$ is constructed (it may either be constructed by $\mathcal{CA}$, or $\mathcal{U}$, or jointly). Now, the issuing protocol is given in Figure 5.1. In case of multiple issuing executions on the same encrypted attribute list, the value z needs to be computed only once.

$$\begin{array}{rcl}
\mathcal{U} & & \mathcal{CA} \\
\text{(knows: } h) & & \text{(knows: } h) \\
 & & z \leftarrow h^{x_0} \\
 & & w_0 \in_R \mathbb{Z}_q \\
 & \xleftarrow{\quad z;a_0,b_0 \quad} & a_0 \leftarrow g^{w_0}, \quad b_0 \leftarrow h^{w_0} \\
\alpha_1 \in_R \mathbb{Z}_q^*, \quad \alpha_2, \alpha_3 \in_R \mathbb{Z}_q & & \\
h' \leftarrow h^{\alpha_1}, \quad z' \leftarrow z^{\alpha_1} & & \\
a_0' \leftarrow h_0^{\alpha_2} g^{\alpha_3} a_0 & & \\
b_0' \leftarrow (z')^{\alpha_2} (h')^{\alpha_3} b_0^{\alpha_1} & & \\
c_0' \leftarrow \mathcal{H}(h', z', a_0', b_0') & & \\
c_0 \leftarrow c_0' + \alpha_2 \bmod q & \xrightarrow{\quad c_0 \quad} & \\
 & \xleftarrow{\quad r_0 \quad} & r_0 \leftarrow c_0 x_0 + w_0 \bmod q \\
a_0 \overset{?}{=} g^{r_0} h_0^{-c_0}, \quad b_0 \overset{?}{=} h^{r_0} z^{-c_0} & & \\
r_0' \leftarrow r_0 + \alpha_3 \bmod q & &
\end{array}$$

Figure 5.1: Brands' issuing protocol [Bra99].

5.2.3 Certificate verification

For a certificate $(h', (x_i)_{i=1}^l, \alpha_1, z', c_0', r_0')$, the verification protocol should be a Σ-protocol for proving knowledge of $((x_i)_{i=1}^l, \alpha_1)$, after the user sent (h', z', c_0', r_0') to the verifier. Brands already recognized the usefulness of selective disclosure. His thesis includes an extended study on how to disclose several relations on $(x_i)_{i=1}^l$ [Bra99, Ch. 3], like how to show that $37x_1 + x_2 \equiv 5x_3 \bmod q$ without disclosing (x_1, x_2, x_3). We will not consider these types of disclosure, we only show how the user can disclose x_1 and prove knowledge of all other values. Note that a protocol for disclosing nothing (the intended protocol at the beginning of this paragraph) is a simplification of the protocol to be constructed.

So the user needs to prove that he knows $((x_i)_{i=2}^l, \alpha_1)$ such that (5.1) holds. As (h', z', c_0', r_0') are sent to $\mathcal{V}$, the equality $c_0' = \mathcal{H}(h', z', g^{r_0'} h_0^{-c_0'}, (h')^{r_0'} (z')^{-c_0'})$ can be easily checked publicly. So $\mathcal{U}$ and $\mathcal{V}$ execute a Σ-protocol for relation

$$\{(h', x_1; (x_i)_{i=2}^l, \alpha_1) \mid g_1^{x_1} h_0 = (h')^{\alpha_1^{-1}} g_2^{-x_2} \cdots g_l^{-x_l} \wedge \alpha_1 \neq 0\}. \tag{5.2}$$

Note that indeed $\alpha_1 \neq 0$ should hold as $h' \neq 1$. The Σ-protocol is given in Figure 5.2.
The protocol can be made non-interactive by setting $c \leftarrow \mathcal{H}(\underline{a})$, where $\underline{a}$ denotes all values sent in the first round. Yet the $\mathcal{H}$ must include an extra value (like a unique number) to prevent replaying by $\mathcal{V}$. As of now, as the equality $c_0' = \mathcal{H}(h', z', g^{r_0'} h_0^{-c_0'}, (h')^{r_0'} (z')^{-c_0'})$ can be easily checked publicly, it is assumed to hold. Moreover, the public values h', z', c_0', r_0' are assumed to be sent to $\mathcal{V}$ before the verification starts.

5.2.4 Security analysis

[Bra99, Sec. 4.5.2] very briefly discusses the security of this scheme. First of all it is stated that x_0 is believed to be securely hidden. In light of Definition 5.1.1: completeness, privacy for $\mathcal{U}$, and the proof that Figure 5.2 is a Σ-protocol are trivial, and simplifications of proofs in Section 6.4. One-more unforgeability is implied by the assumed unforgeability of the Chaum-Pedersen signature scheme (see Section 2.7), the prove is similar to the proof of Proposition 6.4.5. Blinding-invariance unforgeability is assumed, Assumption 5.2.1 [Bra99, Ass. 4.4.5].

$$\begin{array}{rcl}
\mathcal{U} & & \mathcal{V}\\
(\text{knows: } h', z', c_0', r_0'; (x_i)_{i=1}^{l}, \alpha_1) & & \\
u_2,\ldots,u_l,u_\alpha \in_R \mathbb{Z}_q & & \\
a_h \leftarrow (h')^{u_\alpha} g_2^{-u_2}\cdots g_l^{-u_l} & \xrightarrow{a_h; h', z', c_0', r_0', x_1} & \\
 & \xleftarrow{c} & c \in_R \mathbb{Z}_q\\
(r_i \leftarrow u_i + c x_i \bmod q)_{i=2}^{l} & & \\
r_\alpha \leftarrow u_\alpha + c\alpha_1^{-1} \bmod q & \xrightarrow{(r_i)_{i=2}^{l}, r_\alpha} & \\
 & & c_0' \overset{?}{=} \mathcal{H}(h', z', g^{r_0'} h_0^{-c_0'}, (h')^{r_0'}(z')^{-c_0'})\\
 & & (h')^{r_\alpha} g_2^{-r_2}\cdots g_l^{-r_l} \overset{?}{=} (g_1^{x_1} h_0)^c a_h
\end{array}$$

Figure 5.2: Brands' verification protocol [Bra99].

Assumption 5.2.1. If $\mathcal{U}'$ produces, after $K \geq 0$ arbitrarily interleaving executions of the protocol in Figure 5.1 on adaptively chosen $(x_{ji})_{i=1}^{l}$ $(j = 1, \ldots, K)$ a tuple $(h', (x_i)_{i=1}^{l}, \alpha_1, z', c_0', r_0')$, then either this tuple does not satisfy (6.2), or with overwhelming probability there exists a $j \in \{1, \ldots, K\}$ such that $(x_i)_{i=1}^{l} = (\alpha_1 x_{ji} \bmod q)_{i=1}^{l}$.

6. Certificate schemes: ElGamal extensions

In this chapter the certificate scheme of Brands of Section 5.2 [Bra99, Sec. 4.5.2] is combined with the ElGamal cryptosystem, in such a way that all transitions in Figure 3.2 are possible. More precisely, components **A**-**C** of Section 3.2.3 are constructed. We recall these extensions:

A: (Encrypted disclosure of certificates.) In Brands' scheme, an option to disclose plaintext attributes in the certificate verification is included. This can be easily extended to an 'encrypted disclosure', where the user discloses the attributes in encrypted form. This is useful if the verifier is a set of multiparty computation participants ($\mathcal{V} = P$). Its construction is based on composing different Σ-protocols;

B: (Attribute hiding issuance.) We can consider the case where $\mathcal{CA} = P$ obtains many encryptions (e.g. with the 'encrypted disclosure' method) and executes some multiparty computation protocol on it. But different to the standard interpretation of the idea of multiparty computation, now $\mathcal{CA}$ not only outputs the encrypted result, but certifies it as well. More abstractly: $\mathcal{CA}$ possesses an encryption c (with the plaintext unknown), and wants to certify it. In this scenario, $\mathcal{U}$ still learns the attributes;

C: (Encrypted certificate scheme.) In contrast to protocol **B**, in this case *even $\mathcal{U}$ does not learn the plaintext.* So $\mathcal{CA}$ issues a certificate on an encryption to the user, and the user shows the certificate and the encryption to the verifier, so that the scheme is indeed a restrictive blind encrypted certificate scheme (we need the extended Definition 5.1.2). It turns out that the condition $\mathcal{CA} = \mathcal{V}$ is required[1].

Protocols **A** and **B** and scheme **C** are discussed in Sections 6.1-6.3, respectively, and an extensive security analysis of scheme **C** is considered in Section 6.4. In Section 8.1.1 we will discuss in what sense the restrictions '$P = \mathcal{CA}$', '$P = \mathcal{V}$' and '$P = \mathcal{CA} = \mathcal{V}$' constrain the generality of the scheme. We stress that this chapter only considers the ElGamal cryptosystem. Chapter 7 discusses the components in case the Paillier cryptosystem is utilized.

For the ElGamal encryption scheme, we assume that the key generation is implicitly included in the key generation algorithm of the certificate scheme. We work over the same group used for the certificate scheme, $G = \langle g \rangle$ of prime order $q > 2^k$ for security parameter k. The public key for encryption is f, corresponding to secret key λ. This secret key is shared among the n multiparty computation participants P, such that t out of n parties can decrypt. Each participant holds polynomial share λ_i. As this chapter is only concerned with ElGamal encryptions, these will be denoted by $[\![x]\!]$ (without subscript '$_G$'). We recall from Section 3.4 that in case the participants P jointly act as a verifier or certification authority, we just consider them as one player. In particular, if for instance $P = \mathcal{CA}$, the secret keys obtained from the key generation algorithm are implicitly (t, n)-threshold shared. To this end, the same distributed key generation protocol as for sharing λ can be used (Section 2.5.1).

[1]If we facilitate the scheme so that $\mathcal{CA}$ *does* learn the plaintext attributes, but still $\mathcal{U}$ does not, it turns out that we do not need this restriction. See Section 8.1.1.

6.1 Protocol A: encrypted disclosure

As an extension to the verification protocol in Section 5.2, the user discloses *encryptions* of the attributes. It turns out that this extension can be constructed by only making modifications in the verification protocol, Figure 5.2. Without loss of generality we assume the same setting as in Section 5.2.3: the user has a certificate $(h', (x_i)_{i=1}^{l}, \alpha_1, z', c_0', r_0')$ satisfying (5.1), and he discloses x_1 in encrypted form and keeps the other attributes secret. A protocol for disclosing multiple encryptions follows immediately. Also, using the same ideas as in [Bra99, Ch. 3], it is possible to disclose encryptions plus a relation on the encrypted values, e.g. $37x_1 + x_2 \equiv 5x_3 \bmod q$. We assume that $\mathcal{U}$ sends $[\![x_1]\!]$ and not its bit representation, although the latter might be more useful if $\mathcal{V}$ uses the encryptions for multiparty computation (see Section 2.9). The reason for this is that disclosing in the latter way is generally very inefficient.

Together with the public part of the certificate (h', z', c_0', r_0'), the user now also sends an ElGamal encryption of x_1 to the verifier. Therefore, he takes $r \in_R \mathbb{Z}_q$, sets $(c_1, c_2) = [\![x_1, r]\!] = (g^r, g^{x_1} f^r)$, and sends these values to $\mathcal{V}$. The user now proves knowledge for relation

$$\{(h', c_1, c_2; (x_i)_{i=1}^{l}, \alpha_1, r) \mid h_0 = (h')^{\alpha_1^{-1}} g_1^{-x_1} \cdots g_l^{-x_l} \wedge \alpha_1 \neq 0 \wedge c_1 = g^r \wedge c_2 = g^{x_1} f^r\}. \qquad (6.1)$$

The Σ-protocol is given in Figure 6.1.

$$\begin{array}{rcl}
\mathcal{U} & & \mathcal{V} \\
\text{(knows: } h', c_1, c_2; (x_i)_{i=1}^{l}, \alpha_1, r) & & \text{(knows: } h', c_1, c_2) \\
(u_i)_{i=1}^{l}, u_\alpha, u_r \in_R \mathbb{Z}_q & & \\
a_h \leftarrow (h')^{u_\alpha} g_1^{-u_1} \cdots g_l^{-u_l} & & \\
a_1 \leftarrow g^{u_r}, \quad a_2 \leftarrow g^{u_1} f^{u_r} & \xrightarrow{\quad a_h, a_1, a_2 \quad} & \\
 & \xleftarrow{\qquad c \qquad} & c \in_R \mathbb{Z}_q \\
(r_i \leftarrow u_i + c x_i \bmod q)_{i=1}^{l} & & \\
r_\alpha \leftarrow u_\alpha + c\alpha_1^{-1} \bmod q & & \\
r_r \leftarrow u_r + cr \bmod q & \xrightarrow{\quad (r_i)_{i=1}^{l}, r_\alpha, r_r \quad} & \\
 & & (h')^{r_\alpha} g_1^{-r_1} \cdots g_l^{-r_l} \overset{?}{=} a_h h_0^c \\
 & & g^{r_r} \overset{?}{=} a_1 c_1^c, \quad g^{r_1} f^{r_r} \overset{?}{=} a_2 c_2^c
\end{array}$$

Figure 6.1: Brands' verification protocol, with ElGamal encrypted disclosure.

In light of Definition 5.1.1, we need to prove that this verification protocol is a Σ-protocol for (6.1).

Proposition 6.1.1. *The protocol in Figure 6.1 is a Σ-protocol for (6.1).*

Proof. (Completeness). For the first verification equation we have:

$$\begin{aligned}
(h')^{r_\alpha} g_1^{-r_1} \cdots g_l^{-r_l} &= (h')^{u_\alpha + c\alpha_1^{-1}} g_1^{-u_1 - cx_1} \cdots g_l^{-u_l - cx_l} \\
&= (h')^{u_\alpha} g_1^{-u_1} \cdots g_l^{-u_l} ((h')^{\alpha_1^{-1}} g_1^{-x_1} \cdots g_l^{-x_l})^c \\
&= a_h h_0^c.
\end{aligned}$$

Similarly, also the other two equations hold:

$$\begin{aligned}
g^{r_r} &= g^{u_r + cr} = g^{u_r}(g^r)^c = a_1 c_1^c, \\
g^{r_1} f^{r_r} &= g^{u_1 + cx_1} f^{u_r + cr} = g^{u_1} f^{u_r} (g^{x_1} f^r)^c = a_2 c_2^c.
\end{aligned}$$

(Special soundness). Suppose we have two successful conversations $(a_h, a_1, a_2; c; (r_i)_{i=1}^{l}, r_\alpha, r_r)$ and $(a_h, a_1, a_2; c'; (r'_i)_{i=1}^{l}, r'_\alpha, r'_r)$ with $c \neq c'$. Then:

$$(h')^{r_\alpha} g_1^{-r_1} \cdots g_l^{-r_l} = a_h h_0^c \ \wedge\ (h')^{r'_\alpha} g_1^{-r'_1} \cdots g_l^{-r'_l} = a_h h_0^{c'},$$

which gives $(h')^{r_\alpha - r'_\alpha} g_1^{r'_1 - r_1} \cdots g_l^{r'_l - r_l} = h_0^{c-c'}$, implying (as $c \neq c'$):

$$(h')^{r_\alpha - r'_\alpha} = \left(g_1^{\frac{r_1 - r'_1}{c-c'}} \cdots g_l^{\frac{r_l - r'_l}{c-c'}} h_0 \right)^{c-c'}.$$

As $r_\alpha = r'_\alpha$ would contradict the DLREP assumption (Assumption 2.2.4), we can exponentiate both sides to $(r_\alpha - r'_\alpha)^{-1}$, obtaining witnesses

$$x_i = \frac{r_i - r'_i}{c - c'} \text{ for } i = 1, \ldots, l, \qquad \alpha_1 = \frac{c - c'}{r_\alpha - r'_\alpha} \neq 0.$$

In a similar way, from $g^{r_r} \overset{?}{=} a_1 c_1^c$ and $g^{r'_r} \overset{?}{=} a_1 c_1^{c'}$ the witness $r = \frac{r_r - r'_r}{c - c'}$ can be obtained. The witnesses x_1 and r satisfy $c_2 = g^{x_1} f^r$ as well.
(Honest-verifier zero-knowledge). The following distributions of real and simulated conversations are identical:

$$\begin{aligned}
\Big\{(a_h, a_1, a_2; c; (r_i)_{i=1}^{l}, r_\alpha, r_r) \ \Big| \ & (u_i)_{i=1}^{l}, u_\alpha, u_r, c \in_R \mathbb{Z}_q; a_h \leftarrow (h')^{u_\alpha} g_1^{-u_1} \cdots g_l^{-u_l}, \\
& a_1 \leftarrow g^{u_r}, a_2 \leftarrow g^{u_1} f^{u_r}; (r_i \leftarrow u_i + c x_i \bmod q)_{i=1}^{l}, \\
& r_\alpha \leftarrow u_\alpha + c\alpha_1^{-1} \bmod q, r_r \leftarrow u_r + cr \bmod q \Big\}, \\
\Big\{(a_h, a_1, a_2; c; (r_i)_{i=1}^{l}, r_\alpha, r_r) \ \Big| \ & (r_i)_{i=1}^{l}, r_\alpha, r_r, c \in_R \mathbb{Z}_q; \\
& a_h \leftarrow (h')^{r_\alpha} g_1^{-r_1} \cdots g_l^{-r_l} h_0^{-c}, a_1 \leftarrow g^{r_r} c_1^{-c}, a_2 \leftarrow g^{r_1} f^{r_r} c_2^{-c} \Big\}.
\end{aligned}$$

As in both distributions the c is chosen freely and uniformly at random, the protocol is *special* honest-verifier zero-knowledge. □

6.2 Protocol B: attribute hiding issuance

In this case, we have $\mathcal{CA} = P$ who, as a team of multiparty computation servers, outputs an encryption $c_1 = [\![x_1]\!]$ that needs to be certified. Now, $\mathcal{CA}$ is not allowed to learn x_1, but $\mathcal{U}$ is (for the case in which even this is not allowed, we refer to Section 6.3).

We only consider $l = 2$, so we consider two attributes[2] (x_1, x_2). It turns out that for protocol **B** the verification protocol can be executed the same as previously (Section 5.2.3 or 6.1). In particular, also the issuing protocol of Section 5.2.2 remains the same, but must be preceded by a sub-protocol. Indeed, in the mentioned issuing protocol we require that, before the issuing starts, $\mathcal{CA}$ knows $h = g_1^{x_1} g_2^{x_2} h_0$ and $\mathcal{U}$ knows h, x_1, x_2. But now, both players only know c_1 (it can be assumed that $\mathcal{U}$ already knows c_1). More formally, the general issuing protocol is given in Figure 6.2.

We briefly summarize the setting. We consider a user $\mathcal{U}$ and a certification authority $\mathcal{CA}$, which is represented by a set of n participants $P = \{\mathcal{P}_1, \ldots, \mathcal{P}_n\}$. The key generation algorithm in Section 5.2.1 resulted in a group $G = \langle g \rangle$ of prime order q, as well as values $h_0, g_1, g_2 \in G$, for which the certification authority knows the discrete logarithms x_0, y_1, y_2, respectively. These secret values

[2]Note that $l = 1$ is impossible since then x_1 is not perfectly hidden in $h = g_1^{x_1} h_0$.

$$\begin{tikzcd}
\mathcal{U} & & & \mathcal{CA} \\
(\text{knows: } c_1) & & & (\text{knows: } c_1) \\
 & {} \arrow[r, leftrightarrow, "\text{sub-protocol}"] & {} & \\
(\text{knows: } h; x_1, x_2) & & & (\text{knows: } h) \\
 & {} \arrow[r, leftrightarrow, "\text{Figure 5.1}"] & {} &
\end{tikzcd}$$

Figure 6.2: Attribute hiding issuing protocol, general.

are (t, n)-threshold shared among the participants P. Furthermore, for the same group G an ElGamal cryptosystem is set up with public key $f = g^\lambda$, where λ is (t, n)-threshold shared among the participants as well. Both $\mathcal{U}$ and $\mathcal{CA}$ know encryption $c_1 = (g^r, g^{x_1} f^r)$ for some $r \in_R \mathbb{Z}_q$, and the sub-protocol should result in $h = g_1^{x_1} g_2^{x_2} h_0$, where $\mathcal{U}$ knows x_1 and $x_2 \in_R \mathbb{Z}_q$.

Using a 'private output protocol', the required sub-protocol can be implemented easily. In a private output protocol, only *one* participant (or more generally, a restricted number of participants) learns the encrypted value. In Section 2.9.1, a protocol for private output is given. Now, the sub-protocol can be implemented by first executing the private output protocol on c_1, and then letting the user take $x_2 \in_R \mathbb{Z}_q$ and construct h plus a Σ-protocol for proving that h is constructed correctly. However, the fact that the ElGamal cryptosystem and the certificate scheme are over the same group allows us to construct a more efficient protocol. Participants $\mathcal{U}$ and $\mathcal{CA}$ execute the sub-protocol in Figure 6.3, where Σ is a Σ-protocol for proving knowledge of the construction of c_2. The protocol only works if x_1 is from a restricted domain as the user would otherwise not be able to compute this value, as explained in Section 2.5.1.

$$\begin{tikzcd}
\mathcal{U} & & & \mathcal{CA} \\
(\text{knows: } c_1) & & & (\text{knows: } c_1; y_1, y_2, \lambda) \\
x_2, t \in_R \mathbb{Z}_q, \quad c_2 \leftarrow (g^t, g^{x_2} f^t) & {} \arrow[r, "c_2"] & {} & \\
 & {} \arrow[r, leftrightarrow, "\Sigma"] & {} & \\
 & {} & {} \arrow[l, "h"'] & h \leftarrow D(c_1^{y_1} c_2^{y_2}) h_0 \\
x_1 \leftarrow \log_{g_1}(h/(g_2^{x_2} h_0)) & & & \\
(\text{knows: } h; x_1, x_2) & & & (\text{knows: } h)
\end{tikzcd}$$

Figure 6.3: Sub-protocol for attribute hiding issuing protocol, ElGamal.

We note that the equation $h = g_1^{x_1} g_2^{x_2} h_0$ is satisfied:

$$\begin{aligned}
c_1^{y_1} &= (g^r, g^{x_1} f^r)^{y_1} = (g^{ry_1}, g^{x_1 y_1} f^{ry_1}) = [\![g_1^{x_1}]\!], \\
c_2^{y_2} &= (g^t, g^{x_2} f^t)^{y_2} = (g^{ty_2}, g^{x_2 y_2} f^{ty_2}) = [\![g_2^{x_2}]\!],
\end{aligned}$$

from which the claimed immediately follows. The participants P can compute $D(c_1^{y_1} c_2^{y_2}) h_0$ by first executing Protocol 4.2.4, resulting in an encryption $\tilde{c} = c_1^{y_1} c_2^{y_2}$ which can then be threshold decrypted. Clearly, $\mathcal{CA} = P$ cannot learn the plaintext values x_1, x_2 as Σ is honest-verifier zero-knowledge and c_1, c_2, h all perfectly hide x_1, x_2. Finally, the protocol does not leak y_1, y_2, λ, since computing $[\![\tilde{c}]\!] \leftarrow c_1^{y_1} c_2^{y_2}$ as well as its decryption can be simulated (Protocol 4.2.4 and Section 2.5.3).

6.3 Scheme C: encrypted certificate scheme

The main scheme of the thesis is discussed in this section. In this scenario, $\mathcal{CA} = P$ has an encryption that needs to be certified. However, both $\mathcal{CA}$ and $\mathcal{U}$ are *not* allowed to learn the encrypted value. It turns out that we need $\mathcal{CA} = \mathcal{V}$. This drawback will be discussed in Section 8.1.1.

For simplicity, throughout the remainder of this chapter, an ElGamal encryption of x will be written as $[\![g^x]\!] = (g^r, g^x f^r)$ for some $r \in_R \mathbb{Z}_q$ (instead of $[\![x]\!]$). This is done (a) for stressing the construction in g, and (b) since the scheme also involves encryptions of the type $[\![h_0^x]\!] = (g^r, h_0^x f^r)$. We recall that the key generation setup for the cryptosystem is implicitly included in the key generation algorithm below. As we use ElGamal, we need the encrypted values to be from a restricted domain. We consider the following scenario: as a set of multiparty computation servers, $\mathcal{CA}$ outputs encryptions $c_i^* = [\![g^{x_i^*}]\!]$ with $x_i^* \in \{0,1\}$ for $i = 1,\ldots,l-1$, constituting the encrypted attribute list (since in the issuing protocol an l-th encryption will be added, for consistency we call the resulting certificate a certificate on l attributes). The scheme can be easily generalized to different domains. In case we have only one multiparty output c_1^*, and $x_2,\ldots,x_{l-1}$ are attributes known to $\mathcal{CA}$ (or any other generalization of this type), see Section 8.2.

Following Definition 5.1.2, the algorithms/protocols (keygen, issue, verify) are constructed in Sections 6.3.1-6.3.3, respectively. The security of the scheme is discussed in Section 6.4.

6.3.1 Key generation

On input of security parameter k, the certification authority obtains public key (q, g), with $q > 2^k$, and $(h_0, \hat{f}, (g_i)_{i=1}^{l}, (f_i)_{i=1}^{l-1})$, together with secret key $x_0, (y_i)_{i=1}^{l}, (\phi_i)_{i=1}^{l-1} \in_R \mathbb{Z}_q$ satisfying

$$h_0 = g^{x_0}, \qquad \hat{f} = f^{x_0}, \qquad \forall_{i=1}^{l}: \; g_i = g^{y_i}, \qquad \forall_{i=1}^{l-1}: \; f_i = g^{\phi_i}.$$

The encrypted attributes set consists of tuples of $l-1$ ElGamal bit encryptions. A certificate on an encrypted attribute list $p^* = (c_i^*)_{i=1}^{l-1} = \left([\![g^{x_i^*}]\!]\right)_{i=1}^{l-1}$ is a tuple $(h', (c_i')_{i=1}^{l}, \alpha_1, z', (z_i')_{i=1}^{l}, c_0', r_0')$ satisfying:

$$\begin{gathered} c_0' = \mathcal{H}([c_i', z_i', (c_i')^{r_0'}(z_i')^{-c_0'}]_{i=1}^{l}; h', z', g^{r_0'}h_0^{-c_0'}, (h')^{r_0'}(z')^{-c_0'}) \\ \text{and } ((c_1')^{y_1}\cdots(c_l')^{y_l}[\![h_0]\!])^{\alpha_1} \doteq [\![h']\!]. \end{gathered} \tag{6.2}$$

In light of Definition 5.1.2, $p = (h', (c_i')_{i=1}^{l})$ is the public key part with corresponding secret key part $s = \alpha_1$. The function inv is defined as[3]

$$\text{inv}(p) = \left(c_i'(1, f_i^{-1})\right)_{i=1}^{l-1}. \tag{6.3}$$

The element h' is required to be different from 1. If in the issuing it turns out that $h' = 1$, it is declared invalid and the protocol is restarted.

Remark 6.3.1. We elaborate on the 'transformation' from $c_i^* = [\![g^{x_i^*}]\!]$ to $c_i^*[\![f_i]\!] = [\![g^{x_i^*+\phi_i}]\!]$. Suppose the user is issued a certificate on c_1^* instead of $c_1^*[\![f_1]\!]$. In the issuing protocol, he needs to blind this encryption by setting $c_1' \leftarrow c_1^*[\![g^0]\!]$. However, a malicious user $\mathcal{U}'$ can set $c_1' \leftarrow [\![g^0]\!]$, and z_1' likewise such that the first equation in (6.2) is still satisfied. Then, it turns out that also the second equation in (6.2) is satisfied *if and only if* $x_1^* = 0$ (so if and only if $c_1^* = [\![1]\!]$). Consequently, if $\mathcal{U}'$ follows the verification protocol, the verifier will accept if and only if $x_1^* = 0$. This means that the user obtains an oracle for '$x_1^* \overset{?}{=} 0$'. For $c_1^*[\![f_1]\!]$ this attack fails with overwhelming probability.

[3] This essentially means that we have $c_i' \doteq c_i^*[\![f_i]\!]$ for $i = 1,\ldots,l-1$.

6.3.2 Certificate issuance

On input of encrypted attribute list $(c_i^*)_{i=1}^{l-1}$, the issuing protocol is given in Figure 6.4. In case of multiple issuing executions on the same encrypted attribute list, the values $(h, z, (c_i, z_i)_{i=1}^l)$ need to be computed only once. It is clear that the original protocol of Brands, depicted in Figure 5.1, is a simplification of Figure 6.4.

$$\begin{array}{rcl}
\mathcal{U} & & \mathcal{CA} \\
(\text{knows: } (c_i^*)_{i=1}^{l-1}) & & (\text{knows: } (c_i^*)_{i=1}^{l-1}) \\
 & & (r_i)_{i=1}^{l}, x_l \in_R \mathbb{Z}_q \\
 & & (c_i \leftarrow c_i^* \cdot (g^{r_i}, f_i f^{r_i}))_{i=1}^{l-1} \\
 & & c_l \leftarrow (g^{r_l}, g^{x_l} f^{r_l}) \\
 & & h \leftarrow D(c_1^{y_1} \cdots c_l^{y_l}) h_0 \\
 & & z \leftarrow h^{x_0}, \quad (z_i \leftarrow c_i^{x_0})_{i=1}^{l} \\
 & & w_0 \in_R \mathbb{Z}_q \\
 & & a_0 \leftarrow g^{w_0}, \quad b_0 \leftarrow h^{w_0} \\
 & \xleftarrow{h,z,(c_i,z_i)_{i=1}^l;a_0,b_0,\tilde{f},(e_{i0})_{i=1}^l} & \tilde{f} \leftarrow f^{w_0}, \quad (e_{i0} \leftarrow c_i^{w_0})_{i=1}^{l} \\
\alpha_1 \in_R \mathbb{Z}_q^*, \quad \alpha_2, \alpha_3 \in_R \mathbb{Z}_q & & \\
h' \leftarrow h^{\alpha_1}, \quad z' \leftarrow z^{\alpha_1} & & \\
a_0' \leftarrow h_0^{\alpha_2} g^{\alpha_3} a_0 & & \\
b_0' \leftarrow (z')^{\alpha_2} (h')^{\alpha_3} b_0^{\alpha_1} & & \\
\left(\begin{array}{r} \beta_i \in_R \mathbb{Z}_q, \quad c_i' \leftarrow c_i \cdot (g, f)^{\beta_i} \\ z_i' \leftarrow z_i \cdot (h_0, \hat{f})^{\beta_i} \\ e_{i0}' \leftarrow (z_i')^{\alpha_2} (c_i')^{\alpha_3} e_{i0} \cdot (a_0, \tilde{f})^{\beta_i} \end{array} \right)_{i=1}^{l} & & \\
c_0' \leftarrow \mathcal{H}([c_i', z_i', e_{i0}']_{i=1}^l; h', z', a_0', b_0') & & \\
c_0 \leftarrow c_0' + \alpha_2 \bmod q & & \\
 & \xrightarrow{c_0} & \\
 & \xleftarrow{r_0} & r_0 \leftarrow c_0 x_0 + w_0 \bmod q \\
a_0 \stackrel{?}{=} g^{r_0} h_0^{-c_0}, \quad b_0 \stackrel{?}{=} h^{r_0} z^{-c_0} & & \\
\tilde{f} \stackrel{?}{=} f^{r_0} \hat{f}^{-c_0}, \quad (e_{i0} \stackrel{?}{=} c_i^{r_0} z_i^{-c_0})_{i=1}^{l} & & \\
r_0' \leftarrow r_0 + \alpha_3 \bmod q & &
\end{array}$$

Figure 6.4: Issuing protocol for encrypted certification, ElGamal.

6.3.3 Certificate verification

If $\mathcal{U}$ owns a certificate $(h', (c_i')_{i=1}^l, \alpha_1, z', (z_i')_{i=1}^l, c_0', r_0')$ which he wants to show $\mathcal{V}$, he needs to prove that (6.2) holds. To this end, $\mathcal{U}$ sends all values except for α_1 to $\mathcal{V}$ (in line with Definition 5.1.2), and the participants execute the two-party protocol given in Figure 6.5. After the verification, $\mathcal{V}$ can obtain $\llbracket g^{x_i^*} \rrbracket$ by computing $c_i' \llbracket f_i \rrbracket^{-1}$ (for $i = 1, \ldots, l-1$).

Note that the protocol is very similar to a Σ-protocol for relation (similar to the verification protocol in Brands' scheme in Section 5.2.3, we write the equation as follows, rather than as $\llbracket h' \rrbracket \bumpeq ((c_1')^{y_1} \cdots (c_l')^{y_l} \llbracket h_0 \rrbracket)^{\alpha}$. This is merely done for consistency)

$$\{(h', (c_i')_{i=1}^l; \alpha_1) \mid \llbracket h' \rrbracket^{\alpha_1^{-1}} \bumpeq (c_1')^{y_1} \cdots (c_l')^{y_l} \llbracket h_0 \rrbracket \wedge \alpha_1 \neq 0\}.$$

However, now the verifier needs secret input as well, namely the values $y_1, \ldots, y_l, \lambda$, and therefore the protocol cannot be seen as a Σ-protocol. A discussion on this restriction can be found in Section 8.1.1. As a consequence, this time the scheme includes a fourth round. In the protocol of Section 5.2.3 this round is implicit, but now $\mathcal{V}$ has private knowledge, so we also need to prove that $\mathcal{U}$ cannot learn anything about those values. Also, since $\mathcal{V}$ needs private values, this protocol cannot be made non-interactive, unlike Brands' verification protocol. Yet, the number of rounds can be reduced by taking $c \leftarrow \mathcal{H}(\underline{a})$ in a similar way as in Section 5.2.3. Again, as the equality $c'_0 = \mathcal{H}([c'_i, z'_i, (c'_i)^{r'_0}(z'_i)^{-c'_0}]_{i=1}^{l};\ h', z', g^{r'_0}h_0^{-c'_0}, (h')^{r'_0}(z')^{-c'_0})$ can be easily checked publicly, as of now it is assumed to hold. Moreover, the public part of the certificate, $(h', z', (c'_i, z'_i)_{i=1}^{l}, c'_0, r'_0)$, is assumed to be sent to $\mathcal{V}$ before the verification starts.

$$
\begin{array}{rcl}
\mathcal{U} & & \mathcal{V} \\
\text{(knows: } h', z', (c'_i, z'_i)_{i=1}^{l}, c'_0, r'_0; \alpha_1) & & \text{(knows: } y_1, \ldots, y_l, \lambda) \\
u \in_R \mathbb{Z}_q, \quad a \leftarrow (h')^u & \xrightarrow{a;h',z',(c'_i,z'_i)_{i=1}^{l},c'_0,r'_0} & \\
 & \xleftarrow{\quad c \quad} & c \in_R \mathbb{Z}_q \\
r \leftarrow u + c\alpha_1^{-1} \bmod q & \xrightarrow{\quad r \quad} & \\
 & & b_1 \leftarrow \Big[c'_0 \overset{?}{=} \mathcal{H}([c'_i, z'_i, (c'_i)^{r'_0}(z'_i)^{-c'_0}]_{i=1}^{l}; \\
 & & \qquad h', z', g^{r'_0}h_0^{-c'_0}, (h')^{r'_0}(z')^{-c'_0})\Big] \\
 & & b_2 \leftarrow \Big[[\![h']\!]^r \overset{?}{\doteqdot} [\![a]\!]((c'_1)^{y_1} \cdots (c'_l)^{y_l}[\![h_0]\!])^c\Big] \\
 & \xleftarrow{\quad b \quad} & b \leftarrow b_1 \wedge b_2
\end{array}
$$

Figure 6.5: Verification protocol for encrypted certification, ElGamal.

6.4 Scheme C: security analysis

In the construction of Section 6.3, for each issuing there is an encrypted attribute list $p^* = (c_i^*)_{i=1}^{l-1}$, and for the public part of the issued certificate, $p = (h', (c'_i)_{i=1}^{l})$, we have $\text{inv}(p) \doteqdot p^*$ with the function inv defined in (6.3). More generally, the algorithm and protocols in Section 6.3 satisfy the basic requirements of Definition 5.1.2. We prove that the scheme is secure with respect to this definition. We recall from Section 3.4 that we have $P = \mathcal{CA} = \mathcal{V}$, and that we therefore see $\mathcal{CA} = \mathcal{V}$ as *one* participant. This is a natural way to force honest-but-curious behavior, and therefore this is reasonable to assume (see also Sections 3.4 and 8.1.1). In particular, due to Proposition 4.2.7 the plaintext equality test in the verification protocol can be simulated in case $\mathcal{V}$ is a set of multiparty computation servers. We refer the reader to Remark 6.4.12 for a broader discussion of this generalization.

The security proof is divided into five parts, corresponding to the properties stated in Definition 5.1.2, and proven in Sections 6.4.1-6.4.5, respectively. We will consider the five properties for any probabilistic key generation execution $(pk, sk) \leftarrow \text{keygen}(k)$. We assume that this algorithm execution is done properly, i.e. that the system parameters are correctly formed.

For proving blinding-invariance unforgeability and a part of the security of the verification protocol, we need the assumption below. In fact, this assumption is similar to the analogue of Brands' assumption for proving his scheme blinding-invariance secure (Assumption 5.2.1). It is unclear how to reduce this assumption to Brands' assumption, or to prove it differently.

Assumption 6.4.1. If $\mathcal{U}'$ produces, after $K \geq 0$ arbitrarily interleaving executions of the protocol in Figure 6.4 on $\left(c^*_{ji}\right)_{i=1}^{l-1}$ $(j = 1, \ldots, K)$ a tuple $(h', (c'_i)_{i=1}^{l}, \alpha_1, z', (z'_i)_{i=1}^{l}, c'_0, r'_0)$, then either this tuple does not satisfy (6.2), or with overwhelming probability there exists a $j \in \{1, \ldots, K\}$ such that

$$\mathcal{U}' \text{ knows values } (\beta_i)_{i=1}^{l} \text{ such that } (c'_i)_{i=1}^{l} = \left(c_{ji}(g, f)^{\beta_i}\right)_{i=1}^{l}, \tag{6.4}$$

where $(c_{ji})_{i=1}^{l}$ is the list of encryptions coming from the first round of the j-th issuing execution. More formally, there exists a p.p.t. extractor $\mathcal{E}$ that may use $\mathcal{U}'$ as a subroutine and also outputs a tuple $(h', (c'_i)_{i=1}^{l}, \alpha_1, z', (z'_i)_{i=1}^{l}, c'_0, r'_0)$, but additionally outputs the values $(h_j, (c_{ji})_{i=1}^{l})_{j=1}^{K}$ on which the user is issued certificates, and a value $\tau \in \{0, \ldots, K\}$: $\tau = 0$ meaning that (6.4) is not satisfied for any j, and $\tau \neq 0$ meaning that it is satisfied for $j = \tau$, in which case the extractor also outputs a tuple $(\beta_i)_{i=1}^{l}$ satisfying (6.4).

In Appendix A we heuristically argue why this assumption should hold.

6.4.1 Completeness

Proposition 6.4.2. *If both $\mathcal{U}$ and $\mathcal{CA}$ follow the protocol, then for any encrypted attribute list $p^* = (c^*_i)_{i=1}^{l-1}$, the resulting certificate of the issuing execution will be accepted upon verification.*

Proof. On input of $(c^*_i)_{i=1}^{l-1}$, the issuing execution results in a tuple $(h', (c'_i)_{i=1}^{l}, \alpha_1, z', (z'_i)_{i=1}^{l}, c'_0, r'_0)$. We will show that this certificate satisfies (6.2), since this would indeed result in acceptance in the verification execution by Proposition 6.4.8. For the second equation in (6.2), $\llbracket h' \rrbracket \bumpeq ((c'_1)^{y_1} \cdots (c'_l)^{y_l} \llbracket h_0 \rrbracket)^{\alpha_1}$, notice that in the beginning of the issuing execution $\mathcal{U}$ obtained $(h, (c_i)_{i=1}^{l})$ satisfying $\llbracket h \rrbracket \bumpeq c_1^{y_1} \cdots c_l^{y_l} \llbracket h_0 \rrbracket$, and $\mathcal{U}$ blinded each encryption as $c'_i \leftarrow c_i \cdot (g, f)^{\beta_i}$ for $\beta_i \in_R \mathbb{Z}_q$. As moreover $h' = h^{\alpha_1}$ by construction, we obtain:

$$((c'_1)^{y_1} \cdots (c'_l)^{y_l} \llbracket h_0 \rrbracket)^{\alpha_1} \bumpeq (c_1^{y_1} \cdots c_l^{y_l} \llbracket h_0 \rrbracket)^{\alpha_1} \bumpeq \llbracket h \rrbracket^{\alpha_1} \bumpeq \llbracket h' \rrbracket.$$

Remains to prove that the first equation in (6.2) holds, $c'_0 = \mathcal{H}([c'_i, z'_i, (c'_i)^{r'_0}(z'_i)^{-c'_0}]_{i=1}^{l}; h', z', g^{r'_0}h_0^{-c'_0}, (h')^{r'_0}(z')^{-c'_0})$. But in the issuing protocol execution $\mathcal{U}$ puts $c'_0 = \mathcal{H}([c'_i, z'_i, e'_{i0}]_{i=1}^{l}; h', z', a'_0, b'_0)$, so the claim follows if the following equations are satisfied:

$$\forall_{i=1}^{l}: \; e'_{i0} = (c'_i)^{r'_0}(z'_i)^{-c'_0}, \qquad a'_0 = g^{r'_0}h_0^{-c'_0}, \qquad b'_0 = (h')^{r'_0}(z')^{-c'_0}. \tag{6.5}$$

But, simultaneously for a'_0, b'_0:

$$\begin{array}{lll}
\big(g^{r'_0}h_0^{-c'_0} & , (h')^{r'_0}(z')^{-c'_0}\big) & \\
=\big(g^{\alpha_3}g^{r_0}h_0^{-c'_0} & , (h')^{\alpha_3}(h')^{r_0}(z')^{-c'_0}\big) & \{r'_0 \equiv r_0 + \alpha_3 \bmod q\} \\
=\big(g^{\alpha_3}h_0^{\alpha_2}g^{r_0}h_0^{-c_0} & , (h')^{\alpha_3}(z')^{\alpha_2}(h')^{r_0}(z')^{-c_0}\big) & \{c'_0 \equiv c_0 - \alpha_2 \bmod q\} \\
=\big(g^{\alpha_3}h_0^{\alpha_2}g^{w_0}(g^{-x_0}h_0)^{-c_0} & , (h')^{\alpha_3}(z')^{\alpha_2}(h')^{w_0}((h')^{-x_0}z')^{-c_0}\big) & \{r_0 \equiv w_0 + c_0x_0 \bmod q\} \\
=\big(g^{\alpha_3}h_0^{\alpha_2}g^{w_0} & , (h')^{\alpha_3}(z')^{\alpha_2}h^{w_0\alpha_1}\big) & \{h_0 = g^{x_0}, h' = h^{\alpha_1}, z' = (h')^{x_0}\} \\
=\big(g^{\alpha_3}h_0^{\alpha_2}a_0 & , (h')^{\alpha_3}(z')^{\alpha_2}b_0^{\alpha_1}\big) & \{\text{setup } w_0\} \\
=\big(a'_0 & , b'_0\big), &
\end{array}$$

and for e'_{i0}:

$$\begin{aligned}
(c'_i)^{r'_0}(z'_i)^{-c'_0} &= (c'_i)^{\alpha_3}(z'_i)^{\alpha_2}(c'_i)^{r_0}(z'_i)^{-c_0} && \{\text{setup of } r'_0, c'_0\}\\
&= (c'_i)^{\alpha_3}(z'_i)^{\alpha_2}(c'_i)^{w_0}((c'_i)^{-x_0}z'_i)^{-c_0} && \{r_0 \equiv w_0 + c_0x_0 \bmod q\}\\
&= (c'_i)^{\alpha_3}(z'_i)^{\alpha_2}(c'_i)^{w_0}((c'_i)^{-x_0}z_i(g^{x_0}, f^{x_0})^{\beta_i})^{-c_0} && \{z'_i = z_i(g^{x_0}, f^{x_0})^{\beta_i}\}\\
&= (c'_i)^{\alpha_3}(z'_i)^{\alpha_2}(c_i)^{w_0}(g,f)^{\beta_i w_0}(c_i^{-x_0}z_i)^{-c_0} && \{c'_i = c_i(g,f)^{\beta_i}\}\\
&= (c'_i)^{\alpha_3}(z'_i)^{\alpha_2}(c_i)^{w_0}(g,f)^{\beta_i w_0} && \{z_i = c_i^{x_0}\}\\
&= (c'_i)^{\alpha_3}(z'_i)^{\alpha_2}e_{i0}(a_0,\tilde{f})^{\beta_i} && \{\text{setup } w_0\}\\
&= e'_{i0}.
\end{aligned}$$

□

6.4.2 Privacy for $\mathcal{U}$

Proposition 6.4.3. *For any pair of encrypted attribute lists p_0^*, p_1^*, if $\mathcal{U}$ and $\mathcal{CA}'$ engaged in the issuing execution for both lists, obtaining certificates $(p, s, \sigma(p))_0, (p, s, \sigma(p))_1$, then it is hard for passively malicious p.p.t. $\mathcal{CA}'$ to, given $(p, \sigma(p))_b$ and $(p, \sigma(p))_{1-b}$ with $b \in_R \{0,1\}$, guess b correctly.*

Proof. The game played by $\mathcal{CA}'$ and $\mathcal{U}$ is the following: given any two different encrypted attribute lists p_0^*, p_1^*, $\mathcal{CA}'$ and $\mathcal{U}$ engage in an issuing execution for p_j^* $(j = 0, 1)$, $\mathcal{U}$ takes $b \in_R \{0,1\}$ and sends the public parts of the b-th and $(1-b)$-th certificate to $\mathcal{CA}'$ (in that order). $\mathcal{CA}'$ wins if he guesses b correctly. Denote by $Pr(A)$ the success probability of $\mathcal{CA}'$ in this game. We slightly change this game, obtaining game B. Now, in each issuing execution $\mathcal{U}$ sets for each $i = 1, \ldots, l$:

$$c'_i \leftarrow (g,f)^{\beta_i}, \qquad z'_i \leftarrow (h_0, \hat{f})^{\beta_i}, \qquad e'_{i0} \leftarrow (z'_i)^{\alpha_2}(c'_i)^{\alpha_3}(a_0, \tilde{f})^{\beta_i}, \tag{6.6}$$

and executes the remainder as is. (Note that the resulting tuple does not yield a valid certificate as $((c'_1)^{y_1}\cdots(c'_l)^{y_l}[\![h_0]\!])^{\alpha_1} \doteq [\![h']\!]$ need not be satisfied. However, as $\mathcal{CA}'$ does not know the decryption key, he will not notice.) Denote $\mathcal{CA}'$'s success probability in the new game by $Pr(B)$. Now, the only difference between the games is in the encryptions, and as $\mathcal{CA}'$ is p.p.t. and does not have the decryption key, if $\mathcal{CA}'$ is able to distinguish between the two games, he is able to distinguish between the constructions of one of the $6l$ encryptions. Hence, the success probabilities in the different games are of negligible difference by the semantic security of the cryptosystem, Section 2.5.3. Formally, there exists a negligible $\varepsilon(k)$ such that

$$|Pr(A) - Pr(B)| < \varepsilon(k). \tag{6.7}$$

We consider the success probability of $\mathcal{CA}'$ in game B. We will first prove that for any public part of a certificate, and any view on an issuing execution by $\mathcal{CA}'$, there is exactly *one* possible secret random tuple $\mathcal{U}$ could have chosen. In particular this means that from $\mathcal{CA}'$'s point of view, $(p, \sigma(p))_b$ could have come from the 0-th or 1-th issuing execution with equal probability, and similar for $(p, \sigma(p))_{1-b}$. Then, as $\mathcal{U}$ takes his values uniformly at random, $\mathcal{CA}'$ can only succeed in guessing b correctly with probability $\frac{1}{2}$. Hence $Pr(B) = \frac{1}{2}$, which by (6.7) implies that the success probability in the original game is upper bounded by $\frac{1}{2} + \varepsilon(k)$ for negligible $\varepsilon(k)$.
So we prove that for any public part of a certificate, $(h', z', (c'_i, z'_i)_{i=1}^l, c'_0, r'_0)$, and all values a malicious $\mathcal{CA}'$ sees during the issuance execution of a certificate, $(h, z, (c_i, z_i)_{i=1}^l)$ and $(a_0, b_0, \tilde{f}, (e_{i0})_{i=1}^l, c_0, r_0)$ satisfying (as $\mathcal{U}$ accepted)

$$\begin{aligned}
a_0 &= g^{r_0}h_0^{-c_0}, & b_0 &= h^{r_0}z^{-c_0},\\
\tilde{f} &= f^{r_0}\hat{f}^{-c_0}, & \forall_{i=1}^l :\ e_{i0} &= c_i^{r_0}z_i^{-c_0},
\end{aligned} \tag{6.8}$$

there exists exactly *one* possible combination of random values $\alpha_1, \alpha_2, \alpha_3, (\beta_i)_{i=1}^l$ that $\mathcal{U}$ could have chosen to end up with that certificate. The values α_1, α_2 and α_3 are determined by (h, h'), (c_0, c'_0) and (r_0, r'_0):

$$\alpha_1 = \log_h h', \qquad \alpha_2 = c_0 - c'_0 \bmod q, \qquad \alpha_3 = r'_0 - r_0 \bmod q,$$

and for each i, β_i is determined by c_i' as

$$\beta_i = \log_g (c_i')_1 = \log_f (c_i')_2.$$

Notice that we do not require that these random values can be computed efficiently, we just need them to be uniquely determined. Remains to prove that this choice satisfies $c_0' = \mathcal{H}([c_i', z_i', e_{i0}']_{i=1}^l; h', z', a_0', b_0')$. But the issued certificate satisfies $c_0' = \mathcal{H}([c_i', z_i', (c_i')^{r_0'}(z_i')^{-c_0'}]_{i=1}^l; h', z', g^{r_0'}h_0^{-c_0'}, (h')^{r_0'}(z')^{-c_0'})$, from which the desired equality follows if (6.5) is satisfied. But:

$$\begin{aligned}
g^{r_0'}h_0^{-c_0'} &= g^{\alpha_3}h_0^{\alpha_2}g^{r_0}h_0^{-c_0} && \{\text{setup } r_0', c_0'\}\\
&= g^{\alpha_3}h_0^{\alpha_2}a_0 && \{\text{equation (6.8)}\}\\
&= a_0',\\
(h')^{r_0'}(z')^{-c_0'} &= (h')^{\alpha_3}(z')^{\alpha_2}(h')^{r_0}(z')^{-c_0} && \{\text{setup } r_0', c_0'\}\\
&= (h')^{\alpha_3}(z')^{\alpha_2}\left(h^{r_0}z^{-c_0}\right)^{\alpha_1} && \{\text{setup } h', z'\}\\
&= (h')^{\alpha_3}(z')^{\alpha_2}b_0^{\alpha_1} && \{\text{equation (6.8)}\}\\
&= b_0',\\
(c_i')^{r_0'}(z_i')^{-c_0'} &= (c_i')^{\alpha_3}(z_i')^{\alpha_2}(c_i')^{r_0}(z_i')^{-c_0} && \{\text{setup } r_0', c_0'\}\\
&= (c_i')^{\alpha_3}(z_i')^{\alpha_2}(g^{r_0}h_0^{-c_0}, f^{r_0}\hat{f}^{-c_0})^{\beta_i} && \{\text{equation (6.6)}\}\\
&= (c_i')^{\alpha_3}(z_i')^{\alpha_2}(a_0, \tilde{f})^{\beta_i} && \{\text{equation (6.8)}\}\\
&= e_{i0}' && \{\text{equation (6.6)}\}. \qquad \square
\end{aligned}$$

Remark 6.4.4. For the proof to be correct, $\mathcal{CA}'$ may only be passively malicious and work in probabilistic polynomial time, simply because different issuing executions might involve different encryptions. However, if the two encrypted attribute lists encrypt the same values, so $p_0^* \doteq p_1^*$, then the changeover to game B is unnecessary. In particular, the issuing executions become unlinkable in an unconditioned way. This is relevant in case the certification authority issues many certificates on the same attribute list.

6.4.3 One-more unforgeability

The proof of one-more unforgeability relies on tightly reducing the certificates to Chaum-Pedersen signatures. We recall from Remark 2.7.2 that these are tuples (ξ, m, z, c', r') satisfying $c' = \mathcal{H}(\xi, m, z, g^{r'}h_{CP}^{-c'}, m^{r'}z^{-c'})$, where ξ is an arbitrary bit string.

Proposition 6.4.5. *Under the assumption that the blind Chaum-Pedersen signature scheme is secure against one-more forgeries, it is impossible for a user $\mathcal{U}'$ to, after $K \geq 0$ arbitrarily interleaving executions of Figure 6.4 on encrypted attribute lists $p_j^* = (c_{ji}^*)_{i=1}^{l-1}$ ($j = 1, \ldots, K$), with non-negligible probability output $K+1$ different certificates satisfying (6.2).*

Proof. Suppose it is possible, so after K executions of the protocol of Figure 6.4, on $(c_{ji}^*)_{i=1}^{l-1}$ for $j = 1, \ldots, K$, $\mathcal{U}'$ can output $K+1$ different certificates $(h', (c_i')_{i=1}^l, \alpha_1, z', (z_i')_{i=1}^l, c_0', r_0')$ satisfying (6.2), with non-negligible probability. We construct an interactive polynomial time forger $\mathcal{F}$ that is issued K Chaum-Pedersen signatures by a Chaum-Pedersen signer $\mathcal{S}$, possibly on different messages m for each execution $j = 1, \ldots, K$, and uses $\mathcal{U}'$ to output $K+1$ different Chaum-Pedersen signatures. By assumption that is impossible, and hence we obtain a contradiction.
Let $G = \langle g \rangle, h_{CP}$ be the system parameters of the Chaum-Pedersen signature scheme, for which $\mathcal{S}$ knows $x = \log_g h_{CP}$. Now $\mathcal{F}$ simulates the certificate issuer for Figure 6.4 as follows:

1. *Initialization*: For the encryption scheme, $\mathcal{F}$ takes secret key $\lambda \in_R \mathbb{Z}_q$ and publishes $f = g^\lambda$. Furthermore, $\mathcal{F}$ inherits $\mathcal{S}$'s system parameters, and takes moreover $(y_i)_{i=1}^l, (\phi_i)_{i=1}^{l-1} \in_R \mathbb{Z}_q$ and publishes $h_0 = h_{CP}$, $f_i = g^{\phi_i}$ $(i = 1, \ldots, l-1)$, $g_i = g^{y_i}$ $(i = 1, \ldots, l)$, and $\hat{f} = h_0^\lambda$;

2. *Issuing*: For each of the K issuing protocol executions, $\mathcal{F}$ operates as follows (see Figure 6.6)[4]:
   i. *Commitment part 1*: $\mathcal{F}$ takes $x_i^* \in_R \{0,1\}$ and publishes encryptions $c_i^* = [\![g^{x_i^*}]\!]$ (for $i = 1, \ldots, l-1$). He takes $(r_i)_{i=1}^{l}, x_l \in_R \mathbb{Z}_q$, sets $(x_i \leftarrow x_i^* + \phi_i \bmod q)_{i=1}^{l-1}$, sets
   $$c_i \leftarrow (g^{r_i}, g^{x_i} f^{r_i}) \text{ and } z_i \leftarrow (h_0^{r_i}, h_0^{x_i+\lambda r_i}), \text{ for each } i = 1, \ldots, l,$$
   and $h \leftarrow g_1^{x_1} \cdots g_l^{x_l} h_0$. For the setup of z, $\mathcal{F}$ sends $\tilde{m}_0 \leftarrow h$ to $\mathcal{S}$, in order to obtain $\tilde{z}_0$. The forger sends $z \leftarrow \tilde{z}_0$ to $\mathcal{U}'$;
   ii. *Commitment part 2*: $\mathcal{F}$ receives $\tilde{a}_0, \tilde{b}_0$ from $\mathcal{S}$, he sets $(a_0, b_0) \leftarrow (\tilde{a}_0, \tilde{b}_0)$ and $\tilde{f} \leftarrow a_0^{\lambda}$, and for each $i = 1, \ldots, l$ he takes $e_{i0} \leftarrow (a_0^{r_i}, a_0^{x_i+\lambda r_i})$. He sends $(a_0, b_0, \tilde{f}, (e_{i0})_{i=1}^{l})$ to $\mathcal{U}'$;
   iii. *Challenge*: $\mathcal{F}$ receives c_0 from $\mathcal{U}'$ and sends $\tilde{c}_0 \leftarrow c_0$ to $\mathcal{S}$;
   iv. *Response*: $\mathcal{F}$ receives $\tilde{r}_0$ from $\mathcal{S}$ and sends $r_0 \leftarrow \tilde{r}_0$ to $\mathcal{U}'$;

3. *Signature forging*: Now $\mathcal{U}'$ outputs, with non-negligible probability, $K+1$ distinct certificates $(h', (c_i')_{i=1}^{l}, \alpha_1, z', (z_i')_{i=1}^{l}, c_0', r_0')$. For each of these certificates $\mathcal{F}$ computes Chaum-Pedersen forgery
   $$(\bar{\xi}, \bar{z}, \bar{c}, \bar{r}, \bar{m}) \leftarrow ([c_i', z_i', (c_i')^{r_0'}(z_i')^{-c_0'}]_{i=1}^{l}, z', c_0', r_0', h'), \tag{6.9}$$
   and he outputs these $K+1$ Chaum-Pedersen signatures.

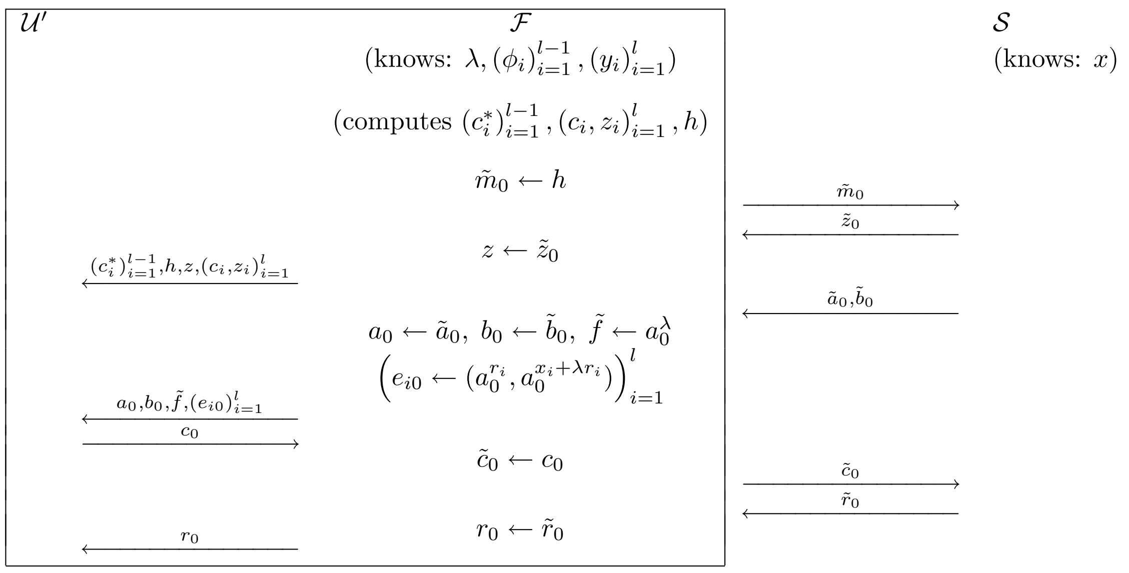
$\mathcal{U}'$
$\mathcal{F}$
$\mathcal{S}$
(knows: $\lambda, (\phi_i)_{i=1}^{l-1}, (y_i)_{i=1}^{l}$)
(knows: x)
(computes $(c_i^*)_{i=1}^{l-1}, (c_i, z_i)_{i=1}^{l}, h$)
$\tilde{m}_0 \leftarrow h$
$\tilde{m}_0$
$\tilde{z}_0$
$z \leftarrow \tilde{z}_0$
$(c_i^*)_{i=1}^{l-1}, h, z, (c_i, z_i)_{i=1}^{l}$
$\tilde{a}_0, \tilde{b}_0$
$a_0 \leftarrow \tilde{a}_0,\ b_0 \leftarrow \tilde{b}_0,\ \tilde{f} \leftarrow a_0^{\lambda}$
$\left(e_{i0} \leftarrow (a_0^{r_i}, a_0^{x_i+\lambda r_i})\right)_{i=1}^{l}$
$a_0, b_0, \tilde{f}, (e_{i0})_{i=1}^{l}$
c_0
$\tilde{c}_0 \leftarrow c_0$
$\tilde{c}_0$
$\tilde{r}_0$
$r_0 \leftarrow \tilde{r}_0$
r_0

Figure 6.6: A visualization of reducing a forgery to a Chaum-Pedersen forgery. The outlined box denotes the adversarial side.

Remains to prove that this reduction works.
Phase 1. The distribution of the system parameters is perfectly indistinguishable from the real system parameter distribution. This is since $h_0 = h_{CP}$, where h_{CP} is taken by $\mathcal{CA}$ to be $\in_R G$, and also $(g_i)_{i=1}^{l}, (f_i)_{i=1}^{l-1}, f \in_R G$, just as in the real protocol. Moreover, we have $\hat{f} = h_0^{\lambda} = g^{x\lambda} = f^x$.
Phase 2. We have to prove that the parameters sent in phase 2i follow the correct distribution, as explained in Section 6.3.2. For the c_i's this is clear. (Note that this also works if the x_i^*'s come from some multiparty computation, as $\mathcal{F}$ has the decryption key.) The equation $z = h^x$ holds since by the Chaum-Pedersen issuing protocol we have $\tilde{z}_0 = \tilde{m}_0^x$. Finally, also the z_i's are correct as
$$z_i = (h_0^{r_i}, h_0^{x_i+\lambda r_i}) = (g^{x r_i}, g^{x(x_i+\lambda r_i)}) = c_i^x.$$

[4]For ease of presentation, the first round of the original protocol in Figure 6.4, the commitment part, is separated into two phases i and ii. That is, firstly $(h, z, (c_i, z_i)_{i=1}^{l})$ is sent to $\mathcal{U}'$, and then $(a_0, b_0, \tilde{f}, (e_{i0})_{i=1}^{l})$.

For phase 2ii, the values $(a_0, b_0, \tilde{f}, (e_{i0})_{i=1}^l)$ need to be of the form

$$(g^{w_0}, h^{w_0}, f^{w_0}, (c_i^{w_0})_{i=1}^l)$$

for some $w_0 \in_R \mathbb{Z}_q$. But by the Chaum-Pedersen scheme specifications, $(\tilde{a}_0, \tilde{b}_0) = (g^{\tilde{w}}, \tilde{m}_0^{\tilde{w}}) = (g^{\tilde{w}}, h^{\tilde{w}})$ for some $\tilde{w} \in_R \mathbb{Z}_q$. Now, following phase 2ii, we have $\tilde{f} = a_0^\lambda = g^{\tilde{w}\lambda} = f^{\tilde{w}}$. Also the e_{i0}'s are correct as

$$e_{i0} = (a_0^{r_i}, a_0^{x_i+\lambda r_i}) = (g^{\tilde{w}r_i}, g^{\tilde{w}(x_i+\lambda r_i)}) = c_i^{\tilde{w}}.$$

For phase 2iv, notice that $\tilde{r}_0$ satisfies $g^{\tilde{r}_0} h_{CP}^{-\tilde{c}_0} = \tilde{a}_0$ and $\tilde{m}_0^{\tilde{r}_0} \tilde{z}_0^{-\tilde{c}_0} = \tilde{b}_0$ by the construction of the Chaum-Pedersen signature protocol, and that therefore also

$$\begin{aligned}
g^{r_0} h_0^{-c_0} &= g^{\tilde{r}_0} h_{CP}^{-\tilde{c}_0} = \tilde{a}_0 = a_0, \\
h^{r_0} z^{-c_0} &= \tilde{m}_0^{\tilde{r}_0} \tilde{z}_0^{-\tilde{c}_0} = \tilde{b}_0 = b_0, \\
f^{r_0} \hat{f}^{-c_0} &= (g^{r_0} h_0^{-c_0})^\lambda \overset{*}{=} a_0^\lambda \overset{**}{=} \tilde{f}, \\
c_i^{r_0} z_i^{-c_0} &= (g^{r_i}, g^{x_i+\lambda r_i})^{r_0} (h_0^{r_i}, h_0^{x_i+\lambda r_i})^{-c_0} \\
&= ((g^{r_0} h_0^{-c_0})^{r_i}, (g^{r_0} h_0^{-c_0})^{x_i+\lambda r_i}) \\
&\overset{*}{=} (a_0^{r_i}, a_0^{x_i+\lambda r_i}) \overset{**}{=} e_{i0},
\end{aligned}$$

where $\overset{*}{=}$ holds since we have $g^{r_0} h_0^{-c_0} = a_0$ (first line), and $\overset{**}{=}$ holds due to phase 2ii. So for $\mathcal{U}'$ it looks like he is executing the protocol with a real issuer, and therefore he can output $K+1$ signatures, with non-negligible probability.

Phase 3. For the $K+1$ tuples, we have to show that (a) these are indeed Chaum-Pedersen signatures and (b) all these signatures are distinct.

(a). Let $(h', (c_i')_{i=1}^l, \alpha_1, z', (z_i')_{i=1}^l, c_0', r_0')$ be a certificate provided by $\mathcal{U}'$ and let $\mathcal{F}$ set $(\overline{\xi}, \overline{z}, \overline{c}, \overline{r}, \overline{m})$ as in (6.9). We have to prove that $\overline{c} = \mathcal{H}(\overline{\xi}, \overline{m}, \overline{z}, g^{\overline{r}}(h_{CP})^{-\overline{c}}, (\overline{m})^{\overline{r}}(\overline{z})^{-\overline{c}})$, but this holds as

$$\begin{aligned}
\overline{c} &= c_0' && \{\text{equation (6.9)}\} \\
&= \mathcal{H}([c_i', z_i', (c_i')^{r_0'}(z_i')^{-c_0'}]_{i=1}^l, h', z', g^{r_0'} h_0^{-c_0'}, (h')^{r_0'}(z')^{-c_0'}) && \{\text{equation (6.2)}\} \\
&= \mathcal{H}(\overline{\xi}, \overline{m}, \overline{z}, g^{\overline{r}}(h_{CP})^{-\overline{c}}, (\overline{m})^{\overline{r}}(\overline{z})^{-\overline{c}}) && \{\text{equation (6.9)}\}.
\end{aligned}$$

(b). Suppose that two distinct certificates $\sigma_1 = (h', (c_i')_{i=1}^l, \alpha_1, z', (z_i')_{i=1}^l, c_0', r_0')$ and $\sigma_2 = (\hat{h}', (\hat{c}_i')_{i=1}^l, \hat{\alpha}_1, \hat{z}', (\hat{z}_i')_{i=1}^l, \hat{c}_0', \hat{r}_0')$, both satisfying (6.2), give rise to the same Chaum-Pedersen signature, that means by (6.9):

$$(h', (c_i')_{i=1}^l, z', (z_i')_{i=1}^l, c_0', r_0') = (\hat{h}', (\hat{c}_i')_{i=1}^l, \hat{z}', (\hat{z}_i')_{i=1}^l, \hat{c}_0', \hat{r}_0'). \tag{6.10}$$

As σ_1, σ_2 both satisfy (6.2), we obtain:

$$\begin{aligned}
((\hat{c}_1')^{y_1} \cdots (\hat{c}_l')^{y_l} [\![h_0]\!])^{\hat{\alpha}_1} &\doteq [\![\hat{h}']\!] && \{\text{equation (6.2)}\} \\
&\doteq [\![h']\!] && \{\text{equation (6.10)}\} \\
&\doteq ((c_1')^{y_1} \cdots (c_l')^{y_l} [\![h_0]\!])^{\alpha_1} && \{\text{equation (6.2)}\} \\
&\doteq ((\hat{c}_1')^{y_1} \cdots (\hat{c}_l')^{y_l} [\![h_0]\!])^{\alpha_1} && \{\text{equation (6.10)}\}.
\end{aligned}$$

But as $h' \neq 1$ by (6.2), this implies that $\alpha_1 = \hat{\alpha}_1$, contradicting the fact that $\sigma_1 \neq \sigma_2$. Hence we obtained $K+1$ different Chaum-Pedersen signatures after only K Chaum-Pedersen issuing executions, which is impossible by assumption. □

Remark 6.4.6. The proposition essentially reduces one-more forgeries for Figure 6.4 to one-more forgeries for the Chaum-Pedersen scheme, and it is thus at least as secure as the Chaum-Pedersen scheme. We stress that this scheme requires $K \leq (\lg k)^\alpha$ in order to guarantee one-more unforgeability, due to the security consideration in Section 2.7.1.

6.4.4 Blinding-invariance unforgeability

The proof of blinding-invariance unforgeability relies on Assumption 6.4.1. However, this assumption is not sufficient: it essentially says that a malicious user cannot with non-negligible probability output any certificate on a *different* plaintext attribute list than he is issued certificates on, while blinding-invariance unforgeability more generally requires that for any attribute list the user cannot output more certificates on it than he is issued. So similar to Brands' scheme [Bra99, Ass. 4.4.5], blinding-invariance unforgeability of our scheme is slightly more general than the corresponding assumption. Therefore, it is stated without proof.

Proposition 6.4.7. *If $\mathcal{U}'$ comes, after $K \geq 0$ arbitrarily interleaving executions of Figure 6.4 on encrypted attribute lists $p_j^* = (c_{ji}^*)_{i=1}^{l-1}$ $(j = 1, \ldots, K)$, with L different certificates satisfying (6.2), then with overwhelming probability for each such certificate $(h', (c_i')_{i=1}^l, \alpha_1, z', (z_i')_{i=1}^l, c_0', r_0')$ there exists a j such that $\mathrm{inv}(h', (c_i')_{i=1}^l) \doteq p_j^*$, and moreover for each such $\mathrm{inv}(h', (c_i')_{i=1}^l)$ the number of certificates encrypting the same attribute list is at most the number of p_j^*'s encrypting that list.*

6.4.5 Secure verification

We need to prove that the verification protocol depicted in Figure 6.5 is a secure two-party protocol for proving knowledge of α_1 such that $(h', (c_i')_{i=1}^l, \alpha_1, z', (z_i')_{i=1}^l, c_0', r_0')$ is a valid certificate. Let us first make some simplifications, due to previous considerations. Firstly, recall from Section 6.3.3 that the equality $c_0' \overset{?}{=} \mathcal{H}(\cdot)$ can be assumed to hold without loss of generality. Secondly, in the introduction of Section 6.4 we concluded that without loss of generality $\mathcal{V}$ can only be passively corrupted. For a more extended discussion on this generalization the reader is referred to Remark 6.4.12. These two observations simplify the verification protocol of Figure 6.5 to Figure 6.7, where $\mathcal{U}$ can be an active attacker, but $\mathcal{V}$ can only be passive.

$$\begin{array}{rcl}
\mathcal{U} & & \mathcal{V} \\
(\text{knows: } h', (c_i')_{i=1}^l; \alpha) & & (\text{knows: } h', (c_i')_{i=1}^l; (y_i)_{i=1}^l, \lambda) \\
u \in_R \mathbb{Z}_q, \quad a \leftarrow (h')^u & \xrightarrow{\quad a \quad} & \\
 & \xleftarrow{\quad c \quad} & c \in_R \mathbb{Z}_q \\
r \leftarrow u + c\alpha^{-1} \bmod q & \xrightarrow{\quad r \quad} & \\
 & & b \leftarrow \left[[\![h']\!]^r \overset{?}{\doteq} [\![a]\!]((c_1')^{y_1} \cdots (c_l')^{y_l} [\![h_0]\!])^c \right] \\
 & \xleftarrow{\quad b \quad} &
\end{array}$$

Figure 6.7: Simplified verification protocol for encrypted certification, ElGamal.

So, we need to prove that the protocol in Figure 6.7 is a secure two-party protocol for $\mathcal{U}$ to prove knowledge of α such that $[\![h']\!]^{\alpha^{-1}} \doteq (c_1')^{y_1} \cdots (c_l')^{y_l} [\![h_0]\!]$. As already noted in Section 6.3.3, the protocol looks very similar to a Σ-protocol for relation

$$R = \{(h', (c_i')_{i=1}^l; \alpha) \mid [\![h']\!]^{\alpha^{-1}} \doteq (c_1')^{y_1} \cdots (c_l')^{y_l} [\![h_0]\!] \wedge \alpha \neq 0\}.$$

However, now also the verifier has private input, and $\mathcal{U}$ might be a malicious participant trying to learn something about the verifier's private input. Therefore, the original definition of Σ-protocol, Definition 2.6.1, does not apply anymore. Yet, this definition will turn out to be useful. We define $L_R = \{(h', (c_i')_{i=1}^l) \mid \exists_\alpha : (h', (c_i')_{i=1}^l; \alpha) \in R\}$. We require that $h' \neq 1$ (conform the construction of Section 6.3.1). Note that the verifier's private input $((y_i)_{i=1}^l, \lambda)$ is determined by the system parameter setup, and is thus fixed from the start.

We need to prove that the protocol in Figure 6.7 is correct, i.e. that it is a proof of knowledge, and that it is secure. Demonstrating that it is a proof of knowledge is captured by proving 'completeness' and 'special soundness' of Definition 2.6.1, for relation R. For security, using the real/ideal-model from Section 2.3.2, we need to prove that the adversary's view on the protocol can be simulated for any allowed adversary structure. As we have two possible adversary structures, $\mathcal{V}$ being passively corrupt or $\mathcal{U}$ being actively corrupt, we need to construct two simulators: a simulator that may use passively corrupted $\mathcal{V}$ as a subroutine, and a simulator that may use actively corrupted $\mathcal{U}'$ as a subroutine. Both simulators need to simulate the conversations of the corrupted party with an honest participant in an indistinguishable way, on any common input $(h', (c'_i)_{i=1}^l)$.

Summarizing, we first prove completeness and special soundness, and then two simulators are constructed: one for the verifier's view and one for the prover's view.

Proposition 6.4.8. *The protocol in Figure 6.7 is complete.*

Proof. For any $(h', (c'_i)_{i=1}^l; \alpha) \in R$ and any $(\mathcal{U}, \mathcal{V})$ following the protocol we have:

$$[\![h']\!]^r \doteq [\![h']\!]^{u+c\alpha^{-1}} \doteq [\![(h')^u]\!]([\![h']\!]^{\alpha^{-1}})^c \doteq [\![a]\!]((c'_1)^{y_1}\cdots(c'_l)^{y_l}[\![h_0]\!])^c,$$

where the last equality holds since $(h', (c'_i)_{i=1}^l; \alpha) \in R$. □

Proposition 6.4.9. *The protocol in Figure 6.7 is special sound.*

Proof. Given $(h', (c'_i)_{i=1}^l) \in L_R$, and two accepting conversations $(a, c, r, 1)$ and $(a, c', r', 1)$ with $c \neq c'$. In particular this means that

$$[\![h']\!]^r \doteq [\![a]\!]((c'_1)^{y_1}\cdots(c'_l)^{y_l}[\![h_0]\!])^c, \qquad [\![h']\!]^{r'} \doteq [\![a]\!]((c'_1)^{y_1}\cdots(c'_l)^{y_l}[\![h_0]\!])^{c'},$$

which implies $[\![h']\!]^{r-r'} \doteq ((c'_1)^{y_1}\cdots(c'_l)^{y_l}[\![h_0]\!])^{c-c'}$, or equivalently,

$$[\![h']\!]^{\frac{r-r'}{c-c'}} \doteq (c'_1)^{y_1}\cdots(c'_l)^{y_l}[\![h_0]\!],$$

as $c \neq c'$. Suppose $r = r'$, this means that $D((c'_1)^{y_1}\cdots(c'_l)^{y_l})h_0 = 1$, which is impossible as $h' \neq 1$. So $r \neq r'$, meaning that we obtained a witness $\alpha = \frac{c-c'}{r-r'} \neq 0$. □

Proposition 6.4.10. *For any common input $(h', (c'_i)_{i=1}^l)$, the protocol in Figure 6.7 can be simulated in a perfectly indistinguishable way, for any passively adversarial verifier $\mathcal{V}'$.*

Proof. Given a common input $(h', (c'_i)_{i=1}^l)$. For any honest prover $\mathcal{U}$ and adversarial verifier $\mathcal{V}'$ following the protocol, the real conversations satisfy the following distribution[5]:

$$\left\{(a,c,r,b) \;\middle|\; u,c \in_R \mathbb{Z}_q; a \leftarrow (h')^u; r \leftarrow u + c\alpha^{-1} \bmod q; b \leftarrow \left[[\![h']\!]^r \overset{?}{\doteq} [\![a]\!]((c'_1)^{y_1}\cdots(c'_l)^{y_l}[\![h_0]\!])^c\right]\right\}.$$

This distribution is perfectly simulated by:

$$\left\{(a,c,r,b) \;\middle|\; c,r \in_R \mathbb{Z}_q; a \leftarrow (h')^r(D((c'_1)^{y_1}\cdots(c'_l)^{y_l})h_0)^{-c}; b \leftarrow \left[[\![h']\!]^r \overset{?}{\doteq} [\![a]\!]((c'_1)^{y_1}\cdots(c'_l)^{y_l}[\![h_0]\!])^c\right]\right\}.$$

Note that the simulator knows the values $((y_i)_{i=1}^l, \lambda)$ as he may use $\mathcal{V}'$ as subroutine, and therefore he can compute $D((c'_1)^{y_1}\cdots(c'_l)^{y_l})$, where D is the decryption function. These distributions are indeed perfectly indistinguishable: for the (a, c, r) part, in both distributions we have q^2 possible uniformly random choices, and b is computed the same in both conversations. □

The construction of a simulator for the view of a malicious prover $\mathcal{U}'$ on the protocol relies on Assumption 6.4.1. We can assume that $\mathcal{U}'$ did $K \geq 0$ certificate issuing queries, and output a tuple $(h', (c'_i)_{i=1}^l, \alpha_1, z', (z'_i)_{i=1}^l, c'_0, r'_0)$. We recall that the equation $c'_0 = \mathcal{H}(\cdot)$ of (6.2) is assumed to hold.

[5]Effectively, $b = 1$ by construction. To keep the simulation clear, it is however denoted in full.

Proposition 6.4.11. *For any common input $(h', (c_i')_{i=1}^{l})$, the protocol in Figure 6.7 can be simulated in a perfectly indistinguishable way, for any actively adversarial prover $\mathcal{U}'$.*

Proof. Given a common input $(h', (c_i')_{i=1}^{l})$. For any prover $\mathcal{U}'$ and honest verifier $\mathcal{V}$, the real conversations are as follows:

- Receive a from $\mathcal{U}'$, send $c \in_R \mathbb{Z}_q$ to $\mathcal{U}'$, and receive r from $\mathcal{U}'$;
- Set $b \leftarrow \left[[\![h']\!]^r \overset{?}{\doteq} [\![a]\!]((c_1')^{y_1} \cdots (c_l')^{y_l}[\![h_0]\!])^c \right]$, and output (a, c, r, b).

We construct a simulator that also has input $(h', (c_i')_{i=1}^{l})$ and may use $\mathcal{U}'$ as a subroutine:

- Receive a from $\mathcal{U}'$, send $c \in_R \mathbb{Z}_q$ to $\mathcal{U}'$, and receive r from $\mathcal{U}'$;
- Use the extractor $\mathcal{E}$ of Assumption 6.4.1 to obtain $(h_j, (c_{ji})_{i=1}^{l})_{j=1}^{K}$ and $\tau \in \{0, \ldots, K\}$;
- Set $b \leftarrow \begin{cases} 1, \text{ if } \tau \neq 0 \text{ and } (h')^r = a h_\tau^c, \\ 0, \text{ if } \tau = 0 \text{ or } (h')^r \neq a h_\tau^c; \end{cases}$
- Output (a, c, r, b).

Remains to prove that these two distributions are indistinguishable, given any common input $(h', (c_i')_{i=1}^{l})$. But the values (a, c, r) are constructed the same in both conversations, remains to show that b is distributed the same in both sets.
Suppose that in the real conversation $b = 1$. By the special soundness property (Proposition 6.4.9), with overwhelming probability $\mathcal{U}'$ knows an α such that

$$[\![h']\!]^{\alpha^{-1}} \doteq (c_1')^{y_1} \cdots (c_l')^{y_l}[\![h_0]\!].$$

By Assumption 6.4.1 (and as $c_0' = \mathcal{H}(\cdot)$ holds), this implies that with overwhelming probability there exists a j such that (6.4) holds, which by definition means that $\tau \neq 0$. It moreover implies that:

$$\begin{aligned} [\![h']\!]^r &\doteq [\![a]\!]((c_1')^{y_1} \cdots (c_l')^{y_l}[\![h_0]\!])^c && \{\text{since } b = 1\} \\ &\doteq [\![a]\!]((c_{\tau 1})^{y_1} \cdots (c_{\tau l})^{y_l}[\![h_0]\!])^c && \{\text{equation (6.4)}\} \\ &\doteq [\![a]\!][\![h_\tau]\!]^c && \{\text{by construction}\}, \end{aligned}$$

from which it follows that $(h')^r = a h_\tau^c$. So by construction the simulator sets $b = 1$ as well.
Conversely, suppose that in the simulated conversation $b = 1$. By construction this means that $\tau \neq 0$ and $(h')^r = a h_\tau^c$. By definition, $\tau \neq 0$ implies that (6.4) is satisfied with $j = \tau$. Now:

$$\begin{aligned} [\![h']\!]^r &\doteq [\![a]\!][\![h_\tau]\!]^c && \{\text{since } (h')^r = a h_\tau^c\} \\ &\doteq [\![a]\!]((c_{\tau 1})^{y_1} \cdots (c_{\tau l})^{y_l}[\![h_0]\!])^c && \{\text{by construction}\} \\ &\doteq [\![a]\!]((c_1')^{y_1} \cdots (c_l')^{y_l}[\![h_0]\!])^c && \{\text{equation (6.4)}\}, \end{aligned}$$

which implies that also in the real execution $b = 1$. Concluding, with overwhelming probability b is computed the same in both conversations, and hence the real and simulated conversations are perfectly indistinguishable with overwhelming probability. □

Remark 6.4.12. In this proof we make explicit use of the fact that $\mathcal{V}$ can be seen as one single honest party, a simplification which is argued in Sections 3.4 and 8.1.1. This generalization is done merely for simplicity, but we briefly consider the case of $\mathcal{V}$ being a set of parties P, of which at most $t - 1$ parties P' may be malicious. As the major part of P is honest, the value c is still randomly chosen from $\mathbb{Z}_q$, so the (a, c, r) part is still constructed indistinguishably in the two conversations. Therefore, we only need to consider the value b. Following a similar argument as in the proof of Proposition 6.4.11, also this value is the same in both conversations, but in the current setting it is accompanied with a transcript of the verification by P of the equality $[\![h']\!]^r \overset{?}{\doteq} [\![a]\!]((c_1')^{y_1} \cdots (c_l')^{y_l}[\![h_0]\!])^c$. For simplicity, note that if $c = 0$ this verification is trivial, and if $c \neq 0$ the equality is equivalent to

$$[\![((h')^r a^{-1})^{c^{-1}} h_0^{-1}]\!] \overset{?}{\doteq} (c_1')^{y_1} \cdots (c_l')^{y_l}.$$

But by construction this operation is exactly the $\mathrm{pet}^{\mathrm{rep}}$ gate (4.6):

$$b \leftarrow \mathrm{pet}^{\mathrm{rep}}_{P(\lambda,(y_i)_{i=1}^l)}([\![((h')^r a^{-1})^{c^{-1}} h_0^{-1}]\!], c_1', \ldots, c_l') = \begin{cases} 1, \text{ if } ((h')^r a^{-1})^{c^{-1}} h_0^{-1} = D((c_1')^{y_1} \cdots (c_l')^{y_l}); \\ 0, \text{ otherwise,} \end{cases}$$

for which in Section 4.2.2 a simulator is constructed that given values $(h', (c_i')_{i=1}^l; a, c, r, b)$ and the shares of $(\lambda, (y_i)_{i=1}^l)$ of the malicious parties P', simulates the transcript in a computationally indistinguishable way.

7. Certificate schemes: Paillier extensions

In the previous chapter, Brands' certificate scheme is extended for the ElGamal cryptosystem. A similar extension for Paillier is discussed in this chapter. We refer the reader to Section 3.2 and the introduction of Chapter 6 for a discussion of the required extensions. Clearly, extending Brands' certificate scheme with Paillier would be very useful in the context of multiparty computation: as we noticed in Section 2.9, for the ElGamal cryptosystem a general multiplication protocol cannot be constructed, which is an important gate in many secure function evaluations. For Paillier, this gate *is* possible, and therefore a Paillier extension of Brands' scheme would offer more functionalities. In particular, our aim is to combine a certificate scheme with multiparty computation of statistics and many of the functions of Section 4.5 are impossible for ElGamal encryptions with unknown bit representation.

In this chapter, protocols **A** and **B** from Section 3.2.3 are constructed for the Paillier cryptosystem. This is done in Sections 7.1 and 7.2, respectively. As the constructions are similar to the constructions for the ElGamal cryptosystem, we consider them as an extension of the results presented in Chapter 6. Since the security analysis is analogous, this is done very briefly. We also considered scheme **C** for the Paillier cryptosystem, but did not find a provably secure scheme. Therefore, a construction of this scheme remains as an open problem.

For the cryptosystem, we assume that the key generation is performed as follows for each scheme. Observe that this setup is implicitly included in the key generation algorithm of the certificate scheme. We consider a modulus $m = pq$ of length at least k, with p, q safe primes, and a value $s \in \mathbb{N}$. The message space is $\mathbb{Z}_{m^s}$. The secret decryption key is μ, which is shared among the n multiparty computation servers P, so that any t participants can decrypt. Each participant holds share μ_i. Throughout this chapter, Paillier encryptions are denoted by $[\![x]\!]_P$. We recall from Section 3.4 that in case the participants P jointly act as a verifier or certification authority, we just consider them as one player. In particular, if for instance $P = \mathcal{CA}$, the secret keys obtained from the key generation algorithm are implicitly (t, n)-threshold shared. To this end, the same distributed key generation protocol as for sharing an ElGamal secret key can be used (Section 2.5.1), as the certificate scheme is over group G of prime order q as well.

7.1 Protocol A: encrypted disclosure

As an extension to the verification protocol in Section 5.2, now the user discloses encryptions of the attributes. Similar to the case of ElGamal encryptions, we only need to adjust the verification protocol of Section 5.2.3. The user has a certificate $(h', (x_i)_{i=1}^{l}, \alpha_1, z', c_0', r_0')$ satisfying (5.1), and he discloses x_1 in encrypted form and keeps the other attributes secret. So in the verification protocol, together with the values (h', z', c_0', r_0'), in the first round the user now also sends a Paillier encryption of x_1. Therefore, he takes $r \in_R \mathbb{Z}^*_{m^{s+1}}$, sets $c_1 = [\![x_1, r]\!]_P = (m+1)^{x_1} r^{m^s}$, and sends this value to $\mathcal{V}$. In contrast to the ElGamal case, now the user needs to prove knowledge of the value x_1 such that it satisfies a statement regarding *two different groups*. Thus we need a verification protocol using integer commitments, Section 2.6.1 with $G_2 = \mathbb{Z}^*_{m^{s+1}}$. We assume that the setup for the integer commitments as in Definition 2.4.2 is done: we have security parameters ℓ_x, ℓ_c, ℓ_s such that $2^{\ell_x+\ell_c+\ell_s+1} < \min\{q, m^s\}$, and we require that $-2^{\ell_x} < x_1 < 2^{\ell_x}$ in order for the protocol to succeed. The new verification protocol is the Σ-protocol in Figure 7.1 for the relation

$$\Big\{(h', c_0; (x_i)_{i=1}^l, \alpha_1, r) \;\Big|\; h_0 \overset{G}{=} (h')^{\alpha_1^{-1}} g_1^{-x_1} \cdots g_l^{-x_l} \wedge \alpha_1 \neq 0 \wedge \\ c_0 \overset{\mathbb{Z}^*_{m^{s+1}}}{=} (m+1)^{x_1} r^{m^s} \wedge (-2^{\ell_x+\ell_c+\ell_s} < x_1 < 2^{\ell_x+\ell_c+\ell_s})\Big\}.$$

$$\begin{array}{rcl}
\mathcal{U} & & \mathcal{V} \\
(\text{knows: } h', c_0; (x_i)_{i=1}^l, \alpha_1, r) & & (\text{knows: } h', c_0) \\
\tilde{r} \in_R \{0, \ldots, \lfloor m/4 \rfloor - 1\}, \quad y \leftarrow h_1^{x_1} h_2^{\tilde{r}} & & \\
u_1 \in_R \{0,1\}^{\ell_x+\ell_c+\ell_s} & & \\
u_2, \ldots, u_l, u_\alpha \in_R \mathbb{Z}_q & & \\
u_r \in_R \mathbb{Z}^*_{m^{s+1}}, \quad u_{\tilde{r}} \in_R \{0,1\}^{k/4+\ell_c+\ell_s} & & \\
a_h \overset{G}{\leftarrow} (h')^{u_\alpha} g_1^{-u_1} \cdots g_l^{-u_l} & & \\
a_0 \overset{\mathbb{Z}^*_{m^{s+1}}}{\leftarrow} (m+1)^{u_1} u_r^{m^s}, \quad a_y \overset{\mathbb{Z}^*_m}{\leftarrow} h_1^{u_1} h_2^{u_{\tilde{r}}} & \xrightarrow{y, a_h, a_0, a_y} & \\
 & \xleftarrow{c} & c \in_R \{0,1\}^{\ell_c} \\
r_1 \leftarrow u_1 + c x_1 & & \\
(r_i \leftarrow u_i + c x_i \bmod q)_{i=2}^l & & \\
r_\alpha \leftarrow u_\alpha + c\alpha_1^{-1} \bmod q & & \\
r_r \overset{\mathbb{Z}^*_{m^{s+1}}}{\leftarrow} u_r r^c, \quad r_{\tilde{r}} \leftarrow u_{\tilde{r}} + c\tilde{r} & \xrightarrow{(r_i)_{i=1}^l, r_\alpha, r_r, r_{\tilde{r}}} & \\
 & & (h')^{r_\alpha} g_1^{-r_1} \cdots g_l^{-r_l} \overset{?}{=} a_h h_0^c \\
 & & (m+1)^{r_1} r_r^{m^s} \overset{?}{=} a_0 c_0^c \\
 & & h_1^{r_1} h_2^{r_{\tilde{r}}} \overset{?}{=} a_y y^c, \quad r_1 \overset{?}{\in} \{0,1\}^{\ell_x+\ell_c+\ell_s}
\end{array}$$

Figure 7.1: Brands' verification protocol, with Paillier encrypted disclosure.

The proof of correctness for Figure 7.1 can be obtained by combining the proofs of Lemma 4.1.2 and Proposition 6.1.1, and is therefore omitted. We have statistical security under the DL Assumption 2.2.1 (recall that DL $\iff$ DLREP), the DCR Assumption 2.2.7 (for Paillier encryption) and the SRSA Assumption 2.2.6 (for the integer commitment).

7.2 Protocol B: attribute hiding issuance

In this case, $\mathcal{CA} = P$ outputs, as a team of multiparty computation servers, an encryption $c_1 = [\![x_1]\!]_P$ that needs to be certified. The participants (t, n)-threshold share the secret decryption key μ. Now, $\mathcal{CA}$ is not allowed to learn x_1 but $\mathcal{U}$ is. Similar to Section 6.2, we only consider $l = 2$, so we consider two attributes (x_1, x_2). The generic approach of Section 6.2 still applies, where the private output protocol (Protocol 2.9.1) is utilized. Recall that in a protocol for private output only one participant (or more generally, a restricted number of participants) learns the encrypted value.

Now, the sub-protocol of Figure 6.2 can be constructed as follows. On input of $[\![x_1]\!]_P$ the protocol results in a value $h = g_1^{x_1} g_2^{x_2} h_0$ where $\mathcal{U}$ knows x_1, x_2. Naturally, again this protocol involves integer commitments, and we assume that the setup is as in Definition 2.4.2. That is, we have security parameters ℓ_x, ℓ_c, ℓ_s such that $2^{\ell_x+\ell_c+\ell_s+1} < \min\{q, m^s\}$, and we require that $-2^{\ell_x} < x_1 < 2^{\ell_x}$ in order for the protocol to succeed.

$$
\begin{array}{rcl}
\mathcal{U} & & \mathcal{CA} \\
\text{(knows: } c_1\text{)} & & \text{(knows: } c_1, \mu\text{)} \\
 & \xleftrightarrow{\text{Protocol 2.9.1 on } c_1} & \\
x_2 \in_R \mathbb{Z}_q, \quad h \leftarrow g_1^{x_1} g_2^{x_2} h_0 & \xrightarrow{\quad h \quad} & \\
 & \xleftrightarrow{\quad \Sigma \quad} & \\
\text{(knows: } h; x_1, x_2\text{)} & & \text{(knows: } h\text{)}
\end{array}
$$

Figure 7.2: Sub-protocol for attribute hiding issuing protocol, Paillier.

Here, Σ denotes a Σ-protocol for $\mathcal{U}$ to prove that he knows x_1, x_2 and that the x_1 in h equals the x_1 in c_1. Using the notation of Protocol 2.9.1, this is a Σ-protocol for the following relation:

$$
\Big\{(c', d, h; x_1, x_2, t) \,\Big|\, c' \overset{\mathbb{Z}^*_{m^{s+1}}}{=} (m+1)^{d-x_1} t^{m^s} \wedge h \overset{G}{=} g_1^{x_1} g_2^{x_2} h_0 \wedge \\
(-2^{\ell_x+\ell_c+\ell_s} < x_1 < 2^{\ell_x+\ell_c+\ell_s})\Big\}.
$$

The Σ-protocol is similar to the one shown in Figure 4.1 for relation (4.1), and is therefore omitted.

8. Remarks and conclusions

In this chapter, we compare the different schemes presented throughout the thesis from an efficiency perspective, and provide recommendations for further research. Some remarks regarding the schemes in Chapters 5-7 are discussed here. In Section 8.1 the certificate schemes, and in particular the differences between the newly discussed encrypted certificate scheme (Section 6.3) and Brands' certificate scheme are discussed. This section also considers the 'restrictions' put on the participants. Section 8.2 discusses the possibility to mix the discussed schemes. Indeed, all schemes are based on extreme scenarios (for instance, *all* attributes are encrypted or *all* attributes are in plain) and these scenarios can be 'mixed'. In particular, it turns out that the efficiency of the scheme can be increased if different schemes are mixed. In Section 8.3, the possibilities to construct an encrypted certificate scheme with respect to the universal construction of Chapter 3 for two Camenisch-Lysyanskaya certificate schemes [CL02, CL04] are considered.
Finally, in Section 8.4 this thesis is concluded, the major contributions are summarized and open questions are discussed.

8.1 Efficiency analysis

In this section, we analyze the efficiency of the schemes proposed in this thesis. In particular, we compare the schemes with Brands' certificate scheme. We also summarize the required assumptions and the 'restrictions' put on the participants.

8.1.1 Participants

In the extensions discussed throughout the thesis, we require that $P = \mathcal{CA}$ (in Sections 6.2 and 7.2) or even $P = \mathcal{CA} = \mathcal{V}$ (in Section 6.3). Clearly, the former is no restriction at all. In fact, considering the $\mathcal{CA}$ as a group of participants is a practical way to guarantee that the $\mathcal{CA}$ is not actively attacking the protocol (so he acts like an honest-but-curious party). The restriction $P = \mathcal{CA} = \mathcal{V}$ however *is* a restriction. Indeed, it means that a certificate is not publicly verifiable anymore. It can only be verified by the certification authority itself. We believe that this is not a big restriction, simply because of the reason why we need the scheme in the first place: $\mathcal{CA}$ is the set of multiparty computation servers. Jointly, they output an encryption and certify it for $\mathcal{U}$. Then, at a later point in time, $\mathcal{U}$ could come back to $\mathcal{CA}$ with the certified encryption, so that $\mathcal{CA}$ can continue doing multiparty computation with that encryption. So in this scenario our assumption that $\mathcal{CA} = \mathcal{V}$ holds in practice.

We note however that it *is* possible to extend the scheme in Section 6.3 to a system where $\mathcal{CA} \neq \mathcal{V}$. Indeed, for the verification protocol $\mathcal{V}$ needs the secret values λ and $(y_i)_{i=1}^{l}$, and $\mathcal{CA}$ only needs x_0 for *the pure issuing protocol*, but he needs the other values for the preliminary setup of the values $(h, (c_i)_{i=1}^{l})$. This means that if $\mathcal{CA}$ is given these values, or in particular if $\mathcal{CA}$ knows the plaintext attributes of the certificate (but $\mathcal{U}$ still does not learn them), certificate issuance is possible only using x_0 and $(\phi_i)_{i=1}^{l-1}$. In this way, the roles of $\mathcal{CA}$ and $\mathcal{V}$ can be separated. This is however a restriction of the original scheme, since now either the $\mathcal{CA}$ needs some kind of oracle for $(h, (c_i)_{i=1}^{l})$, or he has to learn the plaintext attributes.

We think that it is impossible to turn the encrypted certificates into universally verifiable certificates. Indeed, the verifier needs to do a plaintext equality test since the verification predicate involves encryptions of which the prover needs to prove some property. This plaintext equality test can only be done by the owner of the decryption key.

8.1.2 Assumptions

Depending on the different choices of schemes, different assumptions are needed. These are discussed briefly. If more schemes are mixed (see Section 8.2) the required assumptions are the collection of the assumptions corresponding to the separate schemes. We remind the reader of the hierarchical structure of DL $\Leftarrow$ DDH, by which the term DL has disappeared in some places, but it is nonetheless implied. See Table 8.1.

		At verification: disclosure in:		
		plaintext	ElGamal	Paillier
At issuing:	plaintext	DL	DDH	DL+DCR+SRSA
$\mathcal{U}$ knows:	ElGamal	$\lightning$	DDH	$\lightning$

Table 8.1: Comparison of the required assumptions.

8.1.3 Complexity and overhead

Table 8.2 gives an overview of the sizes of the certificates and the communications for the different schemes. We write $|G|$, $|\mathbb{Z}_q|$, etc. separately to denote the type of values involved. Moreover, for simplicity elements in $\{0,1\}^{\ell_c}$ are just assumed to be elements in $\mathbb{Z}_q$. The key generation algorithm is not considered. Also, only the communication between $\mathcal{U}$ and $\mathcal{CA}$ or $\mathcal{U}$ and $\mathcal{V}$ is considered (so the case of $\mathcal{CA}$ or $\mathcal{V}$ being a set of participants P is ignored). Firstly, if $\mathcal{U}$ knows the attributes in plain (the first row of the table), then the only differences among the schemes are in the overhead of the verification protocols. It is clear that the differences in the communication complexities between plaintext disclosure and encrypted disclosure are constant in the number of attributes l. The encrypted certificate scheme from Section 6.3 is clearly more expensive than the standard case where $\mathcal{U}$ learns the plaintext values, but its overhead is only of linear order in l. For the communication complexity for $\mathcal{CA} = \mathcal{V} = P$, extra costs are in the computation of the plaintext equality test in the verification protocol of Figure 6.5, which according to Section 4.2.4 takes $O(nk)$ broadcast complexity. It requires about $O(n+l)$ exponentiations. This can be seen as the performance cost of the encrypted certificates.

$\mathcal{U}$ knows		communication overhead of	
attributes in:	certificate size	issuance	verification, disclosure in:
plaintext	$2\|G\| + (l+3)\|\mathbb{Z}_q\|$	$4\|G\| + 2\|\mathbb{Z}_q\|$	plaintext: $3\|G\| + (l+4)\|\mathbb{Z}_q\|$
			ElGamal: $7\|G\| + (l+5)\|\mathbb{Z}_q\|$
			Paillier: $3\|G\| + (l+6)\|\mathbb{Z}_q\| + 2\|\mathbb{Z}^*_m\| + 2\|\mathbb{Z}^*_{m^{s+1}}\|$
ElGamal	$(4l+2)\|G\| + 3\|\mathbb{Z}_q\|$	$(6l+5)\|G\| + 2\|\mathbb{Z}_q\|$	$(4l+3)\|G\| + 4\|\mathbb{Z}_q\|$

Table 8.2: Comparison of the sizes of certificates and communication sizes. The overhead of the verification protocol is based on the case where the user discloses one attribute and keeps the other $l-1$ secret. For the overhead of the issuance protocol, additional executions of the private output protocol are not considered.

Finally, it seems that further optimization of the scheme in Section 6.3 is not possible. For instance, if in the issuing protocol an encryption, say c_1, is left out of the hash function, then a malicious user $\mathcal{U}'$ can set $c'_1 \leftarrow c_1[\![x]\!]$, where $x \in \{0,1\}$ is unknown to $\mathcal{U}'$. It turns out that then the verification protocol succeeds *if and only if* $x = 0$. So this gives $\mathcal{U}'$ and oracle for bit-decryption[1]. Also, as argued in Remark 6.3.1 the values ϕ_i are really required.

8.2 Mixing different certificate schemes

The schemes described in this thesis are meant to be used for a single purpose. For instance, the scheme with the issuing protocol of Section 6.2 considers the case where encryptions come from multiparty computation and $\mathcal{U}$ learns all values, while the issuing protocol in Section 6.3 discusses the case where $\mathcal{U}$ does not learn *any* of the values. Clearly, these protocols can be combined. In fact, also the schemes from Chapters 6 and 7 can be mixed in straightforward ways (for instance if c_1 is ElGamal encrypted and c_2 Paillier). Also the disclosure protocols can be mixed arbitrarily: it can for instance be the case that $\mathcal{U}$ comes up with a certificate, and wants to show one subset of attributes in plain, and another subset using Paillier encryptions. In particular, Brands constructed protocols to disclose arbitrary relations on the attributes, for instance $x_1+2x_2 \equiv 3x_3 \bmod q \wedge x_4+5x_3 \equiv x_5 \bmod q$, without disclosing the attributes in plaintext [Bra99, Ch. 3]. As mentioned in Section 6.1, these methods can also be combined with disclosing the attributes in encrypted form (protocol **A**). For the verification protocol of the encrypted certificate scheme of Section 6.3.3 (scheme **C**), these methods make no sense as the user simply sends encryptions of the attributes to the verifier.

This generalization for 'mixed' scenarios is immediate. We note however that if $\mathcal{U}$ knows an attribute x_i in plaintext, he can decide among verification protocols 5.2 (plaintext disclosure) or 6.1 or 7.1 (encrypted disclosure), but if $\mathcal{U}$ knows x_i in encrypted form (in Scheme 6.3), he is thrown back on the verification protocol in Section 6.3.3.

Mixing different protocols can reduce the complexity, which can be seen in the following example: Suppose $\mathcal{CA}$'s multiparty computation results in a set of ElGamal encryptions $c_1, \ldots, c_{l-1}$, and $\mathcal{U}$ is not allowed to learn $D(c_1)$ (but he is allowed to learn the other decryptions). In this case, the issuing protocols of Sections 6.2 (for the encryptions $c_2, \ldots, c_{l-1}$) and 6.3.2 (for c_1) can be mixed[2]. Following Table 8.2, for this example the efficiency difference compared to the basic scheme of Brands is of constant order.

8.3 Extending other certificate schemes

In this thesis, the universal approach of Chapter 3 is only applied to Brands' discrete log certificate scheme. An interesting question is how this approach applies to other certificate schemes. We single out the two Camenisch-Lysyanskaya schemes [CL02, CL04]. Due to [BCL06], these are also provided with a protocol for selective plaintext disclosure (as required in Section 3.2.1). It turns out that protocols **A** and **B** from Section 3.2.3 are easily possible. For scheme **C**, the generalization is not as straightforward and it remains as an open problem.

A: (Encrypted disclosure of attributes). This protocol is easily possible for both schemes. The extension relies on exploiting Σ-protocols, possibly with the use of integer commitments (Definition 2.4.2) if it involves different groups. A deep comparison for encrypted disclosure among these schemes and Brands' certificate scheme is not considered, but using one

[1]Also, if x is not a bit, the resulting scheme is an oracle for the equality '$x \overset{?}{=} 0$'.

[2]But the l-th attribute still needs to be in line with the *encrypted* issuance protocol since otherwise $\mathcal{U}$ can learn the value $g_1^{x_1}$ from h, and thus obtains some information about $D(c_1)$.

of the CL-schemes degrades the required assumptions compared to Brands' scheme, as becomes clear from Table 8.3. It is clear that in all cases Brands' scheme is based on weaker assumptions[3];

B: (Attribute hiding issuance). Extending the schemes for the case where the multiparty computation results in an encryption that needs to be certified, with the user still allowed to learn the plaintext, is easily possible. The approach is also based on a private output protocol (Section 2.9.1) and works because both [CL02, CL04] offer the possibility of 'certification on committed values' (where the service provider certifies attributes that are only known by $\mathcal{U}$);

C: (Encrypted certificate scheme). Although not proven, this scheme looks impossible, or at least more complicated to construct. This is because both schemes function with *multi-show* certificates, and this multi-show functionality comes from the possibility of the user to completely blind the values in the certificate before each verification execution (independently from the issuing protocol). Now consider an encryption c, that is certified. Upon verification, the user blinds c to $c' = c[\![0, r]\!]$ for random r and sends it to the verifier. But for both CL-schemes it is impossible for the user to prove knowledge of r as the verifier does not know c, and therefore an attacker setting $c' = c[\![x]\!]$ for unknown x would succeed with probability $Pr(x = 0)$. If x is from a restricted domain, this probability is non-negligible and the adversary thus obtains an oracle for '$x \overset{?}{=} 0$'. (See also the footnote on page 30.) So while in Brands' one-show certificate scheme the blinding is done during the issuing execution and the previously described attacks are prevented, for the CL-schemes this attack *is* possible. This implies that a more complex operation is necessary for the CL-schemes, rather than the approach used in this thesis.

Disclosure in:	plaintext	ElGamal, same order	ElGamal, diff. order	Paillier, same mod.	Paillier, diff. modulus
[Bra99]	DL	DDH	SRSA+DDH	↯^{i}	SRSA+DL+DCR
[CL02]	SRSA+DL	↯^{i}	SRSA+DDH	SRSA+DL+ DCR	SRSA+DL+DCR
[CL04]	LRSW	↯^{ii}	LRSW+SRSA+ DDH	↯^{i}	LRSW+SRSA+ DCR

^{i}: The certificate scheme and the encryption scheme work on incomparable groups.
^{ii}: ElGamal relies on the DDH assumption, but the certificate scheme is pairing based.

Table 8.3: Comparison of the assumptions needed if the certificate scheme is extended to encrypted disclosure, for [Bra99, CL02, CL04].

8.4 Conclusions

Via the construction of new protocols, a certificate scheme and secure multiparty computation are connected, in such a way that several new functionalities are possible (Figure 3.2). Specifically, the construction enables a service provider to be a set of multiparty computation participants that obtains encryptions, possibly certified, does some multiparty computation on them and outputs an encrypted result, again possibly certified. Although the construction in this work focuses on one particular certificate scheme, the discrete log based scheme of Brands [Bra99], the construction as described in Chapter 3 is universal, in the sense that it also applies to other certificate schemes, e.g. [CL02, CL04] (as long as the components **A-C** of Section 3.2.3 *can* be constructed for the

[3] But unlike the CL-schemes, Brands' scheme is at many points only *assumed* to be secure, implicitly implying the use of additional assumptions.

certificate scheme).

The most significant contributions in the area of certificate schemes are a reformulation and formalization of the definition of 'certificate schemes' by Brands [Bra99] and the generalization of Brands' certificate scheme for ElGamal encryptions such that the outcome of a multiparty computation can be certified, regardless of the rights of the user and the certification authority to learn the plaintext. In a restricted form, the same generalization is constructed for the Paillier cryptosystem. Particularly, the notion of 'encrypted certificate scheme' is introduced in this thesis and a concrete construction is given and proven secure. This extension is fairly efficient. To our knowledge, such a scheme has not been studied earlier in the literature.

In the context of multiparty computation, as one of the main purposes of our scheme was to perform *statistical* analysis (on certified inputs and possibly with certified outputs), the set of available gates is extended. Some multiparty computation gates were only needed for the construction of the certificate schemes (discussed above), but can be seen as independent protocols as well. The most interesting contributions in the context of multiparty computation are a generalization of the plaintext equality test, so that it is applicable for more general purposes, and the construction of an efficient modulo reduction gate. Moreover, computing statistics over encryptions is made possible. The most remarkable stumbling block was the fact that statistics often require integer divisions, which are not well-defined in finite rings (specifically for ElGamal and Paillier encryptions). Using the mentioned modulo reduction gate, this problem has been solved.

All contributions together describe a complete system for combining multiparty computation and certificate schemes, following the universal construction of Chapter 3. Yet, the concrete construction based on Brands certificate scheme [Bra99], as made in this thesis, is not for free. It lead to two restrictions: the verification is not publicly possible and security is based on an additional assumption, and although these two restrictions are argued unavoidable (Section 8.1) and conceivable (Appendix A) respectively, solving these restrictions remains an open problem.

8.4.1 Further research

Although the scheme proposed in this thesis is complete, its construction could accept some optimizations. We will briefly consider some possible directions for further research.

Other encrypted certificate schemes. In this thesis, the universal approach of Chapter 3 is only applied to Brands' discrete log certificate scheme, and for the ElGamal cryptosystem. For the Paillier cryptosystem, only protocols **A** and **B** of the universal approach are constructed. The construction of an encrypted certificate scheme for the Paillier cryptosystem remains as an open problem (see also Chapter 7). Also, it might be interesting to study this general construction for other certificate schemes in the literature, as mentioned in Section 8.3 where two schemes of Camenisch-Lysyanskaya [CL02, CL04] are considered. Moreover, it might be possible to construct a 'stand-alone' encrypted certificate scheme, that is not based on another certificate scheme in literature.

Proving Assumption 6.4.1. Similar to the scheme of Brands, blinding-invariance unforgeability turned out to be more complicated to prove, and also for the security of our scheme an additional assumption is required. Informally, the assumption states that if a malicious user manages to come up with a certificate, then with overwhelming probability he has been issued a certificate on the same list of attributes. Unsuccessfully, we tried to reduce Assumption 6.4.1 to Brands' assumption (Assumption 5.2.1), and therefore proving this assumption remains open. Moreover, it has not been considered to prove this assumption in the case certificates are issued sequentially.

Publicly verifiable certificates. The proposed scheme requires that upon verification the verifier executes a plaintext equality test. This is because the user does not know the attributes and he has to send the encryptions instead. Therefore, the verification requires a plaintext equality test and the verifier needs the secret key. Although this is argued to be not really a restriction (in the original system design, the only allowed verifier was the certification authority anyway), it would be interesting if the scheme can be constructed without the verifier needing secret information.

Bibliography

[ACS02] Joy Algesheimer, Jan Camenisch, and Victor Shoup. Efficient computation modulo a shared secret with application to the generation of shared safe-prime products. In *Advances in Cryptology - CRYPTO '02*, volume 2442 of *Lecture Notes in Computer Science*, pages 417–432, Berlin, 2002. Springer-Verlag.

[AM09] Divesh Aggarwal and Ueli Maurer. Breaking RSA generically is equivalent to factoring. In *Advances in Cryptology - EUROCRYPT '09*, volume 5479 of *Lecture Notes in Computer Science*, Berlin, 2009. Springer-Verlag. Extended version available at `http://eprint.iacr.org/2008/260`.

[Bat68] Ken Batcher. Sorting networks and their applications. In *Proceedings of the AFIPS Spring Joint Computing Conference '68*, volume 32, pages 307–314, 1968.

[BCC88] Gilles Brassard, David Chaum, and Claude Crépeau. Minimum disclosure proofs of knowledge. *Journal of Computer and System Sciences*, 37(2):156–189, 1988.

[BCDG87] Ernest Brickell, David Chaum, Ivan Damgård, and Jeroen van de Graaf. Gradual and verifiable release of a secret. In *Advances in Cryptology - CRYPTO '87*, volume 293 of *Lecture Notes in Computer Science*, pages 156–166, Berlin, 1987. Springer-Verlag.

[BCKL08] Mira Belenkiy, Melissa Chase, Markulf Kohlweiss, and Anna Lysyanskaya. P-signatures and noninteractive anonymous credentials. In *Theory of Cryptology*, volume 4948 of *Lecture Notes in Computer Science*, pages 356–374, Berlin, 2008. Springer-Verlag.

[BCL06] Endre Bangerter, Jan Camenisch, and Anna Lysyanskaya. A cryptographic framework for the controlled release of certified data. In *Security Protocols*, volume 3957 of *Lecture Notes in Computer Science*, pages 20–42, Berlin, 2006. Springer-Verlag.

[BDD07] Stefan Brands, Liesje Demuynck, and Bart De Decker. A practical system for globally revoking the unlinkable pseudonyms of unknown users. In *Information Security and Privacy*, volume 4586 of *Lecture Notes in Computer Science*, pages 400–415, Berlin, 2007. Springer-Verlag.

[BDZ03] Feng Bao, Robert Deng, and Huafei Zhu. Variations of Diffie-Hellman problem. In *Information and Communications Security*, volume 2836 of *Lecture Notes in Computer Science*, pages 301–312, Berlin, 2003. Springer-Verlag.

[Bea91] Donald Beaver. Secure multiparty protocols and zero-knowledge proof systems tolerating a faulty minority. *Journal of Cryptology*, 4(2):75–122, 1991.

[BF97] Dan Boneh and Matthew Franklin. Efficient generation of shared RSA keys. In *Advances in Cryptology - CRYPTO '97*, volume 1294 of *Lecture Notes in Computer Science*, pages 425–439, Berlin, 1997. Springer-Verlag.

[BG92] Mihir Bellare and Oded Goldreich. On defining proofs of knowledge. In *Advances in Cryptology - CRYPTO '92*, volume 740 of *Lecture Notes in Computer Science*, pages 390–420, Berlin, 1992. Springer-Verlag.

[BGN06] Dan Boneh, Eu-Jin Goh, and Kobbi Nissim. Evaluating 2-DNF formulas on ciphertexts. In *Theory of Cryptography*, volume 3378 of *Lecture Notes in Computer Science*, pages 325–341, Berlin, 2006. Springer-Verlag.

[BGW88] Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation. In *Proc. ACM Symposium on Theory of Computing '88*, pages 1–10, New York, 1988. ACM.

[Bla79] George Blakley. Safeguarding cryptographic keys. In *Proceedings of the AFIPS National Computer Conference '79*, volume 48, pages 313–317, 1979.

[Blu81] Manuel Blum. Coin flipping by telephone. In *Advances in Cryptology - CRYPTO '81*, Lecture Notes in Computer Science, pages 11–15, 1981.

[Bon98] Dan Boneh. The decision Diffie-Hellman problem. In *Algorithmic Number Theory*, volume 1423 of *Lecture Notes in Computer Science*, pages 48–63, Berlin, 1998. Springer-Verlag.

[Bou00] Fabrice Boudot. Efficient proofs that a committed number lies in an interval. In *Advances in Cryptology - EUROCRYPT '00*, volume 1807 of *Lecture Notes in Computer Science*, pages 431–444, Berlin, 2000. Springer-Verlag.

[BP97] Niko Barić and Birgit Pfitzmann. Collision-free accumulators and fail-stop signature schemes without trees. In *Advances in Cryptology - EUROCRYPT '97*, volume 1233 of *Lecture Notes in Computer Science*, pages 480–494, Berlin, 1997. Springer-Verlag.

[BR93] Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *ACM Conference on Computer and Communications Security*, pages 62–73, New York, 1993. ACM.

[Bra93] Stefan Brands. Untraceable off-line cash in wallet with observers. In *Advances in Cryptology - CRYPTO '93*, volume 773 of *Lecture Notes in Computer Science*, pages 302–318, Berlin, 1993. Springer-Verlag.

[Bra95a] Stefan Brands. Off-line electronic cash based on secret-key certificates. In *Advances in Cryptology - LATIN '95*, volume 911 of *Lecture Notes in Computer Science*, pages 131–166, Berlin, 1995. Springer-Verlag.

[Bra95b] Stefan Brands. Restrictive blinding of secret-key certificates. In *Advances in Cryptology - EUROCRYPT '95*, volume 921 of *Lecture Notes in Computer Science*, pages 231–247, Berlin, 1995. Springer-Verlag.

[Bra95c] Stefan Brands. Secret-key certificates. Technical Report CS-R9510 1995, Centrum voor Wiskunde en Informatica, Amsterdam, 1995.

[Bra99] Stefan Brands. *Rethinking Public Key Infrastructures and Digital Certificates - Building In Privacy.* PhD thesis, Eindhoven University of Technology, Eindhoven, 1999. `http://www.credentica.com/the_mit_pressbook.html`.

[Bra02] Stefan Brands. A technical overview of digital credentials. Feb 2002.

[Can95] Ran Canetti. *Studies in Secure Multi-Party Computation and Applications.* PhD thesis, Weizmann Institute, Israel, 1995.

[Can00] Ran Canetti. Security and composition of multi-party cryptographic protocols. *Journal of Cryptology*, 13(1):143–202, 2000.

[CCS99] Arjeh Cohen, Hans Cuypers, and Hans Sterk. *Learning Algebra in an Exciting way.* Springer-Verlag, Berlin, 1999.

[CDM00] Ronald Cramer, Ivan Damgård, and Ueli Maurer. General secure multi-party computation from any linear secret sharing scheme. In *Advances in Cryptology - EUROCRYPT '00*, volume 1807 of *Lecture Notes in Computer Science*, pages 316–334, Berlin, 2000. Springer-Verlag.

[CDN01] Ronald Cramer, Ivan Damgård, and Jesper Nielsen. Multiparty computation from threshold homomorphic encryption. In *Advances in Cryptology - EUROCRYPT '01*, volume 2045 of *Lecture Notes in Computer Science*, pages 280–300, Berlin, 2001. Springer-Verlag.

[CDS94] Ronald Cramer, Ivan Damgård, and Berry Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. In *Advances in Cryptology - CRYPTO '94*, volume 839 of *Lecture Notes in Computer Science*, pages 174–187, Berlin, 1994. Springer-Verlag.

[CFN90] David Chaum, Amos Fiat, and Moni Naor. Untraceable electronic cash. In *Advances in Cryptology - CRYPTO '88*, volume 403 of *Lecture Notes in Computer Science*, pages 319–327, Berlin, 1990. Springer-Verlag.

[Cha83] David Chaum. Blind signatures for untraceable payments. In *Advances in Cryptology - CRYPTO '82*, Lecture Notes in Computer Science, pages 199–203. Plenum Press, 1983.

[Cha84] David Chaum. Blind signature system. In *Advances in Cryptology - CRYPTO '83*, Lecture Notes in Computer Science, page 153. Plenum Press, 1984.

[CKW04] Jan Camenisch, Maciej Koprowski, and Bogdan Warinschi. Efficient blind signatures without random oracles. In *Security in Communication Networks*, volume 3352 of *Lecture Notes in Computer Science*, pages 134–148, Berlin, 2004. Springer-Verlag.

[CL01] Jan Camenisch and Anna Lysyanskaya. An efficient system for non-transferable anonymous credentials with optional anonymity revocation. In *Advances in Cryptology - EUROCRYPT '01*, volume 2045 of *Lecture Notes in Computer Science*, pages 93–118, Berlin, 2001. Springer-Verlag.

[CL02] Jan Camenisch and Anna Lysyanskaya. A signature scheme with efficient protocols. In *Security in Communication Networks*, volume 2576 of *Lecture Notes in Computer Science*, pages 268–289, Berlin, 2002. Springer-Verlag.

[CL04] Jan Camenisch and Anna Lysyanskaya. Signature schemes and anonymous credentials from bilinear maps. In *Advances in Cryptology - CRYPTO '04*, volume 3152 of *Lecture Notes in Computer Science*, pages 56–72, Berlin, 2004. Springer-Verlag.

[CP93] David Chaum and Torben Pryds Pedersen. Wallet databases with observers. In *Advances in Cryptology - CRYPTO '92*, volume 740 of *Lecture Notes in Computer Science*, pages 89–105, Berlin, 1993. Springer-Verlag.

[Cra97] Ronald Cramer. *Modular Design of Secure yet Practical Cryptographic Protocols.* PhD thesis, University of Amsterdam, Amsterdam, 1997.

[CS98] Ronald Cramer and Victor Shoup. A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In *Advances in Cryptology - CRYPTO '98*, volume 1462 of *Lecture Notes in Computer Science*, pages 13–25, Berlin, 1998. Springer-Verlag.

[Dam88] Ivan Damgård. Payment systems and credential mechanisms with provable security against abuse by individuals. In *Advances in Cryptology - CRYPTO '88*, volume 403 of *Lecture Notes in Computer Science*, pages 328–335, Berlin, 1988. Springer-Verlag.

[Dam92] Ivan Damgård. Towards practical public key systems and secure against chosen ciphertext attacks. In *Advances in Cryptology - CRYPTO '91*, volume 576 of *Lecture Notes in Computer Science*, pages 445–456, Berlin, 1992. Springer-Verlag.

[DF90] Yvo Desmedt and Yair Frankel. Threshold cryptosystems. In *Advances in Cryptology - CRYPTO '89*, volume 435 of *Lecture Notes in Computer Science*, pages 307–315, Berlin, 1990. Springer-Verlag.

[DF02] Ivan Damgård and Eiichiro Fujisaki. A statistically-hiding integer commitment scheme based on groups with hidden order. In *Advances in Cryptology - ASIACRYPT '02*, volume 2501 of *Lecture Notes in Computer Science*, pages 125–142, Berlin, 2002. Springer-Verlag.

[DFK$^+$06] Ivan Damgård, Matthias Fitzi, Eike Kiltz, Jesper Nielsen, and Tomas Toft. Unconditionally secure constant-rounds multi-party computation for equality, comparison, bits and exponentiation. In *Theory of Cryptography*, volume 3876 of *Lecture Notes in Computer Science*, pages 285–304, Berlin, 2006. Springer-Verlag.

[DGK07] Ivan Damgård, Martin Geisler, and Mikkel Krøigaard. Efficient and secure comparison for on-line auctions. In *ACISP '07*, volume 4586 of *Lecture Notes in Computer Science*, pages 416–430, Berlin, 2007. Springer-Verlag.

[DH76] Whitfield Diffie and Martin Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):655–654, 1976.

[DJ01] Ivan Damgård and Mads Jurik. A generalisation, a simplification and some applications of Paillier's probabilistic public-key system. In *Public Key Cryptography '01*, volume 1992 of *Lecture Notes in Computer Science*, pages 119–136, Berlin, 2001. Springer-Verlag.

[ECR08] ECRYPT. ECRYPT yearly report on algorithms and keysizes. Technical Report IST-2002-507932, European Network of Excellence in Cryptology, July 2008. `http://www.ecrypt.eu.org/documents/D.SPA.28-1.1.pdf`.

[ElG85] Taher ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In *Advances in Cryptology - CRYPTO '84*, volume 196 of *Lecture Notes in Computer Science*, pages 10–18, Berlin, 1985. Springer-Verlag.

[FFS87] Uriel Feige, Amos Fiat, and Adi Shamir. Zero-knowledge proofs of identity. In *Proc. ACM Symposium on Theory of Computing '87*, pages 210–217, New York, 1987. ACM.

[FH96] Matthew Franklin and Stuart Haber. Joint encryption and message-efficient secure computation. *Journal of Cryptology*, 9(4):217–232, January 1996.

[Fis01] Marc Fischlin. A cost-effective pay-per-multiplication comparison method for millionaires. In *Topics in Cryptology - CT-RSA 2001*, volume 2020 of *Lecture Notes in Computer Science*, pages 457–472, Berlin, 2001. Springer-Verlag.

[FJ06] Strange From and Thomas Jakobsen. Secure multi-party computation on integers. Master's thesis, University of Århus, Århus, 2006. `http://www.cs.au.dk/~tpj/uni/thesis/report.pdf`.

[FMY98] Yair Frankel, Philip MacKenzie, and Moti Yung. Robust efficient distributed RSA-key generation. In *Proc. ACM Symposium on Theory of Computing '98*, pages 663–672, New York, 1998. ACM.

[FO97] Eiichiro Fujisaki and Tatsuaki Okamoto. Statistical zero knowledge protocols to prove modular polynomial relations. In *Advances in Cryptology - CRYPTO '97*, volume 1294 of *Lecture Notes in Computer Science*, pages 16–30, Berlin, 1997. Springer-Verlag.

[FS87] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *Advances in Cryptology - CRYPTO '86*, volume 263 of *Lecture Notes in Computer Science*, pages 186–194, Berlin, 1987. Springer-Verlag.

[FS90] Uriel Feige and Adi Shamir. Witness indistinguishable and witness hiding protocols. In *Proc. ACM Symposium on Theory of Computing '90*, pages 416–426, New York, 1990. ACM.

[Gil99] Niv Gilboa. Two party RSA key generation. In *Advances in Cryptology - CRYPTO '99*, volume 1666 of *Lecture Notes in Computer Science*, pages 785–786, Berlin, 1999. Springer-Verlag.

[GJKR99] Rosario Gennaro, Stanislaw Jarecki, Hugo Krawczyk, and Tal Rabin. Secure distributed key generation for discrete-log based cryptosystems. In *Advances in Cryptology - EUROCRYPT '99*, volume 1592 of *Lecture Notes in Computer Science*, pages 295–310, Berlin, 1999. Springer-Verlag.

[GL90] Shafi Goldwasser and Leonid Levin. Fair computation of general functions in presence of immoral majority. In *Advances in Cryptology - CRYPTO '90*, volume 537 of *Lecture Notes in Computer Science*, pages 77–93, Berlin, 1990. Springer-Verlag.

[GM84] Shafi Goldwasser and Silvio Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28(2):270–299, April 1984.

[GMR85] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof-systems. In *Proc. ACM Symposium on Theory of Computing '85*, pages 291–304, New York, 1985. ACM.

[GMR88] Shafi Goldwasser, Silvio Micali, and Ronald Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM Journal of Computing*, 17(2):281–308, 1988.

[GMW87] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game or a completeness theorem for protocols with honest majority. In *Proc. ACM Symposium on Theory of Computing '87*, pages 218–229, New York, 1987. ACM.

[GSV07] Juan Garay, Berry Schoenmakers, and José Villegas. Practical and secure solutions for integer comparison. In *Public Key Cryptography '07*, volume 4450 of *Lecture Notes in Computer Science*, pages 330–342, Berlin, 2007. Springer-Verlag.

[HSSV09] Sebastiaan de Hoogh, Berry Schoenmakers, Boris Skoric, and José Villegas. Verifiable rotation of homomorphic encryptions. In *Public Key Cryptography '09*, volume 5443 of *Lecture Notes in Computer Science*, pages 393–410, Berlin, 2009. Springer-Verlag.

[JJ00] Markus Jakobsson and Ari Juels. Mix and match: Secure function evaluation via ciphertexts. In *Advances in Cryptology - ASIACRYPT '00*, volume 1976 of *Lecture Notes in Computer Science*, pages 162–177, Berlin, 2000. Springer-Verlag.

[JLO97] Ari Juels, Michael Luby, and Rafail Ostrovsky. Security of blind digital signatures. In *Advances in Cryptology - CRYPTO '97*, volume 1294 of *Lecture Notes in Computer Science*, pages 150–164, Berlin, 1997. Springer-Verlag.

[JS06] Ari Juels and Madhu Sudan. A fuzzy vault scheme. *Designs, Codes and Cryptography*, 38(2):237–257, 2006.

[Jur03] Mads Jurik. *Extensions to the Paillier Cryptosystem with Applications to Cryptological Protocols*. PhD thesis, University of Århus, Århus, 2003.

[Knu98] Donald Knuth. *Art of Computer Programming, Volume 3: Sorting and Searching*. Addison-Wesley Professional, 1998.

[LP04] Yehuda Lindell and Benny Pinkas. A proof of Yao's protocol for secure two-party computation. *Electronic Colloquium on Computational Complexity (ECCC)*, (63), 2004.

[LRSW99] Anna Lysyanskaya, Ronald Rivest, Amit Sahai, and Stefan Wolf. Pseudonym systems. In *Selected Areas in Cryptography '99*, volume 1758 of *Lecture Notes in Computer Science*, pages 184–199, Berlin, 1999. Springer-Verlag.

[LT05] Hsiao-Ying Lin and Wen-Guey Tzeng. An efficient solution to the millionaires' problem based on homomorphic encryption. In *Applied Cryptography and Network Security*, volume 3531 of *Lecture Notes in Computer Science*, pages 456–466, Berlin, 2005. Springer-Verlag.

[Lys02] Anna Lysyanskaya. *Signature Schemes and Applications to Cryptographic Protocol Design*. PhD thesis, Massachusetts Institute of Technology, Cambridge, Massachusetts, 2002.

[MR92] Silvio Micali and Phillip Rogaway. Secure computation (abstract). In *Advances in Cryptology - CRYPTO '91*, volume 576 of *Lecture Notes in Computer Science*, pages 392–404, Berlin, 1992. Springer-Verlag.

[MRS87] Silvio Micali, Charles Rackoff, and Bob Sloan. The notion of security for probabilistic cryptosystems. In *Advances in Cryptology - CRYPTO '86*, volume 263 of *Lecture Notes in Computer Science*, pages 381–392, Berlin, 1987. Springer-Verlag.

[Pai99] Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *Advances in Cryptology - EUROCRYPT '99*, volume 1592 of *Lecture Notes in Computer Science*, pages 223–238, Berlin, 1999. Springer-Verlag.

[Ped91] Torben Pedersen. A threshold cryptosystem without a trusted party. In *Advances in Cryptology - EUROCRYPT '91*, volume 547 of *Lecture Notes in Computer Science*, pages 522–526, Berlin, 1991. Springer-Verlag.

[Ped92] Torben Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In *Advances in Cryptology - CRYPTO '91*, volume 576 of *Lecture Notes in Computer Science*, pages 129–140, Berlin, 1992. Springer-Verlag.

[Pol74] John Pollard. Theorems of factorization and primality testing. *Mathematical Proceedings of the Cambridge Philosophical Society*, 76(3):521–528, November 1974.

[PS98] Guillaume Poupard and Jacques Stern. Generation of shared RSA keys by two parties. In *Advances in Cryptology - ASIACRYPT '98*, volume 1514 of *Lecture Notes in Computer Science*, pages 11–24, Berlin, 1998. Springer-Verlag.

[PS00] David Pointcheval and Jacques Stern. Security arguments for digital signatures and blind signatures. *Journal of Cryptology*, 13(3):361–396, December 2000.

[RSA78] Ron Rivest, Adi Shamir, and Leonard Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, 1978.

[Sch80] Jacob Schwartz. Fast probabilistic algorithms for verification of polynomial identities. *Journal of the ACM*, 27(4):701–717, 1980.

[Sch91] Claus Schnorr. Efficient signature generation by smart cards. *Journal of Cryptology*, 4(3):161–174, January 1991.

[Sch01] Claus Schnorr. Security of blind discrete log signatures against interactive attacks. In *Information and Communications Security*, volume 2229 of *Lecture Notes in Computer Science*, pages 1–12, Berlin, 2001. Springer-Verlag.

[Sch09] Berry Schoenmakers. Lecture notes 'cryptographic protocols'. Eindhoven, March 2009. `http://www.win.tue.nl/~berry/2WC13/LectureNotes.pdf`.

[Sec08] SecureSCM. Secure computation models and frameworks. Technical Report D9.1, SecureSCM, July 2008. `http://www.securescm.org`.

[Sha79] Adi Shamir. How to share a secret. *Communications of the ACM*, 22(11):612–613, 1979.

[Sho04] Victor Shoup. Sequences of games: A tool for taming complexity in security proofs. Cryptology ePrint Archive, Report 2004/332, 2004. `http://eprint.iacr.org/2004/332`.

[ST04] Berry Schoenmakers and Pim Tuyls. Practical two-party computation based on the conditional gate. In *Advances in Cryptology - ASIACRYPT '04*, volume 3329 of *Lecture Notes in Computer Science*, pages 119–136, Berlin, 2004. Springer-Verlag.

[ST06] Berry Schoenmakers and Pim Tuyls. Efficient binary conversion for Paillier encrypted values. In *Advances in Cryptology - EUROCRYPT '06*, volume 4004 of *Lecture Notes in Computer Science*, pages 522–537, Berlin, 2006. Springer-Verlag.

[Ste99] Jacques Stern. Upper bounding the security of a cryptosystem based on discrete log residues, 1999. Manuscript.

[TY98] Yiannis Tsiounis and Moti Yung. On the security of ElGamal based encryption. In *Public Key Cryptography '98*, volume 1431 of *Lecture Notes in Computer Science*, pages 117–134, Berlin, 1998. Springer-Verlag.

[Ver01] Eric Verheul. Self-blindable credential certificates from the weil pairing. In *Advances in Cryptology - ASIACRYPT '01*, volume 2248 of *Lecture Notes in Computer Science*, pages 533–551, Berlin, 2001. Springer-Verlag.

[Wag02] David Wagner. A generalized birthday problem. In *Advances in Cryptology - CRYPTO '02*, volume 2442 of *Lecture Notes in Computer Science*, pages 288–304, Berlin, 2002. Springer-Verlag.

[web09a] Health insurance portability and accountability act, 2009. `http://www.hhs.gov/ocr/privacy/index.html`.

[web09b] Website 'je echte leeftijd', 2009. `http://www.jeechteleeftijd.nl`.

[web09c] Philips Lifeline Personal Alert Service, 2009. `http://www.lifelinesys.com`.

[web09d] Website 'stichting kijkonderzoek', 2009. `http://www.kijkonderzoek.nl`.

[WS07] Jiang Wu and Douglas Stinson. An efficient identification protocol and the knowledge-of-exponent assumption. Cryptology ePrint Archive, Report 2007/479, 2007. `http://eprint.iacr.org/2007/479`.

[WS08] Jiang Wu and Douglas Stinson. On the security of the ElGamal encryption scheme and Damgård's variant. Cryptology ePrint Archive, Report 2008/200, 2008. `http://eprint.iacr.org/2008/200`.

[Yao82] Andrew Yao. Protocols for secure computations. In *Proc. 23rd IEEE Symposium on Foundations of Computer Science*, pages 160–164. IEEE Computer Society, 1982.

[Yao86] Andrew Yao. How to generate and exchange secrets. In *Proc. 27th IEEE Symposium on Foundations of Computer Science*, pages 162–167. IEEE Computer Society, 1986.

[Zip79] Richard Zippel. Probabilistic algorithms for sparse polynomials. In *EUROSAM '79*, volume 72 of *Lecture Notes in Computer Science*, pages 216–226, Berlin, 1979. Springer-Verlag.

Index

A. Heuristic analysis of Assumption 6.4.1

In this appendix, we argue why Assumption 6.4.1 should hold. This is done as follows: firstly, in Section A.1 a proposition is introduced. We call this proposition the 'separation proposition', and use it to make the analysis of the assumption easier to understand. The proposition allows us to split the problem into smaller problems. After the proposition we exemplify what the proposition concretely means for the coming argumentation. Then the actual analysis is given in Section A.2. This is done in several sub-claims that together constitute the whole analysis.

A.1 Separation proposition

The proposition relies on the Schwartz-Zippel lemma, independently proven by Zippel and Schwartz [Zip79, Sch80]. This notation is taken from [HSSV09].

Lemma A.1.1 (Schwartz-Zippel). *Let p be an n-variate polynomial of degree $d \geq 0$ over $\mathbb{Z}_q$ for some q. Let $x_1, \dots, x_n \in_R \mathbb{Z}_q$. Then, the probability that $p(x_1, \dots, x_n) = 0$ is at most d/q.*

We are now ready to introduce and prove the separation proposition.

Proposition A.1.2 (Separation proposition). *Given a group $G = \langle g \rangle$ or order q. For $k, l \in \mathbb{N}$, given l* different *functions $\phi_i : \mathbb{Z}_q^k \to \mathbb{Z}_q$ defined as*

$$\phi_i(R_1, \dots, R_k) = R_1^{\alpha_{i1}} \cdots R_k^{\alpha_{ik}} \bmod q,$$

with $\alpha_{i1} \in \{0, 1\}$. Without loss of generality we assume $\alpha_{i1} = 1$ for $i = 1, \dots, l'$ and $\alpha_{i1} = 0$ for $i = l' + 1, \dots, l$. We require that

$$\max_{i=1}^{l'} \deg \phi_i =: d \ll q, \tag{1}$$

where $\deg \phi_i = \sum_{j=1}^{k} \alpha_{ij}$ is the degree of the polynomial ϕ_i. Now, for $r_1, \dots, r_k \in_R \mathbb{Z}_q$ let $h_j = g^{r_j}$ ($j = 1, \dots, k$) and $g_i = g^{\phi_i(r_1, \dots, r_k)}$ ($i = 1, \dots, l$). Suppose a forger $\mathcal{F}$ is given $(h_j)_{j=1}^{k}$ and $(g_i, \phi_i)_{i=1}^{l}$ for $r_1, \dots, r_k \in_R \mathbb{Z}_q$, and can find a non-trivial tuple $(x_1, \dots, x_l)$ satisfying:

$$g_1^{x_1} \dots g_l^{x_l} = 1, \tag{2}$$

then, under the DL assumption this tuple satisfies

$$g_1^{x_1} \cdots g_{l'}^{x_{l'}} = 1 \text{ and } g_{l'+1}^{x_{l'+1}} \cdots g_l^{x_l} = 1. \tag{3}$$

Proof. Suppose $\mathcal{F}$ can find a non-trivial tuple $(x_1, \dots, x_l)$ satisfying (2) such that (3) does not hold, with probability ε. We construct a simulator $\mathcal{S}$ that uses $\mathcal{F}$ to solve the discrete log problem on input $g, h = g^x$. This simulator operates as follows:

1. The simulator takes $\rho_1, \ldots, \rho_k, \sigma \in_R \mathbb{Z}_q$. He sets $h_1 = g^{\rho_1} h^{\sigma}$ and $h_j = g^{\rho_j}$ for $j = 2, \ldots, k$, and computes the g_i as

$$g_i = h_1^{\rho_2^{\alpha_{i2}} \cdots \rho_k^{\alpha_{ik}}} \quad \text{for } i = 1, \ldots, l',$$
$$g_i = g^{\rho_2^{\alpha_{i2}} \cdots \rho_k^{\alpha_{ik}}} \quad \text{for } i = l'+1, \ldots, l.$$

   He sends $(h_j)_{j=1}^k$ and $(g_i, \phi_i)_{i=1}^l$ to $\mathcal{F}$. Notice that this setup exactly corresponds to setting $r_1 = \rho_1 + \sigma x$ and $r_j = \rho_j$ for $j = 2, \ldots, k$. In particular, for $i = 1, \ldots, l'$ we have

$$g_i = (g^{\rho_1} h^{\sigma})^{\rho_2^{\alpha_{i2}} \cdots \rho_k^{\alpha_{ik}}} = g^{(\rho_1 + \sigma x)\rho_2^{\alpha_{i2}} \cdots \rho_k^{\alpha_{ik}}};$$

2. The forger outputs a tuple $(x_1, \ldots, x_l)$ satisfying (2) but not (3);
3. If $\sigma \sum_{i=1}^{l'} \rho_2^{\alpha_{i2}} \cdots \rho_k^{\alpha_{ik}} x_i = 0$, the simulator halts. Otherwise, he computes

$$x \leftarrow \left(\sum_{i=1}^{l} \rho_1^{\alpha_{i1}} \rho_2^{\alpha_{i2}} \cdots \rho_k^{\alpha_{ik}} x_i \right) \left(-\sigma \sum_{i=1}^{l'} \rho_2^{\alpha_{i2}} \cdots \rho_k^{\alpha_{ik}} x_i \right)^{-1} \bmod q,$$

   and outputs the result.

By the condition in phase 3, x exists. Remains to prove that this value satisfies $g^x = h$. But:

$$\begin{aligned}
g^{\sum_{i=1}^{l} \rho_1^{\alpha_{i1}} \rho_2^{\alpha_{i2}} \cdots \rho_k^{\alpha_{ik}} x_i} &= g^{\sum_{i=1}^{l'} \rho_1 \rho_2^{\alpha_{i2}} \cdots \rho_k^{\alpha_{ik}} x_i} g^{\sum_{i=l'+1}^{l} \rho_2^{\alpha_{i2}} \cdots \rho_k^{\alpha_{ik}} x_i} \\
&= g^{\sum_{i=1}^{l'} \rho_1 \rho_2^{\alpha_{i2}} \cdots \rho_k^{\alpha_{ik}} x_i} \prod_{i=l'+1}^{l} g_i^{x_i} \\
&= g^{\sum_{i=1}^{l'} (\rho_1 + \sigma x) \rho_2^{\alpha_{i2}} \cdots \rho_k^{\alpha_{ik}} x_i} g^{\sum_{i=1}^{l'} -\sigma x \rho_2^{\alpha_{i2}} \cdots \rho_k^{\alpha_{ik}} x_i} \prod_{i=l'+1}^{l} g_i^{x_i} \\
&= \left(\prod_{i=1}^{l} g_i^{x_i} \right) g^{\sum_{i=1}^{l'} -\sigma x \rho_2^{\alpha_{i2}} \cdots \rho_k^{\alpha_{ik}} x_i} \\
&= 1 \cdot (g^x)^{-\sigma \sum_{i=1}^{l'} \rho_2^{\alpha_{i2}} \cdots \rho_k^{\alpha_{ik}} x_i},
\end{aligned}$$

from which $x = \left(\sum_{i=1}^{l} \rho_1^{\alpha_{i1}} \rho_2^{\alpha_{i2}} \cdots \rho_k^{\alpha_{ik}} x_i \right) \left(-\sigma \sum_{i=1}^{l'} \rho_2^{\alpha_{i2}} \cdots \rho_k^{\alpha_{ik}} x_i \right)^{-1}$ directly follows.
We now determine the success probability of $\mathcal{S}$. As the setup in phase 1 is indistinguishable from the distribution of the real, prescribed, case, the transition from phase 1 to phase 3 takes place with probability ε. For the success probability of phase 3, observe that there exists an $i \in \{1, \ldots, l'\}$ such that $x_i \neq 0$, as (3) does not hold, and therefore

$$p(\sigma, \rho_2, \ldots, \rho_k) = \sigma \sum_{i=1}^{l'} \rho_2^{\alpha_{i2}} \cdots \rho_k^{\alpha_{ik}} x_i$$

is a well-defined polynomial (in terms of Lemma A.1.1) of degree $\leq d \ll q$ by (1). Therefore, as $\sigma, \rho_2, \ldots, \rho_k \in_R \mathbb{Z}_q$ the simulator halts in phase 3 with probability at most d/q. Concluding, the overall success probability of the simulator is at least $(1 - d/q)\varepsilon$. $\square$

Clearly, if a forger would only know the values $(h_j)_{j=1}^k$ and not all values $(g_i)_{i=1}^l$, forging becomes even more difficult. We give a small example on how to apply this proposition.

Example A.1.3. Let $G = \langle g \rangle$ be a group of prime order q, and consider $r_1, r_2, r_3 \in_R \mathbb{Z}_q$. Define $h_j = g^{r_j}$ for $j = 1, 2, 3$ and $g_1 = g^{r_1 r_2}$, $g_2 = g^{r_1 r_3^2}$ and $g_3 = g^{r_2 r_3^3}$. Suppose that a forger is given $(h_j)_{j=1}^3$ and $(g_i)_{i=1}^3$, and can find a, b, c such that $g_1^a g_2^b g_3^c = 1$. Then, Proposition A.1.2 on $r_1 \in_R \mathbb{Z}_q$ implies that $g_1^a g_2^b = 1$ and $g_3^c = 1$ should hold. Applying Proposition A.1.2 on $g_1^a g_2^b = 1$ for $r_2 \in_R \mathbb{Z}_q$ results in $g_1^a = 1$ and $g_2^b = 1$. This leads to $a = b = c = 0$.

It now becomes clear that this proposition allows us to split equations of the form (2) with l large. For instance for $r_1, r_2, r_3, r_4 \in_R \mathbb{Z}_q$, the following equation looks complicated, but consecutive application of the separation proposition on r_1, r_2, r_3, r_4 eventually leaves $x_1 = \cdots = x_9 = 0$.

$$(g^{r_1r_2r_3r_4})^{x_1}(g^{r_1r_2r_4^3})^{x_2}(g^{r_1r_2r_3})^{x_3}(g^{r_1r_3^2r_4^4})^{x_4}(g^{r_2r_3r_4})^{x_5}(g^{r_2r_4^7})^{x_6}(g^{r_2r_3})^{x_7}(g^{r_3})^{x_8}(g^{r_4^9})^{x_9} = 1.$$

Remark A.1.4. The restriction in Proposition A.1.2 that $\alpha_{i1} \in \{0, 1\}$ excludes the case of finding a representation over (g, g^x, g^{x^2}) with $x \in_R \mathbb{Z}_q$. However, it can be shown that it is hard for a forger to find a, b, c such that $g^a(g^x)^b(g^{x^2})^c = 1$ if he only has knowledge of g and g^x. The reduction relies on the fact that finding g^{x^2} on input of (g, g^x) is hard [BDZ03]: a forger that is able to output (a, b, c) such that $(g^{x^2})^a(g^x)^b g^c = 1$ can be used to find g^{x^2} as follows: If $a = 0$, we have $x = -cb^{-1}$, and if $a \neq 0$ we obtain $g^{x^2} = (g^x)^{-ba^{-1}} g^{-ca^{-1}}$.

A.2 Heuristic analysis of Assumption 6.4.1

In this section, we analyze several sub-claims, which in turn allow us to heuristically analyze Assumption 6.4.1. For ease of presentation we recall this proposition.

Assumption 6.4.1. If $\mathcal{U}'$ produces, after $K \geq 0$ arbitrarily interleaving executions of the protocol in Figure 6.4 on $\left(c^*_{ji}\right)_{i=1}^{l-1}$ $(j = 1, \ldots, K)$ a tuple $(h', (c'_i)_{i=1}^{l}, \alpha_1, z', (z'_i)_{i=1}^{l}, c'_0, r'_0)$, then either this tuple does not satisfy (6.2), or with overwhelming probability there exists a $j \in \{1, \ldots, K\}$ such that

$$\mathcal{U}' \text{ knows values } (\beta_i)_{i=1}^{l} \text{ such that } (c'_i)_{i=1}^{l} = \left(c_{ji}(g, f)^{\beta_i}\right)_{i=1}^{l}, \qquad (6.4)$$

where $(c_{ji})_{i=1}^{l}$ is the list of encryptions coming from the first round of the j-th issuing execution.

A.2.1 *High-level approach*

Roughly, the analysis is as follows. At first, in Section A.2.2 we abstract the setting, and briefly summarize the goal of a possible forger $\mathcal{U}'$. We only consider two parallel executions on two attributes (for simplicity, we numbered elements in the two executions with subscripts j, see also Figure A.1). We consider an algebraic attack, where the forger exploits the algebraic properties of the group. Then, in Section A.2.3 we summarize the values that $\mathcal{U}'$ knows before initiating the attack (notice that the forger starts the attack *after* the first round in Figure A.1). The general attack of the forger is to choose the challenges c_{10}, c_{20} in the two issuing executions in a smart way, to learn the responses r_{10}, r_{20}, and use those responses to complete the choice of the values in the forged certificate $(h', c'_1, c'_2, \alpha_1, z', z'_1, z'_2, c'_0, r'_0)$. To this end, the forger needs to decide on the values $(h', z', (c'_i, z'_i)_{i=1}^{2})$ before he computes c'_0, as this value is the outcome of a cryptographic hash function. The complete analysis mainly relies on the property that this value c'_0, as well as the responses in the two issuing executions, look uniformly random to the forger $\mathcal{U}'$. As the forger learns the responses *after* he chooses the challenges, and as he learns c'_0 *after* he chooses the other values in the certificate (except for r'_0), this property turns out to reduce $\mathcal{U}'$'s possible choices. In particular, in order to succeed with non-negligible probability, the forger cannot do anything else than setting $h' = h_2^{\alpha_1}$ for some random α_1, where h_2 is coming from the second issuing execution (see Figure A.1). Notice that if the forger would follow the protocol, he would have set $h'_2 = h_2^{\alpha_1}$ as well. Consequently, in Section A.2.9 it turns out that the only way for the forger to decide on the encryptions c'_i, is by setting $c'_i \leftarrow c_{2i}(g, f)^{\beta_i}$ for known β_i $(i = 1, 2)$. But this statement contradicts the goal of the forger, namely coming up with a certificate such that (6.4) is not satisfied for any of the issuance executions.

A.2.2 *General setting*

We consider a group $G = \langle g \rangle$ of prime order q. Without loss of generality we only consider two attributes $(l = 2)$ and two parallel executions $(K = 2)$. For $j = 1, \ldots, K$, $\mathcal{U}'$ is issued a certificate by $\mathcal{CA}$ on c_j^*. To distinguish between the two different certificate issuance executions, all values obtained or generated by $\mathcal{CA}$ in the issuing executions are provided with a subscript $j \in \{1,2\}$. Figure A.1 shows the complete interaction between $\mathcal{U}'$ and $\mathcal{CA}$ used in this analysis. The goal of $\mathcal{U}'$ is to find a tuple $(h', c_1', c_2', \alpha_1, z', z_1', z_2', c_0', r_0')$ satisfying (6.2) for which (6.4) does not hold for all j, with non-negligible probability. We only consider algebraic attacks, where the forger exploits the group structure to decide on the values in the certificate. Furthermore, we only consider the case that $\mathcal{U}'$ decides on a value $h = (h')^{\alpha_1^{-1}}$ instead (the analysis also holds in case we just consider h'), and rename all the other variables to the non-prime names (except for z_1', z_2'). Summarizing, the goal of $\mathcal{U}'$ is to find a tuple $(h, z, c_1, c_2, z_1', z_2', c_0, r_0)$ satisfying

$$\begin{gathered} c_0 = \mathcal{H}(c_1, z_1', c_1^{r_0}(z_1')^{-c_0}; c_2, z_2', c_2^{r_0}(z_2')^{-c_0}; h, z, g^{r_0}h_0^{-c_0}, h^{r_0}z^{-c_0}) \\ \text{and } D(c_1^{y_1}c_2^{y_2})h_0 = h, \end{gathered} \tag{4}$$

without knowing values β_1, β_2 such that $c_i = c_{ji}(g,f)^{\beta_i}$ for both $i = 1,2$ for any $j \in \{1,2\}$.

$$\begin{array}{rcl}
\mathcal{U}' & & \mathcal{CA} \\
\text{(knows: } c_j^*) & & \text{(knows: } c_j^*) \\
 & & r_{j1}, r_{j2}, x_{j2} \in_R \mathbb{Z}_q \\
 & & c_{j1} \leftarrow c_j^* \cdot (g^{r_{j1}}, f_1 f^{r_{j1}}) \\
 & & c_{j2} \leftarrow (g^{r_{j2}}, g^{x_{j2}} f^{r_{j2}}) \\
 & & h_j \leftarrow D(c_{j1}^{y_1}c_{j2}^{y_2})h_0 \\
 & & z_j \leftarrow h_j^{x_0}, \quad \left(z_{ji} \leftarrow c_{ji}^{x_0}\right)_{i=1}^2 \\
 & & w_{j0} \in_R \mathbb{Z}_q \\
 & & a_{j0} \leftarrow g^{w_{j0}}, \quad b_{j0} \leftarrow h_j^{w_{j0}} \\
 & & \tilde{f}_j \leftarrow f^{w_{j0}}, \quad (e_{ji0} \leftarrow c_{ji}^{w_{j0}})_{i=1}^2 \\
 & \xleftarrow{h_j, z_j, (c_{ji}, z_{ji})_{i=1}^2; a_{j0}, b_{j0}, \tilde{f}_j, (e_{ji0})_{i=1}^2} & \\
\dots & \xrightarrow{\quad c_{j0} \quad} & \\
 & \xleftarrow{\quad r_{j0} \quad} & r_{j0} \leftarrow c_{j0}x_0 + w_{j0} \bmod q \\
a_{j0} \stackrel{?}{=} g^{r_{j0}}h_0^{-c_{j0}} & & \\
b_{j0} \stackrel{?}{=} h_j^{r_{j0}}z_j^{-c_{j0}} & & \\
\tilde{f}_j \stackrel{?}{=} f^{r_{j0}}\hat{f}^{-c_{j0}} & & \\
(e_{ji0} \stackrel{?}{=} c_{ji}^{r_{j0}}z_{ji}^{-c_{j0}})_{i=1}^2 & & \\
\dots & &
\end{array}$$

Figure A.1: The setting for the heuristic analysis of Assumption 6.4.1 (executed in parallel for $j = 1,2$). Afterwards, $\mathcal{U}'$ outputs a forgery satisfying the properties explained in Section A.2.2.

A.2.3 *Initial knowledge of $\mathcal{U}'$*

In the issuance of the encrypted certificates in Figure A.1, the forger is not involved in the protocol until he obtains $(h_j, z_j, (z_{ji})_{i=1}^2; a_{j0}, b_{j0}, \tilde{f}_j, (e_{ji0})_{i=1}^2)$ for $j = 1,2$. After he obtains these values, his attack starts. Concretely, all values the forger learned up to this point are the following for $j = 1,2$:

- The public key $(g, h_0, f, \hat{f}, g_1, g_2, f_1)$ by the key generation algorithm of Section 6.3.1. These values satisfy, for $x_0, \lambda, y_1, y_2, \phi_1 \in_R \mathbb{Z}_q$:

$$h_0 = g^{x_0}, \qquad f = g^{\lambda}, \qquad \hat{f} = g^{\lambda x_0}, \qquad g_1 = g^{y_1}, \qquad g_2 = g^{y_2}, \qquad f_1 = g^{\phi_1};$$

- The values c_j^* encrypting attributes $x_j^* \in \{0,1\}$;
- The values (c_{j1}, c_{j2}) and h_j satisfying, with $r_{j1}, r_{j2}, x_{j2} \in_R \mathbb{Z}_q$:

$$c_{j1} = (g^{r_{j1}}, g^{x_j^*} f_1 f^{r_{j1}}), \qquad c_{j2} = (g^{r_{j2}}, g^{x_{j2}} f^{r_{j2}}), \qquad h_j = g_1^{x_j^*+\phi_1} g_2^{x_{j2}} h_0,$$

as well as all other values $(z_j, (z_{ji})_{i=1}^2; a_{j0}, b_{j0}, \tilde{f}_j, (e_{ji0})_{i=1}^2)$ from the protocol in Figure A.1.

Table A.1 summarizes the elements and their relationship to secret exponents x_0, w_{10} and w_{20}. We stress that the exponents of all elements in the table are (polynomial combinations of values) in $\{x_0, w_{10}, w_{20}, \phi_1, \lambda, y_1, y_2, r_{ji}, x_{j2}\} \subset_R \mathbb{Z}_q$, and these values are unknown to $\mathcal{U}'$. Note that $\mathcal{U}'$ indeed does not know $(c_j^*)^{x_0}$ although he knows $c_{j1}^{x_0}$ satisfying $c_{j1} \doteq c_j^*(1, f_1)$ (for $j = 1, 2$).

	g	f	g_1	g_2	f_1	c_1^*	c_{1i}	h_1	c_2^*	c_{2i}	h_2
expon. x_0:	h_0	$\hat{f}$	-	-	-	-	z_{1i}	z_1	-	z_{2i}	z_2
expon. w_{10}:	a_{10}	$\tilde{f}_1$	-	-	-	-	e_{1i0}	b_{10}	-	-	-
expon. w_{20}:	a_{20}	$\tilde{f}_2$	-	-	-	-	-	-	-	e_{2i0}	b_{20}

Table A.1: All values the forger $\mathcal{U}'$ knows at the beginning of his forging attempt.

A.2.4 *General design of attack*

Now, the basic strategy $\mathcal{U}'$ could have applied is that he constructs the challenges c_{10}, c_{20} in Figure A.1 in a smart way, in order to obtain r_{10}, r_{20} according to the issuing protocol, and use that knowledge to construct the certificate $(h, z, c_1, c_2, z_1', z_2', c_0, r_0)$, where he freely chooses the final r_0, and takes c_0 to be the outcome of the hash function such that the certificate satisfies (4). The same attack is used in the argumentation why [Bra99, Ass. 4.4.5] (Assumption 5.2.1) should hold.
So in Figure A.1, for $j = 1, 2$ he can choose c_{j0} arbitrarily, and will obtain values r_{j0} such that:

$$\begin{aligned} a_{j0} &= g^{r_{j0}} h_0^{-c_{j0}}, & b_{j0} &= h_j^{r_{j0}} z_j^{-c_{j0}}, \\ (e_{ji0} &= c_{ji}^{r_{j0}} z_{ji}^{-c_{j0}})_{i=1}^2, & \tilde{f}_j &= f^{r_{j0}} \hat{f}^{-c_{j0}}. \end{aligned} \tag{5}$$

The goal of $\mathcal{U}'$ is to find values $(h, z, c_1, c_2, z_1', z_2'; a_0, b_0, e_{10}, e_{20}; r_0)$ such that

$$a_0 = g^{r_0} h_0^{-c_0}, \qquad b_0 = h^{r_0} z^{-c_0}, \qquad (e_{i0} = c_i^{r_0} (z_i')^{-c_0})_{i=1}^2, \tag{6}$$

where c_0 is computed as $c_0 = \mathcal{H}(c_1, z_1', e_{10}; c_2, z_2', e_{20}; h, z, a_0, b_0)$. These values moreover need to satisfy the second equation in (4), such that (6.4) does not hold for any j. An important observation is that the values c_0, r_{10}, r_{20} can be considered to be uniformly and independently randomly distributed in $\mathbb{Z}_q$, as c_0 is the outcome of a cryptographic hash function and $r_{j0} = c_{j0} x_0 + w_{j0} \bmod q$ for random w_{j0}. The forger can choose r_0 freely, and therefore without loss of generality we assume this value to be chosen last.

A.2.5 *Intermediate summary and overview of further analysis*

Summarizing, we considered the possible attack in Section A.2.4. Briefly, the forger can query the $\mathcal{CA}$ on two arbitrarily chosen values c_{10}, c_{20} in order to obtain r_{10}, r_{20} satisfying (5). These r_{j0} are mutually independent. He needs to come up with values $(h, z, c_1, c_2, z_1', z_2'; a_0, b_0, e_{10}, e_{20}; r_0)$ satisfying (6), with c_0 the outcome of a cryptographic hash function. He will do so by setting all values except r_0 as algebraic combinations of group values of Table A.1, for instance $h = h_1^{\alpha_1} h_2^{\alpha_2}$ for arbitrarily chosen α_1, α_2, and he will obtain c_0 directly after he chooses all these values.
The rest of the analysis is as follows: firstly in Section A.2.6 the attack will be simplified. That

is, for a moment we will forget about the second and third equations of (6) and only consider the possible choice of a_0. Then in Section A.2.7 we extend this scenario so that also $b_0 = h^{r_0} z^{-c_0}$ is considered, and we will investigate the possible choices of b_0, h, z by $\mathcal{U}'$. Finally, in Section A.2.9 we extend to the last equation in (6) and the second equation of (4). It then turns out that it *is* hard for $\mathcal{U}'$ to forge a certificate such that (6.4) does not hold for any j, as it becomes clear from the conclusion in Section A.2.10.

A.2.6 *Simplification to* $a_0 = g^{r_0} h_0^{-c_0}$ *of (6)*

In this simplification, $\mathcal{U}'$ needs to choose a_0 such that $a_0 = g^{r_0} h_0^{-c_0}$ holds, where $c_0 = \mathcal{H}(a_0)$ and r_0 is chosen arbitrarily. He can also choose challenges c_{j0} arbitrarily ($j = 1, 2$) to obtain responses r_{j0} satisfying (5).

Possible choices by $\mathcal{U}'$. Using Table A.1, the simplest way for $\mathcal{U}'$ to construct the value a_0 is by setting

$$a_0 = a_{10}^{\gamma_1} a_{20}^{\gamma_2} h_0^{\gamma_3} g^{\gamma_4}. \tag{7}$$

The reason that this is the simplest option follows from the separation proposition: the right-hand side of equation $a_0 = g^{r_0} h_0^{-c_0}$ only consists of elements g, h_0, and therefore the left-hand side should consist of elements only having these two terms as well (recall from (5) that $a_{j0} = g^{r_{j0}} h_0^{-c_{j0}}$ for $j = 1, 2$). However, in the equation $a_0 = g^{r_0} h_0^{-c_0}$ the forger can choose r_0 himself, and therefore we can without loss of generality ignore the term g^{γ_4} in (7).
So summarizing, the forger can arbitrarily choose $\gamma_1, \gamma_2, \gamma_3$. He can also arbitrarily choose c_{10}, c_{20}, as well as r_0. He obtains r_{j0} directly after he chooses c_{j0} ($j = 1, 2$). These r_{j0} are randomly and mutually independently distributed. $\mathcal{U}'$ obtains $c_0 = \mathcal{H}(a_0)$ directly after he chooses $\gamma_1, \gamma_2, \gamma_3$.

Interpretation of possible choices. Together with the first equations of (5) and (6), equation (7) implies $(g^{r_{10}} h_0^{-c_{10}})^{\gamma_1} (g^{r_{20}} h_0^{-c_{20}})^{\gamma_2} h_0^{\gamma_3} = g^{r_0} h_0^{-c_0}$, where $h_0 = g^{x_0}$. Applying the separation proposition on $x_0 \in_R \mathbb{Z}_q$ results in

$$r_{10}\gamma_1 + r_{20}\gamma_2 \equiv r_0 \bmod q, \tag{8}$$

$$-\gamma_3 + c_{10}\gamma_1 + c_{20}\gamma_2 \equiv c_0 \bmod q. \tag{9}$$

Analysis of possible choices. Suppose $\gamma_1 = \gamma_2 = 0$. Then (9) implies that $c_0 = -\gamma_3 \bmod q$, but this only happens with negligible probability $1/q$ as c_0 is uniformly random obtained after choosing γ_3. Thus $(\gamma_1, \gamma_2) \neq (0, 0)$. We now extend to the second equation in (6).

A.2.7 *Extension to* $b_0 = h^{r_0} z^{-c_0}$ *of (6)*

We extend the scenario to the second equation in (6). Now, in addition to a_0, the forger $\mathcal{U}'$ needs to choose values b_0, h, z such that $b_0 = h^{r_0} z^{-c_0}$ holds, where $c_0 = \mathcal{H}(a_0, b_0, h, z)$ and r_0 is chosen arbitrarily. He can also choose challenges c_{j0} arbitrarily ($j = 1, 2$) to obtain responses r_{j0} satisfying (5). We first consider the possible choices that $\mathcal{U}'$ has for the values h, z, b_0: firstly the construction of h is analyzed using (4), and then the constructions of z, b_0. Next, these possible choices will be interpreted.

Possible choice of h by $\mathcal{U}'$. The forger can decide on h by setting it as a combination of all values in Table A.1, that is by writing $h = g^{\alpha_1} h_0^{\alpha_2} \cdots$ for arbitrarily chosen constants $\alpha_1, \alpha_2, \ldots$. It however turns out that, using the second equation in (4), the possible choice of h can be simplified. In this paragraph we only focus on the form of h, the encryptions c_1, c_2 will be discussed later. The approach is as follows: we consider h to be an algebraic combination of all values in Table A.1, and then simplify h by applying the separation proposition and Remark A.1.4 to equation

$$h = D(c_1^{y_1}) D(c_2^{y_2}) h_0. \tag{10}$$

Firstly, we consider the separation proposition on r_{ji} for $j \in \{1,2\}$, $i \in \{1,2\}$. Notice that in Table A.1 r_{ji} only occurs in c_{ji}, z_{ji} and e_{ji0}, so for the left-hand side of (10) we are left with six terms (notice that $c_{ji} = (g^{r_{ji}}, g^{x_{ji}} f^{r_{ji}})$ results in of two terms). For simplicity, the left-hand side leaves (corresponding to the left equation in (3))

$$(g^{r_{ji}})^{\alpha_1}(f^{r_{ji}})^{\alpha_2}(g^{x_0 r_{ji}})^{\alpha_3}(f^{x_0 r_{ji}})^{\alpha_4}(g^{w_{j0} r_{ji}})^{\alpha_5}(f^{w_{j0} r_{ji}})^{\alpha_6}, \tag{11}$$

for arbitrarily chosen $\alpha_1, \ldots, \alpha_6$. These terms also occur on the right-hand side of the equation, but with an extra power y_i ($i = 1,2$). Then, the separation proposition on y_1 and y_2 leaves that the value in (11) equals 1. Applying the separation proposition in a straightforward manner on λ, x_0 and w_{j0} then implies that $\alpha_1 = \cdots = \alpha_6 = 0$. Concluding, for $j = 1,2$ and $i = 1,2$, c_{ji}, z_{ji}, e_{ji0} are *no* term of h. A similar argument holds for c_j^* ($j = 1,2$). This leaves

$$h = g^{\alpha_1} h_0^{\alpha_2} a_{10}^{\alpha_3} a_{20}^{\alpha_4} f^{\alpha_5} \hat{f}^{\alpha_6} \tilde{f}_1^{\alpha_7} \tilde{f}_2^{\alpha_8} g_1^{\alpha_9} g_2^{\alpha_{10}} h_1^{\alpha_{11}} z_1^{\alpha_{12}} b_{10}^{\alpha_{13}} h_2^{\alpha_{14}} z_2^{\alpha_{15}} b_{20}^{\alpha_{16}} f_1^{\alpha_{17}}, \tag{12}$$

for arbitrarily chosen $\alpha_1, \ldots, \alpha_{17}$.
Note that the exponent y_i ($i = 1,2$) may occur in (10) in squared form: encryption c_i may have term $g_i = g^{y_i}$, which in (10) results in a term $g^{y_i^2}$. However, the term $g^{y_i^3}$ does not occur in the equation as no element in Table A.1 has term $g^{y_i^2}$. Accordingly, we can apply Remark A.1.4 on (10) to eliminate all factors in (10) with a term y_i. In particular, the right-hand side of the equation becomes h_0. Using (12), this simplification leads to (recall that $h_j = g_1^{x_j^* + \phi_1} g_2^{x_{j2}} h_0$)

$$g^{\alpha_1} h_0^{\alpha_2} a_{10}^{\alpha_3} a_{20}^{\alpha_4} f^{\alpha_5} \hat{f}^{\alpha_6} \tilde{f}_1^{\alpha_7} \tilde{f}_2^{\alpha_8} h_0^{\alpha_{11}} (h_0^{x_0})^{\alpha_{12}} (h_0^{w_{10}})^{\alpha_{13}} h_0^{\alpha_{14}} (h_0^{x_0})^{\alpha_{15}} (h_0^{w_{20}})^{\alpha_{16}} f_1^{\alpha_{17}} = h_0,$$

or equivalently,

$$g^{\alpha_1} h_0^{\alpha_2 + \alpha_{11} + \alpha_{14} - 1} a_{10}^{\alpha_3} a_{20}^{\alpha_4} f^{\alpha_5} \hat{f}^{\alpha_6} \tilde{f}_1^{\alpha_7} \tilde{f}_2^{\alpha_8} (h_0^{x_0})^{\alpha_{12} + \alpha_{15}} (h_0^{w_{10}})^{\alpha_{13}} (h_0^{w_{20}})^{\alpha_{16}} f_1^{\alpha_{17}} = 1.$$

On this equality, the separation proposition on $w_{j0} \in_R \mathbb{Z}_q$ ($j = 1,2$) gives:

$$\begin{aligned}
1 &= (g^{w_{10}})^{\alpha_3} (g^{\lambda w_{10}})^{\alpha_7} (g^{x_0 w_{10}})^{\alpha_{13}}, \\
1 &= (g^{w_{20}})^{\alpha_4} (g^{\lambda w_{20}})^{\alpha_8} (g^{x_0 w_{20}})^{\alpha_{16}}, \\
1 &= g^{\alpha_1} (g^{x_0})^{\alpha_2 + \alpha_{11} + \alpha_{14} - 1} (g^{x_0^2})^{\alpha_{12} + \alpha_{15}} (g^{\lambda})^{\alpha_5} (g^{\lambda x_0})^{\alpha_6} (g^{\phi_1})^{\alpha_{17}}.
\end{aligned}$$

The separation proposition and Remark A.1.4 can be applied to these equations to conclude that all exponents need to be 0. In particular, using (12) the forger would have to choose arbitrarily chosen constants $\alpha_1, \ldots, \alpha_5$ such that

$$h = h_0^{1 - \alpha_1 - \alpha_2} h_1^{\alpha_1} h_2^{\alpha_2} g_1^{\alpha_3} g_2^{\alpha_4} z_1^{\alpha_5} z_2^{-\alpha_5}. \tag{13}$$

Possible choice of b_0, z by $\mathcal{U}'$. Using Table A.1, the simplest way for $\mathcal{U}'$ to construct the values b_0, z is by setting

$$b_0 = a_{10}^{\delta_1} a_{20}^{\delta_2} b_{10}^{\delta_3} b_{20}^{\delta_4} h_0^{\delta_5} g^{\delta_6} h_1^{\delta_7} h_2^{\delta_8} g_1^{\delta_9} g_2^{\delta_{10}}, \tag{14}$$
$$z = z_1^{\epsilon_1} z_2^{\epsilon_2} h_1^{\epsilon_3} h_2^{\epsilon_4} h_0^{\epsilon_5} g_1^{\epsilon_6} g_2^{\epsilon_7} g^{\epsilon_8}. \tag{15}$$

The reason why this is the simplest option is the same as for the choice of a_0 in (7). So summarizing, the forger can arbitrarily choose

$$\Gamma := \{\alpha_1, \ldots, \alpha_5, \gamma_1, \ldots, \gamma_3, \delta_1, \ldots, \delta_{10}, \epsilon_1, \ldots, \epsilon_8\}, \tag{16}$$

provided that $(\gamma_1, \gamma_2) \neq (0,0)$. He can also arbitrarily choose c_{10}, c_{20}, as well as r_0. He obtains r_{j0} directly after he chooses c_{j0} ($j = 1,2$). These r_{j0} are randomly and mutually independently distributed. $\mathcal{U}'$ obtains $c_0 = \mathcal{H}(a_0, b_0, h, z)$ directly after he chooses Γ.

Interpretation of possible choices. Substituting the equations in (5) into b_0 of (14), we obtain

$$\begin{aligned} b_0 &= (g^{r_{10}} h_0^{-c_{10}})^{\delta_1} (g^{r_{20}} h_0^{-c_{20}})^{\delta_2} (h_1^{r_{10}} z_1^{-c_{10}})^{\delta_3} (h_2^{r_{20}} z_2^{-c_{20}})^{\delta_4} h_0^{\delta_5} g^{\delta_6} h_1^{\delta_7} h_2^{\delta_8} g_1^{\delta_9} g_2^{\delta_{10}} \\ &= g^{r_{10}\delta_1 + r_{20}\delta_2 + \delta_6} h_0^{-c_{10}\delta_1 - c_{20}\delta_2 + \delta_5} g_1^{\delta_9} g_2^{\delta_{10}} h_1^{\delta_7 + r_{10}\delta_3} h_2^{\delta_8 + r_{20}\delta_4} z_1^{-c_{10}\delta_3} z_2^{-c_{20}\delta_4}. \end{aligned} \tag{17}$$

Now, substituting (13, 15, 17) into $b_0 h^{-r_0} z^{c_0} = 1$ results in:

$$\begin{aligned} & g^{r_{10}\delta_1 + r_{20}\delta_2 + \delta_6 + c_0\epsilon_8} h_0^{-c_{10}\delta_1 - c_{20}\delta_2 + \delta_5 - r_0(1-\alpha_1-\alpha_2) + c_0\epsilon_5} g_1^{\delta_9 - r_0\alpha_3 + c_0\epsilon_6} g_2^{\delta_{10} - r_0\alpha_4 + c_0\epsilon_7} \\ & \cdot h_1^{\delta_7 + r_{10}\delta_3 - r_0\alpha_1 + c_0\epsilon_3} h_2^{\delta_8 + r_{20}\delta_4 - r_0\alpha_2 + c_0\epsilon_4} z_1^{-c_{10}\delta_3 - r_0\alpha_5 + c_0\epsilon_1} z_2^{-c_{20}\delta_4 + r_0\alpha_5 + c_0\epsilon_2} = 1, \end{aligned} \tag{18}$$

where the values $(g, h_0, g_1, g_2, h_1, h_2, z_1, z_2)$ can be found in Table A.1. Consecutively applying the separation proposition on $y_2, x_{j2}, x_0 \in_R \mathbb{Z}_q$ $(j = 1, 2)$ results in the following equations (recall that $h_j = g_1^{x_j^* + \phi_1} g_2^{x_{j2}} h_0$ and $z_j = (h_j)^{x_0}$ with $g_2 = g^{y_2}$):

$$\delta_{10} - r_0\alpha_4 + c_0\epsilon_7 \equiv 0 \bmod q, \tag{19}$$

$$\delta_7 + r_{10}\delta_3 - r_0\alpha_1 + c_0\epsilon_3 \equiv 0 \bmod q, \tag{20}$$

$$\delta_8 + r_{20}\delta_4 - r_0\alpha_2 + c_0\epsilon_4 \equiv 0 \bmod q, \tag{21}$$

$$-c_{10}\delta_3 - r_0\alpha_5 + c_0\epsilon_1 \equiv 0 \bmod q, \tag{22}$$

$$-c_{20}\delta_4 + r_0\alpha_5 + c_0\epsilon_2 \equiv 0 \bmod q. \tag{23}$$

Back to (18), this leaves

$$1 = g^{r_{10}\delta_1 + r_{20}\delta_2 + \delta_6 + c_0\epsilon_8} h_0^{-c_{10}\delta_1 - c_{20}\delta_2 + \delta_5 - r_0(1-\alpha_1-\alpha_2) + c_0\epsilon_5} g_1^{\delta_9 - r_0\alpha_3 + c_0\epsilon_6},$$

on which the separation proposition easily results in (note that this is also implied by the DLREP Assumption 2.2.4)

$$r_{10}\delta_1 + r_{20}\delta_2 + \delta_6 + c_0\epsilon_8 \equiv 0 \bmod q, \tag{24}$$

$$-c_{10}\delta_1 - c_{20}\delta_2 + \delta_5 - r_0(1-\alpha_1-\alpha_2) + c_0\epsilon_5 \equiv 0 \bmod q, \tag{25}$$

$$\delta_9 - r_0\alpha_3 + c_0\epsilon_6 \equiv 0 \bmod q. \tag{26}$$

Analysis of possible choices. We analyzed the possible constructions of a_0, b_0, h, z, and concluded that the choices Γ of (16) need to satisfy (8-9) and (19-26). More abstractly, the forger can arbitrarily choose Γ, provided that $(\gamma_1, \gamma_2) \neq (0, 0)$. He can also arbitrarily choose c_{10}, c_{20}, as well as r_0. He obtains r_{j0} directly after he chooses c_{j0} $(j = 1, 2)$, and he obtains c_0 directly after he decides on Γ. All these values need to satisfy the mentioned equations. The values c_0, r_{10}, r_{20} are independently and uniformly randomly distributed, and this observation will be used to simplify equations (8-9, 19-26).

First of all, notice that if $\mathcal{U}'$ computes c_0 *after* he obtained r_{10}, r_{20}, he has only a negligible probability of success: as the r_{j0} are obtained *after* c_{j0}, this means that he computes c_0 *after* he knows Γ and c_{j0} $(j = 1, 2)$. But as c_0 is uniformly random, (9) only holds with probability $1/q$. Therefore, we assume that $\mathcal{U}'$ computes c_0 before he computes c_{20}, and without loss of generality we assume that also c_{10} is computed before c_{20}. In particular this means that $\mathcal{U}'$ obtains all his values in the following order: he chooses Γ and directly obtains c_0, then he takes c_{20} and directly obtains r_{20}. Finally, he fixes r_0. Note that in this approach $\mathcal{U}'$ clearly has a significant probability of success: indeed, following the protocol, that is by taking $c_{20} \leftarrow c_0$ and $r_0 \leftarrow r_{20}$, is included in this approach. We will show that the forger cannot do better than following the second protocol execution, and that he must set $c_i = c_{2i}(g, f)^{\beta_i}$ for known β_i (for $i = 1, 2$), which in particular means that (6.4) holds for $j = 2$.

Until now, we have not yet considered how $\mathcal{U}'$ chooses the values c_{10}, r_{10}. In the following, we elaborate on the above described strategy, and consider the values c_{10}, r_{10} as well. More precisely, the forger has the following two possibilities (we denote by '$a \overset{R}{\to} b$' that b is random and obtained by $\mathcal{U}'$ directly after choosing a):

A. c_0 is computed *before* choosing c_{10}, i.e. he obtains the values in the following order:

$$\Gamma \overset{R}{\to} c_0 \to c_{10} \overset{R}{\to} r_{10} \to c_{20} \overset{R}{\to} r_{20} \to r_0.$$

As $(\gamma_1, \gamma_2) \neq (0,0)$, without loss of generality (by symmetry) we can assume that $\gamma_2 \neq 0$;

B. c_0 is computed *after* obtaining r_{10}. In particular that means that the forger decides on Γ after obtaining r_{10} (as c_0 is obtained *directly* after choosing Γ). More concretely, he obtains the values in the following order:

$$c_{10} \overset{R}{\to} r_{10} \to \Gamma \overset{R}{\to} c_0 \to c_{20} \overset{R}{\to} r_{20} \to r_0.$$

Now, suppose $\gamma_2 = 0$, then by (9) we have $-\gamma_3 + c_{10}\gamma_1 \equiv c_0 \bmod q$, which happens with probability $1/q$ as c_0 is random and obtained *after* c_{10} and Γ.

So for both scenarios we require $\gamma_2 \neq 0$. We will now further analyze the two scenarios A and B, and in particular what the equations (8-9, 19-26) imply for the choices by $\mathcal{U}'$ of the values Γ, c_{10}, c_{20} and r_0. Therefore, we make the following steps, which apply to both scenarios:

i. Merging (8) with (20, 21) gives:

$$\delta_7 + r_{10}(\delta_3 - \alpha_1\gamma_1) - r_{20}\gamma_2\alpha_1 + c_0\epsilon_3 \equiv 0 \bmod q,$$
$$\delta_8 + r_{20}(\delta_4 - \alpha_2\gamma_2) - r_{10}\gamma_1\alpha_2 + c_0\epsilon_4 \equiv 0 \bmod q.$$

Notice that both equations only involve Γ, c_0, r_{10} and r_{20}. But in both scenarios, r_{20} is a random and independently distributed value obtained *after* Γ, c_0 and r_{10} are fixed. Suppose that in the first equation $\gamma_2\alpha_1 \not\equiv 0 \bmod q$. Then, this equation is equivalent to $r_{20} \equiv (\delta_7 + r_{10}(\delta_3 - \alpha_1\gamma_1) + c_0\epsilon_3)(\gamma_2\alpha_1)^{-1} \bmod q$, of which the right-hand side is fixed when r_{20} is computed. But as r_{20} is random, this equation only holds with negligible probability $1/q$. Thus, to have a non-negligible probability of success, $\mathcal{U}'$ must take Γ such that $\gamma_2\alpha_1 \equiv 0 \bmod q$. Similarly, he needs $\delta_4 - \alpha_2\gamma_2 \equiv 0 \bmod q$. Notice that $\gamma_2\alpha_1 \equiv 0 \bmod q$ implies that $\alpha_1 = 0$ as $\gamma_2 \neq 0$. In scenario A, for similar reasons we now moreover obtain that $\gamma_1\alpha_2 \equiv 0 \bmod q$;

ii. Similarly to step i, using (8) and the fact that $\gamma_2 \neq 0$, equations (26, 19, 22) result in $\alpha_3 = \alpha_4 = \alpha_5 = 0$, and (24) gives $\delta_2 = 0$;

iii. Merging (8) with (25) now gives:

$$-c_{10}\delta_1 + \delta_5 - (r_{10}\gamma_1 + r_{20}\gamma_2)(1 - \alpha_1 - \alpha_2) + c_0\epsilon_5 \equiv 0 \bmod q,$$

where we use that $\delta_2 = 0$ by step ii. For similar reasons as in step i, and as $\gamma_2 \neq 0$, this implies that $1 - \alpha_1 - \alpha_2 \equiv 0 \bmod q$. In particular, $\alpha_2 = 1$ since $\alpha_1 = 0$.

Summarizing, $\alpha_1 = \alpha_3 = \alpha_4 = \alpha_5 = 0$ and $\alpha_2 = 1$, which by (13) results in $h = h_2$. Moreover, as $\alpha_2 = 1$, step i gives $\delta_4 = \gamma_2$. As $\alpha_5 = 0$, this combines (9) and (23) into

$$c_0(\epsilon_2 - 1) \equiv \gamma_3 - c_{10}\gamma_1 \bmod q. \tag{27}$$

We now analyze this equation for the two cases A and B separately.

A. As $\alpha_2 = 1$, we have $\gamma_1 = 0$ by step i. Therefore, (27) simplifies to $c_0(\epsilon_2 - 1) \equiv \gamma_3 \bmod q$ from which, as c_0 is obtained as a random value *after* Γ is fixed, follows that $\epsilon_2 = 1$. In particular this implies that the forger needs to set $c_{20} \leftarrow c_0\gamma_2^{-1} \bmod q$ and $r_0 \leftarrow r_{20}\gamma_2 \bmod q$;

B. As c_0 is random, and computed *after* the forger chooses c_{10} and Γ, we again require $\epsilon_2 = 1$. So in order for the forger to succeed with non-negligible probability, he needs to set $c_{20} \leftarrow c_0\gamma_2^{-1} \bmod q$ and $r_0 \leftarrow r_{20}\gamma_2 + \delta \bmod q$ for some $\delta \in \mathbb{Z}_q$ (recall that $\mathcal{U}'$ chooses Γ *after* he obtains r_{10}, hence $r_{10}\gamma_1$ is just some value arbitrarily chosen by $\mathcal{U}'$).

Thus, the forger has more degrees of freedom in scenario B. He has to take his values such that $h = h_2$, $c_0 = c_{20}\gamma \bmod q$ and $r_0 = r_{20}\gamma + \delta \bmod q$ for some $\gamma, \delta \in \mathbb{Z}_q$ with $\gamma \neq 0$, both chosen *before* computing c_0.

A.2.8 *Intermediate summary and overview of further analysis*

Summarizing, to have non-negligible success probability, $\mathcal{U}'$ must take:

$$h = h_2, \qquad c_0 = c_{20}\gamma \bmod q, \qquad r_0 = r_{20}\gamma + \delta \bmod q, \tag{28}$$

for some $\gamma, \delta \in \mathbb{Z}_q$ with $\gamma \neq 0$, chosen *before* computing c_0. But this still leaves the forger $\mathcal{U}'$ freedom. However, as explained in Section A.2.5 we are still considering a simplified scenario where we left out the third equation in (6): the forger also has to choose encryptions (c_i, z'_i, e_{i0}) satisfying $e_{i0} = c_i^{r_0}(z'_i)^{-c_0}$ (for $i = 1, 2$), and these encryptions moreover need to satisfy the second equation of (4), i.e. $h = D(c_1^{y_1}c_2^{y_2})h_0$, and have to be decided upon *before* c_0 is computed. In Section A.2.9 we will consider the possibilities that $\mathcal{U}'$ has for choosing these values. We recall that $\mathcal{U}'$'s original goal is to obtain a certificate such that (6.4) does not hold for any j.

A.2.9 *Extension to* $e_{i0} = c_i^{r_0}(z'_i)^{-c_0}$ *of (6)*

Until Section A.2.7, we considered the possibilities that $\mathcal{U}'$ has for choosing the values h and z. We have shown in the previous sections that the only possibility for $\mathcal{U}'$ is to choose h, c_{20} and r_0 as in (28), where r_{20} and c_0 are random values and obtained after γ, δ are fixed. We will now consider the possible choices for the values $(c_i, z'_i, e_{i0})_{i=1}^2$ such that (6) and the second equation of (4) are satisfied.

Interpretation of possible choices. Independent of the choice of $(c_i, z'_i, e_{i0})_{i=1}^2$, by virtue of (28) these values need to satisfy $e_{i0} = (c_i^{r_{20}}(z'_i)^{-c_{20}})^\gamma c_i^\delta$ for some $\gamma, \delta \in \mathbb{Z}_q$ with $\gamma \neq 0$, chosen by $\mathcal{U}'$ before computing c_0 and thus before fixing c_{20} and obtaining r_{20}. As in the issuing executions in Figure A.1, $\mathcal{CA}$ constructed $r_{20} \leftarrow w_{20} + c_{20}x_0 \bmod q$, this equation implies after simplification

$$e_{i0} = (c_i^{w_{20}})^\gamma (c_i^{x_0}(z'_i)^{-1})^{c_0} c_i^\delta.$$

But the values $(c_i, z'_i, e_{i0}, \gamma, \delta)$ need to be known before c_0 is computed. Therefore, the choice by the forger really needs to satisfy that $c_i^{x_0}(z'_i)^{-1} = 1$ and that $c_i^{w_{20}}$ is known before c_0 is computed (as $\gamma \neq 0$), for both $i = 1, 2$. So independent of the construction of c_i, $\mathcal{U}'$ at least needs to know $c_i^{x_0}$ and $c_i^{w_{20}}$. The problem of constructing such c_i is related to the KEA assumption [Dam92, WS07, WS08]. Briefly, the KEA assumption states that if a forger is given $(x_0, x_0^a, \ldots, x_n, x_n^a)$ for $x_0, \ldots, x_n \in G$ and $a \in_R \mathbb{Z}_q$, and outputs a pair (x, x^a) for $x \in G$, then with overwhelming probability he knows constants $\omega_0, \ldots, \omega_n$ such that $x = \prod_{i=0}^n x_i^{\omega_i}$. Now for the attack by $\mathcal{U}'$, by the KEA assumption he can only succeed if he sets c_i as an algebraic combination of values in Table A.1 for which he *also* knows the x_0-th and w_{20}-th power. Concluding, c_1, c_2 can only be of the form:

$$\begin{aligned} c_1 &= c_{21}^{\alpha_1}c_{22}^{\alpha_2}(g^{\alpha_3}f^{\alpha_4}h_2^{\alpha_5}, g^{\alpha_6}f^{\alpha_7}h_2^{\alpha_8}), \\ c_2 &= c_{21}^{\beta_1}c_{22}^{\beta_2}(g^{\beta_3}f^{\beta_4}h_2^{\beta_5}, g^{\beta_6}f^{\beta_7}h_2^{\beta_8}), \end{aligned}$$

for arbitrarily chosen $\alpha_1, \ldots, \alpha_8, \beta_1, \ldots, \beta_8$.

Analysis of possible choices. The two encryptions c_1, c_2 in particular need to satisfy $h = D(c_1^{y_1}c_2^{y_2})h_0$ of (4), where $h = h_2 = g_1^{x_2^*+\phi_1}g_2^{x_{22}}h_0$. This is equivalent to stating that c_1, c_2 need to satisfy

$$g_1^{x_2^*+\phi_1}g_2^{x_{22}} = D(c_1^{y_1})D(c_2^{y_2}). \tag{29}$$

First we notice that by the definition of the ElGamal decryption function,

$$\begin{aligned} D(c_1) &= (g^{x_2^*+\phi_1})^{\alpha_1}(g^{x_{22}})^{\alpha_2}g^{\alpha_6}f^{\alpha_7}h_2^{\alpha_8}(g^{-\lambda})^{\alpha_3}(f^{-\lambda})^{\alpha_4}(h_2^{-\lambda})^{\alpha_5}, \\ D(c_2) &= (g^{x_2^*+\phi_1})^{\beta_1}(g^{x_{22}})^{\beta_2}g^{\beta_6}f^{\beta_7}h_2^{\beta_8}(g^{-\lambda})^{\beta_3}(f^{-\lambda})^{\beta_4}(h_2^{-\lambda})^{\beta_5}. \end{aligned}$$

Using the separation proposition repeatedly on equality (29), we obtain:

$$\begin{aligned}
\text{for } x_0 \in_R \mathbb{Z}_q &\Longrightarrow 1 = (h_0^{y_1})^{\alpha_8}(h_0^{-\lambda y_1})^{\alpha_5}(h_0^{y_2})^{\beta_8}(h_0^{-\lambda y_2})^{\beta_5} \\
&\Longrightarrow \alpha_5 = \alpha_8 = \beta_5 = \beta_8 = 0, \\
\text{for } \phi_1 \in_R \mathbb{Z}_q &\Longrightarrow g^{y_1\phi_1} = (g^{y_1\phi_1})^{\alpha_1}(g^{y_2\phi_1})^{\beta_1} \\
&\Longrightarrow \alpha_1 = 1, \beta_1 = 0, \\
\text{for } x_{22} \in_R \mathbb{Z}_q &\Longrightarrow g^{y_2 x_{22}} = (g^{y_1 x_{22}})^{\alpha_2}(g^{y_2 x_{22}})^{\beta_2} \\
&\Longrightarrow \alpha_2 = 0, \beta_2 = 1.
\end{aligned}$$

This leaves for (29):

$$1 = (g^{y_1})^{\alpha_6}(g^{y_1\lambda})^{\alpha_7-\alpha_3}(g^{y_1\lambda^2})^{-\alpha_4}(g^{y_2})^{\beta_6}(g^{y_2\lambda})^{\beta_7-\beta_3}(g^{y_2\lambda^2})^{-\beta_4},$$

which after separating for $y_1 \in_R \mathbb{Z}_q$ results in:

$$1 = (g^{y_1})^{\alpha_6}(g^{y_1\lambda})^{\alpha_7-\alpha_3}(g^{y_1\lambda^2})^{-\alpha_4}, \qquad 1 = (g^{y_2})^{\beta_6}(g^{y_2\lambda})^{\beta_7-\beta_3}(g^{y_2\lambda^2})^{-\beta_4}.$$

On these two equations Remark A.1.4 applies as $\mathcal{U}'$ does not know g^{λ^2}. Hence we obtain that $c_i = c_{2i}(g,f)^{\beta_i}$ $(i = 1,2)$ for known $(\beta_i)_{i=1}^2$ is the only possible choice for $\mathcal{U}'$, and this contradicts to $\mathcal{U}'$'s original goal which was to find a certificate such that (6.4) is not satisfied for any of the issuance executions.

A.2.10 *Concluding result*

For $K = 2$ and $l = 2$, we considered the possible attack strategy of a malicious forger $\mathcal{U}'$. Specifically, we analyzed the possibilities of the forger in an attack where he tries to combine the values he knows (Table A.1) into a possible certificate. It turned out, however, that the only way in which $\mathcal{U}'$ can have a non-negligible probability of success is by setting $h = h_2$, where $h_2 = g_1^{x_{21}} g_2^{x_{22}} h_0$ is coming from the second issuance execution in Figure A.1. Even stronger: the forger needs to set his certified encryptions c_i as $c_i = c_{2i}(g,f)^{\beta_i}$ for β_i known (for $i = 1,2$). This contradicts $\mathcal{U}'$'s original goal: constructing a certificate satisfying (4) such that (6.4) does not hold for any j.